Domain Specific Modeling Support for ArCon

by

Leila Azari

LIU-ID/LITH-EX-A--13/064—SE
2013-12-01
Institutionen för datavetenskap
Department of Computer and Information Science

Master thesis

Domain Specific Modeling Support for ArCon

by

Leila Azari

LIU-IDA/LITH-EX-A--13/064—SE
2013-12-01

Supervisor: Gert Johansson (Combitech AB)
Olena Rogovchenko (IDA)

Examiner: Kristian Sandahl
“If in physics there's something you don't understand, you can always hide behind the uncharted depths of nature. You can always blame God. You didn't make it so complex yourself. But if your program doesn't work, there is no one to hide behind. You cannot hide behind an obstinate nature. If it doesn't work, you've messed up.”

-Edsger W. Dijkstra
Abstract

One important phase in software development process is to create a design model of the system which follows all the architectural rules. Often the architectural rules are defined by the system architect and the system model is designed by the system designer. The architect defines the rules in a text file where no standard or pattern is followed. Therefore, there is always the risk of violating the architectural rules by the designer. So manual reviews on the system model should be done by the architect to ensure the system model is valid.

In order to remove this manual checking which can be erroneous and time consuming ArCon (Architecture Conformance Checker) was developed by Combitech AB. ArCon is a tool which lets the architect define the architectural rules in the format of UML (Unified Modeling Language) models where the elements of the model have different meaning than the standard UML. ArCon can read this model and extract architectural rules from it and check the system model against those rules and then print all the rule violations.

ArCon is an open source tool i.e. free for everyone to download and use. Currently, it supports Papyrus as the UML modeling tool. Papyrus is integrated to Eclipse platform and is a general purpose modeling tool. It supports users with all types of UML diagrams and elements.

The idea for this thesis work was to implement a new feature for ArCon in order to facilitate the design process for system designers. The feature should provide the system designers only those types of elements which they are permitted to add to a specific fraction of the system model. The list of permitted element types should be extracted from the architecture model where all the architectural rules are defined in advance. This new support in ArCon was named Domain Specific Modeling (DSM) support.

To evaluate the effect of DSM support on the system designers performance a few test sessions, called usability tests, were performed. The participants in the test sessions were a representative sample of software designers. After analyzing the data collected from the test sessions, the pros and cons of the new support were discovered. Furthermore, a few new ideas for enhancing DSM support were generated.
Acknowledgments

First of all, I would like to express my sincere appreciation to my supervisor, Gert Johansson, for his support and guidance throughout this entire master thesis. Furthermore, I would like to thank Johan Norberg for introducing me to the topic as well for his kind support on the way.

I would also like to take this opportunity to express my gratitude to all the staff members at Combitech office in Linköping. Particular thanks go to Johnny Larsson for the opportunity and his generous support on many respects.

Last but not least, I am forever grateful to my parents and my sister for their love, support and encouragement through my entire life.
List of abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSML</td>
<td>Domain Specific Modeling language</td>
</tr>
<tr>
<td>ArCon</td>
<td>Architectural Conformance checker</td>
</tr>
<tr>
<td>IDEs</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>EMF</td>
<td>Eclipse Modeling Framework</td>
</tr>
<tr>
<td>GMF</td>
<td>Graphical Modeling Framework</td>
</tr>
<tr>
<td>EMP</td>
<td>Eclipse Modeling Project</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language</td>
</tr>
<tr>
<td>DSL</td>
<td>Domain Specific Languages</td>
</tr>
<tr>
<td>MDD</td>
<td>Model-Driven Development</td>
</tr>
<tr>
<td>OMG</td>
<td>Object Management Group</td>
</tr>
<tr>
<td>MDA</td>
<td>Model-Driven Architecture</td>
</tr>
<tr>
<td>MDT</td>
<td>Model Development Tools</td>
</tr>
<tr>
<td>GPLs</td>
<td>General Purpose Languages</td>
</tr>
<tr>
<td>HCI</td>
<td>Human/Computer Interfaces</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>CTA</td>
<td>Concurrent Think-Aloud</td>
</tr>
<tr>
<td>RTA</td>
<td>Retrospective Think-Aloud</td>
</tr>
</tbody>
</table>
Think aloud protocol ................................................................. 25
5.1 Why Think-aloud matters for usability evaluation .................................................. 25
5.2 Think-aloud advantages ......................................................................................... 25
5.3 Think-aloud disadvantages ..................................................................................... 26
5.4 Making both variations more efficient ................................................................. 26
5.5 Research questions ............................................................................................... 27
5.6 Some available researches on comparing the two variations .................................. 27
5.7 Enhancing concurrent think-aloud approach ......................................................... 29
5.8 Combination of the two methods .......................................................................... 30
5.9 Factors influencing verbalization problems .......................................................... 31
  5.9.1 The type of task studied ..................................................................................... 31
  5.9.2 The type of instructions given for verbalization ................................................. 31
5.10 Conclusion ............................................................................................................ 32
Chapter 6................................................................................................................. 33
Test Plan....................................................................................................................... 33
  6.1 Study Goal ............................................................................................................. 33
  6.2 Variables ............................................................................................................... 34
  6.3 Within-subject or between-subject study ............................................................. 35
  6.4 Method selection ................................................................................................. 35
  6.5 Participant selection ............................................................................................. 36
  6.6 Test Location and Environment ......................................................................... 37
  6.7 Metric Selection .................................................................................................... 38
  6.8 Task Definition and Assignment ......................................................................... 38
    6.8.1 Counterbalancing .......................................................................................... 39
    6.8.2 Training sessions ......................................................................................... 39
  6.9 Test Plan Table ..................................................................................................... 40
  6.10 Pilot test ............................................................................................................... 41
Chapter 7..................................................................................................................... 42
Experimental Results .................................................................................................. 42
  7.1 Experimental setup and procedure ....................................................................... 42
  7.2 Analyzing the data ............................................................................................... 44
  7.3 Participant experiences .......................................................................................... 44
    7.3.1 Effort and complexity .................................................................................... 45
    7.3.2 Error rate ....................................................................................................... 47
    7.3.3 Reliability ....................................................................................................... 48
    7.3.4 User satisfaction ............................................................................................ 50
  7.4 Questionnaires ..................................................................................................... 51
    7.4.1 Effort and complexity .................................................................................... 52
    7.4.2 Reliability ....................................................................................................... 53
    7.4.3 User satisfaction ............................................................................................ 53
    7.4.4 Usefulness ...................................................................................................... 54
    7.4.5 Participants suggestions for ArCon enhancement .......................................... 55
Chapter 8..................................................................................................................... 56
  8.1 Conclusion ............................................................................................................. 56
  8.2 Further research and recommendation .............................................................. 57
List of Figures

Figure 3.1. Overview of the code design .................................................................................. 11
Figure 3.2. Sequence diagram for DSM support on popup menu ........................................ 13
Figure 4.1. Bailey’s Human Performance Model .................................................................. 15
Figure 4.2. Usability testing through the product lifecycle .................................................. 17
Figure 7.1. Task complexity ............................................................................................... 46
Figure 7.2. Error rate ......................................................................................................... 48
Figure 7.3. DSM support reliability .................................................................................... 49
Figure 7.4. User satisfaction of DSM support .................................................................... 50
Figure 7.5. Participants rating on task complexity .............................................................. 52
Figure 7.6. Participants rating on DSM support reliability ................................................ 53
Figure 7.7. Participants rating on satisfaction of the tool .................................................... 54
Figure 7.8. Participants rating on ArCon’s usefulness ........................................................ 54
Figure 0.1 Task A-Diagram a ............................................................................................. 58
Figure 0.2. Task A-Diagram b ........................................................................................... 59
Figure 0.3. Task A-Diagram c .......................................................................................... 59
Figure 0.4. Task A-Diagram d ......................................................................................... 60
Figure 0.5. Task B-Diagram a .......................................................................................... 61
Figure 0.6. Task B-Diagram b .......................................................................................... 62
Figure 0.7. Task B-Diagram c .......................................................................................... 62
Figure 0.8. Task B-Diagram d .......................................................................................... 63
Figure 0.9. Task B-Diagram e .......................................................................................... 63
Figure 0.10. Assigning dependency to an architecture model in a system model .............. 69
Figure 0.11. An example of architecture model in ArCon .................................................. 70
Figure 0.12. An example of a system model in ArCon following the arch model represented in figure 8.11 ............................................................................................................. 71
Figure 0.13. Popup menu supported by Papyrus ................................................................. 72
Figure 0.14. Popup menu supported by ArCon DSM support ............................................ 73
Figure 0.15. A comparison between Papyrus and ArCon Palettes .................................... 74
List of tables

Table 6.1. Minimum requirements for different tester groups ................................................................. 36
Table 6.2. Requirements for each participant type .................................................................................. 37
Table 6.3. Distribution of the participants ............................................................................................... 37
Table 6.4. The order of tasks assigned to the subjects ............................................................................ 39
Table 6.5. Test plan .................................................................................................................................. 40
Table 7.1. The amount of effort required for tasks A and B .................................................................. 43
Table 7.2. Three scopes which caused user confusion or difficulty during the test sessions .......... 45
Table 7.3. Cases in which the participants made a mistake during the design process .................... 47
Table 7.4. Cases which represents tool reliability .................................................................................... 48
Table 7.5. Verbalizations stating user satisfaction ................................................................................... 50
Table 8.1. Post task questionnaire ........................................................................................................ 64
Table 8.2. Post session questionnaire .................................................................................................... 65
Chapter 1

Introduction

This chapter presents an introduction to the thesis work including the study purpose and the description of the problem. In addition, the disposition of the report is described.

1.1 Background

The purpose of Model-Driven Development (MDD) is to represent design information in a set of semi-formal or formal models to allow automation supported by tools. The goal is to avoid error-prone and time-consuming manual work in order to maintain consistency between design artefacts. One of the important design artefacts that guide system design so that it meets the desired system quality is the software architecture model [Bass et al. 2003].

The architecture model in MDD is defined in the form of packages and components designed in a high level structure. But generally the architectural design rules and constraints which should be followed in design process are defined by architect in an informal text file. Therefore, manual reviews for the final system designs are always required. The manual checking causes low review quality, longer software life cycle and high risk of making a mistake by the reviewer. In addition, using unstructured text files for architectural rules has become a problem for MDD because MDD relies on formal models to be capable of providing automation.

Mattsson, 2009, proposed an approach to model the architectural design rules in an abstraction level in the form of an Unified Modeling Language (UML) model, where the UML constructs have a different meaning than the standard definition. As a proof-of-concept, ArCon (Architectural Conformance checker) tool was developed by Combitech AB to validate any system model against a set of architectural rules which are defined in an UML model (Combitech, 2013).

ArCon reads the models from an UML tool and checks a system model against a set of architectural rules defined in another UML model (meta-model) called Architectural Rules Model. Finally, it produces a report where all the violations are presented. Combitech selected Papyrus (Papyrus Project) as the modeling tool which is an open source tool in Eclipse and supports UML models.
1.2 Study purpose

Combitech AB is now working to further develop ArCon to add more functionality and enhance its strengths. My master thesis was a contribution to this goal. The scope of my work was to provide a domain specific modeling support, called DSM support, based on the meta-model that ArCon uses for specification of architectural rules.

The requirement was to provide a support related to the element type while adding elements to the system model. The support should follow the rules defined on a specific domain which was represented through the architectural rules. To be more specific, this functionality offers the list of permitted element types which the designer is permitted to add to a specific fraction in the system model. The permitted element type list should be extracted from the meta-model which ArCon should use as a mean for extracting architectural rules.

As a result, it was expected that the development process should become easier, faster and also many mistakes should be avoided if designers use this DSM support. The mistakes which developers could make in the absence of this support could be because of either the lack of knowledge of how to follow an architectural model defined under ArCon’s constraints, or getting confused with the huge and complicated architectural model(s).

In order to measure how the new functionality in ArCon improved system designer’s performance and facilitated design process, a usability study was performed. The results are presented in chapter 7.

1.3 Intended audience

This report is intended for software developers who have knowledge in modeling, UML and software design. As a reminder, a brief introduction to the required technologies is given in chapter 2.

1.4 Report overview

This section gives a quick overview to the content of each chapter and describes how the report is constructed.

- Chapter 2 – Technologies
  The second chapter gives a brief description of the tools, standards and concepts that were used in this thesis project. The purpose is to give a brief introduction to those audiences who are unfamiliar with the mentioned concepts.

- Chapter 3 – Domain specific modeling support
  The third chapter explains in details the problem and its domain and the motivation of the thesis work. The solution for the problem is presented by a short explanation about the design and implementation approach. Finally, a user manual on how to use the domain specific modeling support in ArCon is provided.
• **Chapter 4 – Theoretical background**
  The fourth chapter deals with usability evaluation concept, its purpose and also the requirements for conducting a usability test and analysing the results.

• **Chapter 5 – Think-aloud method**
  The fifth chapter presents roughly a detailed study on think-aloud protocol. The chapter is to explain why think-aloud matters for usability experts and what are its cons and pros. Generally, it provides a comparison of two varieties of think-aloud protocol: concurrent and retrospective think aloud. In addition, some case studies are presented as examples.

• **Chapter 6 – Test plan**
  The sixth chapter goes over the test plan and preparations I made before conducting the usability tests. It describes all preparations including variables in the study, the method selected for performing the usability experiment, participant selection, task definition and all the other preparations which should be determined for the test plan.

• **Chapter 7 – Experimental results**
  This chapter discusses the results collected from the usability experiment planned in chapter 6.

• **Chapter 8 - Conclusions**
  The last chapter summarizes the thesis work and the conclusions of the usability experiments. It also provides some recommendations for possible improvements in the future.
Chapter 2

Technologies

This chapter gives a short description about the technologies and standards which are related to this thesis work. The target readers are those who are unfamiliar with the mentioned topics in this chapter.

2.1 Domain specific language

Domain Specific Languages (DSL) play an important role in enhancing design productivity by decreasing the amount of effort for maintenance and development (Ankica Barišić, 2011). Therefore, they have been used increasingly in Model-Driven Developments (MDD) technologies. They support developers to use abstractions that are part of the real domain world. It results in increasing the productivity compared to when the developers use General Purpose Languages (GPLs). DSLs are designed for a specific context, domain or company, to facilitate performing tasks for users (Ankica Barišić, 2011). DSL frameworks have been added to IDEs (Integrated Development Environment) such as EMF (Eclipse Modeling Framework) and GMF (Graphical Modeling Framework) in Eclipse Modeling Project (EMP). A well-known example of a Domain Specific Modeling language (DSML) is UML when it extends elements with tagged values and stereo types.

As mentioned before, the purpose of the DSL is to increase productivity. This is reached by means of improving the abstraction level beyond current languages and also providing the solution using the concepts of the domain. A domain can be user interface, communication, banking and robot control. Narrower focus on smaller domains enables more automation and effectiveness.

Regarding to the purpose of the DSLs, they can be considered as Human/Computer Interfaces (HCI) (Ankica Barišić, 2011). Therefore, they shouldn’t cause quantitative problems such as long time learning for the users, user inconvenience and dissatisfaction. Usability is a quantitative software characteristic which is a key factor for any DSL adoption.
2.2 UML

Generally, modeling languages in software engineering describe models composed of diagrams and symbols. Each model describes different aspects or the constraints of either the problem or solution. Its purpose is to increase the productivity.

UML is a general purpose modeling language proposed by Object Management Group (OMG) within Model-Driven Architecture (MDA) framework. It comprises a wide set of different diagrams and symbols and supports mostly object-oriented programming languages. In other words, it is considered as a family of languages which is tailored to software systems specification. By applying profiles, one can apply domain constraints and create a domain specific language in UML, such as ArCon. Therefore, some standard languages have been extended based on UML such as SysML. SysML is a UML profile for system engineering which is considered as a DSL. Although, UML cannot be applied to some domains, such as user interaction design, but it can be easily used and has increased the productivity especially by help of profiles (Marco Brambilla, 2012). For more details please see UML homepage (UML).

2.3 Eclipse

Eclipse is a not-for-profit community for organizations who want to develop open source software. It is an open source project which was created by IBM in 2001. In order to both increase the adoption of Eclipse in open source projects and support Eclipse community, Eclipse foundation was established.

Eclipse offers a multi-language, open source development platform which comprise of a number of tools, frameworks and runtimes to support developers in developing and managing software along project lifecycles. Eclipse is extensible and customizable through Plugins. Plugins are used to add functionality on top of runtime systems.

2.4 Papyrus

Papyrus is an extension in the Eclipse platform. It is a component of the Model Development Tools (MDT) and can be used as a standalone tool. It offers an environment for editing any kind of EMF model especially in the form of UML2 (as defined by OMG), MARTE and SysML.

Papyrus enables support for DSL by providing UML profiles and Static SysML profiles and specific editors used by SysML. It also allows customization of different parts of it such as model explorer, property editors and diagram editors, etc. This powerful customization mechanism is a key feature in Papyrus which enables users to create their Papyrus perspectives in order to create the same look but as a domain specific language editor.

There are a number of extension points in Papyrus which allow customization. In chapter 3 I will explain in more details how I used some of them to be able to implement DSM support. On Papyrus webpage, these extension points are mentioned but no document for guiding the developers has been uploaded yet.
So, a long time and a lot of effort was invested in this thesis in order to find the right libraries and extension points.

2.5 ArCon

In Model-Driven Development (MDD) all design information is captured in a set of formal or semi-formal models where tools are used to automatically keep them consistent. The goal is to increase abstraction level and omit manual work for maintaining consistency between design artefacts. Traditionally, the architectural model of a system is presented in a form of UML model. This model is a high level and abstract model. Therefore, it is supported by a rule set which includes constraints and it is often explained informally in text format. So, manual checks have to be done to ensure that all constraints and rules have been followed in the detailed design. This step is a bottleneck in MDD where all other activities in design the process have been automated. It is time consuming and designers need to wait a long time for architecture’s approval to be able to move to the next step. These conditions can also lead to low quality and maybe erroneous reviews by the architect.

The basic idea for ArCon was to provide a method to present all the architectural design rules in a UML model as a meta-model and in an abstraction level where UML constructs have been rendered a different meaning than what is prescribed by the standard. So the rules can be easier to understand for both developers and designers and also the system design can be verified automatically. This idea was proposed by Anders Mattsson at Combitech AB.

In order to verify that Mattson’s approach worked correctly, ArCon was developed to implement the method and provide the intended automation. It reads the models from an UML tool and checks a System Model against a set of architectural rules defined in another UML (meta-) model, called architecture model. Finally, it produces a report where all violations are presented.

ArCon was first developed for IBM Rational Rhapsody. Rhapsody provides a collaborative design, development and testing environment for software engineers and system engineers. But later, Combitech selected Papyrus, an open source modeling tool which supports UML, as the tool to be supported by ArCon. Now, the company is working to further develop ArCon to enhance its functionalities and quality. My master thesis was a contribution to this goal. All documents and source code for ArCon is available at (Combitech, 2013). The last version of ArCon, including my work, is adapted to Papyrus in Eclipse Kepler.
Chapter 3

Domain Specific modeling support

In this chapter, I will explain the study motivation, customer specification and give an overall view of the code implementation.

3.1 Problem solving

Problem solving is applying our knowledge in order to find a solution to an unfamiliar task (Alan Dix, 2003). Simon and Newell proposed a theory called problem space theory. They found problem solving very dependent on the problem space. Based on this theory, a problem has an initial state and goal state and problem solving is to use operators to move step by step from the initial state to the goal state. However, often the problem space is huge or complex, so heuristics are applied to move correctly toward the goal state.

Regarding their model, problem solving is applied in a well-defined domain. In reality, the knowledge on the domain which is required to solve the problem, could be a part of problem or sometimes determining the goal can be difficult. These types of problems require significant knowledge and skills about the domain. For example, to solve a programming language problem one needs to have a good knowledge about the programming language and the domain the program operates in.

A well-known example of problem solving in domain specific space is the chess game. The start state is opening board position and some legal moves (operations) will be done which move the player through different states toward the goal state. Some studies have been done by Chase and Simon on this scope. The purpose of their study was to compare behaviour of chess master and less skilled players and finally they got interesting results (Alan Dix, 2003). They observed that the number of moves they took didn’t differ that much but the time for making a decision was considerably different. In fact, experience and knowledge of the expert results in an efficient and accurate solution.

The same conditions are held for system designers when they use papyrus or other UML modeling language tools where they must follow both UML and architectural rules. Their initial state is an empty model and their target state is a system model which follows the constraints, indicating the domain of the system, introduced by software architect. So, creating a system model can be hard for the designers.
The idea of this thesis is to provide a support for them based on the system domain in order to increase their productivity and satisfaction when they use ArCon.

3.2 The problem

For any kind of diagram or any type of element in a diagram, Papyrus proposes a list of elements on the palette or on the pop-up menu in the model explorer view. Since Papyrus cannot be very specific to a special domain or problem scope, users are always faced with a very long list of nodes and edges proposed by Papyrus. So, the list always contains many elements which are not required for the domain in which the problem is defined.

On the other hand, in the case that the model is a system model, the designers often don’t have enough knowledge about the architectural rules since they are defined by architect. Although they can run ArCon anytime and check whether they broke any rules, the hardship of the problem is finding the right steps they should take while creating a system model. So, if the designers could get any hint about the architectural rules while creating a system model the process of solving the problem would be easier and fewer errors would be made by the designers.

The idea for this thesis work was to provide a domain specific modeling support. In the case that end-users assign an architectural model to their system model, they should see a filtered list of element types on GUI. Considering the architectural constraints, the proposed element types should be the ones which are allowed to be added to the selected element on the diagram editor. This makes the development process easier, faster and also can help to avoid mistakes which developers could make on deciding on the correct type of element.

Since the mentioned support concerns with human computer interaction, it has to supply user satisfaction and convenience. Therefore, a usability study was done to evaluate some quantitative characteristics of both new interface implemented by ArCon and Papyrus standard interface. The goal was to compare the two interfaces and assess how this thesis work could make improvement in ArCon in term of usability. The background theory study is presented in chapter 4 and 5. Preparation steps for the experiment and the test plan are presented in chapter 6 and the results of the experiment is discussed in chapter 7.

3.3 User specification

As mentioned in the previous section, two enhancements in the user interface are demanded in order to provide a Domain Specific Modeling (DSM) support for system designers. User specification is presented in the form of acceptance test cases as listed below. These modifications are only needed when the user work on a system model which is assigned to an architecture model.
3.3.1 Palette display modification

The requirement is to filter the items on the palette. It means that instead of showing a long menu of nodes and edges, only the ones which are defined in the referenced architectural model should be visible to the user. The filtering on palette items happens if:

- The user opens a system model.
- The user changes the selection on the diagram editor.

3.3.2 An extension to the right click menu on model explorer

User right clicks on an element in the model explorer view. If ArCon recognizes the container model as a system model and if a dependency to an architectural model is set for it, the user should then see one additional title for “new child” option on the right click menu. The title for the new menu item is called “New ArCon child”. The new menu item should only list the elements (nodes and edges) which are permitted for the selected element following the architectural rules.

Please note that some snapshots and examples are provided in the user manual. The user manual is included in appendix D.

3.4 Interaction style

The dialog between computer and human can be seen as interaction. There are different types of elements that make up interactive applications. Most common interface elements are command lines, menus, form-fills, spread sheets and so forth. Considering the problem scope in this master thesis, pop-up menu and palette were the target components.

3.4.1 Pop-up menu

Menus are one of the main features of window-based formatted interfaces. They provide a list of operations that can be performed on a specific context in a special time. Menu options need to be well named, logically grouped and meaningful to help recognition in order to enable users to find what they are looking for. In addition menus are inefficient when they have a long command list.

Pop-up menus, also known as context menus, are the subset of menus and appear when a user right clicks on an object in UI. They often present state- or context-sensitive options. Although in some cases they are used to access more global actions, they often propose options related to the selected object.

Some usability analysts criticize context menus, because they can confuse even the experienced users when it must only be activated in a limited part of UI. Furthermore, the usability decreases when the command list on the menu becomes very long. Although, Windows and Mac have different implementation styles for menu, in the book [Macintosh Human Interface Guidelines, p87], it is explicitly expressed that “don't use pop-up menus for commands”.

9
3.4.2 Palette

A palette is a set of elements for choosing from a limited number of choices. It can be designed as a programmer’s guide, a user’s manual, an engineer’s technical specification, etc. A GIU palette can be either a tool palette or colour palette. A palette is a window with icons, buttons or other GUI controls which provide a quick access to symbols or commands.

In some applications interaction can enter several modes and it is hard or even impossible for users to know which mode is active. Tool palettes are a collection of icons that are such a reminder for the purpose of different modes.

3.5 Implementation approach

Since the thesis work was a part of the ArCon project ordered by Ericsson AB, I worked in a team with two other software developers. The team followed scrum methodology and the tasks were divided into a few numbers of sprints.

For this project Eclipse Juno was used. But the targeted platform was Eclipse Kepler which was released in June this year. Therefore after finishing development, the code was adapted to Kepler. The software will be developed in Java and is to be integrated in Eclipse using the plug-in concept.

3.6 Implementation challenges

To be able to implement the desired requirement, this project’s tasks were very dependent on Papyrus library and also the extension points through which developers can customize Papyrus user interface. Therefore, one of the main challenges in this project was to find the right extension points and APIs that Papyrus provides as well as finding the way they should be applied in the plug-in which I created.

On the other hand, in some cases Papyrus uses Eclipse API and extensions. So, developers need to use those APIs as well in order to make any modification. Since I couldn’t find any good documentation for Papyrus developers and unfortunately my questions on the forum didn’t receive any clear answer, I had to go through the source code and try to explore and find the solution. Due to complexity of Papyrus libraries and considering that it is a fairly huge software, I had to spend too much time on the exploration and code tracing. My lack of knowledge about Eclipse APIs and environment can be another reason for taking long time to accomplish the result.

3.7 Design and implementation

An overview of the plugin I implemented is presented in the form of class diagram in figure 1.1. The whole project is composed of a few packages and classes cooperating together. The whole project is specified as a package called ArCon in the diagram. The whole project has dependencies to three external libraries belonging to Eclipse and Papyrus.
Model listener class is an extension of “org.eclipse.papyrus.infra.core.model” extension point. It is used in this project to be responsible for event handling. The extension of this extension point should implement an interface called org.eclipse.papyrus.infra.core.resource.IModel. Therefore, the methods listed below should be implemented for that class:

- public void loadModel(IPath path) {} // is called when a model is opened
- public void saveModel() throws IOException {}// is called when the modifications on the model are saved
- public void unload() {} // is called when the model is closed
Any time a model is opened in Papyrus environment, this class in my code catches the event. If the opened model is assigned to an architecture model (please see appendix D in order to find out how a system model is assigned to an arch model) this class sends the referenced architecture model as an input to a class in ModelTreeStructure package and ask for a new object of type Arch_PermittedElements as output.

Arch_PermittedElements class traverses the arch model and stores the required data in the format of tree structure. Therefore, an element of type Arch_PermittedElements contains a tree structure. To create the tree, for each element in the arch model, it adds a node to the tree structure containing its stereotype, type and a few other properties. Then it traverses that element and retrieves the sub-elements it contains. For each sub-element it adds a child node to the corresponding node. This process is called recursively until the whole model is traversed and the tree is created completely.

Model listener receives the created object of type Arch_PermittedElements and adds it to a list, called OpenedFile. OpenedFile is a global static hashmap object. It keeps track of all opened system models and the key to retrieve an entry from it is considered as the full path to the corresponding model. To sum up, whenever a new model is opened, if it has a dependency to an arch model, an object of type Arch_PermittedElements will be requested by ModelListener and then will be stored in OpenedFile. On the other hand, whenever the opened model is closed, its path will be searched in the hashmap and if an entry will be found, the ModelListener will drop it from the list.

PalettePopupListManager class is the responsible class for creating the permitted element list for the selected element. A listener called SelectionListener is defined in ModelListener class. It listens to the diagram editor and if the selection is changed it calls a method named prepareListToShow() from PalettePopupListManager class. This method retrieves the corresponding object from OpenedFile and traverses the tree structure in it, finds the corresponding node for the selected element and collects all types of sub-elements for it. The key to find the right node in the tree is the stereotype name assigned to the selected element.

After finalizing the list of permitted elements, the elements on ArCon palette will be refreshed by calling a method from PaletteManager class. The same scenario happens if the user selects an element on the model explorer view and then right clicks on it. The only difference is that in this case ArConMenuCustomization class takes the responsibility for creating ArCon menu and its submenus.

The interactions between my code, called ArCon, a system model defined as a Papyrus model, the model explorer view and an end user is presented as a sequence diagram in figure 1.2.

In order to create a menu and its submenus, one extension point of Eclipse, named org.eclipse.ui.menus, was used. In this extension point the class which I defined (ArConMenuCustomization) inherited from a class called ExtensionContributionFactory. To manage the palette, an extension point provided by Papyrus was applied. The extension point is called org.eclipse.papyrus.uml.diagram.common.paletteDefinition was extended where the class org.eclipse.papyrus.uml.diagram.common.service.PluginPaletteProvider was extended.
3.8 Performance and memory usage

The final code was tested by the biggest available sample of an arch model at Combitech AB. The processing time for each selection changed event was roughly 0.03 seconds and the memory usage didn’t seem to be considerable, because the tree structure is compact and the data it keeps is only in the format of strings. Therefore, it is expected that for even bigger arch models there will be no risk of memory overflow or a considerable memory overhead. In fact, the memory this tree structure requires is too less compared to the real size of the papyrus model.
Chapter 4

Theoretical background

The human user, is the one for whom the software is designed to assist. Therefore, in order to design something we need to know the target user’s limitations, capabilities and tasks which are hard or even impossible for them to do. The purpose is to facilitate the way people solve the problems and accomplish their tasks. Since the user interface is what end-users interact with, evaluating the interface design is a key point to discover any problem which users encounter while performing their tasks. Therefore, usability tests are designed to evaluate and disclose the design problems.

This chapter gives an introduction to the usability test, its value, usability metrics and the principles for selecting the right evaluation method. Finally, a description about how to develop a test plan and conduct test sessions is discussed.

4.1 Purpose and value of the usability testing

It is important to develop products with a high-quality User Interface (UI), because UI is a part of the software that the customers experience. Therefore, UI has an effective impact on the product success. Developing a good user interface should concern users, their requirements and problems and simplicity of the Graphical User Interface (GUI) (Johnson, 2007).

Testing is conducted to follow a specific objective. The purpose for every test and desired degree of result precision is often stated explicitly in the early steps of testing plan. Any test can be aimed at verifying both functional and non-functional properties. Non-functional properties include performance, usability, reliability etc.

Usability testing concerns with how easy it is for users to learn and use the software and how effectively the software works to support user tasks and help users to recover from their errors. The International Standard Organization (ISO 9241-11) defined usability as “the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use”.

It is suggested to do early usability tests in order to be able to have early correction in design. Skipping usability tests doesn’t really save money because the software actually has to be tested in the marketplace if it would not be tested by developers. Therefore, more than what the company saves by skipping the tests will be lost due to low-level design. Furthermore, it is not necessary to plan for an expensive usability test since very low-cost tests can resolve design issues (Johnson, 2007)

4.2 What makes something less usable?

Baily RW proposed three factors, shown in figure 3-1, in order to achieve usability in computer systems. Underlying abilities and skill of the person who interact with the system, social or technical context and environment and finally activity type are the three parameters which he suggested to consider during development of usable software (Johnson, 2007).

![Figure 4.1. Bailey's Human Performance Model](image)

Furthermore, Rubin and Chisnell mentioned a few reasons due to usability issue and what makes something less usable (Hearst, 2009). One of those reasons is that development focuses more on the system not on the ultimate end user. Therefore, the problem is that the developers put the most emphasis on the activity component, shown in figure 4.1. This problem is exactly what ArCon aimed to solve. ArCon considers the context and the users (human) as well. Although, developers can create their system models by Papyrus the long item lists on palette and submenus offered by papyrus make task completion more time consuming and erroneous when they must follow a certain set of rules and constraints. Based on usability metrics, the UI should propose the shortest and easiest way for targeted end users to help them perform their tasks efficiently (Johnson, 2007)

4.3 Usability metrics

Metric is a way of measuring a particular thing. Usability study is a qualitative scale in software development. Its target is to assess the quality of the user interface and to determine how easy it is to use the product. But, based on the testing goals determined by the team, it can measure some quantitative measurements as well. The usability metrics determine the problem’s magnitude. So the developing team can assign a priority level to those problems. Therefore, without any usability evaluation the problem’s severity will be just a guess (Johnson, 2007).
Usability metrics should be observable and comparable. So, they must be quantifiable and to be turned into numbers in some way. Tullis and Albert divided usability metrics into five categories (Thomas Tullis, 2008): performance metrics, issue-based metrics, self-reported metrics, behavioural and physiological metrics, combined and comparative metrics. Two of them were of interest for my study which is explained in the following.

4.3.1 Performance metrics

They are the most valuable metrics. They enable the test conductor to measure effectiveness and efficiency of the product. Furthermore, they are very useful to evaluate the magnitude of a usability problem. Because many times you know that an issue exists but you don’t know how big the issue is. In order to be able to measure performance metrics, one needs to define tasks precisely in advance. There are five types of performance metrics presented below.

1. Task Success

It refers to if the participants can complete their task. If they fail, something needs to be fixed. So, before the test session, we need to define the criteria for success on the task. Otherwise, you are not able to collect clean success data.

2. Time-On-Task

Task time is a significant metric to evaluate efficiency. Therefore, it can disclose how much the costs can be saved. If the user performs the task more often, task completion time becomes more important. It is recommended to include only successful tasks because it often takes a long time for users to give up trying.

3. Errors-rate (Effectiveness)

Errors are certainly related to usability issues. Because they are incorrect actions, such as selecting the wrong choice in a drop down menu or taking incorrect order of actions, that can lead to task failure. This metric is useful when you need to find the actions which result in task failure, when errors result in a considerable loss in efficiency or when they result in significant costs.

4. Efficiency

Efficiency is suitable when you are concerned with both time on task and the amount of physical and cognitive effort needed to complete a task. In the case of effort measurement, the effort can be measured by the number of steps or actions taken by the participant.

5. Learnability

It can be assessed by the amount of time and effort that user takes to become proficient with the product. Also it can be measured by performance metrics such as efficiency, errors, task time and number of actions or task success. For tests that trials are performed in one session, memory loss
decreases. So, it is harder to measure this metric. When measuring the learnability, we should consider what is considered as a trial and how many trials to assign to the study.

4.3.2 Self-reported metrics

In order to gather these metrics, you ask the users to tell you about their experience. It is important to know what to ask and in what form to ask. This type of metric is important because it enables users to tell you how they feel, their perception and experience with the tool. Self-reported metrics are measured by rating scales such as Likert or semantic differential scale. This type of data is collected either after completing each task, called post-task rating or after the end of entire test session, called post-study rating.

4.4 Types of usability test concerning product life cycle

Based on what you need to understand about your product, you can apply the correct type of test at the correct moment through the product lifecycle as shown in the figure below (Hearst, 2009).

![Usability testing through the product lifecycle](image)

**Figure 4.2. Usability testing through the product lifecycle**

Exploratory tests are conducted quite early in development process and its purpose is to examine the preliminary design concepts. Assessment or summative tests are conducted either early or midway of the product life cycle, usually after the high level design is established. The objective is to expand the exploratory test result by evaluating how the basic concepts have been effective.

Finally, validation test, referred as verification test, is intended to evaluate the usability of the product against settled benchmarks and standards. The measurement takes place close to product release and the aim is to check if the product’s predetermined usability standards have been addressed before the release and why if it doesn’t. The standards normally originate from usability purposes which were defined early.
in the project. This type of test can cover different aspects. One of its objectives which is the goal in this thesis is measuring user performance. It refers to how well the user can perform tasks and how they rank the product’s design and functionality.

There is also another type of test called comparison test. This type is not associated with any specific phase in development life cycle. If it is done toward the end of the lifecycle it can be used to compare the product with a competitor’s product to find out which one is easier to use and the advantages and disadvantages of each design. In order to compare the designs, performance and preference data are collected.

I decided to conduct a validation test combined with comparison test. In the case that valid statistical results are needed, the designs should vary along a single dimension. But in my work, the visual design is constant and the functionalities vary. In addition, I preferred a more informal, observational and qualitative study. During conducting this kind of test, participants are usually asked to find aspects which make one design better than the other. In my study, these comparisons were not very complicated since the designs were similar.

4.5 Basics of usability study

Conducting usability test follows the same principles as conducting a controlled experiment for classical experiments. With a formal approach, the following steps should be applied (Hearst, 2009):

- Formulating a hypothesis. A hypothesis determines what you expect to happen during testing. For example, “Help in design A will enhance speed and declines error rates rather than when users use Help in design B”.
- Select a representative random sample. The characteristics of the target population must be determined and then you need to choose fairly enough number of samples randomly.
- The users sample size must be sufficient. The total number of participants as well as a sufficient group size must be employed in order to avoid erroneous conclusion.
- Experimental controls must be tightly employed. Regardless of how the statistical results are significant, the experiment must be controlled carefully. Otherwise, the results validity can be called into question.

Under controlled conditions, the formulated hypothesis is done by isolating and manipulating variables. Then through an appropriate statistical method the hypothesis will be either rejected or confirmed. But for a few reasons it is not often possible to follow the preceding approach. The most important reasons which make it impossible are organizational constraints on time and cost and the emphasis for high speed development. In addition, a high level of background knowledge and training is needed to conduct those researches. Often the human factor specialists are needed for these types of studies. Otherwise the study will lead to worse situation since it is misleading. There are more problems such as lack of enough participants. But, probably the most important reason is that classical strategy is quantitative. It doesn’t obtain qualitative information about how the quality of two designs differs. Therefore, the testing experiment is normally done more informal.
Although experimental rigor is necessary for conducting any study, but the results don’t prove that a product works with 100% certainty because of some limitations. For example, the participants are rarely the exact representative of the target population and testing situation is not often as the actual situation. However, despite of these limitations, if usability testing is conducted carefully and precisely at appropriate time it can indicate potential usability problems.

4.6 Necessity to develop test plan

The test plan determines the how, when, why and what of your usability experiment. Test plan is necessary because it is as a blueprint for the test. It determines how exactly you will test your product and also it defines required resources to accomplish your objectives. Without them details of testing get ambiguous. In addition, it serves as communication vehicle. Because it is a document that all involved members in the project can review and check if their needs are being addressed.

The format for test plans can vary based on the formality required in the organization and test type. In the following parts I am going to explain each part of a test plan in more details.

4.6.1 Test objectives and research questions

In this part, the reasons for doing the test are described from the organization point of view in a high level of abstraction. The most important thing is that the test goals must be tied to business goals within the firm and address the problems.

The issues that needs to be focused and answered in this section must be as accurate, observable (measurable), precise and clear as possible. This is the most important part of the test plan because the problems which are the key concerns of the developer team are addressed on this part in details. Normally, after discussion with the developer team, marketing personnel and so on, you can generate more clear research questions.

4.6.2 Participants

Finding the representative subjects to participate in test session is often hard due to the fact that they are busy, scarce or probably expensive. To overcome this obstacle Jeff Autor Johansson (Johnson, 2007) suggested following some tips:

- Some aspects of the product don’t really need the users to have experience or knowledge in a specific domain. For example if a medical software uses a table widget that is designed in windows applications, there is no need to have the doctors check if the goal was achieved.
- Cheaper and easier to recruit participants can be used instead of experienced users in a specific domain. For example, computer science students can be used instead of professional developers, medical students instead of doctors and middle managers instead of top executives.
Selecting the wrong participants makes the test results questionable and of low value. To select the right participants, one needs to identify skills, knowledge and behaviours of people who will use the tool. This description is known as user profile which determines the characteristics of target audience. For the test, participants are selected based on this profile and of course by considering the money, resources and time constraints.

To create a detailed and obvious user profile one need to look in some potential resources. One of the important resources is product requirement document and functional specification such as use cases because they include a fairly good description of the target audience. In addition, product manager has always a clear understanding of the target users whom the product is intended.

To classify participants based on their expertise, it is not always correct to use time and usage frequency as expertise measurements. Moreover, it is not acceptable to simply let participants rate themselves as a “beginner”, “intermediate” or “expert” because they have different interpretations.

After deciding on the target user, the sample size is one of the most important things that should be determined at this step. For a true experimental design you need minimum 10 to 12 participants for each condition to be able to achieve statistic results, suggested by Rubin and Chisnell. But for less formal testing, studies show that four to five people for each design can expose roughly 80% of usability deficiencies.

Determining the number of participants depends on many factors such as the degree of confidence that you expect for results, availability of participants and the session durations. But, ultimately, you need to balance it with time and resource availability. Normally, a few numbers of participants are recruited at the first stages of software development since a small sample size reveals major usability problems. But later, close to the project completion, more participants are needed in order to identify remaining issues.

After defining the target user attributes and the sample size, you should look for sources to be able to find participants from. One of the easiest and cheapest source, but not necessarily the best, to select the participants is company’s internal employees. The other source can be friends and family who have desired characteristics but you should make sure that the relationship doesn’t affect their performance. There are other opportunities such as customers through sale representatives, other society and associations, people referred from co-workers, newspaper advertisements or personal networks.

### 4.6.3 Requirements to conduct test session

This part will discuss all facets of the usability test. A very detailed description is necessary for many reasons. For example, it reveals additional resources such as a receptionist who meets participants when they first arrive; it enables the others to find out what will happen in details so they can make comments or suggestions; and it enables you to focus on the task and what has to be done in details to be well prepared before performing the test.
4.6.3.1 Method selection

Evaluating the UI can be performed either by a usability expert, more common in early stage of software development, or by the real users. Experts pick the common problems and criticize on the most important purpose of the software. Heuristic evaluation, discount usability engineering and cognitive walk through are the variations of this strategy. On the other hand, participant-based evaluation involves some real people. Different methods are proposed for this evaluation type such as cooperative evaluation, co-discovery and controlled experiments (David Benyon, 2005).

The right method for conducting an usability test is chosen depending on when in the product life cycle you want to evaluate your design, what types of measures you are looking for, the evaluation purpose and your resource availability.

Each evaluation technique has its own strengths and weaknesses. Eight factors are suggested in (Alan Dix, 2003) in order to help testers to distinguish different techniques and select the most appropriate one.

1. The stage in the software life cycle. Evaluation at design phase needs design experts and is analytic while implementation evaluation is experimental and requires user involvements.
2. The evaluation style; laboratory or field studies. In laboratory there is the possibility to perform controlled experiments and observation. But you lose the naturalness of real environment. The ideal design process is to include both styles, probably conducting laboratory experiments at early phases and field study on the implementation.
3. The level of subjectivity or objectivity of the method. Subjective methods rely on the evaluator’s assessment, knowledge and expertise. On the other hand, objective studies provide repeatable results regardless of who performs the evaluation.
4. Measurement types; quantitative vs. qualitative. Quantitative measurement is numeric and can be statistically analysed. Quantitative results are harder to analyse but discloses important details which cannot be extracted from the numbers. The measurement type is often related to the objectivity or subjectivity of the method. The subjective techniques provide qualitative results. But this is not a hard rule. Because qualitative data can be represented in a quantitative scale such as user preferences.
5. Information required. The level of information required by the evaluator may range from low-level (for example, the most readable font) to high-level (such as “is the system usable?”). Low-level data is often collected by an experimental study while high-level information can be gathered by query techniques.
6. Immediacy of participant’s response. Some methods like think aloud gather information at the time of task performance. But they affect the way subjects work. Other techniques such as post-task walk through suffer from bias in recall. Because the recall may be incomplete.
7. The interference level. Immediate evaluation methods are intrusive, hard to interpret and are dependent on the information that can be provided.
8. The required resources. Resources include time, equipment, evaluator’s expertise, subject and context.

Design and surface features is not the only concern for the designers (David Benyon, 2005). Another major concern for interaction system designers is implementation evaluation. Evaluation of implementation of a system is different from design evaluation. It can be done without a direct user involvement, for example the designers or expert evaluators can evaluate it. Another difference between the two evaluation strategies is that for implementation evaluation one needs to have an actual
implementation of the system. This can be a simulation of the system without functionality or a fully implemented system (Alan Dix, 2003).

The main goals of the implementation evaluation are assessing the system’s functionality and the effect of interface on the user and identifying any problem with the system. In term of system functionality, it is important that the system be designed in a way to enhance and facilitate user task performance. To satisfy this feature of the system, not only appropriate functionality is needed but also the actions which the users need to perform the task must be clearly reachable. One way to evaluate functional capabilities is to assess user’s performance with the system. In this way, the effectiveness of the system in supporting the tasks will be assessed. To measure impact of system on user, some aspects such as how easy the system is to learn and its usability should be considered. In addition, identifying specific problems in design can be disclosed when unexpected results or confusion happens during test sessions. These problems can be related to both usability and functionality issues (Alan Dix, 2003).

There are a variety of techniques for user centre evaluation: experimental or empirical methods, observational methods, and techniques which the book (Johnson, 2007) calls query techniques.

In imperial method, some attributes of the subject behaviour is measured under a controlled condition. To perform this test, the subjects must be chosen and variables and hypothesis are tested. There are a number of observational techniques such as think-aloud, protocol analysis, automatic protocol analysis and post-task walkthroughs. Query techniques such as interviews and questionnaire are less formal than controlled experiments but still very useful in eliciting the details of user’s view point to the system. Therefore, some problems will be collected which were not considered by designers. Query techniques are cheap and simple and they will provide supplementary data to other methods (Alan Dix, 2003).

4.6.3.2 Task list

In this section the list of tasks which the participants will perform should be explained. The tasks are developed in two steps. First, the task description is developed for the project team. So the details must be supplied enough as the reviewers can check their validity corresponding to company’s goals.

Then, the tasks are expanded into fully detailed task scenarios, which will be presented to the subjects.

In the test plan four main areas should be covered:

1. Usually a short, one line, description of the task.
2. The materials and equipment required to do the task.
3. Your definition of the task completion success. Often, the developers have very different opinions on this. So it is very important to include successful completion criteria (SCC) for each task in order to define its boundaries.
4. Decide on the benchmark. Benchmark evaluates the subject’s performance during the test and will help to evaluate the test results more precisely. Time-on-task is a good metric for this purpose. For early tests, i.e. exploratory tests, it is better to measure the error rates.

To be able to create the suitable task list which discloses the usability problems one needs to do a task analysis. Task analysis concerns with identifying the domain, goals, intentions and tasks for the users of
the interactive systems. Task analysis is analysing the way people perform their tasks which includes the things users need to know to do the task and the steps they should take to accomplish the task goal. The sources to collect this type of data are documentation, observation and interview. The results from the task analysis are used for many purposes such as manual and tuition, requirement collection and system design and detailed interface design. If an object oriented method is used for task analysis, then it is useful to associate the right actions to each object. Therefore, for each object, a menu of possible actions can be displayed.

4.6.3.3 Test location, environment and equipment
In this section, the location, environment and equipment for performing the test should be explained. The location of the test is dependent on the type of target users, which type of test you are going to take (exploratory, comparative, etc.), is the place accessible to participants and so on.

4.6.3.4 Test moderator tasks
Clarifying the moderator’s task is important when the moderator is not well familiar with the test process. One session in the test plan called “session online” is considered to clarify the moderator’s role in the experiment. In addition, moderator must take detailed notes while recording the participant’s behaviour and comments.

4.6.3.5 Data to be collected
Depending on the test goal, both performance measures (such as error rates and time to complete a task) and preference measures (participant ranking and answers to questions) can be measured either quantitatively or qualitatively. Listing down the evaluation measures enables you to scan the test plan in order to check if the test will give you the data you expect.

4.6.3.6 Debrief the participant
The final opportunity to understand why any error, omission or difficulty happened during testing session is debriefing with the participant. Although usability test uncovers the problems debriefing sheds light on how those mistakes or errors occurred because the participant tries to verbalize rationale and thought process behind each action. To be more specific, performing the tasks exposes “what” of performance and debriefing indicates the “why” of performance.

4.7 Tasks after the test
After conducting a usability test the collected data should be converted from a raw format into tables, diagrams and charts. Then, they should be analyzed to draw the conclusions and made recommendations for the product enhancement.
4.7.1 Data analysis

For a comprehensive analysis, one should compile, summarize and analyze data. This enables to answering all research questions and develops recommendations. In addition, it may be necessary to summarize the preference data which come from post-test or post-task questionnaire or debriefing and surveys in order to pull out critical notes. The best evaluation report includes both quantitative and qualitative summaries. Qualitative summaries can be free-form questions and types of errors. Quantitative summaries include errors rates, task completion time and user rating.

After transforming the raw data into more patterned and understandable format, it is time to analyse data to determine which component(s) is the culprit and where the source of errors is by help of the data collected through notes or by camera records and also by considering the participants background.

4.7.2 Result presentation and conclusion

There will be discussion to show how the research questions have been answered by the experiment. You will provide recommendations for future improvement and suggest follow-up research. You need it when your study result raises new questions which need further study or test.
Chapter 5

Think aloud protocol

Think-aloud protocol is one of the most widespread techniques used by usability test professionals. It is a valuable approach to find out how the participants think while doing a task with the software. Also it is a way to discover the problems users encounter while performing the assigned task based on their verbalization. Think-aloud protocol has two variants: concurrent and retrospective think-aloud. In Concurrent Think-Aloud (CTA) participants verbalize their thoughts, doubts, irritation, surprise or any other feelings while performing the task. But in the later approach, Retrospective Think-Aloud (RTA), the participants complete the task first and then talk about their problems and what passed through their mind during performing the task. The later method seems closer to the way end users work with the product regularly even though the laboratory environment and predetermined tasks makes the situation artificial to some extent (Jong, 2003).

5.1 Why Think-aloud matters for usability evaluation

User interface is the part of product which users, the most important people in product evaluation, are faced with. So users rate the usability of the application based on the perception they have about the design and how they feel comfortable while using it (Seman, Hussein, & Mahmud, 2010). Consequently, since using think-aloud is a powerful tool for measuring usability, this kind of test becomes more necessary for product evaluation.

5.2 Think-aloud advantages

Think-aloud test discloses the source of problems and confusions in the software as well as why those kinds of problems exist and how the user tries to work around it. It is considerable that not all problems are disclosed from the participant’s verbalization when using CTA method. Instead, when participant becomes quiet for some seconds, it indicates that he is trying to solve a problem. Moderator should pay attention to this incident and remember to ask him about it later.
In addition, think-aloud technique reveals how participants think about the product and whether it matches with the way it was designed. Since this method reveals participant’s expectations and perceptions about the product, it is a useful approach for early research to explore and evaluate participant’s mental model about how the product should work. Moreover, concurrent approach helps some people such as old people, children and cognitive disable people to concentrate more on the task they are performing. In the cases that participants may find CTA distracting or difficult, retrospective method could be used as a more effective method (Jeffrey Rubin, 2008).

5.3 Think-aloud disadvantages

One unavoidable drawback in this methodology is that participants often filter their thoughts consciously or unconsciously. So it is impossible to know everything that participants think about the product in both variations of think-aloud method. Some participants avoid retrospective method because they misunderstand how it works in a way that they think it is necessary to rationalize and revise their behaviour rather than just verbalizing what happened and why. In this case, it is suggested to demonstrate to participants how the method works by simply doing an unrelated task for a few seconds and also let them try it once as trial and ask any question to clarify the way this technique works (Jeffrey Rubin, 2008).

As a matter of fact, one problem with concurrent think-aloud approach is that participants may do the task differently when they have to express their thoughts orally from regular time when they do a task individually without any verbalization. Consequently, they may perform the tasks worse than usual or false alarms may be generated (Jong, 2003).

On the other hand, in concurrent think-aloud user performance is slower. So it should be avoided for the tests in which it is needed to measure the time of task completion. Furthermore, it becomes exhausting for participants to verbalize when the task takes long time to be completed. Despite the objectives mentioned for CTA, it prevents errors that might can be made in a normal situation that user works silently with the software because it increases mindfulness and participant’s attention to the task.

5.4 Making both variations more efficient

Despite Nielsen, Clemmensen and Yssing (Janni Nielsen, 2002) and Ericsson and Simon (K. Anders Ericsson, 1993) who prefer introspection, some other researchers such as Boren and Ramey (Boren, 2002) recommended speech communication during concurrent think-aloud testing sessions. In fact, they believe that when moderator provides some probes to acknowledge participants, he motivates them to keep talking and results in more useful information rather than the traditional method in which the moderator sits silently beside the participants during the whole test session. In speech communication fields of study, it is proven that speaker needs to hear at least some verbalized sounds such as OK, uhum and ohhh to ensure that the listener is paying attention to him (Romaine, 1999), (Nigel Ward, 2000) (Yngve, 1970).

The problem with retrospective technique is that participants have no stimuli for verbalization retrospectively which results in negative effect on the comments they express. Best solution suggested, is
to record the performance by video recording or log files. These records not only help participants to remember all their thoughts in the period they were performing the task, but also stop them from inventing thoughts which endangers the validity of the method. If test moderator applies stimuli to retrospective verbalization, then benefits from both approaches (work silently and think-aloud) will be accomplished (Van den Haak, de Jong, & Schellens, 2004). It is notable that all the researches discussed in section 8 used records for retrospective approach.

### 5.5 Research questions

To compare two variations, some aspects were taken into account in most of the studies. The most important aspect of usability test is usability problem detection which includes both number and nature of the problems. This factor was aimed in most of the studies investigated on comparing two variations.

The other important aspect, which should be considered for comparing two variations of think-aloud method, is how participant’s task performance differs within two methods. The last aspect relates to individual experience of all participants while performing the test, i.e. how they feel about test situation, tasks and test method. To sum up, there are three research questions which this paper focuses on:

- How do concurrent and retrospective think-aloud methods vary in number and nature of problem discovery?
- How do participants’ task performances differ?
- How do participants evaluate each method?

### 5.6 Some available researches on comparing the two variations

Despite the popularity that think-aloud protocol has, very few scientific attempts have been done on evaluating its reliability and validity. As a result, Haak and Jong made a research project to study drawbacks and benefits of three approaches: working silently, thinking aloud concurrently, or thinking aloud retrospectively. They tested an online library catalogue. The results disclosed that verbalization detects more problems rather than when participants do the task silently as well as using the two methods of verbalization resulted in the same types and number of problems. The most important output they got was that subjects in retrospective approach did the task in a higher rate of success and less number of observable problems than subjects in concurrent verbalization method. The authors think that higher workload in concurrent verbalization is the reason of declining participants performance while working on the task (Van den Haak, de Jong, & Schellens, 2004).

To make a deeper study, Haak and Jong with help of Schellens employed and compared two variations of think-aloud protocol to test the usability of online library catalogues in 2004, which involved three aspects of comparison discussed in section 7 and in addition studied the relevance of detected problems (Van den Haak, de Jong, & Schellens, 2004). In term of number of problems distributed over the same task and relevance of the problems detected in each method, there was no difference between two methods. On the whole, the important difference was the way problems were detected. In concurrent
verbalization most of problems were detected during observation while retrospective technique mostly revealed the problems by means of verbalization after task completion.

Furthermore, an important discovery in term of task performance and successful task completion in Haak, Jong and Schellens study was that both variations had the same result despite the fact that participants in CTA had more workload. This result was against the previous study results in which user performance was worse for the tasks which were easier than the tasks in the present study. Although it seems that there is no dependency between task difficulty and reactivity, Haak and Jong suggested in the previous study to investgate more studies on it in the future.

Practically, CTA is a strong method based on the results founded in this study. The output is similar to RTA in terms of quantity and relevance of problems reported. But in term of time and costs, CTA has practical benefits in a way that it is less time consuming compared to RTA which needs double time. On the other hand in RTA participant’s performance are recorded which increases test resource costs. Finally, the authors cited that it is early to make any confirm conclusion about the preference of using CTA method because in term of the numbers of individual problems detected and the number of unique problems per condition, the CTA was less productive rather than the other approach. In addition, there is one more problem with CTA which needs more investigation. Since it reveals considerably larger number of observable problems than RTA, those problems are more valid in the case that the task completion involves many visual steps.

Hoc and Leplat used and evaluated three different variations of verbalization concerning think-aloud instructions for a task in which participants had to sort the order of a number of letters on the computer screen. The first method was simultaneous verbalization (the participants were asked to verbalize their thought while doing the task), the second one was subsequent verbalization with aid of record of their performance and behaviour and the last approach was subsequent verbalization without any recall help (Jean Michel Hoc, 1983). It is noticeable that both, the tasks assigned to the participants (which was a logical puzzle) and the way the results were analysed (i.e. the strategy that participants used to perform the task was more considered than problems that they encountered) in this study, is not matched to the usability testing situation (Jean Michel Hoc, 1983).

They discovered that aided subsequent verbalization was the best method since it did not destroy automation and ended in more satisfaction and very correct verbal report. Unaided subsequent verbalization was found as the worst method specially when there was a long time between task execution and verbalization. The researchers uttered that unaided subsequent verbalization should be avoided because it does not produces valid data. They recommend not using simultaneous verbalization for problem-solving sessions because it takes longer to complete the task (Jean Michel Hoc, 1983).

Bowers and Snyder implemented two variations of think-aloud protocol (concurrent and aided retrospective supported by video records of the participant’s performance to recall their thoughts) in a usability test to compare usability of two different sizes of monitors. Each subject was asked to perform 12 tasks on each monitor while for each task different number of windows was required to work with in order to complete the task. Each participant’s performance was evaluated by three factors for each monitor and method: 1. the number of steps participants performed to complete the task, 2. the total time for completing each task, 3. the number of participant’s errors. Also subjective data was collected.
concerning task difficulty rate and level of participants’ satisfaction. They found differences between information were collected from two different methods. For concurrent method, participants gave more procedural information while in retrospective method they reported more explanations and ideas about the design. But the performance and subjective information such as task completion time was the same in both methods (Victoria A. Bowers, 1990).

Although Haak and Jong found Bowers and Snyder results interesting, they criticized it because they did not record the number and types of the problems that the subjects encountered in each method. Because the problem discovery is one of the most important and most critical goals of usability test which was ignored in their study (Van den Haak, de Jong, & Schellens, 2004).

In an investigation, Page and Rahimi, 1995 applied two variations of think-aloud usability method - concurrent and aided retrospective- in order to analyse both intentional errors (mistakes) and unintentional errors of human-computer interaction. Their results supported Bowers and Snyder (1990) findings beside additional information they collected to discover problems in design which was forgotten in Bowers and Snyder study (Colleen Page, 1995).

On the other hand, Capra, 2002, evaluated two variations of the think-aloud method based on critical incidents gathered from user reports in a study (Capra, 2002). Although his participants preferred to do concurrent approach, their report strength in both methods were equal. Based on the incident in reports, the author judged that both techniques resulted in the same number of incidents while there was no difference in the level of participant engagement and incident. He found longer performance time in retrospective method as the only difference between two methods. Moreover, the researcher could measure task performance by RTA which is an important parameter to find the usability level of the product. Capra suggested investigating a study to evaluate and compare the contents of the incidents reported in both methods.

5.7 Enhancing concurrent think-aloud approach

It is likely that participants use verbal fillers or they get silent during CTA test session. In addition, the verbalization itself cannot provide all behavioural information. This issue is always ignored by usability researchers. Cooke (2010) used video containing participants’ real-time eye movements during verbal fillers and silences to gather insight information about participants’ behaviour at those times in order to gain a perfect picture of their experience (COOKE, 2010). He found out the moments that participants did not verbalize, were important to the moderator because it was possible that processing was slowed down at these moments as a result of usability problems. Because the eye movement records revealed that participants were still engaged with the task at those instances. He suggested moderators to take note carefully at those moments and ask about it later when participants would complete the task.

To enhance concurrent verbalization technique, Cooke and Cuddihy investigated a study (Cuddihy, 2005) on applying eye-tracking in think-aloud technique for a website. They used eye-tracking since there were some problems which couldn’t be solved in think-aloud method based on the nature of test strategy and its conditions. For example, for hard tasks which take participants mental concentration or when participants are not proficient in verbalizing their thoughts, no comprehensive and correct information can
be obtained by prompting them with encouraging statements or by asking them to explain about it after completing the task. They discovered that even highly skilled people participate in think-aloud test sessions, failed to verbalize what they were looking at while they were looking for a piece of information on the website. So the eye movement data supplied hints about where users expected to find information on the webpage. The researchers sited that when users hesitated to click a link and instead scanned other links, it implied that they had low confidence level in the link they were about to click. The level of confidence, which was collected from participants’ eye movement, disclosed the level of quality of links name in a website. Hence, they suggested the test facilitator to note down questions regarding certain eye movements or eye gaze position to clarify them with the participant after task completion.

On the other hand, in (Technology, 2009) it is suggested to refuse eye-tracking with concurrent think-aloud because participants produce some eye movements in CTA which they won’t do if they perform the task silently. For example, they often look at some special areas or look away from the screen while they are verbalizing. Instead, RTA is recommended when eye movement data will be analysed.

5.8 Combination of the two methods

Taylor and Dionne (2000) investigated a study to discover complicated reasoning process in problem solving (TAYLOR K. Lynn, 2000). They applied verbal report to their test strategy to be able to follow participants’ mental processing procedure. More than discovering the type of problem-solving strategy that participants selected in two variations of think-aloud, the study was investigated on assessing the effect of using these methods both separately and together.

They found out that using both approaches together provides richer information about problem-solving strategy knowledge rather than when each method was applied alone. The enhancement of results was in two ways: the spectrum of knowledge accessed and the level of details gained. In term of spectrum of knowledge, concurrent reporting resulted in more detailed information about small step taken and the actions performed during problem-solving procedure. Whereas more global data concerning conditional knowledge, beliefs and strategy acquisition knowledge was revealed within retrospective reports because the participants tended to report major steps taken in the process. Therefore, despite the limitations exists for applying two methods together, the authors believe that both methods are essential to gather comprehensive information about problem-solving knowledge. Since the concurrent data was the basis for retrospective report, many parts of two reports intersected. As a result, retrospective reports decreased level of interference that was required for interpreting concurrent report data since it elaborated and clarified concurrent report.

In addition to generate more information, using both methods together enhanced validity and reliability of gained information. In terms of reliability, retrospective reports which designed as a semi-structured interview based on concurrent reports, ensured that all categories of information were gathered whenever possible. In terms of enhancing validity, retrospective data provided opportunity to verify and clarify the interpretations of concurrent report when researchers must draw inferences of interpreting data. To put it differently, participants retrospectively reported on the actions they took during problem-solving session.
So they retrieved memory of their own strategy and did not retreat problem-solving knowledge which is a high risk possibility when using retrospective approach alone.

5.9 Factors influencing verbalization problems

Using verbalization has some problems. Leplat & Hoc studied these problems and the factors which affect them. They classified these factors into three groups: the type of task is given to participants (for example searching for a piece of information or solving a puzzle), the type of instruction for verbalization and finally the time of verbalization (Jean Michel Hoc, 1983).

Benefits and drawbacks of two strategies related to the time of verbalization (simultaneous and retrospective) were discussed through some case studies in section 8 in this paper. In the following two other factors are discussed.

5.9.1 The type of task studied

More the participants engage with the task during performance, more the verbalization is useful. But since participants can only verbalize what is represented to them, Ericsson and Simon (1980) expressed that sometimes verbalization gets too hard depending on the representation, because in those cases participants need a hard coding or inference to be able to verbalize the representation.

Moreover, sometimes the process of performing task changes after a while and begins to become automated and as a result the performance increases. In this case, when the task becomes more regulated, participants get less interested to verbalize it (Jean Michel Hoc, 1983).

5.9.2 The type of instructions given for verbalization

Thinking aloud instructions have been popular because the participants point at minimal irrelevant and unnecessary comments about the task while performing it or after that when they verbalize their performance based on the video records. But the instruction should be in a way that guarantees completeness and excludes criticism (Jean Michel Hoc, 1983). For example, in the study was done by Hoc and Lepat, moderators asked the participants: "say out loud what you say to yourself while you do the task". They discovered that the instruction of think-aloud resulted in a higher level of exigency and validity, at least in the French language, because participants had no time to play with words while performing the task (Jean Michel Hoc, 1983).

There are higher level instructions for this kind of test. For instance, the participants are asked to talk about the way (the procedure) they perform the task which includes the states that they pass to complete the task. In a higher level, the participants may evaluate and give an acceptable explanation to justify why they select a procedure for the given task (Jean Michel Hoc, 1983).
5.10 Conclusion

Comparing the results of mentioned researches implies that despite the longer time RTA test takes, it is more preferred by participants as a reason of the fact that retrospective verbalizing approach is more similar to the way users normally work with the product lonely. Furthermore, participants do the similar task in RTA with higher performance rather than when they have to verbalize during performing the task. Especially when tasks are more difficult, performance in concurrent method gets worse since participants have more workload during test session. Finally, most of researches found no difference in terms of number and nature of problems found by RTA and CTA but the way problems showed up were different in two variations of think-aloud protocol. To put it differently, most of the problems in concurrent verbalization are revealed during observation and task performance while in RTA problems are mostly detected during verbalization after task completion.

On the other hand, if performance time does not matter for the research and test results, concurrent think-aloud protocol is proven to be better a approach for children, old and disable people since it helps these groups of people to have more concentration on the tasks they are performing beside it takes shorter time for each test session. In addition, if test moderator makes the test session more communicative with some short probes, he would get more benefits of this strategy since this technique minimizes the time participants get silent and encourage them to talk more. The last but not least important way to improve concurrent think-aloud is using eye-tracking to obtain information about the moments in which participants are silent during the test session and also to calculate the level of confidence they have while performing the assigned task.

After all, applying two methods together by Taylor and Dionne (2000) resulted in richer, more valid and more reliable data in the case that task nature was problem-solving. This method not only increases test costs considerably, but also has some limitations. The researchers need to balance the potential for improving value of data collected and the costs needed for applying complementary measures.

To conclude, more researches on comparing CTA and RTA is needed in order to clarify their differences and their own advantages and drawbacks based on the variety of task nature that could be problem solving task or searching for a piece of information or any other kind of task. Some important parameters which could be considered in the future investigations on comparing the two variations of think-aloud protocol are: criteria(s) for selecting people who the test is going to be taken with (such as age, language, culture and believes), task difficulty level, the product type, the goal of usability test which could be disclosing design problems or procedural problems, dependency between task difficulty and reactivity, differences between the content of incidents reported or any other parameter which could affect the results or disclose differences between two variations.
Chapter 6

Test Plan

This chapter describes how the usability test was planned. The information required for the test plan are discussed in details in order to clarify the requirements for conducting the test sessions, the approach to conduct the tests and the test goals.

6.1 Study Goal

System designers have to model the system under the constraints which the architect defines in advance. The constraints are known as architectural rules. The problem is that there is always the risk for the designer to violate the architectural rules or get confused about how they should create their design, because there is not a standard format for expressing architectural rules. Instead, they are often presented in a text file format.

In order to solve this problem, ArCon introduced an approach to formalize the architectural roles. Therefore, the system design can be automatically checked against those formatted rules and any violation will be reported to the designer immediately. But, this capability of ArCon hasn’t solved the whole problem yet. The designers need to get more support from ArCon while creating a system model. In another word, ArCon could tell them what errors they made in their model. But it couldn’t guide them step by step during the design process in order to help them avoid those mistakes.

Therefore, the aim for this thesis work was to provide support during the design process not after completing the model. This enhancement in ArCon was expected to make an improvement to design process. To provide this support, users should get domain specific guides through UI while they are making their own system design. So, any time the users want to add an element to the diagram, instead of getting confused with long lists of element types on the palette or right click menu, they will have only the list of elements which are permitted to be added to a specific fraction of the model.

For this support, the user interface design wasn’t supposed to be created from the scratch. Instead, the right approach was to follow Papyrus user interface design. Because any change in the current UI design or any new component on Papyrus perspective could result in user confusion and dissatisfaction. Also it could cause negative effects on efficiency and increase error rates, especially for the experienced Papyrus users who are used to its UI.
The aim of my usability study was to measure a number of performance metrics and also the level of user satisfaction when users get domain specific modeling support on the UI by ArCon. So, I planned and conducted a usability study in order to answer the following questions:

- If this master thesis made any improvements with respect to usability among the different user types?
- What are the main usability issues of the implemented support?

6.2 Variables

The first step in the usability study is to identify both the independent variables (what you want to manipulate) and dependent variables (what you want to measure as the study results). The best source for extracting the variables are research questions i.e. the questions which should be answered after analysing the results of the tests. I extracted the variables for my experiment as listed below.

I. Independent variables

1. User expertise level. An important parameter in user performance was the user’s knowledge and experience level in UML, Papyrus or even any other modeling language and tool and ArCon.
2. Treatments. The user interface and what it provides for the testers is an important factor affecting their performance and satisfaction level. In this study, participants were expected to work with two user interface components, ArCon palette and Papyrus palette, which were similar in design but they supported different contents.
3. Tasks. Definitely the complexity of the task affects user performance. In some cases the more complicated the task is, the more usability issues are revealed.
4. The order of performing the tasks. If there are similar tasks that the participants in usability test should do, the later tasks will be certainly performed in a higher performance level because of learnability effect. Therefore, task order can affect the study results. With respect to the goal of my usability study, user experience level and treatments were the independent variables which were of interest in this study. Therefore, I tried to control the other two variables in order to minimize the role and affectivity of them on the test results. So, I could collect more precise results concerning the variables which I was interested in.

Note that the tasks and the order of performing them can be seen as compound variables. They couldn’t be completely removed in the study but they could be diminished. To avoid task effects, I defined two tasks, A and B, which are roughly similar in the complexity level and the amount of effort they need to be completed. So, the task variable couldn’t have a considerable effect on the results. On the other hand, to bias the learning effect a training session was planned before conducting the test. Moreover, the tasks A and B were shifted between each group members. In another word, in each team one person started the test with task A and the other started with task B. This technique is called counterbalancing. The detailed description about the tasks and their order is presented in section 6.6.

II. Dependent variables

2. User satisfaction level.
3. Other self-reported issues.

Performance metrics were collected during test sessions while user satisfaction and self-reported issues were extracted from questionnaires and open-ended questions.

6.3 Within-subject or between-subject study

If the study’s goal is to compare the results of different tasks performed by each participant, such as comparing error rate applying two different treatments, the approach is called within-subject study. For this type of study you don’t need a large sample size since participants are not individually compared to the other. Instead, the focus of study is the treatments and to discover differences between them.

On the other hand, if the results are analysed in order to compare dependent variables between different subject types, such as comparing error rates for different types of target users, the study is called between-subject test. I applied this approach in order to measure and compare three general groups of target users (introduced in section 6) in order to discover how the usability level was affected for different groups when they took the two different treatments.

6.4 Method selection

Considering the factors discussed in 4.7.3.1, two types of user-centre evaluation techniques were selected for the study: observational method and query technique. Query techniques were applied in order to disclose self-reported metrics discussed in 5.1.

Since think aloud protocol lets you to see how and when the subject gets stuck while performing the task you can find out where the problems come from. Also, you can measure the metrics more specific to those sections in UI which you need to evaluate and ignore the other parts. Therefore, I applied think aloud as the observational method for my experiment in order to take advantage of this possibility. Furthermore, I could use the other advantages of think aloud for collecting observational metrics which were discussed in chapter 5.

Considering the testing environment and circumstances in this experiment a minimum level of knowledge about UML, Papyrus, system design principles and ArCon’s rules and constraints were necessary for subjects in order to be able to perform the tasks. Although the tasks were defined in a way to minimize user engagement with other requirements such as Papyrus and focus more on the DSM support in ArCon, there was no way to completely omit those variables. For example, it is normal that new users to Papyrus make mistakes or spend longer time on modeling due to their lack of experience. Therefore, by applying think aloud I could distinguish where the source of usability issues was and ignore the ones that were irrelevant to my study goal. As a result, this approach enabled me to gather more accurate and reliable data for measuring the metrics of interest.

Last but not least advantage of selected approaches is that both are cheap and effective for evaluating usability metrics.
6.5 Participant selection

Considering Popovic’s state (Thomas Tullis, 2008), all interactive UI designs should support novice to expert users and facilitates their interactions with the system. Therefore, to have a cost-effective usability study it is vital to recruit all possible types of intended users in order to uncover a wide range of usability problems.

Therefore, a cognisance of distinction between experts and non-expert users has to be determined prior to study their comparisons in interacting with a system. Among different empirical definitions for the differentiation between experts and non-experts, knowledge (particularly domain specific knowledge) and the time spent on working on a specific system (Jochen Prümper, 1991) are the most common criteria. Furthermore, they can differ in the problem solving approach (Mantere, 2001).

Knowledge and time factors have a strong overlap. But focusing on the knowledge Prumper (Jochen Prümper, 1991) believes that non experts operate at a lower cognitive level than expert do. Furthermore, non-experts need more assistance and encouragement in reporting the problems than experts do (Papyrus Project). But they are still necessary to help to disclose those usability issues which experts cannot reveal.

User classification in my study was based on knowledge criteria. Basically, intended users of ArCon should have background knowledge in a few areas in order to be able to use ArCon. The basic requirement is knowledge about both UML and system design. A higher level of requirement is having experience with Papyrus or any similar modeling tool. The last but not least requirement is experience with ArCon. Definitely, more users know in the mentioned required areas, easier they can work with ArCon. But I defined a minimum acceptable level and necessity criteria for each requirement shown in the table 6.1. Any lack of knowledge in other areas was supported in training sessions before the test.

Table 6.1. Minimum requirements for different tester groups

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Minimum acceptable level</th>
<th>Necessity (requirement level)</th>
</tr>
</thead>
<tbody>
<tr>
<td>UML</td>
<td>Had a course and did a few assignments</td>
<td>Basic</td>
</tr>
<tr>
<td>System design</td>
<td>Had a course and did a few assignments</td>
<td>Basic</td>
</tr>
<tr>
<td>Papyrus</td>
<td>Knows how to create models with this tool</td>
<td>Intermediate</td>
</tr>
<tr>
<td>ArCon</td>
<td>Know the rules and constraints and used at least once for checking the design validity</td>
<td>Advanced</td>
</tr>
</tbody>
</table>

There was a temptation to take more specific information, for example what projects/assignments they had done using each of the subjects above, how long they had experience with each of those subjects or when was the last time they applied them. This information might help to analyse the results of test but they were not necessary for participant selection.

Please notice that the requirements presented in table above have overlap. Any users who have worked with ArCon, certainly has knowledge about Papyrus, UML and system design. On the other hand, the users who have experience with Papyrus or any other modeling tool know about UML and system design.

In the next step I needed to classify the entire population of end users and then select a fairly enough number of participants from each category. I used the table above for classification. The necessity of
requirement was the factor which I used to distinguish different categories. Therefore, I ended up with three groups: Introductory, intermediate and advanced and put each sample in the group which he/she could satisfy the minimum acceptance level for demanded requirements.

<table>
<thead>
<tr>
<th>Level</th>
<th>Desired requirement(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introductory</td>
<td>UML and system design</td>
</tr>
<tr>
<td>Intermediate</td>
<td>UML and system design</td>
</tr>
<tr>
<td></td>
<td>Papyrus</td>
</tr>
<tr>
<td>Advanced</td>
<td>UML and system design</td>
</tr>
<tr>
<td></td>
<td>Papyrus</td>
</tr>
<tr>
<td></td>
<td>ArCon</td>
</tr>
</tbody>
</table>

The distribution of participants who were requited for this study is shown in table below.

<table>
<thead>
<tr>
<th></th>
<th>Number of participants</th>
<th>Age</th>
<th>Career</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced</td>
<td>2</td>
<td>25-30</td>
<td>Software developer, Project manager</td>
</tr>
<tr>
<td>Intermediate</td>
<td>2</td>
<td>30-35</td>
<td>Software developer</td>
</tr>
<tr>
<td>Introductory</td>
<td>2</td>
<td>25-30</td>
<td>Newly graduate in computer science</td>
</tr>
</tbody>
</table>

Clearly, the more individuals you have in every subset of population, it will help to achieve better conclusion. But due to limitations in cost and resources, I couldn’t recruit more participants especially for the advanced group because ArCon is a new product and currently has a few end users.

6.6 Test Location and Environment

To be able to create a good test plan, test location and environmental conditions need to be identified beforehand. The ArCon’s users are almost software designers. Therefore, the test can be done in any place similar to an office where a software developer works. I chose one of the meeting rooms in the Combitech office for conducting the whole tests.

To have a more precise comparison between users performance, all tasks were performed on one computer because different computers may differ noticeably in a few parameters like speed and monitor size. But a set of setups were required on that computer i.e. Eclipse, Papyrus and ArCon should be installed. There was no need for internet. I used a voice recorder in order to capture participants’ performance. So, I had the possibility to collect some data that as a moderator I might miss during the test sessions.
6.7 Metric Selection

It is not always possible to satisfy all the usability attributes since there are conflicting usability requirements. First, you need to identify all of them and then you should determine which have the higher priority based on study purpose and targeted users.

The most important goal of the implemented support in ArCon was to improve the amount of effort needed to complete a task and also reduce the error rates and increase the possibility of task success. All of these metrics are chosen from the performance metrics category and think-aloud method, as discussed in chapter 5, is a strong observational approach to measure them.

Although think-aloud is a good technique for extracting usability issues and measuring the performance metrics, I was also interested to collect more data about the effect that new feature in ArCon had on the quality of the user interface. Therefore, I selected self-reported metrics as a supplementary approach to the study. It enabled collecting information about overall user satisfaction and the usability problems that the participants find.

To collect data for self-reported metrics, paper questionnaires were used. Open-ended questions in questionnaires were represented. For each questionnaire 7 points Likert scale was applied. Usually participants respond in a more positive way when they are asked directly to answer the questions. This is called social desirability bias (Nancarrow & Brace, 2000). To avoid it, I left them alone in the room while they were filling the questionnaire.

A post-task questionnaire was proposed to each participant after completing each task. The questionnaires are similar for both tasks and they focus more on the treatment the testers took to perform the task. To capture performance differences between different categories of users (between-subject study) a post-session questionnaire was prepared. The questions were about ArCon and how satisfied the testers were with it. Both questionnaires are provided in the report’s appendix C.

6.8 Task Definition and Assignment

Always, testing different tasks is on demand but it is costly. Therefore, the focus in my experiment was more on the critical tasks, the ones which provide more specific and precise results. On the other hand, there was a risk that participants learn to perform the tasks while doing the first task and then they do the second task faster and easier.

In order to fully cover what I needed to evaluate, I designed two tasks A and B (described in details in section 7.1). The tasks focused on creating a valid system model following an architecture model. They were similar in complexity level and also the amount of effort and interaction with UI in order to be completed. But they were apparently different. To mitigate learning affects the following cases were applied.
6.8.1 Counterbalancing

Counterbalancing is a technique for biasing learnability. In this technique, the order of tasks for participants is randomized or balanced. As a result of variation in task orders, the effect of learning transfer will decrease, especially for the non-experienced group.

All the subjects in my experiment performed task A without any Domain Specific Modeling (DSM) support and task B was performed with DSM support. Therefore, with respect to counterbalancing, the order of task performance for each participant was decided as table below.

<table>
<thead>
<tr>
<th>Group</th>
<th>Participant</th>
<th>Task order</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced</td>
<td>Johan</td>
<td>A, B</td>
</tr>
<tr>
<td></td>
<td>Gert</td>
<td>B, A</td>
</tr>
<tr>
<td>Intermediate</td>
<td>Rickard</td>
<td>A, B</td>
</tr>
<tr>
<td></td>
<td>Joel</td>
<td>B, A</td>
</tr>
<tr>
<td>Introductory</td>
<td>Mana</td>
<td>A, B</td>
</tr>
<tr>
<td></td>
<td>Babak</td>
<td>B, A</td>
</tr>
</tbody>
</table>

6.8.2 Training sessions

A training session was planned before each test session. So, the participant had the opportunity to get more familiar with Papyrus user interface and ArCon’s constraints. This training session was more necessary for those participants who hadn’t used Papyrus or ArCon for a long time or had no experience of them.

6.8.2.1 Test manual

A printed document explaining the basic introduction to ArCon, UML, Papyrus and the process of the test session was handed out to the participant during the training session. The explanations were simple and just enough for being able to create a system model by Papyrus following ArCon’s rules. The manual is attached to the report appendix D.

6.8.2.2 Small examples

I offered a few simple architectural model examples to the participants as practice and asked them to create the simplest system model which satisfies all the rules and constraints defined in that architectural model as an exercise.
### 6.9 Test Plan Table

As mentioned in Chapter 4, in order to conduct a usability study, a good test plan should be prepared prior to the test session. The plan is a blueprint for the test and serves as a communication vehicle in the company. Table 6.5 presents the test plan which I prepared.

<table>
<thead>
<tr>
<th>Table 6.5. Test plan</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Introduction</strong></td>
</tr>
<tr>
<td>The purpose is to collect qualitative data from participants' experience using ArCon in order to measure the level of usability that ArCon provides by domain specific modeling support. Generally, data about effectiveness, efficiency, and user satisfaction level will be collected.</td>
</tr>
<tr>
<td>A few usability tests are planned. The test target is to compare the usability level between ArCon and Papyrus Palettes considering three different user types.</td>
</tr>
<tr>
<td><strong>Test Scope</strong></td>
</tr>
<tr>
<td>Currently ArCon provides a number of support and functionalities for the system designers. But this test is specifically planned to evaluate the level of usability that ArCon provides in order to guide users in a way to decide which element type they can select while dragging elements from the palette to their system model. The support is based on the model’s domain which is extracted from another model called architectural model.</td>
</tr>
<tr>
<td><strong>Participants</strong></td>
</tr>
<tr>
<td>There are 6 participants divided into 3 groups based on their knowledge and experience level in required qualifications. They are my friends or the internal employees at Combitech.</td>
</tr>
<tr>
<td><strong>Methodology</strong></td>
</tr>
<tr>
<td>Before the test session a training session will be held. There are two similar tasks (A, B) defined for all participants. But the tasks are performed in reverse sequence in each group in order to omit learning effect. For each task participants should create a system model following an architectural model. The two models designed for tasks A and B are very similar in complexity and workload in order to be completed. To measure user satisfaction level each participant will be asked to fill out three questionnaires, one after completing each task, and one at the end of the test session. On the other hand, performance metrics are measured by applying think-aloud protocol.</td>
</tr>
<tr>
<td>Time slot needed for different groups vary. Because introductory group need more time for training and performing the tasks. Finally, error rate, efficiency, and satisfaction level will be collected and then analysed following both between-subject approach.</td>
</tr>
</tbody>
</table>
### Equipment

An empty meeting room in the office, laptop and voice recorder.

### Session outline

Arrangements before the test (10 minutes)
Have the participant in location.
Give an introduction to participants (10 minutes)
Make a brief introduction to usability studies.
The role of the participant in the study and its importance.
Session’s outline.
Training session (10-60 minutes)
Give an explanation on how to create models in papyrus and how the system models can be designed based on ArCon’s constraints.
Perform task A (20-90 minutes)
Post-task questionnaire (10 minutes)
Perform task B (20-90 minutes)
Post-task questionnaire (10 minutes)
Post-session questionnaire (10 minutes)

Note: Performing the tasks continues until scenarios are completed or the maximum time allocated has elapsed.

### Research questions

If this master thesis made any improvements with respect to usability among the different user types?
What are the main usability issues of the implemented support?

---

### 6.10 Pilot test

In order to improve the quality and efficiency of the study, a small experiment, called pilot test, was designed to be conducted prior to the real experiment. Generally, the pilot test reveals deficiencies, tests feasibility, time and cost for the planned study in a small scale.

For my study it was helpful to test equipment, practice myself as test conductor, estimate the length of the test session, determine the quality of the questionnaires, find the lack of information in training session, find any ambiguity or confusion for participants, discover any other important issues and make any last minute adjustments.
Chapter 7

Experimental Results

This chapter will go into the collected results from the usability test described in chapter 6. All the results are presented in form of bar charts and related discussion is provided. The results are divided into two categories. First the results collected from user verbalizations and my own observation during the test sessions is presented. Finally, the collected data from questionnaires and open-ended questions are reported.

7.1 Experimental setup and procedure

Six persons were chosen as the representatives of system designers in this study. Based on their background knowledge and experience they were divided into three categories as “expert”, ”intermediate” and “beginner”. The usability tests took place in a meeting room located at Combitech AB in Linköping. Testers were invited to come and do the tests individually. For each person, the whole test sessions took one to four hours depending on the evaluator’s speed which was related to his/her background knowledge, experience and personal problem solving pace.

Each participant was asked to contribute in two test sessions. For each session, one task was assigned to the tester. Before starting the test sessions, both written and verbal instruction was given to the participants. For each task an architecture model was given to the testers and they were asked to create the simplest system model following the given architecture model.

Every evaluator performed one task with and one task without receiving the treatment. The treatment in this experiment was a Domain Specific Modeling DSM support implemented as my thesis work for ArCon. The goal in this master thesis was to implement a DSM support through GUI to ease design process for system designers. The current DSM support is provided both on the palette and a popup menu on one of the Papyrus views called Model Explorer. Since, the DSM support on both sections has exactly the same functionality, logically the type of problems users find is the same either they use the DSM support on palette or popup menu. On the other hand, it is more comfortable for users to work with palette and diagram editor because those sections are more interactive. Therefore, I asked everyone to use just the
DSM support on palette, called ArCon palette, when they were asked to get the treatment during their experiment.

Since the selected method for this experiment was concurrent think-aloud, the participants were asked to verbalize their impression of the user interface and the steps they took while the performed each task. During the test sessions, I recorded participant’s verbalization by a voice recorder. Therefore, I used the records later to collect data, which was of interest for this experiment, through their verbalization. Also, I observed their performance during the sessions and tried to write down notes whenever I noticed that the participant was faced with any usability problem.

After performing each task participants received a questionnaire to score how they experienced their participation. After performing both tasks, the participants were asked to fill in another questionnaire which was intended to compare the experience they had for each task.

The only current DSM support implemented in ArCon is suggesting the type of elements which users are permitted to add to a fraction of system model. Therefore, to evaluate the implemented support, the focus in this experiment was only on the element type. In addition, in this experiment, applying the right stereotype to each element was a must for each task and couldn’t be ignored. Therefore, in order to define almost complex and difficult tasks, the amount of effort to complete the task was defined in terms of total number of element types, minimum total number of all elements, minimum number of each element type and also minimum number of times when evaluators needed to apply a stereotype to an element. Based on my assumptions for evaluating the complexity of the task, two architecture models for two tasks, called Task A and Task B, were defined for this experiment. The minimum amount of effort for each task is represented in table 7.1. In addition, the snapshots of the both architecture models are provided in the appendices A and B.

There were many other DSM details which were not supported by DSM support, such as permitted stereotypes which are applicable to an element or naming constraints. So, in order to decrease the effect of the other types of architectural rules which were not in the scope of this thesis, I asked the participants to ignore them and don’t spend time on setting those rules in their system models. In other words, the testers were asked to create a system model focusing on dragging the right element type to the diagram and also applying the correct stereotype to them and ignoring the other architectural rules.

<table>
<thead>
<tr>
<th></th>
<th>Task A</th>
<th>Task B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum total number of elements</td>
<td>49</td>
<td>45</td>
</tr>
<tr>
<td>Minimum number of packages</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Minimum number of classes</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>Minimum number of edges</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>Minimum number of properties</td>
<td>23</td>
<td>21</td>
</tr>
<tr>
<td>Minimum number of operations</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Minimum number of stereotype to apply</td>
<td>16</td>
<td>17</td>
</tr>
</tbody>
</table>
The major goal for this usability study was to evaluate if the current DSM support resulted in any enhancement in usability of the ArCon and also I was interested to discover issues and the required enhancements for DSM support which facilitate the design process more for the system designers and minimize design errors.

### 7.2 Analyzing the data

The test intention was to study participant’s navigation and all actions which the participants took to complete their system design model. So, the usability issues that had arisen in the process of system design, either with or without any domain specific modeling support, were detected. Applying think aloud protocol, a situation is counted as a problem when verbal indicators which express a problem is monitored, such as incomprehensibility, doubt, confusion and any annoyance relating the use of the tool.

For this study, I collected usability problems through questionnaires, participants’ verbalization and my owned observation. Then I classified, measured and analyzed the data with a focus on a few usability problems called variables in my study. In general, four variables are analyzed and discussed in this chapter:

1. Efficiency: is called as effort and complexity
2. Error rates
3. Reliability
4. User satisfaction

Reliability in this study means the level in which the evaluators could rely on ArCon DSM suggestions and could progress their design without any need to check the ArCon meta-model containing architectural rules. In the next section (7.3) each of the above variables are discussed in details and results are presented in form of bar chart. Apart from the mentioned usability problems, participants were also encountered with other problems which were not included in results, such as troubles caused by their unfamiliarity with Papyrus tool or freezing the Eclipse environment.

### 7.3 Participant experiences

In this section, all the usability issues which I found after performing the usability tests and analysing evaluators’ verbalizations and my own observations are presented in the scope of the five mentioned variables.

In sections 7.3.1 to 7.3.4, the overall distribution of each of the usability problem types, is defined and discussed. The results will be compared regarding if the participant used DSM support or not. So, the improvements supported by ArCon DSM support as well as the required improvements in order to make the tool more usable in the future will be discussed.
7.3.1 Effort and complexity

The issues which caused any confusion, difficulty and extra effort for the subjects during the design process are presented in this section. Based on user verbalizations and my observations, three main areas were detected as the source of confusion or difficulty:

1. Comprehending the architecture model (ArCon’s meta-model)
2. Papyrus user interface
3. The DSM support provided by ArCon

<table>
<thead>
<tr>
<th>Table 7.2. Three scopes which caused user confusion or difficulty during the test sessions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArCon’s meta-model (architecture model)</td>
</tr>
<tr>
<td>Papyrus user interface</td>
</tr>
<tr>
<td>The DSM support provided by ArCon</td>
</tr>
</tbody>
</table>

Table 7.2 shows the detailed explanation for the cases which happened regarding each of the problem scopes caused confusion and extra effort. The data collected from the test sessions is represented in the format of bar chart in the figure below. Please note that Papyrus interface is not related to the treatment variable in this study. But it is presented in the results in order to provide a comprehensive source of usability issues related to efficiency metric.

A few problems in Papyrus user interface which caused user confusion was detected in this study. For example, when testers applied a stereotype to an element sometimes the stereotype didn’t show up in the diagram editor. Beginner and intermediate users had considerably more confusion because they were not experienced with Papyrus. Clearly, they couldn’t interact with it easily. For example, evaluators had difficulty to find the right spot in Papyrus user interface in order to apply stereotype.

Unfortunately, the DSM support itself resulted in confusion for the testers. One of the cases of confusion repeated for all the tester levels. They kept dragging and dropping elements from ArCon’s palette.
assuming that DSM support provides multiplicity checking so the palette would be empty whenever all
the required elements were added.

Another confusion caused by DSM support was observed within intermediate and novice testers. In that
case, the suggestions on the palette were different from the participant’s own interpretation of the
architecture model. This condition resulted in both correct and wrong decision on element type. Mana,
one of the beginner testers, took more time to double check the architecture model. Since she was not well
familiar with the rules she selected the right element type presented on ArCon palette. Joel, one of the
intermediate participants, faced with the same situation. He was confused for a while and then he assumed
that there might be a bug in the software and that is why the palette showed a wrong element. So, he used
Papyrus palette in order to add the element which he thought was correct but in fact was wrong. When I
asked him why he made that mistake, he stated that: “the palette doesn’t provide enough explanation in
order to lead the user. That is why the user won’t be convinced and wants to figure it out himself/herself.”

The last but not the least source of confusion was the architecture model. In ArCon, the architecture
model is a meta-model for the system model. All the architectural rules for the system model are defined
in this meta-model. Since ArCon used UML elements for describing the architectural rules where the
elements have a different meaning than the standard UML the participants, specially non experts, had
difficulty to interpret that meta-model and extract the architectural rules correctly. It was surprising that
even one of the experts also experienced this type of confusion.

![Figure 7.1. Task complexity](image-url)
As the statistics show, the confusion distribution was roughly similar for intermediate and beginner levels either they got treatment or not. But in general, they had significantly a higher frequency than the advanced testers and the treatment was effective in decreasing their confusion and effort.

In sum, DSM support had a positive effect on design process with regard to declining the risk of user confusion in the case they try to find out what a construction in the architecture model stands for.

7.3.2 Error rate

In this section error rates is represented and compared regarding if the testers performed their tasks with or without getting the treatment. The purpose is to find out if the DSM support could help the testers to decrease the error rate. In order to measure this variable the most important thing was to define clearly what was counted as an error in the experiment. Based on the scope of my usability tests explained in section 7.1, and my observation during the test sessions, I found four types of errors in general which are listed in table 7.3.

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The participants made a wrong decision on an element type, element stereotype or element multiplicity.</td>
<td></td>
</tr>
<tr>
<td>The participants forgot to add required element to a fraction.</td>
<td></td>
</tr>
<tr>
<td>The participants forgot to apply stereotype to an element.</td>
<td></td>
</tr>
<tr>
<td>By mistake, the participants dragged a wrong element from Papyrus palette to the diagram.</td>
<td></td>
</tr>
</tbody>
</table>

The result of the study on error rate variable either the testers got treatment or not is represented in the bar chart below. As the statistics show, although there was a slight improvement for advanced group using the DSM support, in general it didn’t help the advanced and intermediate evaluators to make a more error prone design. In addition, error rates for the two user levels were roughly similar either they took treatment or not. Despite of the intermediate and advanced levels, DSM support had a noticeable effect on the beginner testers with regard to error frequency.

With respect to the error types observed during the test sessions, there was a significant difference between advanced and beginner levels. The advanced participants made mistakes regarding the element multiplicity or stereotype which seemed be the cause of tiredness or distraction but not as a result of the lack of their knowledge. In contrary, beginners’ errors distributed to all error categories presented in table above. This can be the reason why DSM support could considerably improve the beginners performance considering the error rate variable. In stands to reason that, a filtered list on ArCon palette can lower the risk of some of the mentioned errors in table 7.3. For example making wrong decision on the element type and also the possibility of dragging the wrong element type from the palette will be omitted if users use ArCon DSM support, because ArCon’s palette never shows any wrong element for any fraction of the system model.
This section presents the level in which the participants found the DSM support reliable. A complete description of possible scenarios for DSM support reliability is shown in table 7.4. It is noticeable that most of the cases in this category were collected by observation.

| DSM reliability | The participants added the suggested element on the palette before they look at the architecture model. |
| DSM support ignorance | The condition in which the users made a mistake in design process while they could avoid it if they used the DSM support. |

The collected data is presented in form of bar chart in the diagram below. The results of the study show that the advanced group relied the most on DSM support. Considering the participants in my experiment, the advanced participants relied on domain specific suggestions on the palette three times more than the beginner and intermediate testers.

It is noticeable that the advanced testers relied on the palette mostly toward the end of the test session because they knew the meta-model well by then and didn’t need to check it again. As a result, the palette was just a reminder or confirmation to their own solution. So they dragged an element from ArCon’s palette without checking the architecture model. That is why the number of time they relied on ArCon palette is considerably more than the other two groups.
Regarding the reliability on the palette among beginner and intermediate participants, dragging the elements directly from palette to the diagram without checking the meta-model wasn’t always because of their complete trust on DSM support. Similar to advanced testers, they did it because after a while they knew the architecture model and what they needed to add to a specific fraction on the diagram. But since they were not experts they had to check the meta-model more often.

For beginners, the palette provided a good guideline for them in order to interpret the parts of the meta-model where they were unfamiliar or uncertain about their decision. Therefore, in addition to reliability, DSM support for novice evaluators was also a help in order to learn more about ArCon’s rules and constraints.

Apart from differing in terms of the reasons for ignoring the DSM support, all the categories resulted in equal frequency. While tiredness and losing the concentration was observed as the reason for this issue among beginners and advanced groups unreliability to the tool was the cause of ignorance among intermediate group.

Every participant, did a final review on both architecture and system models before they announced that they were done with the task. For the task in which the participants got DSM support, I was looking forward to see if they used the ArCon palette as guidance when they were checking different fractions of the system model. But, none of the testers used it unfortunately. This case was not taken into account as “DSM support ignorance” category since in this study the DSM support wasn’t meant to be used in that way. Instead it was supposed to be a help to facilitate the design process. The only reason for mentioning it here is to consider it as a factor which can be used for future enhancements.
To conclude, the new DSM support couldn’t guarantee reliability to the users. Based on my observations and their verbalizations, the reason is that it didn’t provide comprehensive information on the domain. So, the users couldn’t progress designing by just relying on this support. Therefore, they tended to deny it or rely on it when they already knew well about the architectural rules.

### 7.3.4 User satisfaction

Table 7.5 represents a complete description for two main issues related to user satisfaction variable discussed in this section i.e. participant’s verbalization about their satisfaction of DSM support and also participants claimed for more domain specific modeling support.

<table>
<thead>
<tr>
<th>Participants verbalized their satisfaction of DSM support</th>
<th>Participants stated that the short list on the palette provided by ArCon DSM support is convenient and they liked that they didn’t need to scroll a long list on the palette. Participants complained about the long list of element on Papyrus palette. Participants mentioned that the palette helped them to make a right decision for required element type.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participants verbalized that they need more DSM support</td>
<td>The participants mentioned that they need more DSM support in order to design easier.</td>
</tr>
</tbody>
</table>

The collected data through observations and verbalizations during the test sessions related to the mentioned scopes resulted in diagram below.

![Figure 7.4. User satisfaction of DSM support](attachment:image.png)
With regards to the overall user satisfaction, generally the participants in this study liked DSM support since the number of times they verbalized their satisfaction of the support is considerable. In addition, there was no significant difference between the level of satisfaction between advanced, intermediate and beginner participants.

Apparently, the overall impression of DSM support on the users didn’t relate to their level of expertise. The only difference is the manner in which they liked it and verbalized their satisfaction. Since the advanced testers were well familiar with the tools, they were more affected by the short list on ArCon palette. In another word, the ease of using a short list on the palette instead of scrolling a long list of elements was the biggest cause of satisfaction among the advanced users. On the other hand, the beginners were mostly satisfied with the help they took from DSM support in order to either make the right decision or prevent them from making any mistake. For them there were more big challenges than being annoyed by scrolling the element lists. The most surprising verbalization in this experiment was stated by one of the beginners. He said: “The DSM support helps me not to think as an idiot!”

Several participants commented that although the current DSM support can be considered as an enhancement, they prefer to have more DSM support in order to do their design with less confusion and fewer errors. As it was expected, the beginner participants verbalized and asked for more DSM support more than the other two groups. Clearly, the reason is that they were almost new with Papyrus and ArCon. The most important features that all participants asked for are:

1. Get also a filtered list of only permitted stereotypes.
2. In addition to the element type, DSM support should provide the permitted number of elements for each element type. For example, if a class should contain three operations, the palette shouldn’t show the operation on its drawer after adding all the three operations to that class.

### 7.4 Questionnaires

As mentioned in section 6.5, participants were asked to fill in three questionnaires: one post task questionnaire after completing each task and one post session questionnaires which was given to them at the end of their test session.

Concurrent think-aloud is an effective observing method to evaluate usability. But since the users sometimes stop verbalizing during the test session, I used post task questionnaires in order to extract more data (Thomas Tullis, 2008). Post task questionnaires were intended to give some insight into the task the users performed. In contrary to post task questionnaire, the goal for post session questionnaire was to let the testers compare the two tasks based on their overall experience during the test sessions.

The two types of questionnaires were designed based on 7-point Likert scale ranging from “Strongly agree” to “strongly disagree”. Furthermore, some open ended questions were provided in each questionnaire to let the users to write down their comments or suggestions. In-depth rating and open ended questionnaires are an effective overall evaluation. Because the tester has fully interacted with the product and can evaluate it more precisely (Thomas Tullis, 2008).
The questions in both questionnaire types were designed with a focus on three scope of interest: task complexity, reliability and user satisfaction. Moreover, post session questionnaire contained a few questions in order to let the users to score the usefulness of ArCon in general.

In order to analyze the rating scale data, I assigned a numeric value to each of the scale positions as weight. Depending on the question, I assigned either 1 or 7 to “Strongly agree”. Regarding each variable for each participant group across the two tasks, related questions were analyzed by summing the weights and then computing the average points.

### 7.4.1 Effort and complexity

As the diagram below shows, in general, the evaluators believed that DSM support decreased the design complexity and effort. Despite the advanced and beginner testers who scored DSM roughly 50% effective in decreasing the complexity (Beginners: \((6.5-2.9)/7=0.51\), Advanced: \((5-2.1)/7=0.41\), intermediate testers found the support less effective \(((5.6-3.9)/7=0.24)\).

![Figure 7.5. Participants rating on task complexity](image)

Similarly, through post session questionnaire, intermediate users gave the highest score of tool complexity to DSM support. It means that they found design process still complicated even when they received treatment. In contrary to intermediates, through post session questionnaire, beginner participants gave the highest score to the usefulness of DSM for decreasing the system design complexity. It seems that this small DSM support was a good complimentary to novice’s knowledge. So they could design their models with less confusion and work.

The average of overalls scores in post session questionnaires is 3.2 out of 7 \(((1.2+5.8+2.7)/3=3.2)\). It shows that DSM support was successful in order to minimize the complexity but the level is still high and there is a need for more improvement in the tool to omit confusion in design process as much as possible.
7.4.2 Reliability

With respect to reliability, DSM support provided by ArCon received considerably a high score. In general, beginners found the DSM support less reliable than the other two groups. The same result for beginners was observed and collected during their performance. One explanation for this fact could be that beginners had lack of knowledge and experiment. So, they needed more data and guidance in order to be convinced and rely on the DSM support.

![Figure 7.6. Participants rating on DSM support reliability](image)

7.4.3 User satisfaction

With regard to user satisfaction, the DSM support was scored almost high by all participants. Although the beginners scored DSM support lower than the other two groups the overall average of user satisfaction was noticeably high, roughly 5.2 out of 7.

One of the open ended questions was aimed to let the testers explain the reason they were satisfied about the support. All the comments are listed below:

- I liked the focused palette with short list.
- I like it hides the tools which I don’t need.
- I could easier make decision in the type of element which saved me time.
- It increased my efficiency.
- I don’t have to look so much in the architecture model.
- I liked it helping me to know what to add.

Based on the testers’ verbalizations, beginners got the most advantage of the palette by getting guides in order to understand the meta-model better. Therefore, the treatment decreased their confusion and caused a high satisfaction. In contrary, the short list and ease of scrolling it was the most interesting aspect of DSM support among advanced evaluators.
7.4.4 Usefulness

In post session questionnaire, some questions were presented in order to let the users to evaluate ArCon tool based on their experiment during the whole test session. My purpose was to deliver an overall user evaluation about ArCon to Combitech. Since the result could be specifically interesting for them in order to find out how the different target users would rank this tool.
Fortunately, all the participants ranked ArCon’s usefulness more than 6 ((6.5+6.7+6.3)/3=6.5). The result shows that the project has been successful and if some enhancements, including the usability problems which were detected in this study, will be implemented the tool could be significantly popular among the system designers.

7.4.5 Participants suggestions for ArCon enhancement

Among open ended questions, there was a question which was intended to let the participants express their opinions and ideas for DSM support enhancement. In general, the collected results are listed here:

- I need a tool to tell me clearly what I should do.
- I need also support on stereotypes. Because the lack of it lead me to confusion.
- I need the palette check multiplicity for the elements.
- The tool really helped me when I knew what to do. But it wasn’t helpful when I didn’t know what to do. DSM support should represent more information than just permitted element types.
- When I get stuck it doesn’t give further information.
- Give me some more information about how the architecture rules transfer to system design.
- It would be nice to get info about multiplicity when adding elements.
Chapter 8

Conclusion

8.1 Conclusion

The contribution of this master thesis was to implement a Domain Specific Modeling DSM support for
ArCon aiming at increasing the usability and enhancing the user performance. As a complementary to the
work, a usability study was planned and performed to discover if any enhancement in usability and user
satisfaction level was resulted by the provided support.

In order to control and minimize the other variables in this study, the test was limited to only those types
of architectural rules which were related to the scope of current DSM support implemented in ArCon i.e.

element type, element stereotype and element multiplicity. During the whole test sessions, four types of
variables; efficiency, error rates, reliability and user satisfaction, were measured based on user
verbalization and my own observations.

Considering the results, DSM support seemed to be successful in increasing ArCon’s usability since fairly
high user satisfaction was both observed during the test sessions and scored by the participants. It had the
most effect on decreasing user confusion and task complexity (improving efficiency) during the design
process for beginner and intermediate evaluators. Unlike, it didn’t make any improvement for reliability.

With respect to error rate, DSM was helpful only for beginner participants while no improvement was
observed among intermediate and expert evaluators. In stands to reason that, the level of information
ArCon’s DSM support provides is lower than the level of knowledge of non-beginner users. So it often
doesn’t provide extra information for them during design process in order to help them avoid mistakes.

As a result of lack of information that ArCon’s palette provides, it was observed that the DSM support
can be sometimes the source of confusion. Because it cannot support the users with enough information to
let them know why the element types it shows are the correct ones.

Generally, all the participants had to move often back and forth between system model and architecture
model. The reason was that they couldn’t memorize all architectural rules such as element multiplicity or
stereotype names. This made the design process time consuming and complicated resulting in user
frustration. In the real world, users face with a more complex and huge architecture model. So,
memorizing the rules is harder and users will spend more time on shifting between system and
architecture model to check the rules. To conclude, DSM support was shown to have fairly high effect on
ArCon usability especially among beginners with ArCon. But in order to provide a considerable higher usability level for all types of ArCon users, some improvements is needed. The focus should be on increasing the information which ArCon DSM support provides to the end users.

However, I must sound a note of warning. I am not an experienced usability conductor. So, despite the fact that I tried my best to be as precise as possible, definitely the tests I conducted are not guaranteed to conclude in precise results. On the other hand, if the test were conducted in a usability lab with advanced equipment such as screen video capturing, more precise data could be collected. The last but not the least, due to limitation of resources the sample size in this study might be insufficient but still effective in discovering the main usability issues.

8.2 Further research and recommendation

The basic idea for ArCon was to help Model-Driven Development in order to automatically check system models against the architectural rules. This imposed both the architect and the system designer to learn how the architectural rules should be presented in the form of UML model in ArCon. Although ArCon has automated validity checking of the system model it is still hard for designers to follow the rules in another meta-model called architecture model. The goal of this master thesis was taking the first step to solve this difficulty for ArCon users. The idea was to provide a domain specific modeling support only for indicating the type of required elements during the design process.

Despite the current support resulted in high user satisfaction, there is still the need for a higher level of DSM support. The most highlighted requirements resulted from my usability study are concerning the demand for getting support on permitted stereotypes and also getting hint about element multiplicity. But there are other rules such as naming constraints, permitted end point element for an edge or the number and properties of parameters belong to an element of operation type.

In my idea, the ideal situation is when all the architectural rules can be transferred and presented on GUI while the designer is configuring the system model, as it was provided for just the element type in my master work. Because it helps the designer to be the least engaged and confused by ArCon’s meta-model and interpreting the architectural rules. Furthermore, possible errors will be avoided and the DSM support will become more reliable.
APPENDIX A: Task A

The architecture model was composed of four numbers of diagrams. Snapshots for all the diagrams are presented here.

![Diagram]

Figure 0.1 Task A-Diagram a
Figure 0.2. Task A-Diagram b

Figure 0.3. Task A-Diagram c
Figure 0.4. Task A-Diagram d
APPENDIX B: Task B

The architecture model was composed of five numbers of diagrams. Snapshots for all the diagrams are presented here.

![Diagram](image)

Figure 0.5. Task B-Diagram a
Figure 0.6. Task B-Diagram b

Figure 0.7. Task B-Diagram c
Figure 0.8. Task B-Diagram d

Figure 0.9. Task B-Diagram e
### APPENDIX C: Questionnaires

#### Table 0.1. Post task questionnaire

<table>
<thead>
<tr>
<th>Post-task questionnaire</th>
<th>Strongly disagree</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>Strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Complexity in decision making</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I think I need the support of an expert or more domain specific modeling support from</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>the tool in order to be able to design system models correctly.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>If I use this tool, as a system designer, I have no confusion or uncertainty on</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>deciding what type of element I can add to a fraction.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I needed to check the architecture model carefully before I could add any element to</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a fraction on the diagram.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The system supports all information I need in order to create system model without</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>any confusion.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I need more domain specific modeling support. So I can be sure about the validity of</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>the system model that I create.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Effort</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Making decision on the type of element which I like to add to a fraction of diagram</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>is easy by help of this tool.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A significant part of task effort was checking the architecture model to find out what</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>type I could add to a fraction.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Reliability</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tool suggestions on palette and popup are reliable and I feel certain adding any of</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>those to the selected fraction on the diagram.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Reliability and user satisfaction</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I could find an element quickly in the lists on palette or popup menu.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall, I am satisfied with the amount of time it took to create the model using</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>this tool.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall, I am satisfied with the ease of creating a valid system model with this tool.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall, I am satisfied with the supporting information from the tool to in order to</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>create valid system models.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Open-ended questions</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Do you have any suggestion to make any change in the current tool in order to</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>enhance user performance and ease of task completion and succeed?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 0.2. Post session questionnaire

<table>
<thead>
<tr>
<th>Post-session questionnaire</th>
<th>Strongly disagree</th>
<th>Strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Complexity in decision making</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>By ArCon, it requires less effort to create a valid model.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>It was more comfortable creating system model by ArCon’s support.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ArCon saves me time on creating a system model.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ArCon support helped me to get less confused while creating the models.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I like domain specific modeling support more for making decision easy rather than shortening the list on palette.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>User satisfaction</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I am satisfied with domain specific modeling support from ArCon.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Usefulness</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I found ArCon useful.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I will use it in the future.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I believe system designers will need it.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I will suggest ArCon to friends.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Reliability</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I found ArCon reliable.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Open-ended questions</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>What you liked most and what you liked least about ArCon?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>What did you find the most useful about ArCon?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Do you have any suggestion in order to enhance the domain specific modeling support by ArCon?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX D: Usability test instruction and DSM user manual

I. Test process

The whole test session is divided into the following sections:

1. Training session: you read this manual and then we will have a discussion about the manual’s content to be sure that you know as enough as being able to start the test. A step by step example is provided in section 5. But I will offer you more simple examples which you can practice and get familiar with both Papyrus and ArCon.

2. Test session: I will give you two architecture models (defined by ArCon’s constraints) and ask you to create a system model for each. One should be done with and another without ArCon’s help. The tasks are similar to the examples we work together during training session but they will be more challenging.

3. Questionnaires: After each test I will ask you to fill a questionnaire and after completing the two tasks you will have a general questionnaire to fill.

Please:

1. Ask questions during the test session if you think you are stuck but don’t expect to get always a clear answer.

2. When using ArCon and you need to add an element to a fraction, trust on the types of the elements which ArCon provides you on the palette or popup menu. They are all permitted for you to add to the selected fraction. Please don’t waste your time on finding out by yourself or checking its validity.

3. Don’t be scared of making mistakes! You are not being tested, the system is. So, you are here to help me to find possible confusions or the mistakes which the users can make when they use this tool.

4. Please fill the questionnaires carefully. Since we would like to find ArCon’s problems and deficiency that will be great if you can express what you will discover in the questionnaires.

II. ArCon

Software architecture model is one of the important design artifacts which guides system design so that it meet the desired system quality. In Model-Driven Development (MDD) the architecture model is defined in the form of packages and components designed in a high level structure. But generally the architectural design rules and constraints which should be followed in design process are defined by architect in an informal text file.
Therefore, for a detailed system design the risk of misinterpreting or getting confused by text formatted rules is high. So, a manual review is required to check if all architectural rules are met. This manual checking should be done by architect and it causes low review quality, longer software life cycle and still risk of making a mistake by the reviewer.

To solve the problem, Mattsson, 2009, proposed an approach to formalize and model the architectural design rules in an abstraction level in an UML model (called architecture model) instead of text files, where UML constructs have a different meaning than the standard definition. As a proof-of-concept, ArCon (Architectural Conformance checker) tool was developed by Combitech to validate system models against the architectural rules defined in an architecture model. Expressly, ArCon reads the system models from an UML tool and checks them against a set of architectural rules defined in another UML model (architecture model). Finally, it produces a report of any violations of architectural rules.

III. Papyrus

Papyrus is an extension in Eclipse platform. It is a component of the Model Development Tools (MDT) and can be used as a standalone tool. It offers an environment for editing any kind of EMF model especially in the form of UML2 (as defined by OMG). Currently Papyrus is selected as an open source UML modeling tool which ArCon can support.

IV. Study motivation

By following ArCon and formalizing the architectural rules designers can check their model validity after they create their system models. But there is always the risk for the designer to violate architectural rules or get confused while creating the design. Particularly, ArCon could tell them what errors they had made in their model after they completed the design. But it couldn’t guide them step by step during design process in order to help them to avoid those mistakes.

Therefore, the aim for my thesis work was to provide support during the design process. So, users can get domain specific guides via user interface while they are modeling their own design for the system.

The current Domain Specific Modeling (DSM) support provided by ArCon is offering the types of element the designer is permitted to add to a specific fraction of the diagram.

Without any DSM support, users need to check the architectural model regularly in order to find out what type of element they should add to a fraction in their design model. But using the DSM support offered by this work, ArCon checks the architecture model and offers the permitted elements for the fraction you select on your system model. So, it makes design process easier and less erroneous.

The aim of this usability study is to measure a number of performance metrics and also the level of user satisfaction with and without DSM support from ArCon. So we can see if any improvement in usability has been achieved by this work.
V. ArCon user manual

V.1. Creating dependency to architecture model(s)

ArCon needs Architectural rules in order to analyse a system model. ArCon can read rules from one or more Architecture models and analyse the system with respect to the union of rules represented in those models. If an additional rule should conflict with a previously given rule, ArCon shall present a warning to the user and discard the new rule. The designer of the system needs to specify which Architecture models should be applied to the project. This is done using dependencies in a separate diagram in the system model.

To define a dependency to one or more architecture model in your system model, open the system model and follow the following steps:

1. Create a new class diagram in your model.
2. Drag the system model from model explorer view into the new diagram.
3. For each architecture model add an element of type Model to your diagram. Set the focus on the added element on diagram editor. Open properties view. Go to advanced tab and set the URI to the path to corresponding architecture model.
4. To create dependencies from your system model to another architecture model use a Dependency edge connecting two models. You should also use Dependency for creating hierarchy of architecture models.
69

Please note that for the architectural models in the same level such as Arch_Model1 and Arch_Model2 in the snapshot above there is no certainty that which will be read and checked first by ArCon. The order can have influence in case there are conflicting rules.

V.2. An example showing how ArCon works

Before showing how DSM support works, please look at the example below in order to have a more clear view on how ArCon works. Furthermore, this example can illustrate how this feature can help you to model easier, because it decreases the possibility of making mistakes.

Following the arch model presented in the figure 8.11, you can create a system model presented in figure 8.12. A package called “Inheritance_impl” is added and the stereotype “Relationship_pkg” is applied to it. Since there is an association between Relationship_Pkg and Inherited in the arch model, you are allowed to add a class with stereotype “Inherited” to this package in system model such as “Inherited_Impl2” in figure 8.12.
In order to know what sub-elements and edges you can add to the class you should look at the arch model. Inherited class in this example has a comment, a few numbers of operations and properties and a realization edge to class “Heat_Sensor”. In addition, it has a generalization edge to “Inheritance”. Based on ArCon’s constraints this means that all properties and sub-elements belonging to Inheritance should be inherited by “Inherited” class i.e. two properties and a generalization to Water_Sensor. Therefore, the only elements you should use are comment, property, operation, generalization and realization.
V.3. DSMsupport

In order to use DSM support from ArCon:

1. Download and add ArConDS.jar file to the eclipse plugins folder.
2. Open or create your system model file using papyrus.
3. Apply required profiles to your model.
4. Create a class diagram for architecture dependency in your model, following instructions in section V.1.
5. Save the model. Now, you will have DSM support by three components explained below. If you made any modification to the architecture dependency diagram or the profiles you applied, you need to close the model and open again.

You will have support on two parts: popup menu and palette.
V.3.1. Popup menu

If you select a fraction on the Model Explorer view and right click on it you will see an additional submenu called "ArCon New Child" which shows only permitted elements for that fraction.

Figure 0.13. Popup menu supported by Papyrus
Note that, Papyrus doesn’t show the edges on the popup menu in model explorer. Instead, the edges can be added on diagram by drag and drop from the palette. To provide a comprehensive DSM support, permitted edges are also included under “ArCon new child” menu. But since they shouldn’t work there, they are gray and cannot be executed. You can see the difference between ArCon and Papyrus menus in snapshots 8.13 and 8.14.

Figure 0.14. Popup menu supported by ArCon DSM support
V.3.2. Palette

Two drawers are added to the palette called “ArCon Nodes” and “ArCon Edges”. You have the possibility to hide these drawers if you don’t need them. Please see Papyrus user manual in order to learn how to hide/show palettes. Figure 8.15 compares the palettes provided by Papyrus and ArCon. As you see, for the example explained before ArCon shows only the five elements.

Figure 0.15. A comparison between Papyrus and ArCon Palettes

Similar to popup menu, by using ArCon’s palette you don’t need to check and analyze architecture model in order to find out what types of elements you are permitted to add to the selected fraction on the
diagram. Moreover, you won’t be bothered with a scrollable long list of elements. You will get just the elements that you need on ArCon’s palette and popup menu.

Please note that filtering on palette is done based on the fraction that is selected in the diagram editor while the list on popup menu is created based on the selection on the model explorer view.

If you for some reasons need to break the architecture rules, you can always use the Papyrus standard palette or popup menu.

For a complete user manual please see the link below:
References


77

På svenska

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extra-ordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

In English

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© [Författarens för- och efternamn]