The User Organisation: Structure and Governance in an Open Source Project

Jose Christian

A thesis submitted in partial fulfilment of the requirements of the University of Brighton for the degree of Doctor of Philosophy.

August 2015

CENTRIM - Centre for Research in Innovation Management.
University of Brighton

Abstract

User communities are a valuable source of innovation, where user-specific ideas and prototypes are developed and shared freely. Firms have begun to acknowledge the extent of their value at each stage of the innovation process, often resulting in commercially successful products. The current understanding of user communities, however, relies heavily on the study of software developers working on established open source projects. In recent years, with the emergence of web-based information and communication technology, open source projects and hacker communities have become much more complex, supporting a wider range of contributions from a diverse group of actors. While these technological advances have led to changes in the collaborative innovation, they have not been reflected in the user innovation literature.

Using the CyanogenMod project as a case study, this study explores how a user community manages multiple forms of contributions for the creation of multiple products and services. This study finds that the multiple forms of contributions create a number of communities of practice within the project, each with its own governance mechanism and leadership structure. In addition, the findings show that the activities of the different communities within the project are managed centrally by a separate leadership team who are responsible for setting and implementing long-term plans for the direction of the project as a whole. In light of these findings, this study proposes that the governance and social structures observed in the CyanogenMod project resemble a User Organisation rather than a user community, where the control over the stages of innovation can give users control over the direction and outcomes of the project.

I declare that the research contained in this thesis, unless otherwise formally indicated within the text, is the original work of the author. The thesis has not been previously submitted to this or any other university for a degree, and does not incorporate any material already submitted for a degree.

Signature .....

August, 2015

Contents

List of Tables

List of Figures

Glossary

AOSP Android Open Source Project. (1) The consortium of companies that are directly linked to the maintenance of the Android operating system. (2) The non proprietary version of Android which does not contain Google apps and services.

BSD Berkley Software Distribution.

FLOSS Free/Libre Open Source Software.

Forking To create a copy of a software's code and establish a new branch of development.

FSF Free Software Foundation.

Gerrit A web based software for submitting and reviewing software changes.

Git A software distributed revision control system developed by Linus Torvalds for non-linear open source software development.

GitHub A website which hosts repositories using the Git system.

GNU GNU in Not Unix.

GUI Graphical user interface.

ICT Information and Communication Technology.

IRC Internet Relay Chat. Instant messaging technology.

OSS Open Source Software.

PDA Personal Digital Assistant..

PDP Programmed Data Processors.

Porting Modifying the software to make it work in a different setting from what it was originally intended.

ROM Read Only Memory. Used to describe previous PDA and mobile phone operating systems that were directly installed into the hardware and was non-editable.

UI User Interface.

UX User Experience.

Preface

My love for computers and programing began in 1986, with an asteroid shooting game for the Apple IIe computer. The game involved travelling through a seemingly endless tunnel shooting asteroids as you escaped from the pursuing aliens. The game would increasingly get faster and more difficult, making it almost impossible for a seven year old to keep up, let alone finish it. Having read the computer manual, and after learning a few commands, I decided to type"`list`" into the console; the game's entire source code appeared in front of me, written in the surprisingly easy to read BASIC language. After spending an entire day reading the code, finding out what each line did, and how it all came together, I added a simple "`GO TO`" command and edited one line of text. I was then able to complete the game in less that two minutes, with the maximum score possible. The overwhelming sense of achievement I felt that day has remained with me until this day.

Almost twenty years later, I was able to recapture that same feeling when I decided to install Linux on an old laptop, although this time I was not alone. Anybody who attempted a Linux installation back in the early to mid 2000s will attest to the difficulties in making things work. I spent a month on forums and mailing lists trying to find solutions to all my problems; why is my wireless card not working? Why is the sound mono rather than stereo? It was during this time that I realised Linux, and Open Source, was more than just software — it was a band of brothers. Every day we joined forces and went to war against error messages, kernel panics, connection problems, and hardware incompatibilities. Each battle we won (or ignored) brought us closer together, as a community, to the software, to the project, and to the ideal of software freedom.

With the release of the Android operating system in September 2008, a new

battle ground was established — smartphones and mobile devices. After purchasing a Nexus One in early 2010, I went straight to forums, facebook groups, and mailing lists in order to arm myself with enough information to fully take control of my own smartphone. This is when I was first introduced to the CyanogenMod project.

While I had been familiar to open source communities for many years, the CyanogenMod project presented what I consider to be a new and more sophisticated type of online community. From the very start, this project had its own infrastructure; a dedicated forum, a wiki, IRC channel, and repositories, providing a wide range of services for its users. This was far more organised than any other community I had seen before. In addition, the project offered a range of different softwares that allowed non-technical users to modify and take control of their smartphones. It was this level of ownership the project had over all its related products and services that sparked my curiosity.

This study represents a period in the life of the CyanogenMod project, an account of the individuals involved, the things they do, and how they do them. It is an academic account of a User Organisation, a new type of governance and social structure in a user-led project, building on previous literature on user community governance and structure. What this study is not able to capture, however, is the sense of camaraderie, exploration, and achievement that arises from the interactions that take place within these communities. Open Source communities are not just a valuable external source of innovation and resources for producer firms, they are the location where individuals embark on a journey of learning and develop a belief in the importance of software freedom.

Acknowledgement

I would like to express my gratitude towards my supervisors, Steve Flowers and Nick Marshall, for their enormous support, guidance, and enthusiasm while patiently guiding me through the PhD experience. I would also like to thank faculty members and fellow PhD candidates at CENTRIM and SPRU for inspiring and encouraging me from the start. It has been an incredible experience being surrounded by some of the most talented and dedicated academics I have ever had the honour to meet.

To Carolina, thank you for always being there no matter what – I would have given up years ago had it not been for you. To my parents and my sisters, words will never be able to express how grateful I am for the unconditional love and support I have always received from you. Finally, to Alan, Sabina, and the kids, your friendship has been like a breath of fresh air and a reminder of how nice life can be.

Part I

Background of the Study

Chapter 1

Introduction

1.1 Introduction

The user innovation literature refers to open source projects as the archetypal representation of collective user innovation (von Hippel, 2005), where community processes can be observed and studied in great detail (von Hippel, 2001). Through the study of open source projects, the user innovation literature has been able to determine the key governance factors that allow a group of volunteers to coordinate their activities in order to design and develop a product or service. These findings, however, have been based on the assumption that open source projects are composed of a single community of practice working on a single output. This assumption is being challenged by a new generation of open source projects that incorporate a wide range of non-programming contributions from their users, allowing them to provide various outputs in the form of both products and services. This study therefore aims to reinterpret the notion of governance in the context of this new generation of open source projects.

Communities have emerged as a fundamental part of user innovation, and play an important role in the development, diffusion, and success of an innovation. User communities[1] are important in the development of new

[1]In this study, the term *user* describes any individual that uses the product. This term can further be divided into *user-contributors* (who actively contribute to a project) and *user-consumers* (who use the product but do not contribute to its development). The rest of this study will focus primarily on user-contributors who actively participate in the development or discussion of the project's products and services. At the same time, however, as with Cox (1998), Kuk (2006) and Schneider et al. (2013), it sees user-consumers as potential user-contributors who can become key to the innovation process and project success.

innovation by providing and facilitating the exchange of information and knowledge, to both users and firms, for the initial stages of product development (von Hippel and von Krogh, 2006). Equally, user communities can play an important role during the designing and building of these new products by testing and providing feedback, allowing further improvements and modifications (Franke and Shah, 2003). Finally, user communities can be central to diffusion of innovation, as members are themselves potential consumers who can also assist others understand and use these innovations (Harrison and Waluszewski, 2008). Despite the community's importance in the user innovation process, the understanding of the internal processes that allow these communities to functions has had to rely on other bodies of literature, potentially limiting the scope of the studies.

The major processes that take place inside these user communities have been less explored in the context of user innovation, relying instead on the study of open source communities from the perspective of Information Systems or Organisation Theory. For instance, the field of Information Systems has added to the understanding on collaborative development, focusing on the processes that help in the coordination and delegation of programming activities (Niederman et al., 2006). The field of Organisation Theory has also contributed in the understanding of community processes and structure by focusing on the relationships between individual volunteers and its effect on work assignment (e.g. Demil and Lecocq, 2006). By relying on these fields of study, however, the user innovation literature adopts a similar approach to project membership by focusing on communities of practice.

Both sets of literature, either explicitly or implicitly, view open source projects as communities of practice, where membership to the project is based on contributions related to software development. Dominant open source social structure models, such as those presented by Yamauchi et al. (2000) and Crowston and Howison (2005), for instance, imply that only users who engage in software development activities are project members. This emphasis has led to the established division of labour within open source communities to be focused on roles that are related to or that specifically deal with the

development of software, such as bug reporting, bug fixing, or user-to-user assistance. Similarly, the current understanding of governance has been shaped by studies on how groups of programmers interact with each other, coordinate their activities, and motivate other developers to contribute to the project.

The study of user communities follows a very similar approach by focusing on practice as the determinant of membership, often emphasising the communal learning that takes place through the transfer of practice-specific knowledge. For instance, studies that look at the development of extreme sports equipment (e.g. Lüthje et al., 2002) focus on groups of individuals that share the same interest. The emphasis on practice as a community allows studies to focus on the creation and exchange of knowledge between users, one of the key themes in the study of user innovation. While knowledge within user communities has been explored in the study of sticky information, the mechanisms that make key collective processes possible have been less explored.

One of the key mechanisms that facilitates collective innovation in communities is that of governance, a theme explored byt the user innovation literature in the study of hybrid communities. The literature on hybrid communities focuses on the relationship between firms and communities and how this can have an effect on the way individuals interact with one another. Studies on hybrid communities also deal with issues of ownership and control over innovations, proposing ways through which firms can benefit from groups of users. Once again, these studies have largely looked at open source projects and the firms that are involved with them, emphasising the benefits of user communities in the development of software. As a result, the understanding of how user communities are governed from the user innovation perspective is currently limited and fragmented.

Although being a key factor in collective innovation, governance has a wide range of interpretations in the literature (Markus, 2007), with one of its themes being the coordination of activities. Studies on open source communities have focused on understanding the key mechanisms and tools that are in place in order to manage software development, while at the same time, looking at the key individuals involved in the process and the roles that they carry out (e.g.

Crowston et al., 2005). The dominant view of open source projects is that authority is mainly gained through centrality, which is in turn measured by the weight of participation of each individual. This means that the individuals who contribute the most, through either programming or discussions, are key to the process and in general hold a key role in the software project. In addition, open source communities have been found to be composed of subgroups, each of which carry out a particular role within the software development process, such as bug fixing, bug reporting, and testing (e.g. Nakakoji et al., 2002). It is then the interaction and the codependency between these subgroups that allows the open source projects to produce high quality software.

Although the ability for users to contribute to an open source project through a wide range of activities has recently been facilitated by the development of web technologies, this has not been reflected in the literature on Open Source or user communities. During recent years, open source projects have developed and used new tools to help with the overall production of software, which has considerably changed the way users engage with the project. Established projects such as Linux and Apache made use of the web technologies that were available in the early to mid 1990s, which were basic repositories and email-based mailing lists. These two dominant technologies have formed the basis of studies on open source projects, focusing on the group of individuals that either contribute to the repository or who are active in the mailing list. Since then, however, there have been a number different tools that have improved and changed collaborative projects, such as dedicated forums with a wider range of discussions, dedicated wikis that provide different types of information, and new software versioning systems that facilitate the process of contributing to the code base. The emergent use of these tools, however, has not been fully reflected in the studies on user innovation or on open source.

The use of a wide range of technologies and tools has therefore seen the rise of a new generation of open source projects, which engage with users in a much broader way and goes beyond the core practice of software development. One such project is CyanogenMod, a modified version of the Android operating system, which is maintained and distributed by users. The project itself began in 2008

with the main aim of improving the software's stability, security, and efficiency. With the growth in popularity and use of the CyanogenMod operating system, the project owner, Steve Kondik, along with other core contributors, set up a dedicated web space for the project, which included its own website, repository, forum, internet relay chat (IRC) channel, wiki, and bug tracker. At the time of writing, the project actively seeks contributions from users in a large range of ways, such as translating the software to other languages, documentation, user assistance, and promotion.

While many existing studies on open source communities are based on the assumption that all contributions are centered around software development, the CyanogenMod shows a clear example of projects seeking multiple forms of contributions, therefore adding more complexity in terms of project governance. Building on the communities of practice approach, the different forms of contributions could potentially form boundaries within open source projects based on the practice which each individual carries out. These boundaries can emerge as the result of the different technologies, expertise, and skills required for each type of contributions; such as programming skills for software development, language skills of translating, and artistic skills for graphic designs. While the current understanding of governance is based on a single community composed of a single practice, these findings may not be able to explain how a project with multiple communities and multiple practices can be managed.

As stated previously, the dominant understanding of governance is based on mechanisms that are directly related to software development, and therefore this raises the possibility that the same mechanisms may not be applicable to other forms of contributions. For instance, the use of software licenses has been studied as a key aspect of community governance, being able to have a direct effect on participation and contributions (see Franck and Jungwirth, 2002). Other studies focus on the self-assignment of programming activities within a project (see Crowston et al., 2005). Equally, the current understanding on community joining, specialisation and the contribution of gifts from new members, are focused on programming activities (see von Krogh et al., 2003).

As a result, questions still remain as to whether these governance mechanisms are applicable for non-programming contributions since the tools and skills required may be different.

This study aims to fill this gap in understanding by analysing governance in the CyanogenMod project, taking into account all of the activities that add value to the project's various products and services. This study uses three main data sources, documentation, participation metrics, and semi structure interviews. The project's extensive documentation outlines the rules and processes that shape both contributions and interactions. In addition, using web scraping techniques[2], 585,375 contributions by 74,666 project members were recorder and analysed to determine participation patterns. Finally, interviews with key project members were used both at the start of the study and at the end to inform data collection and to help its interpretation.

1.2 Thesis Outline

The thesis begins with a historical perspective on the hacker movement that gave rise to - and helped shape - open source software development. While there have been a number of attempts at providing a history of hacker communities and open source development, most of the existing accounts do not discuss the importance of the technological environment. Chapter 2 therefore recounts the changes in computer technology and the effect it had on the development of one of the most complex software products, the operating system. It divides the evolution of the hacker movement and open source development into three generation that are exemplified by the Unix operating system, the Linux kernel, and Android. The first generation looks at the emergence of hacker communities working on minicomputers at AT&T's Bell labs and Berkley. This first generation was crucial in the development of the Unix operating system and is credited with being the cradle of the open source movement. The second generation of hackers emerged in the early 1990s as a result of the rise of personal computer and the internet, giving way to seminal projects such as the Linux kernel. This second generation has

[2]Techniques focused on the mass extraction of information from web pages.

been the subject of many studies from where we get our current understanding of how open source development works. Finally, the third generation of hackers has emerged in recent years with the mass adoption of smartphones, tablets, 3D printers, and SBCs (single board computers) such as the Raspberry Pi, along with advances in internet-based ICTs. The CyanogenMod project, a community modified version of Android, exemplifies the current generation, presenting a prime example of a new form of collaborative open source project where users can make a wide range of contributions to its multiple products and services.

Chapter 3 presents a review of the literature on user innovation and open source communities. The literature review shows that, while community governance can have a significant effect on innovation in user communities, the findings are largely limited to groups of developers, therefore presenting only a partial picture. Chapter 3 begins by reviewing the user innovation literature, finding that the focus has traditionally been on the individual lead user, the difficulties in transferring information, and the use of toolkits to facilitate innovation. When viewing communities, the user innovation literature often sees them as a source of information for lead users and firms, relying on other fields of study for the understanding of community governance. The other fields of study, however, focus on governance mechanisms that apply directly to software developers, basing their studies on second generation projects like Linux, where software development was seen as the primary form of contribution. In light of the new generation of open source projects, the chapter ends by highlighting the main gaps in the user innovation literature, which are the narrow view of community membership and the focus of governance on programming activities.

Chapter 4 presents the research design for this study. It begins by outlining the guiding aim of this thesis: To extend the current knowledge on project governance by understanding how a project coordinates multiple forms of contributions. This guiding aim is then broken down into four supporting questions: (1) how users contribute to the project; (2) the communities that are formed around the different practices; (3) governance mechanisms at the community level; and (4) governance mechanisms at the project level. A single

case study approach is used to address these questions, with the emphasis being on the collection of in-depth data from a revelatory case. The CyanogenMod open source project was chosen due to its prominence as the most popular project of its kind and because of the accessibility to data.

Chapter 5 presents the data collection and analysis, it outlines the techniques that were used to collect the data, the size of the data sets, and the names and details of the interviewees. The study used three sources of data, participation metrics, project documentation, and semi-structured interviews with project leaders. Participation metrics were used to determine the type of contributions made to the project, the timing of the contributions, who the key contributors are, and the structure of the communities. Project documents were used to determine the rules of contributions, the history of the project, and the official roles of the project leaders. Finally, the semi-structured interviews were used to further clarify issues of leadership, the history of the project, objectives, role formation and delegation.

Chapter 6 contains an overview of the Android Open Source Project and a brief history of CyanogenMod. The chapters discusses the aims of the Android operating system, taking a technical perspective on its characteristics in order to highlight its attractiveness for hacking communities. In addition, it discusses some of the issues that hackers have been attempting to address, particularly those of unnecessary manufacturer modifications and inadequate security. Finally, it presents an overview of the start of the CyanogenMod project in the XDA forum, looking at its relationship with the hacking community in general and its long-term objectives.

Chapter 7 looks at the different products and services being offered by the CyanogenMod project. While the project's main output is a modified version of the Android operating system, the project also provides a wide range of supporting softwares and services to help users. For instance, it provides software that simplifies the installation process, as well as how-to tutorials on technical and non-technical subjects. These products and services are significant in that they present a wider range of opportunities for users to contribute to the project.

Chapter 8 looks at the various ways in which users contribute to the project, finding that users make a wide range of contributions such as translation, graphic design, and documentation. The chapter highlights the different types of contributions that are made to the project through the repository, the wiki, and the forum. In addition, it finds that the number of users that contribute through software development is significantly smaller than those that make other types of contributions. Finally, it finds that each type of contribution created its own community of practice, each with a separate location for related discussions.

Chapter 9 presents the results on governance at both the community and project level. At the community level, the results show that each practice had its own governance mechanisms depending on the tools being used, their importance to the core product, and the availability of resources and skills. In addition, it finds that, at the project level, the project leaders were able to set long-term goals and implement these through a number of mechanisms that included outsourcing solutions from other open source projects.

Chapter 10 presents the interpretation of the results. This chapter focuses on the implications the different types of contributions has on membership, as well as the effect governance has the process of collective innovation. It addresses the issue of membership to an open source project and its effect on what adds value to the project, arguing that the various forms of contributions add to positive network externalities that can themselves lead to project success. It also addresses the issue of governance at the level of the project, finding that the governance mechanism in the project were important in allowing project leaders shape the direction of the project by retaining control over the scope and timing of contributions. Finally, it is through these finding that this study proposes the User Organisation as a new style of governance in collective innovation which accommodates multiple forms of contributions while at the same time allowing project owners to retain control over the innovation process.

Chapter 11 presents the conclusion of this thesis, providing a summary of the findings, limitations of the study, and future research. The limitations of

the study are based on the single-case study approach, which raises some issues in terms of generalisability. In addition, it discusses some of the challenges of measuring participation, particularly in the context of the open source environment. Finally, the chapter contains items for future research, further determining the full extent of how firms can benefit from user communities, the possible application of new forms of hybrid models, and the effect social structure has on collective innovation.

Chapter 2

History of Open Source Development

2.1 Introduction

Hacker[1] communities have been central to the emergence and evolution of open source software. Changes in the technological environment of these communities has led to major changes in the way complex software is collaboratively developed. For instance, advances in microprocessors and the subsequent rise of the minicomputer during the 1960s led to the widespread development and adoption of the Unix operating system. Equally, an increase in accessibility to the internet in the 1990s led to the rise in popularity of the Linux operating system, allowing users from all over the world to use and contribute to the project. In recent years we have once again seen a change in the nature of computing and collaborative development, where the emergence of internet-based information and communication technology (ICT) tools and the rise of mobile devices, has led to yet another change in the way complex software is written.

While there have been a number of attempts at providing a history of hacker communities and open source development, most of the existing accounts do not discuss the importance of the technological environment. For instance, key

[1] In this study, the definition of *hacker* is taken from Eric Raymond's *The Jargon File Version 4.4.8* which describes it as "A person who enjoys exploring the details of programmable systems and stretching their capabilities."

literature, such as the one presented by Raymond's (2001) *"The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary"*, emphasises the ideological aspect of the hacker movement, looking at the freedom of using and modifying software to suit individual needs. Equally, in other historical accounts, such as Levy's (2001) *"Hackers: Heroes of the Computer Revolution"*, the emphasis is on the origins of the activity of software modification and the intellectual challenges and motivations that led to it. A significant number of the existing historical accounts therefore focus on the ideological and motivational aspects of the hacker movement, without placing much emphasis on the changes in the technological environment that made this possible. This chapter discusses the symbiotic relationship between advances in hardware and ICT and hacker communities in the development of one of the most important components of modern computing, the operating system.

This chapter is structured as follows. Section 2.2 presents an overview of the modern operating system, providing a view on its different components and the complexity in their integration. Section 2.3 then introduces the first generation of open source development, where advances in minicomputers led to the emergence of the Unix operating system. Section 2.4 goes on to discuss the second generation, where the mass adoption of the internet had a key role in the development, diffusion, and popularity of Linux. Section 2.5 looks at the third and current generation, where the maturity of internet based development tools and the rise of mobile devices have given rise to Android-based community projects. Finally, Section 2.6 expands on the latest generation of software projects and hacker communities working on technologies that include 3D printing, drones, and decentralised internet networks.

2.2 Operating Systems

Prior to discussing hacker communities, it is important to understand the complexities of the modern operating system and how this drives collaborative software development. A modern computer is made of hardware and software, where the software itself allows the user to control the hardware through the

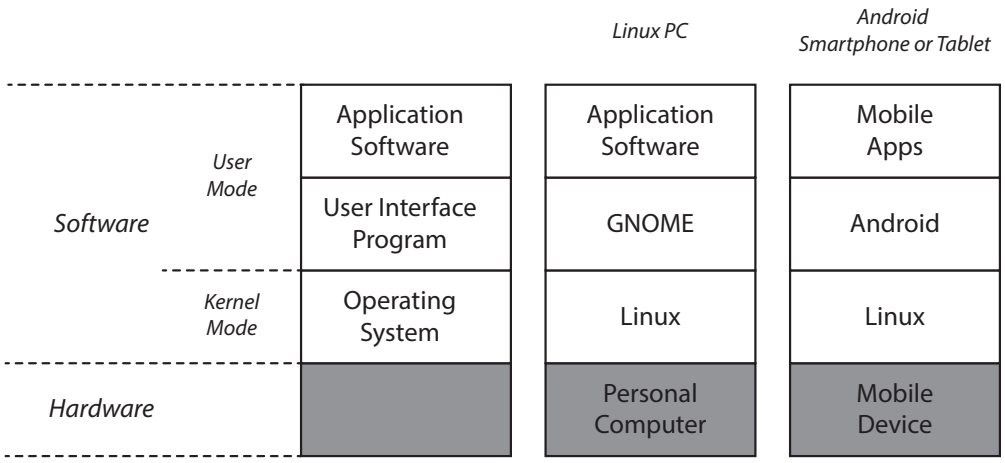


Figure 2.1: The Operating System in context. Adapted from Tanenbaum and Bos (2014).

input, process, and output of data. Figure 2.1 shows where the operating system lies within the context of the personal computer and the more recent mobile devices. The hardware itself is manipulated by the software, which in turn is divided into two parts, the kernel and the user-mode. The user-mode is composed of both the user interface and the application software. The user interface program is the desktop environment which the user interacts with in order to manage files and application software, in a Linux system these are referred to as the desktop environment, with the most popular ones being GNOME, KDE, or Xfce. The application software layer consists of different applications that add specific functions for the user, such as a word processor, a web browser, or a spreadsheet.

In an operating system, both components of the user-mode are not as essential as the kernel, since it is the kernel that operates the hardware and is responsible for all the processes. It is the kernel that is responsible for handling the input and output of data to the machine, communicating to peripheral hardware, and executing instructions (processes). Among its modern functions, it is also responsible for managing multiple processes through the allocation of memory and error handling (Tanenbaum and Bos, 2014). The user-mode layer, on the other hand, has the primary function of providing a better human-machine interaction for users to access kernel functions.

Tanenbaum and Bos (2014, p 5) describe the complexity and size of operating systems when compared to application software.

> Operating systems differ from user (i.e. application) programs in ways other than where they reside. In particular, they are huge, complex, and long-lived. The source code of an operating system like Linux or Windows is on the order of five million lines of code. To conceive of what this means, think of printing out five million lines in book form, with 50 lines per page and 1,000 pages per volume. It would take 100 volumes to list an operating system of this size – essentially an entire book case... And this is only for the part that runs in the kernel.

The complexity and importance of the operating system have been key factors in promoting collaborative software development, where a large number of individuals with different skills are required for its development. It is through the development of operating systems like Unix and Linux operating systems, where we most see the emergence and evolution of collective open source software development. In addition, advances in computer technology and ICT have also played an important role in the evolution of how collaborative and distributed software development takes place. The next section will discuss the rise of affordable computers and the emergence of distributed development of an operating system, Unix.

2.3 Affordable Computing and Unix

Up until the early to mid 1960s, the computer industry was dominated by large mainframe computers, with IBM commanding over 70% of the market (DeLamarter, 1986). These computers were predominantly used for batch processing of information, and were built specifically for the type of data being handled (either scientific or commercial). In addition, these computers were typically modified and maintained by an in-house team of engineers, who would write software to meet specific data processing needs within their organisation. Because these computers were expensive and specialised on a particular function, they were unfavourable for software development as their usage would typically be restricted and highly controlled. At MIT (Massachusetts Institute of Technology), for instance, the IBM 704 mainframe computer would be

constantly serviced by a number of specialist engineers, and those that were allowed to submit data for processing were not allowed direct access to the computer itself (Levy, 2001). This would change with the introduction of smaller more affordable computers that would allow users to experiment with software development.

The minicomputers did not have the same power or processing speed as the larger mainframes, but at 6% of the cost, their main advantage was that these new types of computers made software development more accessible, and users were able to carry out a larger range of functions. One of the effects of the introduction of the minicomputer was the emergence of a software industry, where dedicated firms were now able to write and distribute software that would then be able to run on a wide install base (Steinmueller, 1996). The main company that dominated the computer market was DEC (Digital Equipment Corporation), which released a range of computers called the PDP (Programmed Data Processors) in the early 1960s. The release of the PDP range signaled the start of exploratory software development, where users could now justify using the relatively inexpensive equipment for their personal projects. One such project was the Unix operating system, which Ken Thompson wrote over the space of a weekend on a PDP minicomputer.

The use of these microcomputers in MIT, Xerox's Palo Alto Research Centre, and AT&T's Bell Labs, were to have a significant effect on the hacker movement and open source development for years to come. DEC donated one of the first PDP-1 minicomputers to MIT, where computer enthusiasts were able to use it to develop innovative "hacks"; playing video games, adding functionality, and improving the performance of the computer (Levy, 2001). Similarly, while establishing the Palo Alto Research centre (which would become the source of the modern computer with the development of the computer mouse, graphical user interface, and the laser printer) all researchers agreed that their preferred computer would be the PDP-10 (Hiltzik et al., 1999). Most significantly, however, was the development of the Unix operating system by Ken Thompson in Bell Labs, using a PDP-11, which would not only introduce a unified operating system, but would also change the way software

was developed.

Ritchie and Thompson (1978) explain that one of the main achievements of the Unix operating system was that it demonstrated that "...a powerful operating system for interactive use need not be expensive either in equipment or in human effort." (p365). Ritchie and Thompson supported their argument by stating that the operating system ran on a PDP-11, which had limited memory and processing power when compared to mainframe computers. At its launch, the Unix operating system grabbed significant attention, particularly from Professor Bob Fabry, from the University of Berkley, who later managed to obtain a copy. Soon after, a number of Berkley students began working on the Unix source code, modifying, improving, and adding new features to the software, and then distributing these changes to others. It was Berkley's modified Unix (known as Berkley Software Distribution, or BSD) that benefited the most from distributed development.

The significance of the BSD Unix was that it was the first time that user feedback and modifications were directly incorporated into the software. It was Bill Joy, a Berkley student who would later become one of the co-founders of Sun Microsystems, who was responsible for compiling new versions, answering queries, and incorporating user feedback into the software (DiBona and Ockman, 1999). In this early form of distributed software development changes to the software were sent by post or suggested over the telephone, evaluated and, if they were of a good enough quality, they would be included in the software (Leonard, 2000).

Although the official copies of Unix were not free, in that a license fee had to be paid to AT&T for its use, users were still able to view and modify the source code. In other words, AT&T would charge commercial users a fee in order to obtain a copy of the operating system, once obtained, users were free to modify the source code. The main reason was that the software was distributed "as is", in non-compiled source code (as opposed to a working binary file), and with very little support from AT&T[2]. In order to respect this software license, BSD had a policy of distributing their branch of Unix to users who already had a licence agreement with AT&T, although this was not always checked or

[2]Due to a court order, AT&T were not allowed to enter the software industry because of problems with monopoly.

enforced (DiBona and Ockman, 1999). Therefore, although Unix was subject to a licence agreement, users were legally free to modify the software and share those modifications with others.

The Unix operating system became a commercial closed-source software in the early 1980s with the breakup of AT&T Co. The Unix commercialisation gave rise to a number of companies, such as Sun Microsystems, IBM, SCO (Santa Crux Operations), and SGI (Silicon Graphics Inc.) all of which released their own commercial version. As a direct result of this, Richard Stallman, an MIT researcher, decided to create the FSF (Free Software Foundation) and start the GNU (GNU is Not Unix) project, with the aim of creating a *free* (as in freedom) open source operating system that would replace Unix. While the Unix operating system presented one of the earliest cases of distributed open source software development, it was Stallman's FSF which would give a voice and a direction to free and open source software as a movement.

Despite a number of innovations in software[3], the GNU project was not able to achieve the creation of a full operating system, as a number of attempts stalled and delayed the release of the software. The creation of a fully non-proprietary open source operating system would only become a reality with the release of the Linux kernel by Linus Torvalds, a university student from Finland. The Linux project through the 1990s would also be the start of a second generation of hackers and open source software development, where the internet and microcomputers (now known as Personal Computers or PCs) would play a prominent role.

2.4 The Internet and Linux

In 1991, Linus Torvalds began working on a small operating system, with the aim of developing a free Unix-like system that was inexpensive and gave users a wider range of freedom. Ultimately, Linus was trying to create a free version of the Minix[4] operating system, which at that time had restrictions in its use and modification. On the 25th of August 1991, Linus went to the Minix online user

[3]Specifically the GCC (GNU Compiler Collection), which is still one of the most widely used C compilers for both Unix and Linux.

[4]Stands for Mini-Unix. A smaller version of Unix originally released by Andrew S. Tanenbaum in 1987 to be used for educational purposes only.

group and posted the following message.

> Hello everybody out there using minix
>
> I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).
>
> I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)
>
> Linus (torvalds@kruuna.helsinki.fi)
>
> PS. Yes it's free of any minix code, and it has a multi-threaded fs. It is NOT protable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(.

Due to the wide increase in internet usage, the Linux project was able to make use of the emerging ICTs to gain a large user-base. The project made use of FTP (File Transfer Protocol) technology to distribute the software, as well as Usenet groups and mailing lists that allowed contributors to communicate and coordinate activities. Although the GNU project also used FTP back in the mid 1980s, it was not as successful in its use due to the relatively low internet usage at the time (Bretthauer, 2002). The usage of early internet-based ICT is, according to Raymond (2001), typical of this generation of open source projects, a generation that includes other well-studied projects such as Apache, Sendmail, and Mozilla.

Although Linux itself was originally not seen as being a commercial threat, it defied all expectations to become one of the most used and widely distributed operating systems. Large organisations, from IBM to NASA, have used Linux in a wide range of machines, from super computers to robots, and it has even flown on the space shuttle (Torvalds, 1999). The mass adoption and commercial success

of Linux also led to the creation of new sectors within the software industry, those that did not rely on the ownership of intellectual property.

The rise and adoption of Linux also gave way to a new form of business model in the software industry, where the supporting products and services became primary revenue sources. During the 1980's, the major revenue stream of the software industry was the direct sale of pre-packaged software (Steinmueller, 1996). With the rise in popularity of Linux and open source, new companies were emerging that were taking advantage of the operating system and were developing new business models around the sale of supporting services (Dahlander and Magnusson, 2008). One of the most prominent and successful companies was RedHat, which would distribute their version of the Linux operating system for free, but would charge users for supporting services.

It is due to the commercial success, the development process, and the resulting effect on the software industry, that Linux has become one of the most widely studied cases of open source development to date. Crowston et al. (2012), for instance, note that 58.8% of open source studies have looked at Linux. Lee and Cole (2000) explain the project's popularity by stating that the "Linux project has demonstrated the feasibility of a large-scale on-line collaboration effort where developers and users can be one and the same" (p 2).

While Linux became successful in its own right, it still owed some of its success, particularly its adoption among established businesses, to a concerted effort by free software proponents and hacker communities to promote open source development as a viable alternative to commercial software. Raymond (2001) notes that the decision by Netscape Communications Corporation to make their internet browser open source, in a bid to defeat Microsoft's Internet Explorer, led to the creation of a PR strategy with the aim of legitimising the use of open source software. This effort included in its core the direct promotion of Linux, as Raymond (2001) states "Promoting Linux must be our main thrust... If Linux can't consolidate the breakthrough, nothing else will, pragmatically speaking, have a prayer."(p 213).

The flexibility and openness of the Linux operating system has made it possible to be used in a large number of different devices. As Torvalds (1999)

explains "[Linux] is used in embedded systems; it is used to control robotic devices; it has flown on the space shuttle. I'd like to say that I knew this would happen, that it's all part of the plan for world domination. But honestly this has all taken me by surprise." (p38). A recent addition to this list is the smartphone operating system, Android.

The versatility of Linux allowed it to be used in smaller hand-held devices, and has now turned into an integral component of the Android operating system. Android was first developed by Android Inc in 2003, with the aim of producing an operating system that could compete with the commercial counterparts for the emerging sector of hand-held mobile devices. With the commercial success of mobile devices, such as personal digital assitants and smart phones, came the emergence of a new type of hacking community which not only displayed a wider range of technical skills, but also had a more developed understanding of the ICTs that made collaborative development easier. The hardware-software integration and the recent advances in internet-based ICTs have once again changed the technological environment, simultaneously affecting how open source projects work.

2.5 Mobile Devices and Android

Google bought the smartphone operating system Android in 2005. In 2007, it joined forces with device manufacturers, carriers, and software companies to form the OHA (Open Handset Alliance) , to which it handed Android's development while still retaining the lead role. The aim of the OHA was to promote Android as an open smartphone platform in an effort to decrease the industry's reliance on proprietary platforms. To actively maintain and distribute Android, as well as to encourage participation from the public, Google established the AOSP (Android Open Source Project). It is through the AOSP that Google distributes the official and unmodified operating system – commonly referred to as the AOSP version.

Android is the most popular smartphone operating system, commanding over 50% of market share (Pettey, 2011). The operating system draws from two popular open source projects, Linux and Java, allowing it to be customised to

suit the needs of Google and its commercial partners. The open source nature of most of its technology has also allowed users to modify it and distribute it within user communities. Its flexibility and highly customisable nature has allowed it to work on other devices, such as games consoles, smart TVs, and tablets.

The need for an open source platform emerged with the increased popularity of mobile devices in the early to mid 2000s, which saw the merge in functionality between PDAs (Personal Digital Assistants) and mobile phones. The PDAs can be tranced back to the 1980s, with the second generation of PSION organisers which were the first hand held devices with an operating system that could digitally store diary entries and contact details. Further changes in technology improved their functions, where the Pilot 1000, developed by Palm Inc and released in 1996, for the first time incorporated a touch user interface which required a stylus pen, moving away from the traditional mechanical keyboard of previous PDA generations.

The merge between the PDAs and the mobile phones occurred in the early 2000s, with the introduction of key software and hardware that was able to incorporate and integrate both technologies. As the technology of mobile phones continued to improve, other features were added to the mobile phone, primarily data transfer that would make internet browsing and email exchange possible. This would then lead to advances in the operating systems in order to use the advances in hardware, where proprietary operating systems such as PalmOS by Palm Inc, Windows Mobile by Microsoft, and Blackberry OS by Research In Motion[5] added many of the PDA functions to mobile phones for the first time. Among some of the key devices of this first generation of these devices, called smartphones (also referred to as mobile PDA phones or pocket PCs), were the Blackberry 6200 series and the O2 XDA devices running Windows Mobile.

In noting that there was a "...tremendous potential in developing smarter mobile devices ..." (Beavis, 2008), Andy Rubin co-founded Android Inc in 2003, and began developing the Android operating system. Keeping with the original intention of creating a non-proprietary operating system, Google has justified the use of open source technology, particularly Linux, with the following statement.

[5]Now BlackBerry Ltd.

> Android is intentionally and explicitly an open-source – as opposed to a free software – effort; a group of organizations with shared needs has pooled resources to collaborate on a single implementation of a shared product. The Android philosophy is pragmatic, first and foremost. The objective is a shared product that each contributor can tailor and customize.

It is therefore the open source software license that has allowed manufacturers and carriers to modify the operating system in order to adapt it to a wide range of devices and uses. While Android was originally intended to be used in smartphones, it is now estimated that there are approximately over 18,000 different types of devices using Android (OpenSignal, 2014), including tablets, entertainment systems, game consoles, and car media consoles. Android's open source license has not just made it possible for a large number of firms to modify it for their needs and profit from its use, but it has also allowed groups of enthusiasts to modify it for their own personal use.

In addition to its commercial success and adoption, Android has also received much interest from hacker communities, who have created projects around the development and redistribution of user-modified versions. Hacker communities specialising on hand held mobile devices began appearing in the early 2000s, working primarily on the modification and redistribution of Windows Mobile operating system for the XDA range of devices. These new hacker communities led to the creation of key forums, most importantly that of the XDA-Forum, that would later move on to the development of Android after its release in 2008. Since then, the XDA-Forum, and other Android hacking communities, have created projects around the development and redistribution of user-modified Android operating system, which they call ROMs[6].

[6]The Windows Mobile operating system was a Read Only Memory (ROM) mode, embedded into the hardware and difficult to modify. Since the XDA community began hacking Windows Mobile ROMs, the term was also applied to the Android operating system, even though it is not stored in ROM mode.

2.6 Third Generation Hackers

The third generation of hacker communities now revolves around smaller devices that make use of specialised software, such as smartphones, 3D printers, drones, and SBCs (Single Board Computers) like the Raspberry Pi. Unlike the first generation of hackers they are no longer restricted to expensive equipment in R&D labs, but are now found in fablabs, maker-spaces, and hacker-spaces, exchanging ideas and learning from each other. Equally, unlike the second generation of hackers, they are no longer confined to mailing lists or users groups, they now enjoy a much wider range of more sophisticated ICT that allows them to share video tutorials and wiki guides to disseminate their ideas and innovations.

The projects that have emerged from this third generation place equal importance to hardware and software. The release of the Raspberry Pi, and open source hardware, with a collection of compatible open source software development tools, has spawned a large community of hackers who have used the device to create a wide range of projects, from a home media centres to video consoles[7]. In addition, websites such as thingiverse.com allows individuals to exchange .stl[8] files for 3D printing, with other communities such as Reddit's r/3dprinting community offering technical support. Finally, the Meshnet Project, aiming to create a decentralised internet, not just typifies the hardware and software integration of this generation, but also stands true to the traditional ethos of the hacker movement, which Levy (2001) describes as follows:

> Hackers believe that essential lessons can be learnt about the systems – about the world – from taking things apart, seeing how they work, and using this knowledge to create something new and even more interesting things. They resent any person, physical barrier, or law that tries to keep them from doing this. (p28)

This resentment towards authority remains to this day, with the Android

[7] A small but significant list of project can be found at http://www.raspberrypi.org/forums/

[8] STL are standard files used for storing Computer Aided Designs for rapid prototyping through 3D printing.

modification community being one of the prime examples. With recent questions about the distribution of user data obtained through mobile devices, both for commercial and governmental reasons, users have now began to tackle the issue of privacy. Projects like CyanogenMod have recently aimed at addressing privacy issues by modifying Android in order to add greater security measures. These modifications include the removal of manufacturer and carrier software and services (including Google) that could make use of user information.

While the first generation of hackers created many of the hardware and software technologies that we have today, it is the second generation from which we have learnt many of the lessons on open source. The current understanding of how the open source communities work and how they coordinate comes from studying key projects that emerged during the 1990s, particularly Linux and Apache. Advances in technology have once again changed the landscape and the area where hackers are now active, therefore having an effect on how these groups work and coordinate their activities. The next chapter provides a review of literature on the study of open source project, predominantly from a User Innovation perspective.

Chapter 3

Literature Review

3.1 Introduction

As mentioned in the previous chapter, the nature of hacker communities and open source projects has changed in recent years. With the availability of new internet-based ICT, and with the change in the nature of computer hardware, user communities have evolved from the early 1990s, when projects like Linux and Apache started. Despite this change, however, the literature on user innovation and open source communities has not yet captured this change, particularly the different ways in which users are now engaging projects. This chapter, reviews the literature on user innovation and open source communities, noting that their study merges three main bodies of literature and therefore share many similarities in approach and in the assumptions that are made.

Following from recent changes in the way that users engage in the innovation and production of a projects, the primary aim of this thesis is to understand how a user innovation community manages multiple contributions from its members. The literature review draws from three areas of study; User Innovation, Organisation Theory, and Information Systems. Table 3.1 is an overview of the literature, the key themes[1], and representative studies. The three areas of study are complimentary to each other whereby each focuses on different aspects of user innovation communities and share many of its

[1]Adapted from Martinez-Torres and Diaz-Fernandez (2014), who classified the bodies of literature based on the journals the relevant studies were published in. The themes covered are not mutually exclusive but instead show a dominance of topics by each body of literature.

approaches to membership and structure.

This chapter argues that the User Innovation literature relies on the field of Organisation Theory and Information Systems for the understanding of community structure and governance. The fields of Organisation Theory and Information Systems deal with open source communities by focusing on the programmer, since, as Crowston and Howison (2005) point out "... without the production of code there would be nothing to study" (p4). While Open Source projects present an ideal empirical setting for the study of user communities (von Hippel, 2001), the focus on programmers has limited the scope of user involvement in these projects. The approach of focusing on developers only may be key to understanding how the internet has changed the processes of software development, but may not go far enough in capturing the recent changes and maturity of related technologies. Broadening the scope of study by incorporating multiple forms of user contributions to an open source project will therefore add to our current understanding of user innovation communities.

This chapter is structured as follows, Section 3.2 places User Innovation within the larger field of Innovation Studies, contrasting it with both Integrated and Open Innovation. Section 3.3 presents three dominant themes in user innovation studies – Lead Users, Sticky Information, and Toolkits – showing that the current literature has predominantly focused on individuals, knowledge, and commercialisation. Section 3.4 discusses user innovation communities in greater depth, finding that the reliance of the User Innovation literature on programming activities has narrowed down the definition and boundaries of the users involved in the innovation process. Section 3.5 then reviews the literature on open source community governance, looking at both its definition and purpose, concluding that the scope of governance focuses solely on one social group. Finally, Section 3.7 presents a summary of the three key gaps in the literature and sets the main objective of this thesis.

Body of Literature	Themes / Concepts	Detail	Literature
User Innovation	Free Revealing	The free transfer of information and innovation within the context of the user community.	von Hippel and von Krogh (2006); Morrison et al. (2000)
	Hybrid Communities	The fabrication of a user community by a producer firm.	Shah (2006); von Hippel and von Krogh (2006)
Organisation Theory	Communities of Practice	Defines the boundaries and purpose of the user community. Based on the practice of software development.	Zhao and Bishop (2011); Lattemann and Stieglitz (2005); Xu et al. (2005)
	Core-Periphery	The structure of the community based on the patterns of participation or interaction of users. Also used for development coordination	Crowston and Howison (2005, 2006); Crowston and Scozzi (2002)
Information Systems	Governance	Wide range of ides from the perspective of coordination. Focusing on how open source communities are able to coordinate activities.	Kilamo et al. (2012); Koch (2009)

Table 3.1: Bodies of Literature used in this study.

3.2 User Innovation

The objective of this section is to provide an overview of the user innovation literature, including its origin, the role of users, and the areas where it has been observed. This section notes that user innovation is seen as a part of open innovation, where the source of ideas and key stages in the innovation process are external to the firm, leading to user innovation being viewed from the perspective of the producer firms. In addition, studies have shown that users are involved and have key roles at various stages of the innovation process, most importantly during the stages of idea formulation and diffusion. Finally, the section presents an overview of the wide range of areas and examples where user innovation has been observed, from open source development to farming in developing countries.

This section looks at a broad overview of the User Innovation Literature. Subsection 3.2.1 discusses User Innovation in relation to Integrated and Open innovation. Subsection 3.2.2 discusses the role of users in the innovation process. Finally, Subsection 3.2.3 presents a review of the empirical studies on User Innovation.

3.2.1 User Innovation in Innovation studies

Users have been found to be a valuable source of innovation, providing new ideas and support for innovating processes. The study of user innovation is seen as part of the open innovation paradigm which sees value in sources of innovation that are external to the firm. The primary focus of innovations coming from users is on the actual use of the innovation rather than its commercialisation, making it therefore different from traditional in-house and main stream open innovation paradigms. Users, who can be both individuals as well as firms, have been the source of a number of innovations in products, processes, and services in a wide range of industries and sectors. This section presents an account of user innovation and its relation to innovation studies.

User innovation focuses on knowledge and innovations that originate from the use of a product or service, with the main benefit being its use rather than economic appropriation. The emphasis of user innovation, therefore, is on the study of innovations that occur outside of the boundaries of the firm and where its main benefit is its use. User innovation has been observed in a wide range of industries and sectors, where it has led to the improvement or emergence of new products and services through processes of knowledge sharing and collaboration.

User innovation can be seen as part of the Open Innovation paradigm, which looks at how firms can benefit from innovation activities outside of its boundaries. As Chesbrough (2006) states, open innovation can be contrasted to the established Schumpeterian paradigm which saw the rise of corporate Research and Development (R&D) laboratories, a system which was further supported by intellectual property rights. This model of innovation is what Bogers and West (2012) refer to as vertical integration, where key innovation activities such as development, manufacturing, and distribution of new products occur with the resources and within the boundaries of a producer firm.

Bogers and West (2012) highlight the differences between traditional producer, open, and user models of innovation by looking at the relationship with other innovators, the spillovers, the stake holders, and the locus of innovation (Table 3.2). The main difference between the three models of innovation are based broadly on the locus of innovation, where in vertical

Attribute	Vertical Integration	Open Innovation	User Innovation
Question	How do firms control end-to-end innovation process?	How can firms maximise innovation effectiveness?	How can users be supported to become innovators?
Stakeholders	Firms.	Firms and other firms in value network.	Users and Producers.
Success Measure	Profit.	Profit.	Quantity of significant innovations.
Locus of Innovation / Knowledge	Within the firm	Outside the firm.	Within users.
Type of Innovator	Organisational	Organisational	Individuals and user firms.
Spillovers	Blocked	Paid	Free.

Table 3.2: Models of Innovation. Adapted from Bogers and West (2012).

integration it is within firm boundaries, in open innovation it is outside the firm boundaries, and in user innovation it is within the user-base. In terms of the relationship with other innovators, Bogers and West state that in producer or traditional innovation there is no relationship, while in open innovation the relationship is that of exchange, and in user innovation it is co-operative. In terns of spillovers, in producer innovation spillovers are stopped, in open innovation they are controlled, and in user innovation they are free. Finally, the uniqueness of user innovation then comes into play in terms of the stake holders, where in both producer and open innovation this is the firm, in user innovation it is the user.

The benefits of user innovation include increased commercial success, quicker time to market, reduced cost of research and development, and a method of diffusion (von Hippel, 2005). More so, innovations that incorporate user knowledge in their design or implementation are also more likely to have more follow on innovation (Chatterji and Fabrizio, 2012). Products that can be customised by users have also been found to be of more value from the perspective of the consumer, meaning that users may develop a higher willingness to pay when compared to standard products (Franke et al., 2009).

Users involved in innovative activities can be individual consumers, enthusiasts, professionals or firms, and are called user innovators when the main benefit of their innovation is its use. Innovation by individual users has been studied in a wide range of industries, predominantly the extreme of outdoors

sports equipment (Shah, 2000; Franke and Shah, 2003; Lüthje, 2004; Schreier et al., 2007) as well as watches (Franke and Piller, 2004), jewelry (Fuller, 2006), household products (Nishikawa et al., 2013), and fashion (Di Maria and Finotto, 2008). Typically, during the innovation process, individual users socialise with other users to create user communities. A significant portion of the studies on user innovation communities has focused on two main industries, open source and software development (von Hippel, 2001; von Hippel and von Krogh, 2003; West and Lakhani, 2008), and extreme sports equipment (Von Hippel, 2001; Franke and Shah, 2003; Schreier et al., 2007).

Users that innovate can be classified as intermediate or consumer innovators depending on the nature of the innovation (Bogers et al., 2010). The intermediate users can be described as firms that use and innovate technologies in order to produce other products, while the consumer users are those that are end users or end consumers. Intermediate user innovation has been studied in a wide range of high tech industries, such as semi conductors (Adams et al., 2012), computer aided design (von Hippel, 1986), medical devices (Chatterji and Fabrizio, 2012), and manufacturing machinery (Bai et al., 2006). Adams et al. (2012), for instance, focus on user firms to determine the volume of innovation in high technology firms, finding that user firms were a major source of innovations. De Jong and von Hippel (2009) also look at user firms in the context of high-tech firms in the Netherlands, noting that approximately 25% firms went on to innovate software and manufacturing processes for their own use. Such an example is the computer aided design and open source software, to which both individual consumer users and firms and organisations are also responsible for innovating.

Besides product and service innovation, users can also innovate by changing or adapting the way products are used (Faulkner and Runde, 2009). For instance, Andersen (2012) looked at the use of mobile phones in the health system in Norway finding that users innovated by adapting their use of mobile devices in order to better engage with health professionals. Faulkner and Runde (2009) made a similar observation, where users changed the use of the traditional record players in order to create different sounds and a new form of

music style called scratching[2]. In these studies, therefore, the innovation occurred not in the development or a product or process but the use of an existing products in order to solve a problem.

Although user innovation has a wide range of benefits in a number of industries, it can be affected by issues of access to knowledge, legal implications, or commercial influence. Braun and Herstatt (2006), looking at the context of seed companies, state that there are four main barriers that can affect user innovation, which are legal, market, technological, and social. The legal barriers make reference to laws that do not allow the modification or the customisation of products or services, such laws cover copyrights and patents, placing restrictions on what users can do to the product. Here Fisher III (2010) observes that in recent years there has been an increase of producers taking user innovators to court for copyright or intellectual property reasons with mixed results, highlighting the need for user innovation policy. The market barrier refers to the bargaining power of the producers, who are able to force users into contractual obligations that restrict them or forbid them from innovating, making them rely on producer innovation only. The technological barrier refers to the complexity of innovating, which may refer to the fact that users may not have the technological skills or resources to innovate. Finally, social barriers focus on the stigma of innovating, with an example being the social view of certain practices such as hacking, which may not be seen favourably or may have negative connotations. Braun and Herstatt therefore argue that these four barriers are becoming more prominent as the awareness of user innovation increases.

3.2.2 User Roles in the Innovation Process

Users can be involved in innovative activities at various stages of the innovation process. Chu and Chan (2009) develop a framework to classify the stages within the innovation process at which users can be involved in, which also indicates the roles that they carry out. The first stage of the process is where problems are

[2]Faulkner and Runde (2009) describe scratching as manipulating a record on a turn table to create "... rhythmic scratching sound that could be juxtaposed against and used to complement music playing on the second turntable" (p 17).

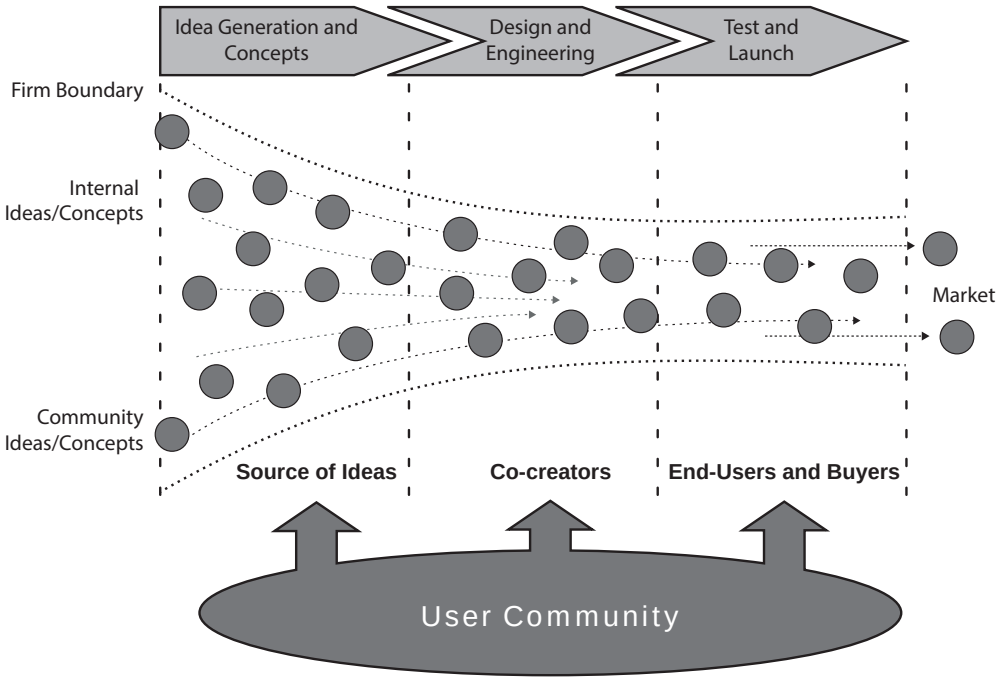


Figure 3.1: Innovation Patterns. Taken from Chu and Chan (2009).

found and ideas are gathered, this is where the individuals find a problem and then seek to satisfy it, with this stage being key to the Lead User theory. The second stage sees the user exchange ideas with others either through a network or a community, with processes such as free revealing, testing and feedback playing an important role. Finally, the third stage looks at the way in which users innovate once a product has been adopted, through customisation or related innovations.

The first stage, of finding a problem and a possible solution, is one of the most developed ideas within the user innovation literature, where the lead user theory is a key example of this initial process. The lead user theory (explained in more detail in Subsection 3.3.1) notes that some users can come up with ideas that are ahead of the trend of normal users (von Hippel, 1986, 1989). Firms can benefit from lead users by assessing their ideas and using them as a marketing tool for new product development (von Hippel, 1990b; Herstatt and von Hippel, 1992). Besides innovative ideas, however, users can also be the source of innovation themselves, whereby users can prototype ideas which can then have the potential for commercialisation and wide distribution (von Hippel, 2001; Baldwin et al., 2006). An example of this can be seen in Durugbo and Pawar (2014) who look at

the semiconductor industry and note that users have two roles, either as problem solvers or as co-designers through the exchange of ideas within producer firms. At this first stage of the innovation process, therefore, users can innovate through problem finding and solving, and can create two kinds of output through ideas or prototypes.

In the second stage of the innovation process, users carry out further development and testing, with many of the activities taking place within the context of a network or community of like-minded users. This stage looks at users as being able to provide feedback and testing for some of the innovations that have been made by other users. A key example of this is the debugging and the testing that takes place in open source communities. Numerous studies have shown that user innovators are willing to share their innovation, allowing others to test, improve, and fix the code (Lakhani and von Hippel, 2003). Studies have found that users can and do benefit from freely revealing and sharing their innovations by facilitating the distribution and mass adoption of the innovations as well as benefiting from the large pool of users who can provide feedback or improvements. Franke and Shah (2003), for instance, find that user innovators' willingness to freely reveal their innovation to others in the community has a positive effect on the assistance they receive from the community. On the second stage of the innovation process, therefore, users can innovate existing products and services through improvement and minor changes.

In the third and final stage of the innovation process, users can be involved and also facilitate the adoption and diffusion of innovations. At this stage of the innovation process, users can play an important part in the success of a product through word-of-mouth referral to other users (e.g. Harrison and Waluszewski, 2008) and through the creation of network externalities (Parker and Van Alstyne, 2010). Lead users, for instance, can help in the adoption of new technology by being the champions of new technology and making other users aware of its benefits and novel uses (Douthwaite et al., 2001), which can also work in the way of re-launching previously failed innovations (Harrison and Waluszewski, 2008). At the same time, user communities can in some cases compete with commercial distribution channels (von Hippel, 2001).

	User	Manufacturer	Supplier	Other
Scientific Instruments	77%	23%	0%	0%
Semiconductor and Printed Circuit Board Process	67	21	0	12
Pultrusion Process	90	10	0	0
Tractor shovel-related	6	94	0	0
Engineering plastics	10	90	0	0
Plastics additives	8	92	0	0
Industrial gas-using	42	17	33	8
Thermoplastics-using	43	14	36	7
Wire Termination equipment	11	33	56	0

Table 3.3: Functional Sources of Innovation, taken from von Hippel (1988)

3.2.3 Empirical Studies

User innovation has been studied and observed in a wide range of settings and industries, each of which has helped study different aspects of the phenomena. For instance, early empirical studies on medical and scientific instruments helped develop and support the lead user theory and the lead user methodology. Equally, user innovation has also been observed in groups of individuals working on open source software and extreme sports equipment which has helped understand key aspects of the collective innovation process. The phenomena of user innovation has therefore not been restricted to small areas or particular industries, as they can be observed in a wide range of areas from specialised products to consumer goods. As von Hippel (1988) notes, innovation occurs in a wide range of industries and sectors, albeit in different degrees.

Early user innovation studies looked at very specific high-tech industries and were important in the early development of the lead user theory and the lead user method. For instance, von Hippel (1976) studied innovation in the scientific equipment, specifically Nuclear Magnetic Resonance Spectrometers and Ultraviolet Spectrophotometers, finding that 80% of the useful innovations originated from users themselves, who he terms *lead users*. In addition, Urban and von Hippel (1988) use innovations in computer aided designs (CAD) for printed circuit boards in order to formulate and test a method for identifying lead users. Finally, von Hippel (1998) looks at Application-Specific Integrated Circuits (ASICs) and Computer Telephony Integration (CTI) systems to

observe that users typically customise these systems with the aid of toolkits.

Another area of interest has been in the field of extreme sports equipment, where observations and findings have added to the discussion of sticky information and the user entrepreneur. Some of the examples of user innovation in extreme sports include studies on windsurfing (von Hippel, 2001), mountain biking (Lüthje et al., 2005), kite surfing (Franke et al., 2006; Schreier et al., 2007; Schreier and Pruegl, 2008), and rodeo kayaking (Baldwin et al., 2006; Hienerth, 2006; Hyysalo, 2009; Hienerth et al., 2014). These studies have been important in gaining a better understanding on knowledge sharing and commercialisation. Franke and Shah (2003), for instance, look at a range of extreme sports communities and how their members share ideas and prototype innovations. It was this study which was one of the first to focus on the user innovator in the context of a community, and how the community itself could provide assistance to user innovators. Franke et al. (2006) also use the kite surfing community in order to test the basic principles of the lead user theory for potential commercially successful innovations, finding that both the expected benefits and the 'ahead of trend' characteristics can predict commercial attractiveness. Finally, Hienerth (2006) looks at user innovation in the kayak industry, finding that users are able to manufacture and commercialise their innovations with low-cost manufacturing techniques, and were able to do so before producers.

Software development, and particularly open source software, is another area that has been widely studied, where the focus becomes the study of the user community and the roles of users in collaborative innovation. von Hippel (2005) states that open source projects present a prime example of user innovation, where projects are typically initiated by a lead user satisfying their personal need. The most widely used open source case studies have been the Linux and Apache project, which have allowed researchers to explore a wide range of topics such as brand creation (Füller et al., 2013), mundane tasks (Lakhani and von Hippel, 2003), toolkits (Franke and von Hippel, 2003), user networks (von Hippel, 2007), and co-creation (von Hippel and von Krogh, 2003). Some studies in user innovation, however, have also looked at

commercial software development, taking examples from commercial software and the communities of users who innovate them. A few examples of user innovation in commercial software include the modification of video games (Prügl and Schreier, 2006; Hau and Kim, 2011) and the creation of plugins (Jeppesen and Frederiksen, 2006; Dahlander and Frederiksen, 2011). These studies have largely focused on issues of knowledge exchange between community members, and between the community and firms.

Studies on fashion items have been used to study a range of user innovation aspects such as success, adoption, and value perception. For instance, Di Maria and Finotto (2008) look at the fashion industry, noting that user involvement in the design stages of development can help increase consumer's perception of the products value, leading to a higher willingness to pay. Similarly, Franke et al. (2010) look at fashion items such as scarves and T-shirts, noting that users' involvement in their design leads to a feeling of accomplishment which also increases willingness to buy. Nishikawa et al. (2013) then studied Muji, in the design of house-hold items, finding that user-designed products were commercially more successful than producer-designed products. Finally, Fredberg and Piller (2011) studied the sportswear company Adidas, finding that user involvement did not just improve innovations, but was also helped in the adoption of those innovations.

The findings and the theories in user innovation have also been applied and studied in the health care industry, helping to find new forms of engaging with patients and providing more personalised healthcare. Teixeira and Suomi (2010), for instance, look at incorporating users into the design stage of a system for disease control, arguing that users can provide feedback in the form of innovative ideas. Andersen (2012) looks at user innovation in the use of mobile phone services in order to engage with health services in rural Norway. Finally, De Couvreur and Goossens (2011), look at rehabilitation engineering design[3], finding that user innovation is an effective and low cost method in customising prosthetics.

Finally, the user innovation literature has also been explored in the context

[3]The design of prosthetics and orthotics for disabled patients.

of economic development, looking particularly at agriculture and the adoption of new technology. Aoki (2009) takes the user innovation perspective, particularly the free sharing of information and innovations, to propose the user, or the farmers, themselves could be the source of new agricultural plant varieties in order to offer an alternative to commercial seeds. Equally, Douthwaite et al. (2001) suggest incorporating farmers from developing countries in the design and testing of new agricultural equipment, with the aim of increasing the new technology's success. Finally, Braun and Herstatt (2006) make the observation that user innovation of agricultural seeds in China may be limited as a result of commercial pressures and lack of knowledge from the local population.

3.2.4 Section Summary

This section presented an overview of the user innovation literature, looking at where it fits within the broader field of innovation studies, the role users have in the innovation process, and the areas where user innovation have been studied. Within the field of innovation studies, user innovation forms part of open innovation. In contrast to the traditional Integrated model of innovation, where all innovation activities occur within the boundaries of the firm, Open Innovation argues that firms can increase the effectiveness of innovation by incorporating external stakeholders in the process (Chesbrough, 2006). While in Open Innovation the external stake holders are typically firms that form part of the value chain, in User Innovation the key stake holders are either individuals or organisations that use the product, process, or service (Bogers and West, 2012).

Users can have a number of roles within the innovation process, where they can be the source of new ideas, part of the co-creation process, or can help diffuse new innovation. The type of innovation within each of the stages also differs, along with the individuals and the mechanisms that are in place. For instance, during the first stage of the innovation process, the literature looks primarily at lead users and their motivations for innovating. In later stages of the process, however, the focus becomes the user within the context of a social group, either

a community or a network, where the social relations become important.

Studies have shown that users innovate in a wide range of industries albeit in varying degrees. User innovation has been observed in highly technological industries, such as scientific instruments, medical engineering, Computer Aided Design, and software. It has also been studied in consumer goods, such as extreme sports equipment, such as rodeo kayaking, mountain biking, household goods, and watches.

3.3 Dominant Themes in User Innovation

This section presents three dominant themes in user innovation studies; lead users, sticky knowledge, and toolkits. These themes have been developed from the perspective of users as a source of novel ideas and innovation, therefore focusing on the first stage of the innovation process. The study of user innovation began with lead users, who are described as individuals that are more likely to be ahead of the curve and show new needs that will be common to all users in the future. For firms to make use of this information it is required that lead users are able to communicate with the producers. This led subsequent studies to discuss the difficulties of knowledge transfer, leading to the conceptualisation and further development of sticky information. Finally, the management of sticky information was further explored by the literature through the study of toolkits, which can allow firms to gather information in codified form which they understand, or by allowing users to meet their own needs through them.

This section looks at the literature on each of these themes, along with the different ways in which their study has evolved. Subsection 3.3.1 looks at the early studies of user innovation, which began with the lead-user theory, providing a definition and a method for identifying them. Subsection 3.3.2 moves on to knowledge management between users and the firm, the emergence of sticky knowledge, and its significance in knowledge sharing and dissemination of innovation. Finally, Subsection 3.3.3 goes on to discuss the literature on toolkits, how these present a way of overcoming difficulties in knowledge transfer and can facilitate user innovation.

3.3.1 Lead User

Early user innovation literature focused on the lead user as a market research method for identifying new product ideas in fast moving industries. von Hippel (1986) describes lead users as "...users whose present strong needs will become general in a marketplace months or years in the future." (p 791). For this study, von Hippel draws on market research literature, stating that lead users can be the source of novel ideas which firms can use in the early stages of product development. This initial study then goes on to identify four steps firms need to take in order to incorporate lead users into their innovation process, which are: (1) identifying potential trends, (2) identifying lead users in that trend who have both experience and need, (3) analysing lead user data, and (4) assessing its suitability in the general market (Urban and von Hippel, 1988; von Hippel, 1989). The focus of this early study is on the method by which firms can obtain lead user information and then act on it. This study is similar to a large body of early user innovation literature that focuses on identifying lead users (most notably Urban and von Hippel, 1988; von Hippel, 1990b; Herstatt and von Hippel, 1992).

The lead user method has since been applied to a number industries, leading to its further refinement and a better understanding of its benefits. The lead user method was first applied in Urban and von Hippel (1988), where the identification of the lead user firms took place through the use of questionnaires, where hardware and software innovations for computer aided designs (CAD) were mapped, finding that 28% of respondents exhibited lead user characteristics. The lead user method was applied once again by Herstatt and von Hippel (1992), who looked at industrial products and materials in construction, finding that new product concepts could be found faster and cheaper using this method. Another application of the lead user method was made by Franke et al. (2006), who tested the method in the context of kite surfing. In their study, Franke et al. found that the two key characteristics of the lead user method, the expected benefit and being ahead of the trend, corresponded to a higher likelihood of innovation success. Additionally, they also found that the likelihood of a user innovating a product increases if they are at the leading edge of a market place, rather than the likelihood depending

Step I: Start of the Lead User process.	Building an interdisciplinary team. Defining the target market. Defining the goals of the Lead User involvement.
Step II: Identification of Needs and Trends	Interviews with experts (market/technology). Scanning of literature, internet, databanks. Selection of most attractive trends.
Step III: Identification of Lead Users	Networking based search for Lead User. Investigation of analogous markets. Screening of first ideas and solutions generated by Lead Users
Step IV: Concept Design	Workshop with Lead Users to generate or to improve product concepts. Evaluation and documentation of the concepts.

Table 3.4: Process of the Lead User Method. From Lüthje and Herstatt (2004)

solely on the attractiveness of the innovation.

Lüthje and Herstatt (2004) extended the lead user method by looking at the processes that are involved in identifying lead users and evaluating their ideas (Table 3.4). This four step approach provides further information as to the activities that take place at each stage of the process, the actors involved, and some of the outcomes. It expands on von Hippel's original framework by using multiple sources of information for identifying needs and trends, and then includes lead users in a workshop during the concept design. Lüthje and Herstatt, therefore, introduce a more depth in-depth approach with the inclusion of lead users in the first stages of commercial development, something that was not explicit in the first model by von Hippel (1986).

While the early literature focused on the lead user as a source of ideas, studies later went on to address and acknowledge that users can also manufacture and prototype innovations, with examples coming from the software and extreme sports industries[4]. For instance, it is often argued that the individuals who start an open source project can be classified as lead users, who create software solutions that may be adopted by other users (Lüthje and Herstatt, 2004). Lüthje and Herstatt argue that Linus Torvalds, as a lead user, developed a working prototype of the Linux operating system, which was then further developed and improved by other users. Lead users as prototype manufacturers is once again supported by Shah (2000), who found that lead users were not just the source of

[4]Although as early as von Hippel (1976) it was acknowledge that users also prototype innovations, these early studies have predominantly focused mainly on ideas.

novel ideas or product modification in scientific instruments, but were also early manufacturers.

The lead user phenomenon has been observed and measured in a wide range of empirical settings, from scientific equipment to building materials. Lüthje and Herstatt (2004) point out that lead users have been studied in the development of equipment for outdoor sports, medical surgery, extreme sports, pipe hangers[5], and computer aided design. Slaughter (1993) looks at lead users in the building industry, finding that these users are more likely to solve user-based problems than manufacturers. Douthwaite et al. (2001) look at seeds farming in Asia, noting that the adoption in new industry technology is dependent on the users' ability to understand and replicate the technology, something that can be achieved by firms incorporating lead users in the design stage.

Alahuhta et al. (2008), however, note that there might be limits to the type of innovation that can come form the lead user method. In their study, Alahuhta et al. aimed to determine whether or not lead users genuinely were the source of innovative ideas. Their study looks at the assessment of lead user ideas for the innovation of mobile services, which were then classified in terms of how innovative these are. The results showed that the ideas presented by the lead users were limited in terms of novelty, where existing technology was expanded, but no form of radical ideas were found. This, they argued, may be down to the users' lack of imagination or foresight, as users may know possess the knowledge or expertise to understand the technology. Finally, it can also be argued that the study utilised workshops consisting of a wide range of individuals from all backgrounds and ages, with university students forming the largest groups. These individuals may therefore lack the expertise and knowledge that has been associated with lead users in previous studies.

With the emergence of social media, the lead user method was once again expanded in order to take into account new forms of web-based ICT. Hung et al. (2011), for instance, find that lead users can be identified in social media through their high level of activity and by the way in which they use relevant tools at

[5]Pipe hangers are typically used to secure pipes to ceiling.

their disposal. Eisenberg (2011) extended the lead user method to include online communities such as forums and discussion spaces, adding reverse snow-balling as a method for lead user identification. These studies also helped highlight some of the challenges of internet-based research, where identifying objects of study may be straightforward, but getting them to respond to emails and to agree to interviews may prove more difficult.

The introduction of new forms of web-based ICT has also led to the discussion about the challenges firms face when engaging with lead users. For instance, Eisenberg (2011) states that the lead user method must be supported internally by the firms conducting the search, as the increase in accessibility to a larger population may make it more difficult to sort and evaluate ideas. Both Alahuhta et al. (2008) and Eisenberg (2011) agree that it is important for firms to be open minded about the problems and solutions originating from users, adding that firms should spend more time processing relevant information before and after contacting lead users. These two studies, therefore, represent the last of the increases in terms of lead users as a method for determining future trends.

As a final addition to the lead user method, von Hippel et al. (2009) introduce the method of pyramiding, with the aim of helping search for lead users in rare or hidden areas. The main problem highlighted by von Hippel et al. was that it has become difficult to find key individuals in large and poorly mapped search spaces. As a result, they introduce the idea of pyramiding, which states that the people who are knowledgeable on one subject could be used as referrals to other people who are even more knowledgeable. Using this method of finding lead users, von Hippel et al. were able to identify most key individuals through a relatively small sample of 28.4%. The pyramiding method is similar in approach and function to snowball sampling, a method used in social science to study sensitive subjects by identifying them through referrals (Biernacki and Waldorf, 1981).

Lead users have also been studied in the context of a larger network of users or communities, where the focus is on lead users and their relationship with others in the community. Morrison et al. (2000), for instance, looked at the lead user within the network of individuals that were using the OPAC Library System software, finding that users freely reveal their innovation and then move them

on to other users in the network or community. What makes this study different from the others is the fact that it is no longer looking to integrate lead user knowledge into the firm, as with the previous studies, but instead focuses on the exchange of innovations from lead users to other users.

Similarly, a number of studies have looked at lead users in terms of diffusion of innovation, classifying them as opinion leaders and innovative users of technology. Rose et al. (2007), for instance, look at the outdoor equipment industry in the UK from a historical perspective, noting that innovation within the industry was guided predominantly through user-centric design by lead users. Rose et al. go on to state that the relationship between lead users and the firm has become a vital part of the industry's success. Hung et al. (2011) find that lead users typically use technology or toolkits in innovative ways, and are responsible for the diffusion and communication of innovations to others in the community. Harrison and Waluszewski (2008) provide an alternative view of the role of lead users in the diffusion of innovations in the biosensor industry. In their study, Harrison and Waluszewski find that lead users played an important role in the relaunch of unsuccessful products by innovating the product's use and educating other users. As a result, although there were no changes in product specification, the lead users were seen to help in the adoption of the innovation by changing the way it is used.

Finally, the lead user theory has also been used to explain more general problems concerning economic development. Scheraga et al. (2000) for instance, take the lead user perspective to address the problem of cultural distance in the dissemination of new technology to a developing country. Their study looked at the introduction of solar cookers in Haiti, finding that lead users could be used to adapt the product to accommodate local perceptions and help its adoption. In a similar study, Douthwaite et al. (2001) look at the improvements in agriculture in developing countries, stating that the role of the lead user among the local community of farmers is important for the adoption of new agricultural technology.

To summarise, one of the earliest themes in the user innovation literature has been the lead user, as both a theory and a method. The lead user theory has

focused on the location of ideas that can help in the development of innovations, while the lead user method provides an approach for identifying users with that information. The literature on lead users has covered a wide range of areas, including methodologies, techniques, and benefits, and has been revised in order to incorporate changes in technology, particularly new forms of web-based ICT. The lead user literature, it has to be noted, focuses on the individual and the knowledge they possess, crucially not taking into account collaborative product development.

3.3.2 Sticky Information

The concept of sticky information was developed by von Hippel (1990a), in order to highlight the difficulty of transferring information between users and producers, as well as between users themselves. von Hippel (1998) states that there are three main reasons why the information can be regarded as sticky, which are (1) information attributes, (2) attributes of the individuals seeking the information, and (3) the organisational structures of where the transfer takes place[6]. The first reason is down to the characteristics of the information itself and how it is encoded. The second reason is down to the characteristic of the individuals that seek this information, what are their capabilities in being able to make sense of the information and the skills to act on it. Finally, the third reason is down to the organisational structure of the location where the information is being transfered, such as a network or community of users, which includes the relative distance between the information giver and the information receiver. In addition, von Hippel (1994) argues that problem solving activities will move to the location where the relevant knowledge resides. If the required knowledge is spread across different locations, problem solving activities can move between these locations in an action known as 'iteration'. If the cost of iteration is too high, however, problem solving tasks can be split into smaller tasks that will be given to each location.

[6]Sticky information shares a number of similarities with Sticky Knowledge, where Szulanski (1996) states that the stickiness is based on the characteristics of the knowledge being transfered, the source of knowledge, the recipient of knowledge, and the context in which it takes place. Sticky information, however, focuses on sharing information for problem-solving while sticky knowledge focuses on sharing knowledge for the transfer of best practices.

Studies have since provided empirical evidence for the effect of sticky information and its location on innovation activities, both in the context of a firm and within user communities. Ogawa (1998) for instance, looks at 24 user innovations in Japanese convenience stores, finding evidence that the location of the information has an effect on innovation activities. Similarly, Lüthje et al. (2005) found that users in a mountain biking community are more likely to solve problems if they already possess the skills and knowledge to do so, therefore reducing the need to use information that is difficult to obtain. Lüthje et al. go on to argue that users will find both problems and solutions within their location, with the likelihood of innovating increasing as the stickiness of knowledge decreases.

Studies have suggested that sticky information should be viewed in terms in degrees of stickiness rather than being dichotomised, and that stickiness may be part of the environment in which the user is active. The information itself, however, can also be sticky in degrees, with An-Hua and Peng (2009) arguing that there are certain levels of stickiness rather than it being binary – either sticky or not, with some information being more sticky than others. In their study, An-Hua and Peng argue that producer firms should use lead users in order to minimise the level of stickiness. Scheraga et al. (2000), add to the stickiness of the information by highlighting the cultural aspect, whereby the difference in culture from those that have the information and those that require it, may also increase its stickiness.

Sánchez-González et al. (2009) further expand the classification by stating that there are two kinds of sticky information, information about user needs, and information about product technology. In their study, Sánchez-González et al. look at data from Spanish businesses, finding that that firms choose to gather need information from users through cooperation when that information is seen as sticky, as firms may be aware of the importance of user specific information. Equally they argue that, where user information is less sticky, other methods, such as market research, may prove less costly and equally successful.

The concept of sticky knowledge makes the general assumption that user information comes from lead users, who are knowledgeable about both problems

and solutions. Franke et al. (2009) note that the benefits of products made through user contributions or customisation may be more valuable to consumers, but it all depends on several characteristics, particularly the idea that users themselves know what their needs are. This is a point that was picked up by Heiskanen and Lovio (2010), who state that some users may have important ideas, but may also be first time users, therefore complicating their ability to transfer that information to producers. In their study, Heiskanen and Lovio looked at house energy innovation, concluding that some house builders were very particular abut their ideas being implemented by the contractors, but were nonetheless first time users. That is to say, although their ideas were important for the building of the house, these ideas could not be transfered completely to the producers because the users did not have previous experience in building houses.

In order to facilitate the transfer of sticky information, toolkits can be used to allow producers to re-codify information so that it can be transfered from user to producer with minimal loss. von Hippel and Katz (2002) observe that in some high technology industries user needs change constantly, and therefore information is required more often. In order to facilitate the flow of sticky information from users to producers, von Hippel and Katz propose the use of toolkits to help diminish or get rid of the need to transfer that information. Toolkits can empower the users to solve the problem, therefore moving the problem solving activity to the users rather than it having to be carried out by the producers. In addition, a study conducted by Helminen and Ainoa (2009) supports this by finding that user needs, in relation to shopping mall design, was easily conveyed through the use of toolkits, noting that the more flexibility a toolkit offers to the user, the better the quality of information being transfered.

To summarise, sticky information in the context of user innovation has been used to describe some of the challenges that are faced in the transfer of user-specific information. Unlike the early lead user literature, sticky information begins to take into account the social context in which these users operate, as it looks at the relationships between users and firms in the transfer of information.

The studies on sticky information, however, still focus on the individual user and their information, rather than on collaborative processes in user innovation.

3.3.3 Toolkits

In the user innovation literature, the primary function of toolkits is as an effective method of dealing with the issue of sticky information, which can be done in two ways. The first method consists of unsticking the information by giving users the ability to transfer ready-made solution by simplifying the characteristics of the information and the way it is codified. The second method consists of completely removing the need for the transfer of information and instead allowing the users to create their own solutions with the toolkits provided.

Toolkits play an important role in the transfer of information from the user to the producer, by re-codifying information and allowing it to be transfered at a lower cost. von Hippel (2001) notes that producer firms may face difficulties when meeting user needs as the required information may change frequently or may be too sticky to transfer. von Hippel therefore proposes that toolkits may allow manufacturers to "... abandon their attempts to understand user needs in detail in favour of transferring need-related aspects of product and service development to users along with an appropriate toolkit." (p247).

Two seminal studies on user innovation toolkits, von Hippel (2001) and von Hippel and Katz (2002), state that a toolkit should contain five key elements. First, it has to allow the user to carry out a cycle of trial and error within the design, therefore giving users the ability to test a number of solutions. The second element of a toolkit is the provision of an appropriate solution space, which is the limits or the scope of what is being produced. The third element is user-friendliness, which is based on engaging users without making them learn new skills, but instead to help them use the skills they already possess. The fourth element is the module library[7], which is a set of often-used elements that may help speed up user design. Finally, the fifth element is based on the notion of the translation between user design to production, which should be error-free and

[7]A module library can be defined as a collection of standard commonly used features that can be incorporated into new designs (von Hippel and Katz, 2002).

should lead to the production of what the user has designed.

Studies have found that the composition of toolkits, particularly that of the library and the space, can greatly affect the transfer of information from user to producer. Helminen and Ainoa (2009), for instance, find that minimising restrictions in both the library and the space of a tool kit can significantly improve the transfer of information. Dahan and Hauser (2002) then note that web technologies themselves can be adapted by firms in order to create toolkits that will allow users to tell the firm exactly what they need, throughout the entire life cycle of the project rather than just the design phase.

While the study of toolkits in user innovation initially emphasised their use during product development, other studies have explored their usefulness in customisation after product release. This form of toolkit is known as an embedded toolkit, which allow the user to change the functionality or characteristic of the product at the stage of consumption in order to meet their specific needs (Franke and Piller, 2003). This is different from the purpose of using the toolkits to allow users to design a product for producer firms to manufacturer, which is what von Hippel and Katz (2002) cite as being the fifth element of toolkits. Steiner et al. (2009) argue that embedded toolkits provide users with flexibility in terms of how products are used. This flexibility has the added advantage of allowing firms to mass produce a standard product while at the same time meet diverse user needs through customisation. Franke and von Hippel (2003) expand on embedded toolkits and mass customisation in open source software. Their study looks at the Apache Security software, which can be customised by their users, finding that a wide range of needs are met through the toolkits. In addition, they also find that the ability for users to customise the software makes the users happier with the product, as their needs are specifically met.

The value of toolkits on consumer perception was further explored by Franke and Piller (2004), who find that the customisation of watches through the use of toolkits increases the consumer's willingness to pay. Value is also added by what Franke et al. (2010) refer to as the "I designed it myself" effect, which they argue also generates a higher willingness to pay. Piller et al. (2010) and

Steiner et al. (2011) then go on to discuss the implications of embedded toolkits on user innovation and new product development in the automobile industry. Their findings support the higher willingness to pay from the user perspective, but they also introduce the element of enjoyment in customising or modifying the product.

Finally, while the study of toolkits in user innovation focuses on producer-made toolkits, Prügl and Schreier (2006) have found that users themselves create and modify their own toolkits. Prügl and Schreier look at toolkits used for modifying the Sim computer game, finding that lead users do not just use the official toolkits, but they also develop their own toolkits to cater for their more sophisticated needs. The modification takes place in terms of widening both the solution space and of the module library, and was achieved by adding a few files which would allow users to select a wider range of characters in the game.

The study of toolkits in user innovation focuses on facilitating the innovation process by providing tools that tackle the issues of information transfer between users and firms. Studies highlight two types of usage; first, it can be used to reduce the stickiness of user information so that it can be used at the design stage, and second, it can also be used at the consumption stage where users can customise the product to meet their needs. What has to be noted, however, is that much of the literature on toolkits still focuses on individual innovators, and does not fully take into account collaborative innovation and its processes.

3.3.4 Section Summary

This section presented the three major themes in the study of user innovation; lead users, sticky knowledge, and toolkits. The review shows that the dominant perspective of the user innovation literature is on the individual, their characteristics, the knowledge they possess, and their relationship to producer firms. Lead user theory is the most developed theory in user innovation, setting the foundations for the study of the phenomenon by addressing the conditions and the motivations for users to innovate a wide range of products. From early studies, the lead user theory and methodology focused on obtaining innovative ideas from users, leading to a higher potential of commercially successful (see

Urban and von Hippel, 1988; von Hippel, 1989, 1990b). With further studies, users were not just found to be a source of ideas, but they were also observed to produce prototypes and test a range of solutions. The theory has been extended and perfected several times in order to clarify the methodology and to incorporate changes in the environment (see Schreier and Pruegl, 2008), particularly the emergence of new web-based ICT (see Kaiser and Muller-Seitz, 2008). Finally, the lead user theory has also been used as a theoretical perspective to understand diffusion of innovation within the context of user networks or communities (Harrison and Waluszewski, 2008), as well as through the adoption of new technologies in developing countries (Scheraga et al., 2000).

Sticky information adds to the user innovation literature by acknowledging the difficulty of transferring information from lead users to producer firms (Lüthje et al., 2002). The stickiness of the information is broadly as a consequence of the way that the information is codified, the amount of information that needs to be transfered, tools that are used, and cultural context. In the user innovation literature, the most explored form of unsticking the information is through the use of toolkits (Franke and von Hippel, 2003).

Toolkits play a key role in user innovation by either facilitating the transfer of sticky information, or removing the need to transfer information from user to producer. The toolkits themselves can be used in two different ways, they can either be implemented by the firm at the design stage, or they can be embedded in the product to allow users to customise them. By using it in the design stage, toolkits help in the re-codification of user-based information so that it can be understood and acted on by producer firms. On other hand, by embedding toolkits into the product itself, the firm's need to understand and act on user-based information is reduced, as the toolkits allow the users to customise the product according to their own need.

As shown in this section, user innovation studies have focused on key users (lead users), the challenges or transferring their information (sticky information) and possible solutions to these challenges (toolkits). The focus on users, knowledge, and techniques are focused on the design stage and the adoption stage of the innovation process, while collective innovation has not

been fully explored. In the next section we will look at the study of user communities in the user innovation literature.

3.4 Communities in User Innovation

User innovation communities have largely been seen as the location where user knowledge resides, therefore being a valuable source of information that can aid users and firms in the innovation process. This view has guided the literature to focus on the free exchange of information through free revealing, and the relationship between firms and communities and the effect this may have on the creation and exchange of user information. This section is structured as follows. Subsection 3.4.1 notes that, in the user innovation literature, the value of the community has either explicitly or implicitly focused on it being a source of information for the innovation process. Subsection 3.4.2 goes on to discuss the relationship between user communities and firms, focusing on the need for firms to understand users and their needs in order to benefit from their knowledge. Finally, Subsection 3.4.3 discusses firm-hosted user communities, called hybrid communities, looking particularly at the impact the relationship between firms and communities has on participation and engagement, and therefore knowledge creation.

3.4.1 As a Source of Information

In the early user innovation literature, the key function of the user community was as a source of information for lead users[8]. In one of the early community studies, von Hippel (2001) describes three conditions that make user communities possible. The first being the need for users to have an incentive to innovate, which may be brought about by their own needs. The second condition is that users are willing to freely reveal their innovations or share their ideas with other like-minded individuals. Finally, the third condition is the ability of the community to provide a viable alternative to commercial production and distribution. It is

[8]In the context of user communities, Morrison et al. (2000) make the distinction between community members that innovate (lead users) and those that do not.

von Hippel's second condition of free revealing that has dominated the study of user innovation communities by focusing on their function as a source of user information.

It is the discussion of free revealing which guides much of the research on user communities, particularly their role as a repository of valuable user information. Taking from Harhoff et al. (2003), von Hippel and von Krogh (2006) state that free revealing occurs when "... all intellectual property rights to that information are voluntarily given up by that innovator and all parties are given equal access to it – the information becomes a public good." (p 295). The benefits of free revealing focus on the access to information by user innovators to facilitate the development, adoption, and diffusion of their innovations (von Hippel and von Krogh, 2006). Free revealing within the context of the community has led researchers to view the user community as the location of resources which users can access in order to facilitate innovation activities. Chandra and Leenders (2012) explain that innovators benefit from community interaction, finding in their study that "... as exposure to information from communities, friends and acquaintances grew, a second or third innovation appeared." (p 472). Similarly, Franke and Shah (2003) find that communities "... provide the innovator with access to resources" (p172), in the form of information, links, and access to others. It is through free revealing, therefore, that user communities have been described as a valuable source of ideas and suggestions (Di Maria and Finotto, 2008).

The community as a source of information is not just important for the development of innovations, but can also provide valuable information leading to their diffusion and commercialisation. Franke and Shah (2003) find that exchanging knowledge in user communities through giving and receiving user assistance, has a positive effect on the diffusion of those innovations within the community. Similarly, Shah and Tripsas (2007) find that user innovators who free reveal, can benefit from a user community by testing, experimenting, and adapting their innovations, giving them the ability to evaluate the innovation's commercial viability. In both cases, therefore, the user community is seen as a source of information that can help innovators during the diffusion stage of the

innovation process.

Through the concept of free revealing, user communities are therefore seen as the location where lead users are able to access information and other resources for the development, diffusion, and commercialisation of their innovations. This view of the community as a source of information, however, focuses on the relationship between individual lead users and a larger social group of users, rather than on the mechanisms and processes of collaborative innovation. Another area that has been researched in the user innovation literature is the relationship between the user community and producer firms, where communities are seen as complimentary assets to a firms' innovation activities.

3.4.2 As a Complimentary Resource

The view of the user community as a source of innovation and information has not just been studied in terms of free revealing between community members, but it has also been looked at in terms of the user community and firms. In this perspective, we see the different ways in which firms can use communities as a source of information and innovation, also leading to the inclusion or creation of these communities into the firm. Dahlander and Wallin (2006), for example, state that the innovation communities in general represent a complementary asset, being able to be the source of new product ideas and stating that firms must have firm-sponsored individuals working in the community in order to gain access and influence. Similarly, Jeppesen and Frederiksen (2006) state that firms can go into user communities in order to benefit from user innovation, which they describe as being both complimentary and alternative to the product's use. Similar to the literature on sticky information, these studies emphasise the relationship between users and firms in relation to the exchange of user information, rather than the processes of co-development.

The relationship between communities and firms has also been studied in terms of the skills that firms must have in order to gain benefits from user communities. Di Gangi and Wasko (2009), for instance, state that the effective appropriation of innovations from communities can be made provided that firms

understand the technical requirements and the reasoning behind that innovation. Within this, it is stated that firms must be able to understand the problem that the innovation is trying to solve, otherwise they may not be adopted. This finding is reinforced by Durugbo and Pawar (2014), who state that firms must be able to listen to and to take into account ideas and innovations coming from users.

In these studies, the relationship between the user communities and the firm is based on the notion of access to these communities rather than their creation. Dahlander and Magnusson (2008), looking at open source projects, state that there are three ways in which firms can gain from the user communities; *accessing*, *aligning*, and *assimilating*. Dahlander and Magnusson describes *accessing* as the view that the user community is an extension of the firm. Studies like Balka et al. (2014) take this perspective explicitly by viewing user communities as a complimentary asset, concluding that there are two ways in which a firm can engage with them, either through accessing an already existing community or through the creation of a new community. Within the user innovation literature, the emphasis has been consistently on making use of existing communities, which then work as an extension of the firm's resources. This approach is similar to early lead user studies which emphasised users as a valuable external source of information and innovation for the firm.

A significant portion of the user innovation literature therefore focuses on user communities as a complimentary resource for producer firms. The emphasis of these studies is on the knowledge possessed by a group of individual users, rather than the processes and mechanisms that make collaborative innovation possible. In addition, the studies that view communities as complimentary resources originate from the understanding that firms make use of already existing communities, leaving the creation of user communities to the field of hybrids.

3.4.3 Hybrid Communities

The study of hybrid communities has largely focused on open source communities purposely built by a firm in order to benefit from collective

software development. Sharma et al. (2002), for instance, make the case for creating user communities by highlighting their benefits, such as faster design and time to market as well as improved quality and lower costs. Although the benefits of collaborative innovation begin to be discussed, the emphasis of hybrid studies is on the deliberate creation and management of these communities for commercial purposes.

The study of hybrid communities differs from previous user innovation approaches in that the issue of community management (referred to as governance) is explored in greater detail. Studies have found that, to benefit from hybrid communities, firms must strategically manage their relationship with the community through a combination of governance, transparency, and openness. Balka et al. (2014) and Sharma et al. (2002), for instance, point out that there should be a balance between the openness and control that firms have in their relationship with user communities, stating that selective openness can benefit value creation for firms without dampening user innovation. In addition, Deodhar et al. (2012) note that, while in user communities there is free revealing, in hybrid communities you have selective revealing, meaning that some aspects of the software are kept proprietary. Franke et al. (2013) add to this by showing that user participation is affected by the notion of distributive fairness and procedural fairness[9], which are judged according to the terms and conditions that are set by the sponsoring firm. These studies therefore present a different perspective to the study of user communities, where the challenges of collective innovation and a volunteer workforce are being discussed in the context of a sponsoring firm.

In addition, the effect that the product characteristic has on collective innovation has also been studied, focusing primarily on the characteristics of the software being developed. For instance, Casadesus-Masanell and Llanes (2011) look at four main types of openness and closed source in software, looking at how much each of these is able to both create and capture value. First, where the software is completely closed source, there is higher value

[9]Distributive fairness makes reference to the fairness that is perceived by the users in terms of the outcome that is selected from user input, while procedural fairness refers to the actual process of how the outcome is selected.

capture, but it also has the lowest value creation. The completely open source software then had the highest value creation, but also had the lowest value capture. From the mixed models, a open core and closed edge had a slightly higher value capture but a lower value creation when compared to software that was closed core with open edges.

To summarise, the literature on hybrid communities in user innovation begins to discuss some of the issues faced in collective innovation. While it moves away from individual innovation activities, the discussion of group processes are focused on the relationship with their commercial host firms. In addition, the empirical setting of these studies has been primarily open source projects, therefore focusing on governance issues that may be specific to the development of software.

3.4.4 Section Summary

This section looked at the study of communities in the user innovation literature, finding that a significant number of studies have so far focused on the exchange of information between users, as well as between users and firms. A portion of the studies have seen user communities as the source of information and resources that can provide lead users with valuable information at the early stages of the innovation process. This approach follows on from much of the literature on lead users and free revealing, and focuses on the development and testing of new products. In addition, the view of the user community as a source of information focuses on the relationship between lead users and a wider group of users, but does not look into the processes of collaborative innovation.

Another portion of the user innovation literature sees communities as a complimentary resource, focusing on the relationship between the community and producer firms. These studies focus on the firm's ability to access and make sense of ideas originating from user groups, while at the same time viewing these groups as a complimentary external resource that can be valuable to their innovation activities. Within this approach, studies have looked at the benefit of accessing user communities along with the problems firms might face when internalising or making sense of user-specific information. Once again, the focus of the community as a complimentary resource is on the exchange of

information between user groups and producer firms, rather than the internal community processes.

The literature on hybrid communities begins to address some of the issues encountered in collaborative innovation, focusing on the deliberate creation and management of user communities for software development. Some of the issues discussed, however, still deal with the issue of information, particularly the ownership of information and its effect on the efficiency in the creation and appropriation of value. The hybrid literature, however, presents some of the early discussion on community governance, albeit in the context of software development.

From this review it is evident that while issues of knowledge exchange have been central to the discussion of user communities in the user innovation literature, the understanding of how these communities are governed has so far only been explored in the context of software development and their relationship with host firms. Understanding user communities has been important in determining who the key user innovators are and who possess the relevant information, something that is important in the relationship between community and producer firms. The understanding of how these communities operate, however, has been largely based on studies of open source projects, with the user innovation literature relying heavily on the fields of Organisation Studies and Information Systems. The following section contains a review of the current literature on governance in the context of open source communities.

3.5 Governance in Open Source Communities

The previous section looked at the study of communities in the user innovation literature, finding that the study of the processes and mechanisms for collaborative innovation have been addressed primarily in the context of hybrid communities. In addition, it discussed the effect the relationship between communities and firms can have on the creation and appropriation of value, noting that a firm's management of the community can affect participation and information sharing.

This section goes on to discuss the management of processes and mechanisms of collaborative development in open source communities. The user innovation literature relies on the study of open source projects to understand the mechanisms and processes involved in collective innovation, with the assumption that the findings can be generalised to other types of user communities. As von Hippel (2001) notes "...open source software projects provide natural laboratories for studying [user innovation communities]." (p86). This section presents a more in-depth view of governance in the context of open source projects, taking from other bodies of literature such as Organisation Theory and Information Systems. This section describes how a number of different mechanism, such as intellectual property protection and control of social behaviour, are used in projects to manage and coordinate activities.

This section is structured as follows. Subsection 3.5.1 looks at the wide range of definitions of governance used in the study of open source. Subsection 3.5.2 then discusses governance in terms of its purpose and what it aims to achieve.

3.5.1 Definition of Governance

The term governance in open source communities is very broad and can encompass a number of tools, structures, and processes, all aiming to manage its members. To obtain a definition it may therefore be important to understand what is being governed, who by, and for what purpose. From the literature it is either implied or explicitly stated that open source communities are self-governed, therefore making the community members both the source and the recipients of governance. O'Mahony (2007) adds to this by stating that there are five key factors in self-governance: (1) independence in terms of resources and control, (2) pluralism, which represents different views, approaches or methods, (3) representation, where ideas from individuals are taken into account, (4) decentralised decision making, and (5) open participation. O'Mahony therefore provides some of the community characteristics that must be present for self-governance.

A closer look at the literature, however, shows that studies on open source governance have used multiple definitions to describe a wide range of practices,

mechanisms, or approaches to community management. Markus (2007), for instance, reviews the literature on open source governance and finds seven different definitions, some of which look at institutional mechanisms like licencing, and others that look at social factors such as reciprocity and idealism. De Laat (2007), however, observes that most of the definitions can be classified into two approaches, the first being governance in a *strict* sense or in a *broad* sense. The strict approach to governance focuses on the cooperation and the coordination of the volunteers, while the broader definition looks at a wider range of issues that focus on the conditions for participation.

Through a review of the literature, Markus (2007) proposes a global definition of governance, taking a similar approach to De Laat's *strict* perspective. Markus borrows and adapts the concept of governance in the public sector to define open source governance as "...the means of achieving direction, control, and coordination of wholly or partially autonomous individuals and organisations on behalf of an OSS development project to which they jointly contribute." (p 152). This definition focuses on three key things, direction, control, and coordination of activities, while at the same time it acknowledges the diverse type of individuals who contribute to it and the different types of ownership. The definition is therefore broad enough so that it can be applied in a number of areas and to a wide range of projects, but it is also open enough to allow for a wide range of interpretations.

3.5.2 Purpose of Governance

Markus (2007) proposes three main purposes of community governance: (1) as a solution to social action dilemmas, (2) to create a climate conducive to participation, and (3) to solve coordination problems in software development. Viewing governance as the solution to social action dilemmas emphasises the reasons why individuals would be willing to participate and contribute to an open source project, looking at tools and mechanisms that deal with a wide range of issues such as motivation and free riding. In this context, studies have found that governance can be used to motivate individuals through both intrinsic and extrinsic factors. At the same time, open source projects can use

governance mechanisms to offer incentives for participation, which may be in the form of social acceptance and status within a community (Zeitlyn, 2003; Bonaccorsi and Rossi, 2006), as well as the idea of contributing not just to the project but to a given practice (von Krogh et al., 2012). From the examples given in the literature it can be said that this perspective of governance is focused on the psychological aspect of participation at the level of the individual.

The view of governance for the purpose of creating a climate for contribution looks at mechanisms that attract new users into the project and help them contribute. Markus (2007) notes that the basis of this approach are the decisions that are made by individuals in order to determine which community they are more likely to join. Franck and Jungwirth (2003) take this view when they make the observation that the primary purpose of governance is to make sure that both the interests of those who contribute as well as those who do not, are looked after. In addition, governance can also affect the climate of contribution through the rewards that are given or are available to individuals within the project (Zeitlyn, 2003; Fuller, 2010). This approach therefore revolves around the idea of project openness and control, where studies have found that a high level of control can diminish participation (West, 2003; Lattemann and Stieglitz, 2005). Finally, the relationship between the project leaders and general contributors can also have an effect on participation, as empirical evidence has shown that poor leadership can have a negative effect on participation and project success (Cox, 1998).

The third view of governance as a mechanism for solving coordination problems focuses on the mechanisms that allow the efficient use of resources and minimises duplicated effort to achieve a common goal. The coordination approach also takes into account a number of things, such as the structure of the software (Baldwin and Clark, 2006) and the effect that this has in duplicating effort or making sure that people are aware of the project itself. In addition to the way in which activities are coordinated, another aspect that is observed is on the quality and the characteristics of the software itself, where numerous studies have noted that governance plays an important role in

software quality (Capra et al., 2008; Carillo and Okoli, 2008; Hossain and Zhu, 2009). This second approach to governance therefore focuses on the effect that governance has on the specific activity of software development, which includes division of labour, efficiency, and quality.

Finally, O'Mahony and Ferraro (2007) view the purpose of governance as a dynamic element of management and coordination which, along with leadership style, needs to change in order to address specific issues within the project. Looking at the Debian project, O'Mahony and Ferraro identify four key governance stages: (1) De-facto governance, (2) Designing Governance, (3) Implementing Governance, and (4) Stabilizing governance. These four governance stages emerged as a result of a change in project ownership and subsequent lack of leadership legitimacy in the Debian project. O'Mahony and Ferraro observed that each stage required a different leadership style, noting that during the stage of governance Stabilisation, for example, the project would benefit most from a leader that was focused on developing the organisational aspects of governance. This study therefore shows that the purpose of governance evolves to meet the specific needs of the project at a particular point in time, which should then be supported by a change in leadership styles.

3.5.3 Section Summary

This section reviewed the current understanding of governance in open source communities. The study of governance in user innovation has largely relied on the fields of Organisation Theory and Computer Science. Governance, while an important topic for the study of user communities, has not been explicitly addressed by the user innovation literature. This has led to a wide range of interpretations regarding not just its definition, but also its purpose (Markus, 2007). Despite the ambiguity in meaning, governance remains an important aspect of the community that producer firms must understand in order to maximise their value for the innovation process, as explained in Section 3.4.

A significant portion of the findings on governance in user innovation communities has been based on studies that focus on the development of

software by a group of individuals. Studies have approached governance through three main perspectives, as a solution to social action dilemmas, to improve participation, and to improve coordination (Markus, 2007). Governance as a solution to social action dilemmas focuses on the means that are used in order to avoid free-riding (Nov and Kuk, 2008), increase motivation (von Krogh et al., 2012), and norms of reciprocity (Wiertz and de Ruyter, 2007). On the other hand, governance as a way to improve participation looks at the means used in order to create a climate that may compel individuals to join a project and sustain levels of contributions (West and O'Mahony, 2008; Baldwin and Clark, 2006; von Krogh et al., 2003). Finally, governance as coordination focuses on the means used to achieve a common goal and to manage effort between individuals (Iannacci, 2005; Kamei et al., 2008; Hemetsberger and Reinhardt, 2009).

The current understanding of governance as coordination in user communities has been based on the study of groups of programmers working on open source, therefore presenting only a partial view of modern open source projects where other types of contributions take place. The merging generation of open source projects incorporate a wide range of contributions, such as graphic design and translations, therefore raising additional questions about governance. As West and Lakhani (2008) state, there could be "... differences between software communities and those that (like Wikipedia) produce other types of information goods" (p6). This therefore could mean that governance for coordination may be different depending on the type of contribution being made by the user.

In addition to the issues of coordination, another important aspect of community governance is that of structure, as it plays an important part how projects are run in terms of division of labour, resources, and leadership. The next section will therefore look at the current understanding on social structure of open source projects from which most of the user innovation literature is based.

3.6 Social Structure of User Innovation Communities

The user innovation literature sees the social structure of open source communities as a key facilitator of community governance (such as Dahlander and Frederiksen, 2011; Sharma et al., 2002; Cattani and Ferriani, 2008). As Crowston and Howison (2005) point out, "...investigating social structure is a useful way to understand team practices. It allows research to begin to investigate questions of coordination, control, socialization, continuity and learning – all topics of great interest in studies of collaborative groups." (p3).

Most of the early literature on open source communities not only describes its social structure as being one of its defining features, but also the source of its advantage over traditional forms of software development (Crowston and Howison, 2005). Raymond (1998) describes this structure as being very much like a Bazaar, a chaotic non-hierarchical community where communication occurs freely between all members. It has been argued that the decentralised and unrestricted flow of information between members creates the right condition for Linus' Law to take place by ensuring that, when a solution is found, it can be effectively communicated to all project members (von Krogh et al., 2003). This understanding is based on the assumption that the lack of hierarchies and other barriers facilitates the sharing and access to information for all community members.

Subsequent studies have found this not to be the case, pointing not only to a centralised structure in terms of communication, but also to a division of labour through task specialisation (Crowston and Howison, 2006). This section reviews the literature on open source community structure, drawing from the literature on User Innovation, Organisation Studies, and Information Systems. Subsection 3.6.1 reviews the literature on the description of user communities, finding that the focus on product development activities has led to the assumption that these communities are formed around a practice, therefore defining its boundaries and membership. Within the boundaries of the community of practice, Subsection 3.6.2 discusses the two dominant approaches to determining community structure,

through participation patterns based on either code contributions or community interaction. Finally, Subsection 3.6.3 looks at the current understanding of how community members move within the social structure to positions of authority.

3.6.1 As Communities of Practice

While the user innovation literature does not have a unified definition for community (West and Lakhani, 2008), studies have either explicitly or implicitly referred to these as communities of practice, in order to focus on activities that are central to the innovation process. The communities of practice literature focuses on learning within a context, through open discussion and exchange of knowledge (Lave and Wenger, 1991), therefore providing a suitable perspective for the study of information transfer between groups of users. In addition, communities of practice provides a boundary and a basis for membership, determining who belongs and who does not (Wenger and Snyder, 2000).

Early user innovation literature looked at the perspective of the user being in the best position to innovate through situated learning[10]. For instance, Slaughter (1993) concludes that, in the building industry, learning-by-doing and user innovation itself are intrinsically linked, noting that "...during the implementation stage [learning by doing] is a key ingredient in producing high value user innovations." (p92). Bai et al. (2006) equally note that in both service and product innovation, users play an important role by providing innovations that are a direct result of learning-by-doing, noting that in their study "...the machine user always does some modifications to the machine by himself in the using time, which means learning in using." (p2).

Communities of practice also emphasise learning between its members, which is an important part that has been explored in greater detail by studies on user communities. Key aspects of the user innovation process, particularly free revealing, are reinforced in the communities or practice literature, where Wenger (1999) emphasises the willingness of individuals to share their ideas in

[10] Although Lave and Wenger (1991) state that there is no clear definition of situated learning, it emphasises both knowledge and actions within a social context.

order to learn best practices. Learning it self has been found to be a key factor in user communities, with Lakhani and von Hippel (2003), for instance, finding that learning and improving a practice is one of the key motivating factors for participating in user communities. Franke and Shah (2003) explicitly describe the user communities as a community of practice, due to its focus on the development of tools and techniques for the practice of extreme sports. Franke and Shah go on to argue that it is the common practice and, the knowledge generated around it, that allows users to develop specific needs and solutions. Therefore, the same mechanisms that facilitate innovation in user communities, are also some of the same mechanisms that facilitate learning in communities of practice.

The situated learning aspect of users innovation has also been highlighted by Coutts et al. (2005), noting that for users to innovate a product, they must first be familiar with its use. In their study of short messaging service (SMS) technology, Coutts et al. had the objective of exploring the likelihood of adoption when the users had no prior experience with the service. In their methodology, they sought to give users experience and the ability to explore the technology in the context of every day life, in other words, to facilitate the use and be able to make the users comfortable with the technology. The study therefore emphasises learning-by-doing, through adoption and familiarisation, as an important first step in user innovation.

In addition to situated learning, communities of practice has also allowed researchers to use the *practice as a community* approach in order to establish the boundaries of the social groups being studied. As expressed by Wenger (1999) and Thompson (2005), there are two key dimensions to determining the boundaries of a community of practice, Structural and Epistemic parameters. Structural parameters focus on the structural characteristics that add focal points to the communities, this includes several aspects such as boundary objects and brokers, therefore moving away relying solely on practice. Epistemic parameters, on the other hand, focus on what (Wenger, 1999, p 125) refers to as "...indicators that a community of practice has been formed...", consisting of 13 characteristics that may define a community of practice. To

identify community boundaries, therefore, Structural parameters therefore focus on the focal points of a community, while Epistemic parameters focus on characteristics that become inherent in a group once a community of practice has been formed.

Wenger (1999) notes that one of the perspective of community of practice is to view the practice itself as the source of cohesion between its members, which is then expressed through mutual engagement, joint enterprise, and shared repertoire, therefore allowing observers to create boundaries by determining who belongs and who does not. Mutual engagement, sees the interaction between users as a key determinant of membership in the community. In the user innovation literature, mutual engagement can take a number of forms, such as socialisation or the sharing of modification, design, or prototyping of a new or existing product, to new and novel ideas. Morrison et al. (2000), for instance, look at all the individuals who interact with each other, creating a network of individuals who use the OPAC library software, whether they innovate or not. Equally, Kuk (2000, 2006) takes interactions between individuals as the sole determinant of group membership. On the other hand, studies such as Zeitlyn (2003) and von Krogh et al. (2003), see mutual engagement as the sharing of information or solution without the emphasis on socialisation.

The second dimension of a community of practice is the joint enterprise, which can be described as the common goals, rules, and norms that emerge within a community as a result of cooperation and collaboration (Wenger, 1999). In user innovation, joint enterprise can be linked to the three conditions for user innovation communities as stated by von Hippel (2001), who talks about free revealing and the social norms that make this possible. Membership to the community is therefore based on the exchange of information about a given practice. For instance, Hau and Kim (2011) regard the community as the group of individuals who freely share ideas about games in general, where they note that users are willing to reveal this information because they have a sense of belonging and also expect others to reciprocate. Similarly, some open source studies such as Stewart and Gosain (2006) regard ideology as one of the defining factors of membership.

The third dimension of the community of practice is shared repertoire, which includes routines, tools, and common actions that form part of the practice (Wenger, 1999). It is this shared repertoire which looks at how the tools are used within the community and how this can bring about the identification and the classification of individuals within that community. For instance Morrison et al. (2000) make the assumption that community membership is based on the tools that are use, particularly the library software system being innovated. Similarly, Prügl and Schreier (2006) look at a community of gamers, determining membership based on the use of the toolkits for the customisation of game characters. Equally, Dahlander and Frederiksen (2011) define community membership based on the use of the Popellerhead software, rather than on interaction or contributions to the project. In addition, Strandburg (2009) takes this perspective when defining the scientific community, emphasising the use of tools and materials as an indicator of membership. These studies therefore focus on the tools that are being used as a requirement for membership.

The user innovation literature has therefore either explicitly or implicitly approached user communities as one formed by practice. The close links between the exchange of information through free revealing and the emphasis on learning by doing have been an important factor in bringing the two concepts together. The resulting implications of this, however, is that it forms boundaries across a particular practice, and may therefore not capture other activities that are not central to the innovation of the output. For instance, in open source project, membership to the community, and therefore its boundaries, are based on contributions to the project through programming activities. This therefore has several implications in terms of the project's social structure its governance, where other social groups within the project may go unobserved.

3.6.2 Approaches to Open Source Social Structure

The previous subsection established that membership to open source communities is based on activities that are related to software development.

	Production Approach	Interaction Approach
Definition of Community	Community of practice, team, technical community.	Social network, user community
Boundaries	Common practice.	Network connections.
Information Space	Repository	Forums or mailing lists.
Population	Homogeneous	Heterogeneous
Roles	Focused on software development	Linked to the use of the software.
Determinant of Leadership	Meritocracy. Influence a result of quality and quantity of contributions to the project.	Social Capital. Influence a result of repeated interaction in the community.
Governance	Resource management.	Membership management.
Coordination	Through control of production.	Through Control of social interaction.
Representative Studies	Fielding (1999), Franck and Jungwirth (2002), Yamauchi et al. (2000), Nakakoji et al. (2002), Lee and Cole (2003), O'Mahony and Ferraro (2007).	Glass (2003), Lakhani and von Hippel (2003), Lehmann (2004), Monteiro et al. (2004), Crowston and Howison (2005), Lin (2005).

Table 3.5: Dominant Approaches to Open Source Research

Equally, the current understanding of community structures in open source is based on the activities of individuals that either carry out software development or exchange information about software development. This subsection presents the two dominant approaches to the study of open source community structure, with the assumption that it represents only a subgroup of those that contribute to the project.

There are two main approaches towards the study of social structure of open source projects (Table 3.5), through production, which focuses on software development, and through interaction, which focuses on communication patterns between members (Crowston and Howison, 2005). The production approach looks at structure in terms contributions made to the project, where governance is based on the control of resources, and participation is defined as code contributions. In contrast, the interaction approach looks at structure in

terms of network connections, where the community can be defined as a social network of interaction, where governance is seen as the control of behaviour, and where participation is defined as communication between members. Both of these approaches use multivariate statistics to determine issues such as the links between individuals, the level of participation, and the type of programming activity they embark on (Martinez-Torres and Diaz-Fernandez, 2014). Finally, as Martinez-Torres and Diaz-Fernandez (2014) point out, these approaches are based on the literature on communities of practice, as it "...refers to the process of social learning that occurs when people having a common interest in some subject or problem collaborate over an extended period to share ideas..."(p58).

The production approach is one of the most used in the study of open source communities, focusing primarily on the creation and maintenance of code. The rationale of the production approach is that, without software production there would be nothing to study (Crowston and Howison, 2005). This approach defines the group of users as either a community of practice or project team, where the population is seen as homogeneous in terms of practice and skill (von Hippel, 2005). The study of community governance is guided by the emphasis on software development, where contributions to the project, in terms of programming, are believed to determine influence and leadership within the project (Fielding, 1999; Yamauchi et al., 2000). Nakakoji et al. (2002) and Lee and Cole (2003), for instance, state that it is through the contribution of software code that individuals are able to gain formal roles in the community, and therefore be able to influence the direction of the project.

In an interaction approach, the boundaries of the community become much broader and are mainly determined by interaction between individuals, with the community being typically defined as a user community or social network. Studies such as Lakhani and von Hippel (2003) use *user community* as their definition, therefore removing the emphasis on software development as a prerequisite for membership and gaining the ability to look at a wider range of contributions. The underlying assumption is that these communities are composed of a heterogeneous population with diverse goals, beliefs, and skills

(Lin, 2005). At the same time, Crowston and Howison (2005) state that interaction between users will determine their influence within the group, arguing that repeated interaction may propel individuals to the centre of the discussion. O'Mahony and Ferraro (2007) add that a high level of interaction is a prerequisite for attaining a leadership role, although they do acknowledge that the perceived impact of the contribution also plays a big role.

Although both approaches make similar assumption as to which members have influence, both differ in terms of how this influence is obtained. The production approach sees programming contributions as the determinant of influence, while the interaction approach sees communication as a determinant of influence. Crowston and Howison (2005), for instance, state that the production approach is flawed, since it does not take into account communication-based concepts like communal learning, coordination, and collaboration. At the same time, it could be argued that the interaction approach may not be able to measure individuals who are key to the software development process but do not interact in the community, something observed by de Laat (2007).

Production-based Structure

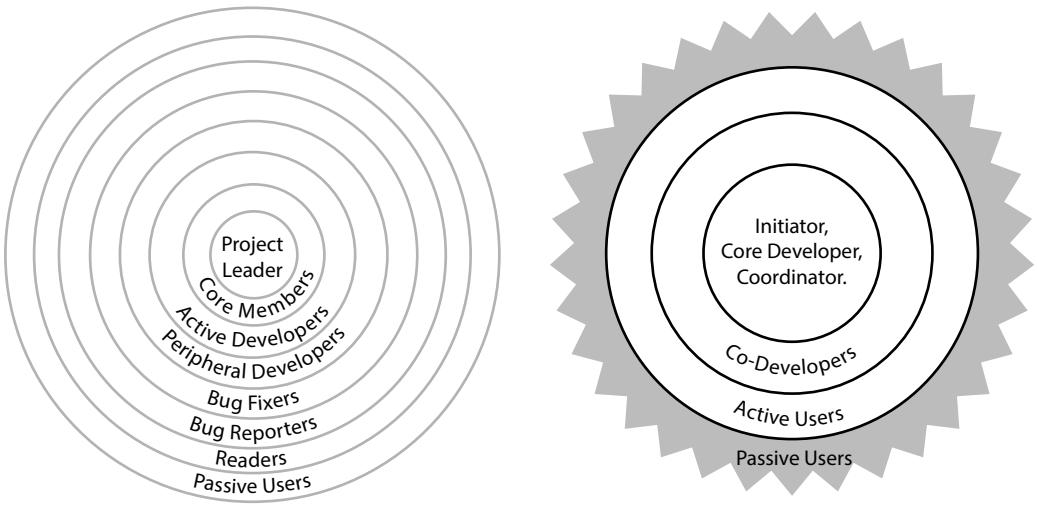
The multilayer onion structure represents the production approach to the social structure of open source projects, placing individuals who contribute most at the centre and the ones that contribute less in the periphery. The dominant form of contribution is programming, and the quantity and quality of the contribution can determine an individual's influence in the project. The main criticism of this approach is that it relies heavily on the amount of contribution but does not take into account the flow of information. The flow of information is important in order for Linus' law to take effect, whereby non-restricted flow of information means that individuals can collectively solve problems.

Early studies on the social structure of open source projects agree that there is a marked difference between users in the community; the differences being not only in terms of roles but also influence. These differences create subgroups, each of them differing in influence, size, and associated roles. One of the first to

state the difference in influence within a project was Cox (1998), who spoke about a core group of developers who controlled the project. Raymond (1998) had previously made the same observation, where he stated that, although communication within the project was chaotic, there was some structure in terms of roles and level of contribution. At the same time Raymond stressed the influence of the project owner in terms of management and coordination of the project, therefore implying that certain individuals may be more important than others.

The differences in role and size of the community subgroups may be down to the programming skills of the individuals. Raymond (1998), for instance, stated that there were two key roles in open source development, co-developers and beta testers. Cox (1998) makes the same observation, but explicitly states that the difference between the groups lies in the programming skills, where there will be a relatively small number of very good programmers, and a much larger group of average programmers. In their study of the Apache project, Mockus et al. (2002) found that the community was divided into three groups of different sizes, where the core group was the smallest in size and the periphery was the largest. It was also observed that the smallest group was in control of the base code, the medium-sized group was responsible for bug fixing, and the largest group reported bugs. Mockus et al. add that the layers are determined by the skill required to carry out the role, with the central layers being occupied by roles that require a higher level of skill and the peripheral roles a lower level of skill.

Since software development is a highly skilled activity (von Krogh et al., 2003) only a small number of community members may be skilled enough to contribute through code fixes, while the majority will contribute through low-skilled tasks such as testing. Lee and Cole (2000), for instance, divided the Linux community into three groups, which they labelled Maintainers, Developers, and Bug Reporters, where the two central roles, the maintainers and developers, required individuals to contribute to the project by submitting code, while the bug reporters did not. As with Mockus et al. (2002), Lee and Cole state that each group is of different size, with an exponential increase from maintainers to developers to bug reporters. As a result of this difference in group size and skill



Seven layer community Structure. From Nakakoji et al (2002).

Four layer team structure. From Crowston and Howison (2005)

Figure 3.2: Open Source Community Onion Structure.

level, the onion-like structure has been viewed as the typical structure of open source communities, with centrality being measured according to the number of individuals in the group, which was itself affected by the skill level required to fulfill that role.

While the early literature stressed the subgroup size and skill level as the determinant of centrality in the non-hierarchical structure, subsequent studies formalised the multilayer onion structure but instead looked at influence as the determining factor of centrality. The onion structure was first formally proposed by Nakakoji et al. (2002), who introduced a seven layered structure based on the roles community members held in the project. The resulting structure, based on influence, is similar to the previous ones where the central roles are seen as the most influential ones and the peripheral ones as the least influentials. In the same way as previous studies, the central roles are mainly highly skilled programming, while the peripheral roles are mainly low-skilled such as readers or passive users. Nakakoji et al. (2002) state, however, that the role the individuals play in the project is an indication of their influence, which themselves are gained through contribution of code.

Moon and Sproull (2002) state that the roles individuals have in the project resemble a form of expected behaviour, where the tasks and responsibilities are

known, and therefore these increase as they are moved to the centre. Dahlander and O'Mahony (2011) state that these communities have vertical authority, where progression is achieved by obtaining a more central role to the project. Dahlander and O'Mahony also makes it clear that these roles are achieved by the community members through contribution of code and technical problem solving discussions. Similarly, Feller and Fitzgerald (2000) state that the contribution is important, but that the quality of the contribution is also taken into account to measure the impact and the effect that it may have in increasing the individual's influence in the community. The attainment of influence in the group is therefore dependent on the programming skills of the individuals, as well as on the quantity of contribution.

The main criticism of basing the social structure on production is that it fails to take into account important concepts such as collaborative development and free revealing, which are heavily reliant on the free flow of information, and which represents a key advantage of open source software development. Crowston and Howison (2005) argued that the social structure of open source projects should be based on the way that information flows between individuals, making it possible for the researcher to understand how problems are solved collectively. They therefore criticised measuring the social structure of open source projects in terms of production because it fails to show how this structure was conducive to the free flow of information. As a result, Crowston and Howison believe the interaction approach, through the use of social network analysis, would be better at determining the open source social structure.

Interaction-based Structure

The interaction approach to social structure places individuals in a network according to their level of communication, with the number of connections determining centrality and influence in the project. The basis of this approach lies in the assumption that the flow of information is the defining characteristic of open source development and therefore individuals who are more socially active are the ones that may have a higher level of influence over others. Using this approach, researchers have been able to apply other concepts of social

network theory such as weak ties and strategic interaction, in order to explain how the community coordinates activities and divides labour. The main criticism of this approach is that it does not take into account the type of contribution an individual makes, and at the same time, it assumes that all the individuals who contribute to the project are also active in the observed social space.

The use of social network analysis (SNA) works with the assumption that there is no hierarchical structure in the community, and that the importance of an individual can be measured not through vertical hierarchy, but through horizontal centrality (Nakakoji et al., 2002). This centrality is determined by the number of connections to other members, with the assumption that the flow of information could be considerably disrupted should that individual be removed from the network (Crowston and Howison, 2005). Focusing on the flow of information strengthens the importance of communication and free revealing between project members. Through the use of SNA, researchers have been able to address a number of different phenomenon, such as weak ties (Lakhani, 2006), social capital (Singh et al., 2011), and strategic interaction (Kuk, 2006).

The interaction approach was formally introduced by Crowston and Howison (2005), who criticised the production approach for not taking into account the importance of communication and free revealing in open source software development. Using this method, the observation was made that some individuals are more connected than others, therefore being able to differentiate between a core and a periphery and give the network values of centrality. The difference between those with high levels of centrality and those with low levels of centrality can be explained through what Kuk (2006) describes as strategic interaction, where individuals in the periphery will actively seek those who are more knowledgeable to interact with. Xu et al. (2005) refer to this as preferential attachment, defining it as the increased likelihood that new individuals will interact with others who already have a high number of connections – calling it the "the rich get richer" phenomenon (2005, p4). These two concepts are important in the study of open source as a social network, since strategic interaction and preferential attachment can be used to justify the

relationship between network centrality and project influence (Valverde et al., 2006).

A number of studies have since built on the link between centrality and influence by attaching formal project roles to individuals based on their network position. Studies such as Lakhani (2006) and Dahlander and O'Mahony (2011), for instance, look at flow of information inside and between networks, where each of the individuals will have a different role depending on their position. In their study, they find that peripheral contributors are more likely to have innovative ideas, as they may draw from multiple sources. In addition, however, it is the core contributors who would be most able to innovate as they would have access to a wider range of resources.

The findings on the link between network position and an individual's ability to innovate match the current understanding on the flow of information through social networks. The core, for instance, is the most established section of a group, with the highest level of prestige (Wasserman and Faust, 1997), they are also the least likely to have creative ideas (Granovetter, 1973). Actors that occupy the core, however, tend to have stronger ties with other individuals and can therefore exert influence through the use of that close relationship (Dahlander and Frederiksen, 2011; Uzzi, 1997), and a wider access to the majority of the group (Perry-Smith and Shalley, 2003). Core individuals, however, may be reluctant to adopt innovative ideas because these may pose a risk to their established reputation (Becker, 1970). Core individuals can also have a high level of homogeneity between them, meaning that they do not have access to different types of information (Burt, 2004). Therefore, although core individuals have the ability to influence others in the group, it may be difficult for these individuals to see the value of new ideas (Hargadon, 2005), or to be able to solve problems in a new way (Schilling, 2005).

Peripheral individuals may have access to other sources of information, and can therefore introduce new ideas, but are less likely to have influence over others in order to implement them (Becker, 1970). Peripheral individuals are those that have weaker ties with the social group, as a result, they are more likely to move between or have access to a number of social groups and different experiences

(Jeppesen and Lakhani, 2010). This cross-group spanning exposes the individual to a wider range of information and processes, which can increase their ability to generate creative ideas (Schilling, 2005). This concept, also referred to as weak ties, was further expanded by Lakhani (2006), who found that individuals in the periphery are more likely to introduce new ideas or practices to the group. On the other hand, the relative detachment from the group means that the periphery members are less able to influence others and to get their ideas heard and implemented (Cattani and Ferriani, 2008; Dahlander and Frederiksen, 2011). Cattani and Ferriani (2008) spoke about the people in the middle as being most creative in the group, arguing that the group members placed between the core and the periphery are more likely to have both traits, influence and creativity. Dahlander and Frederiksen (2011) refer to these individuals as cosmopolitans, noting that they are the most successful at implementing new ideas and therefore being the primary source of innovation.

One of the biggest criticisms of this approach is that it assumes that all the individuals who contribute to the project participate in one social group. Dahlander and O'Mahony (2011), however, found that the activity of individuals changed after gaining more responsibilities, noting that they may turn more into coordination roles which may decrease their activities in the community, therefore decreasing their centrality in the network. At the same time, Cox (1998) points out that average programmers may sometimes be the most vocal, which can then increase their network centrality and give them a false indication of influence. Most importantly, using the network approach will limit the scope of the study to one team working in the project. Any form of coordination or division of labour will then be done at the team level, therefore not being able to account for other teams of individuals who are contributing to the project in other ways.

3.6.3 Attainment of Centrality and Authority

The discussion on the attainment of centrality by peripheral contributors has been addressed in the user innovation literature within the context of open source software. Studies have predominantly focused on the production-based structure of communities, where centrality has been linked with authority over

the project and the activities of others (see von Krogh et al., 2003; Dahlander and O'Mahony, 2011). These studies conclude that individuals need to make technical contributions in order to attain membership to the project. Once they have attained membership, individuals will need to coordinate activities and span community boundaries in order to attain authority.

von Krogh et al. (2003) deal with the issue of community joining and specialisation, finding that there is a 'joining script' which individuals need to follow in order to become project members. This script is based on the observation of open source communities, where users who contributed to the project through working code fixes were more likely to be given access to the repository, therefore becoming members. In their study, however, von Krogh et al. only look at attaining membership by gaining access to its resources, and do not look into how members gain authority over others.

Dahlander and O'Mahony (2011), on the other hand, look at how members gain authority within an open source project, which they term 'progressing to the centre'. In their study, Dahlander and O'Mahony find that the technical contributions mentioned by von Krogh et al. will allow users to acquire membership[11] to the project but not authority. Instead, Dahlander and O'Mahony argue that authority can be gained by participating in coordination discussions and by communicating across sub-project boundaries.

This shares some similarities with Lave and Wenger's (1991) proposition of legitimate peripheral participation in communities of practice. Lave and Wenger note that a progression is made from apprentice to master through 'learning by doing' and through the exchange of knowledge. Where it differs is that von Krogh et al. (2003) note that the act of joining a project is a result of providing a 'gift' from the knowledge the individual already possesses, rather than one they have learnt as a result of community membership. At the same time, the findings by Dahlander and O'Mahony (2011) state that the progression is based on other activities besides the community's core practice.

For both the production-based and the interaction-based model of social structure, centrality is seen as an indication of authority and importance to the

[11]In their study, von Krogh et al. refer to formal membership, where contributors are added to an official list of members.

project. The current understanding is that technical gifts will give membership to the project while coordinating activities and cross boundary communication leads to authority. One of the key questions, however, is whether the coordinating discussions and the cross-boundary communication is as a result of authority. That is to say, whether after attaining a position of authority project members are more likely to exert that authority by engaging in coordinating work and cross-boundary communications.

3.6.4 Section Summary

This section presented the current understanding of the social structure of user innovation communities. A significant number of studies on user community structure focus on open source projects, which may be typically published in journals in the field of Organisation Theory or Information Systems (Martínez-Torres, 2012). The findings in these studies have informed the user innovation literature about key aspects of the innovation process, primarily coordination and exchange of knowledge.

Although the early open source community literature made reference to a Bazaar structure where communication is chaotic and un-coordinated (Raymond, 1998), other studies have found that open source communities are controlled through a centrality-based structure (Kamei et al., 2008). This therefore leads to a separation between two broad groups of users, those that participate the most (referred to as *core*) and those that participate less (referred to as *periphery*). Following studies have found that the role and influence an individual has over a project may be determined by their level of involvement, or centrality (see Crowston and Howison, 2006; Lakhani, 2006; Cattani and Ferriani, 2008). The literature points to two approaches to calculating the centrality of community members, through production and through interaction.

Using the production approach, researchers have identified a multi-layered onion-like structure based on project roles, with highly skilled roles in the core and the lower skilled roles in the periphery (Nakakoji et al., 2002). The type and level of participation is used as an indication of influence within the community, where the individuals who participate the most, and whose contribution is seen

as being valuable, will be located at the core of the structure (Crowston and Howison, 2005). The individuals with lower levels of participation, and those who carry out less crucial roles, will be located in the peripheral layers.

Using the interaction approach, researchers have identified a three layered onion-like structure based on an individuals effect on the flow of information. According to Dahlander and Frederiksen (2011), individuals located in the peripheral layers are more likely to introduce new ideas to the community, but lack the influence to implement them. Individuals located in the core layer will have a higher level of influence within the group, but will lack new ideas. Finally, the individuals located in the middle will have both influence and new ideas, and are more likely to innovate in the project (Cattani and Ferriani, 2008).

In addition to the effect of centrality in authority, a few studies have also looked at the attainment of key roles within the project. These studies have found that users progress to the center by making significant technical contributions as well as through high levels of participation in coordinating activities.

The findings on the structure of open source communities, however, are based on the assumption that users contribute to the project through the practice of software development. As with community governance, the current understanding of the structure of user innovation communities is based on the study of one social group in an open source project, without taking into account other groups of users that contribute through different practices.

3.7 Chapter Summary

This chapter presented a review of the user innovation literature with an emphasis on user communities and the processes of collective innovation. From the literature reviewed, user innovation focuses primarily on individual user innovators (lead users) that possess valuable knowledge that may be difficult to transfer (sticky information) to other users or to firms. To understand processes and management that facilitate collective innovation in user communities, the user innovation literature has had to rely on the fields of Organisation Science

and Information Systems and their study of open source projects. By doing so, it has traditionally approached user communities as communities of practice, focusing primarily on a single social group involved in the development of a single product. In addition, the management of user communities has so far focused on the study of groups of programmers working on open source projects, potentially presenting only a partial view of modern open source projects.

User innovation is part of the Open Innovation field of study which, in contrast to the Schumpeterian model of innovation, looks at sources of innovation that are external to the firm. While the Open Innovation paradigm focuses on issues of efficiency in commercial innovations (Bogers and West, 2012), user innovation focuses on their attractiveness through the inclusion of user ideas (von Hippel, 2005).

The three dominant themes in the study of user innovation focus on key user innovators and the exchange of information. A significant part of the user innovation literature looks at the lead user as a source of novel and innovative ideas, and methods through which firms may be able to identify them. Other studies focus on the challenges in transferring information between lead users and firms, leading to the conceptualisation of sticky knowledge and its varying types. Finally, another body of literature looks at toolkits, which can be used to facilitate the sharing of information at the design stage of development or by allowing users to modify the products themselves. These dominant themes, which form a significant percentage of the user innovation literature, deal primarily with individual users and the information they possess, rather than processes and mechanisms of collaborative user innovation.

The study of communities in user innovation follows much of the same approach as the above themes, once again focusing on individuals and their knowledge. User innovation studies began to look at communities and networks of users as the location where lead users could obtain valuable information and assistance. Equally, user communities have been referred to as a complimentary resource for producer firms, who can strategically access them in order to support their innovation activities. In the studies of hybrid communities, the

processes of collective innovation begins to be addressed, where community management or governance is described as being key to the innovation process. The focus of the hybrid literature, however, remains on the firm and how their relationship can affect collective innovation.

While being one of the most important factors affecting collective innovation, governance in the user innovation context has not been clearly defined. Although there is no universal definition, a review of the literature finds that governance serves three key functions: motivation, coordination, and as a solution to collective social action (Markus, 2007). The understanding of governance in user communities, however, relies of certain assumptions that are made regarding membership to the community and its social structure. In addition, key findings that have been generalised to user communities originate from other fields of study, particularly Organisation Theory and Information Systems, retaining some of their assumptions.

By relying on the field of Organisation Theory and Information Systems, the processes and mechanisms of collective user innovation have been predominantly focused on groups of software developers. In the key literature, the study of open source communities has been approached from the perspective of communities of practice, therefore placing greater emphasis on the group of users who contribute to the project through software development. Both fields of study do not take into account the larger group of users that contribute to open source through non-programming activities, such as documentation or graphic design. The current understanding of both the governance and structure of user innovation communities may therefore currently represent only a partial view of what occurs in a modern open source project.

3.7.1 Gap in the Literature

The literature review presents three main gaps in the current understanding of governance in user innovation communities. First is the issue of membership, participation, and boundaries of the social group that form part of the project. Cohen (1985) states that identifying community membership and boundaries is

an important factor, as these "...encapsulate the identity of the community and, like the identity of an individual, is called into being by the exigencies of social interaction."(p12). The literature reviewed in this chapter states that membership to the project is characterised by formal access to the repository, or contributing to the project through software development or programming discussions. Restricting project membership to those that contribute through software development therefore fails to take into account the much larger group of users that add value to the project through other types of contributions. In addition, the assumptions regarding membership and boundaries raise a number of issues in terms of community governance, in that it does not currently take into account the effect the different types of contributions may have in terms of coordination.

The second issue raised by the literature is on the reliance of the community perspective and the effect it has on the assumptions being made about the relationship between individuals in the project. As Bell and Newby (1971) state, in community studies "Theory defines what is relevant and what is not; in a very real sense it determines what is seen."(p64). Relying on the term *community* therefore emphasises non-hierarchical structures, placing more emphasis on horizontal authority as a form of governance. Most importantly, the approach taken by the researchers means that what has so far been observed has been one social group within a project, with a very narrow definition. This small group will invariably have the same practice and will be almost homogeneous, and therefore, when it comes to division of labour, the researcher will only identify roles and tasks relating to software development. At the same time the community approach will mean that the study of governance will typically focus on the management of individuals in the context of a community, rather than the management of work groups in the context of an project. Focusing on one community within a project may be similar to the study of a single functional silo within a much larger organisation. By removing the assumption of open source projects resembling communities, it may be possible to find a more complex social structure that incorporates a wider range of practices.

Finally, while Dahlander and O'Mahony (2011) present a view of the dynamic nature of formal managerial roles, not much is known about the nature and characteristics of these roles in the wider context of an open source project. Dahlander and O'Mahony focused on the attainment of roles of authority inside a community of developers, but they do not specify how these same roles may apply across various communities within the same project. It is therefore not well understood whether the same mechanisms are active in other communities that form around different practices, or whether these are specific to each work group. In addition, it is not known whether there is a difference between authority at community level and authority at the project level.

This thesis will therefore aim to fill these gaps in knowledge by approaching the study of open source governance from the user innovation perspective, widening the definition of membership and taking into account all activities that add value to the project. This thesis will aim to determine the different ways in which users contribute to an open source project and their effect on governance. In addition, it will look at the differences between each community of practice involved in the project, and the way in which these are governed at the project level. The following chapter will present the research aims and objectives.

Part II

Methodology

Chapter 4

Methodology

4.1 Introduction

The previous chapter presented three main gaps in the literature which this thesis aims to fill. First it finds that in the user innovation literature, the view of community membership is narrow, where only one community of practice is studied. Second, this has the effect of viewing the project itself as being composed of only one group of community of practice, therefore basing its governance and structure on one type of contributors. Finally, although governance mechanisms are important for collaborative innovation, these mechanisms have focused on community level coordination and have not yet been explored at the project level.

This chapter presents a research method for identifying governance within an open source project where users contribute through a number of practices. Section 4.2 presents the four primary research questions, aiming to determine (1) how users contribute to the project, (2) the different communities active in the project, (3) how each community is managed, and (4) how the project is managed. These four research questions are further composed of smaller supporting questions that direct the line of enquiry for each research question. Table 4.1 provides a summary of the primary and supporting research questions.

Section 4.3 presents the research design of this study. To answer the research questions, this study takes a pragmatist approach, with the emphasis

1. *How can users contribute to an open source project?*
1.1. What licences are used by the project?
1.2. What products and services are offered by the project?
1.3. What is the project's development process?
1.4. What ICT is used by the project?
1.5. What kind of contributions are made through each ICT?
2. *What communities of practice do these form?*
2.1. What are the managerial roles within each type of contribution?
2.2. Who are the key contributors?
2.3. How are managerial roles formulated and delegated?
2.4. What are the rules that govern participation for each type of contribution?
3. *How does each community manage its activities?*
3.1. What is the structure of each community?
3.2. What are the patterns of contributions for the core and periphery?
3.3. What is the relationship between the core and managerial roles?
3.4. What are the processes of contributing to the project?
4. *How does the project manage all its activities?*
4.1. What are the goals of the project?
4.2. Who are the official project leaders?
4.3. What are the roles of the project leaders?
4.4. What type of contributions are made by the project leaders?
4.5. What is the relationship between each community of practice and the project leaders?

Table 4.1: Summary of research questions

on utilising the methods and techniques most suitable for each question. This study therefore utilises a single in-depth case-study of the CyanogenMod project, with the emphasis being on obtaining richness of data of a revelatory case.

4.2 Research Questions

The review of the literature in the previous chapter on user innovation and user innovation communities highlighted three main aspects that form the focus of this thesis: (1) The different ways in which users can contribute to an open source project; (2) The effect different types of contributions has on social structure; (3) The effect different types of contributions has on governance. The overall objective of this thesis is therefore to determine how an open source project manages multiple forms of contributions. To do so, this study is guided by four research questions which deal with participation, social structure, community governance, and project governance. Table 4.1 presents a summary of the research questions.

The first two research questions focus on the nature of membership of the user-led project. The current view of project membership is both explicit in determining the characteristics of membership or implicit in detailing the reasons why individuals become part of the project. From the explicit perspective we see studies referring to programming activities and access to programming tools as being an indication of membership (e.g. von Krogh et al., 2003). At the same time, from an implicit approach, we see that the projects are often referred to as communities of practice, emphasising only one practice and, as a result, one form of contribution (see West and Lakhani, 2008). As explained in the previous chapter, this therefore presents a limited view of membership of user-led projects, where users may be able to contribute through a number of practices.

The first two research questions therefore address this gap in knowledge by reevaluating the breadth of participation in an open source user community. The first question will focus on determining the different types of contributions made for the development and maintenance of products or the provision of related

services. The second research question will determine the effect the different types of contributions have on the social structure of the project.

Research Question 1: ***How can users contribute to the project?***

Research Question 2: ***What communities of practice do these contributions form?***

Once the different communities of practice have been identified, it may then be possible to identify how each of these communities are governed. As mentioned before, the current understanding assumes that all contributions are related to programming, therefore making the assumption that the governance of programming activities are applicable to all project contributors. The second gap in knowledge this thesis will address is the way in which the different types of contributions from each community of practice are governed.

Research Questions 3: ***How does each community manage its activities?***

This third research questions will look at governance at the community level, looking particularly at the coordination approach to governance. The assumption will be that the project aims to achieve a goal, and it will be this goal that directs coordination activities. Therefore, this will include the goals of the project, the activities around the project, the development process, rules of participation, and community structure.

The third gap in knowledge this thesis will address is focused on leadership roles within each community and the project. The reviewed literature shows that the current understanding of participation and authority within the project is limited to the way in which users gain access to resources or obtain formal roles of authority. This thesis will therefore add to this current understanding by looking at how roles are defined and delegated at both community and project level.

Research Question 4: ***How does the project manage all its activities?***

This research question will explore a number of issues regarding roles of authority. First it will look at identifying official managerial roles at the level of the community, looking at how these are defined and delegated. Then it will look at official roles at the level of the project, identifying project leaders, their roles and functions at community and project level. Finally, it will try to determine the relationship between community structure and role delegation, which include community-level activities.

4.3 Research Design

This research takes a pragmatist perspective, utilises a single in-depth case study approach, and uses data mining, documentation, and semi-structured interviews as data sources. The focus of this research is to extend the current knowledge on project governance by understanding how a project coordinates its multiple forms of contributions. Pragmatism will aid the study by providing a perspective that is focused on the problem and is oriented towards real-world practice, giving the researcher the freedom "...to choose methods, techniques and procedures that best meet their needs and purposes" (Creswell, 2003, p11). The pragmatist perspective will therefore provide flexibility to mic a number of methods and techniques in order to address each of the four research questions.

A singe case study approach was chosen for three reasons. First, from a pragmatist perspective, the single case study approach is consistent with previous studies on user innovation and open source communities. The second reason is that the CyanogenMod project presents what Yin (1989) describes as a revelatory case, in that it provides a view of the new generation of hacker communities that deal with hand-held mobile devices with greater software-hardware integration. Finally, a single case study will also allow the researcher to collect in-depth information on the case, providing an opportunity to collect various forms of data that will then help with triangulation.

The research design constitutes three main aspects of research, the paradigm, the strategy for enquiry, and the data collection method (Creswell, 2003). The research paradigm looks at the knowledge claims and theoretical

perspectives taken by the researcher, this is explored in greater detail is Subsection 4.3.1. In terms of the strategy for enquiry, this deals with the choice of methods and the link this has with the research question, covered in greater detail in Subsection 4.3.2. Following that, Subsection 4.3.3 provides further detail as to why CyanogenMod was chosen as a single case study.

4.3.1 Paradigm

This study will take the pragmatist world view, which allows the researcher to focus on the research problem and select methods and techniques that best match the social context. As James (1955) states, pragmatism is focused on settling metaphysical issues by focusing primarily on their practical consequences. In essence, pragmatism encourages the use of multiple forms of methods and techniques, with the emphasis being on benefiting the most from their strengths when understanding social phenomena (Rossman and Wilson, 1985).

The guiding aim of this research is to extend the current knowledge on project governance by understanding how a project coordinates its multiple forms of contributions. To do so, there are four main research questions that will shape the direction of enquiry, each requiring a different technique and type of data. The pragmatist approach will therefore provide flexibility when choosing methods and techniques that will work at the time and within the context of study (Creswell, 2003) for each question.

Studies on open source and user communities use both quantitative and qualitative data in order to study different aspects of the phenomenon. For instance, qualitative data such as interviews and documentation have been used in studies to understand processes and activities within projects (e.g. O'Mahony, 2003). Other studies rely on quantitative data, particularly to study patterns of participation and social structures within the project (e.g. Crowston and Howison, 2005). The pragmatist approach therefore provides an adequate starting point that would allow for the mixture of both world views, allowing not just a more complete view of communities but also for the triangulation of data.

4.3.2 Research Strategy

As with most of the literature on open source, this study will follow a single case study approach, where the emphasis is on obtaining an in-depth view of the CyanogenMod project. In open source literature, case studies are one of the most common approaches, which is facilitated by the unrestricted access and wide availability of large volumes of archival data concerning the project (von Hippel, 2005; Crowston et al., 2008). As mentioned in the literature review, a significant number of studies looking at the social structure of open source projects will first look at a project and then at the social group involved in its development. This can be explained by Crowston and Howison's (2005) observation that, without the project or the creation of code, there would be nothing to study. This study will therefore follow the established approach of using a case study as the main strategy for addressing the question and formalising the gathering of data.

As Crowston et al. (2012) observe "Considering research methods and levels of analysis, the dominant form of FLOSS [Free/Libre Open Source Software] research overall was the study of a single project at the group level" (p7). A benefit of using the case study approach is that it may lend itself to the collection of data through various sources, therefore achieving triangulation (Yin, 1989). In this study, the case study approach will make it possible to gather data from three sources, documentation, observation, and interviews, which will facilitate answering a diverse range of questions from patterns of contributions to role definitions.

4.3.3 CyanogenMod as a Case Study

The CyanogenMod project was chosen as a case study for three main reasons, its position within the context of a larger technological trend, the lack of non-commercial involvement, and its large user-base and popularity. The ROM modification communities represent an emerging technological tend of an ever increasing link between software and hardware, affecting therefore open source software development. Open source projects have primarily looked at the way that development has taken place by only looking at the software dimension to

Name	Google Hits	Estimated Users (Millions)
CyanogenMod	11,800,000	[i]11.6
MIUI	7,140,000	[ii]30.0
AOKP	4,330,000	[iii]3.5
Paranoid Android	3,400,000	
Pac ROM	122,000	

Table 4.2: Other AOSP-based ROMs. [i] stats.cyanogenmod.org, [ii]Horwitz (2014), [iii]Pandey (2013)

it. In recent years, with the increase in the accessibility of hardware and mobile devices, open source has taken a new direction where software implementation can be device specific. Hardware projects such as Arduino and Raspberry Pi, can be said to facilitate making hardware and firmware modification, springing new communities of maker spaces such as fablabs and hacker spaces. Mobile devices present the same type of challenge to developers, where the focus is not just the modification of software, but of the software in the context of a particular hardware. The ROM modifying communities therefore, represent an emerging field in user innovation where both software and hardware skills play an important role in the modification process.

In terms of community modified ROMs, CyanogenMod also represents the largest non-commercial ROM project. In many successful open source projects, there have been instances were firms also participate in the development process for commercial purposes. A prime example of this is the Linux kernel, with a wide range of firms being active in its development, including IBM, HP, and Samsung. In the same case, the AOSP also has a wide range of commercial interest through the Open Handset Alliance (OHA), whose members includes handset manufacturers such as HTC, Acatel, and LG, as well as mobile operators such as Soft Bank, Docomo, and Vodafone. Similarly, within the ROM modification community, MIUI, with the largest number of users, is used by Chinese device manufacturer Xiaomi, who release some of the source code to the public. Xiaomi devices, and therefore the MIUI ROM, is primarily sold in China, Hong Kong, and Taiwan, with their flagship smartphone device, the Mi 2S, outselling Samsung devices in the first half of 2013 (Ho, 2013).

Being non-commercial, the CyanogenMod project will provide an insight as to how governance takes place within a completely volunteer work force. As has been seen in previous studies, the participation in an open source project by a commercial entity will have an effect on the way that the community is governed. The relationships between firms the community, as explored by studies such as Shah (2006), can have an effect on the motivation of individuals and the way in which rules are formed. Focusing on a non-commercial project will make it more probable that the approaches to the coordination of contributions would be formulated solely by the volunteers involved in the project.

The CyanogenMod project is seen as the most popular and widely used customised AOSP ROM. At the time of writing, a search on Google brings out over 11.8 million hits, while the closest non-commercial ROM, AOKP, brings in only 4.8 million. In terms of install base, the CyanogenMod statistics state that the ROM has been installed over 11.6 million times, while the AOKP has an estimated 3.5 million users. Finally, the CyanogenMod project is also one of the oldest AOSP community projects, having been started in 2008, while the AOKP began three years after, in 2011.

Finally, the relative age of CyanogenMod, when compared to other similar projects, means that the project's governance mechanisms are likely to be more established. O'Mahony and Ferraro (2007), for instance, discus how governance mechanisms in open source projects evolve and consolidate over time. The age of the CyanogenMod project, along with its large user base, may be an indication of the project's success, and therefore the efficacy of its governance mechanisms.

4.4 Chapter Summary

This chapter addressed the research questions, the method used for answering the questions, and the resulting methodological issues of this study. The objective of this study is to determine how an open source project manages a wide range of contributions. To meet this aim, the study is guided by four research questions:

Q1 - How can users contribute to the project?

Q2 - What communities of practice do these contributions form?

Q3 - How does each community manage its activities?

Q4 - How does the project manage all its activities?

To answer these questions, the research follows a pragmatist approach in order to focus on the appropriate methods and techniques to answer the question. Consistent with the approach, this study uses multiple sources of data, quantitative and qualitative, helping to obtain a richer set of data and increase validity in the results. Following the dominant methodological strategy for open source community research, a single case study approach was chosen in order to obtain a richer level of of data.

The CyanogenMod project was chosen as the case study for two reasons, the revelatory nature of the case and accessibility to data. CyanogenMod project represents what in Chapter 2 was described as a new generation of open source projects, with a more sophisticated use of ICT and a higher level of hardware-software integration. In addition, the data in the project, in terms of documentation and contributions, is easily accessible, making it possible to obtain a richer data set.

Chapter 5

Data Collection and Analysis

5.1 Introduction

This study utilises three source of data to address the research questions: (1) Participation metrics; (2) Project Documentation, and; (3) Semi-structure Interviews. Data was gathered from the project's multiple web-based ICTs as well as the through interviews with the core team members. The majority of the data was taken from observation and documents that were freely available to the general public. The span of the data collection was three months, while the interviews which supplemented the data took place during June 2013 to August 2013. The process was iterative, which means that the first stage of participation metrics and repository data was taken between January 2013 and March 2013. After the interviews, there was a second round of data collection that took place between December 2013 and January 2014. Table 5.1 presents the research questions and the data used to answer them. Figure 5.1 contains a timeline of when the data was collected and the time frame covered.

5.2 Participation Metrics

Participation metrics were used to answer five supporting questions, regarding the types of contributions made, key contributors, and community structure. Data was extracted from five ICTs used by the project; the repository, bug review system, forum; wiki, and IRC. The data set had a total of 272,897 data points

Research Question	Data source		
	Participation Metrics	Project Documentation	Semi-structured Interviews
1.0 How can users contribute to an open source project			
— 1.1 What licenses are used by the project?		x	
— 1.2 What products and services are offered by the project?		x	
— 1.3 What is the project's development process?		x	
— 1.4 What ICT is used by the project?		x	
— 1.5 What kind of contributions are made through each ICT?	x		x
2.0 What communities of practice do these contributions form?			
— 2.1 What are the managerial roles within each type of contributions?		x	x
— 2.2 Who are the key contributors?	x		
— 2.3 How are managerial roles formulated and delegated?			x
— 2.4 What are the rules that govern participation for each contribution?		x	x
3.0 How does each community manage its activities?			
— 3.1 What is the structure of each community?	x		
— 3.2 What are the patterns of contributions for the core and periphery?	x		
— 3.3 What is the relationship between the core and managerial roles?	x		
— 3.4 What are the rules of contributing to the project?		x	
4.0 How does the project manage all its activities?			
— 4.1 What are the goals of the project?		x	x
— 4.2 Who are the official project leaders?		x	x
— 4.3 What are the roles of the project leaders?		x	x
— 4.4 What type of contributions are made by the project leaders?	x		x
— 4.5 What is the relationship between each community and project leaders?		x	x

Table 5.1: Research Questions and Sources of Data

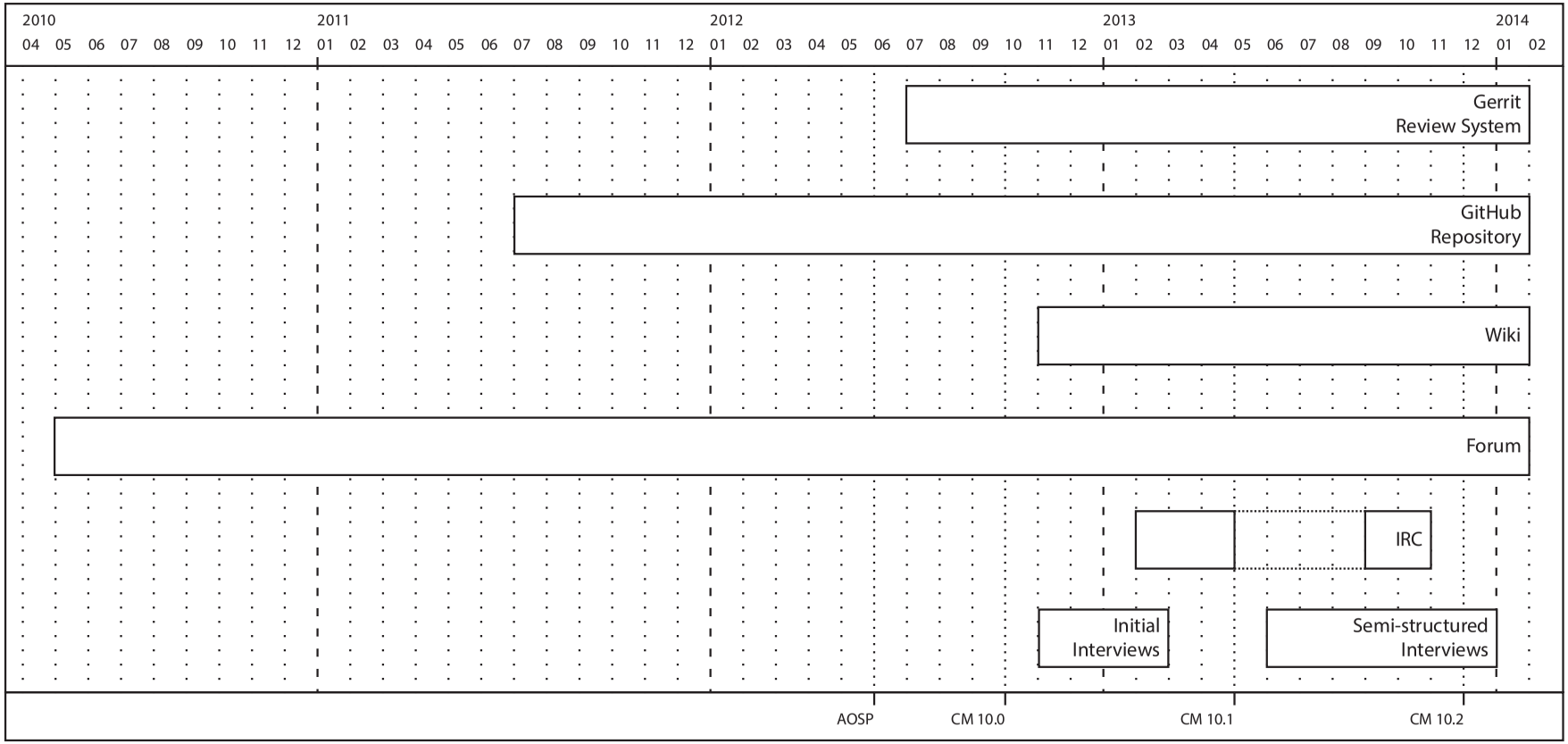


Figure 5.1: Time Length of Dataset and Collection

spanning a period from May 2010 until February 2014.

5.2.1 Questions Addressed

Participation metrics were used to address six of the supporting questions for this research, dealing primarily with contribution patterns and community structure. The first supporting question looks at the types of contributions that users can make through the project.

1.5 What kind of contributions are made through each ICT?

For this question, semantic analysis was used to identify key words that may denote various forms of contributions. The keywords were extracted from descriptions of the submissions in the repository and review system, as well as from messages posted in the IRC and forum. An initial word count provided an overview of key activities, followed by a detailed text analysis using the Natural Language Toolkit (Bird et al., 2009).

The second supporting question identifies key contributors to the project.

2.2 Who are the key contributors?

For this question, data on the user name and their respective number of contributions was taken from all of the project's ICT. The number of contributions were then counted and the individuals with the highest number of contributions were classified as being key to the project . The semi-structured interviews were then added to this question by identifying other key individuals that may not be high contributors.

The third question addressed is that of the core-periphery structure of each community.

3.1 What is the structure of each community?

Question 3.1 builds on Question 2.2 by determining the relative position of other individuals in the project, clustering participation into three categories: Core, Middle, and Periphery. User names for all the contributions were counted from each of the project's ICT. Using the Bradford Law method, developed by

Crowston et al. (2006), the structure of the community was divided into three layers (explained further in Section 5.5).

Once the different sub groups were determined, the findings were compared to official leadership roles in order to address question 3.2 on contribution patterns.

3.2 What are the patterns of contribution for the core and periphery?

As explained in Chapter 3, the current understating of community structure is that there is a difference in the type of contributions made by the core and the periphery (see Nakakoji et al., 2002). Question 3.2 therefore uses the results obtained in Questions 1.5 and 3.1 to determine whether each sub group within the community specialises in a particular type of contribution.

In addition, as Crowston and Howison (2005) notes, there may be a difference between individuals who contribute the most to the project, and those that have a formal leadership role. Question 3.3 therefore uses participation metrics to determine the position within the community of the official project leaders.

3.3 What is the relationship between the core and leadership roles?

Finally, participation metrics were also used to partly address the question on the relationship between project leaders and each community.

4.4 What is the relationship between each community and the project leaders?

Question 4.4 was divided into two parts, with the first being addressed through participation metrics and the second part through semi-structured interviews. Question 4.4 adds to the findings in question 3.2 by taking a broader approach of the activities carried out by the leaders within each community. The first part of this question looked to determine how official project leaders contribute to the project and when. The second part of the question used semi-structured interviews in order to determine activities that are not observable through the project's ICT.

5.2.2 Description of Dataset

The participation data was downloaded at one point in time for each interaction space, it was then stored as html files, and then parsed through a series of programs, written using the Python language, which then resulted in a table of contributor statistics. The data for the repository, the forum, and the wiki were downloaded using the urllib2 Python module, which allowed the entire website to be downloaded in html form. The html files contained all the data in raw form, from which the relevant data was extracted and placed into raw contributions tables. These tables then differed in terms of how they were structured, but in general, they kept the information as to who contributed, when they contributed, and how. The only exception to this being the IRC, which required two periods of data collection, as the technology does not allow historical records to be stored. One of the dangers of data mining is that not all the information is easily extracted, and this may limit the scope of the data collection by making the researcher focus on the data that is easy to extract (Christley and Madey, 2007).

Repository

Data from the repository was downloaded and analysed at two points in time, the first on February 2013 and the second on February 2014. The first part of the data collection was exploratory in nature and was used to familiarise and understand the processes in contributing to the project. Once the initial analysis was made, and once the semi-structured interviews added clarity to the findings, the repository was once again downloaded and more data was extracted from the html.

For the final analysis, key modules were downloaded from the official repository which can be found at the project's GitHub account[1]. The CyanogenMod GitHub account contains approximately 192 repositories for the project, most of which are device specific or have been forked[2] directly from the Android Open Source Project. CyanogenMod's official wiki, however, contains a

[1] github.com/CyanogenMod

[2] A project fork is created when the source code is copied and a new development thread is initiated with that copy.

Location	Data Points	Start	End	Variables
Repository	4,941	August 2010	February 2014	User name, date of contributions, module name, title and description of contribution.
Gerrit	26,432	July 2012	January 2014	user name, date of contribution, module name, title and description, contribution status.
Wiki	7,123	November 2012	November 2013	User name, date of page edit, page name, size of edit.
Forum	231,163	May 2010	June 2014	User name, date of post, post section, contents of message.
IRC	3,238	July 2012	August 2012	User name, date and time of post, content of message.

Table 5.2: Size and description of data sets.

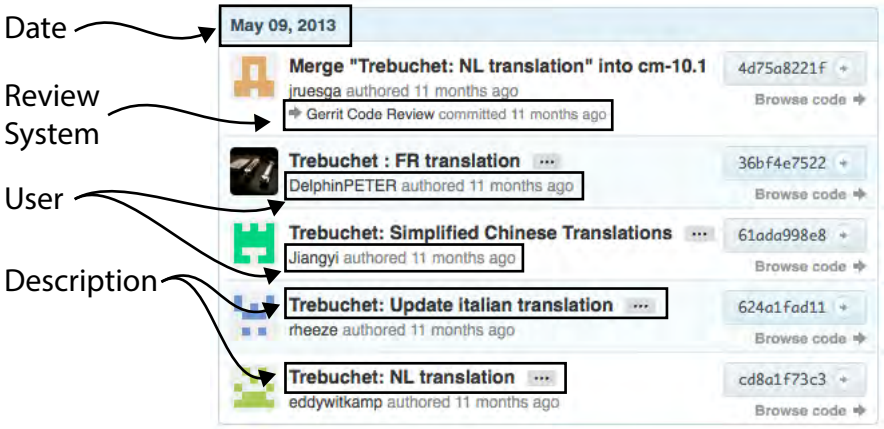


Figure 5.2: Repository Data Page

list of 13 repositories that are directly associated with the project and contain the various modifications and key features that are specific to CyanogenMod. Once the data was downloaded in html form, a Python script (Appendix A) was used to parse through the relevant data and place it into a spreadsheet.

Each contributions contains four fields of data: (1) user name of the author; (2) date of submission to the repository; (3) description of the submission, and; (4) unique identifier. Figure 5.2 shows a sample of how these contributions are listed on the repository. The user name of the author and the date were used to determine the individual who authored the contribution and when the contribution was made. Finally, the description of the submission was used to search for keywords determining the type of contributions being made.

For the participation metrics, the data was analysed only for a period of time spanning the development of CM version 10, from July 2012 until February 2014. Data was extracted from all 192 repositories in the project's GitHub account, with records of contributions dating back to early 2008. The time frame was narrowed down to the development of CM 10.0, 10.1, and 10.2, therefore focusing on the development of the latest version with established processes and rules. The data used therefore begins from the release of the Jellybean AOSP in July 2012 until February 2014, when the data was last collected.

In addition, association networks for each repository was used to further determine patterns of behaviour within the project's repository. A log file was extracted from each file in the repository, obtaining information on each change

to that file which includes the email of the contributor, the location of their contributions, and the details of their contribution. These logs were then cross referenced with the official list of contributors and project leaders found in the project's documentation in order to determine core contributors and translators. Following that, the other email addresses were cross referenced with the projects official Gerrit review system in order to determine which contributions were made directly to the project and therefore eliminate those that were not. From the resulting dataset, association networks were made in order to obtain statistics on each group of contributors and their pattern of contributions.

From the association networks, data was taken in terms of centrality, density, community clusters, and modularity. As explained by Wasserman and Faust (1997), association networks focus on membership to different groups; in this case, each file represented a group and membership was determined through contributions. A network is created when groups are linked through overlapping membership, in this case it is when an individual has made modifications to various files. From the resulting graph, centrality and density were taken as an indication of how much an individual will move between different files. This means that a high centrality and low density would imply that there is specialisation or a focus on particular files. In addition, community clusters were located and the modularity calculated. This would mean that the fewer clusters and lower modularity will mean less specialisation and more movement between files.

Gerrit

The Gerrit review is a web tool which allows the project owners to review bug reports and fixes before they are committed to the main repository. The Gerrit website, therefore, provides information as to when the bugs have been reported, who by, and whether they were then committed to the repository or not. It is this information, along with the repository data, which provides a good indication as to who are the individuals that are providing bug reports, when, how are these acted on, and who approves them. Figure 5.3 shows a sample of the bug and fix list for version 10.1.

Bug Report of Fix	Contributor	Module	ROM Version	Date
fusion3: increase mmc idle timout to 10"	Giulio Cervera	CyanogenMod/android_d...	cm-10.1	May 8, 2013
audioflinger: Fix SoundPool play block when the system playing MP3	Engle Mars	CyanogenMod/android_fr...	cm-10.1	May 8, 2013
[WiP] Settings: Performance: GPU Governor settings	Giulio Cervera	CyanogenMod/android_p...	cm-10.1	May 7, 2013
p1parts: add key led brightness control	Humberto Borba	CyanogenMod/android_d...	cm-10.1	May 7, 2013
"not enough rainbows, 1 star uninstall"	Chris Ennis	CyanogenMod/android_b...	cm-10.1	May 5, 2013

Figure 5.3: Sample of the Gerrit Data

The Gerrit Review data was gathered in January 2014. There were a total of 815 bug reports recorded, the earliest being from August 2012, and the latest being January 2014. Consistent with the repository data, all bug reports were downloaded for the main releases of version 10; version 10.0 (code named Jelly Bean), 10.1 and 10.2. The data was placed on a csv file and participation metrics were taken from it to determine how many contribute, who contributes the most, at what point of the project.

Forum

The full list of members and their contributions was downloaded from the official forum[3]. The list of members is freely accessible[4], and contains information on all of the individuals that have signed up to the forum, along with their official role within the forum, the date they joined, and the number of posts they have started. The focus of the data is on the distribution of participation and on the roles that key individuals have on the forum. Figure 5.4 shows a sample of how the list is presented in the website.

In total the list consisted of 1,232 html pages, which had 60 users listed in each page in descending order. To collect the data, a Python script was written that would look into the html code and extract the relevant information. For the overall statistics on participation and distributions, only users that contributed at least once were taken into account. For the administrative roles within the forum, the files were searched for the four different roles; Administrators, Developers,

[3]forum.cyanogenmod.com

[4]As of February 2014, the member list is no longer accessible due to the strains this was placing on the forum servers.

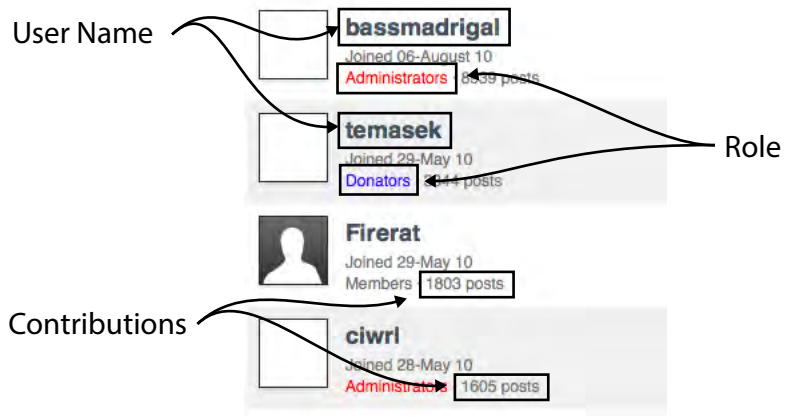


Figure 5.4: Forum List of Participants

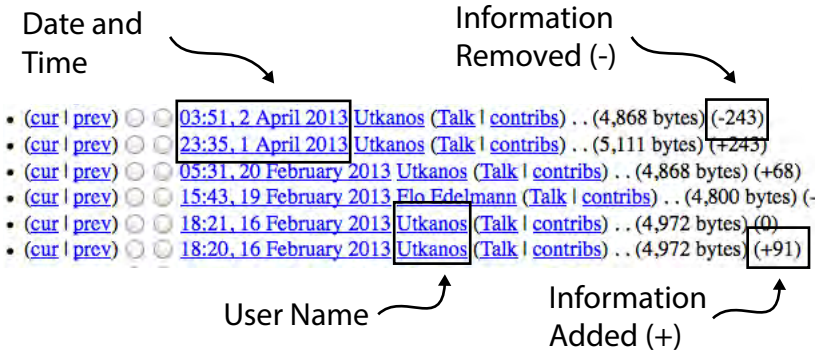


Figure 5.5: Sample of wiki data.

Donators, and Members.

Wiki

The data for the wiki was collected from the official project wiki[5] and it focused on the number of contributors and their participation pattern. The list of the contributors to the wiki was obtained by counting and compiling all the edits that were made to each page. The list that was compiled contained the date and time of the contribution and the user name and whether information was added or removed in the edit. Although negative or positive contributions were recorded for changes in information, these were not taken into account because some editing which removed information may have been updating the page by cleaning and deleting links or information that was out of date. Using these numbers as a proxy for size of contributions may therefore be misleading.

[5]wiki.cyanogenmod.com

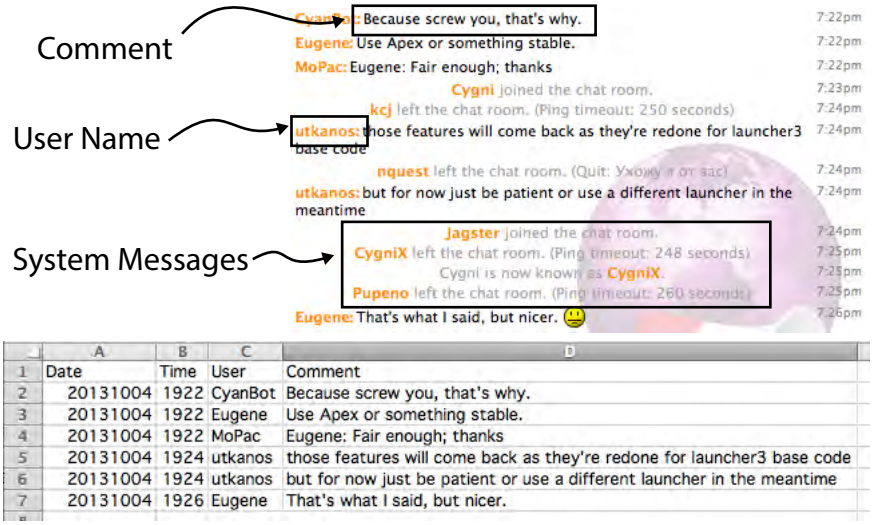
The collection of the data took place on August 2013 and then on January 2014, when all the contribution pages were downloaded into their html source. Three Python scripts were used to mine the data, the first made a full list of all the pages within the wiki, including translations pages. The second script then downloaded the editing information for each page and saved it in html form. Once all the pages were downloaded, a third script then took the data from the html, obtaining the user name and the time of each contribution and placed it in spreadsheet form.

IRC

The IRC demanded a different type of collection as all the interactions were occurring in real time and no log is kept of them. The IRC is a real time communication channel, where users can post questions and receive answers from others instantly, creating real-time problem solving and user to user assistance. The conversations are not kept on record unless a log is kept, requiring a member to sign in to the service and stay logged on for that period of time. As a result, to monitor the IRC, it was necessary to create an account, log on, and keep a record of all conversations during that period, requiring a computer to be connected to the service without interruption for the duration of the data gathering.

In order to obtain data on participation for the IRC, logs were kept of all conversations at two points in times, each lasting approximately one month. The first month was from January 29 2013 to March 2 2013, and the second period was from September 14 2013 until October 16 2013. The data contained in the IRC logs includes conversations between members, the time at which the conversations took place, the conversation itself, and system messages describing other activity. Figure 5.6 provides an example of the information.

The log files were recorded in xml syntax, requiring a separate script to read and then extract the information. The log files' xml is readable to most IRC software, in the case of this research Colloquy was used, and can be then loaded and read. As the data collected covered a period over a month, it was necessary to parse the data and convert it to a table which can then be used for statistical



	A	B	C	D
1	Date	Time	User	Comment
2	20131004	1922	CyanBot	Because screw you, that's why.
3	20131004	1922	Eugene	Use Apex or something stable.
4	20131004	1922	MoPac	Eugene: Fair enough; thanks
5	20131004	1924	utkanos	those features will come back as they're redone for launcher3 base code
6	20131004	1924	utkanos	but for now just be patient or use a different launcher in the meantime
7	20131004	1926	Eugene	That's what I said, but nicer.

Figure 5.6: IRC Sample of Data

purposes. The data parsing was done using a Python script, recording the name of the individual, the time the comment was made, and the contents of the comment. The script then transfered this to a spread sheet which made it possible to look at the data in each field. At the same time, the script also made it possible to clean the system comments, showing only the conversations that took place.

5.3 Project Documentation

The project documentation was used to answer eleven supporting question, focusing on the rules of participation, official project roles, project characteristics, and development processes. The documentation was obtained from the official project website and blog, the wiki, forum, and a number of official social media accounts in Google+, Facebook, and Twitter. The project's documentation spans a period from early 2010 until February 2014, when the data was collected.

5.3.1 Questions Addressed

The project's documentation was used to address four of the five supporting questions from the first research question on how users contribute to the project.

1.1 What licenses are used by the project?

1.2 What products and services are offered by the project?

1.3 What is the project's development process?

1.4 What ICT is used by the project?

The data used for these supporting questions was found in the project's website, wiki, and blog. The licenses used by the project were obtained through the software documentation in the repository, where the information file contained details of contributors, dates, and permissions. Data for questions 1.3 and 1.4 were easily accessible through the wiki and the main website, where sections are dedicated to informing users on how to contribute and what tools to use.

Document data was also used, in conjunction with semi-structured interviews, to determine official managerial roles and contribution rules.

2.1 What are the managerial roles within each type of contribution?

3.4 What are the rules for contributing to the project?

The project's wiki contained a comprehensive list of key managerial roles in each of the ICTs, along with a brief description of the activities that are involved. Rules on how to contribute to the project were found primarily on the wiki, but could also be found through a number of other ICTs, including forum and social media.

Finally, supported by the semi structured interviews, project documents were used to answer four of the five supporting questions regarding leadership at the project level.

4.1 What are the goals of the project?

4.2 Who are the official project leaders?

4.3 What are the roles of the project leaders?

4.5 What is the relationship between each community and project leader?

Information for these four questions was available throughout the project's ICT, particularly those pertaining to the roles of the project leaders and the goal of the project. The goals of the project were typically communicated to

users through the project's blog and social media. As with question 2.1, the wiki provides a list of the project leaders and their roles at the project level, information which was used for questions 4.3 and 4.5.

5.3.2 Dataset Description

The documentation and historical data for the thesis was taken from the official wiki, the project's social media accounts, and their blog. The documentation for some parts, especially from the blog, go as far back as August 25, 2009, when the first post about the project was made. The information is used to answer questions on who the official members are, what are the rules of participation, how are conflicts resolved, and what is the development process. The documentation consists primarily of communications from the official core members to the general public, and therefore provide news of events, such as the decision to introduce new features and future goals. Finally, the documentation has also provided information on external and internal conflicts, where these have been documented and communicated with the general public.

The documents that were used in the analysis were assessed according to validity, authenticity, representativeness, and meaning. Open source projects are typically well documented, as there is an emphasis on transparency and on conveying information to all the project contributors. This makes the availability of project documents very high, however, they still need to be assessed in terms of quality. The documents that were selected have a particular origin, in that they were written by the project leaders and were found on the project's websites. The credibility and representativeness were assessed in terms of who wrote the document and where this document was displayed, with greater emphasis given to those that were written by the official core team and published through official user accounts.

The official wiki[6] provides information on the development process, some of the rules of participation, and contains the list of official developers. The wiki has a large number of sections that are maintained, but not initiated, by the general public, with sections dedicated to the development process, the rules for

[6]wiki.cyanogenmod.org

participation, and additional information. There are two key pages that provide a list of all the modules that are modified by the project, and a breakdown of the origin on many of the ROM's features. There are a few sections on the rules of participation in terms of how to contribute and the steps that are required in order to port[7] the ROM to a different device. Finally, the wiki also contains a list of key developers and contributors[8] along with their roles in the project and their user names. It is this list that has been used to define the official core team of developers.

The blog provides updates on the development process as well as key information regarding the CyanogenMod organisation itself. The blog began in August 2009, where the project initiator, Steve Kondik, would post messages regarding any new releases or changes to the ROM. It has since been used by a number of different official core team members to communicate various things about the organisation and the project in general. The blog also features posts under the title 'This week in CyanogenMod' which serves as a digest to bring the community up to date in terms of the development project and future plans. It is from the blog that information regarding the dates of development and releases are taken, as well as some of the key conflicts that have arisen.

The social media accounts provide more informal and up to date information on the project, as well as some conversations with the general public. The project utilises three main forms of social media, Twitter, Facebook, and Google+. In the same way as the blog, the social media accounts are also used to communicate updates in the project to the general public, the only difference is that the general public are able to respond and start a conversation. Within the social media accounts, the most active is the Google+ account, where there are other communities devoted to the discussion of the project. It is through the social media account that some of the data on the conflict and dealings with other projects has been taken from.

[7]To modify the software so that it works for a set of different hardware.
[8]wiki.cyanogenmod.org/w/Devs

5.4 Semi-Structured Interviews

Semi-structured interviews were used to guide the research, add clarity and help verify the nine questions dealing primarily with rules of participation, managerial activities, and project goals. In total there were 10 project leaders interviewed through email at two points in time, first in November 2012 and the second in June 2013. The initial interviews dealt primarily with general aspects of the project such as work groups, project goals, and processes of contributing to the project. The final set of interviews were used to clarify and verify results that used participation metrics or project documentation.

5.4.1 Questions Addressed

Semi-structured interviews were used to add clarity and verify nine of the supporting research questions. The following two questions relied on both participation metrics and semi-structured interviews.

1.5 What kind of contributions are made through each ICT?

4.4 What types of contributions are made by the project leaders?

For both questions, participation data was used to look at the direct contributions made to the project by general users and project leaders. The semi-structured interviews were then used to identify other forms of contributions that would not be possible to observe, particularly those that were based on managerial and coordination activities.

In addition, semi-structured interviews were used to further validate and add clarity to six questions that used project document data.

2.1 What are the managerial roles within each type of contribution?

2.4 What are the rules that govern participation for each contribution?

4.1 What are the goals of the project?

4.2 Who are the official project leaders?

4.3 What are the roles of the project leaders?

4.5 What is the relationship between each community and the project leaders?

While the project documentation contained the relevant information for the above questions, some were either outdated or did not reflect other observations. The wiki, for instance, had not been updated since they were first published, and therefore included the names of contributors that were no longer involved in the project. For question 2.4, further detail was obtained on the informal rules of participation, as well as their enforcement. In addition, the interviews provided a better understanding of project goals from the different groups involved in the project, making it possible to validate and triangulate findings. As mentioned above, managerial and coordination actives are not typically able to be measured through participation data, therefore the interviews were used to determine questions 4.2, 4.3, and 4.5.

Finally, the semi-structured interviews were used to determine the nature of managerial roles within the project.

2.3 How are managerial roles formulated and delegated?

Interviews with three of the five core team members provided some insight as to the process of formulating roles and how these were delegated. This question was initially used to determine managerial roles within each community, but was also used to determine managerial roles at the project level.

5.4.2 Dataset Description

The semi-structured interviews provided information about the project and about participation which cannot be measured through observation or through the documentation, as well as information about the evolution and origin of the project. The project's wiki lists a total of 118 individuals as being official contributors to the project (Table 5.3). Table 5.4 provides a list of all the interviewees and their role in the project. The official project leaders (Core Team) provided information on the history of the project, the delegation of

Role	Official Contributors
Core team	5
General Developers	11
Official Translators	30
Device Maintainers	65
Wiki Editors	7

Table 5.3: Number of Official Project Contributors

work, and goals of the project. The general developers, official translators, device maintainers, and wiki editors provided additional information about the different rules and processes for contributing to the project.

In addition, the interviews also provided information such as the delegation of work and the roles carried out by official community leaders. The interview with Mauro Batzzano, for instance, focused on how users contribute through translation, including the location where they coordinate activities and what are the processes for contributing. Dean Fanning provided information on what his role was within the IRC, in setting it up and managing, as well as the handing over of the channel to the official members. Finally, interviews with Ahmet Deveci and Abhisek Devkota provided information from the early stages of the project, of how the project grew, how work was delegated, for which documentation is not available.

5.5 Analysis of Data

This section presents an explanation of how the data was analysed for the three key components of the study: (1) How people contribute to the project; (2) structure of the communities, and; (3) governance. Contributions to the project were analysed using keyword analysis of the participation metrics data, and were then verified through semi-structured interviews. Community structure was determined using Crowston and Howison's (2005) implementation of Bradford's Law of distribution, again using participation metrics data to divide the community into three groups of core, middle, and periphery. Finally, this

Project Group	User	Role
Core Team	Abhisek Devkota	Community Relations
	Jef Oliver	Developer Relations
	Prash D	User Interface Lead
General Developers	Björn Lundén	User Experience
	Dean Fanning	Infrastructure
Device maintainers	Pedro Veloso	Porting
	Jens Doll	Maintainer
	Martin Brabham	Porting.
Translation	Marco Brohet	Lead Translator
	Mauro Batzzano	Italian
Wiki	Ahmet Deveci	Wiki Editor

Table 5.4: Interviewees and their roles.

study used a modified version of Markus's (2007) governance framework to determine coordination within the project, using data from project documents and semi-structured interviews.

5.5.1 Contributions to the Project

To determine what types of contributions were made to the project, semantic analysis was used to analyse participatory data in order to identify key words that would indicate different activities. Using the NLTK (Natural Language Toolkit) library (Bird et al., 2009), a Python script (Appendix B) was written to analyse words and their meanings within sentences. The NLTK library was able to select and classify each word according to their lexical category (e.g. nouns, adjectives, pronouns, etc.). A table was then created with the number of times each word was found, and this data was then used to create a word cloud that would facilitate the identification of key activities.

The data used for the semantic analysis was from the repository and review system titles, and from the messages posted to the forum. The titles form the

Managerial		Programming		Non Programming	
1,661	merge	526	fix	1,887	translations
		414	update	950	translation
		228	fixing	77	Chinese
		198	added	77	Italian
		156	reset	59	French
		142	bug		
		91	updated		
		88	cleanup		
		64	code		
		61	preferences		

Table 5.5: Relevant Words for each Category. Some of the top 50 keywords and the number of times they were used to describe the contribution.

repository and the review system contained a short description of the changes being made to the repository. Using this data, it was possible to identify categories for the type of contribution to the repository, some of which can be seen in Table 5.5. From the forum discussions, it was possible to distinguish the subject of the discussion in terms of whether it was discussing software or hardware issues, a new feature, or a bug.

After the initial analysis, both sets of results were further validated through interviews with project leaders or through project documentation. With regards to the types of contributions made to the project, the results obtained from the semantic analysis were validated by the interviewees, who were able to provide further information on other activities that were not observed. For the forum data, the project documentation, which included rules of participation, were referenced in order to determine the validity and accuracy of the results.

5.5.2 Community Structure

The structure of the community was determined using participatory data and analysing it using the Bradford's Law method developed by Crowston et al. (2006). A list of users and a number of posts was generated, and their post percentage was calculated in respects to the rest of the group. The cumulative sum of the top posters would then be calculated until 33% was reached, and this

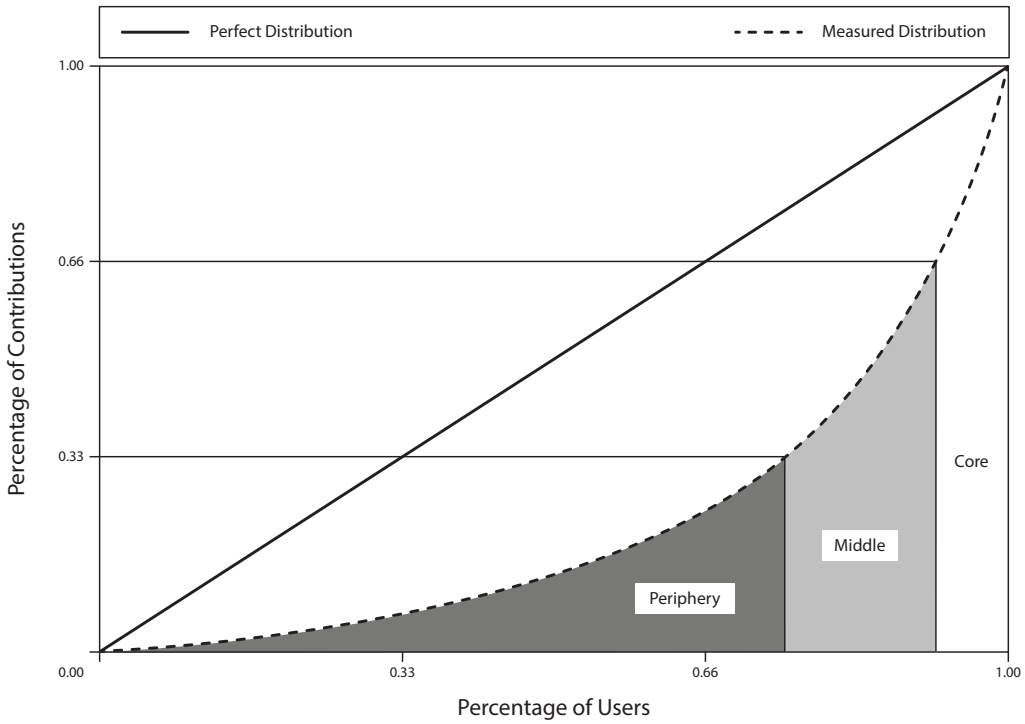


Figure 5.7: Community Structure through participation distribution.

would become the first layer, the Core. The process was then repeated for those who were between 33% and 66% to become the middle, and between 66% and 100% to become the periphery (Bradford, 1985; Black, 2004; Crowston et al., 2006). Figure 5.7 shows how the size of each layer in terms of number of users is calculated.

Using Bradford's law to create a trilayer helps to divide the group into three key subgroups, the core, the middle, and the periphery, which then has a different type of role within the project. Crowston et al. (2006) analyses three methods of determining which project members belong to either the core or the periphery, from the three methods used, Bradford's Law was one that was seen to be more accurate in determining the core group if groups 1 and 2 are classified as such. One of the setbacks is that it does not take into account interactions between individuals, which can then determine how important individuals are within the project. The Bradford law of distribution, although not perfect, is also consistent with the concept of cosmopolitan players as expressed by Dahlander and Frederiksen (2011), who in this case can be considered to be those who belong to the middle group.

Dimensions	Description
Community Rules	Rules that govern membership and interactions between members.
Development Rules	The rules that govern behaviour specifically related to contributions to the project.
Rules about Rules	Determining how rules are established and by whom.
Tools and Information	What tools are required for interaction and contributions.
Social Structure	The structure of the community and the roles that each group of contributors have.

Table 5.6: Governance for Coordination Framework. Adapted from Markus (2007).

5.5.3 Governance and Coordination

The framework for determining governance in the CyanogenMod project was taken from Markus (2007) and Crowston and Howison (2005). As mentioned previously in Chapter 3, this study focuses on governance as a means to solve coordination problems. This study will therefore focus on five dimensions that are important to coordination: (1) community rules; (2) development rules; (3) rules about rules; (4) tools and information, and; (5) social structure. Table 5.6 contains a summary with a brief description.

The community rules and the development rules focus on the way in which coordination can be achieved through setting rules that shape behaviour within the project. Community rules can be described as those that focus on how users interact with each other, what subjects can be spoken about, and what technologies to use in order to interact with other users. Development rules, on the other hand, focus on how users can contribute directly to the project, which takes into account the technology that needs to be used and the procedures that are in place. For this study, development rules were extended to focus not just on the rules that determine how users contribute through programming, but also through a number of different activities.

In addition to community and development rules, another dimension is that of rules about rules, which determine how rules are formulated, by whom, and to

what end. For this study, the rules about rules determine the processes through which new rules are formulated. In addition it also includes the individuals that are involved in creating those rules, what their roles are within the project and what is the breadth of influence of these rules.

Finally, the social rules of the community and the project are also considered as a form of coordination mechanisms, relying on the centralisation of decision-making and the interdependence between layers. Determining the structure of the community is important as a coordination mechanism in that it shows that the relatively small group of individuals with authority are responsible for making decisions. In addition, this also includes the interdependency between the different layers regarding new ideas, where the periphery are more likely to have new ideas and the core are better able to implement them.

5.6 Limitations and Ethics

As with most case study approaches, this research may be high in validity, in terms of the depth of analysis and triangulation in data sources, but it may not be able to be generalised to a wider range of open source projects. At the same time, the sample chosen for interviews will provide only the point of view of the official core team and primarily those that are officially involved with the project. As a result, it does not take into account the perceptions and the day to day realities faced by the majority of the project's contributors.

In terms of ethics, there is no evident ethical problem that may be directly applicable to this study, as there will be no personal information taken from individuals without their consent that is not already available online. The focus of the study is to find the patterns and metrics of participation, which focuses on aggregated data rather than on analysing the behaviour of individual contributors. Where individuals are mentioned, the relevant information has been displayed and at times given by them specifically for this study.

5.6.1 Case Study Approach

One of the major limitations of this study is the case-study approach, which may mean that the generalisability of the findings will be limited. As Yin (1989) points out, a case study is not synonymous to a sample, and therefore the findings will not necessarily apply to other cases. However, it is also important to understand that the strength of the case study is in helping generate theories. The literature on open source communities has therefore relied heavily on the single case study approach in its early stages so as to be able to generate or expand a number of theories in this field. As Stuermer et al. (2009) points out, when studying open source communities, the single case study approach can only generalise the findings to theory but not to other projects. The limitations of generalisability, however, are not an issues for this research as the focus is on understanding a unique case that can present a new empirical setting for the study of open source development and collective user innovation.

5.6.2 Emailed Interviews

The main limitations of the semi structured interviews revolved around the method through which these were carried out, making the process lengthy and potentially affecting the relationship between the interviewer and interviewee. The email interviews were chosen due to their practicality in terms of accessing the relevant interviewees who may be geographically remote, and due to limited resources for the study. The interviews were carried out via email, and the main problem that was expected from the beginning was the loss of interest from the interviewee (see Bampton and Cowton, 2002; Ross, 2007). The loss of interest was experienced in two occasions, where two participants confirmed that they would like to take part, but did not reply to following emails.

The first major limitation of the semi structured emailed interviews was the lack of relationship building, which could have helped in obtaining a greater depth of information and retaining interest among the interviewee. The interviews were carried out after an initial analysis of the repository data, and this first batch of interviews helped clarify most of the processes and rules regarding development.

The information gathered in the interviews was used to guide a second round of data scraping, which meant that there would be a period of no contact with the interviewees. After the second round of data scraping, some of the individuals that were contacted before, were no longer responding to emails. The period of no contact between the first and second collection of repository data made it difficult to re-establish a connection with the interviewees.

In addition to the lack of relationship building, the rapidly changing nature of the project from 2012 to 2013 also made it difficult to maintain interest among those being interviewed. The CyanogenMod project went through major changes during that period, where two interviewees left the project, and where the project itself went through a period of preparing for commercialisation. The project went commercial towards the end of 2013, but there was a period just before that, starting from October 2013, where the official project members were not able to be reached for further information. Once the project went commercial in December 2013, questions about the project began to be diverted through official means of communication.

5.6.3 Ethical Considerations

There are three major ethical issues that are dealt with, the gathering and use of observational data, hacking and piracy, and full disclosure. In terms of the observational data, the main focus of the data is on observing metrics and patterns rather than individual behaviour. Most importantly, the information that was gathered is freely available and accessible to the general public, in most cases this information is made available by the individuals themselves. Ethical consideration may have been needed if data was being taken from specific individuals about their behaviour or personal information, bringing the question of whether informed consent is needed (de Vaus, 2001).

Another consideration is the possibility of illegal activity taking place, primarily in terms of piracy. The subject of hacking and modifying in Android has included some problems in terms of hacking and pirating application software. While the Android OS is open source, and its licence allows the modification and re-distribution, there may be some activity in the community

that may include application software which are not free. CyanogenMod, as an organisation, has made an effort to ban or to stop anything like this from taking place in their community, clearly stating that there should be no discussions on illegal activities. At the same time, the focus of this research is on the official project, which is still open source and available to everyone.

Finally, when it comes to interviews, there was full disclosure and informed consent. The respondents were made aware of the nature of the research and what is expected from the interviews. In the initial contact it was also made clear that their information will be used only for the study and no other party will have access to their details without their consent.

5.7 Chapter Summary

There are three sources of data in this study: (1) Participation metrics; (2) Project Documentation, and; (3) semi-structured interviews. Participation metrics were collected from the project's ICT; the forum, the wiki, the repository, and the review system. The data covers all contributions made to the project, with data on the user names, time of contribution, and a description of the contributions. The project's documentation covers the rules of participation, development processes and the history of the project. The semi-structured interviews were used to triangulate and validate findings and to further provide information on activities and processes that were not observable.

Semantic analysis was used to determine the types of contributions made to the project, where the words used to describe each submission were categorised to define types of contributions. To determine community structure, this study uses Bradford's Law, clustering contributors into three layers; core, middle, and periphery. To determine governance, the study uses an adapted framework which focuses on five key dimensions for coordination: (1) Community rules, (2) development rules, (3) rules about rules, (4) tools and information, (5) and social structure.

The major limitation of this study is its generalisability, in that the findings

may be specific to the project being studied. In addition, the semi-structured interviews focus on the opinion of project leaders, which may represent a biased view of the project, particularly in terms of objectives and processes. The primary ethical issue of this study related to the use of participatory data, which was collected from publicly accessible ICT without the consent of each contributor. This issue, however, is mitigated by not using the real names of the contributors or revealing any other personal information.

Chapter 6

The CyanogenMod Project

6.1 Introduction

This chapter presents an outline of the CyanogenMod (CM) project, looking at its relationship with Android, development process, project organisation, and community participation. CM is a highly centralised and hierarchical project which uses community input at different parts of the development cycle. Community participation changes location and type of input through the process, from a development-focus, to bug fixing, and then to documentation. New members are added to the community after they have successfully ported CM to a new device. A new member will usually bring with them other developers who worked with them to port it to the device.

Google bought Android in 2005 with the idea of introducing a smartphone operating system that would compete with the then popular Blackberry. Because it uses many open technologies, such as the Linux kernel and the Java programming language, Google and their partners have been able to modify the operating system to suit their particular needs. Through this optimisation, device manufacturers and phone carriers have been able to differentiate their products and push their own services to the consumer. It is therefore evident that the open nature of the platform has added to its increase in adoption and network externalities.

The heavy customisation has also lead the user community to modify Android, therefore removing most of the restrictions and services that were

added. The open source nature of the operating system not only allows commercial entities to modify the operating system, but has also made it possible for users to do the same, as there is wide availability of documentation regarding the process. The user communities modifying the Android experience focus on removing modifications made by device manufacturers and carriers and on improving reliability and efficiency. These modifications are then redistributed through on-line user communities, such as CyanogenMod, MIUI, and Replicant.

These modifying communities share many of the same characteristics of open source communities, where individuals share software code and free reveal their modifications. As with many of the open source communities, the CyanogenMod consists of developers who volunteer their time to maintain and add features to the project. The core team of developers carry out the traditional open source tasks such as bug fixing, testing, and implementing new features, while also relying on the user community for bug reporting and project expansion.

This chapter is structured as follows. First we will look at a brief history of Android and CyanogenMod's relationship with it. Then we will look at three main characteristics of the CM project, the development process, its governance, and finally community participation.

6.2 Android Operating System

Google bought the smartphone operating system Android in 2005. In 2007, it joined forces with device manufacturers, carriers, and software companies to form the Open Handset Alliance (OHA), to which it handed Android's development while still retaining the lead role. The aim of the OHA is to promote Android as an open smartphone platform in an effort to decrease the industry's reliance on proprietary platforms. To actively maintain and distribute Android, as well as to encourage participation from the public, Google established the Android Open Source Project (AOSP). It is through the AOSP that Google distributes the official and unmodified operating system – commonly referred to as the AOSP version.

Android is the most popular smartphone operating system, commanding over 50% of market share (Pettey, 2011). The operating system is made up of both open source and closed source technologies, drawing from two main open source projects, Linux and Java, which allows it to be customised to suit the needs of Google and its partners. The open source nature of most of its technology has also allowed users to modify it and distribute it within the user community. Its flexibility and highly customisable nature has allowed it to work on other devices, such as games consoles, smart TVs, and tablets.

This section will first look at the technical aspect of Android, looking specifically at the technologies that are involved and the steps needed for it to work on different devices. It will also look at the extensive modification Google has made to Linux and Java so that it suits the requirements for more efficient energy consumption. Then we will look at modifications made by device manufacturers and carriers, and what effect this has on the overall experience of the operating system. Finally, the section will look at how users modify Android so as to remove some of the commercial modifications and how these user modifications have evolved into collaborative projects, of which CyanogenMod is the most popular one.

6.2.1 Structure of the Operating System

The Android operating system is a collection of closed and open source software that together form a device-specific software platform. It is composed of two major parts, the system software and the application software. The system software provides all the instructions to the hardware devices – such as screen, camera, and speakers –, and is responsible for the operation and functioning of the device (Beck, 1985). The application software, on the other hand, is concerned with particular processes and applications – such as music player, email client, and web browser –, using the device as tool rather than controlling it directly.

The majority of the technologies used in the system software are open source, such as the Linux embedded kernel and Dalvik, Google's own Java implementation. One of the reasons why open source was chosen over

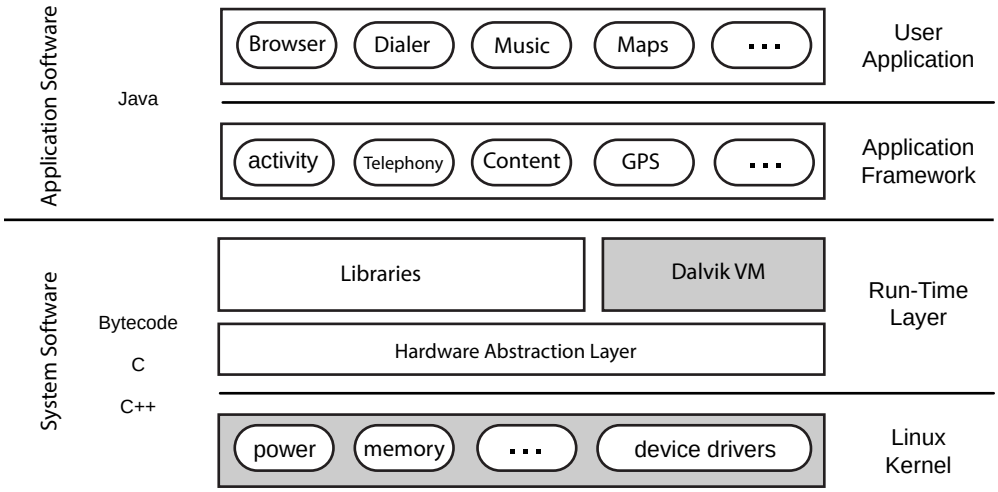


Figure 6.1: Android Operating System structure.

proprietary software was its wide usage and familiarity with the development community. Google have cited the availability of resources and large pool of community support as the deciding factors in choosing Linux and Java (Cleron, 2007). Their open source licences allow Google and its partners to extensively modify the Android operating system so as to suit the particular requirements of mobile devices in terms of resource management and services. The modifications made by Google and its partners to the Linux kernel and to Java, however, have not been ported over to the main open source projects, therefore making Android a separate branch from those projects.

Linux Kernel

Android's Linux kernel can be divided into two important parts, the standard kernel functions and the device drivers, both of which are written in C or C++ programming language. Android uses an extensively modified embedded Linux kernel to perform a number of tasks, such as memory, communication, and process management, as well as providing a file system and hardware manager. Linux was chosen as the kernel because its open source licence allows Google to modify the kernel for Android's needs and at the same time take advantage of the stable and established features, such as memory management, networking, and driver architecture, found in the main Linux project (Cleron, 2007). Google has made the Linux kernel more efficient by improving the kernel's power

management capabilities, which are of particular concern for mobile devices. However, the particular Android modules that deal with power management, called wakelocks, have not been added to the original Linux project[1], effectively making it a separate branch from the original Linux kernel.

The second important component is the device driver system dedicated to hardware management, which acts as a direct link sending instructions from the operating system to the hardware components, such as camera, speakers, and screen. The device drivers make up almost 70% of the Linux operating system, allowing it to work on almost every type of PC hardware (Kadav and Swift, 2012). However, because of memory constraints in mobile devices the majority of these drivers are removed to create an embedded Linux, where only the drivers that are essential for that device are included. Each manufacturer must therefore write the hardware drivers when installing Android on any new device, and must then release these drivers through a similar open source licence.

Application Software

There are two types of application software, the framework and the user applications. The framework applications are the set of instructions that run in the background, such as the notification manager, location manager, and content provider, that carry out functions that user applications can access (Cleron, 2007). The notification manager, for example, is used to send information to the user about events on an application, such as receiving a new message, and can take the form of a pop-up screen or vibrating alert. The location manager gathers data on the user's geographical position, allowing user applications to access the data and use it for functionalities such as determining the user's location in a map. Finally, the content provider allows user applications to access telephone information such as contact details and phone history. The application framework is mainly open source, allowing app developers to make full use of the framework and customise it to their own needs.

[1]The Linux project has strict rules on how contributions are submitted. It was found that the Android wakelocks patches did not have the right documentation, therefore leading to its rejection (Proffitt, 2010).

The final application software layer is the user applications (most commonly known as apps), which includes the phone dialer, a web browser, and a music player. The key ones provided by Google are the email, calendar, messages, and video, and application manager (also known as Google Market, or Play) which all connect to Google services, which are commonly referred to as Google Apps. These apps are closed source and protected by licences which prohibit their distribution without authorisation from Google.

Dalvik

The Dalvik Virtual Machine is an important part of the Android operating system because it allows developers to use the open source Java programming language to create mobile apps, and it allows Google to modify its implementation to suit its needs. Dan Bornstein, creator of the Dalvik VM, states that there were two main reasons why Google chose Java, first because there is a wide availability of tools, resources, and documentation, making it easier for developers to learn and receive support (Bornstein, 2008). A more important reason, however, was the ability for Google to modify Java's existing open source technology for mobile devices, instead of having to build one completely (Bornstein, 2009). The primary objective of the Dalvik VM is to improve and streamline the implementation of Java bytecode on mobile devices. By reducing the size and the structure of the Java bytecode, the Dalvik virtual machine can run the same set of instructions more efficiently, therefore reducing energy consumption.

Applications written in Java are changed to bytecode instructions and then translated to assembly language through the Java virtual machine (JVM). To write a standard Java program, the programmer writes the source code on a *.java* file, and then changes it to bytecode by compiling the source code using the Java compiler program, creating a *.jar* executable file. When the user starts the application, the JVM reads the bytecode instructions in the *.jar* file and translates them to assembly language, passing those instructions to the kernel and device drivers. Using the JVM means that the application software needs only to be written once, and can then run on any hardware and software platform provided a JVM has been installed.

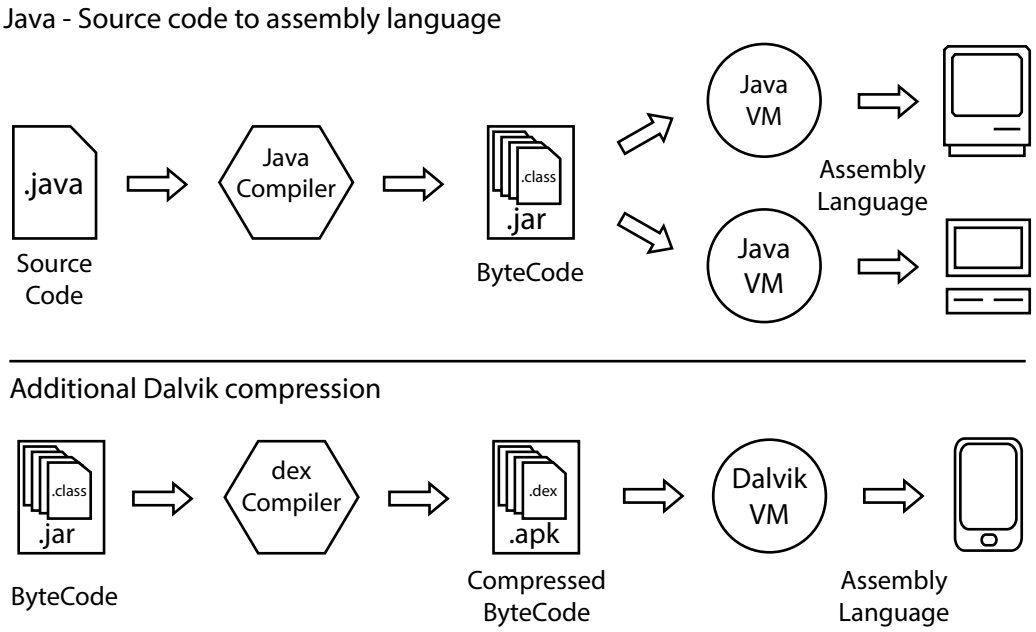


Figure 6.2: Java implementation and Dalvik compression.

Google were able to reduce the size and the efficiency of the *.jar* executable file by further optimising and compressing the bytecode. This is done by passing the *.jar* file through a second compiler, the dex compiler, which rewrites the bytecode in a more efficient way, producing a *.dex*[2] file. The Dalvik VM then interprets the *.dex* file and passes the instructions to the operating system as assembly language. Using the official development kit provided by the AOSP, a programmer will write the program source code and use the Android compiler to automatically create both the Java and the Dalvik bytecode. While on a PC the main benefit of running a JVM is platform independence, in Android the main benefit of Dalvik is in the compression and increased efficiency of the Dalvik bytecode while retaining the use of Java as the programming language.

6.2.2 Commercial and User Modification

The Android operating system is distributed by Google through the Android Open Source Project, this original unmodified version is called the AOSP. The end user, however, will most likely not come in contact with the AOSP version and will

[2]Dalvik Executable

instead use one that has been heavily modified[3]. Device manufacturers modify the AOSP by installing device specific drivers and changing the user interface, so that their products offer a unique experience from their competitors. The mobile network operators (commonly referred to as carriers) will also modify the AOSP by adding restrictions aimed at stopping user modification, and by including application software for their services, such as live streaming and messaging.

The modifications made by the device manufacturers and the carriers mean that the user will not come in contact with the official AOSP version, leading to a fragmentation in user experience (UX). At the same time, the commercial modifications lead to a decrease in hardware performance and functionality, as commercial parties look to add or remove important default features. As a result, Android user communities have worked collaboratively to replace the commercially modified operating system with a user-modified AOSP, focusing on lowering user restrictions and improving performance. While there are a number of user communities carrying out such AOSP modification projects, the most popular one to date is the CyanogenMod project, with more than a million users world wide.

Manufacturers

The AOSP version rarely reaches the end user, as modification is often required for Android to work with each device. There are two types of modifications that are made by the device manufacturers, the first is in the system software and the second in the user experience. To operate the hardware, the operating system must have the correct driver for each component, meaning that device manufacturers must modify the Linux kernel to incorporate the drivers. Each product line may have a different set of hardware components, and therefore the manufacturer will have to have a modified Linux kernel for each product line. The AOSP licence requires the manufacturers to release the source code for each driver.

Finally, the manufacturers also extensively change the graphic user interface

[3]Google partners with a manufacturer every year to release a smartphone with the latest unmodified Android operating system, called the Nexus range. These phones are released yearly with every Android release and are free of all manufacturer and carrier restrictions.

(GUI) so as to create a unique manufacturer-specific experience. Through the GUI, each manufacturer can differentiate itself from their competitors by providing a number of features. The modified GUI come as a collection of applications and services that have either been added or modified within the application software layer. Samsung's Android GUI (called TouchWiz) adds more native apps to the basic install, as well as a larger number of virtual screens on default (Bohn, 2011). HTC's Sense also contains a number of unique default applications and adds greater integration to social networking sites. All of these applications that change the user experience are closed source and are sometimes used by the manufacturers on other non-Android platforms.

Carriers

There are several ways in which the carriers can change the Android experience, from the application software to the system software. The system software can be changed by the carriers so that it can prevent users from modifying or changing its contents. The American phone carrier, Verizon, installs a software in the kernel that will stop the phone from working if it detects any user modifications. In a similar way, T-Mobile has modified Android so that it re-installs the official T-Mobile and manufacturer version automatically if it notices any unauthorised changes (Whitwam, 2010). The main focus of the system software modification by the carriers is to stop the user from modifying or replacing the pre-installed operating system.

The majority of the modifications made by carriers, however, lie within the application software layer, where apps and services are added or removed. The American carrier Sprint, for instance, includes a number of non-standard applications in their Android range, such as Sprint Navigation, a dedicated Sprint Football and NASCAR app, and Sprint TV. These apps come as standard in all Sprint products and cannot be removed. At the same time, Sprint has modified the software libraries so that all Sprint apps utilise only their mobile network for data transfer rather than wi-fi (Whitwam, 2010). Verizon, on the other hand, has removed the default Google search and mail services, and replaced them with their Microsoft counterpart. Finally, British

carrier O2 includes the Priority Tickets and Priority Moments apps as standard in their Android phones, which are used to push retail offers and event tickets to customers directly to all Android devices.

6.2.3 User modification

The manufacturer and carrier modifications to the AOSP version may result in a decrease in hardware performance, restriction of services, and inconsistent experience across devices. To remove these inconsistencies and restrictions, users have reverted back to the AOSP version and modified it to make it work for their device – a process called Porting. The commercially modified version of Android is then removed from the device and replaced by the user-modified AOSP. These modifications are carried out by a number of user groups and are distributed as a complete package, sometimes referred to as ROMs[4].

The first step to modifying an Android device is to *root* it, therefore removing all reading and writing restrictions. In UNIX-based systems, like Linux, the user has complete control of the operating system when they have reading and writing privileges for the root folder of the directory. These privileges allow the user to access and change every file in the system, giving them also the ability to remove or replace the operating system itself. After gaining such access privileges, Android users are then able to remove the producer-modified Android and change it with a user-modified AOSP ROM.

There are a number of ways in which users can modify the AOSP, at both the system and application level, although the most common types of modifications take place at the application level. At the system software level, the user can modify various aspects of the Linux kernel, such as memory and power management. Some of the most common modifications include over-clocking[5] the central processing unit, changing memory allocation, and improving battery life. All these modifications require knowledge of C and

[4]The operating system is also commonly knows as a ROM (read only memory), and it alludes to traditional mobile phones that had the entire operating system coded into their read only memory. Smartphones, on the other hand, resemble PCs in that the operating system is not in the ROM but in active memory, therefore is able to be modified or replaced.

[5]Over-clocking is increasing the clock speed past its default setting, allowing the CPU to carry out a larger number of calculations.

C++, access to the Android Linux kernel, and to the device drivers.

The most common form of user modifications takes place in the application software layers, where individuals make changes to the GUI or its functionalities. Users can modify the GUI by changing the themes, animations, the way messages are displayed, and the functions of the desktop. Another form of application software modification is the replacement of standard apps and services such as email, music, and camera apps with either custom made or other open source alternatives. To modify this layer, users will have to know about Java programming, and the application framework, both of which are very well documented on the AOSP website.

After users have modified the AOSP, they can repackage the operating system and share it with other users, allowing them to contribute their own modifications, thus creating collaborative projects and communities. However, the ROM will only work on the device it was written for, so any new contributor will have to modify the Linux kernel to include drivers for their specific device before they can contribute to the ROM project. Some of the user-modified AOSP projects include MIUI, Replicant, and AOKP. To date, the most popular project is CyanogenMod (CM), which has more that two million users world-wide[6].

6.3 CyanogenMod

Besides bringing consistency across all devices and removing restrictions, one of the main objectives of the CM project is to increase software stability and performance. The project began in 2008 for the HTC Dream smartphone, and was named after Steve Kondik's user name in the xda-developers forum, *cyanogen*. Kondik shared his modification with other forum members, who then started contributing to the project and porting it to other devices. Because the CM project is based on the AOSP version, every time Google releases a new AOSP, the core development team abandons active development of the old version, and start working on the new one; the current project is version 10.0,

[6] `stats.cyanogenmod.com/`

based on Android 5.0[7].

Development of the CM project started on the XDA developer's forum, where Kondik regularly posted his modifications and other users would then contribute with their own suggestions or code modifications. As the project grew in size and complexity, a core team of contributors established processes and structures to cope with the growing user contributions. The CM project now has an official website and wiki, where extensive information on the project can be found, as well as an official forum and internet relay chat channels where users can interact and exchange information. The core team have also formalised the procedures for individuals to contribute, by setting up a distributed revision control system, as well as frequently releasing experimental and testing builds. The CM project currently supports 44 devices, and relies on the user community to extend support by porting to newer devices[8].

The CM community consists of 39 official contributors, known as the core team, and a large user community of more than a million individuals world-wide. Each group has a different role in the project, where the core team are responsible for implementing new features, bug fixing, and maintaining the core project, while the user community is responsible for porting CM to new devices, testing, and bug reporting. Community members can contribute to the project by programing, documentation, and user-to-user assistance.

This section is structured as follows. First we will look at the development process, how the CM community modifies the AOSP and make it available for a number of devices. Then we will look at the structure of the CM community and the roles that each group plays in the project. Finally, we will look at the ways in which all community members can contribute to the project, specifically looking at the different tasks that are carried out through the different areas of

[7]Google names each version of Android in two different ways, through numbers, and through names, so that Android version 5.0 is called Jelly Bean, and 4.0 is called Ice Cream Sandwich. The names themselves do not always follow the numerical system, as a result Android 2.3 and 2.4 share the Ginger Bread name. The names follow alphabetical order, So that versions 2.3, 3.0, and 4.0 are called Ginger Bread, Honeycomb, and Ice Cream Sandwich. CyanogenMod bases their numbers on the names, so that Ginger Bread in CM 7, Honeycomb is CM 8, and Ice Cream Sandwich is CM 9.

[8]The CyanogenMod project makes a distinction between official and unofficial ports. CyanogenMod state that a ROM is only official if it uses the correct tools for maintenance and bug fixing. Doing so will assure that the correct procedures for peer review have been followed and therefore it meets the quality standards.

Release Date	Version Number	Version Name	CyanogenMod Version
09/2008	1.0	Astro	1
02/2009	1.1	Bender	2
04/2009	1.5	Cupcake	3
09/2009	1.6	Donut	4
10/2009	2.0	Eclair	5
05/2010	2.2	Froyo	6
12/2010	2.3	Gingerbread	7
02/2011	3.0	Honeycomb	8
10/2011	4.0	Ice Cream Sandwich	9
07/2012	4.1	Jelly Bean	10
09/2013	4.4	Kit Kat	11

Table 6.1: Android and CyanogenMod version history.

the project.

6.3.1 Development Process

As soon as Google make a new AOSP available to the public, the CM core team begin the development process by creating a master repository for the new kernel. Each project goes through three release stages, each with a different programming focus: the Nightly Builds for experimental development, the Release Candidate for bug fixing, and the Stable Release for user support. The development process uses the Git[9] distributed revision control system (DRCS), and the master repository[10], along with its branches, is hosted on the GitHub website. Modifications that are central to the CM project are merged directly to the master repository, while device-specific modifications remain in separate branches. Throughout the process, there are three programming languages used, C and C++ for system software, and Java for application software.

The CyanogenMod project hosts their master repository in the GitHub website. Consistent with other projects using the Git DRCS, developers must make a local copy of the master repository (called a branch), which becomes

[9]They Git system was developed by Linus Torvalds for managing Linux development, focusing specifically on distributed development.

[10]Torvalds (2007) explains that unlike other control version systems (CVS) that rely on a master-slave relationship between repositories, Git resembles a network were all repositories have equal status. Developers can share code by synchronising their own repositories with others in their network. In practice, however, as is the case with Linux, most developers synchronise their projects with Linus Torvalds' repository, making it the de-facto master copy. The same can be said for the CyanogenMod project, where Steve Kondik's repository has become the de-factor master repository.

their working directory and provides them the freedom to experiment with new features without disrupting others' work. Once the changes have been finalised, the developer alerts the maintainer of the master repository and others in their network, who will then either reject or accept those changes. If the changes are accepted, the developers will synchronise their own repository with the branch, and will then spend time reviewing the code to make sure there are no conflicts[11]. The Git DRCS works by synchronising only the files that have been modified, therefore leaving the rest of the project intact.

The development cycle is similar to that of traditional software development; starting with a design and implementation period, then a release candidate, and finally the official release. The first stage of the process is to modify the AOSP and release the modifications as often as possible. This is done through nightly builds[12], where the focus is on fixing major kernel bugs and implementing features that are not device-specific, such as power management, memory allocation, and user interface customisation. The nightly builds are unsupported, meaning that there is no community support or guarantee they will work properly. Some of these features would have already been used in the previous version, in which case smaller modifications are made so that these still work in the new version. This initial development stage will carry on until all major bugs have been fixed and the build reaches a minimum level of stability, after which the core team freezes development and distributes a Release Candidate for testing purposes, allowing further time for submitting bug fixes, device support, and translations.

The release candidate (RC) is a near complete stable version of CM ROM, which developers can use to test the new features, add device-specific support, and translate to other languages. As with traditional software development processes, CyanogenMod uses a RC to give device maintainers and developers early access to the latest CM ROM, so that their software or device is supported before the release of the final stable ROM. It also gives the opportunity to fine tune the ROM through testing, minor bug fixing, and

[11]During synchronisation, if two developers have modified the same line of code, Git will notify the user about this conflict and will allow them to decide which modification to use.

[12]Nightly builds are informal release stages that show the present stage of development. Since the source code needs to be compiled to machine language, these builds are compiled once a day when development has finished.

translating the user interface to other languages. Since each developer works on their own repository, the device-specific modifications are not merged with the master repository, but instead each branch carries its own evolutionary path throughout the rest of the development process including the final release stage. Once the final modifications have been made, and the RC is deemed stable enough for daily use, each branch releases its own official CM stable version.

The third and final stage of the process is the release of the stable version, where all experimental and preliminary maintenance work has been carried out. The CyanogenMod project describes the stable version as the only one they would recommend for daily use, without fear of losing any data. Even at this final stage, bug reporting and bug fixing will continue, with built-in features in the ROM allowing non-programmers to directly send bug reports to the CM team. In addition, if a new device is not officially supported, users are encouraged to use the stable version for porting CM to untested devices. All the official builds – from Nightly to Stable – are available for downloading at the official CM distribution website[13].

6.3.2 The CM Community

The CM community is defined as the group of individuals who voluntarily contribute to the official CM project. The CM community can be divided into two major groups, the core team, who are named as official contributors to the project, and the general users. Community members can contribute to the project in three main ways, through programming, documentation, and support.

In terms of programming, the core team focuses on implementing new features in the master repository, bug fixing, and maintaining officially supported devices. The user community focuses on porting the project to new devices and reporting bugs. Documentation, which includes Q&A, how-tos, and general information, is provided mainly by the user community, while the core team are responsible for editing and managing the data, as well as writing newsletters for the community. Finally, most of the support is carried out by the user community in the form of

[13] `http://get.cm/`

	Programming	Documentation	Support
Core Team	New Features Bug Fixing Device Maintenance	Editing Wiki Blog Entries	Moderating Wiki and IRC
User Community	Porting to new devices Bug reporting	Writing wiki articles	User-to-user assistance

Table 6.2: CM Community activities

user-to-user assistance, while the core team are responsible for making sure that the rules and regulations are adhered to.

Programming

Programming is the primary form of participation in the project and is carried out by both the core team and the user community. The core programming team, consisting of the project owner, general developers and the maintainers, carry out tasks focused on the development of the central copy of the project, such as design, development, and bug fixing. The user community contributes to the project by carrying out tasks such as porting the CM ROM to new devices and bug reporting, working primarily on forked copies of the project and focusing on expanding the number of devices supported by the project.

The core team has two main roles in the project, to add new features to the master source and to maintain official branches for supported devices. The new features introduced to the project are mainly ones that are device independent, such as improved memory allocation, better security settings, an alternative user interface, and improved memory management[14]. Finally, the core team is also responsible for fixing and maintaining the repositories for officially supported devices. This involves synchronising the device repository with the master one, making sure all conflicts are resolved and all new modifications are working on supporting devices.

The user community carries out several programming tasks, which are testing, bug reporting, and porting the CM ROM to new devices, therefore focusing most of their contribution on device-specific software development.

[14]A more detailed list of CyanogenMod features can be found at `http://goo.gl/Aa2iC`.

The user community are the ones who are more likely to port the CM ROM to new devices[15]. Once the core team have implemented new features and release a stable version, the user community will test these features and report any bugs. If a user's device is not supported, the user can fork the project and modify the kernel, thus creating a device repository from the master. After a user has successfully ported the CM ROM to a new device, they can apply to become a member of the core team, which will then make it more likely that their contributions, in terms of new device-independent features, will be merged directly to the master repository.

Documentation

Documentation can be described as non-programming contributions that aim to inform users about the project or the software. In the CyanogenMod project, there are two main types of documentation, those that provide information on the use of the ROM, and those that provide information about the development process. The roles in documentation can be divided into two, content creation and content management, where most of the content creation is done by the user community, and the content management by the core team. The official CM documentation can be found in two locations, the wiki[16] and the blog[17].

The wiki focuses on information about the use of the ROM and supported devices. The aim of the wiki is to provide general information to users on how to install and run CM on different devices. To do so, the wiki provides step by step tutorials on how to root the device and install the CM ROM, as well as information about modifying and programming the AOSP. The wiki also contains an FAQ section which answers general questions about the process of rooting and modifying ROMs. The majority of the content is contributed by the user community, who are encouraged to write device specific articles, tutorials, and general information. It is then the role of the core team to manage the wiki by editing the articles and listing a number of articles that are needed and their

[15] `http://www.cyanogenmod.com/blog/whats-up-with-cm`
[16] `wiki.cyanogenmod.com`
[17] `www.cyanogenmod.com/blog`

priority level[18].

Documentation on the development process and the running of the project can be found on the official blog. The aim of the blog is to provide the user community with general information about releases and news about the project and the core team. The blog dates back to August 2009, and is written by members of the core team, with Steve Kondik being one of the biggest contributors. The entries originally aimed at announcing new releases and providing the user community with updates on the project in general. Over the years, the blog has now become a source of information on the workings of the core team, including rules of participation and behaviour. At the same time it documents the way in which the project has evolved, and how the core team has dealt with their relationship with other AOSP modifying communities and Google.

Support

The final type of user contribution is support, which consist of users helping each other to solve a wide range of problems, from CM ROM usage to programming. This user support differs from some forms of documentation in that there is direct interaction between individuals to solve a specific problem, whereas in documentation, such as FAQs and tutorials, there is no interaction. Therefore, support can be described as a one-to-one (and sometimes many-to-one) activity, whereas documentation is a one-to-many activity. User assistance is normally carried out by the user community in a number of different locations, primarily in internet forums and Internet Relay Chat (IRC) channels. The assistance that is given will depend on where the interaction takes place and what subject is being discussed. Although some user-to-user assistance can take place through private communication channels such as email, the CM project provides two locations for user support, the CyanogenMod forum[19], and the official IRCs[20].

The official CM forum contains 15 sections, with 13 of them dedicated to manufacturers and their devices. Each device section is further divided in to three subsections, one for general chat and one for each build – experimental ROM and

[18] `http://goo.gl/zaH73`
[19] `http://forum.cyanogenmod.com/`
[20] `irc://irc.freenode.net/cyanogenmod`

the stable ROM. Therefore, most of the assistance in the CM forum focuses on the device and the hardware-related problems. The assistance is provided by the user community at large, where any member is able to post a problem as well as a solution, while the core team is responsible for moderating the forum and making sure users adhere to the rules. Finally, in the general section, users are able to discuss a number of device-specific topics, such as application software, accessories, and data plans.

The IRC channels were formally set up by the CM team so that users can receive assistance in real time. The channels are open at all hours and individuals from all over the world are logged on at different times. The users in the channels give and receive advice on a number of issues, from general usage to software development. The discussions in the IRC channels are not device-specific, and revolve around a number of subjects, but most are focused on programming and access to resources.

6.4 Project Overview

The CyanogenMod project consists of a mobile operating system (known as a ROM), a ROM management software, and an installer. The main software is the ROM, which is based on the AOSP version of the Android operating system. The project itself also offers additional services such as user assistance, developer support, and training. The project began in a hacking forum in 2008, with user Steve Kondik (user name *cyanogen*) releasing his modified version of the AOSP. The project went through a growth period afterwards and it took its own web infrastructure, moving into its own domain in 2009.

Section 6.4.1 looks at the CyanogenMod ROM in relation to the Android operating system, looking at the similarities between them and at the parties involved in their development. The Android operating system is typically released by Google through their flagship devices called the Nexus range and contains all the Google apps and services. The AOSP is a generic version of Android which manufacturers and carriers can modify. Finally, community maintained ROMs, such as CyanogenMod, are based on the AOSP version with user modifications.

Section 6.4.2 then looks at the project's early origins in the XDA forum, their rapid growth, and the move to their own web infrastructure. The project started in the XDA forum, which was a community of hackers working on mobile devices and digital personal assistants. Steve Kondik was active in the XDA forum and then turned his attentions to Android as soon as it was first released.

6.4.1 Android, AOSP, and modified ROMs

There are several operating systems that are available in the ecosystem, Android, AOSP, and the modified ROMs. Each of these versions differ in terms of who maintains them, what devices they are available for, and the default apps and services that they contain. The Android operating system is maintained by Google and is released for the Nexus range of devices. The AOSP is maintained by the manufacturers and carriers, and is released for a number of devices. Finally, the modified ROMs are user maintained and are available for most of the devices. Figure 6.3 shows the relationship between Android, AOSP, and the user modified ROMs.

Google bought the smartphone operating system Android in 2005. In 2007, it joined forces with device manufacturers, carriers, and software companies to form the Open Handset Alliance (OHA), to which it handed Android's development while still retaining the lead role. The aim of the OHA is to promote Android as an open smartphone platform in an effort to decrease the industry's reliance on proprietary platforms. To actively maintain and distribute Android, as well as to encourage participation from the public, Google established the Android Open Source Project (AOSP) (Android.com, 2013a). It is through the AOSP that Google distributes the official operating system (commonly referred to as the AOSP version) to their OHA partners as well as the general public.

The Android operating system is actively maintained by a group of developers working at Google, called the Android development team. The role of this team is to implement new features and to incorporate many of the fixes or improvements that come from the community (see Reardon and Shankland, 2013). The first role of the Android development team is to review and to merge or reject many of the bug fixes from the community, as well as to review any change to code that

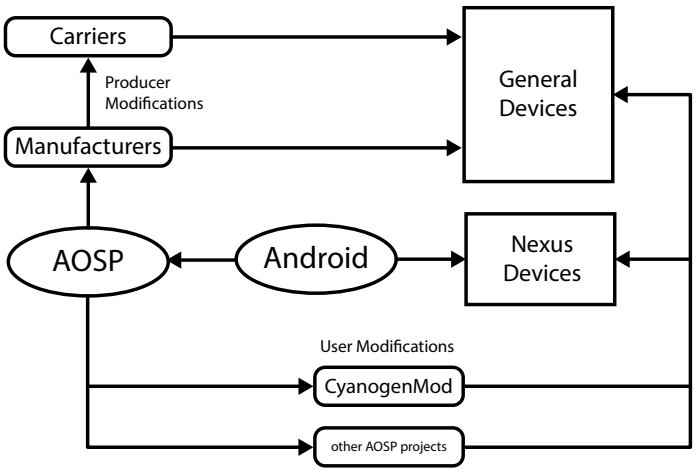


Figure 6.3: Android, AOSP, and User Modified ROMs. The Android Operating System as developed by Google is only used in the Nexus range of devices. The other devices have modified version of the AOSP.

is contributed to the AOSP project. The second role is to then implement new features on the operating system which can be consistent with Google's strategy for the platform. A good example of this is the merging of text messages with Google hangouts, which forces users to use more Google services on their device (see Pierce, 2013). Google releases the Android operating system only through their Nexus range, where they partner with a manufacturer to release a limited number of devices each year.

The AOSP version of Android is a community version of the Android operating system which is then shared with the community at large including the OHA. The purpose of the AOSP is to allow the OHA and the general public to modify the operating system and contribute to the project by submitting bug fixes or by improving the implementation of available features. The main differences between each version are functionalities that are limited in the AOSP or not present at all, such as keyboard gesturing and photographic spheres (Amadeo, 2013b). Furthermore, Google places restrictions in terms of the commercial licencing of the Android Operating System, which has implications in terms of which devices can be called Android-compatible and which ones cannot. This is based on the inclusion of Google proprietary apps, which can be installed by manufacturers as long as they pay a licencing fee of 75 US cents per device (Arthur and Gibbs, 2014), therefore these are not

included in the AOSP version. Because the license is only for commercial use, end users are able to download and install Google's proprietary apps without having to pay a licence fee.

The final form of Android-based operating system are the ROMs[21], which are a user-modified version of the AOSP. These ROMs are typically modified by the end users, who then go on to add much more functionality to it as well as to change the way it looks. The ROMs were at the beginning focused around an individual, who would hack the phone and install the operating system. Once the ROM was working on a phone, they would share it with others in the XDA community, and it would generate a following and contributors. Through the years, one of the most popular ROMs is CyanogenMod.

6.4.2 Origins in the XDA Developers Forum

The CyanogenMod project started in the XDA Developers forum in 2008. Most of the members of the CyanogenMod project had already been active in the forum and had previous experience hacking traditional mobile phones. The XDA forum provided the early project members with a place to share ideas and to share modified ROMs. The forum is still the home to many other AOSP-based ROM projects, such as MIUI and AOKP.

The XDA forum was an off-shot of the XDA Developers website, a site which focuses on the development of software for mobile phones. The website itself caters for the community of programmers that work on mobile technology, predominantly on operating systems such as Android, WebOS, and Windows Mobile. The website contains reviews and news from the mobile industry and contains a number written articles, podcasts, and videos. Finally, the website also hosts the developers' forum, which is one of the most popular software development forums that focuses on software modification and the distribution of modified ROMs.

[21]Traditional mobile phones had a basic firmware which would control all functionality that would be stored as Read Only Memory (ROM). Due to the increase in storage and functionality, smartphones have a dedicated ROM that loads the operating system which controls most of the functionality. Despite Android being predominantly an operating system, the term ROM has still persisted within the community of former mobile phone hackers in the XDA forum, and has since spread within the community of Android hackers.

Originally, the XDA forum was focused on the modification of a range of personal digital assistants, then in 2009 it allowed members to discuss other devices including smartphones and tablets. The XDA forum focused on the Extra Digital Assistants that were manufactured by the Taiwanese company HTC from 2002 onwards (ORB3000, 2010). These devices had a Windows Mobile operating system and were distributed through the O2 carrier under the name XDA (with the X standing for 'Extra'). The forum therefore presented a location where software developers who were modifying and testing these devices could meet and exchange information. In 2008, the XDA forum allowed its users to discuss other devices besides XDAs as well as other operating systems besides Windows Mobile.

Soon after forum rules were relaxed, projects started appearing in the forum which were focused on modifying the AOSP, one of which was CyanogenMod. The projects would be started by a single user who would post a modified ROM on the forum and allow others to use it and provide feedback. One of these projects was a ROM maintained by Ben Gruver, under the user name `JesusFreke`, which had an estimated five thousand users downloading it and was considered the most popular modified ROM at the time (Androidandme.com, 2009). In October 2009, Gruver decided to stop the development of his custom ROM and instead asked his users to switch to another customised ROM run by Steve Kondik, whose XDA user name was `cyanogen` (Gruver, 2009).

The CyanogenMod project and its members were active in the XDA forum, until the project grew in size and complexity in 2010, and the CyanogenMod community invested in its own infrastructure. Originally, the project was run through the forum, where a forum thread would deal with a particular port or version of the operating system. The actual software was typically hosted on a public website which allowed individuals to download freely and install it in their system. Once the project grew in terms of devices being ported, as well as user-participation in the discussions, community member Ahmet Deveci decided to set up and donate web infrastructure which included main page, wiki, and forum.

Although CyanogenMod has its own IT infrastructure, a significant number

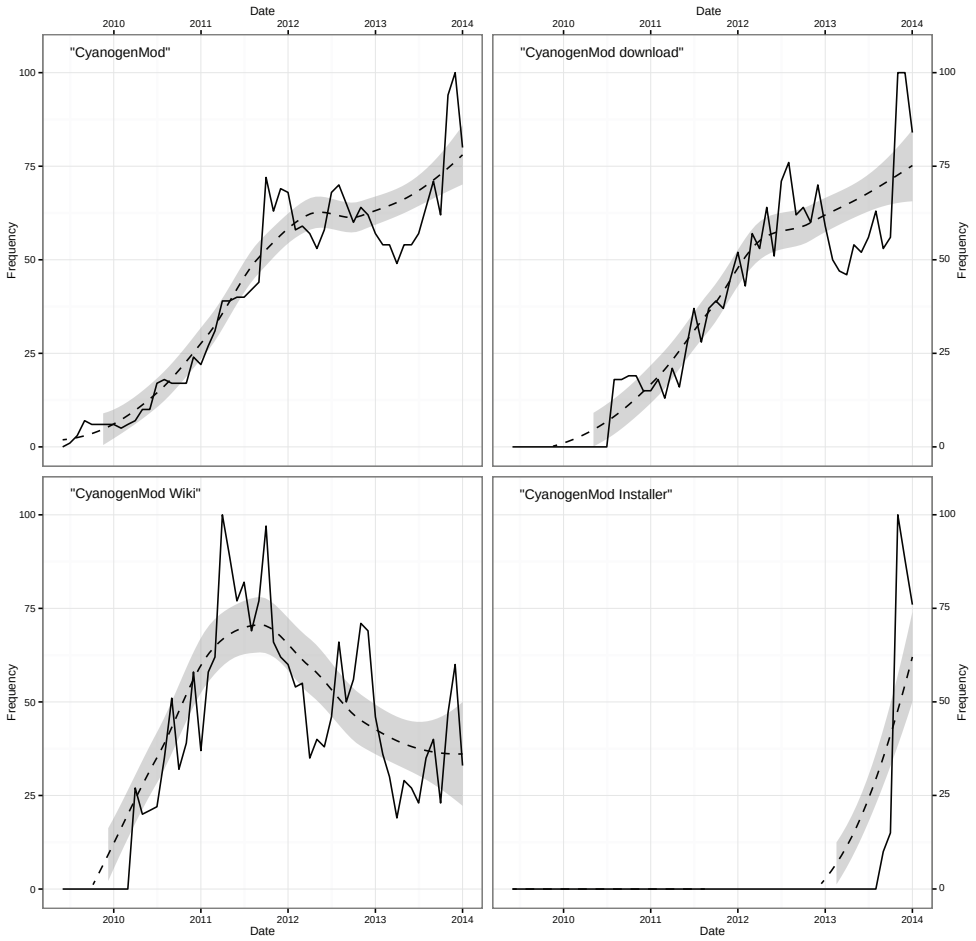


Figure 6.4: Google trends for CyanogenMod and peripheral services and products.

of users continue to be active and to contribute to CyanogenMod discussion on the XDA forum. As of April 2013, the official CyanogenMod forum had a approximately 300,000 registered members, in comparison, the XDA forum have over 4.9 million registered members. Finally, as new devices are being added to the XDA forum, new discussion threads continue to emerge which are all focused on CyanogenMod.

6.4.3 Project Popularity

The CyanogenMod project has seen a steady increase in both popularity and in the number of devices being supported. Although there are no direct numbers in terms of downloads or install based, other proxies can be used to determine the popularity of the project itself. For instance through Google Analytics it is possible to determine the relative interest in search terms that are related to the

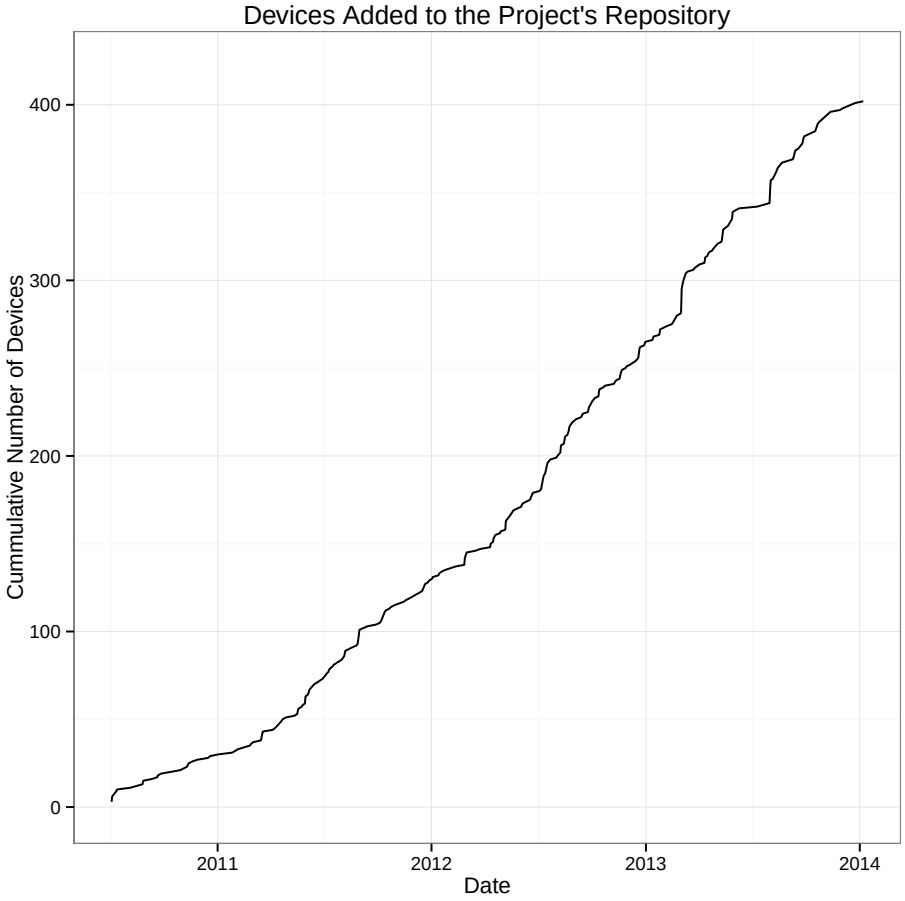


Figure 6.5: The cumulative number of devices added to the official repository.

project. In addition, data from the official repository shows the date in which new devices are added to the project. Both of these proxies have shown a steady increase in popularity and in contributions to the project.

Data from Google Analytics shows that the popularity of search terms related to the CyanogenMod project have increased since the start of the project (Figure 6.4). There were four main search terms analysed: "CyanogenMod", "CyanogenMod download", "CyanogenMod wiki", and "CyanogenMod installer". The resulting search trends show that there has been a significant increase in interest of the project and downloading information since the project began. Equally, the official wiki saw an increase in popularity from early 2010 until late 2011. These figures therefore show that user-consumers and user-contributors have been increasingly searching these terms through the Google search engine.

In addition, the popularity of the project can also be observed in the number of devices being supported by the project. Figure 6.5 shows an increase in the number of devices that have been added to the project's official repository. The data shows that during the observed period, from July 2010 until January 2014, there was a mean average of 9.35 new devices added each month. By January 2014, there were a total of device-specific 402 repositories, with March 2013 seeing 27 new devices added, the highest observed to date. This therefore shows that the official devices supported by the project have increased steadily since the start of the project.

6.5 Project Goals

The project has both an operational goal and a strategic goal. The operational goal looks specifically at the functionality and reliability of the operating system. In terms of function, the goal of the project is to develop an operating system that has better performance and stability over the AOSP. At the same time, the project aims to remove the restrictions and modifications from the manufacturers, and return to the original AOSP.

On a recent post on the official blog (Kondik, 2013), Steve Kondik described the project goals as:

- *Organize, lead, and support our community*
- *Create amazing user experience centered around how YOU work*
- *Security solutions that really work*
- *Stay committed to building the features our users need*
- *No junk*
- *Constant updates*
- *Available on everything, to everyone*

The project's strategic goal is to commercialise the operating system by partnering with a manufacturer or a carrier. The overall goal of Steve Kondik as understood by others in the project, is to develop an operating system that

is good enough for manufacturers to use in their smartphones. At the same time, the core team of developers also have the aspiration of working for the project in a full time capacity.

The strategic goal was partially achieved in September 2013 when they received 7 million USD in funding from Benchmark Capital and Redpoint Ventures. This investment has allowed them to create a commercial company called "CyanogenMod Inc" with offices in Palo Alto, California. The main objective of CyanogenMod Inc is to develop the CMOS full time, allowing them to hire 17 full-time developers and open offices in Palo Alto (Newton, 2013; Amadeo, 2013a). At the same time, on the 23rd of September 2013, the CyanogenMod team announced a partnership with device manufacturers OPPO Mobile, which will see the CyanogenMod operating system installed as default in the new OPPO N1 smartphone (Devkota, 2013).

6.6 Official Project Groups

The project has six official groups which are in charge of a wide range of activities, and these have been specifically listed on the cyanogenmod wiki, at wiki.cyanogenmod.org/w/Devs. In this list, we have a core of 9 who are regarded as the core team in CyanogenMod. This is followed by 2 community staff, 60 device developers, 11 general developers, 29 translators, and 5 wiki maintainers. Table 6.3 has a more detailed description of each of the roles and the number of individuals who had those roles.

The core team are involved in the genenral management of the project and are also key to other supporting activities within the project such as public relations and marketing. The group includes project founder Steve Kondik and key developer Koushik Dutta and Jef Oliver. In addition, the core team also has individuals that are focused on other key aspects, such as the maintenance of the ITC infrastructure. Three of the core team members Keyan Mobli, Ricardo Cerqueira, and Chris Soyars are also listed as a device maintainer. Figure 6.6 offers a simplified depiction of the work delegated within the core team, although the actual distribution of roles is considerably more complicated.

Role	Number	Description
Core Team	9	Carry out a number of jobs, such as community relations, founders, head of merchandising, and main maintainers of supporting software.
Community Staff	2	Manage the Internet chat channels and moderate the forum.
Device Maintainers	65	Maintain the operating system for individual devices.
General Developers	11	Miscellaneous contributions based on programming the operating system and the websites.
Translators	30	Maintainers of the translations to 18 different languages. This also includes some who contribute to the wiki translations.
Wiki Team	5	Main editors and maintainers of the official Wiki.

Table 6.3: Sub Groups of the CyanogenMod Official Key Members.

The community staff consists of two individuals, Dean Fanning (user name *TheDeanius*) and Evan Widger (user name *PsychoI3oy*). Dean Fanning is in charge of moderating the IRC channel, and he was also the person that started the channel. Finally, Evan Widger maintains the bug list, and is responsible for looking at how the bug list in Gerrit functions. Evan Widger was dessignated the job of moderator because of his constant involvement in the Gerrit community, and Abhisek Devkota then gave him the position as it became impossible to coordinate all activities.

The device maintainers are those that are able to officially port CyanogenMod into a new device. Their names are added to the list of official maintainers when they have followed all the rules as specified by the official wiki. These rules specify that all hardware components must work and provide instructions on several technological issues such as not changing the default settings for how the device works. The second important prerequisite is that the main device developer will have to contact the core team, who will then review and suggest changes or fixes.

In the general developer section, there are a number of programmers that are in charge of conducting a range of different roles around the project. This group includes Chris Soyars (user name *ctso*), who is responsible for the website infrastructure, including web hosting and downloads. In addition, there are also Bjorn Lunden (user name *blunden*) who is listed as being head of user and

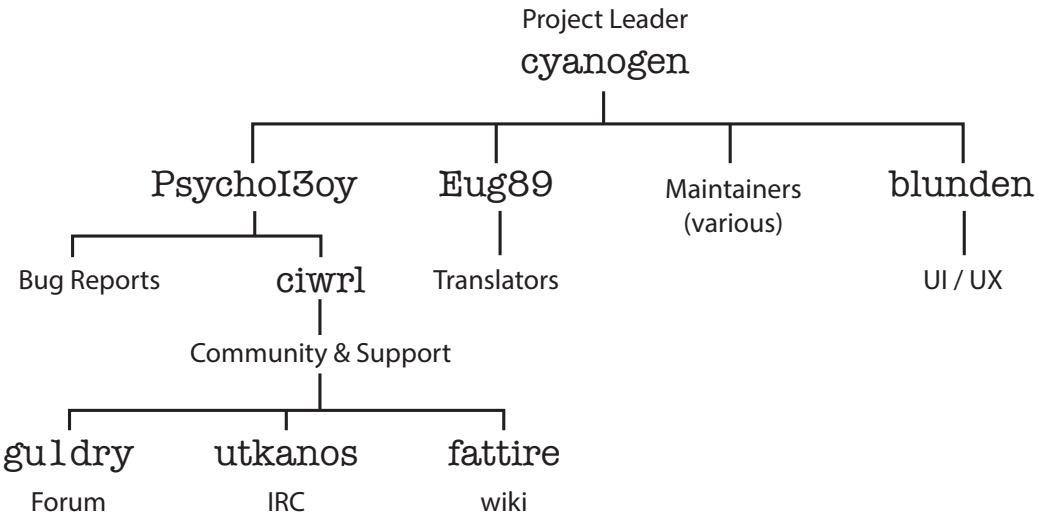


Figure 6.6: Core Team Structure

experience interface.

The translators are those that help with the main operating system, there are a total of 29 translators working on 18 languages. The translators do not all specialise, as users Tanguy Pruvot, Jens Andersen, Danny Baumann, and Alexander Hofbauer not only do translations but also maintain devices.

The wiki team consists of five contributors who are in charge of maintaining and editing the official wiki. The lead maintainer is Drew Suarez (user name *utkanos*) and he has three assistant editors, user name *fatire*, Amod Mulay (user name *White*), and Shawn Alty. The editors mainly make sure that the information is available to users and that it is kept up to date as new devices are launched and new version of Android released. Finally, Flo Edelman (user name *Flo2154*) contributed to the wiki through programming and redesigning the main page. User *fattire* also maintains a device and is listed in the device maintainers.

6.7 Core Administrative Roles

There are three main core administrative roles, goal setting, work delegation, and project communication. The goal setting role is carried out by project founder Steve Kondik, who then communicates this goal to other core team

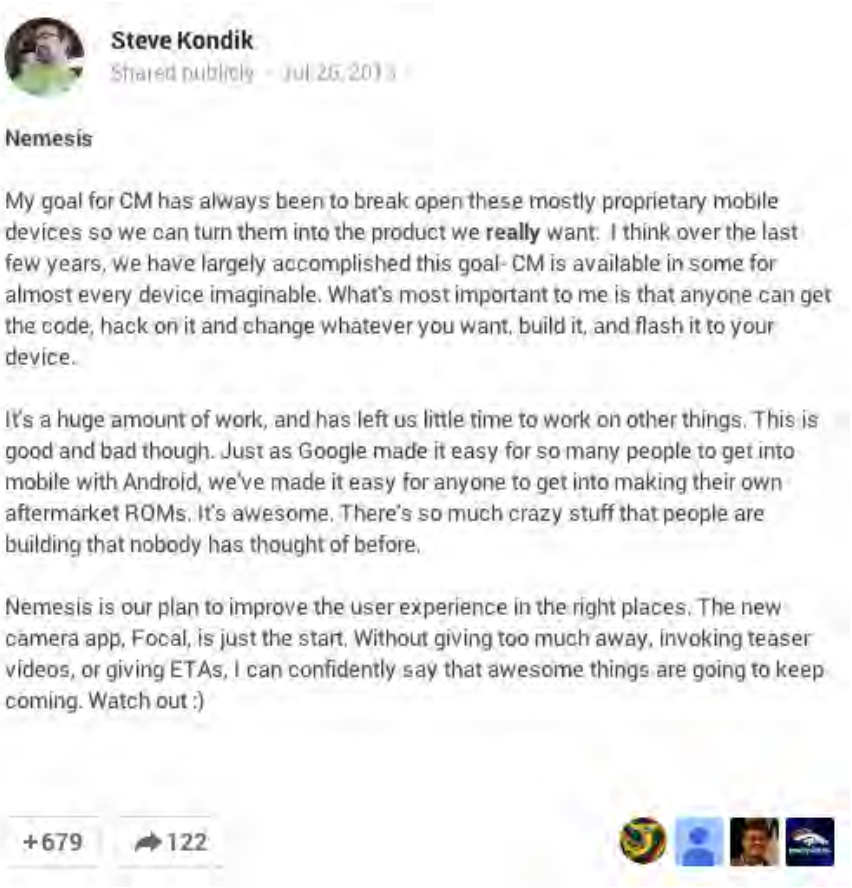


Figure 6.7: Steve Kondik Setting Goals

members. Work delegation is carried out by several individuals within the core team, with the aim of reducing the workload for each individual. Finally, project communication is carried out by Abhisek Devkota, who informs the general contributors and users about the project's progress as well as any future direction.

The goal setting role primarily looks to set the overall objective of the project, as well as to suggest the development of supporting software or the provision of new services to achieve these goals. Figure 6.7 provides an example of the overall objective set by Steve Kondik and the resulting initiative to support it, the creation of the Nemesis camera app[22]. On a similar note, Kondik also states that the biggest barrier for adoption is the complex and device-specific installation. This was then noted as the main reason behind the One Click Installer, which has the aim of facilitating the installation process.

[22]The Nemesis camera app is not included in the latest version of the operating system.

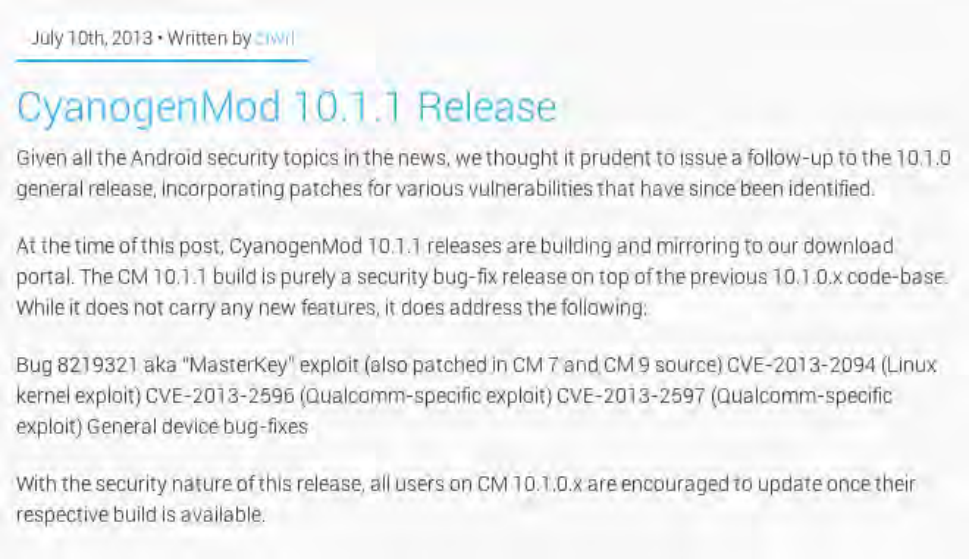


Figure 6.8: Community Relations. Taken from the CyanogenMod official blog.

Work delegation is linked to goal setting but it is also important in other areas of the project, primarily interaction with the general contributors and users. The specific work that has been delegated has been the creation of the learning center and the tutorial on how to install the operating system. The delegation of work has also been done when looking at specific projects, particularly the creation of the CM Account service, which was a job specifically delegated to Chris Soyars because of his expertise in web development.

The majority of the delegated work, however, has been done by other members of the community and it focuses primarily on community management. In the case of Evan Widger (user name *PsychoI3oy*), he was delegated the role of managing the large community of contributors, which included the Gerrit review website, the forum, IRC, and the wiki. As the number of contributors increased, Evan Widger then delegated the managing of the forum, the wiki, and the IRC to Abhisek Devkota. In turn, Abhisek Devkota delegated the day to day management of each interaction space to three other community members.

The role of community relations serves the purpose of transparency and of letting contributors and users know where the project is and where it is headed. This role is carried out by Abhisek Devkota (user name *ciwrl*) through a number of mediums, but primarily through the official blog. The posts differ in content

but are focused on letting the general contributors and users know more about the processes involved in the development as well as future directions for the project. General public announcements have also been made by Abhisek Devkota dealing with security issues, lateness in updates, and major bugs found in recent install procedures. Figure 6.8 shows a typical example of the core team communicating to the general public about security issues and updates made to the operating system as a result of news on Android.

6.8 Licensing

The four main softwares being developed by CyanogenMod are all under the Apache Software License version 2 (ASL2). In the case of the CyanogenMod ROM, the license has been inherited from the AOSP and allows it to incorporate and redistribute user contributions. In addition to this, the project also makes use of the Contributor License Agreement, which reinforces the ownership of public contributions by the CyanogenMod project. This ensures that any modification from non-core team members are owned by the project. There have been a number of cases where the licensing has caused problems for the project, resulting in apps being removed from the project.

Google has provided three main reasons for choosing ASL2 over other open source alternatives (see Android.com, 2013b), citing concerns from the manufacturers as their main reason. In the discussion of why they chose the ASL2, they compare this license to the GNU Lesser General Public License (LGPL), which they regard as an important alternative open source license. The first observation made is that manufacturers may not be able to comply with the requirement of providing the source code, which may not be possible as the operating system is installed as a static system image. The second observation is that the LGPL also allows for customer modification and reverse engineering, which may be something that the manufacturers may not want to agree to. Finally, the last point made is that they have found it hard to make sure that the device manufacturers and developers comply with the LGPL terms. It is therefore the belief of Google that the ASL2 will provide more

flexibility in terms of compliance, and more control in terms of distribution than the LGPL.

In addition to the CyanogenMod ROM, all other software created by the project, ClockworkMod and the One Click Installers and the custom apps, have been released on GitHub under the ASL2. In terms of user contributions, section two of the ASL2 gives CyanogenMod the right of including all contributions from the general public, although these rights are non-exclusive. In order to comply with the license, the author of each module must give credit to the original source (if any), must identify any changes made, and must retain previous copyright attributions.

In addition to section two of the ASLA2, the CLA is also included within the Gerrit review system and the CyanogenMod repository, allowing the project to have complete ownership over all the contributions from non-official team members. The CLA was started specifically to look after the way that people contributed to the project, and this included the need to make sure that the owners of the open source project had ownership over all contributions to then implement their own licensing (Morrison, 2013). This therefore means that the CLA acts as an additional feature that allows the project to implement and cover all contributions from external sources to a licence of their choosing.

While CyanogenMod has ownership over the modified ROM, it does not have ownership over some of the apps that are included, a notable example were those of the Google Apps and the Focal camera app. Google entered litigation against CyanogenMod for the section 4 of their terms of services, which stated that "...any means other than through the interface that is provided by Google for use in accessing Google services..." (Google, 2011). This resulted in CyanogenMod removing Google apps from the ROM, giving users instructions on how to install the apps after installing the ROM. Similarly, CyanogenMod included the Focal camera app, which was later removed when its license did not allow commercialisation (Fingas, 2013).

6.9 Chapter Summary

The Android operating system is built with open source technology, making it easy for producers to modify and take advantage of its supporting networks. The open nature of the Android platform has allowed user communities to modify and improve the AOSP by removing restrictions and unwanted services that were added by the producer.

Google specifically chose the open source technologies to be at the heart of the operating system because it meant they were able to modify them to suit their needs. Google modified the Linux kernel so that it could make better use of the limited resources on mobile devices, and then added to the project by improving energy and memory management. The use of Java as the app programming language gave Google the ability to modify the Java bytecode and change its implementation so that it used the limited resources more efficiently and reduced the memory footprint.

These open technologies have also allowed device manufacturers and network operators to modify the Android platform to suit their needs and to advance their commercial interest. Besides the need for the Android Linux kernel to be modified so that it works for each device, manufacturers have also extensively changed the GUI to produce a manufacturer-specific Android experience which can set their devices apart from others. The network operators have also modified Android by providing extra default apps which make use of their services, such as video streaming and messaging.

These extensive modifications mean that the user will not come in contact with the original Google version of Android, the AOSP. What is more, the modifications carried out by the producers mean that the Android platform may be slower and more inefficient than was originally designed. As a result, and because of Android's open source license, users have hacked their smartphones and replaced the producer-modified AOSP with their own user-modified version. Users have then exchanged their modifications with other users online, and created Android modification communities, with the biggest one being CyanogenMod.

CyanogenMod, like traditional open source communities, relies on users

volunteering their skill and time to contribute to the project. The CM community can be divided into two main groups, the core team, who develop most of the CM-specific features, and the user group, who mainly port the ROM to new devices. At the same time, the core team have established a project infrastructure through which other forms of community collaboration, such as documentation, take place. The core team has also been divided in such a way as to reflect other forms of contribution, where there are core team members who specialise in editing the documentation, and others that look after the websites and forums.

As mentioned in the literature review, by focusing on the community of programmers only, the complexity of modern open source projects has not been fully explored in the current literature. In addition, the social structure within the project may be a reflection of one form of working group within them, not taking into account other key groups, such as project leaders, which could have an effect on the project. The CyanogenMod project therefore presents a complex study with multiple forms of contributions which could improve our understanding on how these are governed.

Part III

Findings

Chapter 7

Products and Services

7.1 Introduction

This chapter looks at the products and services provided by the project. The Cyanogen project maintains one main product, the CyanogenMod ROM, and two supporting software, ClockworkMod and a dedicated installer. The CyanogenMod ROM is the modified and community maintained operating system, which has a number of modifications and customised default applications. Both ClockworkMod and the dedicated installer are supporting application that simplify the process of installing the CyanogenMod ROM on to mobile devices.

In addition, the project provides three main services, a CyanogenMod account, user-to-user assistance, and programming training. The CyanogenMod account is similar to Google's Android Device Manager, which allows users to locate their devices when lost or stolen, as well as to remotely disable them or delete the contents. The user-to-user assistance is provided to users in order to solve minor problems in installation or usage of the CyanogenMod ROM. Finally, programming training is provided to users in the form of step-by-step tutorials with the aim of allowing users be able to contribute to the project through modifying and porting the operating system.

The chapter is structured as follows. Section 7.2 describes the three main software applications, along with the mode of development and ownership. Section 7.3 then describes the three main services provided by the project,

Product	Description	Ownership	Developers
CyanogenMod Operating System	Adds new functionality in terms of new features and default apps. Including the ability to modify the look of the interface as well as added security.	Open Source	Core Team and Developers
ClockworkMod Recovery	Allows users to carry out low level functionality, load different operating systems, and side-load apps.	Open Source	Koushik Dutta
CyanogenMod Installers	Allows users with low technical knowledge to unlock and download CyanogenMod operating system to their device.	Closed Source	Core Team

Table 7.1: CyanogenMod Products. The three main software products provided by the CyanogenMod community.

including the individuals that are responsible for providing them and their target audience. Finally, Section 7.4 goes into further detail regarding the three main stages of development, Nightly, Release Candidate, and Stable Releases.

7.2 Products

There are a total of three software products that are being developed or maintained by the CyanogenMod community, the CyanogenMod ROM, ClockworkMod, and the One Click Installers. Of the three, only the ROM is open for general contributors, while the rest are developed exclusively by members of the Core Team. The CyanogenMod ROM is maintained by the core development team and its repository is open to the public, allowing the public to port it to a new device and make modifications without restrictions. The ClockworkMod Recovery is maintained primarily by Koushik Dutta and although it is part of the CyanogenMod ecosystem it can also be used in conjunction with other modified ROMs. The CM installer is a closed proprietary software that is used to install the operating system for users who do not have the technical skills to do so. Table 7.1 provides a summary of the three main products along with a brief description of their functionality, ownership, and its development.

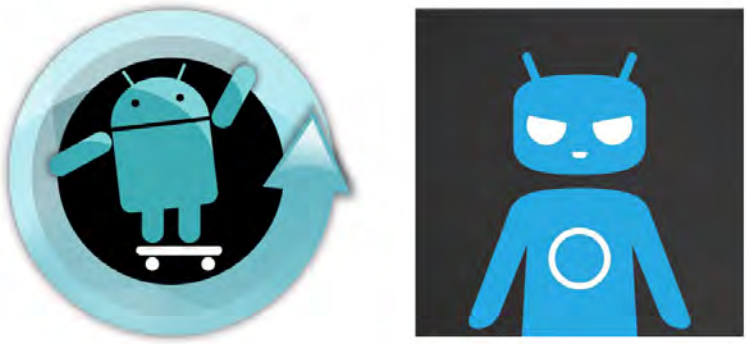


Figure 7.1: CyanogenMod Logo. The old logo (left) and the new mascot and logo (right).

7.2.1 CyanogenMod Operating System

The CyanogenMod operating system is a modified version of the AOSP with twelve system modifications and four changes in the default app. The operating system can be downloaded as a binary package ready for installing through the get.cm website or as source code from the its GitHub repository[1]. It is available for 221 devices, it has been installed over 6 million times, translated to 48 languages, and has gone through 11 major version (stats.cyanogenmod.com). Figure 7.1 shows the old and the current logo for the CyanogenMod project and operating system.

The following subsections look at particular aspects of the CyanogenMod operating system which differentiate it from both Android and the AOSP. First, it looks at three key characteristics advertised on the CyanogenMod website as the key features. Then it describes the key modifications done to the operating system. Finally, it presents a description of the default apps that have been changed.

Characteristics

The CyanogenMod ROM is advertised as being bloatware free, more secure, with faster security and feature updates, and with better performance than Android or AOSP. The first major characteristic is the removal of carrier or manufacturer modifications, which contain a large number of default apps that may limit the performance and functionality of the device. These added apps have a tendency

[1] github.com/CyanogenMod

to slow the system by using resources in order to provide manufacturer or carrier-specific services, such as HTC's Sense User Interface implementation of BlinkFeed (see Kastrenakes, 2013). One of the main selling points is therefore the removal of the bloatware, which then results in a leaner operating system with a lighter footprint[2].

CyanogenMod also advertises increased levels of security in the operating system, adding encryption to messaging services, providing more frequent security updates, and adding security-focused features. The operating system has implemented encryption to its messaging services as default, making the messages only readable by the sender and the receiver (Jeffries, 2014) In addition, manufacturers and carriers may suffer some delays when implementing security fixes to the AOSP on their own modified operating system. CyanogenMod offers much faster and more frequent security updates by passing on any fixes from either the AOSP or from the CyanogenMod community of developers. Finally, the operating system also contains a number of new security-focused features, such as the ability to use VPNs[3] and automatic encryption of messages.

An additional feature of the CyanogenMod project are the security and feature updates from the AOSP, which may sometimes be delayed by the carriers or the manufacturers. The difference is that some devices are not given the new features because these clash with the manufacturers' or carriers' version and therefore are never implemented. At the same time, as mentioned above, some of the devices stop being supported, and therefore do not get the security or the AOSP features at all.

Finally, the CyanogenMod project also provides performance features with a series of modifications that reduce memory requirements and increases speed of execution. The performance can be a number of things and it can also mean the weight and the footprint of the system. At the same time, because the software has been reduced in general it has fewer programs running in the background which are more likely to take resources. Similarly, there are also settings that are

[2]Meaning that it takes a large amount of resources and storage space while running the operating system

[3]Virtual Private Networks give internet connection an added level of private security.

linked to the way that the device uses batteries. Finally, there is a control panel which can determine which apps use resources such as data connectivity, power or memory.

System Modifications

System modifications are those that are hard-coded into the operating system and are not able to be removed or replaced without modifying the source code. Most of these modifications can be found by the user through menus and option settings throughout the operating system, adding a range of functionalities such as system resource monitoring and visual customisation. Table 7.2 has a list of all the features divided into three main categories, CyanogenMod features, general features and user-focused customisation.

CyanogenMod Features are those that are particular only to the CyanogenMod operating system. It includes features like the CM Updater, which helps users upgrade or update the operating system OTA[4] as soon as the latest version is available. In addition, it includes a CM account, whereby individuals can create an account with the CyanogenMod project, allowing them to track their device and control it remotely. Through the CM account, CyanogenMod maintainers are also able to obtain statistics on how the devices and the software are being used.

General Features includes those that are not present in the Android or AOSP versions, and that add functionality to the operating system. This includes new functionality such as a privacy guard, which allows users to determine what permissions each of the apps have (such as access to the camera, access to the microphone, or to internet data). It also includes a file manager, which allows users to access and manage all the files in the device. Finally, it also includes more technical functionalities that allow users to take more control over the devices, such as giving the user root access as well as the creation of a super user.

User-focused Customisations are those that allow users to change the way the operating system looks or works without modifying the source code. This includes the ability to add icons to the settings ribbon on the top banner, and

[4]Over The Air means that the operating system can be updated without connecting the device to a computer.

Feature	Functionality
CyanogenMod Features	
CM Updater	Updates the operating system to the latest version of CyanogenMod.
CM Account	An optional feature which allows users to locate their device and erase data in case it is lost or stolen.
General Features	
Privacy Guard	Allows users to specify the permissions each app can have.
Black List	Allows users to block telemarketers and other unwanted numbers.
File Manager	Provides access and the ability to manage files within the operating system.
Improved Display Controls	Improves settings on screen brightness, rotation, wall papers, notification lights, and battery lights.
Profiles	Changes device settings automatically depending on location or activity.
Navigational Bar	Creates a bar with shortcuts which users can modify.
DSP Manager	Gives greater control over music and audio settings by providing a 5 band equaliser.
Tethering	Allows users to use tethering functionality which may have been removed by carriers.
Development tools	Allows developers to monitor the system and facilitates debugging process.
Root Access	Gives more control over the device by allowing users complete access to all parts of the software.
Super User	Allows better management and control over the device and software by limiting and granting system permissions to apps and users.
Performance Option	For advanced users, gives the ability to make the device use more resources than the recommended setting.
SMS Rate Limit	Limits the number of text messages sent by the device, as a precaution from potential viruses.
User-focused Customisations	
Quick Setting Ribbon	Allows users to customise the top ribbon on the display to add more icons and features.
Setting Customisation	Gives users the ability to customise the main settings menu to make certain functions more accessible.
Trebouchet	A system app that allows users to create, apply, and exchange visual themes for the operating system.
Button Configuration	Gives users the ability to change the configuration of bottom navigational buttons.
Lock-Screen	Allows users to modify the lock screen by adding shortcuts and displaying a wider range of information.

Table 7.2: System Modifications and Features. System modifications and features in CyanogenMod version 11

App Name	Maintainer	App Replaced	Features
Apollo Music Player	Andrew Neil	Google Music	Automatic downloading of artwork, more lock-screen controls, sleep controls, EQ support, customisable themes.
8sms	thinkleft.com	Google Hangouts	Quick reply popups, gesture-based templates, emoji support, customisable themes, light weight.
Calculator (CyanogenMod)	CyanogenMod	Calculator	Complex mathematical calculations, lightweight, wider range of widgets and lock screens, open source.
GalleryNext	CyanogenMod	AOSP Gallery	All pictures placed in a central source, better integration with social networking sites, automatic photo grouping and classification, video playback and gif support, picture editor.

Table 7.3: CyanogenMod Modified Apps. List of the modified default apps found in the latest CyanogenMod operating System.

to change phone themes, including color scheme, icons, wall paper, and fonts. Finally, it also provides users with the ability to modify the lock screen in order to gain direct access to the camera or other apps straight after unlocking the device.

Modified Apps

Modified apps are the applications that are included as default on the ROM and replace the stock apps found in either Android or the AOSP versions. There are a total of four apps which come as standard in the CyanogenMod ROM and that replace the ones found in Android and in the AOSP versions, these are the music player, text messaging, photo gallery, and the calculator. In addition, with the exception of the music app, all the default CyanogenMod apps are also available on the official Google Play store. The Apollo music player and the 8sms messaging apps have both been forked by the Core Team and modified in order to be integrated into the ROM, while the remaining two apps were all developed by Core Team member. Table 7.3 contains a summary of the apps with a brief description of their functionality.

The default music app in CyanogenMod is the Apollo Music app, which contains more features than the default AOSP app such as automatic art and lyrics downloads for each song. The app itself was created by developer Andrew

Neil, with some artwork and user interface elements being provided by A.J. Lopez. The main selling advantage of the Apollo music app is that it can automatically take the artwork for the artists and the albums from the internet. This means that, should a track or album not contain any artwork, instead of displaying an empty space in the player, it will look online to find and match the artwork. Similarly, the lyrics of the songs are also downloaded automatically. In addition, the user has more control over the player through the lock screen, meaning that the device does not have to be unlocked in order to control the music player. Finally, it has a number of other features such as graphic themes, equaliser support, and sleep controls that automatically stops the music after a period of inactivity. The app was removed from the store after MusixMatch filed a case of copy right infringement because of the way it downloaded the lyrics (see Neil, 2012).

The default messaging app has been changed to the 8sms messaging app, which is smaller in terms of footprint and has a number of features such as customisable themes. 8sms replaces Android's Hangouts app and AOSP's SMS Messaging app, it is maintained by software company ThinkLeft.com and is available to the general public through the Play Store. The app is integrated into the CyanogenMod ROM and contains a number of added features such as a quick reply popups, allowing users to reply from the alert banner. In addition, it has support for gesture-based reply templates, emojis[5], and themes. Finally, as of Android KitKat, released in late 2013, the default messaging app was changed to Google's Hangouts, which also incorporates video messaging. As a result, 8sms is seen to have an advantage in terms of size and lightness with a small footprint, as it requires less storage space and uses fewer resources.

The default calculator has been changed to an app called Calculator, which is maintained by a developer known as Xlythe and can be found in the Play Store under the name "Calculator (Cyanogenmod)". The CM calculator differs from the AOSP version in that it can carry out more complex mathematical functions while at the same time remaining lightweight in terms of footprint, and it has a wider selection of widgets. Finally, the app is open source, with its source code

[5]A collection of small images such as happy faces and other icons.

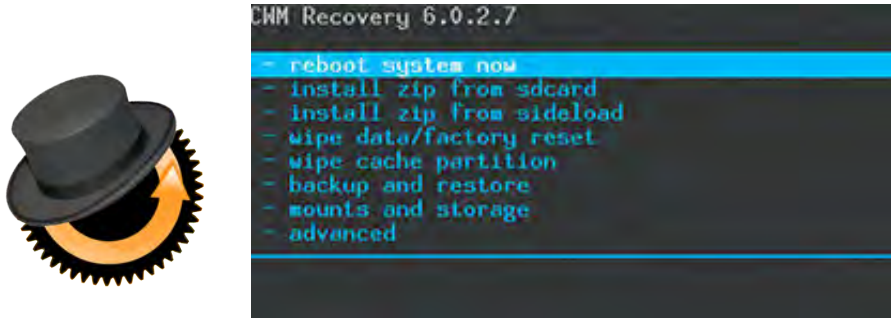


Figure 7.2: ClockworkMod. Logo (left), and the main menu on the (right) showing main functionality. (zteblade3.com)

available to the public through GitHub.

Finally, as of late 2013, the CyanogenMod team were looking to replace the default picture gallery with GalleryNext, developed by the core team and, at the time of writing, available on the Play Store for public testing (CyanogenMod, 2014). The new gallery would add new functionalities by automatically grouping all photographs depending on date of location, by supporting other formats such as gif as well as videos, adding and editing option, and better integration with social networking sites. In addition, it would solve some of the issues that may be faced with the default gallery app, such as a duplication of photographs when placed under different categories, and having a central folder for other applications to access.

7.2.2 ClockworkMod

ClockworkMod is a recovery software which helps manage the installation and backup of Android-based ROMs on a wide range of mobile devices. The software was first written and is currently maintained by developer Koushik Dutta, it is open source and the source code is available through Github[6]. Figure 7.2 shows the logo and an image showing the main functionalities of the software.

The ClockworkMod ROM manager allows the user to conduct low-level operations on the devices, such as rebooting of the hardware as well as creating and restoring backups. It also allows users to manage internal and external memory by deleting the cache partition and conducting a factory reset. The

[6]github.com/clockworkmod

advanced options allows users to carry out more complex tasks such as reformatting and partitioning storage, testing hardware keys, fixing file permissions, and managing battery statistics. Finally, its core functionality is the installation of modified ROMs and Apps, both of which are compressed as zip files. Apps can be installed without access to the official store through a process called *side loading*[7]. A modified ROM can be installed by first saving the zip file to the external memory card, and then installing it through the ClockworkMod menu option.

7.2.3 One Click Installers

The one click installers are composed of two complimentary applications, an app to be installed on the device and another to be installed on a computer. The focus of the installer is to help non-tech savvy users through the unlocking and installation process by automating most of the technical tasks. The Android app is open source and the source code is available through GitHub[8]. The desktop app is not open source, but it is free and it can be either downloaded from CyanogenMod's download page[9] for the Windows version, or through a Google+ group[10] for the Mac version. It supports over 35 devices, which includes all of the Google Nexus range, most of the Samsung premium smartphones, and four variants of the HTC One. Figure 7.3 contains screen shots of the mobile phone and the Windows desktop installer.

To use the One Click Installers, the device app installer is first downloaded from the CyanogenMod website and installed through side loading, then Windows app is downloaded and installed on the computer. Once both apps are installed, the user connects the device to the computer, runs both apps, and is then guided through the process. The Windows installer will download a ROM manager, root the phone, and then install CyanogenMod along with a number of system applications.

[7]The app installer was, for a short time, available in the Google Play Store. It was then removed because it violated rules of participation in that it allowed and facilitated users to void their warranty by unlocking the device.

[8]github.com/cyngn/OneClickAndroid

[9]get.cm

[10]`https://plus.google.com/communities/100663046352711689172`

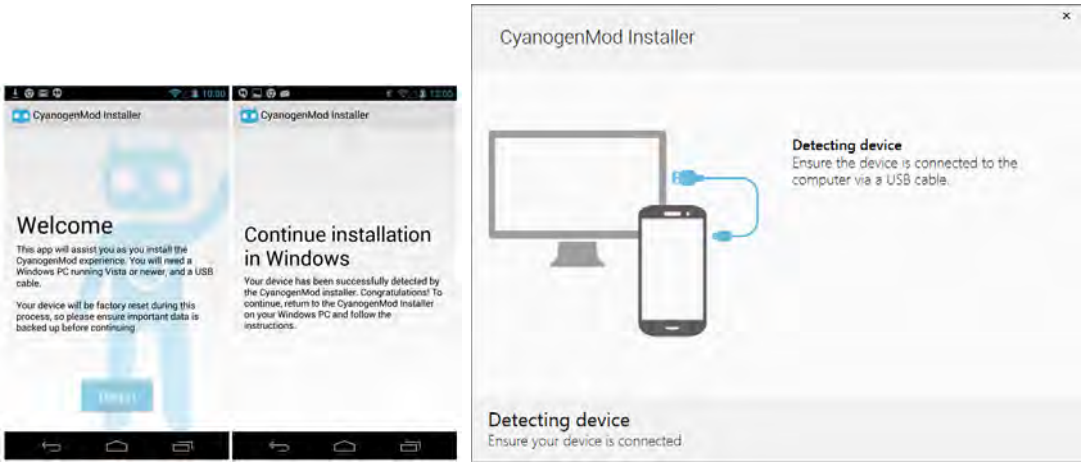


Figure 7.3: CyanogenMod Installer App and Windows program. The Android CyanogenMod installer screen shot - top (rootandroid.net), and the Windows companion program - bottom (arstechnica.com).

7.3 Services

The CyanogenMod project provides three main services to its user base, a CM Account, User Support, and Programming Training. The CM Account offers users the ability to track and manipulate their devices remotely. User support seeks to help individuals solve non-programming issues, such as installing, updating, using the operating system, and fixing minor glitches. Finally, programming training is provided at beginners, intermediate, and advanced level, with the objective of training individuals to modify the operating system and to port it to new devices.

These services are primarily, although not exclusively, provided through the project's ITC infrastructure, with the wiki being the primary source of learning, and the IRC and forum being the primary location for technical support. All of the services also make use of user contributions except for the CM Account, which is solely provided by the CM team using mostly open source technology. Table 7.4 has a brief account of the main services provided, who they are provided by, and who their target audience are.

7.3.1 CM Account

The CyanogenMod Account was announced on the 19th of August 2013, and provides a services of searching, locating, and swiping a smartphone. The feature

Description	Provided By	Target Audience
CM Account: Provides search, locate, and erase devices that have the CyanogenMod ROM installed	CM Team	All users
User Support: Trouble shooting and problem solving for users. Dealing mainly with user-based non-programming problems.	CM Team, Users, and Developers	End users
Programming Training: Helping users develop their skills to become programmers, specifically to contribute to the project.	CM Team	User and Developers.

Table 7.4: Cyanogenmod Services. A list of services provided by the CyanogenMod project.

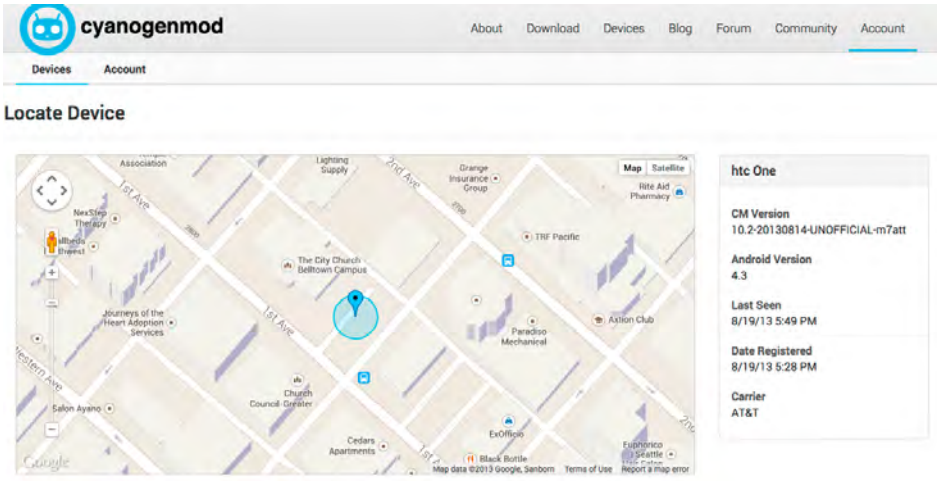


Figure 7.4: CyanogenMod Account. Showing the main functionality of the CyanogenMod Account (androidpolice.com).

works similar to that of Google's Android Device Manager and Apple's Find My Device. The feature means that, should a device be lost or stolen, the owner can be able to locate and track the device through a dedicated website. Once the device is tracked, there is also the option of being able to completely erase all the contents of the device remotely.

The actual service, or the software which provides this service, is open source and the source code can be accessed through GitHub[11]. The objective of this service is to provide an alternative from the commercial counterparts, with increased security through encryption and by avoiding commercial interest. The argument made in the Cyanogenmod blog is that the commercial counterparts place the information of the device in commercial hands, and gives these companies power to locate and manipulate the device. As Abhisek Devkota,

[11] `https://github.com/CyanogenMod/android_packages_apps_CMAccount`

CyanogenMod's head of public relations, states

> There are existing solutions on the market to allow Find/Wipe functionality, but we feel they are inherently insecure, enabling company employees or malevolent attackers to access your location data, or other information, without your permission.

The CM Account, on the other hand, differs from the other services in that the information is encrypted and therefore not accessible to the CyanogenMod core team. Finally, Abhisek Devkota states that the source code has been made accessible through the GitHub repository not just to allow its users to fix and improve the service, but also as a sign of transparency.

> The application is open sourced and Apache licensed. We highly encourage our contributors to participate in a security and privacy review and understand what sets us apart from other solutions.

7.3.2 User-to-user Assistance

User-to-user assistance focuses on solving non-programming problems relating to the installation, update, or general use of CyanogenMod products or the device. This service is usually provided by all three groups of the community to end users using primarily forums, social networking sites, and IRCs. Users typically post problems regarding applications not working properly, the device shutting down frequently, or the battery not lasting long enough. These problems can usually be solved by the end user without having to modify the source code. Through social networking site Google+, a user posts a problem regarding apps that do not work or constantly shut down. Here, three other users provide possible solutions to the problem, non of which require programming.

Although the forum and the IRC are the official locations where user-to-user assistance takes place, other locations that allow discussion, such as social networking sites, such as Google+ or Twitter, can be used. Each of these technologies allows the information to be classified and retrieved in different ways. For instance, in the forum, the structure allows subjects to be classified in terms of device and vendor, so individuals are able to navigate within the forum to find the relevant information. Similarly, the CyanogenMod project has set up

a group on social networking site Google+, where they also have sub categories based on specific devices. These two technologies allow people to post questions in the relevant section, and then other users are able to respond. The IRC on the other hand does not have any subdivisions and are not structured according to any type of theme, therefore being more open to a wide range of discussions.

According to the project's documentation, user-to-user assistance is provided by team members, developers, and end users, depending on the type of question that is asked and the availability of individuals. Core team members offer basic support by providing tutorials and FAQ (frequently asked questions) sections in both the wiki and the official forum. Here they take the user through some typical questions regarding updating the system, managing battery time, and installing Google apps. The developers and end users are more likely to interact on the forums and social networking sites where they can ask and answer to device-specific questions. Some of these questions and comments also provide the opportunity for the developers to file bug reports with the core team when necessary.

7.3.3 Training

The CyanogenMod project provides training for end users and developers in order to improve their programming skills to allow them to modify the Android Operating System and then contribute to the project. The training is targeted towards users who have an interest in becoming contributors, and developers who want to port to a new device or submit a patch to the core team. The majority of the training is provided through the official wiki in two dedicated sections, the first called *Support*, and the second called *Learning Center*, which discusses basic concepts, tools, processes, and rules of contributing.

The CyanogenMod project offers three types of training; Beginner, Intermediate, and Advanced. The Beginner Training looks to get people to understand the basics of the ROM and how to modify it, giving users an introduction to the tools and terminology. Intermediate Training then explains how to use the different tools for Android development and the processes for contributing to the project. Finally, the Advanced training deals specifically

with porting the ROM to new devices and submitting the port to the project.

Beginner Training

Beginner training is aimed at the individuals who are not tech savvy or who do not have much knowledge about the processes and tools involved in Android development. The main part of this is the "Basic Concepts" section in the official wiki, which looks at the concepts that are most commonly used in terms of language in the project. This section goes into a great deal to explain basic concepts such as *rooting*, *jailbreaking*, *side-loading*, and other concepts that are key to the process of installing CyanogenMod. Beginners' training is therefore focused on two things, getting users acquainted with the terminology which covers key processes, and the tools that are used for development.

The information contained in the Beginner's Section aims to inform the reader about key processes that are needed to install the ROM. The beginners section does not contain the explicit procedures on how to root or flash, focusing more on the different methods of achieving it and the tools that are required. The section discusses issues such as partitions and the different types of memories, like RAM and flash storage, while not providing information regarding how to manipulate these. Finally, it explains the different forms of unlocking that can be carried out on a phone, such as SIM unlocking so that it may be possible to use other carriers, and bootloader unlocking so that you can install and run a modified ROM.

Beginner training also looks at some of the key tools in Android development, such as the Android debug bridge (adb) and the fastboot, both of which can be found in the official Android software development kit (SDK). The focus is to explain the functionality of these tools, such as the Android debug bridge, which is the software that allows the Android device to connect and communicate with the PC. The fastboot is a very basic software in the device which allows it to check the hardware that is attached to it, and will then load the operating system. It is this fastboot which can also determine which operating systems can and can not be loaded. These tools are contained within the Android SDK, with the purpose of allowing individuals to test and debug Android apps, but the tools described

above can also be used to help solve issues when modifying the software in the devices.

Finally, the last step of the beginner training includes setting up a programming environment which includes a virtual machine to simulate Android on a PC, as well as connecting to the source code. The setting up process also includes introducing the concepts of the way that Git and GitHub work[12] and tracking the software and downloading it to the computer. It has to be noted that the majority of the instructions for this sections deal with getting the individuals to be able to modify the operating system and to become acquainted with the tools to do so. The focus of the training at this stage is therefore to understand the way that Android modification and the relevant tools operate and how they can be used within only the PC of the user, not interacting with other developers.

Intermediate Training

Intermediate training covers a number of things such as using the above tools for modifying the software, and then goes on to explain how developers can contribute to the project through the code review site Gerrit. Intermediate training begins by teaching users how to build the operating system from source code, and as a result, how to add new features or default apps to it. The focus of the intermediate training is therefore to start looking at how individual users can change the operating system, by adding their own apps, fixing or improving features that already exist, as well as by bundling different apps. The other form of contribution that will fall under intermediate programming training is the translation which also requires knowledge on how to use the repository and submit patches.

During intermediate training, the tutorials cover only the modification of local source code, and therefore does not include collaborative development with others in the project. This requires the use of Git and GitHub, as well as knowledge on how these are linked with the main source code. The focus of this is to let users know how they can manage the source code within their computer, and modify

[12]Git is the technology that facilitates a decentralised development process, while GitHub is a website which hosts projects and uses Git as its underlying technology.

it. Then, once it is modified, it is all about being able to compile it (change it to binary), and deal with any errors that may arise in the process.

Another thing that is included in the intermediate training is the use of the Gerrit review, which is where all possible patches and improvements are submitted to. The patches and improvements are able to be submitted to the Gerrit review and can sometimes be as a response to a bug report being filed. The other use of the Gerrit review is to submit a better implementation of a feature that already exists, and allow the core team to review it. The documentation shows that Utakos is the user who is responsible for reviewing the source code submitted and either approving it or rejecting it.

The final part of the intermediate training is about translating the operating system and the apps, which also requires knowledge and use of GitHub. The translation has to be done in a separate file for each language, that file is then called by the software as it needs it. The translator will therefore have to know where this file is, how to change it, and then how to submit that change to the repository. Translations are therefore submitted just as any other code contribution, using the same tools.

Advanced Training

The advance training focuses on two things, the first is on how users can port to a new device, and the second is on how the port can become an official part of the project. The rest of the advanced training also covers issues such as the way that developers can open their own port so that other developers can contribute. Finally, it also looks to teach individuals how to carry out more technical tasks such as setting up a secure communication line to the repository.

It is clear within the instructions that porting for any device can only be done by an individual if they have already been able to build the operating system for another device. This process is what is covered in the intermediate training, where the purpose is to learn more about the things that are involved when compiling an operating system. At the same time, another prerequisite of porting is also to be familiar with the source code, the vendor modifications, the device manufacturers, and the actual kernel.

Familiarity with the directory is needed when placing the files in the directory which are required to modify the operating system and make it work with new devices. The main reason behind knowing more about the directories is that it is possible to then understand which files need to be changed. The wiki contains a description of the types of changes that need to be made to most of the files in each directory. Once the changes have been made, the source code will then have to be built and then installed to the operating system in order for it to be tested.

7.4 Development Process

The previous two sections looked at the different products and services provided by the CyanogenMod project, finding that there were three software products produced by the project. From those, only the CyanogenMod ROM is open for user contribution, where users are able to submit code changes, translations and then are able to port it to different devices.

This Section focuses on the development process of the CyanogenMod ROM, looking at four key stages in the development, Nightlies, Relsease Candidates, Stable Releases, and porting. The results find that different types of activities dominate each of the stages, therefore requiring different types of skills and probably different groups of individuals working on them. Subsection 7.4.1 looks at the nighly stage of the of the development process, focusing on the rules of who can participate, the objective of this stage, and how it is supported throughout the project. Subsection 7.4.2 talks about the second stage, the Release Candidate, looking at the objective and focus of this stage and how general users are engaged. Subsection 7.4.3 then looks at the final stage of development which is the Stable Release of the software. Finally, Subsection 7.4.4 will look at the activity of porting the ROM to numerous devices, how it's done, and how it fits into the process.

7.4.1 Nightlies

The first stage of the process is to modify the AOSP and release the modifications as often as possible. This is done through nightly builds[13], where the focus is on fixing major kernel bugs and implementing features that are not device-specific, such as power management, memory allocation, and user interface customisation. The nightly builds are unsupported, meaning that there is no community support or guarantee they will work properly. Some of these features would have already been used in the previous versions of the ROM, in which case smaller modifications are made so that these still work in the new ROM. This initial development stage will carry on until all new features have been implemented, all major bugs have been, fixed and the build reaches a minimum level of stability, after which the core team freezes development and distributes a Release Candidate for testing purposes, allowing further time for submitting bug fixes, device support, and translations.

During this first stage of development, the core team, consisting of key members like Koushik Dutta and Steve Kondik, will be involved in the modification process and will not take any form of bug reports from users. This initial stage is used by the core team as a testing version, where new features are typically implemented. Because many of the features may not work, the nightlies are purely experimental and the core team ensure that the users are aware of the risks. At the same time, because it is experimental, the core team does not accept bug reports, in order to focus on feature implementation and to speed development.

Because the nightlies are released to the general public, there are still some early adopting users who may provide some informal feedback. Despite the development of the nighlies being restricted to the core team, their release makes it possible for users to download and install them. The process of installing nightlies, however, is not automated, and so users need to know the process of installing it manually. Finally, although there are no bug reports allowed during the nightly stages, the core team can informally obtain feedback

[13]Nightly builds are informal release stages that show the present stage of development. Since the source code needs to be compiled to machine language, these builds are compiled once a day when development has finished.

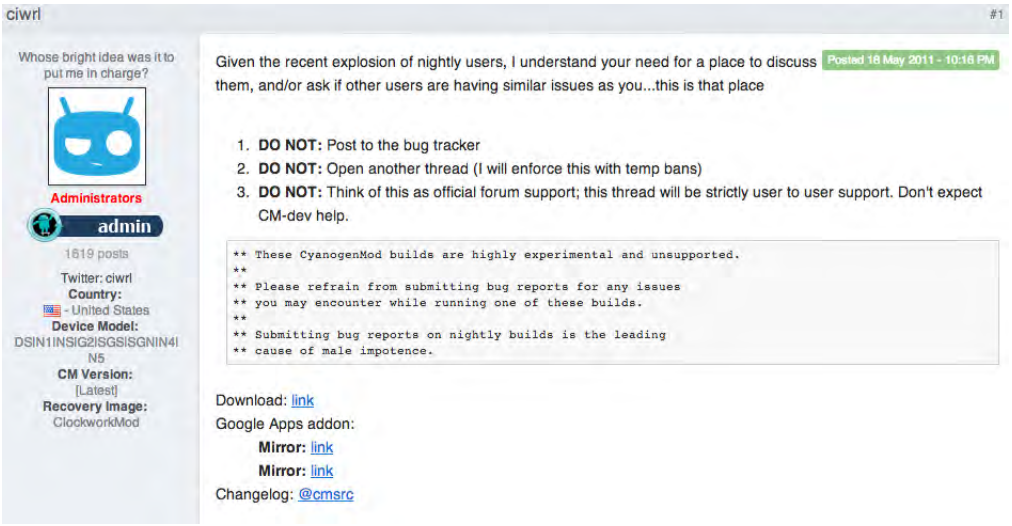


Figure 7.5: Forum conversation regarding nightly releases.

from forum discussions, alerting them of possible issues. Typically, a member of the core team will open a forum discussion on nightlies giving basic information. Figure 7.5 is an example of a forum post made by Abhisek Devkota talking about nightlies with instructions about bug reports.

Although the Nightlies are the first step of development, this experimental stage is continuously used thought the project's life cycle as an experimental platform. Depending on the changes made to the AOSP, the duration of the nightlies may vary. For instance, after the release of Android 4.0, which saw major changes to the operating system, the CyanogenMod team took 8 month to move from the Nightlies to the first Release Candidate. Where the change in the AOSP is minor, as was the case with Android 2.3, this can be reduced 2 months. Even after the project has moved to a Release Candidate, the core team will still use nightlies to implement new features.

7.4.2 Release Candidate

The project documentation describes the Release Candidate as being "[...] feature-complete and fairly well tested, but [with] some minor tweaking that needs done". The main objective of the Release Candidate is to provide developers with a stable version of the new ROM with the new features, with the focus moving from implementing new features to fixing and stabilising the

software so it can be ported to other devices. The release candidate (RC) is a near complete stable version of the modified ROM, which developers can use to test the new features, add device-specific support, and translate to other languages. As with traditional software development processes, CyanogenMod uses a RC to give device maintainers and developers early access to the latest ROM, so that their software or device is supported before the release of the final stable ROM. It also gives the opportunity to fine tune the ROM through testing, minor big fixing, and translating the user interface to other languages. Since each developer works on their own repository, the device-specific modifications are not merged with the master repository, but instead each branch carries its own evolutionary path throughout the rest of the development process including the final release stage. Once the final modifications have been made, and the RC is deemed stable enough for daily use, each branch releases its own official CM stable version.

During the release candidate stage, development is opened to those outside of the official core team so that bug reports and fixes can be made. During the RC stage, users and developers are asked to contribute with bug report and fixes through Gerrit review, although some informal reporting takes place through the forum or social media. The Release Candidates are the first stage at which other developers can contribute to the project by submitting formal bug reports to the project's Gerrit website. These reports and fixes are then reviewed by the official core team and are either approved or rejected, in the case of fixes, or delegated as work in the case of bug reports. Informal reporting also takes place at this stage between general users and device maintainers through forum discussions. Device maintainers open discussions on the forum based on the RC for the device they maintain, similar to the nightly forum discussions (Figure 7.5).

The duration of the RC stage and the number of RC releases depend on how problems are fixed and on the actual device it is installed on. The objective of the RC is to offer developers and maintainers a stable version of the software so that they can be ported to new devices. As a result, the device maintainers will fork the main repository and add their specific modifications so that they work on their device. These modifications are solely based on making sure that the

hardware and software are working properly on each of the devices, and that all major device-specific bugs are covered. As a result, the duration and the number of RCs will differ between devices. For instance, the device Huawei U8500 went through 8 release candidates for CM10.1, while for the Nexus 5 it went through 4.

7.4.3 Stable Release

The third and final stage of the process is the release of the stable version, where all experimental and preliminary maintenance work has been carried out. The CyanogenMod project describes the stable version as the only one they would recommend for daily use, without fear of losing any data. Even at this final stage, bug reporting and bug fixing will continue, with built-in features in the ROM allowing non-programmers to directly send bug reports to the core team. In addition, if a new device is not officially supported, users are encouraged to use the stable version for porting the ROM to untested devices. All the official builds – from Nightly to Stable – are available for downloading at the official distribution website[14].

Typically, by this point, the core team are involved in the development of new features or on working on the next version. The stable releases are maintained primarily by device maintainers and by the community, contributing mainly through minor fixes or improvements to the code. At the same time, the core team will be working on the nightly releases for new versions or with a view of implementing new features.

The stable releases are fully supported by the project, which means that there will be ample documentation and tools that can help users with the installation and any other problems they may face. The installation software that helps users install the ROM on their device, will automatically access the stable release of the ROM. In addition, all the tutorials on how to use the software, as well as any other issues, will be covered by the team for the stable release only. Finally, the documentation found on the wiki, as well as a large section of the forum discussions on use, will focus only on the stable version. This release is therefore

[14] `http://get.cm/`

focused completely on the end user, with the assumption that both the nightly and the release candidates are for the more advanced users.

7.4.4 Porting

Porting is the process of modifying the ROM, by adding hardware drivers and changing settings, in order to make it work on other devices. The porting process happens parallel to the other stages although it is typically advised to be carried out at the end of the development cycle when the Stable Release is made available. The purpose of this stage is to allow users to modify the stable release by adding device specific drivers and settings so that it can work.

The core team members modify the AOSP and release it only for a small number of devices, which are typically in the Nexus range or the most popular high end devices like the Samsung Galaxy range. The Nexus range are ones that are made by Google in direct cooperation with manufacturers such as HTC, LG, or Samsung, and are targeted towards developers, offering them fewer restrictions in terms of development options. In addition, the AOSP is designed and tested by Google using the Nexus range of devices, and therefore becomes a de-facto standard for developers to use when writing Android apps or when modifying the ROM. Other well-known and popular devices, particularly the Samsung Galaxy range, are also some of the first to receive the modified CyanogenMod ROM during the early stages of development.

Porting therefore becomes important for the project because it allows the ROM to be installed in other devices, therefore making in available to a wider audience. Porting is typically done by general users, and it is important part of the process as it helps the core team overcome problems with access to the hardware, with Community leader Abhisek Devkota (user name *ciwrl*) stating on a blog post that the core team "... can only work on the devices [they] own." On a forum post, forum moderator Jeremy Hansen (user name *bassmadrigal*) explain

> CM devs are consumers first. What this means is that they do not divide up devices among other developers, or assign devices like one would at a job. Developers work in their spare time without monetary compensation.

> Because of this, the developers are free to work on any device they choose to purchase.

Porting the ROM to new devices can occur at any time during the development stages, where as soon as the Nightly Releases are made available, general users and device maintainers can begin the porting process. The importance of using them at this stage is that it allows the device maintainers to make the necessary changes and possibly solve any problems with the hardware before the final Stable Release is made available. As mentioned before, the CyanogenMod team advices device maintainers to use either the Release Candidates or the Stable Releases when porting to a new device, as it offers more stability through the process. In practice, however, experienced maintainers start the process with the Nightly releases.

7.5 Chapter Summary

This chapter aims to understand the CyanogenMod in terms of the products and services that they offer. The results show that the project is responsible for developing three main pieces of software, which include the modified ROM, a ROM manager, and easy to use installation tool. The source code for the ROM and the installer are both open and available to the public, while the installer is closed source. In addition, general contributions are accepted only for the ROM, while the ROM manager and the installer are both maintained by the official core team.

The results also show that the project offers three key services, an device account manager, user-to-user assistance, and ROM development training. The device account manager, which allows individuals to find or erase lost of stolen devices remotely, is built on open source technology and was fully developed by members of the core team. The user-to-user assistance takes places in various locations, predominantly through the forum and the IRC, while the training takes place primarily through the wiki.

For each of the products and services that are available through the CyanogenMod project, there are three key areas through which contributions

and interactions take place: the repository, the wiki, and the forum. The CyanogenMod GitHub account contains the repositories for the software that is maintained by the project, including the ROM, installers, and the website. The wiki not only contains training tutorials, but also contains device-specific information for users wishing to install the ROM. The forum is fully dedicated to offer user-to-user assistance and feedback for the project.

Finally, the project has three main development stages, Nightly, Release Candidate, and Stable Releases, with each of these aiming to achieve different objectives. Each of these signals a different stages of software readiness, but they are not sequential and can take place parallel to each other. Finally, porting the ROM to new devices presents a final stage of development, where the centrally modified ROM is copied and made to work on specific devices.

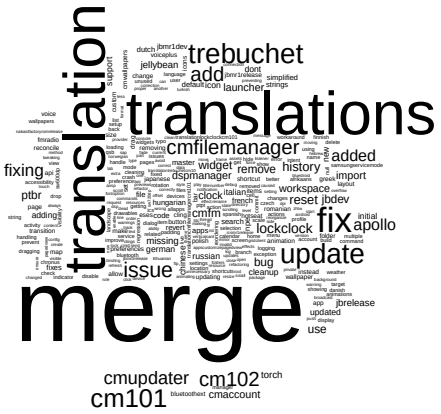
Chapter 8

Contributions to the Project

8.1 Introduction

This chapter builds on the results in the previous chapters by determining how users can contribute to the project and who the key contributors are. Chapter 7 looked at the products and services provided by the CyanogenMod, noting that contributions to the project were made through the repository, the wiki, and the forum. Section 7.4 then finds that the groups of individuals who are active in the development of the ROM differ depending on the stage of development, with the official core developers being more involved at the experimental stages and the peripheral contributors being more active during the release candidate stage. This chapter looks at the types of contributions that are made through the repository, the wiki, and the forum.

The first part of this chapter, Section 8.2 looks at the different ways in which users are able to directly contribute to the project, focusing on the different non-managerial activities carried out by users. The observations show that in both the repository and the wiki, there are a number of different activities that take place besides programming or page editing. In addition, it was also observed that the majority of these contributions took place at different time periods rather than parallel to each other. With regards to the forum, results show that forum discussions revolve around two stages of ROM development. The first being the discussion on the Nightly releases, which tend to be based on user-to-user assistance that is specific to solving issues of compatibility with the hardware.



Translation	1,993 (0.399)
translation	*1,010 (0.202)*
translations	*977 (0.196)*
translated	*6 (0.001)*
Merge	1,680 (0.337)
merge	*1,678 (0.336)*
merges	*2 (0.000)*

Figure 8.1: Most commonly used words in the repository.

The second set of discussions revolves around Stable releases, which tend to focus on issues with the application software and installation problems.

The second part of this chapter, Section 8.3, identifies the key contributors using two approaches, through the official documentation and roles within communities, and through patters of participation. The official documentation reveals that there are 90 contributors who have been classified as being key members of the team. The patterns of participation, however, show that there are a different set of users who are classified as being highly active and therefore key to the project. Finally, official managerial roles at the community level are sometimes held by individuals that have low levels of participation.

8.2 Types of Contributions

This section recounts the observation on the different types of contributions made to the project. The results show that the repository and wiki see multiple forms of contributions besides programming and editing. For the repository, we see a larger set of activities that include translations, graphic design and programming. In the wiki, we see two forms of contributions, information contributions and the maintenance of the infrastructure. For the forum, only user-to-user assistance was observed, where participation was divided into two general level of technical ability.

8.2.1 Repository

The results show that there are three main activities for the project; merges, translations, and programming. In the repository, there are more programming contributions being made, followed by merges, and then translations. The programming contributions are made by the larges group of individuals, followed by translation, and then merges. The data shows that the activities occur at different times of the development cycle, with programming activities taking place at the start and translations taking place towards the end. In addition, it was observed that there are fewer people who merge programming changes to the repository, with the majority being able to merge translations only. Finally, the merging activity gradually moves from merging programming contributions at the start, to then merging translation contributions at the end, with different groups of individuals being involved in each.

From the text analysis of all the titles, the changes made to the repository fell under three main categories; merges[1], translations, and programming. In the CyanogenMod project, contributions to the repository contain titles that indicate the type of changes being made. For example, some contributions contain the word 'Merge' at the start of the title, indicating that a previous contribution has been made part of the software code. Equally, translation changes are marked with the word 'Translation', followed by the language code, such as EN for English. Finally, the programming contributions were then labeled as such by the lack of either 'Merge' or 'Translation' or their related words in the title. Figure 8.1 shows the most commonly used words in the submission titles.

In total there were 4,992 changes to the repository made by a total of 269 contributors, with the most contributions being programming, followed by merges, then translations. From the total number of changes made to the repository, 2,062 (41.3%) were classified as being programming, and were submitted by 174 (64.7%) of the total number of contributors. A total of 1,656 (33.2%) of all submissions were classified as being merges, which were

[1]A *merge* occurs when new code is integrated to the central repository, and requires administrative access.

Group	Users	Translations
Core Team	3 (0.022)	3 (0.002)
General Developers	3 (0.022)	11 (0.009)
Maintainers	8 (0.060)	32 (0.025)
Translators	15 (0.112)	220 (0.173)
General Contributors	105 (0.784)	1,008 (0.791)
Total	134 (1.000)	1,274 (1.000)

Table 8.1: Translation Submissions to the Repository by Group

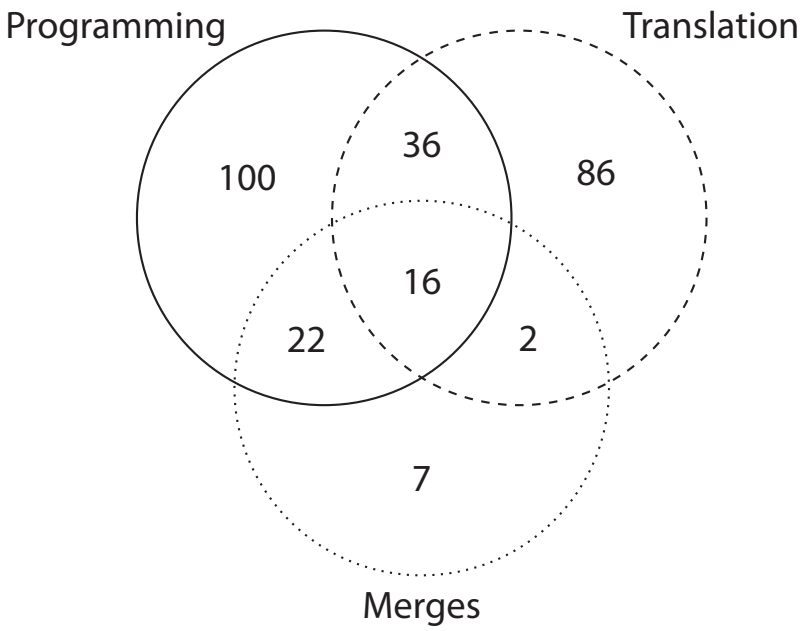


Figure 8.2: Number of users involved in each activity.

submitted by 65 (24.1%) contributors. Finally, 1,274 (25.5%) of submissions were translations, submitted by 140 (52.0%) of all contributors. There were a total of 76 (28.3%) users participating in more than one activity, with the largest overlap being between programming and translation, where 36 (13.4%) contributors were active in out both (see Figure 8.2). Finally, there were 16 individuals who carried out all three activities, out of which 8 had formal administrative roles within the project.

A significant percentage of the programming merges were made before the first stable release was made. Figure 8.3 shows the percentage and total number of contributions made to the repository at certain milestones in the project. For

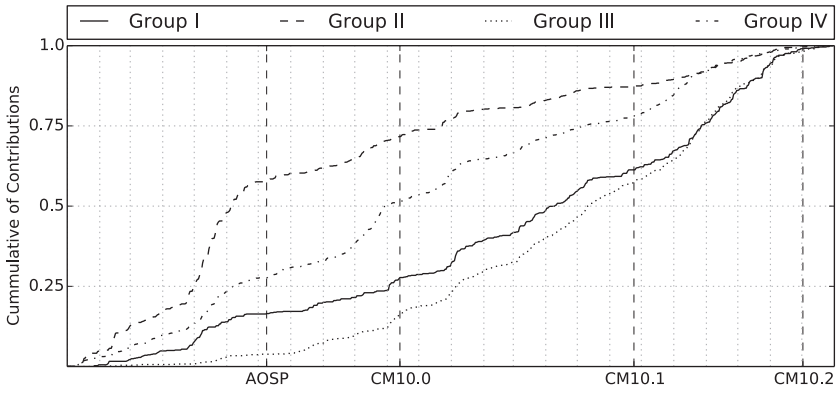


Figure 8.3: Cumulative Contributions for All Groups. Group I - Translation Merges, Group II - Programming Merges, Group III - Translation Contributions, Group IV - Programming Contributions.

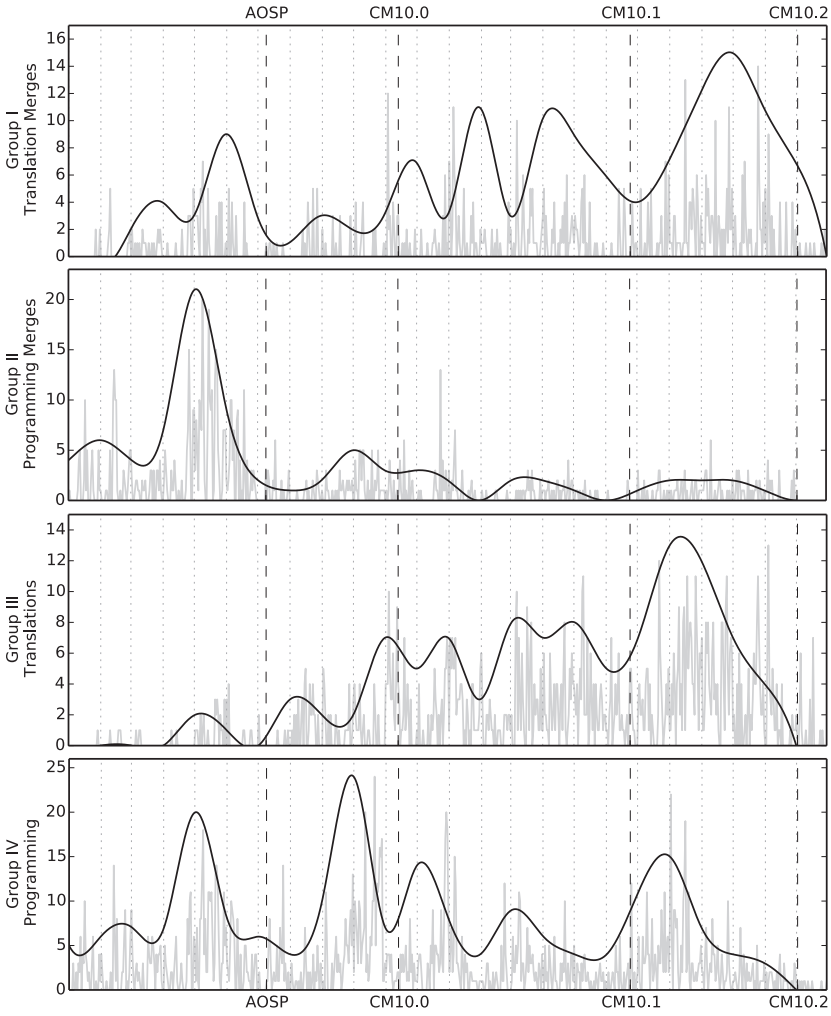


Figure 8.4: When each activity happens

	Group I	Group II	Group III	Group IV
2012-01	0.5 (4)	4.9 (45)	0.1 (1)	3.0 (62)
2012-02	2.2 (16)	12.8 (117)	0.4 (5)	6.5 (133)
2012-03	4.9 (36)	16.8 (154)	0.6 (8)	9.7 (200)
2012-04	7.7 (57)	25.4 (233)	0.8 (10)	13.5 (278)
2012-05	13.9 (103)	48.0 (440)	3.1 (39)	23.4 (482)
2012-06	16.4 (121)	57.6 (528)	3.7 (47)	27.5 (568)
AOSP	**16.5 (122)**	**57.9 (531)**	**3.8 (49)**	**27.7 (572)**
2012-07	17.2 (127)	60.2 (552)	4.1 (52)	30.8 (635)
2012-08	20.0 (148)	61.9 (568)	7.3 (93)	32.9 (679)
2012-09	21.5 (159)	64.8 (594)	9.7 (124)	38.2 (788)
2012-10	23.8 (176)	70.4 (646)	12.2 (155)	50.1 (1,034)
CM10.0	**27.7 (205)**	**71.6 (657)**	**16.5 (210)**	**51.4 (1,060)**
2012-11	28.7 (212)	73.7 (676)	18.6 (237)	53.5 (1,104)
2012-12	32.2 (238)	77.4 (710)	22.9 (292)	60.6 (1,250)
2013-01	39.4 (291)	80.2 (735)	30.1 (383)	64.7 (1,334)
2013-02	41.8 (309)	80.7 (740)	32.6 (415)	66.6 (1,374)
2013-03	48.7 (360)	83.0 (761)	39.9 (508)	71.3 (1,471)
2013-04	54.7 (404)	85.9 (788)	46.6 (594)	74.5 (1,537)
2013-05	59.1 (437)	87.0 (798)	53.9 (687)	76.4 (1,575)
CM10.1	**61.3 (453)**	**87.5 (802)**	**57.6 (734)**	**77.9 (1,607)**
2013-06	62.2 (460)	87.5 (802)	58.3 (743)	78.8 (1,625)
2013-07	67.3 (497)	89.5 (821)	64.5 (822)	84.7 (1,747)
2013-08	75.9 (561)	92.1 (845)	76.8 (979)	92.2 (1,901)
2013-09	86.1 (636)	95.1 (872)	87.2 (1,111)	95.8 (1,975)
2013-10	94.6 (699)	97.9 (898)	93.9 (1,196)	98.1 (2,022)
2013-11	99.1 (732)	100.0 (917)	98.3 (1,252)	99.6 (2,054)
CM10.2	**99.1 (732)**	**100.0 (917)**	**98.3 (1,252)**	**99.6 (2,054)**
2013-12	100.0 (739)	100.0 (917)	100.0 (1,274)	100.0 (2,062)

Table 8.2: Percentage of Submissions. Group I - Translation Merges, Group II - Programming Merges, Group III - Translation Contributions, Group IV - Programming Contributions.

programming merges (Group II) we see that 51.4% of all merges had been made before the release of the first stable version CM10.0. In Figure 8.4 we see that this increase coincides with two spikes in programming contributions, the first one preceding the release of the AOSP, and the second one preceding the release of the first stable version. The first spike in activity occurred between April and June 2012, where a total of 436 programming changes were made by a total of 43 contributors, of which 11 were also part of the projects core development team. During this period, the word analysis showed that 54.7% of contributions were programming related, with the most common words used being 'issue' (101 times), 'fixing' (72), 'widget' (43), 'release' (38), 'adding' (22). The second period of high programming activity was between the middle of September to the end of October in 2012, just before the release of the version CM10.0. During this period there were a total of 320 contribution made by 34 contributors, of which 6 were official members of the project. The text analysis also showed that 40.4,% of contributions were programming related, lower than the previous period of high activity, with the top words being 'issue' (24), 'cleanup' (18), 'actions' (15),

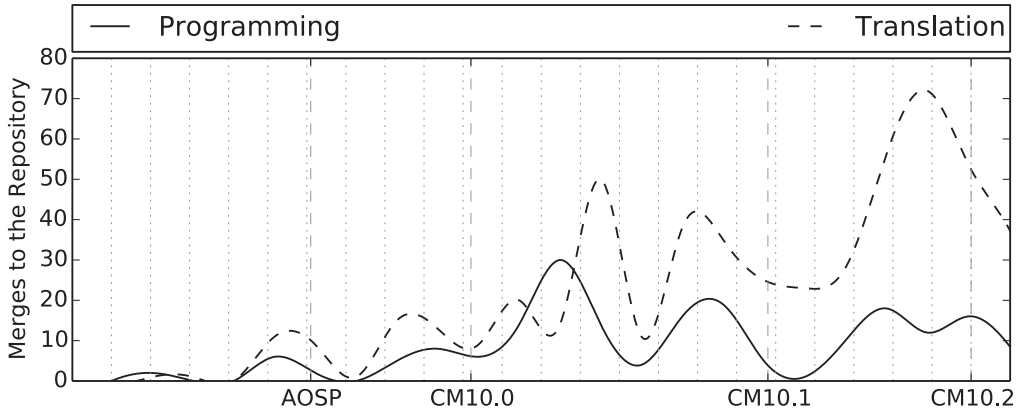


Figure 8.5: Merges to the Repository by official members

'search' (13), and 'strings' (11).

Another significant observation was that translation contributions took place towards the end of the development rather than at the start, with a major peak occurring soon after the release of CM10.1 During the development period, there were a total of 140 individuals contributing through translations, out of which 86 (61.4%) did not contribute through any other way. From Table 8.2 we can see that the translations were mainly added to the repository at a later date than the programming contributions. For instance, it took approximately 10 months for half of all programming contributions to be submitted, and it then took a further 14 months for the second half. In contrast, it took approximately 17 months for 50% of the translation contributions to be submitted, with it then taking another 7 months for the resulting 50%. This means that, there was a mean average of 40.41 translation submission in the first 17 months, which then doubled to 83.86 in the final 7 months.

Finally, further analysis of the merges taken place show that significantly more individuals can merge programming changes as opposed to translation changes. In total there were 47 individuals who had merging permission for the repositories, out of which 3 (6.4%) members did translation merges only, 32 (68.1%) did programming merges only, and 12 (25.5%) did both. From the 12 individuals that carried out both merges, 8 of them were listed as being part of the official CyanogenMod project.

	Programming	Translation	Total
Core	312 (0.443)	12 (0.051)	324 (0.344)
Middle	216 (0.307)	46 (0.194)	262 (0.278)
Periphery	176 (0.250)	179 (0.755)	355 (0.377)
Total	704 (0.748)	237 (0.252)	941 (1.000)

Table 8.3: Code Fixes by Layer

	Programming	Translation	Total
Core Team	42 (0.223)	2 (0.032)	44 (0.175)
General Developers	76 (0.404)	6 (0.095)	82 (0.327)
Maintainers	70 (0.372)	7 (0.111)	77 (0.307)
Translators	0 (0.000)	48 (0.762)	48 (0.191)
Total	188 (0.749)	63 (0.251)	251 (1.000)

Table 8.4: Code fixes by official role

Code Fixes

From the 4,992 submissions to the repository, 941 were classified as being fixes, with 704 (74.8%) of those being programming fixes and 237 (25.2%) of those being translation fixes. Table 8.3 shows the number of fixes submitted by each layer along with the break down if they were programming or translation changes. All three layers posted relatively equal numbers of fixes to the repository, with the largest number being submitted by the periphery with 355 (37.7% of all fixes), and the least by the middle, with 262 (27.8%). When this is broken down, however, we see that just over half of the fixes submitted by the periphery were translations, while the core submitted the largest number of programming fixes.

When the fixes are broken down according to the official roles in the project, however, we see that the number of fixes submitted by official members is 251 (26.7%) of the 941 total fix submissions. Table 8.4 shows the number of fixes made by each of the groups in the official project members. From the results we can see that the Maintainers contributed the largest number of bug fixes to the repository with a total of 82 (32.7% of the total), while the official core team submitted the fewest number with 44 (17.5%). It is also interesting to note that all groups in the official project members also contributed translation changes, with the translators submitting the most. Finally, the translators did not appear to submit any programming fixes.

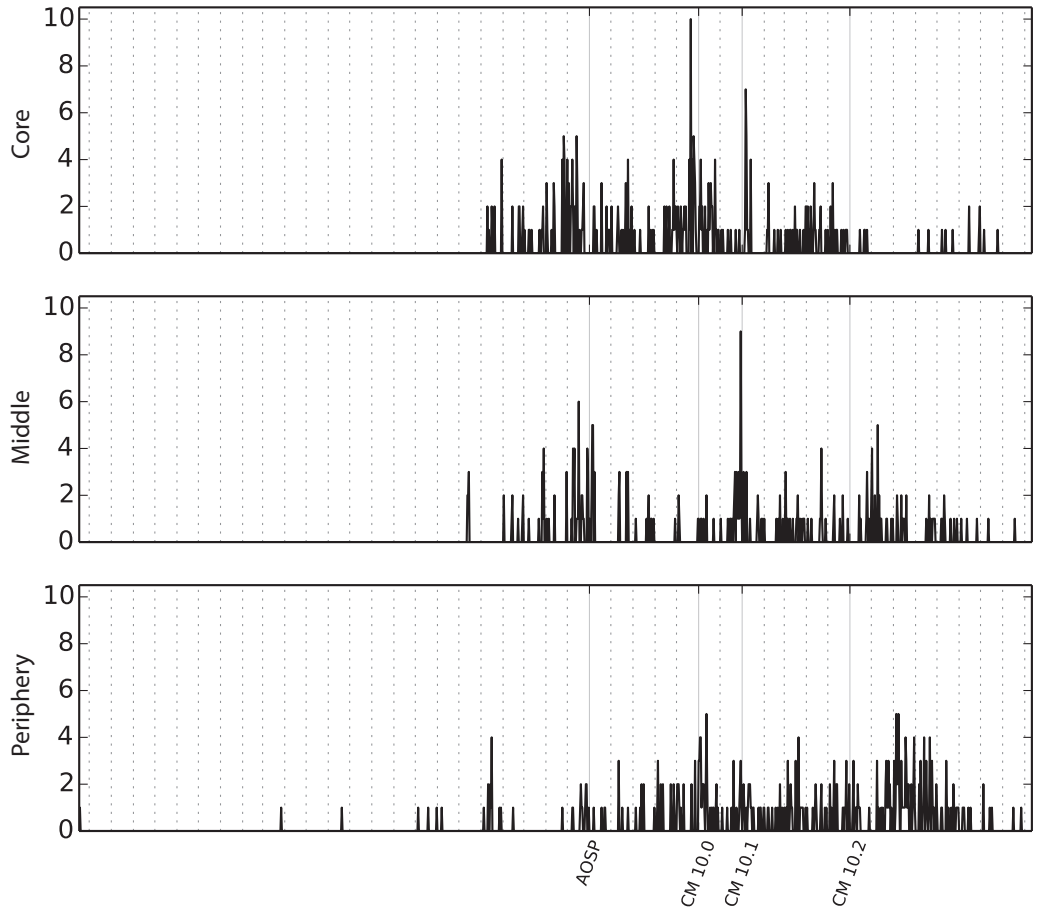


Figure 8.6: Fixes submitted by layer

8.2.2 Wiki

The patterns of activity show that the official wiki editors are responsible for initiating wiki pages and contributing a significant portion of the information. There are two main activities in the wiki section, the first the page initiation and the second is editing. The initiation stage of the wiki is when a user starts a new page, an activity that must also be accompanied by adding information to the page. Editing, on the other hand, is when the page already exists and information is then added or removed, it could signify both an update or minor fix.

From the 1,344 pages that were edited, 251 of them were translations of other pages, therefore leaving 1,091 of pages that contained unique information. In total there were 129 users which initiated a page, with only 4 of those being official wiki editors. From 1,091 pages, the official editors initiated 768 (70.4%) pages,

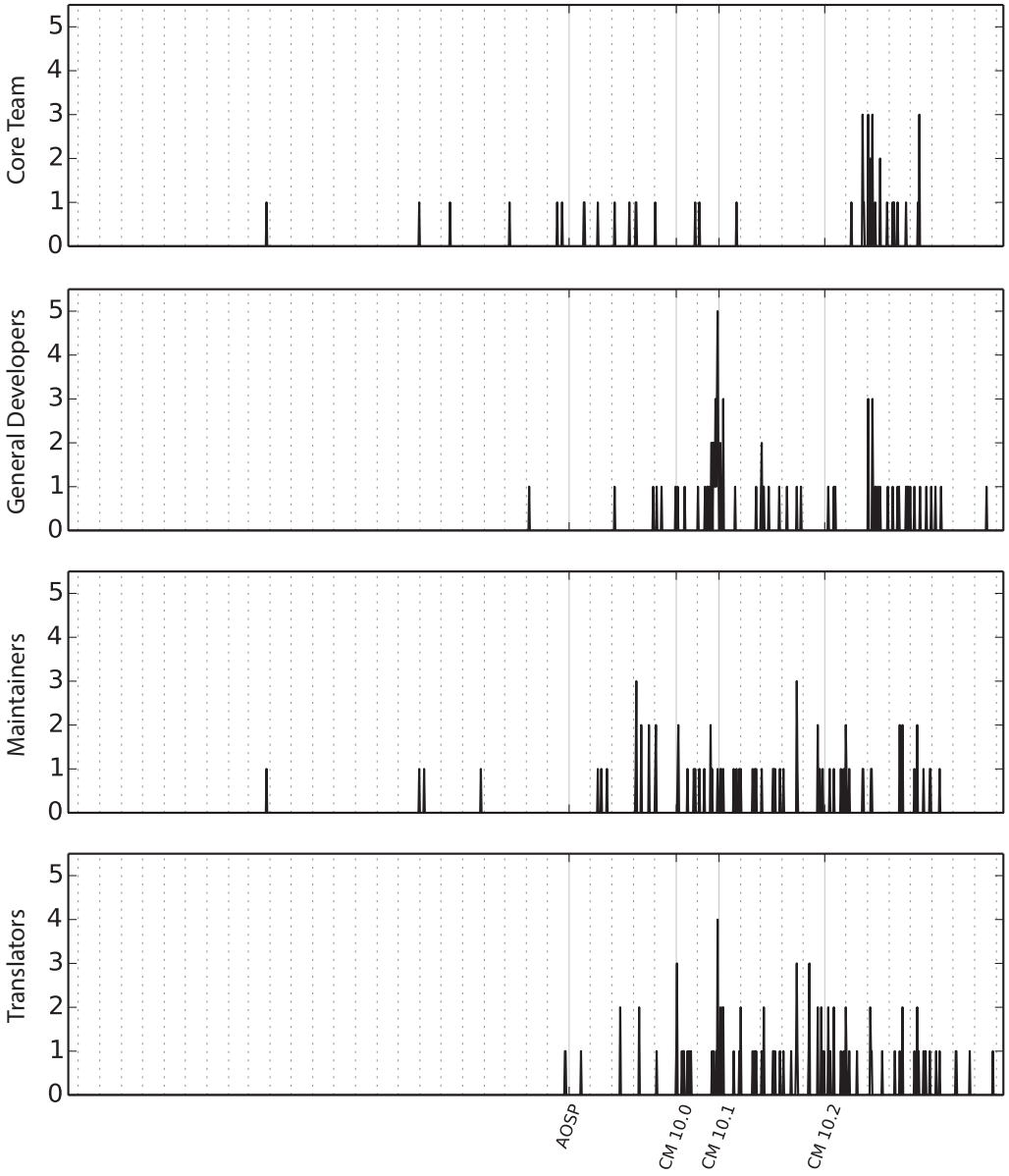


Figure 8.7: Fixes submitted by official role

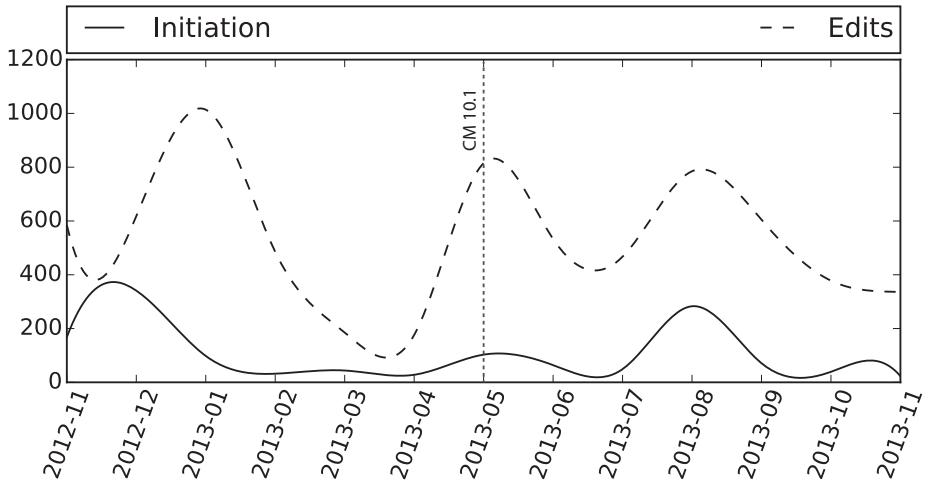


Figure 8.8: Wiki page initiation and edits.

with the remaining 323 (29.6%) pages being initiated by general contributors. On average, the number of characters added during the initiation of a wiki page was 537.88 and 460.86 for official editors and general contributors respectively. In contrast, the average number of characters during the editing phase was 182.44 and 196.61 for official editors and general contributors respectively.

The four official editors were primarily responsible for developing page templates supported devices, which includes instructions on how to install the ROM and any known issues. For most of these pages, therefore, the initiation process of these pages consisted of copying and pasting similar information from one page to another, which was then changed slightly to focus on any particular device. The templates themselves already contained an average of 2,944.53 characters, therefore increasing the average amount of information being measured. In sharp contrast, the general contributors to the wiki would edit the pages by adding relatively smaller amounts of information, with an average of 196.57 character changes per page.

Figure 8.8 shows when the page initiations and subsequent page edits have taken place. The graphs show a significant spike in editing activity shortly after a large number of page initiations took place, between December 2012 and November 2013. A second significant spike took place in May 2013, during the release of of CM version 10.1.

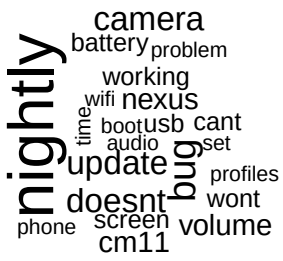


Figure 8.9: Word cloud for discussions on Nightly releases.

8.2.3 Forum

Contributions through the forum are varied in terms of subject and length, and are structured around the project's development stages. The threads are divided into two different sections, the ones that are related specifically to the Nightly versions, and others that are focused only on the Stable version. There were more discussions in the Stable (1,879) section that there were in the Nightly (92).

Discussions on the Nightly builds were mainly focused on issues that may have emerged while porting to new devices, particularly issues with hardware compatibility. Figure 8.9 shows a word cloud of the discussion titles in the Nightly sections of the forum. A significant number of titles were focused on problems that were specific to hardware and to bugs in the software. Most of these seem to be issues with the porting process, where hardware problems are more likely to surface.

Some of the representatives discussion titles for the Nightly section were

— "Annoying Camera Bug"
— "Camera Bug"
— "Problem With Front Camera"
— "Poor Battery Life With High Android OK Keep Awake Time"
— "Media Server Battery Drain"
— "Wifi Analyzer Doesn't Work"
— "Wifi Direct Cause Reboot"

In contrast, the discussions on the Stable version were mainly towards the usage of the ROM and other software problems. Figure 8.10 shows the word

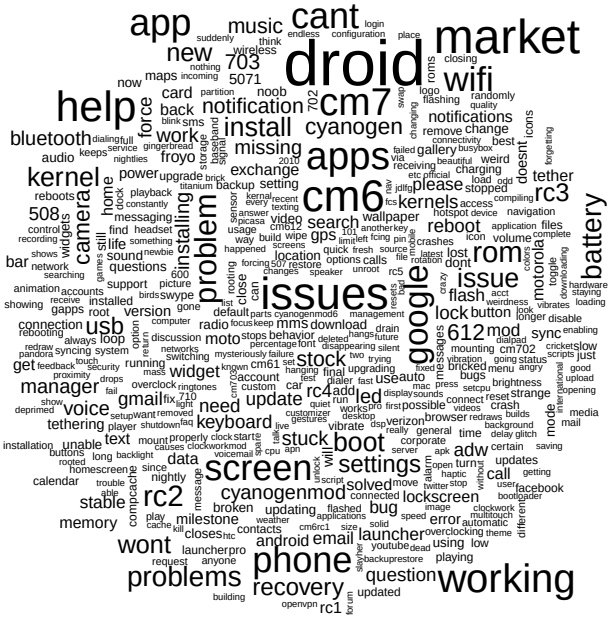


Figure 8.10: Word cloud for discussions on Stable releases.

cloud of the discussion titles for the Stable version of the ROM. The discussions were focused mainly on the usability of the software and issues with apps. Some of the representatives discussion titles for the Stable section were

— “Best Performance out of the Droid?”

— “I Find 703 to be Unstable on my Droid”

— “Changed ROMs and now Missing Market”

— “App Moved to SD Card Now Shows Update in Market”

— “Apps Randomly Closing”

— “Fresh Froyo Cannot Install any Apps”

The results may also suggest that the large number of discussions for the Stable version may be due to it having a larger install base than the Nightly version. The Nightly installs are typically done manually, meaning that there is some significant technical skill required when installing them. On the other hand, the Stable versions are typically installed by the supporting software, lowering the need for complex technical skills. In addition, the majority of the major bugs found in the Nightly version, particularly those relating to hardware, are fixed by

the time the Stable version is released. Meaning that other issues, particularly those relating to usage, may be reported and discussed.

8.3 Key Contributors

This section looks at key contributors to each of the subgroups. It focuses first on establishing who are the highest contributors by counting the number of times these have participated. It then looks at the official list of contributors as stated in the official documentation. Lastly it looks at the formal roles that have been set up and who they have been given to.

The comparison between the three sets of members shows that there are some differences between those who contribute the most and those who have formal roles within each community.

8.3.1 Repository

Key contributors to the repository were determined in two ways, first through the official documentation and second through participation metrics. The results highlight a difference between those who have been mentioned in the project documentation and those that have contributed the most to the development of the software.

Official Contributors and Members

The list of official project members states that there are 115 individuals that contribute to the project through the repository. The largest group is that of the official Maintainers, containing a total of 65 members, and are responsible for porting and maintaining the official ROM for various devices. The Maintainers will set up their own repository for a particular device, and be in charge of accepting contributions to those individual repositories. Some of the maintainers will be working on legacy devices[2], and therefore may not be involved in current versions of the ROM. At the same time, however, they are less likely to contribute

[2]Old devices that are unable to run latest versions of Android.

to the core ROM as they would be more focused on device-specific hardware support. As project leader Abhisek Devkota explains

> [Maintainers] are the folks that manage day to day upkeep of the devices themselves. These can be individuals, or groups/entities: ie FreeXperia for Sony devices, TeamHacksung for Samsung devices, etc. These folks work fairly autonomously and strictly on hardware support.

The second largest group is the Translation contributors, with 30 users being classified as official translators. These users will most likely be active in a number of modules, but primarily they will focus on translating the 13 modules that form the changes to the CyanogenMod ROM. In total there are 19 languages that are officially supported, including Simplified Chines, Japanese, Catalan, and Brazilian Portuguese. From the 19 languages listed, German has the largest number of contributors (6), followed by simplified Chinese (5), Portuguese (3), Russian and French (2), with the remaining 14 languages having only one translator. The Lead Translator is Marco Brohet, in charge of merging and keeping track of all the translations that have been submitted, explains the processes within the group of translators

> We have a private community on Google+, to become a member you must be invited: that is a community of CyanogenMod translators only. Some of us are developers too, and this comes in handy sometimes; but the posts are all about new commits in need of a translation, suggestion requests on translations, and rules about translations.

The last two groups consists of the General Developers and the Core Team. There are a total of 11 contributors who are classified as General Developers, who carry out a number of tasks such as writing or maintaining individual apps, the User Experience, as well as specialising in areas such as programming languages and the kernel. For instance, both Björn Lundén (user name *blunden*) and Nebojsa Cvetkovic (user name *nebkat*) are in charge of the User Experience and User Interface of the CyanogenMod ROM. Björn Lundén describes how he became involved with the project and what activities he carries out

> I found CM and started submitting small changes to it, often fixes for UI[3] stuff Google forgot to update. One day about a year ago [Abhisek Devkota]

[3]User Interface, includes graphics such as icons and fonts. Also referred to as *drawables*.

> asked me what I was working on and if I wanted to be invited into the private CM IRC channel. By that time I had already been submitting drawable fixes for a while and generally kept up to date with all details of CM. I have a formalized role as set up by *cyanogen* (project owner Steve Kondik).

The core team are then the smallest of the official contributors, with 9 members, including project owner Steve Kondik. The core team is composed of the individuals that have been part of the project from its days in the XDA forum, and are in charge of a number of activities but are mainly focused on the general development of the ROM and management of general contributors. The core team are the ones that implement new features and also set up the rules for contributing. Abhisek Devkota explains the role of the core team in assessing features

> First and foremost, we pay attention to the requests that pour in via our social media and blog. A lot of them are off-the-wall or items that will never be implemented (for technical or non-technical reasons), but regardless, its good to see what it is that is being requested. New ideas are then discussed, coded, and discussed some more until refined.

The core team are not just involved in coding new features, they are also responsible for merging existing projects into the ROM. An example of this is the Apollo music app, which was originally written by Andrew Neil, which was forked by the core team and made a default app in the ROM.

To summarise, according to the official documentation, there are four key groups that contribute to the project through the repository, translators, general developers, device maintainers, and the core team. Besides the translators, each of the groups carry out programming activities for the project but contribute in different ways. The device maintainers contribute by porting the project to other devices while the core team and the general developers work with the core CyanogenMod ROM. There was also a difference in contributions between the core team and the general developers, where the core team focused on implementing new features and the general developers on fixing or improving existing features.

	User	I	II	III	IV	Total
core	jruesga	363	14	38	77	492
	therbom	16	65	9	298	388
	winson chung	202	0	181	0	383
	Michael Jurka	163	0	156	0	319
Middle	dvtonder	103	9	46	74	232
	maniac103	99	19	33	69	220
	nebkat	170	0	38	5	213
	cyanogen	18	1	35	47	101

Table 8.5: Core Contributors Based on Participation. (I) Programming, (II) Translation, (III) Programming Merge, (IV) Translation Merge.

Based on Participation

In total there were 269 users that contributed to the development of the 13 CyanogenMod modules. The data shows that there were a total of 4 that were classified as core, 12 as middle and 253 as peripheral. Figure 8.5 shows the user name and the number of contributions each user in the core made.

From the core contributors, only Marco is part of the official team, with Jorge Ruesga working independently from the project, and the other two being employed as Google Android software engineers. Jorge Ruesga (user name *jruesga*) is the original developer for the CyanogenMod file manager app, but he is not part of the official team. The file manager was written independently as an app for Android devices in general, but was then forked and incorporated into the CyanogenMod project by David van Tonder (user name *dvtonder*). Marco Brohet is listed in the official documentation as being the translation leader, who coordinates translations and merges them to the repository, as well as carrying out the German translations. From the contributions to the 13 modules, he made a total of 388 contributions, with 298 (76.8%) of those being translation merges, which is consistent with his role in the project. The other two core contributors are part of the Android project itself, with Mike Jurka working as a Google engineer on Android and Winson Chung, also part of the Google team, specialising on the system framework.

There were more official project members in the middle group, which saw a total of 4 official members contribute a total of 766 times. The most active of these were the two general developers David van Tonder (user name *dvtonder*)

and Nebojsa Cvetkovic (user name *nebkat*). With 120 (51.7%) of David van Tonder's contributions being merges, while 170 (79.8%) of Nebojsa Cvetkovic's contributions being programming. In sharp contrast, project leader Steve Kondik (user name *cyanogen*) contributed a total of 101 time to the repository, with 82 (81.2%) of those being merges.

To summarise, the results show that the users who were classified as being either in the core or the middle were not all part of the official list of contributors to the project. Other users and contributors, such as Michael Jurka, contributed to the development of the AOSP but not directly to the CyanogenMod branch. Because the software licence allows others to fork the project, the contribution of the previous branches are included in the repository log. While the core contributors were important for the development of CyanogenMod ROM, not all of them made direct contributions to the project.

8.3.2 Wiki

Key contributors to the Wiki were determined by looking at the official project documentation and by analysing participation patterns. The official project documentation mention five individuals who have been given official editorial roles in the wiki, all of these have the title of Wiki editor, apart from user *Utkanos* who is labeled as Lead Maintainer. The results show that four of the five official wiki editors are also among the highest contributors to the wiki.

In total there were 700 users who made a total of 8,324 edits to the wiki. From those, 2 were considered to be part of the core and 12 part of the middle, with the rest being in the periphery. One of the official editors, Shawn Alty, although mentioned in the wiki as one of the key contributors, did not make any edits to the wiki. Table 8.6 shows the users who are part of the core and the middle, along with the number of contributions and their roles as stated in the project documentation.

According to the official project documentation, there are five official wiki members who have been given the role of wiki editors. These five individual are responsible for the maintenance of the wiki's IT infrastructure as well as the information that is contained within them. Although there were no editing

User	Contributions	Layer	Role
*Fattire**	1,441	Core	Maintains all sections of the wiki.
*Utkanos**	1,270	Core	Lead maintainer of the wiki.
Delphin PETER	515	Middle	
*White**	393	Middle	Helps maintain the wiki.
Therbom	361	Middle	
Jorge Ruesga	322	Middle	
*Flo Edelmann**	316	Middle	Maintains the wiki and programmed key features into the wiki.
Cnmahj	283	Middle	
Kaik541	132	Middle	
Adriano Tanaka	119	Middle	
Marcos MAZE	99	Middle	
FuzzyBot	88	Middle	
Fixtex	87	Middle	
*Shawn Alty**	0	—	Wiki Maintainer

Table 8.6: Key contributors to the wiki. * Official wiki editors.

contributions observed, Shawn Alty has been key to the maintenance of the wiki by conducting a one-off restructuring and redesign of the wiki.

To summarise, unlike the repository results, the participation patterns closely match the official roles. The main reason for this would be that the wiki was started again at a later stage when it was redesigned in November 2012. This would therefore explain the major effort by both *Fattire* and *Utkanos* who passed the information from the old wiki to the new one.

8.3.3 Forum

The idea of establishing the forum came from the project owner, Steve Kondik, after which, the job of administering it were delegated to various individuals. Abhisek Devkota (user name *ciwirl*), the official forum administrator explains

> When Steve decided it was time to establish CM's forums, he tasked psychoi3oy [Evan Widger] to setup the moderation team. Given my status in the IRC and my presence on the XDA forums, psychoi3oy approached me to head up the CM forums moderation.

Once being given administrative powers in the forum, Abhisek Devkota then delegated the day-to-day running of the forum, focusing on individuals who have hight levels of participation. With the recruitment of moderators being a continuous process

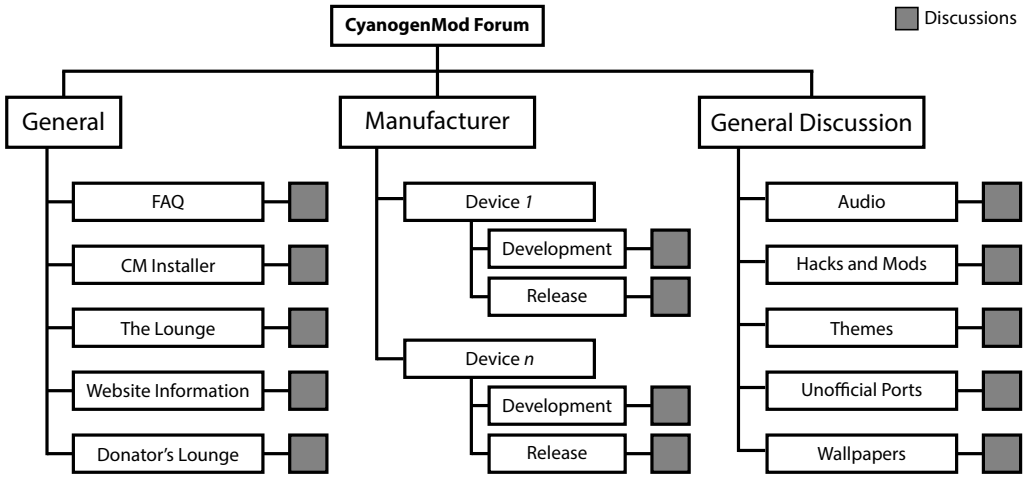


Figure 8.11: Forum Hierarchical Structure. showing sections and where the discussion threads are in relation.

> My frequency in contributing [to the forum] has dropped dramatically. These days, I visit to keep current of feedback and recruiting of new maintainers.

The forum is used by the project to allow users to discuss issues and to offer user-to-user assistance. Both in the wiki and in the web page, the forum is advertised as an area where users can talk to each other and ask for assistance. In addition, users can also exchange ideas between themselves and with the project team in an informal manner. The structure of the forum, which focuses on devices, makes it easier for users to locate relevant discussion threads.

The official forum is divided into 15 different sections, a "General" section, a "General Discussion" section, and 13 manufacturer-specific sections. Figure 8.11 shows the general structure of the CyanogenMod forum, with each manufacturer having the same structure. The General section contains frequently asked questions that give information on installing the ROM, however, it typically contains discussions on how to improve battery life and how to create backups. The General Discussion section focuses on the CyanogenMod specific modifications, and includes a discussion section on hacks and modifications. Finally, the manufacturers are listed on the main page alphabetically, with direct links to the different device threads, known as sub-forums, which themselves contain discussion areas.

The General section is divided into five sub-forums, which deal primarily

with general information about the project and special members' sections. The discussions in the FAQ sub-forum contains general information on usage and how-to guides, covering installation and optimisations. The discussion in this section are locked, meaning that they are just for reference rather than for discussion. The CM Installer section provides information about the installer applications, it contains its own FAQ and provides how-to on using the software, while at the same time being open for general discussion and feedback. The Lounge sub-forum represents a general area where forum members can discuss any topic, including those that are not related to CyanogenMod. The Website Information provides members with an area where they can discuss issues with the official web infrastructure of the project, providing ideas to the core team. Finally, the Donators' Lounge is a private area where members who have donated to the project are free to discuss general issues privately.

The General Discussion section is also divided into five sub-forums, where members are able to discuss customisations issues such as themes and audio. The Audio sub-forum contains discussions on a wide range of issues regarding either the audio playback of the device, such as low volume or lack of playback, or the Apollo music app, such as types of files supported or customisation. The Hacks and Mods sub-forum contains discussions on requests for future features that members would like to be included in the ROM, which main cover the inclusion of specific apps to the ROM. The Themes sub-forum contains discussions on how to implement custom themes as well as user-to-user assistance with any problem encountered, as well as exchange of theme ideas. The Unofficial Ports sub-forum contains discussions and information about porting the ROM to new devices, which is also an area for members to discuss and solve porting problems. Finally, the Wallpapers sub-forum is an area where members are able to exchange wallpaper background images with each other.

The forum also contains a total of thirteen manufacturers' sections[4] with a total of 170 devices. Each of the devices will contain two main sub-forums titled "Development" and "Release". The Development sub-forum contains discussions of the nightly releases, which includes feedback on new features and informal bug

[4]Google is included as a manufacturer because of the Nexus range, although the devices themselves are manufactured by other companies such as HTC, Samsung, and LG.

reports. The Release sub-forum contains discussion on stable releases, focusing primarily on user-to-user assistance and trouble shooting. It has to be noted that, while all device sections will contain both sub-forums, these are created automatically and some may not have any discussions in them. For instance, the Development sub-forum for the Nexus 5 contains a total of 115 discussion threads, while the Development sub-forum for the Sony Xperia Z2 Tablet contains none.

Based on Formal Role

There are two groups with official authority in the forum, the Administrators and the Moderators. The Administrators (admins) are individuals that are able to create or delete forum sections, as well as to grant other forum members with other types of roles, either as another Administrator or as a Moderator. The admins are also able to perform technological duties in the form of updating the forum software and maintaining the database. In the CyanogenMod project, there are a total of 5 administrators, all of whom are part of the core team, including project owner Steve Kondik.

The most active of the Admins is *bassmadrigal*, whose main role is to organise the forum discussions. The structure of the forum was determined by the title of the section, therefore limiting discussions to relevant topics. It is therefore *bassmadrigal* who ensures that all the device-specific areas contain both a "Release" and "Development" sections. In addition, *bassmadrigal* has also added a number of sticky[5] posts which deal with a range of common issues, such as installs and device support requests. Figure 8.12 shows this structure for both the Nexus 5 and the Samsung Galaxy S4 devices.

The second group consists of Moderators (mods), who take charge of the day-to-day running of the forum, making sure that members follow the rules and that posts are related to the relevant section. In order to achieve this, mods are given the privileges of being able to delete posts or conversations, and also to ban or block users. For the CyanogenMod forum, there are a total of 9 moderators, of which only two of them, Drew Suarez (user name *utkanos*) and Evan Widger (user name *psychoi3oy*) are official members of the project.

[5]A "sticky" is a forum post which accepts not replies and which remains on the top of the

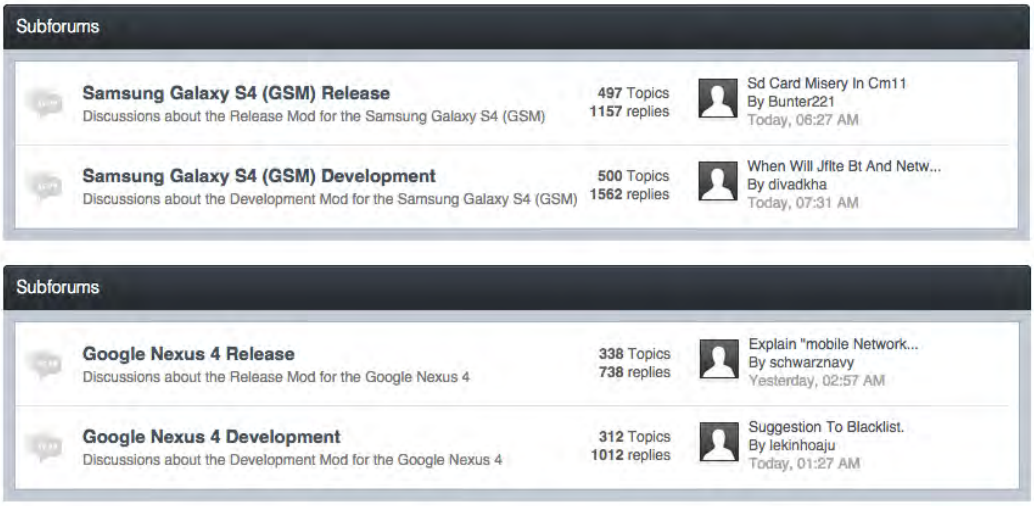


Figure 8.12: Forum Structure as Determined by Discussion Topics.

The most active moderator, *bassmadrigal* is very active in terms of making specific contributions regarding the maintenance of the device threads. His post history shows that much of his activity focuses on linking or redirecting users to other discussion or to wiki pages that contain answers or provide relevant information. This, in part, has been done in order to make sure that related questions are not asked over and over again, therefore reducing repeated questions and duplicate discussions. At the same time, *bassmadrigal* has also added a number of threads that deal specifically with the process of installing and updating CyanogenMod, along with an explanation of what each of the stages of development mean. Once again, the tutorial threads are focused on minimising the number of repeat questions and discussions.

Based on Activity

From a total of 67,023 users, there are a total of 1,037 (1.5%) classified as being part of the core, 9,048 (13.5%) as being part of the middle, and 56,938 (85.0%) part of the periphery. Within these, there were a total of 41 members that were either part of the project team, or who held an official role within the forum. The data shows that 15 of the 1,037 users in the core had either an official role in the forum or on the project. Equally, 11 of the 9,048 in the middle and 15 of the 56,938 in the periphery also had a formal role in the forum or project.

page at all times. This is generally used to write important information to users.

Role	User Name	Posts	Main Activity
Administrators	bassmadrigal	8,939	Enforces rules of participation and organises discussions.
	ciwrl	1,605	Community relations.
	jeagoss	110	Discusses changes on changes to the operating system. Last active in March 2012.
	cyanogen	29	Mainly updates on changes to the HTC Evo4 version. Last active in February 2011.
	chrissoyars	7	Answers technical questions regarding tools for building software.
Moderators	absolutezero	1,542	User assistance. Last active in June 2011.
	skrki	623	User assistance. Last active in June 2012.
	kaik541	598	User assistance. Last active in December 2011.
	pdr447	568	User assistance. Last active in January 2011.
	psychoi3oy	469	User assistance. Last active in April 2011.
	phaseburn	105	User assistance.
	cyanogenmod	55	Announcements and updates. Last active March 2012.
	servili007	40	User assistance and enforces rules. Last active May 2012.
	utkanos	10	User assistance. Last active October 2010.

Table 8.7: Contributions by Administrators and Moderators

8.4 Chapter Summary

The results show that there are different types of participation in each of the areas of contributions. In the repository we find that users can contribute to the project through code modifications to the core ROM, translations, and through porting. In the wiki we see that users contribute through creating new information pages, editing existing pages, and through the modification of the wiki base code. For the forum, the main type of contributions were based on user to user assistance, divided into assistance for both unstable and stable releases.

The results also show that the relationship between official roles and participation levels differs in each location. In the repository, the official project members were not among the highest contributing individuals. In addition, some of the contributors observed were not directly linked to the CyanogenMod project, but were instead members of the official AOSP team. The list of official project members and high contributors was similar for the wiki, where the official editors were observed to be part of the core. In addition, it was also observed that user Shawn Alty holds an official position within the project but his contributions are limited to improving the ICT rather than on editing. Finally, the forum results show that the users with official roles were also

among the highest contributing members, although they were present in similar numbers in the middle and peripheral layers too.

In summary, all contribution areas see a number of different forms of contributions. The key contributors in each of these areas differ in terms of the formal roles they have and their level of contribution. The next chapter will look at how these contribution areas are managed in terms of rules and procedures.

Chapter 9

Governance

9.1 Introduction

The main objective of this study is to determine how an open source project manages a wide range of contributions from its users. Chapter 7 looked at the different products and services that are offered by the project, determining that contributions can be observed through three contribution areas: the repository, the wiki, and the forum. Chapter 8 then determines how the users contribute to the project and who they key contributors are, finding that multiple types of contributions are made in each of the contribution areas, and that those with official roles may not be among the highest contributors.

This chapter looks at governance at the community level, focusing on the roles and the patterns of contributions. In Section 9.2 the results show that the rules for contributing to the project differ according to the area where these are made, as well as the type of contribution being made to the project. In addition, Section 9.3 finds that contributions to the repository are sporadic and occur in bursts, with core contributors being more active at the start of development and the peripheral contributors being more active in the later stages. The difference in the timing of contributions may therefore have an effect on the governance and coordination of contributions.

9.2 Participation Rules and Procedures

This section deals with the rules and procedures for contributing to the project. Contribution rules are defined as a set of norms, either explicit or implicit, which determine how and what users can contribute. Contribution procedures are defined as the tools and the steps that contributors must follow in order to participate in the project. In essence, the key difference is that rules determine *what* contributions can be made whereas procedures determine *how* contributors can be made.

The results show that both the rules and procedures, at the community level, are different for each type of participation. While some rules were applied to all contributors, specifically regarding *when* a contribution could be made, other rules were specific to activity. In addition, the procedures that were in place, such as the use of templates or review systems, were enforced differently for each form of contribution. This was particularly seen in programming and translation contributions, where, although both used the Gerrit review system, programming contributions were scrutinised more than translations.

9.2.1 Repository

The rules and procedures for contributing to the project through the repository are different depending on whether the contribution is for the core ROM, a port, or a translation. In terms of contributing to the core system, procedures are emphasised over rules, where the project documentation offers in-depth step-by-step instructions, including a list of tools, that determine how contributions are to be submitted and accepted to the project, but does not offer any rules regarding the scope of the contribution. On the other hand, when contributing to the project through porting, rules are emphasised over the process, where a five point list provides clear instructions as to what the specifications of the software must be in order to be classified as an official port. Finally, in terms of translations, there is a heavier reliance on templates, which users are asked to copy, therefore restricting contributions to predetermined values. In addition, all three forms of contributions go through peer review through the Gerrit system,

where they can be either accepted or rejected, although the basis for accepting the contributions will vary depending on its type. The objective of the rules and procedures is to ensure consistency and quality across contributions, but at the same time making sure that contributions are consistent with the project goals as set by the owners and the project leaders.

Contributions to the Core ROM

The official documentation does not offer any clear set of rules regarding who can contribute, or what type of programming contributions can be made. The only rule that applies when submitting to the core ROM is that of timing of the contributions, in that submissions during the nightly stage of the development are not accepted. Besides that, users are able to contribute at any time as long as they follow the procedures that are described on the project documentation.

The procedures outlined in the project documentation focused on the use of tools and procedures for modifying and then submitting the code. The documentation emphasises the use of the official Android SDK (Software Development Kit), which can be used as a working environment and to test modifications. In addition, contributors must also use Git software in order to directly connect to the repository and use the system's functionalities, both to alert owners of new changes as well as to obtain changes made by others. Finally, once the modifications have been made, users are asked to submit their changes to the Gerrit review system, where all modifications can be reviewed by the project leaders and, if successful, can then be merged directly to the repository.

Contributions to the core ROM are typically only reviewed when they are submitted by general contributors. In other words, contributions by core team members and other contributors with formal project roles are not reviewed in great detail. As the head of Developer Relations, Jef Oliver (user name *jeagos*) states

> ... [contributions] brought in by other core members. I really don't need to review their work. They know what they are doing. ... [for non-core members] I've rejected several code submissions due to bad coding practices or security concerns.

Besides issues of quality and security, the peer review of submissions from the general contributors operates as a system which limits the type of contribution that is accepted into the core ROM. Despite there not being any specific rules as to what can be submitted, the documentation and the evaluation of code point to a common understanding on the scope of the contributions brought by general users. This assumption is carried in the documentation as well as in the timing of the contributions. For instance, the documentation on contributing to the code begins with the following line

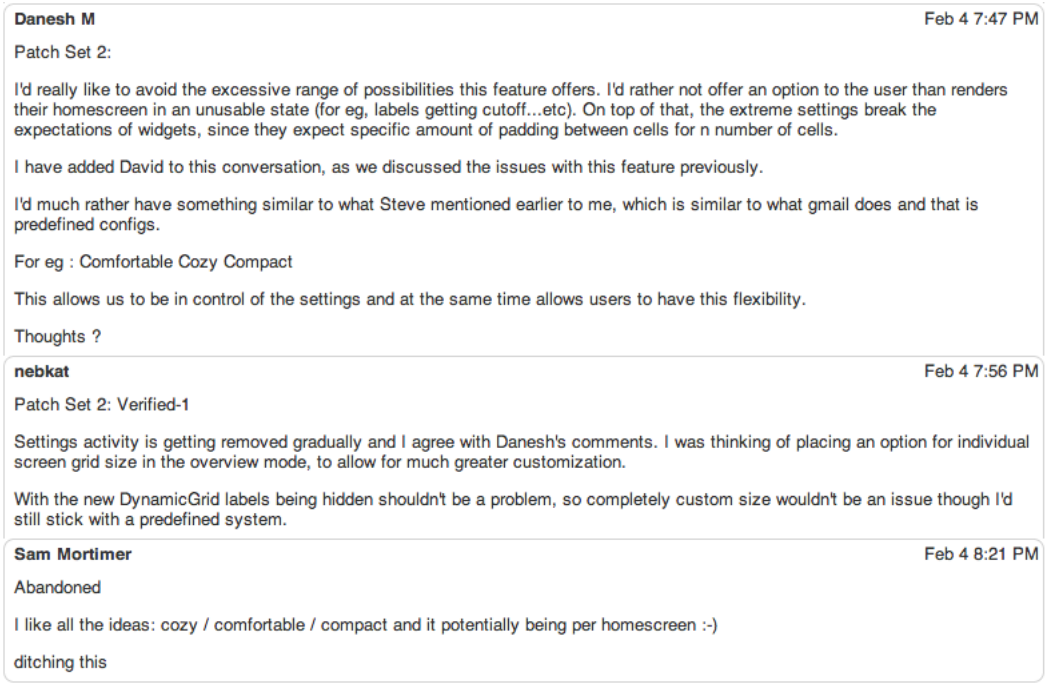
> You can use Gerrit if you find an error in the source code or if you believe you have a better way of implementing a certain feature.

This therefore shows that the assumption is that general users may contribute to the project only through fixing and improving existing code, rather than the introduction of new features. In a similar way, by restricting input from users until the Release Candidate stage, means that users have very little say during the early stages of the design process, where new features are decided on.

Even if users submit a new feature, these are still subject to review through the Gerrit system, where it is then evaluated in terms of the project's objective as envisioned by the project leaders. Jef Oliver is one of the core team members in charge of evaluating submissions, having also the say on whether these are accepted or not, and maintains private and constant contact with project owner Steve Kondik. On the issue of determining whether a new feature is accepted or not, Jef states

> I've helped steer the project for almost 2 years now. I've made judgement calls on feature inclusion and exclusion, device inclusion, and what outside projects we work with. I'm comfortable enough to address Steve (cyanogen) directly on issues and push for things with him.

Submissions made by general contributors are therefore evaluated depending on the common understanding of the direction of the project and what the project leaders wish to achieve. In addition, the views of the project owner, Steve Kondik, are often taken more serious than others. A clear example of this is presented in Figure 9.1, which shows the review of a new feature submitted by a general contributor that has been rejected by the project leaders on the basis that it does



Danesh M Feb 4 7:47 PM

Patch Set 2:

I'd really like to avoid the excessive range of possibilities this feature offers. I'd rather not offer an option to the user than renders their homescreen in an unusable state (for eg, labels getting cutoff...etc). On top of that, the extreme settings break the expectations of widgets, since they expect specific amount of padding between cells for n number of cells.

I have added David to this conversation, as we discussed the issues with this feature previously.

I'd much rather have something similar to what Steve mentioned earlier to me, which is similar to what gmail does and that is predefined configs.

For eg : Comfortable Cozy Compact

This allows us to be in control of the settings and at the same time allows users to have this flexibility.

Thoughts ?

nebkat Feb 4 7:56 PM

Patch Set 2: Verified-1

Settings activity is getting removed gradually and I agree with Danesh's comments. I was thinking of placing an option for individual screen grid size in the overview mode, to allow for much greater customization.

With the new DynamicGrid labels being hidden shouldn't be a problem, so completely custom size wouldn't be an issue though I'd still stick with a predefined system.

Sam Mortimer Feb 4 8:21 PM

Abandoned

I like all the ideas: cozy / comfortable / compact and it potentially being per homescreen :-)

ditching this

Figure 9.1: Contribution not accepted

not match the vision for what the user interface should be. Here, the modification was of the way that the icon menus would be displayed in the modified music app, which allowed users to customise the size of the icons in a menu. This modification was ultimately rejected by team members Danesh Mondegarian and Nebojsa Cvetkovic (user name *nebkat*) because it did not match the project design goals. Most importantly, Danesh Mondegarian refers to a conversation he had with Steve Kondik regarding the design of the menu as a deciding factor for the rejection of the contribution.

The evaluation of what features are included and which ones are not, has also changed throughout the life of the project. For instance, CyanogenMod has become much more selective in what features are included in the ROM by default, with lead of User Experience Björn Lundén (user name *blunden*) explaining

> We want CM to adhere to a certain standard. We try to make our additions feel like they could be stock features implemented by Google. We moved away from the old "add ALL THE THINGS!!" approach we used in CM 7 and earlier when we started reimplementing features for CM 9 (ICS).

To summarise, contributions to the core of the CyanogenMod ROM are open to all users, with extensive documentation provided which describes the

process contributors need to follow in order to do so. Although there are no explicit rules as to what users can contribute, the Gerrit revision system and the timing of the involvement of the general users reduces contributions to minor improvements to existing features implemented by the core team. This therefore means that the project leaders, along with formal project contributors, are predominantly in charge of adding new features to the core system, while the role of the general users is seen as improving or fixing them. Finally, the procedures and the tools that are specified in the documentation also help primarily with the standardisation of contributions in terms of quality, where the tools themselves form a first stage of testing before they are submitted to the project.

Contributions through Porting

The project's open nature allows for both official and non-official porting of the ROM to new devices. Since the ROM is device-specific, the project relies on general users to port Cyanogen to other devices, as the core team members may not have access to them. To port the ROM to new devices, users do not require specific consent from the project owners, so long as they follow the licencing rules, particularly that of attribution. For the port to be included in the list of supported devices, and therefore in the official repository, it must follow a specific set of rules regarding its functionality and its composition.

The procedures that are involved in the porting of the ROM to new devices is not as well documented, as it may be assumed that it is carried out by advanced users who may already be familiar with the required tools and procedures[1]. While the tools and procedures for contributing to the core ROM are explicit and well-defined, there are no tools or mechanisms that have been specified for porting contributions. Instead, the official wiki offers a guide which states which files and which modules need to be modified in order to port to a new device, but it does not specific how to modify them or what tools should be used. The porting itself is carried out by modifying five files found throughout the base of the ROM, with the aim of giving the ROM direct access to the hardware for that device, such

[1]In the project's official Wiki, the section on porting is placed under the Advanced Topics in the Learning section.

as camera, touch screen, processor, and buttons. For the port to be considered part of the project, it must meet the following requirements

Prerequisites

You can save everyone a lot of time by making sure that you have met these requirements:

Full hardware support – Everything– every peripheral, every wireless feature– that is supported by stock *must* be supported by CyanogenMod.

Passed the CTS – The Android Compatibility Test Suite will ensure that your port is functional. If it doesn't pass, get back to work until it does.

Stability – Nothing should crash. Ever. CyanogenMod has a reputation for stability and your port must be rock-solid. This means no "sleep of deaths" either.

No overclock by default – While overclocking options in the Performance menu is well and good, leave the default speeds at stock.

Do not over-volt either – Do not go outside of the spec for power consumption. If you want to provide sysfs methods for upping the voltage, that's up to you. But it must not be a default setting.

You have designated at least one device maintainer and at least one person to be responsible for the device's wiki information. The former will evaluate submissions to the device's repositories. The latter will ensure that the information/documentation about the device on the wiki is always up-to-date and correct. These people must be identified when applying for official recognition.

To summarise, porting the ROM to new devices is seen as an advanced form of contribution, and as such there is a greater emphasis on rules for submitting rather than on the procedures that must be followed. The rules for submitting a new port are therefore much more explicit and are focused on both quality and standardisation of experience from the perspective of the user. At the same time, the processes are less defined, leaving it up to each contributor to determine how to carry out the porting activity.

Translations

Contributing to the project through translation is more restricted in terms of the scope of what can be done when compared to programming. Each module that contains a graphic user interface which users will interact with contains a separate file with the text that will be displayed. This text will be written in an

English

```
<string name="page_recent">Recent</string>
<string name="page_artists">Artists</string>
<string name="page_albums">Albums</string>
<string name="page_songs">Songs</string>
<string name="page_playlists">Playlists</string>
<string name="page_genres">Genres</string>
```

Japanese

```
<string name="page_recent">履歴</string>
<string name="page_artists">アーティスト</string>
<string name="page_albums">アルバム</string>
<string name="page_songs">曲</string>
<string name="page_playlists">プレイリスト</string>
<string name="page_genres">ジャンル</string>
```

Figure 9.2: Sample of the Translation XML

XML format, which will allow users to modify it in order to display a number of languages. Figure 9.2 shows part of the XML file, with the default English (top) and the translated Japanese version (bottom). As can be seen from the figure, to translate the file, the user needs only to change the text within the existing XML tags, therefore limiting the amount of work that needs to be done.

There are specific rules that need to be followed when submitting translations to the project, all which are focused on the integration of the file to the module and the format of the translated text. One of the key rules of translation focuses on the naming of the file being translated, asking contributors to name the files in accordance to the international language code. For instance, the default file for the text is typically named '`strings.xml`', allowing the software to call that file and display the text to the user. If the translation is in Japanese, the translated file should be named '`strings_ja.xml`', while if its for Spanish, the file should be named '`strings_es.xml`'. The reason for this is that the software will then be able to identify the existence of the translation and will therefore be able to use it.

The other rule is regarding the formating of the text being translated, how it should be written and what should be translated. This second rule is somewhat focused on how the text is displayed, where each word should be separated by one

space and each sentence should start with an upper case letter. At the same time, there are certain texts that should not be translated, such as those that should be submitted to the AOSP directly, or that should remain in English so that developers can understand them. Therefore, each submission is judged on both these aspects, that the file is labelled correctly, and that everything is translated except for those that that state otherwise. The focus of these two rules are on the standardisation and integration of the translations, but these do not deal with the issue of quality in terms of accuracy of translation.

Quality and accuracy of translations are assessed through the review process but may, in some cases, be based on trust. Unlike programming contributions, the different languages that are submitted create a problem of peer-review where no other user or contributor may have the knowledge and skill to assess the translation. Translation leader Marco Brohet (user name *therbom*) explains the process assessment

> Due to the complexity of submitting translations to Gerrit, our assumptions were that only people who had sufficient knowledge of the language would want to learn how to fix/add translations. Therefore, there was not a real quality check of the submitted translations. Sometimes only we received feedback from our bug report channels. That's why it was important to have multiple translators working for a language, so that they can give feedback to each other. When I see a new translation patch, I immediately add the translators of that language as a reviewer. If they all give their approval, the translations can be merged. If someone adds or reviews translations regularly, I would add him to our Google+ community and also add him to other translations for that particular language. And when someone is inactive for a while, he is eventually removed. As such, this review system is primarily based on the reputation and experience you have in this project. And if there is only one guy submitting translations, it gets merged directly if he is fine with it, because there is no one else to review. Luckily, this happens rarely.

To summarise, contributing through translations to the project is slightly restricted in terms of what the users are allowed to do. This is reflected by both the rules of contributing to the translation as well as the process of reviewing the translations. The modifications are restricted to a limited number of sentences in predetermined files, which must then be named according to international standard. The approval of the contributions is primarily based on trust, where the quality of the translation is left to the contributor themselves. The main reason

for the difference between programming and translation is that it is more difficult to find individuals that have the required knowledge to correct any mistakes.

9.2.2 Wiki

Contributions to the wiki differ depending on the information that is being displayed as well as the type of page. There is a difference between the device-specific pages and other pages that provide information about the project or about processes. The project leaders are involved in the initiation and maintenance of the project information and educational pages, while the general contributors, specially device maintainers, are mainly responsible for the device-specific pages.

For the project and the educational pages, the project leaders will typically discuss the need to add certain information to the wiki in a private Google+ group, where a consensus will then be reached as to the content and structure of the new page. The official wiki editors, primarily Drew Suarez (user name *utkanos*), *fattire*, and Shawn Alty, will then initiate the page and provide it with most of the information. After the page initiation, general users are then able to update the information.

The procedure for device-specific pages, on the other hand, relies much more on the use of templates, but also gives flexibility to contributors in order to add relevant data. One of the project wide rules states that users that contribute through porting must add the relevant information for that device on the wiki. To help in this process, the project leaders developed easy-to-use templates that can be used to create three types of device-specific pages, how to install the ROM, how to build the ROM, and where to find additional information. The templates are copied by the device maintainers, and small changes in the text are made in order to reflect the differences between devices. Each template contains the following message

```
<!-- NOTICE: THIS IS A GENERIC "BLANK" TEMPLATE TO BE EDITED FOR
YOUR DEVICE AND COPIED TO:

Template:Device_codename
```

```
Where ‘‘codename’’ is the correct codename for your device.  This
template contains all the necessary information to automatically
generate a device information page, installation instructions, a
build walkthrough, and a known issues page.

This template is annotated.  Fill out all required sections, but
feel free to leave optional fields blank.  You can use spaces,
commas, basic wiki formatting, and multiple lines.  Some HTML and
wiki functions can cause issues on the full device listing at
http://wiki.cyanogenmod.org/w/Devices so be sure to check that page
after finishing inclusion of your device.

You can delete this header notice and the tip below when you make
your own device_codename page so that you do not confuse any new
maintainers of that page in the future.  Good luck!  :)  -->
```

Through the use of templates, all device pages contain the same information and the same structure. For instance, the main Device page is a landing page which provides users with a wide range of general information, such as known issues, and links to the maintainers and forums. The Installation page then has sections that take you through the several steps required, including unlocking the device, installing recovery, then installing CyanogenMod. Finally, the build page takes visitors through the process of building the CyanogenMod ROM from source code specifically for that device.

The use of templates means that contributors are restricted in terms of what contents is included in the page, and how the contents is structured. The templates have predetermined sections, where the contributors are then shown where to fill in the gaps. This is made possible because the process of either installing or building the ROM share many similarities across devices. This therefore means that there will be a considerable amount of information that applies to all devices, such as unlocking, sideloading, and restarting sequences. Figure 9.3 shows both the similarities and slight differences between the Nexus 5 and the HTC One devices during the first few stages of installation.

The templates are therefore a useful for saving time for the maintainers and for consistency throughout the wiki. Community Staff member Evan Widger (user name *psychoi3oy*) explains the usefulness of the wiki templates

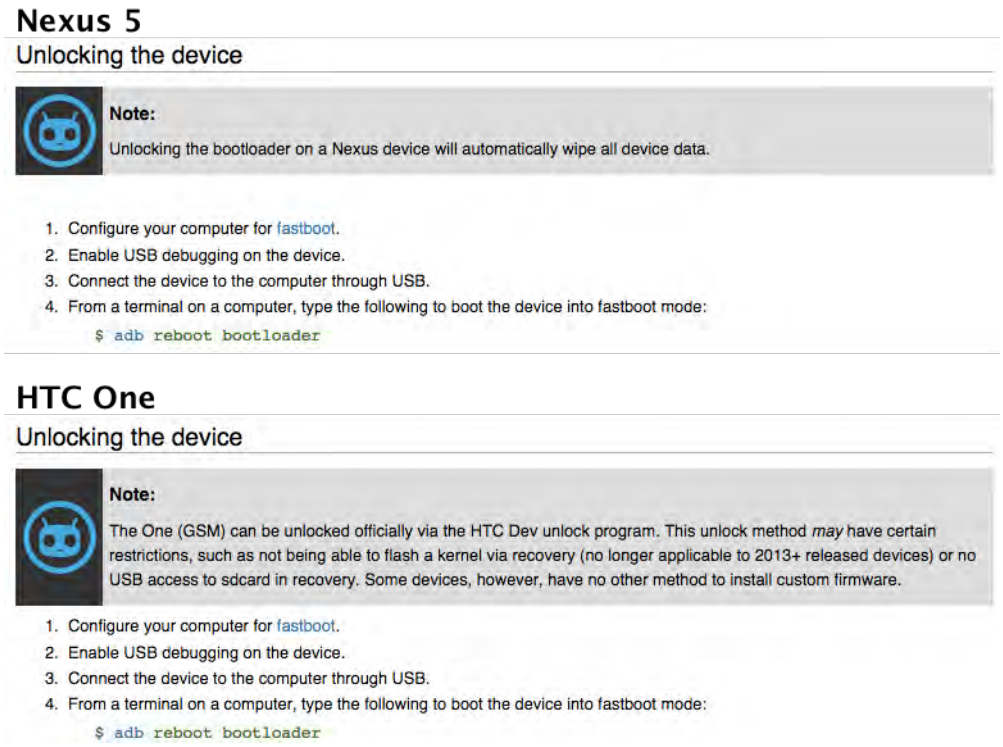


Figure 9.3: Installation Pages - Sample. Shows the first few steps of the installation process.

> Templates are useful but as far as I'm concerned they're just that, an indication of what all the pages generally look like for consistency. Of course devices have different subtleties that require extra steps. If you have to copy the text of a template instead of including it so you can fit something between step 3 and 4 then go ahead. Good instructions and functional devices [is better than] consistency. As long as all the pages are roughly similar I'm good with them.

These templates have to remain flexible, as the inflexibility can also have an effect on whether or not the pages get written. Official lead maintainer Andrew Dodd (user name *Entropy512*) explains the negative effect of the templates

> The problem is, all devices are different, so forcing a "template" down maintainer's throats is bad. I remember trying to work with *Kaik451* on getting the I777 installation page updated... Pretty much anything that would have been useful couldn't be put in the page BECAUSE TEMPLATE!!!! End result was I just said "screw this, I'm not bothering with it" and probably a number of other maintainers did too. So -1 on templates unless maintainers can actually override them.

In order to provide flexibility, and to allow contributors to add device-specific information that may not be included in the template, the project leaders developed 13 key rules that must be followed. These rules

determine a number of key factors that focus on the way that the text is displayed and written, therefore focusing more on controlling the display of the information rather than on the contents of it. The rules are as follows

1. Make it easy to read/digest for anyone. Not just technical people.
2. Don't dumb it down though. The audience is your typical non-professional, non-IT end user with an interest in learning what CM is, and why they would use it. And yeah, there's some smaller percentage of them with a bit more experience who might want to actually start hacking their devices themselves.
3. The goal is to make it EASY for visitors to learn. It's not meant to be in any way elitist or to create artificial hurdles to learning. The wiki is here to help people.
4. Tone should be irreverent and informal. This is fun stuff. But keep it accurate and as compressed as possible without being overly dry and boring.
5. Include references and links to other sites whenever possible. Never be afraid to send someone to another site if it's relevant to the topic– this includes other ROMs or wikis.
6. When writing step-by-step instructions, start with a bolded header that explains the goal, ending with a colon. (**To do some operation, follow these instructions:**) Then number each step individually using the wiki list tag. Keep each step super-clean and simple.
7. Use active, present-tense verbs in instructions. ("Double-click on the icon...")
8. All on-screen menu items should be bolded and listed from most general to most specific. So present the information in the order that the user will encounter it. Also, use text (Under the File menu, choose Open File...) rather than shortcuts like `File->Open File`. It's better to standardize this in as clear a manner as possible. The extra work of typing out the steps will pay off for the reader.
9. It's also a good idea to tell the user what to expect from an action. (If all goes well, your build should now begin.)
10. All terminal activity should be presented in Courier using the `<code>` and `</code>` tags. Always indicate what text comes from the computer vs. what is entered by the user. Precede console inputs with a "`$`" to indicate user input.
11. Similarly, any reference to file names or directories within text should also be in Courier. (Look in the /data directory for a file called README.TXT.)
12. When accurate grammar is needed, check with Strunk and White. (But that doesn't mean you can't use slang and shitty grammar– as long as you're clear. Just sayin'.)
13. Coordinate your pages w/a designer. We should have an uniform, appealing layout and color scheme. Plus tons of screenshots and such are always nice to look at.

To summarise, the project leaders and other official project members decide what information should be added to the wiki in a private group, after which the lead wiki team will then implement it. The pages that are written by general contributors typically make use of templates, therefore restricting users in terms

of the contents, the layout, and the way the information is displayed. At the same time, the project leaders acknowledge the need for flexibility, and have therefore instituted 13 rules that should be followed when general contributors step away from the templates. The focus of both the templates and the rules, however, is limited to the broad subjects (installation, building, and additional information), layout, and the way the information is displayed, rather than on the accuracy of the information provided.

9.2.3 Forum

The forum operates on a mixture of implicit and explicitly rules that governs how users interact, as well as through norms that are guided by internet etiquette. Internet etiquette can be described as a collection of norms that determine how individuals should act in a specific environment, in this case the forum. Members of the forum are expected to understand the principles of using forums in general, and to follow the norms as they would on any other forum. Some of these norms include the miss-use of capital letters, the incorrect size of an image posted, or not attributing information to relevant users. These norms, therefore, are not specific to the project itself, but to the technology that is being used.

In terms of rules that govern participation, these fall under two categories, which are global rules and local rules. The global rules are ones that are relevant to all members regardless of the sub-forum or the section in which they post, meaning that all members have to follow them. The local rules, on the other hand, are those that are specific to the sub-forum or the section where the members are active, and may therefore not apply in other areas. These rules are either implicit, in that they are embedded into structure of the forum, or are explicitly communicated to the members. The forum Administrators are the ones that are in charge of setting the rules while the forum Moderators are largely the ones that enforce them.

The global rules revolve around the etiquette that must be followed by forum members, particularly when it comes to the creation of new discussions. The first rule states that members should make sure that they are aware of the contents of the forum before starting a new discussion, this is done in order to make sure that

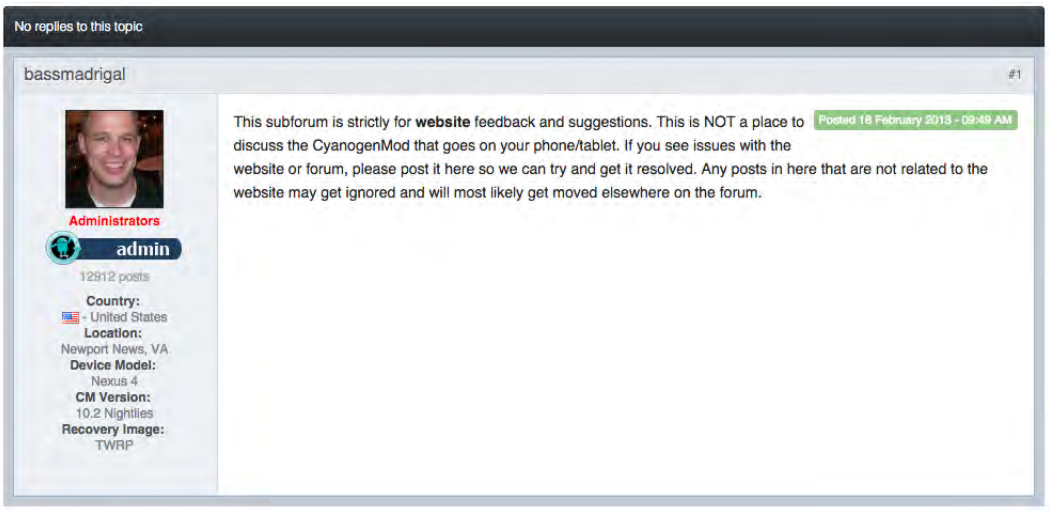


Figure 9.4: Rules for Website Information Sub-Forum.

discussion threads are not duplicated. In addition to this, each device-specific subsection will also contain links to the wiki and other areas where members can be information about the device, so that they do not duplicate the information. The second global rule states that conversations should remain on topic and should be consistent with the part of the sub-forum in which they have been posted. Where the discussions do not meet the rules, they are moved to the relevant location or deleted by the Administrators or Moderators.

The sub-forums will have their own rules, depending on the nature of the sub-forum and the topic that is being discussed or the information that is being requested. The rules for those particular discussions are shown at the top of the lists of discussions and are generally made sticky[2], most of the time they contain the title of 'Please Read'. Figure 9.4 shows the rules that apply to the Website Information sub-forum.

In addition, there is a rule regrading the discussion or distribution of proprietary files and software, which, although specific to the relevant sub-forum, may also be enforced forum-wide. For instance, the Audio sub-forum found in the General Discussion section contains a warning which states that audio files that are subject to copy right laws should not be exchanged. Similarly, the Lounge sub-forum on the General section contains a

[2]A sticky discussion thread means that the discussion will be displayed at the top of the list at all times.

rule stating that the discussion or distribution of proprietary software is not allowed, making direct reference to the Swype app[3] in a sticky post titled "Pirated software in the forums (including Swype)".

To summarise, there are a wide range of rules that are in place in the forum, most of which will be specific to the sub forum and to the topic being discussed. These rules will be communicated to the members explicitly through stickies or implicitly through the structure and organisation of the forum itself. The general rule pertaining to the forum is to make sure that the conversations that are started remain on topic and are relevant to where the conversations are placed. The rules are enforced by both the Administrators and the Moderators who can either remove or move discussions, or by membership management though temporary or permanent bans.

9.2.4 Section Summary

This section looked at the rules that guide contributions to the project. The results show that the rules and processes for contributing to the repository are different depending on the type of contribution that is being made. All of the contributions made to the repository are submitted to the Gerrit review system, where they are then reviewed by the core team in oder to determine quality and suitability to the project. A difference in procedure, however, was observed for translation contribution, which were sometimes not reviewed because of a lack of skill from project members.

In terms of the wiki, we see that there was a significant reliance on the use of templates, this means that those contributing would typically be constrained to pre-determined sections of information. The page templates have been used by contributors in order to provide device-specific information, and has the information sections required already built into them. The contributors need only make smaller edits to the page in order to make the information device-specific.

[3]Swype, a keyboard application for Android devices, sent an official notice to the CyanogenMod team in March 2012, stating that although they would like users to continue to use Swipe, they do not want them to use it without having purchased a licence. As a result, CyanogenMod banned all links to the software on the forum.

Finally, the forum rules were all focused more on the rules of interacting with others and on what can and cannot be posted in the discussion threads. The forum was also structured by the project leaders so that it would reflect the key stages of development, primarily Nightly and Stable releases, therefore limiting the relevant discussions to those sections.

9.3 Contribution Patterns

This section looks at the patterns of contributions in each of the contribution areas. The results show that contributions to the repository occur in bursts of activities at different stages, with the core developers making the majority of the contributions at the early stages and the periphery at the later stages of development. In addition, the results show that changes to the code are made predominantly by the core developers while the translation changes are carried out by the periphery contributors. Finally, the association networks show that core developers, general contributors, and translators all display different patterns of activity within the repository. For the wiki, we see that the core wiki contributors are predominantly responsible for initiating the information pages, most of which were created at the beginning of the observation period. The information in the wiki was moved from an older version on to the new one in November 2012, therefore explaining the high level of contributions and page initiations from the official editors. From the forum we see that participation by the official team members and the core contributors were mainly for the experimental Nightly ROM discussions.

9.3.1 Repository

There were a total of 269 unique users who made a total of 4,992 contributions for the development of version 10 of the CyanogenMod ROM. The data was collected from 13 modules in the official CyanogenMod GitHub account which were directly linked to modifications made to the ROM that are specific to the CyanogenMod project. The data covers a time-frame of 24 months, from January 2012 until December 2013, which covers the period of development for version 10

of the ROM, from the release of the AOSP until the final update, version 10.2.

CyanogenMod has a total of 1,068 repositories in their GitHub account. A total of 98 of those repositories were forked directly from the AOSP project, and are therefore part of the base system, but it does not include the Google Apps. A vast number of the repositories in the CyanogenMod account are device-specific copies of the 98 modules, where each manufacturer and each device will have its own copy and will be labelled as such. In addition, a number of the device-specific repositories are also for past versions of the software which may be still under maintenance.

Users can contribute through the repository by modifying the code of the master copy of the ROM, through porting the master ROM to other devices, or by translating the software to other languages. Users can modify the master code by submitting their own changes through the Gerrit review system, where they will be peer reviewed and either accepted or rejected. The master code contains approximately 98 modules that are also part of the Android operating system, and as such, all changes made to the official Google-rum Android repository are also merged to the CyanogenMod version. In addition, there are an added 13 modules which are specific to the CyanogenMod project, which covers all the added functionality to the ROM.

This sections looks at the participation patterns within the subgroups of the project though the project life cycle, measuring participation on the repository, looking specifically at each of the subgroups, when they participate, and what kind of participation the have. The data gathered was from the thirteen CyanogenMod-specific modules, from the duration between January 2012 until December 2013, therefore covering a total of 24 months.

The results show that those who were classified as being core tended to contribute more at the start of the development cycle, while the periphery tend to be more active towards the end. From the 269 contributors, 4 were classified as being core, 13 as being in the middle, and 248 as being in the periphery. The core contributors originally had more programming contribution at the start, with it then changing towards merging changes to the repository. In contrast, the peripheral contributors were more active in terms of both programming and

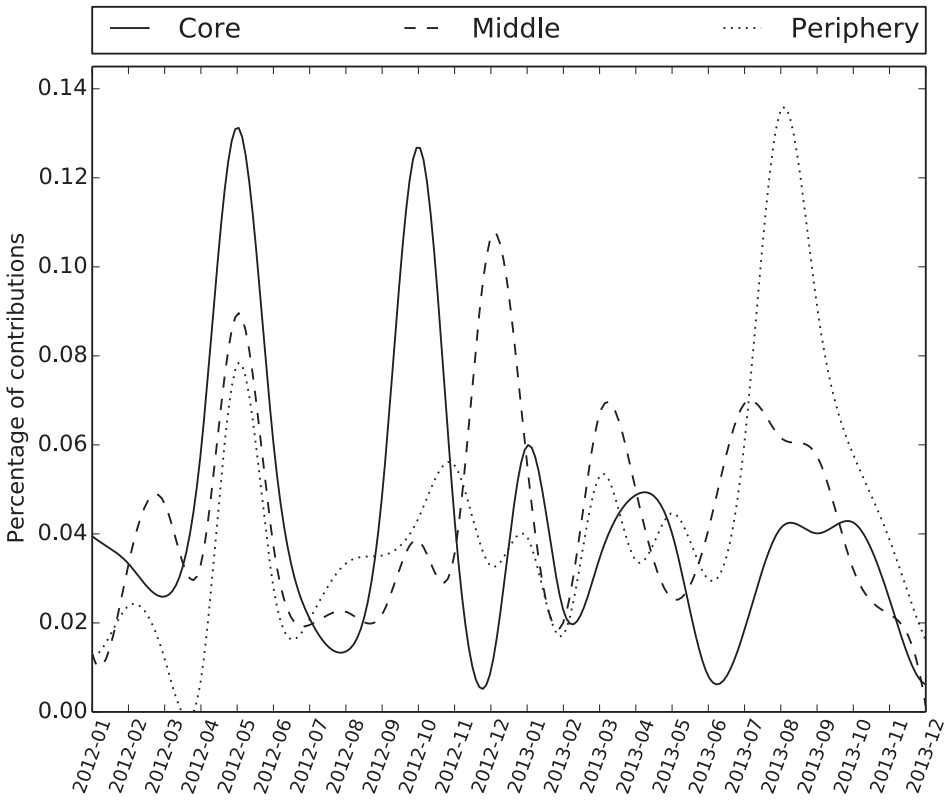


Figure 9.5: Contributions by Subgroups

translation contributions, with the majority of the activity taking place at the end of development. Finally, from all the subgroups, the majority of the contributions made by the core and the middle were programming, while for the periphery it was translations.

The majority of the contributions made by the core tended to happen at the start of the development, with the periphery contributing more towards the end of the cycle. From Figure 9.6 we can see that the core contributions saw two major spikes in contributions, the first one being before the release of the AOSP, and the second one being just before the release of the first stable version. In contrast, we see that the peripheral contributions saw a large spike in activity towards the end of the development, after the release of the second stable version. There is further evidence of this in Table 9.1, where we see that over 50% of all submission made by the core had taken place within the first 10 months. For the middle and the periphery, they reached the 50% mark later, in the 12th and 16th month respectively.

The different layers were also responsible for contributing through different

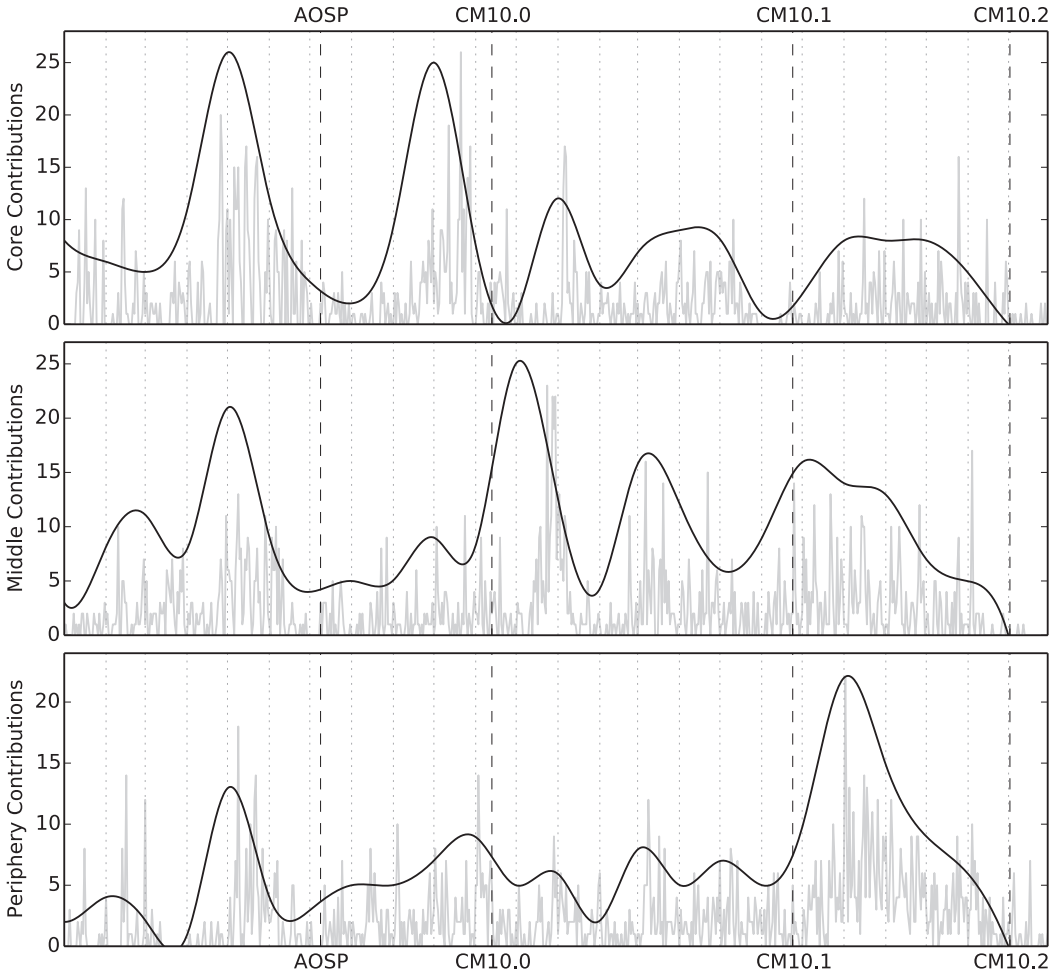


Figure 9.6: Contributions according to sub group.

	Core	Middle	Periphery
2012-01	3.9 (64)	1.4 (23)	1.5 (25)
2012-02	7.3 (119)	4.7 (79)	4.3 (73)
2012-03	9.9 (160)	9.3 (157)	4.8 (81)
2012-04	16.4 (266)	13.0 (219)	5.5 (93)
2012-05	29.2 (473)	21.6 (365)	13.5 (226)
2012-06	34.9 (566)	25.3 (427)	16.1 (271)
AOSP	**35.2 (570)**	**25.5 (430)**	**16.3 (273)**
2012-07	37.0 (600)	27.3 (460)	18.2 (306)
2012-08	38.4 (622)	29.7 (501)	21.6 (362)
2012-09	43.4 (703)	31.7 (535)	25.3 (424)
2012-10	56.0 (907)	35.7 (602)	29.7 (499)
CM10.0	**57.4 (930)**	**37.4 (632)**	**33.8 (567)**
2012-11	60.3 (977)	39.1 (660)	35.1 (589)
2012-12	61.3 (993)	50.4 (851)	38.3 (643)
2013-01	67.2 (1,090)	55.4 (935)	42.5 (714)
2013-02	69.6 (1,128)	57.3 (968)	44.0 (738)
2013-03	73.3 (1,188)	64.1 (1,082)	49.2 (826)
2013-04	78.0 (1,264)	69.0 (1,165)	53.0 (890)
2013-05	81.8 (1,326)	71.6 (1,208)	57.1 (959)
CM10.1	**81.8 (1,326)**	**71.7 (1,210)**	**57.4 (963)**
2013-06	82.6 (1,339)	75.8 (1,279)	60.0 (1,008)
2013-07	84.6 (1,371)	82.6 (1,395)	66.5 (1,117)
2013-08	88.6 (1,437)	88.8 (1,499)	80.2 (1,346)
2013-09	92.6 (1,501)	94.5 (1,596)	88.9 (1,493)
2013-10	97.1 (1,574)	97.7 (1,650)	94.5 (1,587)
2013-11	99.4 (1,611)	99.9 (1,686)	98.5 (1,654)
CM10.2	**99.4 (1,611)**	**99.9 (1,686)**	**98.5 (1,654)**
2013-12	100.0 (1,625)	100.0 (1,688)	100.0 (1,679)

Table 9.1: Percentage of Contributions

Activity	Core	Middle	Periphery
Merging Translations	398 (0.245)	285 (0.169)	56 (0.033)
Merging Programming	383 (0.217)	368 (0.218)	166 (0.099)
Translations	82 (0.050)	391 (0.232)	798 (0.475)
Programming	762 (0.469)	644 (0.382)	659 (0.392)

Table 9.2: Layers and Activities

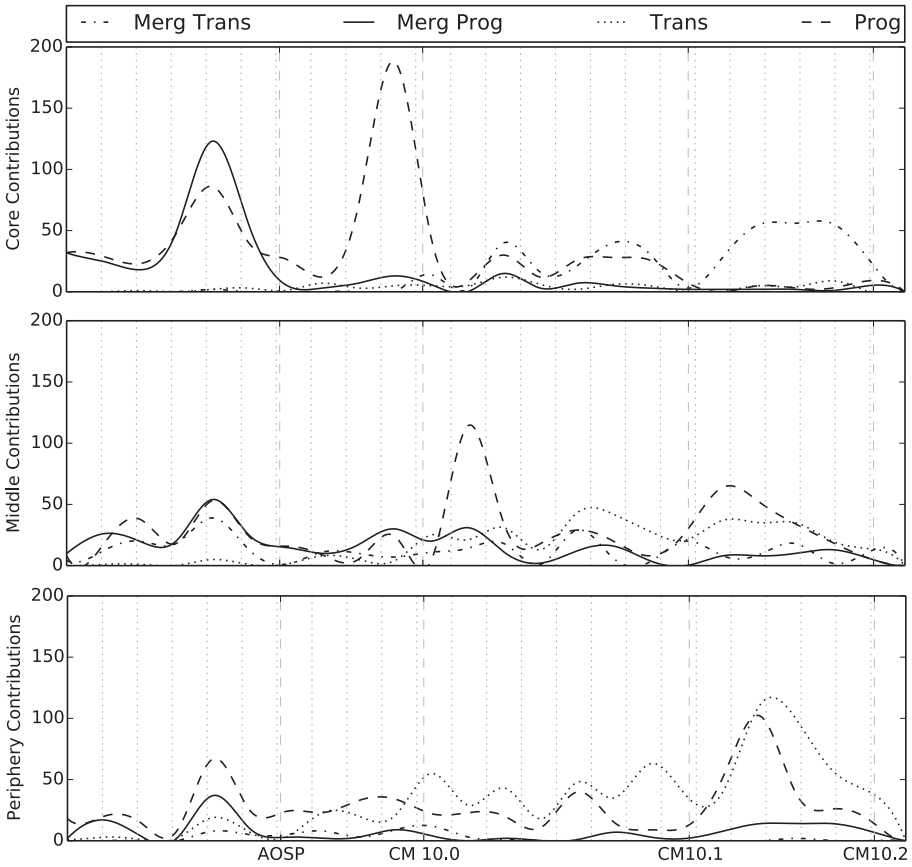


Figure 9.7: Contributions by Type

ways, with the core being predominantly programming and merging, and the periphery being predominantly translations. From Table 9.2 we see that the core focused on two key activities, merging and programming contributions, with them making 48.1% and 46.8% of all core activities respectively. This figure is similar to the middle contributors, with programming and merges being their primary forms of contribution, making 38.7% and 38.2% of their activities. In contrast, the periphery were focused more on translations and programming, with them making 47.5% and 39.2% of all their contributions, while merging was 13.2%.

The type of contribution made by the core changes throughout the time period, from programming to merging, while the peripheral contributions remain largely the same. Figure 9.7 shows when these contributions were made by each of the layers, there we see that the two spikes in activity by the core consisted of different types of contributions, with the first spike seeing more programming merges, and the second spike seeing more programming submissions just before the release of the first stable version. The pattern of activity then changes towards the end, where we see that the core contributors carry out more translation merges after the release of the final stable version. The pattern of activity for the middle sees only one significant spike in activity, taking place soon after the release of the first stable version where more programming contributions take place. Finally, the pattern of activity from the periphery remain relatively constant for all of the types of contributions. This pattern sees a slight increase at the end of the observation period, soon after the release of the final stable version.

To summarise, users can contribute to the repository in three ways, my adding to the core ROM, by porting the ROM to other devices, and by translating the ROM to other languages. Both the rules and the procedures for contributing to the repository then change according to the type of contribution the user is making. The Gerrit review system is used for all types of contributions, and it focuses on assuring quality when the contribution comes from a non-core team member. At the same time, although rules for contributing do not focus on the kind of contributions that are accepted, the Gerrit review, run by the core team, also serves the function of evaluating the contribution based on the view of the

project.

In terms of the key repository contributors, the results point to there being two kinds, users that are key to the CyanogenMod project, and those that are key to external or related projects. Key contributors were found among the official list of team members, with official maintainers and the translators being the most active contributors to the repository. The contributions made by these individuals, which makes up 30% of all contributions to the repository, were focused directly on CyanogenMod. On the other hand, some of the highest contributors who make part of the core and the middle were not directly associated with the CyanogenMod project, with Google developers and independent app developers being among them. The contributions from other sources is a result of the nature of open source itself, where modules can be forked and used in other projects as long as the licenses are followed.

The participation patterns also point to a different approach within the project, where different tasks may be carried out at different times by the core contributors. In terms of tasks, we see that the programming tasks take place at the start of the period studied, with translation contributions generally occurring towards the end. Similarly, we see that the core contributors are more likely to be active during the start of the project, while the peripheral contributors are more active towards the end. Equally, managerial activities such as merging, occurred later on the project, meaning that there were more contributions from individuals that did not have merges privileges.

Breadth of Participation

Data from the association networks show that each of the social groups within the project contributing through the repository have different patterns of behaviour. From the data, it was possible to observe that translators tended to specialise in a smaller number of files, being consistent with each file representing a specific language. In addition, it was also seen that the type of changes made to the repository by peripheral contributors were more diverse that those of the other two groups.

The results suggest that core contributors exhibit low specialisation within

		Users	Contrib	Files	Cent.	Density	Commun.	Modularity
Core	(avg.)	3.769	22.077	49.692	0.275	0.217	3.846	0.174
	(std.)	2.421	23.243	79.851	0.188	0.149	3.158	0.158
Peripheral		44.692	181.538	347.462	0.308	0.186	9.538	0.241
		24.790	159.872	341.459	0.159	0.129	4.255	0.195
Translation		5.923	43.385	82.538	0.240	0.189	6.308	0.131
		5.283	35.495	74.059	0.139	0.122	3.750	0.129

Table 9.3: Table showing the average of the data from the graphs.

the repository, embarking on a wide range of managerial activities that involve numerous files. Table 9.3 presents an overview of the statistics from the association networks from the repository data (the full table can be found in Appendix D). The results show that there was an average of 3.769 core developers making an average of 22.077 changes to an average of 49.692 files in each repository. The activities carried out by core developers in average created networks with a relatively low centrality (0.275) and high density (0.217). The low centrality suggests that core developers do not specialise on a few files but instead modify several files at the same time. This could be consistent with merges and other managerial activities that span a number of files.

In addition, a cluster analysis of the association graphs shows that the activities of the core developers created an average of 3.846 communities with an average modularity of 0.174 in each repository. The relatively low number of community clusters and modularity also indicate lower specialisation, as well as changes that consistently use a number of different files. Figure 9.8 shows the association network for core developers working on the CM Wallpaper module, displaying two clusters with high density.

Peripheral contributors, those who are not mentioned as being formal members of the project, exhibit different patters on behaviour, suggesting higher specialisation and a much broader expanse in activities. Results show that each module had an average of 44.692 peripheral developers, making a total of 181.538 changes to an average of 347.462 files. The association networks created by peripheral contributions displayed a higher average centrality (0.308) but lower density (0.186) than those of the core. A cluster analysis of contributions made by the general developers shows that peripheral

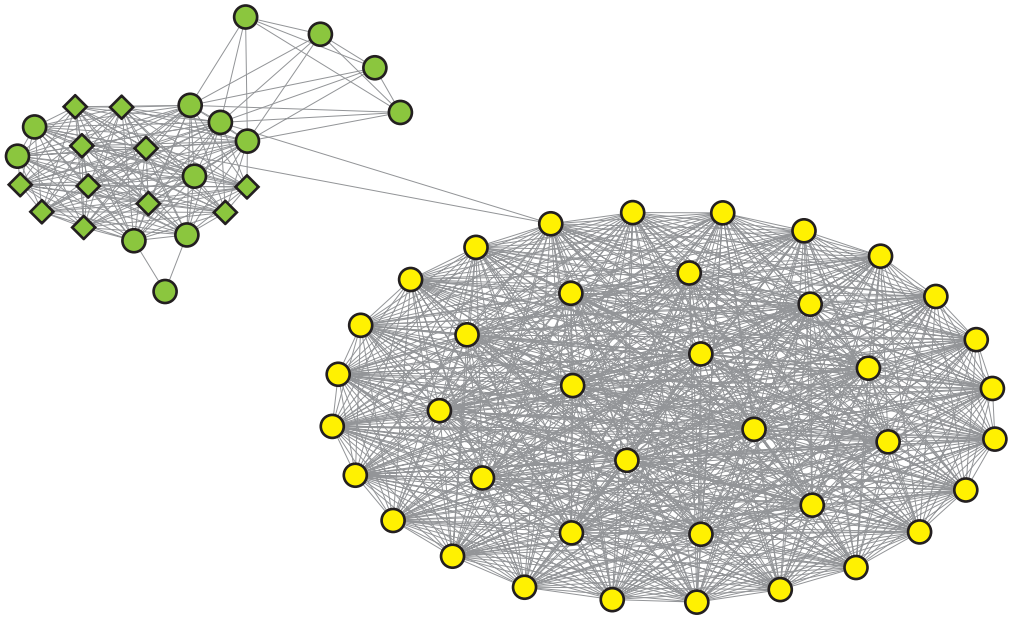


Figure 9.8: Core developers.

contributions created an average of 9.538 communities with a higher modularity of 0.241. The network statistics therefore suggest that peripheral contributors specialise in particular features that span only a few files, therefore resulting in higher centrality and more community clusters. Figure 9.9 shows the association network for general developers working on the CM Wallpaper module, showing a larger number of clusters with varying density.

Finally, translation contributions patters show further specialisation in terms of the type of files being modified. The results show that there was an average of 5.923 translators making an average of 43.385 changes to an average of 82.538 files. The association graphs made from translation contributions displayed a relatively low centrality of 0.240 and a density of 0.189. This means that there were fewer changes made to multiple files by each contributor, suggesting that they specialised in one file rather than in multiple files like peripheral contributors. This is mainly due to the fact that each file represents one language.

A cluster analysis shows that translation contributions created an average of 6.308 communities with a modularity of 0.131 The relatively high number of

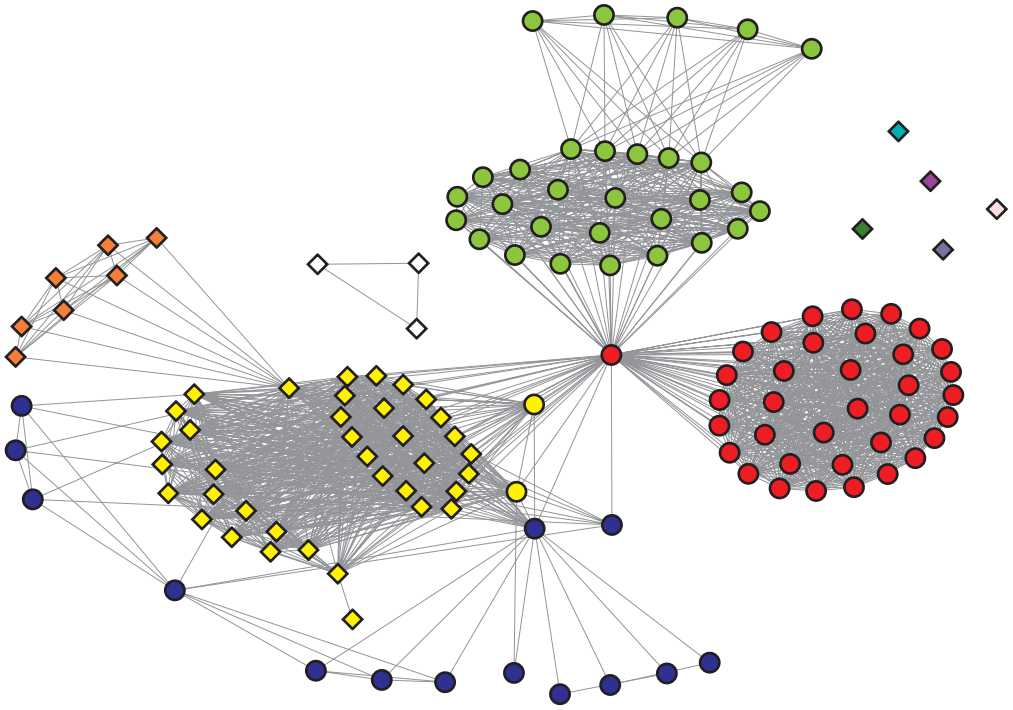


Figure 9.9: Peripheral developers.

communities, together with the lower number of contributors, further supports the idea of specialisation within files. Figure 9.10shows the association network for translators working on the CM Wallpaper module, showing the different communities, the low density, and the focus on translation files.

9.3.2 Wiki

The official project wiki contains 2,265 pages, 1,344 (58.9%) of which have been edited and an additional 931 (41.1%) that have been created but not edited[4]. In total, there were 700 users who made a total 7,123 edits to the pages, over a period spanning 12 months, from November 2012, when the wiki was set up, until January 2014, when the data was collected. The wiki has three main types of pages, one that is educative in nature, others containing user-to-user assistance, and the third type containing general information. The educative pages contain tutorials on how to modify the AOSP and how to use the related tools such as Git,

[4]The wiki has a series of bots which create new pages where relevant information will be added. Such a page could be a translation. This, however, does not mean that the page will contain information, as it depends on users to provide it.

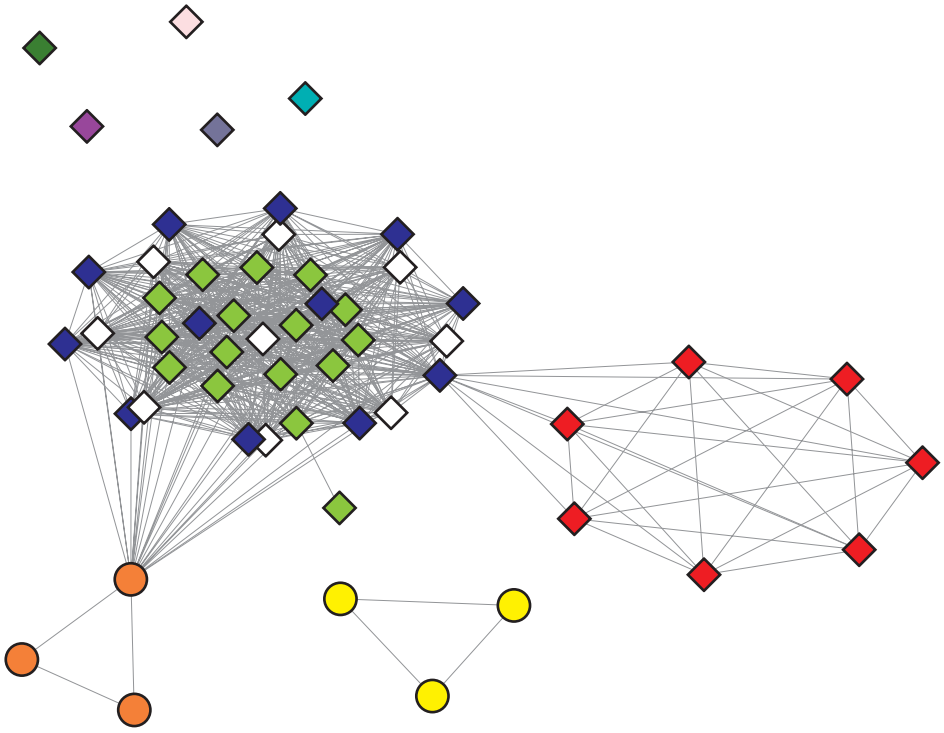


Figure 9.10: Translators.

the Android SDK, and Gerrit. The user-to-user assistance pages focus mainly on device support, informing users how to install the ROM, how to build it from source code, and where to find additional information about the device. Finally, the general information pages provide information about the project itself, such as a short history, a list of official members, and the different roles these members carry out.

The results show that there are differences in terms of when each of the community layers contributes to the wiki, where the core contribute at the start while the periphery are involved in the later stages. This is then reinforced by the time at which the contributions take place, whereby core contributions reached the 50% mark around 39 days after the wiki was first started. This is in contrast to the participation of the middle contributors, who reached the 50% mark 150 days later, on May 2013, with a very sharp increase in contributions around the same time. Finally, the core contributors saw mainly even participation through most of the observed period, with the first 50% of the contributions taking place until August 2013. Figure 9.11 shows the number of contributions made to the wiki divided by layers, and Figure 9.12 shows the cumulative contributions made.

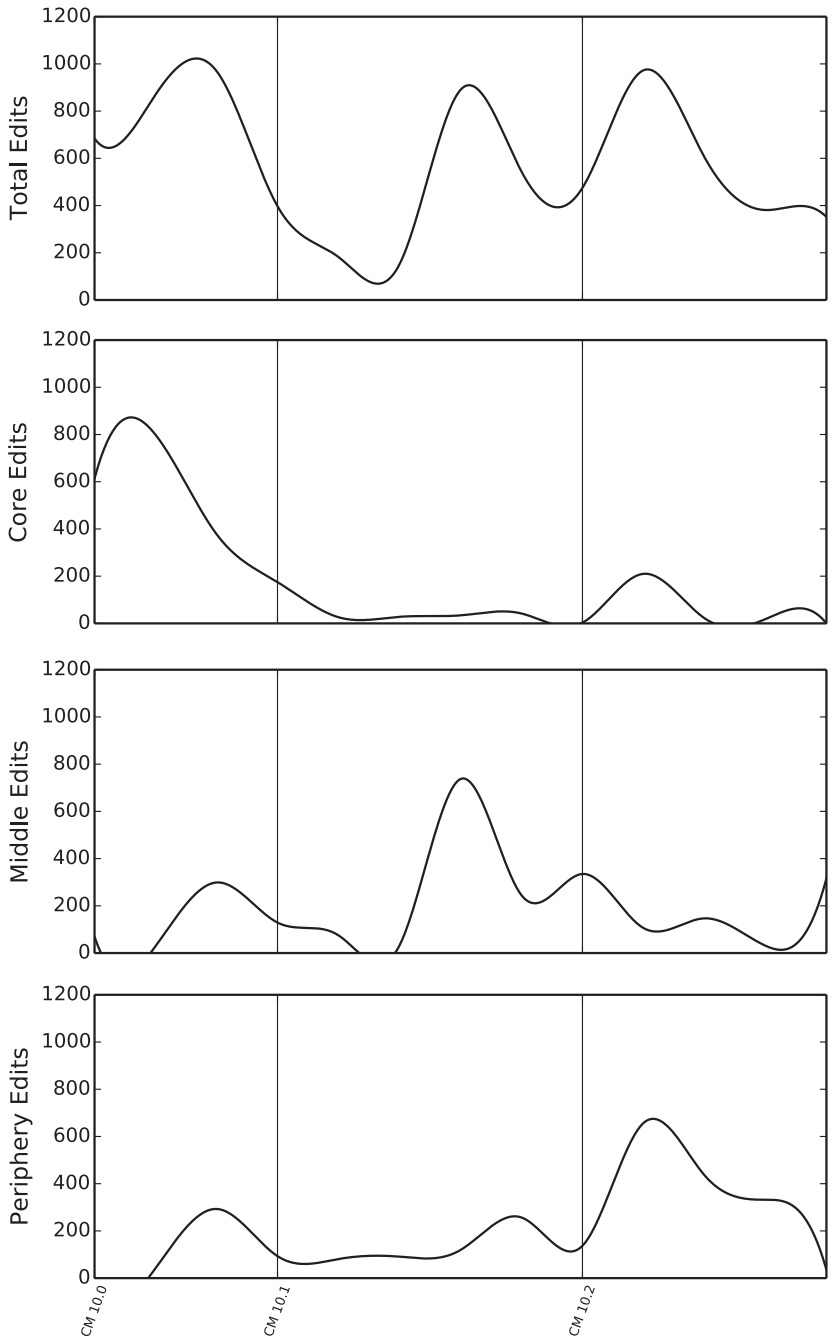


Figure 9.11: Edits to the Wiki

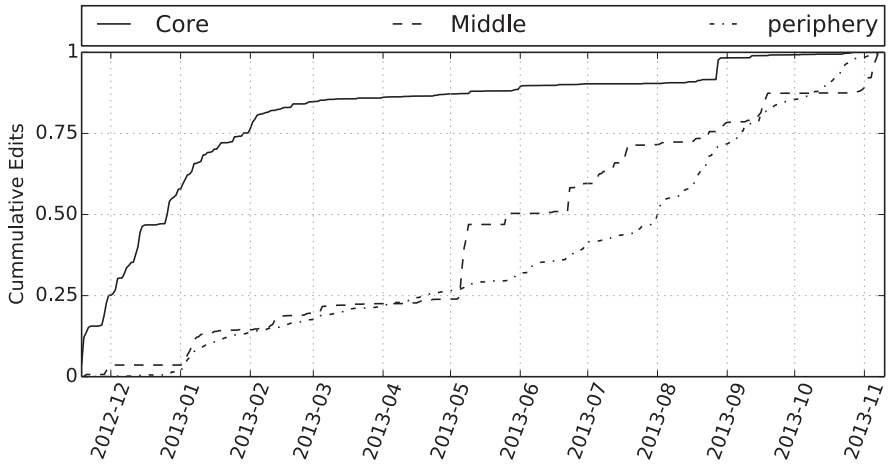


Figure 9.12: Cumulative sum of wiki edits.

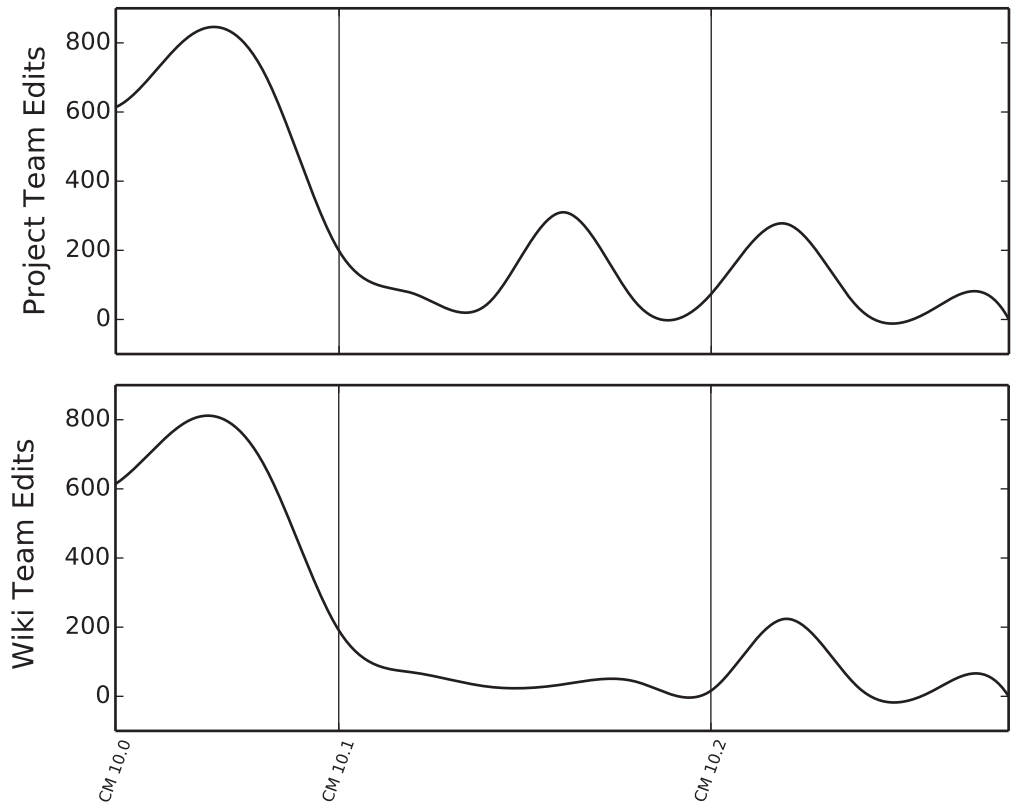


Figure 9.13: Edits to the Wiki

	Wiki Editors		General Contributors	
Date	Pages	Cum. Sum	Pages	Cum. Sum
2012-11	145	145 (0.189)	19	19 (0.033)
2012-12	303	448 (0.583)	37	56 (0.098)
2013-01	45	493 (0.642)	53	109 (0.191)
2013-02	11	504 (0.656)	21	130 (0.228)
2013-03	17	521 (0.678)	27	157 (0.275)
2013-04	5	526 (0.685)	23	180 (0.316)
2013-05	22	548 (0.714)	81	261 (0.458)
2013-06	14	562 (0.732)	50	311 (0.546)
2013-07	0	562 (0.732)	49	360 (0.632)
2013-08	190	752 (0.979)	93	453 (0.795)
2013-09	12	764 (0.995)	59	512 (0.898)
2013-10	4	768 (1.000)	35	547 (0.960)
2013-11	0	768 (1.000)	23	570 (1.000)

Table 9.4: Wiki Page Initiation

In total, the official team initiated more wiki pages that then general contributors, with most of the pages being initiated within the first three months after the wiki was set up. From the 1,344 pages, 768 (57.1%) were initiated by the four official wiki editors, while the remaining 576 (42.9%) were initiated by the general contributors. Figure 9.4 shows the cumulative sum of the time when the pages were initiated, divided into the official editors and the general contributors. From the results, we see that the official editors had two significant periods of initiating pages, the first being from November 2012 until January 2013 where they created a total of 493 pages, which accounts for 64.2% of all the page initiations. On the other hand, the general contributors seemed to initiate pages at a regular rate with no significant periods of higher activity.

In addition, the general contributors, those without a formal role in the wiki, were responsible for making the majority of changes in terms of the amount of information added. From a total of 6,055 page edits, 1,894,854 bits of information were changed in total, with 42.6% (806,945) by 4 of the official editors and 57.4% (1,087,909) by general contributors. Out of the 806,945 bits of information edited by the official editors, 50.5% (407,554) were page initiations. In contrast, for the general contributors, only 33.5% (364,741) of the bits of information edited were for page initiations.

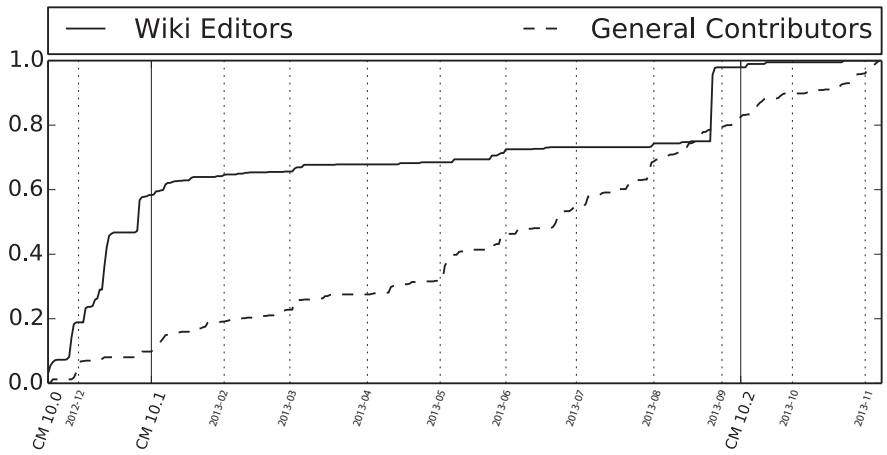


Figure 9.14: Cumulative Sum of Page Initiations

User	Pages
utkanos	491
fattire	314
white	94
flo edelmann	77

Table 9.5: Number of pages edited by Wiki editors.

9.3.3 Forum

There are a total of 67,023 users who made a total of 231,163 posts to the forum. The data spans a period of 50 months starting from May 2010 until August 2014, including the development period of CM 10, with Figure 9.15 looking at all the contributions made during that time. The general trend seems to point to a decline in interaction and usage of the forum, with the highest time of activity being throughout 2011, particularly at the early stages of the year. During 2011, there seems to be two spikes in activity, with the first one being predominantly towards the start of the year and coinciding with the first nightly releases of CM 7. A second spike is seen in October of 2011, which coincides with the first

Users	Pages
574	1
97	2-4
18	6-10
9	11-20
3	21-30
2	31-40
3	41-60

Table 9.6: Pages edited by regular users

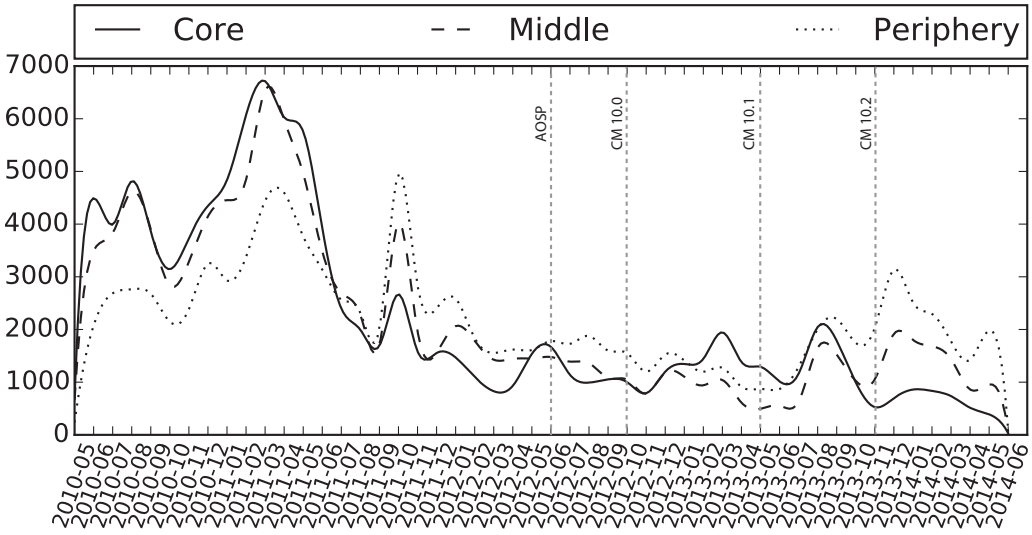


Figure 9.15: Posting patterns for the Forum.

	Core	Middle	Periphery	Total
Devices	164	160	179	179
Users	1,037	9,048	56,938	67,023
Posts	107,029	107,076	107,058	321,163
Experimental	56,372	54,008	51,375	161,755
Stable	50,657	53,068	55,683	159,408

Table 9.7: Statistics by Layers

releases of CM 9. A final increase in activity takes place after the release of CM 10.

In general, the core members of the forum seem to be much more active at the start of the period observed, with the maximum number of posts taking place in March 2011, with a total of 17,699 posts. The middle contributors also reached their peak in contributions in the same month, with a total of 6,517 posts. The peripheral members, on the other hand, were active the most in October 2011, with a total of 4,925 posts.

Figure 9.16 shows the participation patterns for the official project members and for the formal project leaders. From the diagram we see that the project team, which includes the project owner Steve Kondik and community leader Abhisek Devkota.

In terms of the type of discussions that take place, the core members are slightly more likely to participate in experimental discussions while the

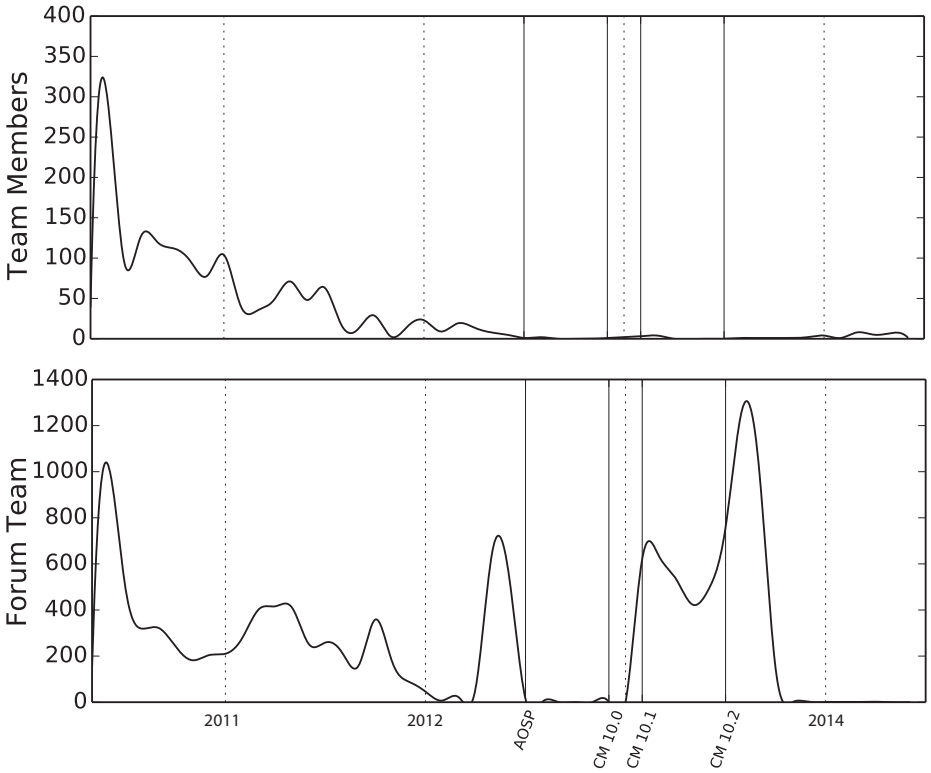


Figure 9.16: Official Project and Forum Leaders

peripheral members participate more on stable discussions. Table 9.7 shows the contributions made by each of the layers and whether they were on discussions about the experimental or stable version of the ROM. In general, there was only a slight difference between the number of posts in the experimental and the stable version, with the larges difference between them being a 5,715 (5.3%) difference between experimental and stable. From the 107,029 posts by core members, 56,372 (52.7%) were in discussions for experimental versions of the ROM, while the remaining 50,657 (47.3%) were in stable conversations. On the other hand, from the 107,058 contributions by the peripheral members, 51,375 (48.0%) were made to experimental threads, while the remaining 55,683 (52.0%) were made in stable threads. Although there difference between both is not large, there is still a slight tendency for peripheral contributors to participate in discussion about the stable version.

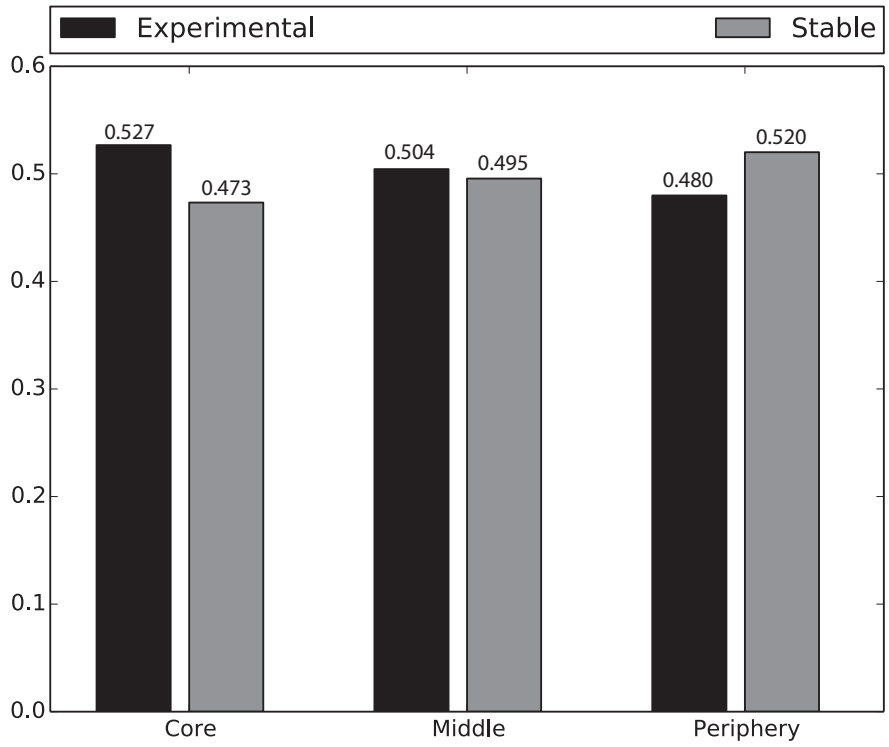


Figure 9.17: Percentage of experimental and stable discussions per layer.

9.3.4 Section Summary

This section looked at the participation patterns for the repository, the wiki, and the forum. The results show that there is difference in terms of when each of the subgroups contributes, with the core contributors being more active at the start of the observation period while the periphery are more active towards the end. Participation in each of the contribution areas is uneven with, bursts of activities taking place at different times, with some significant number of contributions taking place around the release of the stable versions. In the repository the difference in types and timing of participation indicates that the official core developers contribute early in the stage of development while the peripheral contributors participation in the later stages. These differences are further supported by the association networks, which show that each social group had a different pattern of behaviour within the repository.

The same patterns of activity were observed in the wiki, with the official contributors being more active shortly after the creation of the wiki. The wiki was rebuilt in November 2012, which is when we see a spike in activity by the

official wiki editors, who initiated the majority of the pages shortly after that date. The peripheral contributors, those particularly those who were not the official editors, made the majority of the contributions to the wiki, but they were more active at the later stages.

The main difference is that in the forum the official maintainers were given the roles after they were heavily involved in the project while in the wiki they were given the roles which led to them participating even more. The Nightly and Stable sections both had different contributors, where the core and official contributors were active in the Nightly discussion while the majority of the peripheral contributors were in the Stable discussions. The most significant observation was that the users who participated the most in the forum were given formal roles within the community as moderators. Equally, while overall participation decreased, the official project leaders stopped being active in the forum at an earlier period before the release of the AOSP.

Both the repository and the wiki therefore exhibit the same patterns in terms of the timing of contributions from the core and periphery. The repository and the wiki saw the majority of the core activity take place at the start of the observation period with the peripheral activity taking place afterwards. Contributions to the forum have been decreasing, with the project leaders delegating community management roles to other forum members.

9.4 Chapter Summary

This chapter looked at the rules and the social structure in each of the contribution areas, finding that there were significant differences in terms of the rules, how these were applied, and the significance of the core—periphery structure. In Section 9.2, the results show that contributions to the repository, whether these are for programming changes or translations, all had similar rules and procedures. The procedures revolve around the review of submissions to determine their suitability (whether they are compatible with the project's objective), and quality. The main difference, however, was on how the rules and processes were applied, which differed according to who was making the

contribution as well as the type of contribution being made. Contributions that were made by core team members were not typically reviewed, and were instead merged directly to the repository, with the user's reputation serving as an indication of quality. Equally, some translations were also merged directly to the code without being reviewed, with the main reason being that there were no other project members able to verify the quality of the translation.

In addition, the wiki and the forum used different types of rules to different ends, with the wiki focusing on the use of templates and the forum using more permissive rules. The use of templates in the wiki means that the contributors are restricted to adding pre-determined topics, therefore reducing the type of information that is available. The templates themselves were created by the official editors with the original aim of facilitating contributions and achieving a consistency in terms of what the pages contained and how it looked. The rules of the forum were all focused on providing guidelines to members as to what could be discussed and where, with rules focusing on limiting the mention of pirate software and certain topics during the experimental stage.

The contribution patterns in the repository and the wiki show that there is a large amount of effort from the core developers in the early stages, with peripheral contributors being involved in the later stages. In the early stages of software development we see that major work is carried out in terms of the implementation of new features in the repository and the creation of new information pages in the wiki. In the later stages, when the periphery carry out the majority of the contributions, we see that the activities are focused primarily on the maintenance and improvement of existing features and the editing of information pages.

Finally, the different groups observed within the repository also displayed different patterns of activities. The data shows that peripheral contributors specialise in a few files but carry out a broad range of activities. In contrast, those that were classified as being official translators were more likely to modify a fewer number of files, specialising therefore in a particular language. Finally, the core developers would have the broadest pattern of participation by equally modifying large numbers of files within the repository.

Part IV

Interpretation and Conclusion

Chapter 10

Interpretation

10.1 Introduction

The objective of this study is to determine how an open source project manages its different forms of contributions. The review of the literature in Chapter 3 highlighted the current limitations of the user innovation literature, finding that the literature on communities did not fully address issues of governance in collective innovation where users contribute through a range of activities. Equally, the literature on open source communities focuses primarily on the group of programmers, therefore basing much of its understanding of governance on activities based around software development. As a result, this thesis is guided by three broad objectives: To determine (1) the different ways in which users can contribute to an open source project; (2) the effect different types of contributions has on social structure; (3) The effect different types of contributions has on governance.

This chapter presents the analysis of the results. Section 10.2 presents the analysis of the findings for the first research question, looking at the implications on membership and project success. Section 10.3 presents the analysis for the second and third research question, looking at social boundaries within the project and their implication in terms of governance. Section 10.4 presents the analysis for the fourth research question, looking at how the project itself is governed, looking at the key issues of leadership and long-term planning at the project level. Section 10.5 presents an overview of the User

Organisation as a distinct form of governance for user-led open source project. Finally, Section 10.6 presents a discussion on the implications of the findings in terms of leadership and meritocracy, project life-cycle, success factors, and the core-periphery structure.

10.2 How Users Contribute

The first research question aimed to identify the various ways in which users contribute to the project. The findings presented in the previous chapter suggest that users can contribute to the project through a wide range of activities; programming, non-programming, administrative, and maintenance. In addition, the results also show that the group of users that contribute directly through software development is significantly smaller when compared to those that contribute through documentation or user-to-user assistance. The approach taken to the study of contributions to the project provides a new perspective on project membership and the impact that this has on governance. In addition, the findings suggest that certain forms of non-programming contributions may be important for the success and further growth of the project.

The discussion for the results for the first question is structured as follows: Subsection 10.2.1 discusses the nature of membership and their implications in terms of project boundaries. Finally, Subsection 10.2.2 looks at the implication of the various forms of contributions in terms of network externalities, project adoption and success.

10.2.1 Contributions and Membership

The aim of the first research questions was to take a different perspective of how users engage in an open source project to identify the different ways users can add value to the project. This first research question therefore took a different approach from the existing literature, which focused on a single practice for the development of a single product. By taking this approach, it was then possible to understand the implication of the different types of contributions in terms of

membership and roles within the project.

The results showed that users contribute to the project through a wide range of activities that span a number of products and services. These findings challenge traditional assumptions in the open source literature which rely on practice as the determinant of community membership, where project members are identified as those that carry out any form of software development. When looking at how users contribute to CyanogenMod, however, it was possible to identify non-programming contributions to the repository, such as graphic design and translation. These results expand previous findings on membership and roles within open source communities, such as Nakakoji et al. (2002) and Crowston and Howison (2005) where only programming roles are listed. While the focus of a single practice has been more evident in the literature on open source development, the same can be found in the user innovation literature, where the focus still remains on a single practice leading to a single output.

In addition, the results show that there are a larger number of users that are involved in the production of complimentary products rather than the primary software. These complimentary products are software and information that aid in the adoption and maintenance of the operating system. A key example is the translation of the software, an activity that does not add any functionality, enhances or fixes software code. The benefit of translation, however, is that it allows other users that speak different languages to use the software, therefore increasing its user base and therefore potential contributors.

The results also suggest that membership to the project does not focus on a user's involvement in the development of the operating system. For instance, membership to the CyanogenMod project is based on whether users contribute to any of the project's products and services, rather than just the primary output. This presents a different perspective from the one normally taken by user innovation studies, which have focused on single product innovation as a determinant of membership. Early user innovations studies on Lead Users, for instance, focused on the development of specific products, like scientific instruments (e.g. von Hippel, 1976) and computer aided design systems (e.g. Urban and von Hippel, 1988). With the emergence of user innovation

communities, studies kept the same focus on products, such as software library systems (Morrison et al., 2000) and mountain bikes (Lüthje et al., 2002). The findings for this study, however, suggest that the boundaries of the project are larger that the primary output, and may also include a number of related products and services.

When extending the boundaries of the project by redefining project membership, it is possible to see the programming contributions in the context of a much larger workforce that engages in diverse practices. This in itself has implications in terms of how users add value to the project and what the significance is in terms of project success and future performance. It may be possible to explain the impact non-programming contributions, such as user-to-user assistance and translation, has on the project by the way in which these help create positive network externalities.

10.2.2 Impact of Network Externalities

The different perspective on user involvement in the project, and the resulting reinterpretation of project membership, leads to the discussion of the importance of the size and variety of the user group for the success of the project. The literature on open source and user communities states that, in terms of knowledge sharing, larger user groups are beneficial to user-led projects. The findings for the first research question, however, suggest that the variety of activities carried out by other users can also have an impact on project success. For example, while non-programming user contributions, such as translation and documentation, do not contribute directly to the production of the primary output, they are still important to the project as they facilitate the adoption of the software and increase its user-base. Their importance to the project is supported by the rapid growth of interest in the project, as well as by the number of devices that are officially supported by the project, as described in Subsection 6.4.3.

The findings for the first research question suggest that some forms of contributions, specifically user-to-user assistance and tutorials, may increase user numbers through positive network externalities. Katz and Shapiro (1985) describe network externalities as occurring when "the utility that a user derives

from consumption of the good increases with the number of other agents consuming the good." (p 424). From the three sources of network externalities identified by Katz and Shapiro, they describe positive consumption as the increase of post-purchase service resulting from an increase in consumers. In the CyanogenMod project, the documentation and user-to-user assistance provide these post-purchase services.

The results for the first research question therefore suggest that the varied types of contributions can potentially increase the user base of the project by lowering the barrier of entry. The initial popularity of the CyanogenMod meant that within its large user base there were some users who are not programmers but may still feel motivated to contribute to the project. This therefore may have led to an increase in other forms of contributions, such as project documents and non-technical user-to-user assistance. The existence of non-technical support and documentation may have itself allowed less technical individuals to use the software and become future contributors.

Although the literature on open source has acknowledge the importance of a large user base, these have focused on their implication on knowledge creation, knowledge sharing, and problem solving. This study adds to that by stating the importance of a large user-base in increasing positive network externalities, therefore making it possible to lower barriers of entry and adoption through supporting products and services. The practitioner literature highlights the importance of larger groups of contributors in aiding the creation of novel ideas (Cox, 1998) and making the Linus Law possible (Raymond, 1998; Crowston and Howison, 2006). Similarly, the user innovation literature states the importance of large user community for problem-solving (Lakhani and von Hippel, 2003) and for a more efficient innovation process (Hienerth et al., 2014). These two therefore focus on the development of the software, and does not look at the impact group size has on attracting other contributors.

10.2.3 Section Summary

The first research question finds that users can contribute to the project through a range of different practices and help in the development of a number of products

and services for the same project. These findings have implications in terms of how membership to a project is defined and what is the significance of the new membership definition. By having a much broader perspective on who belongs to the project, it is possible to identify the larger variety in forms of contributions that add value to the project. This wide range of types of contributions can have the added advantage of being the source of positive network externalities for the project, as it can potentially lower entry barriers through the provision of supporting products and services.

10.3 Community Governance

The second and third research questions aim to determine group boundaries within the project, and how each of the resulting social groups are governed. The findings show that each type of contribution generates its own community of practice, with their distinct shared repertoire and mutual engagement. Each of these communities of practice is governed differently depending on the type of contributions being made and the tools that are used. These results present for the first time a study of the social boundaries within the project and the impact that these have in terms of governance.

This section is structured as follows. Subsection 10.3.1 discusses community boundaries and the effect these have on community membership, where different communities of practice are generated as a result of a distinct shared repertoire and mutual engagement. Subsection 10.3.2 goes on to discuss community governance, pointing that there are a number of differences that are based on either the type of contribution being made, or the technology being used.

10.3.1 Group Boundaries and Community Membership

This study finds that the different types of contributions create different communities of practice within the project. For instance, it was observed that the organisation, delegation, and coordination of activities for programming and translation were done in separate places by different individuals. Equally, the discussion around the changes and improvements to the wiki, at an operational

level, did not involve general contributors who were engaged in other practices. The creation of different locations for coordinating actives, in terms of dedicated and private communications between members, further highlighting the different social groups within the project. Finally, the different communities of practice within the project overlap and are connected through boundary objects, such as the repository, and through brokers, primarily the project leaders.

The literature on open source and user innovation communities has not explicitly looked at work group boundaries within a project (West and Lakhani, 2008); existing studies implicitly draw project and community boundaries around a certain practice. For instance, the boundaries of open source communities has predominantly been drawn around users that contribute to the project through software development. This simple view of community membership has also led to the definition of project membership itself, where the project is assumed to be formed of a single community of practice. When taking the communities of practice as an approach, the fundamental aspects of community boundaries and membership are often implied and not explored.

As the findings for the first research questions suggest, users can contribute to the project through a number of different practices, which leads to differences between project members in terms of how they interact with each other, what goals they share, and how they use tools. Wenger and Snyder (2000) state that Joint Enterprise, Mutual Engagement, and Shared Repertoire are the key dimensions of practice as the property of a community. Taking this as a basis, the findings therefore suggest that the different types of contributions lead to the creation of different communities within the project as a result of differences in enterprise and socialisation.

The results suggest that each type of contributions developed its own joint enterprise, where the act of setting and pursuing specific goals differed according to the output and the role users have within the community. For instance, the primary goal of the wiki was to create a one-stop location that provides all the information required. On the other hand, the translation community had a different goal of extending the use of the software to other

language users. Similarly, although programmers and translators worked on the same output, their goals also differed. The results find that contributors in each community would create their own operational goals and develop their own processes in order to pursue them.

In addition, each type of contribution had its own distinct area of interaction. This was seen most in the translation community, where the users had their own IRC channel, a dedicated forum section, and their own Google group. The significance of this is that the majority of the interaction between the translators took place in a separate location from other groups of contributors. The same could be said about the users contributing through graphic design, where design-related activities would again be discussed in a separate locations. These findings therefore suggest a more complex social structure with overlapping communities of practice than previously observed.

While each type of contribution led to a difference in enterprise and engagement, a shared repertoire was observed across some activities. This was more evident between programming, translation, and porting, all which used the repository and same release schedule. What made the difference, however, was how these tools were used and how the rules were enforced. For instance, while both translation and programming activities used the repository and the review system, the process of translation relied more one on the use of templates. The same can be said about the individuals that port the software to new devices, who use the same tools as the developers, but interact in a different location and in different groups.

Within this new form of social structure, it can also be seen that the repository, and other areas where contributions and interactions take place, act as boundary objects between the project's different communities of practice. For instance, the repository can act as a modular boundary object, whereby the files contained in them will determine the boundary of the type of contribution being made. Programming contributions will therefore more likely to be made in files that contain code, while translation contributions would be made in XML files. For instance, translators are more likely to modify .xml files rather than .java files, something that was observed in the repository's association

network. In addition, similar to what Wenger (1999) noted, each of the practices influences each other. In the CyanogenMod project, this influence can be seen in the form of rules that apply to all practices, particularly of when to contribute. The different communities of practice, therefore, while they engage and set goals at different locations, are connected through the boundary objects that have the ability to change and influence each other.

This finding therefore adds more to the literature on open source and user communities by finding specifically how the different types of contributions lead to the formation of distinct communities of practice through differences in socialisation and joint enterprise. While the different communities may share tools, techniques and routines, key factors such as repeated interaction goal setting and attainment play an important role in creating boundaries between them. This therefore means that the users are more able to identify and interact with individuals that carry out similar activities rather than with other individuals working in other aspects the project. For instance, while users worked on similar modules for the software, it was the activity they performed within them that determined what work group they belonged to and therefore who they interact with.

10.3.2 Community Governance

With the second research question finding multiple communities of practice within the project, the third research question looked at governance within each community. The results suggest that there are significant differences in governance mechanisms in each community of practice. For instance, the rules that govern participation in the forum are different from those in the repository. Similarly, the delegation of administrative roles, a key component of coordination and governance, differs between communities and is influenced by a range of factors including technology and politics. This study therefore adds to the current understanding of open source and user innovation community governance by challenging the established assumptions about the nature of leadership and coordination at the community level.

The results in this study suggest that the rules on contributions not only

differ among communities of practice, but are also enforced differently. For instance, the rules and procedures that are placed on those contributing to the wiki are different in nature from those that contribute to the repository. At the same time, although the rules that are the same for all those that contribute to the repository, they are enforced differently depending on the type of contribution being made. Programming contributions go through a review stage, while translation contributions may not always do so. The results therefore show that there is a significant difference in governance between communities of practice.

These findings challenge the current assumption that project rules apply to all contributors equally. For instance, O'Mahony and Ferraro (2007) find that the rules of the project apply to all contributors, including the project leaders. Equally, von Krogh et al. (2003) suggest that there are procedures that contributors must follow in order to join the project and gain more rights, implying that these rules apply to everyone regardless of the role contributors have within the project. In the CyanogenMod project, however, the rules for contributing, and therefore joining the project, differ according to the type of contribution and technology, and therefore do not apply to all contributors in equal terms.

A second key aspect of the findings relates to the core-periphery structure and its relationship to community leadership and coordination. The results find that the relationship between core users and administrative roles differs between each community. Currently, the accepted criteria for selecting leaders is based on their activity in the project, where the more active contributors are, the more likely they are of obtaining administrative roles. User activity has typically been measured in two ways, through the number of contributions made to the software (e.g. Mockus et al., 2002), or through the number of interactions within the community (e.g. Dahlander and O'Mahony, 2011). The findings in this study show that the wiki's core-periphery structure supports this view, as the wiki's official editors were the most active. On the other hand, in the community of programmers, some of the users that were most active were not in fact part of the CyanogenMod project.

The key differences between programming and wiki contributors highlights the presence of a series of unexplored variables that have an effect on delegation of administrative roles and activity measurement. For instance, in the community of programmers, administrative roles were given to contributors who had been active in the project for the longest period of time, therefore showing the importance of length of involvement rather than quantity of contributions. The length of involvement in the project may have made it possible for users to be familiar with the project and its owner. For the wiki, on the other hand, the delegation of administrative roles may have preceded the user's high level of activity. This could be as a result of the nature of the administrative role in the wiki, where each administrative action leaves a record on the software and then measured, something that may not be the case in other technologies.

10.3.3 Section Summary

The second and third research question determines that each type of contribution generates its own community of practice with its own form of internal governance. The results show that boundaries within the project are formed along the types of contributions due to differences in operational objectives and in patterns of socialisation. The operational objectives differs between each of the contributions since these are focused on the type of contribution being made, and activities are coordinated in different communication channels that are specific to the type of contributions. In addition, each of the communities has its own governance mechanism, where rules and procedures are applied and enforced differently, depending on a number of factors, such as reputation or level of activity. Finally, in some cases the core-periphery structure was not found to be consistent with the dominant view of leadership and influence in the community.

10.4 Project Governance

The fourth research question dealt specifically with the idea of how the project itself, consisting of multiple communities of practice, is managed and coordinated.

The study finds that the project is coordinated centrally by a group of individuals that have been active in the project from the start and have been delegated responsibilities by the project owner. The relationship between this leadership team and the general community is such that they are able to establish and implement long-term goals.

This section is structured as follows. Subsection 10.4.2 looks at the nature of project leadership in relation to the general contributors. Here we discuss the implication of the multiple communities of practice in terms of project leadership and how these are all integrated and coordinated. Lastly, Subsection 10.4.3 discusses the issue of goal attainment, which deals with how the project is able to set goals and manage its general contributors in order to meet them.

10.4.1 Leadership and Meritocracy

The literature on open source governance has generally focused on merit as the key factor that determines which individuals gain leadership roles in the project. The meritocracy of leadership means that individuals are able to obtain formal roles within the project as a result of their involvement in it. Studies like Roberts et al. (2006) have looked at the issue of meritocracy within the Apache community as the system through which individuals attain new roles. Similarly von Krogh (2003) holds that the meritocracy is due to the individual's knowledge and expertise, and it was this through which they can gain formal roles and more privileges in the project.

On the other hand, O'Mahony and Ferraro (2007) and Capra et al. (2008) find that meritocracy alone does not determine whether the individual will get a leadership role, with quality and importance of contribution being as important. In their study of the evolution of governance in the Debian open source project, O'Mahony and Ferraro found that a contributor's knowledge or skill alone were not the deciding factor as to whether they would gain a better position in the project. Instead, a second thing that comes into play when determining if the users get privileges or not is how widely their contribution is being used and their role in community building. O'Mahony and Ferraro therefore hold that the traditional view of meritocracy in open source projects has been somewhat over

simplified, and that the popularity of contributions and community building is as important.

In the CyanogenMod project, however, the wide use or importance of the code submitted does not necessarily mean the individual will be given a key role in the project, as can be seen from the case of the Apollo music app. While Andrew Neil created the Apollo music app, which is now a standard app in the CyanogenMod ROM, he does not have an official role in the project. This was mainly due to the fact that the app was originally open source, and so the CyanogenMod team were able to fork the code and change it to meet their needs. The Git system also means that when ever Andrew Neil made any fixes or changes to the app, these could be synchronised to CyanogenMod's versions. Therefore, this means that the continuous work of Andrew Neil on the Apollo app could be used without having to make him a part of the official project.

The results also suggest that, besides meritocracy, political and historical factors have an influence on the delegation of leadership roles at the organisational level. O'Mahony and Ferraro (2007) observe that the growing number of users and contributors was a key factor in the evolution of governance structures in the Debian project. In a similar way, the CyanogenMod project was also affected by the growing number of users and contributors, prompting the project owner to define and delegate managerial roles to aid coordination. The managerial roles were delegated to individuals who had been part of the development team from the start of the project, and who had therefore been able to build a close relationship with the project leader; this group has traditionally been referred to as the core team. In addition, the core team has remained unchanged in the last five years, giving the project stability and helping its members generate a mutual understanding of the project's goals. This therefore shows that the coordination mechanisms and organisational structure of the CyanogenMod project emerged as a response to the complexities of coordinating a growing group of users and the close relationship between project members.

This study therefore adds further clarity on the issue of meritocracy by finding that meritocracy is applicable at the community level, but obtaining key positions becomes more complex at the project level. First of all, by the contributions made

in the forum and the IRC, meritocracy is the deciding reason of why individuals become or obtain key positions within the community. In the production areas, on the other hand, individuals that had official roles did not obtain these through merit, but instead through the constant interaction and the personal connection to the project leader. This therefore implies that the attainment of project-level managerial roles, and those for production communities, although initially based on skill, later developed a political dimension.

To summarise, the idea of meritocracy in open source project is more complex than current studies suggest, therefore reinforcing the findings from O'Mahony and Ferraro (2007). Obtaining a leadership position in the project depends not just on skill, commitment, or community building efforts, but could also have a political and historical dimension to it. This is most important when seeing the leadership roles at the organisational level, where the group of individuals that set and implement the project's goals largely obtained their positions by being friends and working together from the early stages of the project.

10.4.2 Leaders and General Contributors

The results presented for the fourth question raise some issues in terms of the relationship between the project leaders and the larger group of contributors. As stated in the previous sections in this chapter, the dominant assumption is that user-led projects consist of a single community of practice working on one output, therefore using the terms communities and project almost synonymously. This has led to the implicit understanding that community leaders are also project leaders. The findings in this study, however, show key differences between project leadership and community leadership, in terms of their role and the function in the project.

One of the key differences between community and project leaders is that project leaders are responsible for determining the overall direction of the project, which includes any future goals they would like to achieve, and setting up plans in order to achieving these goals. Community leaders, however, are generally there to impose the rules of the project and are restricted to their community of practice and the managing of community level operations. The separation of

both roles observed in the CyanogenMod project leads to the observation that the community leaders may not necessarily be project leaders.

Since there is a difference between community and project leadership, this further raises questions on the issue of how these communities, in aggregate, are coordinated at the project level. The current approach to governance relies on what de Laat (2007) refers to as internal governance, which focuses on the internal characteristics of communities. Other issues such as the community's environment and context within a project have been less explored. By identifying the different user communities with their separate rules and structures, it is worthwhile to understand the relationship between these.

The findings of this study help in understanding the relationship between these communities by looking at their position within the project. The study finds that the communities responsible for producing the software and documents are directly managed by the project leaders, while communities involved in providing services are not. This therefore shows that the project leaders have different relationships between communities, where it could be suggested that it is down to their perceived importance to the project.

10.4.3 Goal Attainment

One of the key findings related to software development in the CyanogenMod project is the issue of goal attainment, where the project is able to set specific goals and then implement ideas in order to achieve those goals. The results find that the project leaders are able to set goals and attain those through a number of different mechanisms, primarily through a combination of being lead users, gate keepers, and having the ability to outsource some solutions. This challenges the traditional view of user-led projects which often does not deal with the issues of long-term planning.

This study finds that long term goals in the project are implemented by the leaders at a great cost, lowering the need to rely on general contributors. A key example of this is the development of the supporting software, which was written by one of the project leaders in its entirety in order to meet the objective of lowering the barrier of entry for non-technical users. This therefore

suggests that, in order to implement strategy, project leaders were responsible for implementing action at a great cost in order to meet project objectives. The approach used was similar to the lead user model of innovation, where the community provides support for the innovation activities but is not involved in defining or shaping the innovation itself. Taking this lead user approach means that there is one individual who owns and controls the innovation, while the community has a supporting role through providing new ideas or minor improvements. The implication of this is that the lead user or owner is able to change the particular innovation without requiring the communication of long-term objectives with the general contributors, something which has previously seen as important (e.g. O'Mahony and Ferraro, 2007).

This study also finds that the CyanogenMod project is able to meet long term objectives by acting as gate keepers and controlling what is merged into the project. The selection of what should be merged into the software is done by the project leaders who act as gate keepers and are able to determine what contributions are included in the software based on their long-term objectives. Studies such as Dahlander and O'Mahony (2011), for instance, state the importance of managerial roles and the way in which users are able to distinguish and evaluate contributions according to a number of criteria. These studies have so far used quality and code compatibility as the key criteria for evaluating programming contributions, where it has to be assumed that, as long as those two criteria are met, the changes will be approved. The results in this study, however, find that an extra criteria is whether the contributions are in accordance to the long-term goals of the project.

The final way in which the project leaders are able to attain long-term goals is through outsourcing. In the CyanogenMod project, this was done through the use of open source apps that were incorporated into the operating system. In the literature, Linus' law states that the 'wisdom of the community' lies in its collective knowledge, where a problem will be found by one person, understood by another, and solved by yet another person (Crowston and Howison, 2006). This therefore means that projects often rely on general contributors to implement new features when the project leaders themselves do not have the knowledge

or the resources to do so. The results in this study, however, show that project leaders also have the option to outsource solution from other open source projects, therefore removing the need to rely on general contributors.

Finally, the results also suggest that goals are set and communicated before action is taken, once again challenging the current understanding of how communities work. The results show that in the CyanogenMod project, the leaders will typically set a goal and make a public statement on their social media websites. These goals are, for instance, the need to move to a new version of the AOSP, or the implementation of a new feature such as encryption. After communicating these goals, the leaders would coordinate activities in order to achieve them. This challenges the current assumption on coordination where Yamauchi et al. (2000) find that "...a traditional approach — coordination precedes action — is not appropriate in dispersed collaborations." (p 337). The CyanogenMod project, however, shows evidence that coordination does, in some cases, precede action.

10.4.4 Section Summary

The fourth research question finds that the management of the project itself requires leaders to engage each community distinctly and to use different techniques when implementing long-term goals. The findings show that the project leaders may in some cases be different from the community leaders, and therefore have different relationships with the general contributors. In addition, project leaders are also in charge of setting long-term objectives for the project and implementing these throughout the communities by formulating rules and procedures that affect all types of contributions. Finally, the findings also add to the understanding of project goals and how to implement them by observing that project leaders use the three techniques of being the lead user, acting as gate keepers, and outsourcing solutions.

10.5 User Organisation

The findings in this study suggest that the CyanogenMod project has a more complex form of governance than previously seen in other open source projects. The central coordination of multiple social groups within the project and the ability to set and implement long-term goals is a change from the traditional view of collective innovation by a user community. This study therefore proposes that CyanogenMod resembles a User Organisation rather than a user community.

The user organisation represents and new form of collective production and governance which allows project owners to maximise user input and retain control over the project. The user organisation is a distinct form of open source project governance, with a wide range of user involvement and a top down approach to decision-making. It represents the joining of a diverse range of communities of practice working together to produce goods for a user-led project, all who are coordinated centrally by a leadership team. Finally, unlike the Bazaar and modularity-based style governance, it introduces the element of long-term planning by project leaders, therefore making a distinction between strategy and operations.

In the user organisation, control over the direction of the project can be achieved by controlling where key activities in the innovation process take place (see Figure10.1). By controlling the timing of involvement by the general contributors, project leaders are able to limit the influence users have over product specification, limiting their involvement to maintenance activities. Rules on when general users can contribute mean that the locus of innovation activities during the implementation stages remains with the project leaders, allowing them to add new features and services based on their long-term goals. As general contributors are included in the maintenance stage of the process, their contributions are restricted to smaller changes that will either improve or fix existing features and services.

The control over the innovation process by the project leadership therefore presents a new form of governance for participation, which is distinctly different from the Bazaar and the modular concepts often associated with open source software development. The Bazaar concept of governance, as expressed by

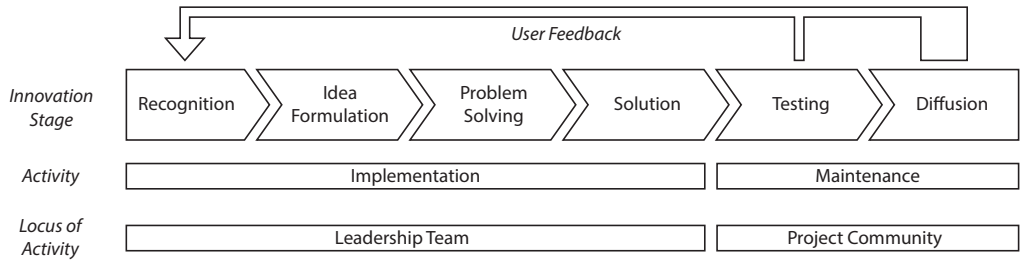


Figure 10.1: Innovation pattern in a user organisation.

Raymond (1998), relies on unrestricted communication and action from all the contributors, creating a black box of processes that leads to a cohesive product. As for the modularity concept of governance, as expressed by Baldwin and Clark (2006), relies on the structure of the software in order to isolate the impact contributions have on the software. In the user organisation, it is the control of the stage of innovation activities that gives project owners control over the software, allowing for a cohesive product to emerge from a number of contributing communities.

In addition, in Bazaar-style governance, the leadership team and general contributors are assumed to belong to the same social group, allowing therefore the exchange of information and new ideas. In the user organisation, the leadership team are isolated from the general contributors in terms of decision making and social interaction, leading to a different relationship than that of the core—periphery structure seen in communities. By creating two different social groups in terms of project leadership and general contributors, the former may approach users as a source of resources and skills external to the project. This relationship mimics that of the hybrid communities where firms choose to access a user base through the creation of a community (see Dahlander and Magnusson, 2008), with the difference that the project is non-commercial therefore having a different effect in the perception of the general contributors (Sharma et al., 2002).

Finally, it is this centralised and hierarchical structure which can facilitate the coordination of multiple contributions to the project, while at the same time focusing on long-term project goals. The hierarchical structure of the user organisation (see Figure 10.2) allows greater levels of control over the different

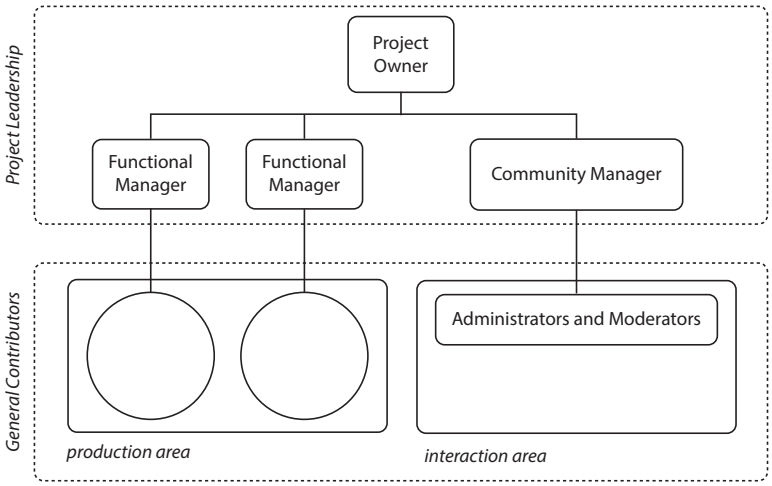


Figure 10.2: Structure and Hierarchy in a User Organisation

type of contributions by designating functional managers to each of these. In addition, these functional managers are given roles by the project leaders, where the close proximity and interaction helps in consolidating an understanding of project objectives and future activities.

10.6 Discussion

The previous sections looked at the significance of the findings in relation to the user innovation literature. It highlights the difference in governance mechanisms found in the CyanogenMod project when compared to the current understanding. It then proposes the User Organisation as a separate and more formal structure for the governance for modern user communities that provide a wide range of products and services.

This section presents a discussion of general issues that deal with the study of open source projects in general, and is structured as follows. Subsection 10.6.1 looks at the emergence of governance and identifies three key stages. Subsection 10.6.2 then looks at the project life cycle in open source project and its importance to determine community and governance evolution. Subsection 10.6.3 presents a discussion on the evaluation of open source project success, looking particularly at community size as an indication of success. Finally, Subsection 10.6.4 looks at the implication of the findings in relation to

the approach of determining the core and the peripheral structure based on contributions to the repository.

10.6.1 Emergence of Governance

This study finds that governance in the CyanogenMod project has gone through three stages: (1) de-facto, (2) design and implementation, and (3) consolidation. In addition, despite the continuity of the project ownership, the project leaders were able display different types of leadership in order to support each of the governance stages. The key factors shaping project governance were both the establishment of a dedicated ICT infrastructure and the challenges of managing a growing and diverse number of contributors. These findings add to the previous understanding on the emergence of governance by identifying two additional factors that lead to a change in governance purpose and style.

The Stages of Governance

The CyanogenMod project went through three stages of governance: (1) de-facto, (2) design and implementation, and (3) consolidation. The de-facto stage took place during its early period, characterised by unilateral decision-making and lack of formal rules specific to the project. Similar to what was observed by O'Mahony and Ferraro (2007), the project owner, Steve Kondik was able to implement software features and other changes without discussing design elements with general contributors. In addition, at this early stage there are no explicit or written rules or formal roles within the project that guided the behaviour of other users. What has to be noted, however, is that the project was hosted on an established public space, the XDA Forum, both users and project leaders were therefore subject to the rules imposed by the forum. Therefore, while there were no rules for the project itself, CyanogenMod users were still subject to other governance mechanisms. This first stage of governance ended when the CyanogenMod project established its own infrastructure, giving it the ability to design and implement their own governance structures.

Both the design and implementation of governance, the second stage,

happened simultaneously as the project moved away from the XDA forum to its own infrastructure. During this stage, the project owner was responsible for designing and then delegating official roles, as well as establishing rules regarding how and when other users could contribute. Decisions regarding the design of the new governance mechanisms were made unilaterally, meaning that there were no public discussion regarding how the roles were going to be defined and who these would be delegated to. This unilateral action was facilitated by the stability and legitimacy of the project's leadership. This is in sharp contrast to what was observed in the Debian project, where O'Mahony and Ferraro (2007) note that general contributors were consulted during the design stage as a result of the leadership's lack of legitimacy. It could therefore be argued that because there was no change in ownership in CyanogenMod, the project leaders were able to retain their legitimacy and make unilateral decisions regrading governance.

The third and final stage of governance was its consolidation with explicit rules and norms on when, how, and where users could contribute to the project. This final stage sees the building and continual modification of key documents that highlight the rules for contributing and interacting within the project. These documents were found in various parts of the project's infrastructure, but could be accessed primarily through the project's wiki. The rules contained within these documents applied to both specific locations of interaction as well as to the wider project. In addition to the rules, pages were set up that described official roles of project members, particularly those that had leadership positions, therefore consolidating and formalising their roles.

Factors Shaping Governance

There were two main factors that affected and shaped the evolution of governance in the CyanogenMod project: the growth in number of contributors and the project's ICT infrastructure. These two factors changed governance as the new mechanisms that were established were focused primarily on dealing with these two. For instance, it was the growth in the number of user-consumers and user-contributors which propelled the project to establish its own infrastructure. At

the same time, because ownership of the ICT gave them control of the locations, it was able to establish and consolidate its own governance structures. These two factors are significantly different from those observed by O'Mahony and Ferraro, who focus more on ownership and leadership.

In addition, it could also be argued that the emerging governance structures are a result of the factors that lead to it. For instance, in the CyanogenMod project, it was the size of the community which led to a change in governance, which therefore focused on mechanisms that addressed this fully. On the other hand, in the Debian project, in was the lack of leadership legitimacy which led to a change, therefore focusing the emerging mechanisms to address this issue.

It could therefore be argued that, the factors affecting governance could also have led to governance being shaped around it. For instance, the catalysts and the events that drew the need for change in governance were markedly different. Where for the Debian project it was a crisis in legitimacy of leadership and, most importantly, a crisis in the legitimacy of its leaders. As a result, the second stage of the emergence required that first the governance structure be more entwined with how decisions are being made and by whom. On the other hand, the CyanogenMod project never went through a crisis of legitimacy in leadership, but rather a coordination crisis, therefore the focus became the creation of a governance structure that would facilitate the coordination of contributions.

Consistent with single case study research, however, the evolution of governance and its resulting structures may be specific to each project. For instance, not all projects require a large number of contributors (see Krishnamurthy, 2002) or go through a leadership crisis, and may therefore not have the need to develop governance mechanisms to address these issues. Equally, even if other projects face similar problems, it may not result in the same governance mechanisms, as internal factors may have an effect on which mechanisms are adopted. While noting the different phases of governance in the Debian project, O'Mahony and Ferraro stated the need for different types of leadership skills in order to deal with it. This therefore reinforces the understanding that leadership style and its context are important for

performance (Fiedler, 1971).

The findings of this study therefore suggest that in the CyanogenMod project, the breadth of participation from user-contributors precipitated the emergence of governance structures. The project's emergent governance mechanisms were therefore designed and implemented to specifically address these issues, leading to a structure that focused on the coordination of a wide range of contributions. Finally, since the legitimacy of the leadership was never truly challenged, the project owner was able to implement the new governance mechanisms unilaterally throughout the project's ICT.

10.6.2 Open Source Project Life Cycle

Some studies, particularly Wynn Jr (2004), Lattemann and Stieglitz (2005), and O'Mahony and Ferraro (2007), have attempted to determine the stages within open source projects and the characteristics of governance that are inherent within them. The usefulness of this is that, as Wynn Jr (2004) argues, it may be interesting to understand the stage where the project is in order to better manage the project and improve its chances of success. This therefore makes the assumption that at each stage of the life cycle, particular activities dominated over others and therefore governance must change in order to enable their coordination. This section will therefore discuss how applicable both models were to the CyanogenMod project, and will then propose a new approach that could be taken in order to determine a project life-cycle.

Despite the importance of determining the stage of the life cycle the project finds itself in, this may not have been explored in greater detail by the literature due to the implied unstructured and emergent approach to development, making development stages seem recurrent and cyclical. This thesis, however, has found this not to be the case in the CyanogenMod project, where clear stages of the development process can be observed and are imposed throughout the entire project. Therefore, it may then serve as a good example of how projects have and how they can use the life cycle in order to determine the stage at which a given project is in.

Limitations of Current Life-Cycles

The two main models on the evolution of governance in open source projects represents a difference in opinion in terms of the factors that drive the evolution of governance, whether it is the size of the community or the change in leadership. Quinn and Cameron (1983) review the different changes within firms and their evolution, noting the differences between the different life-cycle models. The approach taken by O'Mahony and Ferraro (2007) in his emergence of governance is similar to that proposed by Greiner (1972), which, according to Quinn and Cameron (1983) focuses on how organisational problems are solved. In O'Mahony and Ferraro (2007), the major organisational problems revolve around the issues of leadership and control over the project, and the reaction this got from the general community. As with many single-case studies, O'Mahony and Ferraro's findings are specific to the Debian project, its environments, its internal resources, and its history. The model presented by Wynn Jr, on the other hand, focuses on the more generic variable of community growth as they main impact on community governance.

The emergent nature of the projects makes it very difficult for there to be a specific fit or for there to be a precise way of determining at which stage the project is within the life cycle. Both models used were not adequate enough to encompass or to explain the activities that take place and were not there to take into account the way that the project has evolved. For instance, O'Mahony and Ferraro (2007) looks at the Debian project in particular, which then includes the change in leadership, which therefore has a major effect in the way that governance evolved in that project. Indeed, O'Mahony and Ferraro states that it is some of the defining events which then lead to a change from one stage to the next. The main issue is, however, that these major events may be particular to the projects being studied, and may therefore not be completely generalisable.

In the case of O'Mahony and Ferraro (2007), the main event that signals the change in governance from one stage to another is the change in leadership and authority, together with its impact on the general community. In the CyanogenMod case, on the other hand, the project owner is still the initiator, Steve Kondik, and as a result, the issues surrounding leadership or the systems

of leadership established in the Debian example are not the same. As a result, the same type of governance structures and mechanisms are not the same in both of them, such as the democratic forms of decision-making. On the other hand, taking into account the pressures that project size has on the emergence and the change on governance, whereby projects then see the need to add more governance mechanisms in order to coordinate and to manage a larger crowd. This is what was seen in the CyanogenMod project, where size was the key factor for job delegation and for establishing rules of contributions.

Other studies have looked at the size of the community as a catalyst for change in governance. Lattemann and Stieglitz (2005) and Wynn Jr (2004) match the size if the community to the traditional four-staged life cycle of introduction, growth, maturity, and decline, and then allocate particular governance characteristics to it. This second model seems to be a better fit when it comes to determining the stage at which the project it at the moment, taking into account leadership, coordination, level of commitment, and structure. In addition, this model presents a better perspective because it also takes into account the size of the community and the effect it has in terms of the coordination and other governance issues.

One of the key limitations of matching community size with the life cycle is that there may be issues regarding how the size of the community is determined and how growth is dealt with by each project. This thesis looked at the issue of membership to an open source project, expanding from the traditional community of programmers to a community of users who contribute in a number of ways. A similar problem may arise when determining the size of membership, where a lack of clear definition as to who is included and who is not, may present a false size. In addition, as mentioned before, the emergent nature of each project, specifically the way in which projects deal with issues of community size, may make it difficult to formulate a general framework for an open source project life cycle. Such a difficult characteristic is that of the social structure, where formal and decentralised structure are, according to Wynn Jr, more prevalent in mature projects. This type of structure, however, may be a result of significant events within project, as O'Mahony and Ferraro (2007) points out in its framework, and

therefore there could be questions as to how generalisable these are.

Both these models of life-cycles may be limited in their generalisability due to the uniqueness of the project studied by O'Mahony and Ferraro (2007) or by the lack of empirical evidence in Wynn Jr. As seen above, the characteristics of the life cycle stages proposed by O'Mahony and Ferraro are built on the findings from the Debian project, where the governance structure evolved as a result of key indicants within its project. This therefore makes the approach of matching life-cycle stage and governance characteristic somewhat limited, in that it may not be generalisable to a wider range of projects. In addition, the issue of contributor size (rather than download numbers) also play a bigger role in affecting and bringing change to governance, and must therefore be taken into account as a factor. Wynn Jr, for instance, takes the number of downloads and proposes download metrics as a way of determining the life cycle, but provides no empirical evidence that these two are linked.

New Frame-Work For Open Source Project Life Cycle

A new framework for open source life-cycle should also incorporate the key activities that take place at each stage, while at the same time remain flexible enough to take into account the iterative process of software development. From the results we saw that the programming patterns were different throughout the period observed, with programming contributions taking place more at the start of the project and managerial merges towards the end. The patterns therefore show a similarity between what was taking place and the iterative software development model, where stages like design, implementation, and maintenance, seem to be better ways in which to classify project stages. The results presented in this thesis therefore look at the development process of the CyanogenMod project, which may have come out of necessity and it could have been emergent, but they do share similarities with established forms of development.

The key similarities between the traditional software development and the process observed in the CyanogenMod project lies within the general stages of implementation and maintenance. The first stage is on the writing of code to implement new features, something which can be seen in the core ROM on

CyanogenMod towards the first stages of the development. For instance, the implementation can also include a wide range of activities, such as Forking or modifying existing projects, in order to add a new functionality to the ROM. The second stage, on the other hand, looks at a wider range of activities that focus on improving and enhancing existing features. These activities, in the case of CyanogenMod, can include translations and bug fixes.

The issue of time is also one that needs to be taken into account, where the linear aspect of it, as implied by previous models, does not take into account the iterative nature of open source development, where experimental and maintenance tasks take place at the same time but on separate locations. It was de Laat (2007) who also pointed to the use of experimental and stable versions in order to compartmentalise activities of innovation and maintenance. In this study we see that there are three key stages in the development, the experimental, release candidate, and the stable releases. These are both linear and cyclical, and most of the time is better suits the way that the project is working and how individuals engage with the project. These three stages, however, are particular to CyanogenMod because of the peculiarity that the AOSP is being released, therefore giving the project external milestones where production begins and the previous one ends. In another project, which does not depend on external projects, at least two stages of experimental and stable, may continuously run parallel to each other, allowing also a space for experiments.

The parallel experimental and stable model of software development has already been seen in the industry, particularly in web browsers, where the most common open source projects use this combined two system. Google, for example, maintains part of the Chromium project, which is an experimental open source web browser which, once stable, becomes the commercial and stable Chrome. Similarly, Mozilla's Firefox also has an experimental version running parallel to the stable one, called 'Nightly Builds'.

The difference in activities, particularly between initial development and maintenance, is highlighted by Midha and Bhattacherjee (2012) who explicitly state that there are different tasks involved in each stage. This difference in the

type of participation may therefore require different types of governance. The tasks in this stage are therefore characterised, in general terms, as being

- Correction of Errors
- Modifications in order to perform new tasks, and
- Performing old tasks under new conditions

In the CyanogenMod project, therefore, we see that these three activities are more prominent towards the end of the observed period. The patterns of participation also show that the general contributors, or the peripheral contributors in general are the ones that are in charge of maintenance tasks. Although in the CyanogenMod example, it should also be included the translation of the text into other languages, which also has the most amount of contributions.

Having seen that maintenance is a second stage of a larger process, we can further expand on the first stage of development, where new ideas are implemented into the project. It is during this stage that we see very different types of behaviour and different types of activities, such as the implementation of new features and the continual prototyping. While during the maintenance stage we see the emphasis on adding or improving what already exists, in the initial development stage we see the experimental prototyping.

The general consensus that open source software goes through constant and continual prototyping is partly supported by the CyanogenMod project. This type of development process therefore takes into account the fact that not all the features will be known at the design stage, and not all requirements will be understood. Larman and Basili (2003), for instance, make the argument that this type of development, iterative and agile, is best suited for projects where there are some unknown features.

While the iterative model of software development does take into account the unknown and emergent nature of open source, it does not entirely state the location of where this is taking place. For instance, while CyanogenMod clearly shows the difference between the experimental and stable stages, and the

importance of keeping these separate, this has to also be placed into context when looking at the development process. As a result, we then get to understand that different individuals within the projects are active in each of these stages, carrying out particular forms of contributions. From this, we can therefore see that in the experimental location, the core team official project members are carrying out design and experimentation tasks. While on the stable versions, general contributors are carrying out maintenance tasks, such as bug fixing and code improvements.

10.6.3 Project Success

The issue of governance is of importance to open source projects because it can have a direct effect on participation and development efficiency. Poor governance may reduce the number of contributors and lead to a decrease in project activity as well as lead to misdirection of effort. As a result, the number of contributors to the project has been traditionally seen as an indication of project success and correct governance. First of all, projects like Linux and Apache began to draw the attention of academics almost ten years after they began, when they had a large user-base and were therefore seen as being relatively successful. This therefore gave them the potential of studying a project that was now very well established and where the number of individuals using it meant that their future was secure.

Taking this into account, the CyanogenMod project has only been active since 2008, making it under ten years, with different types of contributions and activities in the project. The results presented in this study show that the project has not always received continuous contributions or participation evenly, with the levels of activity changing through time. This can be observed more when looking at the number of contributions that were made to the forum from the time it was started, which shows that the level of participation has been declining. Viewing user contribution and engagement as an indicator of success, may give the impression that the project is losing momentum and may be in the process of ending.

The basis of using the level of contributions to assess the success of a project focuses on achieving Linus' Law through the economies of scope, where the

larger the community is, the larger the potential pool of resources. Having a large user base has been considered to be important because it is from this group of individuals where potential new contributions can emerge. For instance, Schneider et al. (2013) and Kuk (2006) note that users spend a period of time being observants, or lurkers, before they contribute to the project. It is during this period where users are able to determine whether they are able to contribute and are also capable of learning new things. Therefore, this learning activity could potentially lead to allowing individuals to go from being part of the conversation, to being part of the production side.

From the perspective of software development, however, the CyanogenMod project does not seem to require this large pool of knowledge and resources through its large community, as it can benefit from the use of freely available open source modules to compliment their project. While certain forms of contributions are needed, such as porting and translating, the actual modification of the ROM does not solely depend of the availability of users contributing. In order to meet their goals in terms of software design, the core team take existing open source projects and merge them into their ROM, therefore being able to find solutions without relying on their user-base. As a result, the success of meeting their software goals is not dependent on the size of their user-base because they have different ways of acquiring new ideas and implementing them.

To summarise, it may not be possible to determine the success of the CyanogenMod project because of its relatively short time on existence and because it may not rely on the general contributors for feature implementation. The ability for the leaders to implement new features by forking other open source projects, allows them to obtain key parts of the code without having the need to wait for individuals to self-select in order to implement the features they want. As a result, the number of active contributors submitting code to the project may not be an indication of success. On the other hand, looking at the wide range of contributions, such as user-to-user assistance, may be a better indication of project success in terms of its current popularity.

10.6.4 Core Periphery

This thesis looked to understand how governance is achieved in a project that contains a large number of communities of practice within its boundaries. The work of Kevin Crowston is based on understanding the structure of these communities of practice in a bid to understand how governance takes place and how these communities operate. Within his research Crowston et al. (2006) sought to clarify and reconcile opposing concepts in terms of two key subgroups, core and the periphery, and to explain their importance within the project. The first notion of core and periphery relied on the notion that a small number of individuals do a large portion of the work, while a larger group of individuals carry out only a small part of the work. The second notion of core is that individuals who have greater influence within a community are those that contribute the most, and those that contribute less have less influence. The resulting assumption therefore is that the core developers, based on higher participation, are also more likely to be project leaders and hold managerial roles within the project. Crowston et al. therefore argue that understanding who the highest contributors are may also present a good indication as to who are the ones that are in charge of the development process and governance.

The methodological stance that was set up by in Crowston et al. (2006) tests three different methods of defining the core, and tests this on 115 open source projects, finding that Bradford's rule of thirds, explored in greater detail by Black (2004), was the most accurate. The data itself was gathered in terms of contributions made to the repository for 115 projects, where the user names were taken and counted, while at the same time comparing it to a list of official developers obtained from the project documentation. Crowston et al. (2006) states that using Bradford's law for determining the core and the periphery is most useful in this case because it is both easy to implement and it also manages to capture both the official developers and the ones that contribute the most.

This thesis supports the use of this method for both areas of production, the repository and the wiki, where the lists of both core team and core contributors consists of roughly the same individuals. In the results we see that the core contributors were also members of the core team, who were given the role of

managing and editing the information in the wiki. The patterns of contribution and the number of contributions that these individuals made also supports the method applied by Crowston et al. (2006) in that both of them matched completely. On the repository, we see a similar but distinct patterns, where the core team were also found within the first two groups, therefore supporting and still being consistent with the idea of the flexibility in the groups.

The main problem that was encountered, however, was based on the nature of the project and the open source practice of forking and re-using existing code from other open source projects, which then leads to cross-pollination. One of the biggest problems that was seen was in the repository, where some of the core contributors were not the same as the core members, and in fact, these could be traced back to the Android project. Key contributors such as Winson Chung and Michael Jurka are listed as being Google employees that work in the Android project as software engineers. Taken from a module level, we also see that the initiator and developer of the Apollo music application, Andrew Neil, is seen as a key contributor, which would then imply that he was heavily involved in CyanogenMod. The reality, however, is that the Apollo music app was forked and incorporated into the CyanogenMod ROM, therefore bringing with it all the data on the contributions made by Andrew Neil. This therefore points to a cross pollination, in terms of programming contributions, which may give a false impression as to the involvement of certain individuals in the project.

As a result, what the cross pollination means is that counting the number of contributions to the CyanogenMod project as a basis of determining their influence on the project may be somewhat misleading. In the above examples of the Apollo music app, while it was Andrew Neil who contributed the most with the initiation of the music app and most of its development, it was David van Tonder (user name *dvtonder*) who forked and incorporated this app into CyanogenMod. Therefore, the key decisions of adding or implementing a new feature, a key aspect of governance, were carried out by a users with a relatively low level of contribution to that particular module.

The same cross pollination issues that were observed for software development, however, were not seen in the wiki, and this could be because of

the nature of the project and the control and availability of the information. The wiki itself saw a much purer version of the way that people contribute to a project, where the all effort was made specifically for the project and for the wiki pages. As a result, the scope of the wiki made it possible to create a clear picture and to include only the individuals that were contributing directly to CyanogenMod and therefore those that were important for its production.

To summarise, the methods used to determine core periphery community structure may not reflect the nature of the open source ecosystem, which can lead to a misguided view of who the important project contributors are. There is a major problem in counting how much people contribute because of the nature of the open source ecosystem which means that there is cross pollination in the sense of how software is written. This therefore means that determining who the key individuals are within the project simply through quantitative methods may not be as accurate as portrayed earlier. Therefore, because of the open source approach, it may be of use to take into account the roles that contributors have in the project as a better indication of their importance, rather than just the count of contributions.

10.7 Chapter Summary

This chapter presented the analysis of the results of this study. The objective of this study is to determine how an open source project manages its different forms of contributions. In order to do so, it highlights three questions regarding how users engage with the project, how different communities are governed, and how all contributions are managed at the project level. The findings show that the governance mechanisms used in the CyanogenMod project are much more formal that those that have been implied in community based productions, typified by the Bazaar governance. This study finds that the project structure is highly hierarchical and roles equivalent to functional managers have been established and delegated by the project owner in order to coordinate the different activities. This therefore implies that the governance of this project resembles more an organisation rather than a community.

Individuals involved in the project can be divided into two distinct groups, Project Leaders and General Contributors, each with different roles and levels of influence within the project. The project leaders have managerial roles based on activity, such as Lead Programmer and Lead wiki editor, and are responsible for communicating and implementing the goals set by the project owner. The general contributors have operational roles where communities of scope are required, such as user-to-user assistance and maintenance, and where additional skills or resources are needed, such as translation and porting.

To address the first question on user involvement, this study finds that users can engage through a large number of ways besides programming, such translation, porting, graphic design, and documentation. These different types of contributions have an effect on the way that the project attracts new members, in that it creates positive network externalities that lowers barriers of entry to new members. This can potentially have a beneficial effect on the project's future by increase the user-base and the pool of resources.

Addressing the second research questions, each type of contribution creates its own community of practice based on their distinct enterprise and location of socialisation. Each type of contribution has its own operational objectives, with a set of progresses in place that will help in attaining those objectives. In addition, each type of contribution may also share the same ICT and the same processes for contributing, but each of these will still be managed in a different way at the community level.

To address the third question on community management, this study finds that each type of contributions will create its own community, with differences in the way that they are governed in terms of rules, procedures, and their enforcement. In addition, coordination mechanisms within each of the communities differs, where programming contributions are reviewed and assessed by project leaders as a form of coordination, while the wiki relies on the use of templates. Also, while some contributions may share the same set of rules and procedures, their enforcement may differ, as was the case of translation submissions which in some cases could not be reviewed due to lack of skills.

To address the third question on project wide governance, this study finds that there is a small social group composed of project leaders who discuss, sometimes in private, the project's goals and objectives. The leadership team was, in most cases, found to be a distinct social group from the community leaders, and their role included long-term planning. Once a course of action has been established, they will then communicate these goals to their respective community leaders or they would implement it themselves at a high cost.

Chapter 11

Conclusion

11.1 Introduction

This study aimed to determine how a modern open source project manages multiple forms of contributions from users. The study of collaborative innovation in the user innovation literature has largely remained silent when it comes to the processes that are involved in the innovation process, relying instead on other fields of study. The user innovation literature sees open source projects as the archetypical user community, but this has had the effect of limiting their study to groups of programmers. Recent changes in the way open source projects work, has allowed projects to attract a wide range of contributions from its users, therefore making software development one of the many ways in which users can add value to a project. This study therefore aimed to address this gap in the user innovation literature by determining how a modern open source projects coordinates the various forms of contributions from its users.

Using the CyanogenMod project as the single case study, this study uses three data sources and applies semantic analysis to tackle the research questions. The study extracts data on participation from the project's repository, forum, wiki, and IRC through a period spanning from June 2012 until January 2014. The title and description of each contribution was analysed using semantic analysis in order to determine the different types of contributions being made. In addition, project documents were used to determine key rules and processes for contributing to

the project, and semistructured interviews were used to further add clarity and additional information.

The study finds that each type of contribution creates its own community of practice, with its own area of interaction. In addition, the governance mechanisms in each community of practice were found to be different in terms of structure, rules, procedures, and their enforcement. Finally, it finds that project leaders are responsible for managing the different forms of contributions by controlling the scope and timing of contributions. This study therefore proposes that CyanogenMod's governance resembles more a User Organisation than that of a user community. The centralised coordination of the multiple forms of contributions and the ability to set and implement long-term goals presents a different form of collective innovation by a user community. Most significantly, the control over the stages of innovation provides project owners control over the direction of the project, which facilitates long-term planning.

This chapter is structured as follows. Section 11.3 contains a summary of the study' theoretical, empirical, and methodological contributions. Section 11.4 then talks about the limitations of the study, focusing primarily on the generalisability of the findings. Finally, Section 11.5 discusses future areas of studies from both a theoretical and methodological perspective.

11.2 Summary of Findings

The objective of this study was to identify how an open source project manages a wide range of contributions. To do so it sets out four research questions in order to determine how users contribute, what communities there are within the project, how these communities are governed, and what are the implications for project governance. This section presents a summary of the results.

The first research question was on how users contribute to the project. The results show that users contribute to production through a wide range of activities that involve a number of different skills other than programming; such as translation, documentation, and graphic design. In addition, users were also engaged in activities that required economies of scope, such as user-to-user

assistance and porting. The results therefore provide evidence that users contribute to an open source project through a wide range of activities.

The second research question was on the different communities within the project. The results of this study show that each type of contribution creates its own community, where members interact with each other through a number of different ICTs that focus on their practice. For instance, although programmers and translators both contribute through the review system and the repository, each group will interact through their own chat room and social media. These results therefore suggest that each activity creates its own community of practice.

The third research question focused on community governance. The results showed that each of the communities has its own form of governance, which can be determined by the tools being used, the rules, and their enforcement. The rules for participating are predominantly based on the ICT, as can be seen in the similarity between programming and translation rules, and how these differ from the forum and the wiki. In addition, there was also a difference in the way that rules were enforced, where translation contributions were less enforced than programming ones. The results therefore show that each community uses a different style of governance, which depends on the ICT being used and the availability of skills and resources.

The fourth question focused on project governance. The results showed that the project is governed centrally by the project leaders who are responsible for establishing and implementing project goals, which are discussed through private conversations. The majority of the large changes to the project were also implemented by the leadership team, with the emphasis being on executing decisions made in private. Finally, rules and managerial roles for the project are formulated centrally by the leadership team. The results therefore show that the management of the multiple communities within the project is carried out by a group of leaders that interact privately, and who are responsible for setting and implementing project goals.

11.3 Contributions to Knowledge

This section is structured as follows. Subsection 11.3.1 discusses the theoretical contribution of the this study in relation to the literature on open source and user communities. Subsection 11.3.2 looks in greater detail at the empirical contributions, which is the study of a modern open source project with its own infrastructure and close hardware integration. Finally, Subsection 11.3.3 looks at the methodological contributions, a method for determining the dynamic nature of the core periphery relationship, and the use of text analysis to determine the type of contribution being made to repositories.

11.3.1 Theoretical

This study contributes to the user innovation literature by adding clarity to the processes involved in the coordination of activities in collaborative innovation. It addresses a gap in the current literature where communities have been traditionally studied as the source of information for both lead users and firms by focusing more on the mechanisms that help collaborative development. It has also been able to identify boundary objects and brokers within the new context of a complex structure with multiple overlapping communities. By looking at the wide range of contributions to an open source projects, this study was able to determine that governance mechanisms will differ between communities of practice due to the tools being used and the practice itself.

In addition, this study proposes the User Organisation as a new form of governance and social organisation involved in collaborative innovation, making it distinct from the traditional user community. This form of governance takes two aspects, first being from the community level, and the second being from the organisational level. At the community level we see that there is a division of labour not just within programmers or those who contribute to the repository, but also between those that contribute to the project in general, which includes things like graphic design and other related activities. Most of these practices and the tools that are used, generate communities within themselves, meaning that, instead of the social structure being one group connected through interaction, it is

a collection of groups connected through a hierarchy. The user organisation also resembles a hierarchy in that it is composed of multiple communities of practice that report to a leadership team.

In the traditional literature of open source projects, the emphasis on its social structure has been the community of practice, particularly that of software developers. The traditional understanding is that the social structure of the open source projects resembles an onion, with a number of different layers that are interdependent and work together to produce the software. The boundaries themselves in terms of who belongs to the project are typically seen as those who contribute through code or bug reporting, as in the information systems literature, and those that programme and those that use the software, as in the user innovation literature. The emphasis, however, still remains on the software development process and all the activities that are related to it, therefore ignoring other forms of contributions.

This study therefore expands this view by extending project membership to those that contribute through any means, therefore taking into account a larger range of practices and their respective communities. The CyanogenMod project is a prime example of allowing users to contribute to the project through a wide range of means, not just through programming, but also through translation, graphic design, and user-to-user assistance. In order to take all these practices and different types of contributions into account, it was therefore necessary to move away from the community approach to social structure and look at something more complex.

This study was also able to identify boundary object and brokers within the project that help link the different communities of practices. While ICT has been viewed as some of the important factors that help collaborative innovation, it is also useful to see how these have been used as boundary objects that link the different communities within the project. Similar to what was found in Star and Griesemer (1989), each community of practice interacted differently with the ICT tools used in the project. In addition this study was also able to identify the project leader's role as brokers in the project, who were responsible for spreading best practices and establishing and enforcing rules within the project.

By expanding the definition of contribution and participation, both the social structure of the project and its governance gain different characteristics, separate from both an individual community and a system of communities. Demil and Lecocq (2006) attempt to understand what the community governance structure in open source is, and what are the characteristics that make it so. In their theoretical study, Demil and Lecocq make the distinction between the different forms of social structure and the continuum that was established by Thorelli (1986) leading from a marketplace to a network, to a hierarchy. This point particularly related to the relationship between the core open source project and the related communities that are around it which provide supporting products and services, with the network term being here replaced with the term system. As Thorelli (1986) discusses, it is also the aspect of power[1] which determines that structure and the relationship within it.

Although previous studies have looked at the relationship between different open source communities, the relationship has been looked at as a system, where there is cooperation and interdependence, but not control or power from one community to another. There have been previous examples of large open source projects receiving a wide range of types of contributions other than programming, where the structure resembles more a system. An example of a system, or network-like, structure of the related communities around a particular open source project can be seen in a different number of established projects, such as Linux. The Linux kernel, despite its popularity, has not developed a graphical user interface, leading to other projects, predominantly KDE and Gnome, to assume that position. Equally, user-to-user assistance is offered through a wide range of forums and discussion groups, which can be found in a wide range of different areas, such as linuxforums.com, linux.org, or linuxquestions.org among many others. Finally, IRC chat channels are typically divided among the distributions, in other words, there are some that are dedicated to Arch Linux, to Ubuntu, Debian, and so on. The key to this is that

[1]While the term power has a wide range of meanings and implications (see Marshall, 2006), for this study, power is defined as both "The ability to influence the decisions or actions of others" (Thorelli, 1986, p 38), and the ability to determine "... which issues are considered within the decision-making agenda and which are excluded or suppressed" (Marshall, 2006, p 211).

although there are a number of different communities that contribute to the project through a wide range of means, these are not connected to the Linux project directly, and therefore the Linux foundation has no power over them. The CyanogenMod project, on the other hand, presents evidence of a hierarchical structure that controls these different communities centrally, which is led by a small group of project leaders.

To summarise, the theoretical contribution of the study is the User Organisation as a social structure with its unique form of governance, which differs itself from community governance due to its leadership, structure, and project boundaries. Leadership in a user organisation sustains a mechanism through which the project leaders are able to build project goals and to follow them, with or without the need of the peripheral contributors. In addition, the structure of the user organisation is not based on the core periphery structure of open source user communities, instead it consists of a number of smaller communities, with their own formal leader which specialise on a particular function. From the existence of the different communtiies of practice emerges the notions of boundary objects and brokers, which serve functional roles in the project's governance. Finally, the boundaries of who belongs to the project are extended beyond the software development community of practice, and takes into account all contributions that are made to the project.

11.3.2 Empirical

This study provides three empirical contributions to the study of open source user innovation communities. First, it presents an empirical study of a modern open source project which actively seeks and sets up an infrastructure in order to accommodate a number of different contributions besides software programming. Second, the CyanogenMod project presents a study of an emerging trend in open source and hacking communities, where the software is deeply integrated into the hardware, therefore raising a complexity in terms of the work and resources required for the project. Finally, it analyses in great detail the social structures and participation patterns of the different communities of practice, noting the difference of how they are managed.

The modern open source project has a much more intricate ICT infrastructure which allows users to contribute through a wider range of activities. In sharp contrast to the traditional open source projects from the 1990s, such as Apache and Linux, modern projects do not just have a repository or a mailing list, but also make use of web-based ICT that provides documentation, tutorials, and training. The ability to create such a wide range of opportunities for individuals to contribute may have been presented because of the rise of these web-based tools in the last few decades. Technologies that allow for a forum, wiki, and the Git repository system are all based on technology that emerged and then matured in the last 15 years, where people and the use of these tools has increased and people have better knowledge on how to use them. While the early internet-based open source projects did have dedicated forums, IRC, and wikis, these were not under the control of the project leaders and were usually created by other users independently.

The CyanogenMod project, therefore, represents a new generation of open source projects that established, modifies, and maintains its own ICT infrastructure. When the CyanogenMod project grew, it decided to leave the XDA forum and establish its own infrastructure, something which was done by the core team. The project then created its own infrastructure, consisting of a forum, wiki, IRC, and repository, taking donations the general users to pay for the hosting, but always retaining full control over them. In addition, the infrastructure was set up by the core team, as opposed to it being set up by a regular user, therefore giving the team full control over it.

Another emerging trend is that of hardware/software integration, which may have traditionally been more common in high technology industry but has now become popular in consumer goods with mobile devices and wearable technology. The mobile devices present a new form of computer where the software and the hardware are highly integrated, where the limitations of the hardware and their functionality are heavily reliant on the software. For instance, due to mobile devices being battery powered, both iOS and Android operating system use software in order to regulate the use of the battery and to better manage power. This, however, is hardware dependent, and required

knowledge of the hardware that is going to be used, which is why this is regularly done by the manufacturers.

In a similar way, hacking communities are using new open hardware and smaller forms of computers, called minicontrollers, to crate a wide range of devices such as 3D printers and drones in what is an emerging open source hardware movement. In recent years, open hardware projects such as Arduino and Raspberry-Pi, are becoming popular and are being used by a large community of software and hardware engineers to develop a wide range of devices. Online communities are also springing up which are dedicated to the exchange of project ideas and of software that is integrated specifically to the hardware. These projects have been used in other contexts in order to develop 3D printers, drones, and a large number of products which are hardware-specific and require special software in order to be used.

The CyanogenMod project therefore may presents an introduction into the issues in governance that may be faced by projects that are part of the open hardware movement. The same trend that is being seen in the recent open hardware movement and the same problems that are being faced by them, are ones that have been currently faced by the CyanogenMod project and others that deal with mobile devices. The traditional open source project may have required a set of specifications that will allow it to work on specific devices or architectures. Such a project is the Linux 8086 project described by Cox (1998), where the project wanted to port Linux so it could work on the 8086 architecture, which at the time was outdated. The PC architecture created a basic standard which allowed hardware components to work together with minimum changes in code. In addition, the modularity of the hardware in PCs was such that the installation of a hardware component, such as a camera, would be supported by the manufacturing company through the distribution of the relevant drivers. This is not something that comes as standard in the mobile devices, as the hardware components come pre-built into the device, therefore making it difficult to obtain the relevant drivers.

The third empirical contribution of this study is the study of the different ICT-based communities and their comparison in terms of characteristics and structure,

finding a difference in governance between them. West and Lakhani (2008) points that a significant portion of empirical studies on online user communities focus entirely on open source projects. West and Lakhani also note that there are a different number of communities on line which carry out a wide range of activities, and that even from a superficial level it is evident these may be different from each other. Finally, they state that empirical studies of the differences between these communities may show exactly how different they are.

This study therefore addresses this gap in knowledge by analysing the different types of communities which contribute through different activities, finding differences in the significance of the core periphery structure and the types of participation. First of all, we see a difference in the way that each of these communities are governed by the same groups of leaders, where the repository and the wiki are directly managed, while the interaction spaces like the forum and IRC are not. In addition, we see that the significance of the core-periphery structure differs between communities of practice, where the wiki's structure being closer to the studies by Crowston and Howison (2005). On the other hand, in the repository, we see that the core-periphery structure is not an indication of either leadership of formal roles within the project.

To summarise, this study presents an in-depth case study of the CyanogenMod project, which represents an emerging new form of open source project with multiple communities and integration between hardware and software. This new generation of open source projects does not limit contributions to programming, but also allows and is structured to receive, a wide range of contributions from general contributors. In addition, the current trend towards hardware/software integration emerging from smaller devices will also present an emerging feature, as current and future projects are headed this way. Finally, it presents empirical evidence that community governance differs from one community to another, and this difference may be shaped by the ICT through which individuals interact and contribute.

11.3.3 Methodological

This study makes two methodological contributions, first it expands on the analysis of the core periphery structure of online communities, and it also uses text analysis to determine the type of contributions being made. In the first instance, this project expands the core periphery structure by looking not just at who does what, but by also analysing the timing of the contributions and, in the case of software development, the type of contribution. Second, it adds to the current measurement and analysis of participation by identifying the different types of contributions users make to the project.

Traditionally, studies on open source user communities have focused on either one place of interaction or one place of production, using these as the main source of data. The place of interaction has traditionally looked at mailing lists, listing the individuals that are participating in the conversation and sometimes creating links in order to determine key connections and an actor's position in the community. In addition, the production place refers to the repository, where researchers look at the code that has been written, who has written it, and how much, therefore determining predominantly the amount of contributions made to the project. Taking these two sources of information it was therefore possible to see how they are then placed in the position within the community, or the community structure, and therefore the project's social structure.

Although the data used in previous studies do typically represent a significant period of time, the analysis was limited to a one dimensional snapshot of the project at the time the data was collected. Data that has been used in previous studies that look at how much users participate in conversations or how many code submission were made the repository were then added together to determine the core periphery structure of the community. The time dimension of when these contributions were made was largely lost or not taken into account, therefore presenting a very static view of the social structure of the project as it was when the data was collected. Therefore, other questions such as when did the core become the core, or when do the periphery become most active, has not been addressed.

This study addresses these questions by taking into account not just the

position of the user within the core and the periphery, but also the timing of the contributions. For example, in the repository, the data that was gathered covered not just who contributed and what was contributed, but also the date when it was submitted and added. It was therefore possible to first determine the users who were either part of the core or the periphery, and then to determine when those subgroups contributed to the repository. From this data, the results show that in general, the core contribute more at the start of the project, while the periphery become more active towards the end. This therefore adds more information that could potentially help determine and understand a practical approach towards an open life-cycle that is closer to empirical findings.

In addition, previous studies have spoken about software development as a singular activity, with very few differences between initial, experimental, or maintenance-specific activities. Traditionally, the open source studies focus on either software development itself, or on the discussion of software development through mailing lists. The software development cycle, however, contains a number of different steps, such as design, implementation, testing, and release. These steps are not taken into account when asking the question of what they are actually doing, rather than simply looking at people that contribute code it may be worthy to note what it is they do. The two notable exceptions being Midha and Bhattacherjee (2012) who are explicit in stating that it is software maintenance, and Lakhani and von Hippel (2003) who acknowledge user-to-user assistance, as opposed to code contributions.

The second methodological contribution of this study is therefore the analysis of the type of contribution being made to the repository, which has led to a better understanding of the type of activities and at which stage. The data collected from the repository contained both a title and a description of the contribution, giving the contributors the opportunity to communicate to others what type of contribution they were making. In addition, the project used a standard form of standard tags which gave a short description of the type of contribution, such as 'Translation' or 'Fix'. Through this, it was then possible to separate contributions according to three different types; 'Merge', 'Translation', and 'Fix'. It was because of this that the type of contribution was able to be linked to either

the core or the periphery, or within the sense of the subgroups that were observed in the project. This may once again be useful when trying to determine an open source project life cycle.

To summarise, this study makes two methodological contributions, first it presents a dynamic picture of the core periphery structure, and second it goes further in analysing repository contributions by kind. This study finds that the core and periphery have very distinct patterns of contributions, where in some cases we see the core contribute more in the early stages of the project and the periphery being more active towards the end. In addition, the categorisation of the contributions into four categories allowed there to be a better distinction of what was being done and when, finding that fixes to the code occur predominantly towards the end of the observed period and are carried out by the periphery.

11.4 Limitations

The main limitation of this study, as with other single-case open source studies is that of generalisability of the findings. As mentioned in the literature review, the use of established projects means that the assumption is that the resulting governance of such communities is what has led it to success. This is tied with the two approaches to what governance is, in terms of the system which enables the sustained development of the software and the increase in contributions. Studies looking at traditional open source projects like Apache and Linux are, therefore, still seen as key empirical cases that inform the literature on user community innovation.

The generalisation of the findings must be questioned when we take into account the possibility that each of the projects has evolved in a different way, its socio-technical structure is different, and will be faced with a wide range of different external issues. From the socio-technical perspective, a wide range of studies, most notably Baldwin and Clark (2006) and Amrit and Van Hillegersberg (2010), have explored the connection between the technical structure of the software and the impact this has on the social structure and on participation. Baldwin and Clark (2006) found that modules create a small

working team within a project, with it's own module lead maintainer and therefore structure. Amrit and Van Hillegersberg (2010), on the other hand, state that in order for the cost of coordination to be reduced, the social structure must, and often does, match the technical structure. The implication of this is that the organisational structure found in the CyanogenMod project may have emerged as a result of internal and external elements, therefore making it unique.

Besides the uniqueness in structure, from the contingency perspective, each project will also have its own evolutionary path which may be a result of internal and external events. The primary work that looks at key events as factors that influence governance emergence is O'Mahony and Ferraro (2007), who focused on key internal events within the Debian project in order to determine the reason why governance structure were chosen. Similarly, Cox (1998) also present a very particular case of the event of isolating the core from the periphery, which was particular to their specific project and led to their project's failure. These issues, however, were present in their particular projects that were studied, and are not the same issues that affect all projects in general.

The findings presented in this study, therefore, may be unique to the CyanogenMod project, where issues of leadership, size, and control may have been confronted and tackled differently than in other projects. One of the key events therefore can be said to be the fact that the leadership of the project has remained entrenched with the project owner, and most importantly, the project owner has remained part of the project for so long. This mode of leadership, or deflection of leadership can be seen with other projects like Linux, which sees the leadership and vision aspect of the project founder being an influential figure, but not the dominant one in the project. Therefore, CyanogenMod presents a key difference in the owner's ability and desire to remain part of the project and to lead it.

Despite there being numerous reasons as to why and how projects evolve differently, one event that seems to affect all the project, including CyanogenMod, was the growth in size of the community. The issue of size is something that a number of projects have explicitly have to dealt with, as was the case of the

Debian project as studied by O'Mahony and Ferraro (2007), who stated that the size of the community brought about a need to change governance. This same issue was seen in the CyanogenMod project, where the size of the community and the number of contributions that were submitted prompted the project leaders to set up a dedicated infrastructure and to designate project roles between them.

The issue of size, however, may not be as important as some people believe, as there may be reasons why an open source software does not require a large pool of contributors in order to be successful. So far, the assumption has been that the success of an open source project is evaluated in terms of the number of individuals that contribute to the project, where a higher number means the project is successful. It is this assumption that also guides the view of the function of governance, which is to allow people to contribute and to create an environment that is better for people to contribute. The reason why this is so can be traced to Linus' law, where the larger number of individuals participating in the maintenance of the software ensures that it will be improved constantly. Some of these studies, however, have been based on large software projects such as Linux, whereas smaller software projects may not require a large group of contributors and the governance structures that follow.

In the same way that the technical structure of the software affects its social structure, so can the size of the software code have an effect on the number of programmers contributing to it. This therefore leads to the possibility that smaller projects, in terms of lines of code, may be more likely to have a smaller number of users contributing to it, regardless of the project's governance. The number of contributors, therefore, should not be an indication of the project's success without first taking into account the characteristics of the software. Other aspects of a project's impact, particularly usage and integration with other projects, could serve as a better indication of success.

Other methodological limitations include the interviews that may have been given by the respondents. These interviews, although providing valuable information otherwise not accessible, may be one sided and may not provide an accurate account of events. This is because the respondents may have an idealised account of their contributions and involvement in the project,

therefore possibly presented a subjective view of their role.

Finally, while this study focused on the project's governance from the perspective of the project leaders, it does not offer a view from the general contributors. The limitation of the study is therefore that the governance mechanisms used may be idealised by the project leaders who may assume that they are working effectively. Speaking to the general contributors, on the other hand, may have offered a different perspective on the impact the governance mechanisms have on motivation.

To summaries, one of the limitations of this project is on the generalisability, looking at whether the organisational structure present is representative of other open source projects. The open source literature has acknowledged the emergent way in which both structure and governance have evolved in different software projects, noting these as being similar to contingency theory. This is consistent with Morgan's (1986, p 49) observation that "The appropriate form depends on the kind of task or environment with which one is dealing with". The only common ground that could be generalised, however, is the fact that size of the community, in terms of contributors, does play an important part in the evolution of project governance. Community size, however, may only be an issue in open source projects where the software characteristics, such as complexity or lines of codes, introduces the need for a change in governance and social structure. The resulting structure and governance system, however, will depend on the decisions of the community, in a democratic community, or of the leader, in an autocratic community. The objective of this study, however, is to produce an account of how the CyanogenMod project deals with the multiple forms of contributions from the various communities of practice active within the project.

11.5 Future Research

This study was a single in-depth case study of a user organisation, as mentioned previously, this open source project may very well be unique and unrepresentative of modern projects. The CyanogenMod project as a single case study has been helpful in generating the theoretical proposition of the user

organisation, providing what Yin (1989) describes as analytic generalisation. Future studies could therefore study a larger number of open source projects that present similar characteristics to the user organisation governance model in order to obtain statistical generalisability. Doing so, it may be possible to determine whether or not there are key factors that affect all projects equally, such as size, and whether these factors may lead to a specific type of governance model. In addition, other projects and a comparative case study could also incorporate the idea proposed by O'Mahony and Ferraro (2007), by looking at significant events and the factors that led to a change in governance.

Future research should also explore the full benefits firms have when engaging with user communities. Much of the current literature places emphasis on knowledge, efficiency, and speed of development as the key benefits of user communities. This study showed that users can also add value to the project by providing supporting skills and services, such as translation and documentation. Future studies could therefore explore the wide range of benefits firms can have, such as accessing complimentary skills and resources, as well as determining how best to manage them, specifically within the literature on hybrid communities.

Additionally, the User Organisation also poses new questions regarding the effect social structure has on collaborative innovation. In the literature, terms such as 'crowd', 'community', and 'network' are used to describe social groups that are involved in user innovation. These terms, however, are often not clearly defined and may sometimes be used interchangeably. Future studies should look to study the significance of these types of groups in terms of the relationship between individuals, the social roles within them, and the effect it has on each stage of the innovation process. These studies could further help create a framework for strategically creating the right social group for specific innovation tasks.

Future research could also focus or mention the distinction between contributions through the creation of new methodologies that can determine what type of contributions are made and when. This study explored four different types of contributions that were made to the repository through

semantics analysis which determined what type of contribution was being made. Future research could incorporate the same techniques used in bibliometric studies where subject and classification of academic papers is done through complex text analysis. Incorporating complex text analysis methods it may then be possible to determine patterns of the types of contributions being made, which can in turn lead to the initial view of a project life cycle.

The CyanogenMod project, for instance, had the core ROM and the added and separate ports for each of the devices, and it was also observed that there were different groups that were in charge of the different aspects of the project. This is something that was equally seen in other projects, particularly in Baldwin and Clark (2006) where there was a core part of the software highly controlled by the project leaders, while the modules were largely the domain of the general contributors. The notion of the modularity of the software as discussed by Baldwin and Clark focused on the way in which it allowed individuals to be able to contribute to the project with a low coordination cost. The effects of the way in which this structure can be used in order to retain project control while at the same time allowing other contributors the freedom to implement solutions that they would like to see.

Future studies should therefore look at how the structure of the software can be used in order to retain overall control of the project while at the same time giving general contributors the freedom to add to the code base as they see fit. In the CyanogenMod project, there was a clear division and freedom given to the general contributors which meant that, while the core team retain control over the base ROM, the general contributors still had the ability to change and modify the ROM. This was then reinforced by the notion of the official and non-official ports, where general contributors were given the space to experiment and implement non-canonical modifications without rules or restrictions. This may be one of the reasons why contributions were still engaged in the project despite the close control by the core team, which is one of the dimensions that can effect motivation and participation (Shah (2006)).

Added control over the base code, and the provision of a separate experimental area, for both the core team and general contributors, could then potentially lead

to the question about the formulation and the implementation of strategy in open source. Previously, the general idea of the community-based model of software development was that, because of the lack of formal authority and the volatility of the contributors, it would not be possible to implement strategy. One of the key findings of this study is that the governance of the CyanogenMod project allowed the project owners to create and to fulfil specific goals due to the control they had over the base of the project, the ROM. This control allowed the core team to implement features that were chosen in order to fulfill specific project goals which were determined by the core team.

To summarise, future studies should focus on extending the validity of the findings in terms of the user organisation as well as expanding the methods used and incorporating strategy into the analysis. Future studies can further validate the findings in this study by finding and analysing other empirical examples of projects with the same form of governance. The use of more complex text analysis and taking into account the structure of the software, it may also be able to expand our knowledge in terms of the possible life cycle stages in open source development as well as the control over the software. Finally, when looking at control over the software, the notion of strategy in terms of design and project control should also be expanded, to determine the different ways in which it can be achieved without affecting participation.

Epilogue

In late 2013 Steve Kondik opened negotiations to raise venture capital for the CyanogenMod project, initiating the path towards its commercialisation. Final negotiations with Benchmark Capital and Redpoint Ventures were closed in April 2014, leading to an initial investment of approximately 23 million USD. Following this, Steve Kondik set up Cyanogen Inc., a company that would deal specifically with the commercial aspect of the operating system. Soon after, two offices were set up, one in Seattle and the other in Palo Alto, employing the core team members that were part of the project from the start. In the months that followed, device manufacturer Oppo started shipping their flagship device, PlusOne, with CyanogenMod pre installed, finally fulfilling Steve Kondik's objective.

The development of the operating system was divided into two streams; CyanogenMod remained as the community-maintained version, while the new CyanogenOS became its commercial counterpart. The new operating system is partially open, containing all the modifications made to the community version while incorporating a number of proprietary apps and services. The commercialisation of the project led to a negative reaction from the community, who expressed their fears regarding the appropriation of users' contributions. Cyanogen Inc has addressed this by updating the rules of contribution and the project's documentation to emphasise the software licenses used.

Since the early days of Cyanogen Inc, much has changed in both the company and in the smartphone modification movement. In the first few months of 2015, Cyanogen Inc went through a number of changes in its personnel, with all of the founding members being replaced, leaving Steve Kondik as the only member from the XDA forum days. In addition, the communities revolving around ROM modifications are in a transitory state, as

the technical issues the modified ROMs were solving are now being addressed by manufacturers and other user-led projects. With Motorola leading the way, Samsung and HTC have now began to release simpler versions of Android, with less customised apps and services. Most significantly, the user-led Xposed project (another child of the XDA Forum) can provide the same level of customisation for Android users without having to go through the extensive and sometimes complex process of installing a new operating system. The key advantage of Xposed is that users are able to install it on their current operating system, therefore removing the need to port software to new devices.

This study remains an account of how a successful open source project managed its wide range of contributions in order to provide multiple products and services to its users. The leadership team of the project were able to turn Cyanogen into a commercial company, extracting the structures and governance from its non-commercial days and using these as a basis for their new company. The style of governance and structure described in this study, therefore, allowed the project owner to set and meet the objectives and commercial ambitions of the project.

Bibliography

Adams, P., R. Fontana, and F. Malerba (2012). User knowledge in innovation in high technologies: An empirical analysis of semiconductors. *International Journal Of Technology Management 58*(3-4), 284 – 299.

Alahuhta, P., P. Abrahamsson, and A. Nummiaho (2008). On exploring consumers technology foresight capabilities an analysisof 4,000 mobile service ideas. *Ice-B 2008: Proceedings Of The International Conference On E-Business None*(None), 169 – 176.

Amadeo, R. (2013a). Cyanogenmod goes pro with cyanogen inc. and $7 million in funding. http://goo.gl/zP8iCN [4 Dec 2013].

Amadeo, R. (2013b). Google's iron grip on android: Controlling open source by any means necessary. http://goo.gl/uTSbpL [Accessed Dec 2013].

Amrit, C. and J. Van Hillegersberg (2010). Exploring the impact of socio-technical core-periphery structures in open source software development. *Journal of Information Technology 25*(2), 216–229.

An-Hua, G. and T. Peng (2009). The approach for enterprises to realize technological innovation. *2009 International Conference On Information Management, Innovation Management and Industrial Engineering, Vol 3, Proceedings None*(None), 499 – 502.

Andersen, P. H. (2012). Participation in innovation communities: Strategies and contingencies. In *Collaborative Communities of Firms*, pp. 59–73. Springer.

Androidandme.com (2009). Interview with android hacker jesusfreke. http://androidandme.com/2009/02/news/interview-with-android-hacker-jesusfreke/ [accessed 28 Jan 2014].

Android.com (2013a). The android source code; governance philosophy. http://source.android.com/source/index.html [accessed 28 Jan 2014].

Android.com (2013b). Licenses. http://source.android.com/source/licenses.html [accessed 8 March 2014].

Aoki, K. (2009). Free seeds, not free beer: Participatory plant breeding, open sourceseeds, and acknowledging user innovation in agriculture. *Fordham Law Review 77*(5), 2275 – 2310.

Arthur, C. and S. Gibbs (2014). Why google android software is not as free or open source as you may think. http://gu.com/p/3m5yd/tw [Accessed 24 Jan 2014].

Bai, H., S. Yang, and J. Zhong (2006). The new role of service for smes in machinery manufacturing industries:a case study from china. *2006 International Conference On Service Systems and Service Management, Vols 1 and 2, Proceedings None*(None), 663 – 667.

Baldwin, C. Y. and K. B. Clark (2006). The architecture of participation: Does code architecture mitigate free riding in the open source development model?. *Management Science 52*(7), 1116 – 1127.

Baldwin, C. Y., C. Hienerth, and E. von Hippel (2006). How user innovations become commercial products: A theoretical investigation and case study. *Research Policy 35*(9), 1291 – 1313.

Balka, K., C. Raasch, and C. Herstatt (2014). The effect of selective openness on value creation in user innovation communities. *Journal of Product Innovation Management 31*(2), 392–407.

Bampton, R. and C. J. Cowton (2002). The e-interview. *Forum: Qualitative Social Research 3*(2).

Beavis, G. (2008). A complete history of android. http://www.techradar.com/news/phone-and-communications/mobile-phones/a-complete-history-of-android-470327[accessed February 2005].

Beck, L. (1985). *System Software: An Introduction to System Programming*. Addison-Wesley Publishing Company.

Becker, M. H. (1970). Sociometric location and innovativeness: Reformulation and extension of the diffusion model. *American Sociological Review 35*(2), pp. 267–282.

Bell, C. and H. Newby (1971). *Community Studies: An Introduction to the Sociology of the Local Community*. London: Geroge Allen & Unwin.

Biernacki, P. and D. Waldorf (1981). Snowball sampling: Problems and techniques of chain referral sampling. *Sociological methods & research 10*(2), 141–163.

Bird, S., E. Loper, and E. Klein (2009). *Natural Language Processing with Python*. O'Reilly Media Inc.

Black, P. E. (2004). Bradford's law. in Dictionary of Algorithms and Data Structures [online] http://www.nist.gov/dads/HTML/bradfordsLaw.html [Accessed 05-02-2013].

Bogers, M., A. Afuah, and B. Bastian (2010). Users as innovators: A review, critique, and future research directions. *Journal of Management 36*(4), 857–875.

Bogers, M. and J. West (2012). Managing distributed innovation: Strategic utilization of open and user innovation. *Creativity and Innovation Management 21*(1), 61–75.

Bohn, D. (2011). Samsung packs a ton of new features into 'touchwiz nature ux' on android 4.0 for the galaxy s iii. theVerge.com – 17/Aug/2012. `http://goo.gl/amvHF`.

Bonaccorsi, A. and C. Rossi (2006). Comparing motivations for individual programmers and firms to take part in the open source movement: From community to business. *Knowledge, Technology, & Policy 18*(4), 40 – 64.

Bornstein, D. (2008). Google i/o 2008 - dalvik virtual machine internals. [Youtube Video] - `http://youtu.be/ptjedOZEXPM`.

Bornstein, D. (2009). Dalvik overview and q&a. [Youtube Video] - `http://youtu.be/RF62c4HgbU4`.

Bradford, S. (1985). Sources of information on specific subjects. *Journal of information Science 10*(4), 173–180.

Braun, V. and C. Herstatt (2006). Barriers to userinnovation: The paradigm of permission to innovate. *2006 Ieee International Conference On Management Of Innovation and Technology, Vols 1 and 2, Proceedings None*(None), 176 – 180.

Bretthauer, D. (2002). Open source software: A history. *Information Technology and Libraries 21*(1), 3–10.

Burt, R. S. (2004). Structural holes and good ideas. *American Journal of Sociology 110*(2), 349 – 399.

Capra, E., C. Francalanci, and F. Merlo (2008). An empirical study on the relationship between software design quality, development effort and governance in open source projects. *Software Engineering, IEEE Transactions on 34*(6), 765 –782.

Carillo, K. and C. Okoli (2008). The open source movement: A revolution in software development. *Journal of Computer Information Systems 49*(2), 1 – 9.

Casadesus-Masanell, R. and G. Llanes (2011). Mixed source. *Management Science 57*(7), 1212 – 1230.

Cattani, G. and S. Ferriani (2008). A core/periphery perspective on individual creative performance: Social nework and cinematic achievements in the hollywood film industry. *Organization Science 19*(6), 824 – 844.

Chandra, Y. and M. A. A. M. Leenders (2012). User innovation and entrepreneurship in the virtual world: A study of second life residents. *Technovation 32*(7-8), 464 – 476.

Chatterji, A. K. and K. Fabrizio (2012). How do product users influence corporate invention? *Organization Science 23*(4), 971–987.

Chesbrough, H. W. (2006). Open innovation: A new paradigm for understanding industrial innovation. In *Open Innovation: Researching a New Paradigm*. Oxford: Oxford University Press.

Christley, S. and G. Madey (2007). Analysis of activity in the open source software development community. In *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on*, pp. 166b–166b. IEEE.

Chu, K.-M. and H.-C. Chan (2009). Community based innovation: Its antecedents and its impact on innovation success. *Internet Research 19*(5), 496 – 516.

Cleron, M. (2007). Androidology - architecture overview. [Youtube Video] - `http://youtu.be/QBGfUs9mQYY`.

Cohen, A. P. (1985). *The Symbolic Construction of Community*. London: Routledge.

Coutts, R., P. Coutts, and K. Alport (2005). Understanding the user within the innovation spiral. *Mobile Information Systems Ii 191*(None), 47 – 62.

Cox, A. (1998). Cathedrals, bazaars and the town council.

Creswell, J. W. (2003). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches* (2 ed.). London: SAGE Publications.

Crowston, K. and J. Howison (2005). The social structure of free and open source software development. *First Monday 10*.

Crowston, K. and J. Howison (2006). Hierarchy and centralization in free and open source software team communications. *Philosophy & Technology 18*(4), 65 – 85.

Crowston, K. and B. Scozzi (2002). Open source software projects as virtual organisations: Competency rallying for software development. *Software, IEE Proceedings - 149*(1), 3 – 17.

Crowston, K., K. Wei, J. Howison, and A. Wiggins (2008). Free/libre open-source software development: What we know and what we do not know. *ACM Comput. Surv. 44*(2), 7:1–7:35.

Crowston, K., K. Wei, J. Howison, and A. Wiggins (2012). Free/libre open-source software development: What we know and what we do not know. *ACM Computing Surveys (CSUR) 44*(2), 7.

Crowston, K., K. Wei, Q. Li, U. Y. Eseryel, and J. Howison (2005). Coordination of free/libre and open source software development.

Crowston, K., K. Wei, Q. Li, and J. Howison (2006). Core and periphery in free / libre and open source software team communications. *Proceedings of the 39th Hawaii International Conference on System Sciences* (1 - 7).

CyanogenMod (2014). Introducing: Gallerynext. https://plus.google.com/+CyanogenMod/posts/TwPYXCxEz2H [accessed Jan 2014].

Dahan, E. and J. R. Hauser (2002). The virtual consumer. *The Journal of Production Innovation Management 19*, 332 – 353.

Dahlander, L. and L. Frederiksen (2011). Core and cosmopolitans: A relational view of innovation in user communities. *Organization Science Articles in Advance*, 1 – 20.

Dahlander, L. and M. Magnusson (2008). How do firms make use of open source communities? *Long Range Planning 41*(6), 629 – 649.

Dahlander, L. and S. O'Mahony (2011). Progressing to the center: Coordinating project work. *Organization Science 22*(4), 961 – 979.

Dahlander, L. and M. W. Wallin (2006). A man on the inside: Unlocking communities as complementary assets. *Research Policy 35*, 1243 – 1259.

De Couvreur, L. and R. Goossens (2011). Design for (every)one: Co-creation as a bridge between universal design and rehabilitation engineering. *Codesign-International Journal Of Cocreation In Design and The Arts 7*(2), 107 – 121.

De Jong, J. P. J. and E. von Hippel (2009). Transfers of user process innovations to process equipment producers: A study of dutch hightech firms. *Research Policy 38*(7), 1181 – 1191.

de Laat, P. B. (2007). Governance of open source software: State of the art. *Journal of Management & Governance 11*(2), 165–177.

De Laat, P. B. (2007). Introduction to a roundtable on the governance of open source software: particular solutions and general lessons. *Journal of Management and Governance 11*(2), 115–117.

de Vaus, D. (2001). *Research Design Is Social Research*. SAGE Publications.

DeLamarter, R. T. (1986). *Big Blue: IBM's use and abuse of power*. Dodd, Mead & Company.

Demil, B. and X. Lecocq (2006). Neither market nor hierarchy nor network: The emergence of bazaar governance. *Organization Studies 27*(10), 1447–1466.

Deodhar, S. J., K. Saxena, R. K. Gupta, and M. Ruohonen (2012). Strategies for software-based hybrid business models. *The Journal of Strategic Information Systems 21*(4), 274 – 294.

Devkota, A. (2013). Oppo partners with cyanogen inc. http://goo.gl/uyIRvR [4 Dec 2013].

Di Gangi, P. M. and M. Wasko (2009). Steal my idea! organizational adoption of user innovations from a user innovation community: A case study of dell ideastorm. *Decision Support Systems 48*(1), 303–312.

Di Maria, E. and V. Finotto (2008). Communities of consumption and made in italy. *Industry and Innovation 15*(2), 179 – 197.

DiBona, C. and S. Ockman (1999). *Open sources: Voices from the open source revolution*. O'Reilly Media, Inc.

Douthwaite, B., J. Keatinge, and J. Park (2001). Why promising technologies fail: The neglected role of user innovationduring adoption. *Research Policy 30*(5), 819 – 836.

Durugbo, C. and K. Pawar (2014). A unified model of the co-creation process. *Expert Systems with Applications 41*(9), 4373–4387.

Eisenberg, I. (2011). Lead-user research for breakthrough innovation. *Research-Technology Management 54*(1), 50 – 58.

Faulkner, P. and J. Runde (2009). On the identity of technological objects and user innovations infunction. *Academy Of Management Review 34*(3), 442 – 462.

Feller, J. and B. Fitzgerald (2000). A framework analysis of the open source software development paradigm. In *Proceedings of the twenty first international conference on Information systems*, ICIS '00, Atlanta, GA, USA, pp. 58–69. Association for Information Systems.

Fiedler, F. E. (1971). Validation and extension of the contingency model of leadership effectiveness: A review of empirical findings. *Psychological bulletin 76*(2), 128.

Fielding, R. T. (1999). Shared leadership in the apache project. *Commun. ACM 42*(4), 42–43.

Fingas, J. (2013). Focal camera app removed from cyanogenmod, launched as standalone beta. http://www.engadget.com/2013/09/23/focal-camera-app-launches-as-standalone-beta/ [accessed March 2014].

Fisher III, W. W. (2010). The implications for law of user innovation. *Minnesota Law Review 94*(5), 1417 – 1477.

Franck, E. and C. Jungwirth (2002). Reconciling investors and donators - the governance structure of open source.

Franck, E. and C. Jungwirth (2003). Reconciling rent-seekers and donators–the governance structure of open source. *Journal of Management and Governance 7*(4), 401–421.

Franke, N., P. Keinz, and K. Klausberger (2013). Does this sound like a fair deal?: Antecedents and consequences offairness expectations in the individuals decision to participate infirm innovation. *Organization Science 24*(5), 1495 – 1516.

Franke, N., P. Keinz, and C. J. Steger (2009). Testing the value of customization: When do customers really prefer products tailored to their preferences? *Journal Of Marketing 73*(5), 103 – 121.

Franke, N. and F. Piller (2003). Key research issues in user interaction with user toolkits in a masscustomisation system. *International Journal Of Technology Management 26*(5-6), 578 – 599.

Franke, N. and F. Piller (2004). Value creation by toolkits for user innovation and design: The case of the watch market. *Journal Of Product Innovation Management 21*(6), 401 – 415.

Franke, N., M. Schreier, and U. Kaiser (2010). The i designed it myself effect in mass customization. *Management Science 56*(1), 125 – 140.

Franke, N. and S. Shah (2003). How communities support innovative activities: An exploration of assistance and sharing among end-users. *Research Policy 32*(1), 157 – 178.

Franke, N. and E. von Hippel (2003). Satisfying heterogeneous user needs via innovation toolkits. *Research Policy 32*(7), 1199 – 1215.

Franke, N., E. von Hippel, and M. Schreier (2006). Finding commercially attractive user innovations: A test of lead-user theory. *Journal of Product Innovation Management 23*, 301 – 315.

Fredberg, T. and F. T. Piller (2011). The paradox of tie strength in customer relationships for innovation: A longitudinal case study in the sports industry. *R & D Management 41*(5), 470 – 484.

Fuller, J. (2006). Why consumers engage in virtual new product developments initiated by producers. *Advances in Consumer Research 33*, 639 – 646.

Fuller, J. (2010). Refining virtual co-creation from a consumer perspective. *California Management Review 52*(2), 98 – 122.

Füller, J., R. Schroll, and E. von Hippel (2013). User generated brands and their contribution to the diffusion of user innovations. *Research Policy*.

Glass, R. L. (2003). A sociopolitical look at open source. *Commun. ACM 46*(11), 21–23.

Google (2011). Google apps: Terms of service. https://www.google.com/apps/intl/en/terms/user_terms.html [accessed March 2014].

Granovetter, M. S. (1973). The strength of weak ties. *American Journal of Sociology 78*(6), 1360 – 1380.

Greiner, L. E. (1972). Evolution and revolution as organizations grow.

Gruver, B. (2009). Calling it quits. http://goo.gl/B8LCr6 [3 Dec 2013].

Hargadon, A. B. (2005). Bridging old worlds and building new ones: Towards a microsociology of creativity. In L. Thompson and H. Choi (Eds.), *Creativity and Innovation in Organisational Teams*, pp. 199 – 216. Lawrence Erlbaum Assosiates.

Harhoff, D., J. Henkel, and E. Von Hippel (2003). Profiting from voluntary information spillovers: how users benefit by freely revealing their innovations. *Research policy 32*(10), 1753–1769.

Harrison, D. and A. Waluszewski (2008). The development of a user network as a way to relaunch an unwantedproduct. *Research Policy 37*(1), 115 – 130.

Hau, Y. S. and Y.-G. Kim (2011). Why would online garners share their innovation conducive knowledge inthe online game user community? integrating individual motivations and social capital perspectives. *Computers In Human Behavior 27*(2), 956 – 970.

Heiskanen, E. and R. Lovio (2010). User- producer interaction in housing energy innovations. *Journal of Industrial Ecology 14*(1), 91–102.

Helminen, P. and J. Ainoa (2009). User innovation toolkits in product development: Qualitative study in shopping center design. *Iced 09 - The 17Th International Conference On Engineering Design, Vol 5: Design Methods and Tools, Pt 1 None*(None), 265 – 273.

Hemetsberger, A. and C. Reinhardt (2009). Collective development in open-source communities: An activity theoretical perspective on successful online collaboration. *Organization studies 30*(9), 987–1008.

Herstatt, C. and E. von Hippel (1992). From experience: Developing new product concepts via the lead user method: A case study in a "low tech" field. *Journal of Product Innovation Management 9*, 213 – 221.

Hienerth, C. (2006). The commercialization of user innovations: The development of the rodeokayak industry. *R & D Management 36*(3), 273 – 294.

Hienerth, C., E. Von Hippel, and M. B. Jensen (2014). User community vs producer innovation development efficiency: A first empirical study. *Research Policy 43*, 190 – 201.

Hiltzik, M. A. et al. (1999). *Dealers of lightning: Xerox PARC and the dawn of the computer age*. HarperCollins Publishers.

Ho, V. (2013). Xiaomi beats samsung to top china's smartphone charts. http://techcrunch.com/2013/08/12/xiaomi-beats-samsung-to-top-chinas-smartphone-charts/ [accessed 10/Apr/2014].

Horwitz, J. (2014). Where do xiaomi's 30 million miui users live, and what do they do on their phones? http://www.techinasia.com/xiaomi-30-million-miui-users-in-58-countries-infographic/ [accessed 10/Apr/2014].

Hossain, L. and D. Zhu (2009). Social networks and coordination performance of distributed software development teams. *The Journal of High Technology Management Research 20*(1), 52 – 61.

Hung, C.-L., J. C.-L. Chou, and T.-P. Dong (2011). Innovations and communication through innovative users: An exploratory mechanism of social networking website. *International Journal Of Information Management 31*(4), 317 – 326.

Hyysalo, S. (2009). User innovation and everyday practices: Micro-innovation in sports industry development. *R & D Management 39*(3), 247 – 258.

Iannacci, F. (2005). Coordination processes in open source software development: The linux case study.

James, W. (1955). Pragmatism and four essays form the meaning of truth.

Jeffries, A. (2014). Cyanogenmod rolls out encrypted text messaging by default. http://www.theverge.com/2013/12/9/5191778/cyanogenmod-rolls-out-encrypted-text-messaging-by-default-whisper-systems [accessed 31 Jan 2014].

Jeppesen, L. B. and L. Frederiksen (2006). Why do users contribute to firm-hosted user communities? the case of computer-controlled music instruments. *Organization Science 17*(1), 45 – 63.

Jeppesen, L. B. and K. R. Lakhani (2010). Marginality and problem-solving effectiveness in broadcast search. *Organization Science 21*(5), 1016 – 1033.

Kadav, A. and M. Swift (2012). Understanding modern device drivers. *ASPLOS'12 3*(7), 1 – 12.

Kaiser, S. and G. Muller-Seitz (2008). Leveraging lead user knowledge in software development-the case of weblog technology. *Industry and Innovation 15*(2), 199 – 221.

Kamei, Y., S. Matsumoto, H. Maeshima, Y. Onishi, M. Ohira, and K. Matsumoto (2008). Analysis of coordination between developers and users in the apache community. *Open Source Development, Communities and Quality*, 81–92.

Kastrenakes, J. (2013). Sense 5.5 update for the htc one lets you finally kill blinkfeed. http://www.theverge.com/2013/10/24/5023820/htc-one-sense-5-5-update-rolling-out-android-4-3-disable-blinkfeed [accessed 31st Jan 2014].

Katz, M. L. and C. Shapiro (1985). Network externalities, competition, and compatibility. *The American Economic Review 75*(3), 424 – 440.

Kilamo, T., I. Hammouda, T. Mikkonen, and T. Aaltonen (2012). From proprietary to open source – growing an open source ecosystem. *Journal of Systems and Software 85*(7), 1467 – 1478.

Koch, S. (2009). Exploring the effects of sourceforge. net coordination and communication tools on the efficiency of open source projects using data envelopment analysis. *Empirical Software Engineering 14*(4), 397–417.

Kondik, S. (2013). A new chapter. http://www.cyanogenmod.org/blog/a_new_chapter.

Krishnamurthy, S. (2002). Cave or community?: An empirical examination of 100 mature open source projects. *First Monday*.

Kuk, G. (2000). "when to speak again": Self-regulation under facilitation. *Group Dynamics: Theory, Research, and Practice 4*(4), 291–306.

Kuk, G. (2006). Strategic interaction and knowledge sharing in the kde developer mailing list. *Management Science 52*(7), 1031 – 1042.

Lakhani, K. R. (2006). *The Core and the Periphery in Distributed and Self-Organizing Innovation Systems*. Ph. D. thesis, Massachusetts Institute of Technology.

Lakhani, K. R. and E. von Hippel (2003). How open source software works: Free user to user assistance. *Research Policy 32*, 923 – 943.

Larman, C. and V. R. Basili (2003). Iterative and incremental development: A brief history. *Computer 36*(6), 47–56.

Lattemann, C. and S. Stieglitz (2005). Framework for governance in open source communities. In *System Sciences, 2005. HICSS '05. Proceedings of the 38th Annual Hawaii International Conference on*, pp. 192a.

Lave, J. and E. Wenger (1991). *Situated learning: Legitimate peripheral participation*. Cambridge university press.

Lee, G. K. and R. E. Cole (2000). The linux kernel development as a model of open source knowledge creation. *Unplublished, Haas Scool of Business, University of California, Berkley*, 1 – 51.

Lee, G. K. and R. E. Cole (2003). From a firm-based to a community based model of knowledge creation: The case of the linux kernel development. *Organization Science 14*(6), 633 – 649.

Lehmann, F. (2004). Floss developers as a social formation. *First Monday 9*(11).

Leonard, A. (2000). Bsd unix: Power to the people, from the code. http://www.salon.com/2000/05/16/chapter_2_part_one/[accessed February 2015].

Levy, S. (2001). *Hackers: Heroes of the Computer Revolution*, Volume 4. Penguin Books New York.

Lin, Y. (2005). The future of sociology of floss. *First Monday Special Issue #2: Open Source*, 1 – 5.

Lüthje, C. (2004). Characteristics of innovating users in a consumer goods field anempirical study of sportrelated product consumers. *Technovation 24*(9), 683 – 695.

Lüthje, C. and C. Herstatt (2004). The lead user method: An outline of empirical findings and issues for future research. *R&D Management 34*(5), 553–568.

Lüthje, C., C. Herstatt, and E. von Hippel (2002). The dominant role of local information in user innovation: The case of mountain biking. *MIT Sloan Working Paper*.

Lüthje, C., C. Herstatt, and E. von Hippel (2005). User-innovators and local information: The case of mountain biking. *Research Policy 34*(6), 951 – 965.

Markus, M. L. (2007). The governance of free open source software projects: Monolithic, multidimensional, or configurational? *Journal of Management & Governance 11*(2), 151–163.

Marshall, N. (2006). Understanding power in project settings. *Making projects critical*, 207–231.

Martínez-Torres, M. (2012). A genetic search of patterns of behaviour in oss communities. *Expert Systems with Applications 39*(18), 13182–13192.

Martinez-Torres, M. R. and M. C. Diaz-Fernandez (2014). Current issues and research trends on open source software communities. *Technology Analysis & Strategic Management 26*(1), 55 – 68.

Midha, V. and A. Bhattacherjee (2012). Governance practices and software maintenance: A study of open source projects. *Decision Support Systems 54*(1), 23–32.

Mockus, A., R. T. Fielding, and J. D. Herbsleb (2002). Two case studies of open source software development: Apache and mozilla. *ACM Trans. Softw. Eng. Methodol. 11*, 309–346.

Monteiro, E., T. Osterlie, K. Rolland, and E. Royrvik (2004). Keeping it going: Everyday practices of open source software. *Norwegian Univ. of Science and Technology Unpublished*.

Moon, J. Y. and L. Sproull (2002). Essence of distributed work: The case of the linux kernel. *First Monday 5*(11), 1 – 21.

Morgan, G. (1986). *Images of Organisations*. SAGE Publications.

Morrison, J. (2013). Clahub: Easy contributor agreements on github. http://jayunit.net/2013/01/09/clahub-easy-contributor-agreements-on-github/.

Morrison, P. D., J. H. Roberts, and E. Von Hippel (2000). Determinants of user innovation and innovation sharing in a local market. *Management science 46*(12), 1513–1527.

Nakakoji, K., Y. Yamamoto, Y. Nishinaka, K. Kishida, and Y. Ye (2002). Evolution patterns of open-source software systems and communities. *Proceedings of the International Workshop on Principles of Software Evolution.*, 1 – 10.

Neil, A. (2012). Apollo no longer in the play store. https://plus.google.com/115877233289242058609/posts/CWYYeJH6xLc [accessed Jan 2014].

Newton, C. (2013). Fork in the road: Cyanogen raises $7 million to build a better version of android. http://goo.gl/14IH4S [4 Dec 2013].

Niederman, F., A. Davis, M. E. Greiner, D. Wynn, and P. T. York (2006). A research agenda for studying open source i: A multi-level framework. *Communications of the Association for Information Systems 18*(1), 7.

Nishikawa, H., M. Schreier, and S. Ogawa (2013). User-generated versus designer-generated products: A performance assessment at muji. *International Journal of Research in Marketing 30*(2), 160 – 167.

Nov, O. and G. Kuk (2008). Open source content contributors' response to free-riding: The effect of personality and context. *Computers in Human Behavior 24*(6), 2848 – 2861. Including the Special Issue: Electronic Games and Personalized eLearning Processes.

Ogawa, S. (1998). Does sticky information affect the locus of innovation? evidence from the japanese convenience-store industry. *Research Policy 26*(7), 777–790.

O'Mahony, S. (2003). Guarding the commons: How community managed software projects protect their work. *Research Policy 32*(7), 1179 – 1198.

O'Mahony, S. (2007). The governance of open source initiatives: What does it mean to be community managed? *Journal of Management & Governance 11*(2), 139–150.

O'Mahony, S. and F. Ferraro (2007). The emergence of governance in an open source community. *Academy of Management Journal 50*(5), 1079 – 1106.

OpenSignal (2014). Android fragmentation visualized. http://opensignal.com/reports/2014/android-fragmentation/[accessed February 2015].

ORB3000 (2010, October). Xda-developers: The history -part one-. [XDA-Developers] `http://www.xda-developers.com/feature/xda-developers-the-history-part-one/` [Accessed 12/April/2013].

Pandey, R. (2013). Aokp rom crosses 3.5 million users; releases android 4.3 based nightlies. http://www.androidbeat.com/2013/09/ aokp-rom-crosses-3-5-million -users-releases-android-4-3-based-nightlies/ [accessed 10/Apr/2014].

Parker, G. and M. Van Alstyne (2010). Innovation, openness & platform control. In *Proceedings of the 11th ACM conference on Electronic commerce*, EC '10, New York, NY, USA, pp. 95–96. ACM.

Perry-Smith, J. E. and C. E. Shalley (2003). The social side of creativity: A static and dynamic social network perspective. *The Academy of Management Review 28*(1), 89 – 106.

Pettey, C. (2011). Gartner says sales of mobile devices grew 5.6 percent in third quarter of 2011; smartphone sales increased 42 percent. Gartner Research – 24/Aug/2012 – `http://www.gartner.com/it/page.jsp?id=1848514`.

Pierce, D. (2013). Android 4.4 kitkat: Google's simpler, integrated operating system designed for every phone. http://www.theverge.com/2013/10/31/5049672/android-kit-kat-4-4-google-software-operating-system [accessed 28 Jan 2014].

Piller, F., C. Ihl, and F. Steiner (2010). Embedded toolkits for user co-design: A technology acceptance study of product adaptability in the usage stage. *43Rd Hawaii International Conference On Systems Sciences Vols 1-5 (Hicss 2010)*, 164 – 173.

Proffitt, B. (2010). Garrett's linuxcon talk emphasizes lessons learned from android/kernel saga. [Linux.com - 16/Aug/2012] `http://goo.gl/6YJbp`.

Prügl, R. and M. Schreier (2006). Learning from leading-edge customers at the sims: Opening up the innovation process using toolkits. *R&D Management 36*(3), 237–250.

Quinn, R. E. and K. Cameron (1983). Organizational life cycles and shifting criteria of effectiveness: Some preliminary evidence. *Management science 29*(1), 33–51.

Raymond, E. S. (1998). The cathedral and the bazaar. *First Monday 3*(3), 1 – 45.

Raymond, E. S. (2001). *The Cathedral & the Bazaar: Musings on linux and open source by an accidental revolutionary*. O'Reilly Media, Inc.

Reardon, M. and S. Shankland (2013). Google engineers: We're trying to fix android fragmentation. http://news.cnet.com/8301-1023_3-57584973-93/google-engineers-were-trying-to-fix-android-fragmentation/ [accessed 28 Jan 2014].

Ritchie, O. and K. Thompson (1978). The unix time-sharing system. *Bell System Technical Journal, The 57*(6), 1905–1929.

Roberts, J. A., I.-H. Hann, and S. A. Slaughter (2006). Understanding the motivations, participation, and performance of open source software developers: A longitudinal study of the apache projects. *Management Science 52*(7), 984 – 999.

Rose, M. B., T. Love, and M. Parsons (2007). Path-dependent foundation of global design-driven outdoor trade in the northwest of england. *International Journal Of Design 1*(3), 57 – 68.

Ross, D. A. R. (2007). Backstage with the knowledge boys and girls: Goffman and distributed agency in an organic online community. *Organization Studies 28*(3), 307 – 325.

Rossman, G. B. and B. L. Wilson (1985). Numbers and words combining quantitative and qualitative methods in a single large-scale evaluation study. *Evaluation review 9*(5), 627–643.

Sánchez-González, G., N. González-Álvarez, and M. Nieto (2009). Sticky information and heterogeneous needs as determining factors of r&d cooperation with customers. *Research Policy 38*(10), 1590–1603.

Scheraga, C. A., W. M. Tellis, and M. T. Tucker (2000). Lead users and technology transfer to less-developed countries: Analysis, with an application to haiti. *Technology in Society 22*(3), 415–425.

Schilling, M. A. (2005). A small world network model of cognitive insight. *Creativity Research Journal 17*(2), 131 – 124.

Schneider, A., G. Von Krogh, and P. JäGer (2013). "what's coming next?" epistemic curiosity and lurking behavior in online communities. *Computers in Human Behavior*.

Schreier, M., S. Oberhauser, and R. Prügl (2007). Lead users and the adoption and diffusion of new products: Insights from two extreme sports communities. *Marketing Letters 18*, 15 – 30.

Schreier, M. and R. Pruegl (2008). Extending lead-user theory: Antecedents and consequences of consumerslead userness. *Journal Of Product Innovation Management 25*(4), 331 – 346.

Shah, S. (2000). Sources and patterns of innovation in a consumer products field: Innovations in sporting equipment. Technical report, Sloan Working Paper.

Shah, S. K. (2006). Motivation, governance, and viability of hybrid forms in open source software development. *Management Science 52*(7), 1000 – 1014.

Shah, S. K. and M. Tripsas (2007). The accidental entrepreneur: The emergent and collective process of user entrepreneurship. *Strategic Entrepreneurship Journal 1*, 123 – 140.

Sharma, S., V. Sugumaran, and B. Rajagopalan (2002). A framework for creating hybrid-open source software communities. *Information Systems Journal 12*, 7 – 25.

Singh, P. V., Y. Tan, and V. Mookerjee (2011). Network effects: The influence of structural capital on open source project success. *MIS Quarterly 35*(4), 813 – A7.

Slaughter, S. (1993). Innovation and learning during implementation a comparison of user andmanufacturer innovations. *Research Policy 22*(1), 81 – 95.

Star, S. L. and J. R. Griesemer (1989). Institutional ecology,translations' and boundary objects: Amateurs and professionals in berkeley's museum of vertebrate zoology, 1907-39. *Social studies of science 19*(3), 387–420.

Steiner, F., C. Ihl, F. Piller, and R. T. Tarman (2011). Embedded toolkits: Identifying changing user needs during product usage. *Emj-Engineering Management Journal 23*(4), 3 – 13.

Steiner, F., R. T. Tarman, J. C. Ihl, and F. T. Piller (2009). Learning from the customer: Identifying changing user needs during product usage through embedded toolkits for user innovation. *Proceedings Of Picmet 09 - Technology Management In The Age Of Fundamental Change, Vols 1-5 None*(None), 696 – 706.

Steinmueller, W. E. (1996). The us software industry: An analysis and interpretive history. In D. C. Mowery (Ed.), *The International Computer Software Industry: A Comparative Study of Industry Evolution and Structure*, Chapter 1. Oxford: Oxford University Press.

Stewart, K. J. and S. Gosain (2006). The impact of ideology on effectiveness in open source software development teams. *Mis Quarterly*, 291–314.

Strandburg, K. J. (2009). User innovator community norms: At the boundary between academic and industry research. *Fordham Law Review 77*(5), 2237 – 2274.

Stuermer, M., S. Spaeth, and G. von Krogh (2009). Extending private collective innovation: A case study. *R&D Management 39*(2), 170 – 191.

Szulanski, G. (1996). Exploring internal stickiness: Impediments to the transfer of best practice within the firm. *Strategic management journal 17*(S2), 27–43.

Tanenbaum, A. S. and H. Bos (2014). *Modern operating systems*. Prentice Hall Press.

Teixeira, J. and R. Suomi (2010). Chronic patients as developers of innovative healthcare information systems. *Proceedings Of The 4Th European Conference On Information Management and Evaluation None*(None), 381 – 387.

Thompson, M. (2005). Structural and epistemic parameters in communities of practice. *Organization Science 16*(2), 151–164.

Thorelli, H. B. (1986). Networks: between markets and hierarchies. *Strategic management journal 7*(1), 37–51.

Torvalds, L. (1999). The linux edge. *Communications of the ACM 42*(4), 38–39.

Torvalds, L. (2007). Linus torvalds on git. YouTube – 20/Aug/2012 – `http://youtu.be/4XpnKHJAok8`.

Urban, G. L. and E. von Hippel (1988). Lead user analyses for the development of new industrial products. *Management science 34*(5), 569–582.

Uzzi, B. (1997). Social structure and competition in interfirm networks: The paradox of embeddedness. *Administrative Science Quarterly 42*(1), 35 – 67.

Valverde, S., G. Theraulaz, J. Gautrais, V. Fourcassie, and R. Sole (2006). Self-organization patterns in wasp and open source communities. *Intelligent Systems, IEEE 21*(2), 36 – 40.

von Hippel, E. (1976). The dominant role of users in the scientific instrument innovation process. *Research Policy 5*, 212 – 239.

von Hippel, E. (1986). Lead users: A source of novel product concepts. *Management science 32*(7), 791–805.

von Hippel, E. (1988). *Sources of Innovation*. Oxford: Oxford University Press.

von Hippel, E. (1989). New product ideas from lead users. *Research Technology Management 32*(3), 24–27.

von Hippel, E. (1990a). The impact of "sticky" information on innovation and problem-solving.

von Hippel, E. (1990b). Predicting the source of commercially valuable user innovation via lead users. *Advances in Telecommunication Management 1*, 1.

von Hippel, E. (1994). Sticky information and the locus of problem solving: Implications for innovation. *Management Science 40*(4), pp. 429–439.

von Hippel, E. (1998). Economics of product development by users: The impact of sticky local information. *Management Science 44*(5), 629 – 644.

von Hippel, E. (2001). Innovation by user communities: Learning from open-source software. *MIT Sloan Management Review 42*(4), 82 – 86.

Von Hippel, E. (2001). Open source shows the way: Innovation by and for users–no manufacturer required. *Sloan Management Review 42*(4), 82–86.

von Hippel, E. (2001). Perspective: User toolkits for innovation. *Journal of product innovation management 18*(4), 247–257.

von Hippel, E. (2005). *Democratizing Innovation*. Massachusetts: The MIT Press.

von Hippel, E. (2007). Horizontal innovation networks—by and for users. *Industrial and corporate change 16*(2), 293–315.

von Hippel, E., N. Franke, and R. Pruegl (2009). Pyramiding: Efficient search for rare subjects. *Research Policy 38*(9), 1397 – 1406.

von Hippel, E. and R. Katz (2002). Shifting innovation to users via toolkits. *Management Science 48*(7), 821 – 833.

von Hippel, E. and G. von Krogh (2003). Open source software and the private-collective innovation model: Issues for organization science. *Organization Science 14*(2), 209 – 223.

von Hippel, E. and G. von Krogh (2006). Free revealing and the private-collective model for innovation incentives. *R&D Management 36*(3), 295 – 306.

von Krogh, G. (2003). Open-source software development. *MIT Sloan Management Review 44*(3), 14 – 18.

von Krogh, G., S. Haefliger, S. Spaeth, and M. W. Wallin (2012). Carrots and rainbows: Motivation and social practice in open source software development. *MIS Quarterly 36*(2), 649 – 676.

von Krogh, G., S. Spaeth, and K. R. Lakhani (2003). Community, joining, and specialization in open source software innovation: A case study. *Research Policy 32*(7), 1217 – 1241.

Wasserman, S. and K. Faust (1997). *Social Network Analysis: Methods and Applications*. Cambridge: Cambridge University Press.

Wenger, E. C. (1999). *Communities of Practice; Learning, Meaning, and Indentity*. Cambridge: Cambridge University Press.

Wenger, E. C. and W. M. Snyder (2000). Communities of practice: The organizational frontier. *Harvard Business Review 78*(1), 139 – 145.

West, J. (2003). How open is open enough?: Melding proprietary and open source platform strategies. *Research Policy 32*(7), 1259 – 1285. Open Source Software Development.

West, J. and K. R. Lakhani (2008). Getting clear about communities in open innovation. *Industry & Innovation 15*(2), 223 – 231.

West, J. and S. O'Mahony (2008). The role of participation architecture in growing sponsored open source communities. *Industry and Innovation 15*(2), 145 – 168.

Whitwam, R. (2010). Verizon vs at&t vs sprint vs t-mobile: How each carrier controls android. Tested.com – 18/Aug/2012 – `http://goo.gl/nQPWO`.

Wiertz, C. and K. de Ruyter (2007). Beyond the call of duty: Why customers contribute to firm-hosted commercial online communities. *Organization Studies 28*(3), 348 – 376.

Wynn Jr, D. E. (2004). Organizational structure of open source projects: A life cycle approach.

Xu, J., Y. Gao, S. Christley, and G. Madey (2005). A topological analysis of the open souce software development community. In *System Sciences, 2005. HICSS '05. Proceedings of the 38th Annual Hawaii International Conference on*.

Yamauchi, Y., M. Yokozawa, T. Shinohara, and T. Ishida (2000). Collaboration with lean media: How open-source software succeeds. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, CSCW '00, New York, NY, USA, pp. 329–338. ACM.

Yin, R. (1989). *Case Study Research: Desig and Methods*. SAGE Publications.

Zeitlyn, D. (2003). Gift economies in the development of open source software: Anthropological reflections. *Research Policy 32*(7), 1287 – 1291.

Zhao, X. and M. Bishop (2011). Understanding and supporting online communities of practice: Lessons learned from wikipedia. *Educational Technology Research and Development 59*, 711–735.

Appendix

A HTML Github Python Parser

```
#!/usr/bin/env python
# coding=utf-8

# parses the html in the repository downloaded pages to get the data
# Jose Christian, 2013.

import bs4
from datetime import datetime
from os import listdir,mkdir

## Function for parsing html ##
# search directory for html files
def _parse_repo(str_name):
    str_directory = '%s/raw_html/' % str_name
    str_out_file = '%s/%s_data.csv' % (str_name,str_name)

    # full output list
    list_full_output = []

    # loop through each file in the directory
    for str_file in [i for i in listdir(str_directory) if i.endswith('.html')]:

        # open file
        with open(str_directory+str_file,'r') as fr_html:
            str_html = fr_html.read().decode('utf-8')

        # pass on to html reader
        bs4_html = bs4.BeautifulSoup(str_html)

        # find and list all the commit groups
        list_rest = bs4_html.find_all('ol','commit-group')

        # loop through each of the results
        for bs4_entry in list_rest:

            # find and list all entries
            list_each = bs4_entry.find_all('li','commit-group-item')

            # loop through each entry
            for bs4_item in list_each:

                # find the time of commit and format it
                str_raw_date = bs4_item.find('time').text
                dt_date = datetime.strptime(str_raw_date,'%B %d, %Y')
                str_date = dt_date.strftime('%Y%m%d')

                # get the author name
                str_author = bs4_item.find(rel='author').text

                # get the commit title
                str_title = bs4_item.find('a',class_='message').text

                # find and label type of contribution
                if 'translation' in str_title.lower():
```

```
                    if str_title.startswith('Merge'):
                        str_type = 'tm'
                    else:
                        str_type= 't'
                elif 'translation' not in str_title.lower():
                    if str_title.startswith('Merge'):
                        str_type = 'pm'
                    else:
                        str_type = 'p'

                # create the output
                str_outline = '%s,%s,%s,%s' % (str_date,
                                               str_author,
                                               str_type,
                                               str_title)

                # populate output list
                list_full_output.append(str_outline.encode('utf-8'))

        # check code has run with no errors
        print '%s - OK' % str_file

    # create output file
    fw_output = open(str_out_file,'w')
    fw_output.write('\n'.join(sorted(list_full_output)))
    fw_output.close()

# list the names of each repository
list_names = ['apollo','bluetoothext','clock','cmaccount',
              'cmfm','cmupdater','cmwallpapers','dspmanager',
              'fmradio','samsungservicemode','torch','trebuchet','voiceplus']

# loop through the list of names and run the above function
for str_dir_name in list_names:
    _parse_repo(str_dir_name)
```

B Semantics Analysis

```
#!/usr/bin/env python
# coding=utf 8

# Semantic analysis of commit titles
# Jose Christian , 2013.

import nltk
import bs4
import os
import pandas as pd
import numpy as np

# create list of html files to be analysed
str_dir = 'data_files/'
list_files = [x for x in os.listdir(str_dir) if x.endswith('html')]

# global output list
list_nn = []

# loop through files
for str_file in list_files:
    # open files
    with open(str_dir+str_file , 'r') as fr:
        str_html = fr.read()

    # read html
    bs_html = bs4.BeautifulSoup(str_html)

    # find relevant commit title
    list_bs_ct = bs_html.findAll(class_='commit-title ')

    # loop through commit titles as string
    for bs_ct in list_bs_ct:
        str_title = bs_ct.a.text.lower()

        # split and tokenize titles
        nltk_tok = nltk.word_tokenize(str_title)

        # tag each token
        nltk_tag = nltk.pos_tag(nltk_tok)

        # extract key words
        for each_tag in nltk_tag:
            if 'NN' in each_tag[1]:
                list_nn.append(each_tag[0])

# count
se_count = pd.Series(list_nn).value_counts()

# save to output file
se_count.to_csv('output_data.csv')
```

C Initial Interview Questions

1. How would you describe your role in the CM project?
2. Have you ever been told in any way, what your specific role in the CM project is? If so, by whom?
3. Has anyone told you to do something specific for the project? If so, who?
4. Has there been a time when you have said 'This is not my job' or 'someone else should be doing this'?
5. In your opinion, what are the different groups (translators, developers, etc) that contribute directly to the ROM?
6. Do you regularly speak to non-programming CM members? If so, who and why?
7. Who do you go to if you have a problem you cannot solve?
8. What kind of judgement calls would you be comfortable making?
9. Who do you think is responsible for making the big decisions in CM?

Module	Layer	Users	Contri-butions	Files	Centrality	Density	Commu-nities	Modularity
apollo	Core	3	4	53	0.019	0.481	2	0.000
	Peripheral	60	243	439	0.272	0.240	9	0.234
	Translation	5	70	113	0.263	0.246	8	0.070
bluetoothext	Core	1	4	3	0.000	0.500	1	0.000
	Peripheral	16	110	450	0.239	0.141	9	0.324
	Translation	1	15	263	0.199	0.338	7	0.028
cmaccount	Core	4	29	43	0.368	0.265	4	0.137
	Peripheral	47	215	171	0.565	0.130	4	0.433
	Translation	4	68	79	0.415	0.240	3	0.112
cmfilemanager	Core	3	21	23	0.435	0.097	8	0.339
	Peripheral	76	627	891	0.514	0.108	9	0.467
	Translation	6	103	126	0.321	0.206	8	0.107
cmupdater	Core	6	37	48	0.515	0.242	3	0.168
	Peripheral	63	257	163	0.412	0.182	7	0.216
	Translation	6	83	139	0.348	0.176	6	0.227
cmwallpapers	Core	5	13	63	0.172	0.239	2	0.329
	Peripheral	39	83	134	0.446	0.100	11	0.469
	Translation	9	42	55	0.352	0.229	10	0.081
dspmanager	Core	5	27	22	0.148	0.052	10	0.486
	Peripheral	73	186	116	0.254	0.067	11	0.558
	Translation	17	46	36	0.244	0.113	6	0.278
fmradio	Core	0	0	0	0.435	0.097	0	0.000
	Peripheral	4	48	229	0.054	0.454	5	0.005
	Translation	0	0	0	0.000	0.000	0	0.000
lockclock	Core	6	88	307	0.439	0.168	6	0.330
	Peripheral	66	272	782	0.212	0.366	7	0.035
	Translation	5	86	123	0.445	0.203	6	0.348
samsung	Core	0	0	0	0.000	0.000	0	0.000
service	Peripheral	18	22	27	0.346	0.090	14	0.180
mode	Translation	4	4	11	0.178	0.327	3	0.000
torch	Core	8	22	22	0.248	0.245	5	0.113
	Peripheral	68	108	84	0.408	0.235	6	0.170
	Translation	16	28	20	0.175	0.026	15	0.355
trebuchet	Core	5	13	33	0.344	0.167	7	0.153
	Peripheral	31	168	1010	0.235	0.305	12	0.036
	Translation	1	16	105	0.187	0.351	7	0.102
voiceplus	Core	3	29	29	0.456	0.270	2	0.206
	Peripheral	20	21	21	0.050	0.002	20	0.000
	Translation	3	3	3	0.000	0.000	3	0.000

Table 1: Table showing data from the association network for the CyanogenMod repository.

D Network Statistics