Exploring How the Attribute Driven Design Method is Perceived

Nour Ali,

University of Brighton, Brighton, UK, n.ali2@brighton.ac.uk

Carlos Solis

Amazon, Dublin, Ireland, csolis@amazon.com

Abstract. A method for designing software architecture based on achieving quality attributes is the Attribute Driven Design (ADD) method. This method has not been explored in terms of users' perception on its usefulness and easiness of use. Our goal is to study the perceptions that software engineers with no or little experience industrially in designing software architecture using the ADD. In this paper, we describe an empirical study that we conducted on master students to measure their perceptions of the ADD after using it. We performed two experiments; one with students enrolled in the Software Architecture module in 2010 and repeated it with the students of the same module in 2011. Our main findings are that the subjects perceive ADD method as useful, and that there is a relationship between its usefulness and willingness of use. However, the subjects' opinion was that they did not agree that the ADD method is easy to be used. We also discuss several problems that the subjects faced when using ADD.

Keywords: Attribute Driven Design, Quality Attributes, empirical study, software architecture, TAM

1. Introduction

Software architecture is a coordination tool among the different phases of software development. It bridges requirements to implementation and allows reasoning about satisfaction of systems' critical requirements [23]. Quality attributes [14] are one kind of non-functional requirements that are critical to systems. The Software Engineering Institute (SEI) defines a quality attribute as "a property of a work product or goods by which its quality will be judged by some stakeholder or stakeholders" [33]. They are important properties that a system must exhibit such as scalability, modifiability or availability [38].

A method for designing software architecture based on quality attributes is the one defined by the SEI called the Attribute Driven Design (ADD) method [3], [38]. The ADD method is based on an iterative process for designing software architecture based on applying architectural tactics and patterns which satisfy quality attributes. This method has been applied in different domains such as fault tolerant systems [39], adopted in companies as stated in [23] and is composing part of curricula for training software architects [34].

Research has been performed in evaluating the different artefacts, and activities of ADD [27], [22]. However, not much scientific research has reported on experiences using this method [10]. We are investigating how software engineers with no or little industrial experience in software architecture design and quality attributes perceive the usefulness and easiness of use of the ADD method. Design methods can only be effective if users adopt and accept them. Although novice software engineers are not experienced software architects, it is interesting to understand how they perceive these kinds of methods to be part of their training and if they are willing to adopt them in the future. In addition, we are interested in understanding the problems they face when applying the ADD.

This paper presents an empirical study to explore ADD method users' perception. We apply the Technological Acceptance Model (TAM) [6], which investigates people's attitudes and behaviour towards accepting an innovation. We designed a survey with questions that follow the TAM model in order to measure ADD method perceived usefulness, ease of use and willingness of use. We performed an experiment in the year 2010 with 12 participants, and repeated it in the year 2011 with 7 participants. All the participants were postgraduate students in the software engineering master programme, enrolled in the software architecture module of each year. After they received 8 hour training on the ADD method, and designed and documented a software architecture following the ADD method for one month and two weeks, they were asked to fill a survey.

Our findings reveal that the subjects find ADD method useful and that this has a positive influence on their willingness of using it. However, the subjects have a neutral/negative opinion on its easiness of use and a neutral opinion on their willingness to use it. We also discuss the problems that the subjects faced when using the ADD and which have influenced their opinions.

The paper is structured as follows: In Section 2, we describe the ADD method and the TAM model. Section 3 explains the empirical study. Section 4 describes the data and the results of the study. Section 5 presents a discussion about the problems faced by the subjects when using ADD method, an analysis of the results, several lessons learnt from the study, and the validity threats. Finally, section 6 presents conclusions and further work.

2. Background

In this section, we give an overview about the ADD method and the TAM model.

2.1 Attribute Driven Design (ADD) Method

The Attribute Driven Design (ADD) method is an iterative approach, proposed by the SEI, for designing software architecture in order to meet functional requirements as well as quality ones [3]. ADD method uses a recursive decomposition process based on the quality attributes that a system needs to fulfill. At each stage, tactics and architectural patterns are chosen to satisfy some qualities, and functionality is allocated to instantiate the architectural element types. A practical example applying ADD is presented in [39].

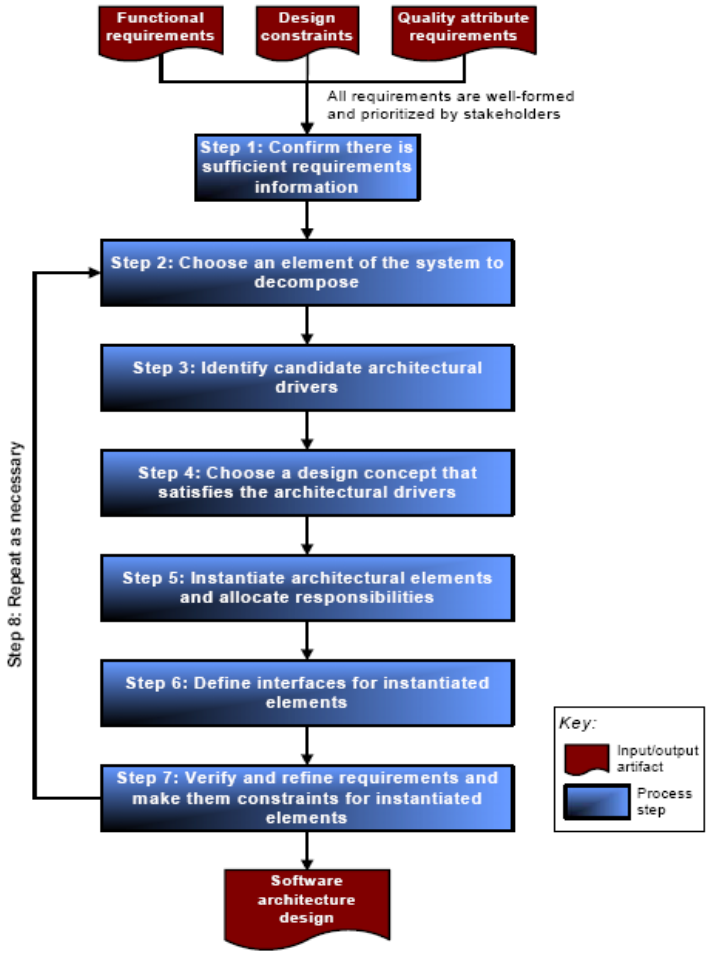

Figure 1. **Steps of ADD taken from [38]**

In the following, we give a brief explanation of each ADD step based on [38] (see Figure 1). **In the first step**, a list of stable and prioritized requirements including functional, constraints and quality attributes have to be available to start ADD. Functional requirements are expressed using use cases, and quality attributes are expressed using quality attribute scenarios templates (stimulus- response) [3]. **In the second step**, the iterative process can start. If you are in the first iteration, your system is the element to be decomposed. If not, you choose an architectural element to start decomposing.

In the third step, the ranking performed by stakeholders of the requirements is combined with rankings based on their impacts on the architecture. Five to six high-priority requirements are chosen and are called candidate architectural drivers. **In the fourth step**, the architectural element types and their relationships are chosen. To identify them, the design constraints and quality attributes (candidate architectural drivers) are used to identify patterns and tactics (design options for achieving quality attributes) that satisfy them, and architectural views are partially captured.

In the fifth step, the chosen architectural element is decomposed into children elements. Responsibilities are allocated to the children elements using use cases, and architectural views from step 4 are complemented. In **the sixth step**, the services and properties that are provided and required for each element are defined. Finally, in **step seven**, architects verify that the functional requirements, quality attributes and design constraints have been met in the element decomposition and that the child elements functional and quality attribute requirements are allocated. After the last step, a new iteration of ADD can begin. Each iteration decomposes an architectural element into children elements. The number of iterations applied is a decision that architects have to perform and depends on the abstraction of the architecture.

Example: Imagine a Travel Booking Agency Web Portal that needs its software architecture designed. By using the ADD method, the first step would be to have stable functional requirements and quality attributes expressed as scenarios. For example, a modifiability quality scenario is “The system shall allow a developer to modify the user interface with no side effects in less than 3 hours”.

In the second step, the iterative process starts and the whole travel booking agency system is the architectural element to be decomposed. In the third step, based on the importance to stakeholders and to its relative impact on the architecture, modifiability is the most ranked architectural driver. In the fourth step, the architectural tactics classified for modifiability in Bass et al. [4] are studied and chosen. For example, “Prevent Ripple Effects” is a modifiability tactic and a subtatic is to “Use an Intermediary”. An architectural pattern that satisfies these tactics is the Model View Controller (MVC). In the fifth step, the travel booking system is separated into the Model, View and Controller elements. For example, a customer will have a view showing details of their booking and profile and the agency employee will have a view of the different car hire, hotel and flight companies. In the sixth step, the services of properties of the MVC are defined. In the seventh step, the architect verifies that the MVC has met modifiability. As MVC

is a well-documented pattern it does. Then, the architect can start another iteration to decompose the Model element.

ADD method has been used to design architectures of different domains such as machine learning [8], embedded systems [27] or geographical information systems [16]. Ferreri and Madhavji performed an exploratory study in [11] to research whether architects with requirements engineering knowledge develop better quality software architectures than architects without. In their study, the ADD method was used for developing the architecture. However, their objective was not to study how humans perceive its usefulness and easiness of use.

ADD method has been evaluated in [22] and in [19] from an ontology point of view and has been compared with other architecture design methods such as Siemens Four Views [20] or Rational Unified Process [25]. Falessi et al. [10] evaluate ADD from software architects' needs perspective. They use nine categories which are believed to be real needs of software architects and use them to compare the design methods, including ADD. These categories include tool support, ability to react to requirements, support for architectural views, etc. It can be concluded from [10] that the ADD has been minimally validated empirically by humans.

2.2 Technology Acceptance Model

The technology acceptance model (TAM) [6] is used to explain the individual acceptance of new technology or innovations. In the TAM model, accepting or rejecting an innovation is measured through two variables, which are "perceived usefulness" and "perceived ease of use". An innovation is perceived as useful when its users believe that it would help them perform their job. However, if an innovation is useful it might not be adopted because it requires too much effort to be used. An innovation is perceived as easy to be used when people believe that using an innovation would be effortless. The third TAM variable is "willingness of use" that represents people's intention of using an innovation.

TAM has been used for assessing perceived usefulness and ease of use of technologies such as web tools in education [37] and in software engineering techniques such as business process modelling [9]. In this paper, we adopt the TAM model to evaluate the acceptance of the ADD method by the students.

3. The Empirical Study

In this section, we describe the study that was conducted to explore how users perceive the ADD method.

3.1 Research Questions

In this study, our objective is to investigate on the following research question.

RQ: How do students perceive the ADD method?

To answer this question, we have divided it into these two sub questions:

RQ1: How do students perceive the usefulness and easiness of use of the ADD method?

The hypotheses to be contrasted to answer this research question are the following:

H1: ADD is perceived as useful.

H2: ADD is perceived as easy to use.

H3: There is willingness of using ADD.

H4: Perceived usefulness of ADD is correlated with willingness of use.

H5: Perceived easiness of use of ADD is correlated with willingness of use.

H6: Perceived usefulness of ADD is correlated with easiness of use.

RQ2: What are the problems that users face when applying the ADD?

To answer this question, we will ask the subjects about their problems and understand the practices they have used.

3.2 Experiment Design and Study Variables

The design of this study is a survey design, and a non-experimental design [29], [24]. Since we performed the same study in two years, the one performed in 2011 is considered to be a closed replication [28]. For the study, we designed a questionnaire to contrast the hypotheses, which is composed of three sets of questions (items) in order to measure the three TAM variables: perceived usefulness, perceived ease of use, and willingness of use (see Table 1).

The questionnaire consists of 12 questions using a 7-point (1–7) Likert scale (1, extremely unlikely; 2, quite unlikely; 3 slightly unlikely; 4, neither; 5, slightly likely; 6, quite likely; 7, extremely likely). We deal with Likert scale as ordinal data, because the answers to the questions can be meaningly ordered from lowest to highest but the intervals between the scoring are not equal psychological intervals [21].

We adapted the items of the TAM model. The items of perceived usefulness are the same of TAM but we focused the questions on the ADD method and architecture design. In the case of perceived usefulness, some of the original TAM items were focused on users interacting with tools or technologies. We removed the items as "*I would find it easy to get X to do what I want it to do"*, *"My interaction with X would be clear and understandable"* or *"I would find X to be flexible to interact with"* [5]. We left the ones shown in Table 1. To measure the willingness of use, we have one question that asks the users if they will use ADD for designing software architectures.

Table 1. Items to measure ADD method perceived usefulness, perceived ease of use and willingness of use

Items regarding "Perceived Usefulness" (PU)	
PU1	Using ADD would improve my *performance* [1]in Architecture Design
PU2	Using ADD would increase my *productivity* in Architecture Design
PU3	Using ADD would enhance my *effectiveness* [2]in Architecture Design
PU4	Using ADD would enable me to accomplish Architecture Design tasks *more quickly*
PU5	I find that ADD would be *useful* in Architecture Design
PU6	Using ADD would *make it easier* to design architecture
Items regarding "Easiness of Use" (PEU)	
PEU1	*Learning* to use ADD is easy for me
PEU2	It will be possible to use ADD without *expert help*
PEU3	It is easy for me to become *skilful* at using ADD
PEU4	It is easy to *remember* how to perform the ADD
PEU5	I would find ADD *easy to use*
Items regarding "Willingness of use" (WU)	
WU1	I *will use* ADD for designing software architectures.

3.3 Participants and training

We used availability (or convenience sampling) [29], where the participants were all graduate students in the Software Architecture module of software engineering master programme, of the University of Limerick. In the first experiment conducted in 2010, they were 12 participants, divided into 4 architecting teams. In the experiment conducted in 2011, they were 7 participants, divided into 3 groups. 1 group had 3 members, and the 2 other groups had 2 members.

The students had some background in requirements engineering, including use cases or viewpoints and Software Design including the knowledge of several design patterns, and modelling in UML. They did not have previous knowledge in specifying quality attribute scenarios which is needed for using the ADD method or using a pattern driven method.. Previously of getting the training in ADD, the students had previous experience using artefact driven architecture design and use case driven methods, which they used in other projects. They had also previous knowledge in Rational Unified Process (RUP).

The students were given the following training background as part of the ADD:

- 2 hour lecture reviewing definitions about software architecture, its concepts, elements and architectural views;
- 2 hour lecture about Quality Attribute Scenarios that included understanding Quality attributes by using the Quality Attribute Scenarios and achieving Quality attributes through Tactics. The contents of the lecture were based on chapters 4 and 5 of Bass et al. [4] and [38].

[1] Performance: Effort expended to achieve results

[2] : Effectiveness: Doing the right thing without mistakes

- 2 hour lecture about architectural styles/patterns that presented many architectural styles found in the literature. The contents of this lecture were based on styles/patterns presented in Chapter 4 and 11 of Taylor et al. [36] and Chapter 3 of Shaw and Garlan [29].
- 2 hour lecture was about designing architectures using the ADD method, which contents were based on Chapter 7 of Bass et al. [4]. In addition, few minutes were also dedicated to explain documentation of architectures using templates.

3.4 The architecting project

In the two years this study has been performed we used two different projects, but tried to keep their requirements size and difficulty similar. This was decided in order not to allow students of the year 2011 to be able to copy the solutions of students of previous years. In the year 2010, the system to be architected was about a web portal for the university library that allows users to search for different kinds of publications. The web page should be accessible from desktops, or mobile devices. In the year 2011, the project was about a travel agency web portal that allows users to search for hotels and flights, and book them.

These projects were chosen since both of them are web portals; needed integration with different kinds of external systems such as payment systems; and need to satisfy many quality attributes such as security, modifiability, scalability, performance and availability. At the same time, the students are familiar with both the library and travel agency web portals and this would minimize the possibility of having problems in understanding their requirements.

All teams were responsible for designing and documenting independently of the other teams the software architecture using the ADD method. Each team was given the option to decide on the number of iterations. They were also asked to use a template for documenting the software architecture. This template was adapted from [14], which included sections to document quality attribute scenarios [4], use cases, constraints, architectural styles and patterns, rationale, views, assumptions, architectural elements, interfaces, risks, and implementation. We also asked them to fill in a template for each iteration of the ADD, documenting the decisions and the rationale taken in each step. This allowed us to ensure that they were following the ADD method correctly and to understand the rationale behind the decisions that they took.

3.5 Data Collection

The teams were given a month and two weeks for handing in software architecture documentation, and a list of problems they faced when applying the ADD method in designing the architecture. There was no interaction between the students and the instructor during this period of time.

Later, the questionnaire of Table 1 was given to each of the students and they were asked to fill it during 20 minutes. Each participant has been asked to respond to each statement in terms of their own degree of agreement or disagreement. The questionnaire was filled in anonymously in order not to provoke doubt between the subjects that their answers would affect their module marks.

4. Results

In the following, we describe the results of the two studies that were conducted.

4.1 Questionnaire Reliability

To measure the reliability of the questionnaire, we calculated the Cronbach's α reliability index for each dimension [29]. Cronbach's α indicates the internal consistency of a set of questions. When Cronbach's α value is above 0.70, it is an acceptable value, although in exploratory studies a threshold of 0.60 is also acceptable [13].

In the case of the perceived usefulness, the Cronbach's α values for 2010 and 2011 are above 0.7, which indicates that the items are a highly reliable measure for the dimension. The Cronbach's α values for perceived ease of use are higher than 0.6, and below 0.7. This indicates that the reliability of the questions in this dimension is acceptable since this is an exploratory study.

Table 2. Validation of questionnaire using Cronbach's α

Dimension	2010	2011
Perceived Usefulness	0.834	0.843
Perceived Ease of Use	0.625	0.666

4.2 Descriptive Statistics

According to Stevens [35], means and standard deviations cannot be used to describe ordinal data. Instead, medians and percentile measures should be applied to describe ordinal data [21],[35]. We describe the items using the median, interquartile range and the mode. Table 3 summarizes the results for items of the usefulness, easiness of use, and willingness of use dimensions for both experiments of 2010 and 2011.

Usefulness of ADD method. For the experiment of 2010, it can be noticed that the median of all items is equal or higher to 5. For the experiment of 2011, the median of the items is equal to 5 instead for PU4, which indicates a slight disagreement that ADD enables to accomplish Architecture Design tasks *more quickly*. In conclusion, we can notice that there can be a positive agreement of perceived usefulness towards the ADD method.

Ease of Use of ADD method. It can be noticed that for the experiment of 2010, the median is equal to 4 for all items excluding PEU1 (Learning), which is 3. The medians for the experiment of 2011are different from the ones of 2010 except for the items PEU1 and PEU2. The medians of 2011 subjects' opinions are negative about items PEU3, PEU4 and PEU5.

Willingnes of Use. For the experiment of 2010, the median and mode were the neutral value (4), and for the experiment of 2011 the median was also neutral, but its mode had the values of 3 and 5.

Table 3. Usefulness, Easiness of Use, and Willingness of Use Results

	Median		Interquartile Range		Mode	
	2010	**2011**	**2010**	**2011**	**2010**	**2011**
Performance (PU1)	5.5	5	1	3	6	5
Productivity (PU2)	5	5	2	2	5	6
Effectiveness (PU3)	5	5	1	1	5	5
More Quickly (PU4)	5	4	1	1	5	3
Useful (PU5)	6	5	1	2	6	5
Make it Easier (PU6)	5	5	2	1	5	5
Learning (PEU1)	3	3	3	3	3	5
Expert Help (PEU2)	4	4	2	3	4	4
Skilful (PEU3)	4	2	3	3	4	2
Remember (PEU4)	4	3	3	1	4	3
Easy To Use (PEU5)	4	2	2	1	4	2
Will of Use (WU1)	4	4	2	2	4	3 & 5

4.3 Hypotheses tests.

In statistical terms, H1 is defined as the median of usefulness that is greater than 24 (which is the sum of 6 answers in the Likert scale with a value of 4), H2 is defined as the median of ease of use is greater than 20, and H3 as the median of willingness of use is greater than 4. These hypotheses can be checked with a one-tailed Wilcoxon signed rank test.

Table 4. Results of the Wilcoxon signed rank test.

	H1		H2		H3	
	2010	**2011**	**2010**	**2011**	**2010**	**2011**
W-	1	3	28.5	26	28.5	10.50
W+	54	25	26.5	2	23	10.50
Ties	2	0	2	0	5	1
Wilcoxon (One-tailed 0.05)	10	3	10	3	3	2

The test has a significance of 0.05. Our hypotheses are accepted if W- is less or equal than the Wilcoxon statistic for N = sample size - ties. In both experiments, H1 is accepted, and H2 and H3 are rejected.

To check the hypotheses H4, H5, and H6 and to study the relationships between the variables, we have used the Spearman's ρ correlation coefficient (see Table 5). Notice that the Spearman's ρ coefficient for the relationship between perceived usefulness of ADD and the willingness to use ADD is 0.681 in 2010 and 0.963 in 2011, and the p-values are positive and significative for both years. Therefore, H4

hypothesis is accepted in both experiments. H5 and H6 hypotheses are rejected because the Spearman correlations between perceived ease of use and willingness to use, and perceived ease of use and perceived usefulness are not significative.

Table 5. Spearman ρ correlations among the dimensions

		2010			2011		
		PU	**PEU**	**WU**	**PU**	**PEU**	**WU**
Perceived Usefulness (PU)	Correlation Coefficient	1	0.210	0.681	1	0.655	0.963
	Sig. (2-tailed)	.	0.512	0.015	.	0.111	0.001
Perceived Ease of Use (PEU)	Correlation Coefficient	0.210	1	0.208	0.655	1	0.523
	Sig. (2-tailed)	0.512	.	0.797	0.111	.	0.228
Willingness of use (WU)	Correlation Coefficient	0.681	0.208	1	0.963	0.523	1
	Sig. (2-tailed)	0.015	0.797	.	0.000	0.228	.

*. Correlation is significant at the 0.05 level (2-tailed).

5. Discussion

In this section, we discuss issues which were faced by the subjects when using ADD method, we interpret and analyze the results, discuss lessons learnt and the validity threats.

5.1 ADD issues faced by subjects

In the following, we discuss the problems that the teams reported they faced when using the ADD method, and how we relate them to the current literature. The teams of 2011 did not report any new problem which was not mentioned by the teams of the 2010 experiment. We have classified the issues into four main categories.

Team Workload Division and Assignment: The ADD method does not explicitly focus on giving guidelines on how a team of architects divide their workload [38]. This can be emphasized on the first iteration. One team of 2010 reported, "*Attribute Driven Design presented a conceptual challenge in terms of dividing workload within the group initially."* It has to be noted that none of the teams that participated in the 2011 experiment reported explicitly this problem. A reason for this could be due to the fact that two of the teams had only two members.

According to the practical example reported in [39], the architecture team perform the first iteration. However, no details are given on how the team divided its workload in the first iteration. In the later iterations of ADD, architects with expertise in specific qualities are assigned to perform the architecting of specific iterations (architectural elements), and iterations which are independent of each other can be performed in parallel by different architects. For example, an architect with expertise in fault tolerance is assigned to perform iterations for fault tolerance components and another architect to perform the ones related to start-up. However,

even this practice is not documented as a guideline or practice in the ADD steps. As future improvement of ADD, specific practices for team workload division and assignment can be explicitly included.

No consensus in terminology: ADD method terminology used to refer to its artefacts is different from the rest of the literature. This problem was faced by the subjects to communicate among them and to look for the appropriate patterns, or tactics needed. A clear example is the usage of the words architectural patterns and architectural styles interchangeably in the literature. A team in 2010 reported, *"it was found that several institutions use identical terms to refer to the same object (such as diagrams, patterns, tactics etc.). This was found by the group to be quite frustrating as we often found ourselves looking at that same material and referring to it using different terms"*. A team in 2011 also reported, *"The terminology surrounding software architecture can be ambiguous at times and is used with varying meanings across the literature surveyed"*.

In relation to this problem is that the different architecture design methods proposed by different organizations or researchers use different terminology. This discourages the reuse of other researchers' experiences, practices and guidelines in the software architecture field. An effort towards addressing this problem is the architecture design model proposed by Hofmeister et al. [19]. This model is a result of comparing and contrasting the different architecture design methods, including the ADD. This kind of work can help in finding out the commonalities and differences in the architecture design methods based on the activities and artefacts, and to encounter which method can complement each other.

Another team reported that they had difficulty in finding patterns that satisfy quality attributes because the language used in patterns does not directly indicate the quality attributes. They reported as follows, *"The language used in patterns and ADD greatly differs and patterns don't always immediately identify what quality attributes they support."* This problem has been indicated in the literature as in [15], and [17]. Although this problem does not only affect the ADD method, it affects in general software architecture designs which are based on quality attributes.

ADD First Iteration: The teams reported that during the first iteration they found difficulties. A team in 2010 reported, *"The first iteration is very confused and it is hard to decompose the system into elements."*, and a team in 2011 reported, *"As the system was the element to decompose in this iteration, the scope was very wide and so it was a challenge to determine where to start"*. It has been noticed that there is no complete example of ADD where the first iteration is demonstrated [39]. In this iteration the number of architectural drivers and requirements is the highest. For the subjects it was difficult to determine which architectural drivers to choose at the first iteration.

For example, in step 2 and 3 when choosing the architectural element to decompose and the candidate architectural drivers, in the second and later iterations of ADD, the subjects chose the architectural element to decompose based on criticality of the functional and quality attribute scenarios associated with it or based on the quality attributes. In the first iteration, it is hard to understand which are the most critical functional and quality attributes and from where to start.

Mapping Quality Attributes to Tactics, and Tactics to Patterns: This problem is specific to step 4 where the types of elements and their relationships based on tactics, architectural styles or both are identified. We have noticed that the users

have performed the following practices in this step to map the quality attributes to tactics or patterns: they used tactics and defined their own patterns, they chose existing patterns and then verified that each pattern satisfies a set of tactics and quality attribute scenarios, or they reflected on several tactics to satisfy a quality scenario and then chose a pattern. To identify the elements and their relationships they have added elements responsible for a quality attribute and the functionality or elements that have been defined in the architectural style.

The teams found it complicated to choose or find tactics that satisfy quality attributes. In addition, they found complicated defining, evaluating and assessing which architectural patterns are suitable to implement the tactics and quality attributes. A team reported in 2010, "*Difficulty in locating suitable patterns to fulfil tactics identified for a quality attribute"* another reported, *"It was not clear how to relate the tactics to components."* In 2011, a team reported, *"It is difficult to find one style to fit every aspect of the software architecture. It is also difficult to identify architectural pattern combinations to the context"*.

In the literature, these problems have been reported and considered, but still much has to be performed. Bachman et al. [1] describe steps for deriving architectural tactics. These steps include identifying candidate reasoning frameworks which include the mechanisms needed to use sound analytic theories to analyze the behavior of a system with respect to some quality attributes [2]. However, this involves that architects need to be familiar with formal specifications which are specific to quality models. Research tools are being developed to aid architects integrate their reasoning frameworks [7], but still reasoning frameworks have to be implemented, and tactics description and how they are applied has to be indicated by the architect. It has also been reported in [31] that some quality attributes do not have a reasoning framework.

Architecture prototyping is an approach to experiment whether architecture tactics provide desired quality attributes or not, and to observe conflicting qualities [3]. This technique can be complementary to ADD method. However, it has been noted to be quite expensive and that "substantial" effort must be invested to adopt architecture prototyping [5].

Harrison and Avgeriou analyze the impact of architectural patterns on quality attributes, and how patterns interact with tactics [17], [18]. The documentation of this kind of analysis can aid in creating repositories for tactics and patterns based on quality attributes. ADD depends on these kinds of repositories which currently do not exist.

5.2 Analysis of the Results

In both years, we can **accept** our hypothesis "*H1: ADD is perceived as useful"*. This emphasizes literature reviews such as [10] where ADD method is one of the best methods which meet software architects' needs. This can be attributed to the fact that it deals with quality attributes in an explicit way.

On the other hand, we **reject** the hypothesis "*H2: ADD is perceived as easy to use*". It can be noticed that the median for the item PU5 in the year 2010 was neutral, however, for the subjects of 2011 the median is 2, i.e., they answered that they

quite unlikely perceive ADD method as easy. In addition, if we compare W+ to the Wilcoxon statistic, we can know if the median of ease of use is less than the neutral value. We can confirm that for 2010, H2 is rejected with a neutral opinion, and for 2011, H2 is rejected with a negative tendency, i.e., ADD is not perceived as easy. An interpretation for these results can be that a good amount of the 2010 subjects felt they need more training in the ADD method steps or that they need to practice it more times to decide whether it is easy or not. The subjects of 2011 felt it was difficult and they expressed when explaining their problems that they would require to be more professional and knowledgeable, specifically in understanding quality attributes and applying architectural patterns.

The median of the willingness of use is the neutral value of the likert scale, and confirming it with hypothesis test we reject the hypothesis "*H3: There is willingness of using ADD"*. There can be many interpretations for having a high percentage for neutral answers. One of them can be that due to the fact that our subjects do not practice an industrial job as architects at the moment of the study, and they cannot predict if they will use ADD or not. Another possible interpretation is that the subjects do not know other methods for designing architectures other than the ADD method, and they believed that they needed to have knowledge in other methods to make an appropriate choice.

We found out there is a positive correlation between usefulness and willingness of use in both years. In addition, the study points out that there is no significant correlation between perceived ease of use and willingness to use, as well as a no significant correlation between usefulness and easy to use. Therefore, we accept hypothesis *H4: Perceived usefulness of ADD has a positive incidence over its willingness of use*. Hypotheses *H5: ADD's perceived easiness of use has a positive incidence over its willingness of use* and *H6: Perceived usefulness of ADD has a positive incidence over its easiness of use* are rejected.

Our results could indicate that users of ADD method are driven to adopt it primarily because of its usefulness in architecting rather than on how hard or easy it is to be used. The issues reported in the previous section could have influenced the perceived usefulness and easiness of use in the following ways:

- **Team Workload Division and Assignment:** This issue could have influenced the perceived usefulness of the subjects. Since the workload and assignment can affect the productivity in the architecture design (PU3) and in accomplishing the tasks quicker (PU4).
- **No consensus in terminology:** This issue could have influenced the perceived easiness of use of the subjects. Specifically, the learning of the terminology of ADD could have been perceived hard (PEU1), and the subjects believed that they needed expert help (PEU2), and that they were not skilful enough (PEU3). In addition, the usefulness of the ADD can have been slightly influenced negatively due to this issue because the subjects perceived that they would not be doing the right things (effectiveness PU3).
- **Attribute Driven Design First Iteration:** This problem could have influenced negatively on the easiness of use. Specifically, the learning (PEU1) of the ADD was difficult, and the subjects felt they needed expert help (PEU2). Also, the usefulness of ADD has been affected negatively due to the lack of indication of

how to make decisions such as choosing the architectural drivers felt that they are not effective (PU3).

- **Mapping Quality Attributes to Tactics, and Tactics to Patterns:** This problem could have affected negatively on the usefulness of the ADD method. Since they had problems in identifying patterns and tactics, they could have perceived that they are less effective (PU3) and that the design of their architecture is not easy to be performed using ADD (PU6). In addition, the easiness of use of ADD is affected negatively by most of its items (learning, needed expert help, or being skilful).

5.3 Lessons Learnt

Several of the lessons we learnt from this study can be applied in training software architects and in focusing research directions. From a training perspective, we have learnt that:

- Students perceive that ADD is useful and that ADD enhances their effectiveness in Architecture Design (PU3). From this, we can learn that ADD is appropriate to be part of the software architecture curriculum since the students value its usefulness in giving them guidelines that they can follow when architecting.
- To increase the easiness of use of the ADD, more training has to be given to students by concentrating on the problems they faced. These problems can be software architecture technical but also other aspects. For example, more training on software architecture concerning tactics and the use of architectural styles has to be performed. In addition, although many of the students received previous training in requirements engineering they did not have clear what quality attributes are and had not specified quality attribute scenarios previously.

From a research perspective, we have learnt that:

- Although the ADD does not cover behavioural aspects, the software architecting process is human related e.g., assembling teams, and team work with other architects. Much research should concentrate in this direction. The SEI has also recognized this and is working on combining ADD with a team software process [30].
- Detecting the practices that students/users perform in each ADD step, and the specific challenges they face aids improving its teaching and practical application. As we discussed previously, we have analyzed some of the practices related to some of the problems. It would be interesting to understand how the combination of the different practices chosen on each step can affect the later steps and the resulting architecture.
- Tools for supporting the ADD steps: An integrated tool that supports the ADD step by step would be of great value. This can include verifying that each step has been performed correctly, suggesting recommended practices, and facilitating the voting and the design decision making. In addition, the tool can contain

a repository of the well-known and available tactics and patterns related to quality attributes. This would reduce the effort of users to look for the scattered knowledge.

5.4 Threats to Validity

Since our research design is non-experimental and we cannot make cause-effect statements, internal validity is not contemplated [29].

Face validity- The questions were shown to two researchers that are not involved in this research. They indicated that the terms efficiency and productivity which are often used in TAM questions are not easy to understand. As a result, the terms were explained in the introduction of the questionnaire.

Content validity- The questionnaire used is based on the established model of TAM for measuring usefulness and ease of use. The items in the questionnaire are similar to the questions used in several studies which have followed TAM.

Criterion validity- We have checked if the results behave according to the theoretical model (TAM). In this case, the criterion validity can be checked by the Spearman's ρ correlation. The correlations among the variables behave in the theoretical expected way. In addition, other TAM studies have also found similar correlations [6].

Construct validity- The internal consistency of the questions was verified with the Cronbach's α. For minimizing bias errors, the researchers did not express to the participants opinions nor have any expectation. The surveys were collected anonymously. In addition, the researchers are not related to the creation of the ADD and the results of the study do not affect them directly.

Conclusion validity- The main threat is the small sample used. However in order to have more meaningful results we have used non-parametric tests instead of parametric tests. This is an exploratory study, the hypotheses built in this study can be used in future studies to be validated with a richer sample. In respect to the random heterogeneity of subjects, the participants have more or less the same design experience, and have received the same training about software architecture design.

External validity- The results can be generalized to novel software architects that have received formal training in software architecture design and in the ADD method. We have repeated the experiment in order to confirm our initial findings with students. The domain of the project was changed in the two experiments, but both of them are web applications with similar characteristics. Untrained architects, and experience architects in practice may have different perceptions than the ones found in this study. however we believe there is common practices to all business related (not critical or real time domains). There is a threat that the academic context is not similar to industrial. In our case, we did not restrict the teams to work in specific hours and times such as in a lab. The development of the tasks was flexible. They were also given a deadline as in real world to deliver the architecture documentation.

Ethical validity- The questionnaire questions and the study method were approved by The Research Ethics Committee of the University of Limerick.

6. Conclusions and Further Work

This paper describes an exploratory study to evaluate the perceived usefulness, easiness of use and willingness to use of the ADD method. The evaluation performed follows the TAM model. The study was performed in an experiment in 2010, and was replicated in 2011 in order to ensure the results obtained in 2010. We have noticed that we have achieved similar results in both experiments, since the acceptance or rejection of our hypothesis did not change.

The subjects are students who had no previous industrial experience in designing software architecture, and who received basic training on the ADD method. They have perceived ADD software architecture design as useful and believe that it has made it easier for them to design architectures. In addition, they have not perceived ADD as easy to be used. In both years, the hypothesis "ADD is perceived as easy" is rejected but in 2010 the students had a neutral opinion while the students in 2011, found the ADD as not easy to be used. In both years, the students have a neutral opinion about their willingness to use it. We have also found that perceived usefulness has a positive correlation with willingness to use. However, easiness to use is not correlated to willingness to use nor usefulness.

We have also discussed the problems that the subjects faced when designing with ADD. These problems concern: 1) Team Workload Division and Assignment, 2) No consensus in terminology, 3) The first iteration of ADD, and 4) Mapping Quality Attributes to Tactics, and Tactics to Patterns. These problems could have affected the subjects' opinion in determining their neutral answers to perceived easiness and their slight agreement with the usefulness of ADD.

In the near future, we would like to replicate this study in order to increase our sample, and improve our questionnaire. We are also planning to replicate this study in an industrial setting in order to take into account more experienced architects, and compare their perception with the results of the students.

One of our objectives is to research whether our findings in ADD can be generalized to other software architecture design methods. In this way, we can compare and contrast the different methods from software architects perspective. In addition, we would like to make studies to understand who are the right people which hold the appropriate skills for carrying out ADD. This is important to be known in the software architecture community in order to improve the training and further research in the field of design methods.

References

[1] Bachmann, F., Bass, L., Klein, M., Deriving Architectural. Tactics: A Step Toward. Methodical Architectural. Design. CMU/SEI-2003-TR-004. ESC-TR-2003-004, 2003.

[2] Bachmann, F., Bass, L., Klein, M., & Shelton, C. *Designing software architectures to achieve quality attribute requirements*. Software IEE, Vol. 152 Issue 4 - pp. 153-165.

[3] Bardram, J.E., Christensen, H.B., Corry, A. V., Hansen, K.M., and Ingstrup, M., "Exploring Quality Attributes using Architectural Prototyping," in *Proc. of First International Conference on the Quality of Software Architectures*, LNCS, vol. 3712, September 2005, pp. 155-170.

[4] Bass, L., Clements, P., and Kazman, R., Software Architecture in Practice (2nd Edition), 2nd ed. Addison-Wesley Professional, April 2003.

[5] Christensen, H. B., Hansen, K. M., An empirical investigation of architectural prototyping. Journal of Systems and Software 83(1): 133-142 (2010).

[6] Davis, F. D. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*, 13(3), 1989, pp. 319-340.

[7] Diaz-Pace, A., Kim, H., Bass, L., Bianco, P., and Bachmann, F. 2008. Integrating Quality-Attribute Reasoning Frameworks in the ArchE Design Assistant. In *Proc. of the 4th international Conference on Quality of Software-Architectures: Models and Architectures* LNCS vol. 5281, pp. 171-188.

[8] Dulva Hina, M., Tadj, C.,K., Ramdane-Cherif, A., "Attribute-Driven Design of Incremental Learning Component of a Ubiquitous Multimodal Multimedia Computing System", *Canadian Conference on Electrical and Computer Engineering,* pp.323-327, May 2006.

[9] Eikebrokk, T.R., Iden, J., Olsen, D.H., Opdahl, A.L., "Determinants to the Use of Business Process Modeling," *43rd Hawaii International Conference on System Sciences*, pp.1-10, 5-8 Jan. 2010.

[10] Falessi, D., Cantone, G., Kruchten, P., "Do Architecture Design Methods Meet Architects' Needs?," Sixth Working *IEEE/IFIP Conference on Software Architecture (WICSA'07), pp. 44-5*3, 2007.

[11] Ferrari, R., and Madhavji, N.,"Software architecting without requirements knowledge and experience: What are the repercussions?," J. Syst. Softw., vol. 81, no. 9, pp. 1470-1490, 2008.

[12] M. Fishbein, I. Ajzen, "Belief, Attitude, Intentions and Behavior: An Introduction to Theory and Research", Addison-Wesley, Boston, 1975.

[13] George, D., Mallery, P., "SPSS for Windows Step by Step: A Simple Study Guide and Reference", 17.0 Update (10th Edition). Allyn & Bacon. 2009.

[14] Gorton, I., "Essential Software Architecture", Springer-Verlang, 2006.

[15] Gross, D., and Yu, E., "From non-functional requirements to design through patterns," *Requirements Engineering*, vol. 6, no. 1, pp. 18-36, February 2001.

[16] Habli, I. and Kelly, T. "Capturing and Replaying Architectural Knowledge through Derivational Analogy", In Proc. of the Second Workshop on Sharing and Reusing Architectural Knowledge Architecture, Rationale, and Design intent, 2007.

[17] Harrison, N. B., and Avgeriou, P., "Leveraging Architecture Patterns to Satisfy Quality Attributes", European Conference on Software Architecture, LNCS, pp. 263-270, 2007.

[18] Harrison, N. B., and Avgeriou, P., "How do architecture patterns and tactics interact? A model and annotation", *Journal of Syst. Softw.*(Article in Press).

[19] Hofmeister, C., Kruchten, P., Nord, R. L., Obbink, H., Ran, A., and America, P. A general model of software architecture design derived from five industrial approaches. *J. Syst. Softw.* 80(1), 2007, 106-126

[20] Hofmeister, C., Nord, R. L. & Soni, D. 2005. Global analysis: Moving from software requirements specification to structural views of the software architecture. IEE Proceedings Software 152(4), 187–197.

[21] Jamieson, S., "Likert scales: how to (ab)use them.", *Medical education*, vol. 38, no. 12, pp. 1217-1218, December 2004. [Online]. Available: http://dx.doi.org/10.1111/j.1365-2929.2004.02012.

[22] Kannengiesser, U. and Zhu, L. 2009. An ontologically-based evaluation of software design methods. *Knowl. Eng. Rev.* 24, 1 (Dec. 2009), 41-58.

[23] Kazman, R., Bass, L., and Klein, M., "The essential components of software architecture design and analysis," Journal of Systems and Software, vol. 79, no. 8, pp. 1207-1216, 2006.

[24] Kitchenham, B., and Pfleeger B.A., "Personal Opinion Surveys", In Shull, F., Singer, J., and Sjøberg (Eds.), *Guide to Advanced Empirical Software Engineering,* Springer-Verlag, 63-92, 2007.

[25] Kruchten, P. 2004. The Rational Unified Process: An Introduction. Addison-Wesley.

[26] Laitenberger, O.; Dreyer, H.M., "Evaluating the usefulness and the ease of use of a Web-based inspection data collection tool", *Fifth International Software Metrics Symposium,* 1998, pp.122-132.

[27] Lee, J., Shin, G.S., "Quality Attribute Driven Architecture Design and Evaluation for Embedded Operating System," *Advanced Communication Technology, 2008,* pp. 367-371.

[28] Lindsay, R. M., Ehrenberg, A.S.C, "The Design of Replicated Studies", The American Statistician, Vol. 47, N0. 3(Aug, 1993), pp. 217-228.

[29] Mitchell M.I. Research Design Explained. Fifth edition, Thomson-Wadsworth, 2004.

[30] Nord, R., McHale, J., Bachman, F., "Combining Architecture-Centric Engineering with the Team Software Process", Technical Report CMU/SEI-2010-TR-031, December 2010

[31] Remco, C., Van Vliet, H., "QuOnt: an ontology for the reuse of quality criteria." ICSE Workshop on Sharing and Reusing Architectural Knowledge, pp. 57-64, 2009.

[32] Shaw, M., and Garlan, D., Software architecture Perspectives on emerging discipline, Prentice Hall, 1996.

[33] Software Engineering Institute. Software Architecture Glossary http://www.sei.cmu.edu/architecture/start/glossary/ (2010)

[34] Software Engineering Institute. Architecture Training http://www.sei.cmu.edu/architecture/start/training/index.cfm (2010).

[35] Stevens, S.S., On the Theory of Scales of Measurement, Science, New Series, Vol. 103, No. 2684 (Jun. 7, 1946), pp. 677-680.

[36] Taylor, R., Medvidovic, N., Dashofy, E., Software Architecture, Foundations, Theory, and Practice, Wiley Publisher, 2010.

[37] Toral, S. L., Barrero, F., and Martínez-Torres, M. R. Analysis of utility and use of a web-based tool for digital signal processing teaching by means of a technological acceptance model. *Comput. Educ.* 49, 4 (Dec. 2007), 957-975.

[38] Wojcik, R.; Bachmann, F.; Bass, L.; Clements, P.; Merson, P.; Nord, R.; & Wood, B., "Attribute-Driven Design (ADD), Version 2.0", Technical Report CMU/SEI-2006-TR-023, SEI, 2006.

[39] Wood, W. G., "A practical Example of Applying attribute-Driven Design (ADD), Version 2.0", Technical Report CMU/SEI 2007-TR-005, SEI, 2007.