Making a usable modeling tool for the distributed tool architecture promoted by a Linked Data approach

Da Zhang
Abstract

Making a usable modeling tool for the distributed tool architecture promoted by a Linked Data approach

Da Zhang

A tool chain is a distributed tool architecture. It is formed by the tools, in this case software systems, which are integrated by means of a linked data approach. This tool integration approach enables data of tools to be integrated by linking data according to semantic relations. Tools in the chain share the linked data by following the OSLC specifications.

During the model-based tool chain development, the tool chain model need to be visualized to provide good support for both designing and collaboration. Given this background, the problem addressed is how to make a usable modeling tool for modeling the OSLC tool chain. This report presents the methods used, the experience of designing the modeling tool and diagrams as well as the results from a usability test of the resulting prototype.

Based on the encouraging results of the usability test, the approach used in this research can be seen as a possible approach to make a usable modeling tool for modeling the OSLC tool chain.
Contents

4.1.1 Affinity Diagram .............................................. 22
4.1.2 Design visioning and Prototypes ............................ 26
4.2 Implementation Phase ........................................... 30
   4.2.1 EMF structured data model ................................. 30
   4.2.2 User Interface Implementation ............................ 32
4.3 Post-implementation Phase ..................................... 37
   4.3.1 Task Completion Result .................................. 38
   4.3.2 SUS Questionnaire ......................................... 38
   4.3.3 Users' comments and suggestions ....................... 38

5 Discussion ................................................................ 41

6 Conclusion .................................................................. 43

7 Future Work ................................................................ 45

A Moderator Script .......................................................... 46
B Test Task ..................................................................... 51
C System Usability Scale Form ........................................ 54

Bibliography .................................................................... 56
Chapter 1

Introduction

1.1 Background

This thesis project was conducted at Scania Tekniskt Centrum located in Södertälje, Sweden. It is a part of ESPRESSO project[1], a collaborative project between Scania and KTH. The ESPRESSO project aims to develop and adapt model-based techniques that improve the quality and reduce the cost for development of embedded systems in trucks, and especially safety critical systems.

The development of embedded systems in the truck requires the collaboration of various specialist teams and engineering disciplines. Many software systems and tools are used by engineers in their work in order to meet these engineering disciplines.[2] In this multi-disciplinary development environment, software systems and tools need to be integrated to support the collaboration among different specialist teams. Productivity in the development process can be improved in a large extent if these systems and tools are integrated well[3].

A distributed tool architecture following the Linked Data[4] approach is promoted to be used in the tool integration solution of ESPRESSO project. In this architecture, instead of building a centralized database which managing data of all tools, each tool follows the unified rules to share data with each other. Tools are linked together as a tool chain, where tools are integrated in data level.

During the development of a tool chain, the tool-chain model plays an important role. A set of rules and definitions like OSLC specifications[5] are used to describe data of tools. Since data of tools changes frequently during the tool-chain development, a model of tool chain is necessary for keeping the consistency of data descriptions and making the change traceable. In addition, tool-chain models are usually used in different tasks of
the tool-chain development, like communications between stakeholders, design validation and architecture analysis, etc.

The study of this thesis is about how to make an understandable tool-chain model and how to implement a usable graphical modeling tool for managing the model.

1.1.1 Linked Data

Linked data\[4\] are data linked through web techniques. Tools and software systems are able to share and explore others’ data by using linked data techniques. It is an approach to build semantic web\[6\]. Semantic web promote the common protocol and technologies used by tools to expose, share their data and consume data of other tools through web. In semantic web, data should be linked according to their semantic relations. For example, a student takes a course. ”Take” is seen as one of the semantic relations between the student and the course.

1.1.2 Resource Description Framework (RDF)

Resource Description Framework is usually called with its abbreviation, RDF. It is the specification defined by World Wide Web Consortium (W3C)\[7\]. It includes a variety of syntax notations and data serialization formats used to model the information of the web resources. It is the fundamental technique to implement Linked Data. Linked Data are provided and consumed in RDF format. RDF is also used for describing how data are linked.

1.1.3 Open Services for Lifecycle Collaboration (OSLC) specifications

Open Services for Lifecycle Collaboration\[8\] is an open community building practical specifications for integrating software, abbreviated as OSLC. It defines a list of specifications called OSLC specifications\[9\], which are used for integrating tools and software systems with the approach of linked data. In this thesis project, all tools and software systems are integrated by following the OSLC specifications.

1.1.4 Tool Chain

A tool chain formed by tools integrated by following OSLC specifications\[10\] is called OSLC tool-chain. See picture below,
In the picture, the big circle container represents tool. The items contained in the circle represent data managed by the tool. After all tools have followed OSLC specifications to publish their data, all data have been linked and shared through this tool-chain. It means that a data level integration has been made among these tools. This picture explains how a tool chain is formed and how linked data approach is applied in tools integration.

1.2 Purpose

To realize the model-based approach to system integration, an easily understandable data model of the tool chain and a tool for managing this data model are foundation in the first step. Collaboration between developers from different teams is very important especially in a multi-disciplinary development environment. An easily understandable model can give developers a better understanding of the problem they discuss about. Besides, a data model management tool provides efficient ways to update the data model when changes happen frequently especially in the design phase of tool chain.
In this thesis project, a visualization of the tool chain model will be created. Also a data model management tool will be designed and developed to allow users to create and edit data model information graphically.

At the end of thesis project, the developer and other stakeholders of tool integration should be able to use the modeling tool to model the tool chain used in the use case on which this thesis project is based.

1.3 Research Questions

The overall goal of this thesis project is to model the tool chain and to visualize the model as model diagrams. Moreover, users should be able to manage information of the model through a modeling tool. Usability of the modeling tool should be considered during the whole development process. A key concept of a highly usable modeling tool is an understandable visualization. With this awareness, two research questions are formulated,

RQ 1:
What constitutes a usable system for modeling data of OSLC tool chain?

RQ2:
How can such a system be implemented?

1.4 Scope

The project was carried out between January, 2015 and June, 2015 at Scania Tekniskt Centurm in Södertälje. The scope of this thesis was mainly focused on implementing modeling tool of tool chain. The project was carried out in the process of literature studies, user centered design, implementation and usability evaluation.

1.5 Limitations

The thesis project has certain limitations. One of the biggest one is that it is a case study research based on a specific scenario. The context of scenario is truck manufacturing. So the results are not verified in other cases of tool chain modeling. Accordingly, generalizing the results to other scenarios and domains must be investigated.
Chapter 2

Related Work

2.1 Resource Description Framework (RDF) Model

As explained in the Resource Description Framework (RDF) Model and Syntax [11], RDF, the resource description framework, provides interoperability between applications exchanging machine-understandable information on the Web. It is a type of metadata used for describing web resources. The resources described by RDF could be anything that can be named via a URI.

RDF model is a model for representing named properties and their values. Both normal attributes of resources and relations between resources are represented. In the model, resources being described and values describing resources are nodes in a directed labeled graph. The arcs connecting two nodes is propertyType.

[resourse R] ---propertyType P---> [value V]

It is read as "V is the value of the property P for resource R", or left-to-right; "R has property P with value V".

For example, Bob’s full name is Bob Smith,

Bob --- fullname ---> "Bob Smith"

This is a triple in RDF model.

A collection of triples with the same resource is an assertions. Assertions are particularly useful when describing a number of properties of the same resource.
Bob --- fullname ----> "Bob Smith"
|                     |
|--- know ----> Lina |
|                     |
|--- livein ----> "Sweden"

In the document of Resource Description Framework (RDF) Model and Syntax[11], a directed graph is used for visualizing RDF triple. It uses eclipse to represent node, arrow to represent arc and rectangle to represent literal value such as string, integer, etc. Here is one sample from the document.

![RDF Graph Sample](image1.png)

Figure 2.1: RDF Graph Sample[11]

It tells that John Smith is the Author of the document whose URL is http://www.bar.com/some.doc”. In this example, the doc is resource represented as eclipse, author is the arc represented as arrow and John Smith is a string name represented as rectangle. the ”bib” before author is its namespace.

The Bob example above could be drew as the diagram below,

![RDF Graph Sample](image2.png)

Figure 2.2: RDF Graph Sample

In this example, Bob and Lina are assumed as resources having URIs, Bob Smith and Sweden are string values.

RDF model is used for modeling specific resources’ attributes and relations. And directed graph diagrams are understandable and scalable. However, the model of tool chain is the model of RDF model. It is a metamodel of RDF model. The tool chain model should be able to model resources, relations and values from the perspective of metadata. For example, the student element in tool chain model represents the student class. In RDF model, it only represents one student object. Despite RDF model can’t be used directly to model a tool chain, the tool chain model should be based on the RDF model.
2.2 Enterprise Architect

Enterprise Architect is an information modeling software system developed by Sparx Systems. It includes many build-in types of models of database, business process, RDF, etc. Users are able to do simulation and validation on models. Picture below shows its user interfaces,

![Enterprise Architect](image)

**Figure 2.3: Enterprise Architect**

However, it doesn’t have an OSLC tool chain model. A user can’t use the build-in model to model a tool chain. Some general models like UML model could be used instead. However, it brings many issues, like many OSLC specifications are not supported and some information would be missed in the model. Due to lack of customizability, enterprise architect can’t be used for tool chain modeling. But its user interfaces and layout design could be referenced in the design of tool chain modeling tool.

2.3 yEd Graph Editor

yEd Graph Editor is a diagram drawing tool developed by yWorks. It provides different unit elements with which a user draw modeling diagrams easily and quickly. In this tool,
these build-in elements could be added with some intuitive interactions like dragging and dropping. In addition, a user is able to customize elements and components of diagrams with different shapes, colors, sizes, etc. A modeling diagram can be exported as many common image formats. Picture below shows its user interfaces.

![Enterprise Architect](image)

**Figure 2.4: Enterprise Architect**

Since yEd Graph Editor’s high customizability allows users to draw customized modeling diagrams very quickly, the user can also draw the tool chain model diagram with this tool. However, due to unified rules for drawing tool chain model, people can only design and draw it all by themselves. It often makes others very hard to understand the modeling diagram, since different people have different preferences on how tool chain model should be represented. But interactions used in yEd Graph Editor for drawing modeling diagram could be referenced when making tool chain modeling tool.

### 2.4 Visual Thinking For Design

To make understandable model diagrams and develop usable modeling tool, the design should be based on general visual theories and design principles. Using approaches based on these theories is helpful to improve usability of the system for modeling tool chain.
This book contains general theories of information visualization and human’s visual perception. The approach to how color should be used in the design and the theory of how patterns are formed and recognized by human are used as references in the design of tool-chain model diagrams and modeling tool interfaces.

A tool-chain model usually contains several different types of elements like tool, property, domain, resource, etc. The model diagrams should includes all these elements. Based on theories of this book, many factors like colors, shapes, layouts of these elements in the diagram have influences on efficiency of reading and locating model information.

The book introduces approaches to how the efficiency could be improved. For example, increasing contrast between objects and their background makes objects more highlighted. This approach could be used in tool-chain diagrams. It enables user to focus on diagram elements where important information is usually contained. Distraction from diagram background can also be eliminated in most of cases. For another example, when searching information, our brain and eyes deal with visual information in the top-down process. In this process, part of our brain constructs a crude map of the characteristics of the information we need. And then another part of our brain will constructs a series of eye movements to all potential areas. This theory could be applied to the design of diagram elements. Combinations of colors, shapes and textures could be used to make characteristics of different elements. More easily distinguished characteristics enable users to find information more efficiently.

2.5 Lodlive project

LodLive project provides a demonstration of the use of Linked Data standards (RDF, SPARQL) to browse RDF resources. The application aims to spread linked data principles using a simple and friendly interface with reusable techniques[12].

Picture below shows a example diagram of Lodlive,

In the diagram, resources and resource properties are represented by dots and arrows. The links, semantic relations, between data are traceable and explorable. User is able to explore the network of linked data by clicking small dots.

Despite this is a diagram of RDF data rather than tool-chain data, the part of visualization of data links could still be referred in the tool-chain diagram. The traceable and understandable visualized data links are also very important in tool-chain diagrams.
Figure 2.5: Lodlive data diagram
Chapter 3

Methods

This thesis project can be divided to 3 phases. The first phase is user research. Its purpose is to figure out users’ requirements and design prototype of user interfaces. The second phase is implementation. In this phase, several possible solution are investigated and evaluated. The modeling tool will be implemented with the proper solution according to the evaluation results. The third phase is usability testing. In this phase, a semi-qualitative usability testing is performed with participants having knowledge of OSLC tool chain. The purpose is to find usability problems and requirements missed in the first phase. Methods of each phase will be illustrated and explained in details in the following part.

3.1 User Research - User-Centered Design Approach

3.1.1 User-Centered Design

The term of User-centered Design is used in Norman’s book named The Design of Everyday Things[13]. And it is widely used after his another book, User-Centered System Design: New Perspectives on Human-Computer Interaction[14]. In this book, Norman use this term to describe design based on the needs of user instead of secondary issues like aesthetics.

As described in Notes on User Centered Design Process (UCD)[15],

"User-Centered Design (UCD) is a user interface design process that focuses on usability goals, user characteristics, environment, tasks, and workflows in the design of an interface. UCD follows a series of well-defined methods
and techniques for analysis, design, and evaluation of mainstream hardware, software, and web interfaces. The UCD process is an iterative process, where design and evaluation steps are built in from the first stage of projects, through implementation.”

The UCD process is divided to 4 parts according to ISO 9241-210:2010 Human-centred design processes for interactive systems[16]. It is able to be iterated. See picture below,

![User Centered Design Process](image)

**Figure 3.1: User Centered Design Process, ISO 9241-210[16]**

The UCD used in this thesis project also follows this standard process. Due to time limitation, there is no iteration.

### 3.1.2 Rapid Contextual Design

Rapid contextual design contains a list of techniques used for realizing the UCD process. The book named Rapid Contextual Design: A How-to Guide to Key Techniques for User-Centered Design[17], as a handbook of key techniques for UCD, introduces how to perform rapid contextual design with step-by-step guides. It covers each step of the UCD process, including contextual interview in the first step, interview interpretation, work modeling, affinity diagram building in the second step, visioning, paper prototypes in the third step and paper prototype reviews in the final step. As mentioned in the book, people should adjust the approaches to their particular situations.

Techniques are explained in the order that they are used.

**Contextual Interview**
As explained in the book,

"Contextual Interviews are one-on-one interviews conducted in the user’s workspace that focus on observations of ongoing work. Conducting a thorough Contextual Inquiry interview is more than observing and recording the user’s current tasks. You want and need to discuss what is happening in the moment with the user."

In this project, the purpose of contextual interview is to understand how users model a tool chain, investigate the required information in a tool chain model and find how users use tool chain model in daily work.

3 contextual interviews had been done with 3 users. Each user represents one type of stakeholders of the modeling tool. The first type is the architect of a tool chain. He/she is responsible for designing tool chain architecture and build the tool-chain model. The tool chain model plays an important role in displaying and verifying architect’s design. The second type is the tool-chain developer. His/her work focuses on implementing and testing the tool-chain. He/she should know detailed specifications of resources’ definition and tool adaptors of the tool chain. The third type is the business process engineer. He/she verifies whether the tool chain is able to support workflows of the company.

Each interview session takes around 1.5 hours at interviewee’s workplace. During the interview, the interviewer act as a listener and a observer. And the interviewee explains his/her work related with tool chain model and shows how he/she does the work. The interviewer will ask questions when he finds something he doesn’t understand or some working cases he wants to dig more deeply. The audio of the whole interview is recorded and analyzed in the interview interpretation session.

**Contextual Interview Interpretation Session**

The purpose of interpretation session is to find and capture users’ requirements. Raw data of contextual interviews, like interviewer’s notes and interview audio recordings, are reviewed and interpreted. When any key point revealing user’s issue is found, it will be summarized and written down as affinity notes.

**Building an Affinity Diagram**

The affinity diagram is a hierarchical diagram grouping affinity notes under labels representing user’s requirements. It exposes key requirements and acts as the voice of users.
All affinity notes are in the bottom level of the affinity diagram. Notes will be grouped if they tell similar things. In this first round grouping, each group contains 3 to 6 notes. If any group contains more than 6 notes, it should be split into 2 groups. If any group contains less than 3 notes, it is better to regroup the notes. Each group is summarized according to notes’ content and in user’s tone. The second round grouping is based on the results of first round grouping and follows the same rules. Building an infinity diagram for 3 contextual interviews requires 3 rounds grouping. Finally, the infinity diagram should be able to reveal user’s key issues and core requirements.

**Visioning and Making Prototype**

A visioning is a hand-drawn, graphical representation. It shows how a function works in particular scenario. Prototype is based on the results of visioning. Prototype could be drawn by either pen and paper or prototyping tools. It should be able to tell how user interfaces look like and how interactions are performed. The prototype will be used in the prototype review interviews with users.

**Prototype review interview**

After the prototype is finished, The prototype review interviews are conducted with participants of contextual interviews. This purpose is to verify whether users’ requirements have been met. During the interview, interviewer shows how user can solve their problem or do their work with the new user interfaces. User’s feedback and comments will be used to improve the user interfaces.

**UCD Project Plan**

In this thesis project, 3 contextual interviews were conducted. And prototype of design were reviewed and updated. According to recommendations from the book[17], a 3 weeks project plan is made. See the table on next page,
### Schedule of UCD phase

<table>
<thead>
<tr>
<th>Day 1</th>
<th>Day 2</th>
<th>Day 3</th>
<th>Day 4</th>
<th>Day 5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>3 contextual field interviews, interview interpretation, affinity notes building</strong></td>
<td><strong>HALF DAY</strong>&lt;br&gt;First contextual interview (around 1.5 hour)</td>
<td><strong>HALF DAY</strong>&lt;br&gt;Interview interpretation and affinity notes building</td>
<td><strong>SAME as Day 1 for</strong>&lt;br&gt;second interview. Discuss findings and reset focus to ensure the quality of the data.</td>
<td><strong>ALL DAY</strong>&lt;br&gt;Continue unfinished interpretation and affinity notes. Start to analyze affinity notes and build affinity diagram.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Day 6</th>
<th>Day 7</th>
<th>Day 8</th>
<th>Day 9</th>
<th>Day 10</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Vision, identity UI elements, UI design, paper prototype</strong></td>
<td><strong>ALL DAY</strong>&lt;br&gt;Review found issues and come up with design ideas. Evaluate and discuss design ideas, UI elements, etc.</td>
<td><strong>ALL DAY</strong>&lt;br&gt;Finish design thinking and identifying UI elements. Start to build paper prototype.</td>
<td><strong>ALL DAY</strong>&lt;br&gt;Build paper prototype.</td>
<td><strong>ALL DAY</strong>&lt;br&gt;Finish paper prototype building and prepare interviews for prototype evaluation with users.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Day 11</th>
<th>Day 12</th>
<th>Day 13</th>
<th>Day 14</th>
<th>Day 15</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>3 prototype review interviews, discuss and make UI changes, finalize prototype.</strong></td>
<td><strong>HALF DAY</strong>&lt;br&gt;First interview. (around 1 hour)</td>
<td><strong>HALF DAY</strong>&lt;br&gt;Analyze and extract user’s feedbacks by interpreting interview. Discuss possible prototype changes.</td>
<td><strong>SAME as Day 11 for</strong>&lt;br&gt;second interview.</td>
<td><strong>ALL DAY</strong>&lt;br&gt;Finalize prototype changes and update the prototype.</td>
</tr>
</tbody>
</table>

---

**Figure 3.2: Project Plan of User Centered Design Process**
3.2 Implementation

Implementation discussed in this report is about techniques used for implementing the modeling tool.

Graphical tool-chain modeling is one of the most important requirements for the modeling tool. The modeling tool should enable user to model a tool chain and draw modeling diagrams through graphical interfaces.

Customizability is another important requirement. According to the UI design in last phase, elements of modeling diagrams need customized appearance, size and semantic meaning.

After evaluating possible implementing solutions, Eclipse Modeling Framework (EMF) is used to build the structured data model. In addition, Sirius, an eclipse project built on EMF, is a framework providing the graphical user interfaces. It is used for implementing UIs for creating and managing EMF structured data model.

3.2.1 Eclipse Modeling Framework (EMF)

Eclipse Modeling Framework (EMF)[18] is a modeling framework and code generation facility for building tools and other applications based on a structured data model. EMF is a common standard for data models on which many technologies and frameworks are built.

In the modeling tool, EMF is used for building structured data model of tool chain. The structured model contains data of tool chain model elements like resource, resource properties, domain, tool adapters, etc. The structured data model also contains data of relations, constrains and dependencies of the elements. For example, resource is contained by domain. EMF data model provides a reflective API for manipulating EMF objects generically.

The core EMF framework includes a meta model for describing structured models and defining rules, relations and constrains. A meta model is created and used to describe types and attributes of model elements as well as how elements should be related. For example, a tool adapter element has a attribute called adapter name. And the tool adapter should be contained by tool chain, resources should be managed by the tool adapter. All the information and rules are defined in the meta model and are applied to the generated structured data model.
3.2.2 Sirius

Sirius is a UI framework built on EMF structured model. In Sirius, elements in the structured data model could be represented by graphical objects. In addition, it includes interactions, like drag and drop, double click, move, resize, for managing the graphical objects. The interactions is able to be combined and customized to implement the structured data model management operations.

General model diagram is supported by Sirius. There are 3 basic graphical objects in the diagram. See the list below,

- Node
- Container
- Edge

Node is the most basic object representing the element in the structured data model. For example, a resource can be represented as a node. A node could be displayed as dot, square, ellipse, image, etc. An icon can be added, too. Picture below shows an example of node,

![Figure 3.3: Node Example](image)

The composition relation between elements could be represented by the container object. In the model diagram, a container can have nodes and sub-containers inside. A container can be displayed in 3 ways including gradient, parallelogram and image. There are 2 types of containers. One is list container. Nodes in the list container are shown as listed items. The other is free container. Nodes in the free container can be placed in any position. And nodes are shown as customized styles. Picture below shows both types of container,

![Figure 3.4: Container Example](image)
Relations between objects are represented as edges. Edge is a line with arrow at the end. There are two types of edges. One is element based edge. It represents an element in the structured data model. The semantic meaning of this element is a specific relation between nodes. The other is relation based edge. This type of edge shows when the represented relation exists. Picture below shows an example of edge.

![Figure 3.5: Edge Example](image)

In addition, there are tool components in Sirius. Functions and operations are implemented based on them. For example, creation tools of node, container and edge are used for creating both graphical objects of the diagram and elements of the structured data model. Reconnect edge component is used for managing edges and relations. There are many other tool components like double click, delete element, selection wizard, pop-up menu, etc. With these tool components, elements of the structured data model can be not only represented as graphical diagram objects but also created, edited and updated directly.

In addition to diagrams, treeview is supported, too. It displays the model in a hierarchy structure. And searching information in a treeview is efficient. Picture below is a treeview example.

As shown by the example, elements in the model are represented as items in the treeview. Icons can be attached to items. If there are sub items, the parent item can be expanded.

Sirius uses the viewpoint approach. With this approach, a diagram or treeview representation can be generated from the view of any element in the data model instead of the root element. Since a structured data model often contains elements from multiple domains, this approach enables only part of the model to be visualized or the same model to have multiple different representations for the same structured data model. For example, a tool chain model can be visualized both from the view of domain and from the view of tool chain architecture. Or a separated diagram is able to be created for each tool adapter with one structured data model.
3.3 Usability Testing

As Nielsen said in his book, usability testing is the fundamental usability method. It provides direct information about how people use the system being tested and what their exact problems are with concrete interfaces[19].

The reason to do usability testing is that design decisions are informed by data gathered from representative users to expose design issues so they can be remedied, thus minimizing or eliminating frustration for users[20].

In this project, usability testing is used as a method to test usability of the modeling tool. The result helps to verify whether the modeling tool is usable. It also exposes potential usability problems that should be solved in the future.

3.3.1 Test Questions

The test result should answer questions below,

- Are diagrams in the modeling tool understandable?
- Can a user model a tool chain with the modeling tool easily?
- Do modeling diagrams contain the information user want?
- Is there any issue for user to understand the modeling diagrams?
- Is there any issue for user to use the modeling tool?
3.3.2 User Characteristics

One pilot test is performed to make sure the test is able to be performed as expectation. 6 participants from the target user group go through the test. All participants should have the background knowledge of OSLC and linked data. Table below shows participants characteristics.

<table>
<thead>
<tr>
<th>Participant type</th>
<th>Desired number of participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pilot</td>
<td>1</td>
</tr>
<tr>
<td>Regular</td>
<td>6</td>
</tr>
<tr>
<td>Total</td>
<td>7</td>
</tr>
</tbody>
</table>

Participants were recruited among software engineers of Scania and OSLC tool chain researcher of KTH in Sweden. Their ages range from 28 to 36. There were 2 males and 1 female. They all do the OSLC tool chain related work. So they all possess the background knowledge and were familiar with the linked data approach.

3.3.3 Test moderator role

Test moderator is responsible for coordinating the whole test. At first, test moderator should introduce the test to participant. Moderator needs to prepare test environment and equipment. After testing, moderator will have a post interview with the participant. Finally, moderator give the participant an SUS form to fill in. A moderator script is written to make sure that the moderator is able to go through the same things for all tests. See moderator script on Appendix A.

3.3.4 Test Design

The test takes around one hour. At first, moderator briefly introduces the modeling tool and the knowledge of OSLC and linked data with a dummy use case. Then the moderator will assist participant to go through a tutorial about how to use the tool. During the tutorial, participant will learn basic functions and interactions by modeling a simple tool chain for the dummy case. After tutorial is finished, participant will model another tool chain for a more complicated use case. The description of use case is written in text. The participant needs to read the description and model the tool chain. During the test, moderator won’t answer any question except issues caused by technical bugs. The participant have to finish the modeling task by himself/herself. After the test, a post interview is conducted with the participant. Questions in the interview are mainly related with the usability of modeling tool and how the user understands the modeling diagrams. At the end, the participant need to fill in a System Usability Scale (SUS)
form. SUS is a reliable, low-cost usability scale, created by John Brooke, that can be used for global assessments of systems usability [21]. See SUS form in Appendix C.

3.3.5 Test Task

The only task in this test is modeling the tool chain for the given use case. The participant needs to find his/her own way to finish the task. There isn’t the best solution. The purpose is to make usability issues and problems exposed during the task. See detailed task description in Appendix B.

3.3.6 Test environment and equipments

The test should be performed in a quite room to make sure that participant won’t be disturbed. The modeling tool is run on one laptop computer with 15 inches screen, on which Microsoft windows operating system is installed. Mouse and keyboard are used in the test.

3.3.7 Data to be collected and evaluation measures

The task complete rate is measured. Time cost won’t be measured, because different approaches will be used by participants in this open task. Then the comparisons between different participants’ time cost is meaningless.

Qualitative data is collected during the test and from the post interview. Questions below will be asked in the post interview,

- Do you understand the model diagrams and the treeview?
- Is there anything you find it’s hard to understand? Why?
- Can you easily find the information you want?
- Have you met any problem to locate the information?
- What do you like most in the model diagrams or in the modeling tool?
- What do you want to change or improve for the model diagrams or the modeling tool?
Chapter 4

Results

This project consists of 3 phases. They are pre-implementation, implementation and post implementation. Results of each phase are described and explained separately in this chapter.

4.1 Pre-implementation Phase

In this phase, 2 tasks were finished. One is user research. The goal of this task is to find users’ problems in their work and understand how they work. The other is making design decisions and prototype. The goal of this task is to generate solutions to users’ problems, make and improve prototypes.

4.1.1 Affinity Diagram

3 contextual interviews were made, and each one took around 1.5 hours. The interviewer took notes and recorded audio for the interview. After the interviews were finished, all nodes and audio recordings were reviewed and interpreted as affinity notes.

74 affinity notes were created. All notes were printed as small paper cards and used for building the affinity diagram. After 2 rounds’ notes grouping, a 3 levels affinity diagram was built. 7 items were extracted. They represented users’ problems and requirements. See the results in next 3 pages,
• Data definition, data relationships, data sources, etc. These information should be included in the tool chain modelling and visualized in the modelling diagram.
  o I want to see data relationships, data definition, data source, etc from the modelling diagram.
    ▪ Architect cares the data relationships most in a tool chain modelling.
    ▪ Data definition and data relationships are important in tool chain modelling diagram.
    ▪ People want to go for details information. For example, the data relationships, dependency, etc.
    ▪ Both people using the tool chain and tool chain developer are concerned with data, how data are linked, what’s dependencies, etc.
    ▪ Data modelling diagram is used for verifying with checking data relationships.
    ▪ People using the tool s are more interested in data linking and transformation in a tool chain modelling diagram, because they want to understand how this tool chain integration can support their work.
    ▪ Relationships among resources need to be provided.
  o I want detailed information of data, adapter and relationships to be visualized in the diagram.
    ▪ Data is shown with in table with properties.
    ▪ Data are linked by arrows, text on the arrows are used for describing relationships.
    ▪ Data from different data sources are shown in different colors.
    ▪ Provides practical graphic components for drawing, for example, database component.
  o Data can also be "linked" according to business logic, this relationship need to be considered, managed and visualized in the modelling, too.
    ▪ Duplicated data need to be linked.
    ▪ Two resource property might describe the same thing, but they are in different names, and they can’t be replaced by the other one.
    ▪ Actions like log, synchronize, notification, will be triggered if condition is true.
    ▪ User has option to choose whether linked data are synced or not.
    ▪ When resources are linked for synchronization, conditions might be used as for creating, updating, deleting.

• Code generator is useful for generating the code, but it’s not easy to learn and use.
  o Code generator is good at generating code, but it is not user-friendly for designing adapter model.
    ▪ After learning and understanding, code generator is easy to use.
    ▪ Linking data in code generator is not user friendly.
    ▪ Data connection between tool and adaptor can’t be automatically generated.
    ▪ Code generator is good at generating the framework code, but developer still need to go into the code.
  o Learning how to use code generator takes much time and is not easy.
    ▪ Model in code generator doesn’t make a clear view to show how this adaptor works.
    ▪ Developer from industry can’t use code generator, since it is too complicated
    ▪ It’s hard to find how adapter model maps to code script.
    ▪ Learning to use code generator is not easy and cost time.
    ▪ Documentation of code generator is not helpful.
    ▪ It takes time to understand and learn to learn code generator.
  o Developer builds tool adapter by defining adapter model via code generator.
    ▪ Adapter developer needs to know the resources provided by adapter before developing, both resources need to be provided and the resources need to be consumed, as well as where the consumed data sources.
    ▪ Developer builds adapter model to create adapter framework via code generator.
    ▪ Adapter model need to be provided for creating adaptors.

• Data related operations like linking data, data definition should happen in the tool adapter.
  o Tool adapter should be responsible for data management including data definition, relationships.
    ▪ DIM can be used to model data of adaptor after data interfaces have be implemented.
- An data integration model (DIM) is created for managing data interfaces and integration.
- Resources which are managed by this adaptor need to be provided to the code generator.
- Relationships are defined in the code generator.
- Resources management need to be done in adaptor.
- Another model is created for resources to make adaptor developer could define resources basing on it.
  - How data is linked is managed and defined by tool adapter.
    - Relationships is created between resources from different adaptors.
    - When linking data, user select resource properties from the list of available properties.
- **I want to model and draw tool chain with a standardized domain specific language. If I have to do it by myself, the diagram is hard for others to understand.**
  - People use drawing tools they like to design and draw the modelling diagram by themselves.
    - The modelling diagram is generated manually.
    - Architect need to draw and design the tool chain components by themselves.
    - An editors software is used to draw modelling diagram called yEd.
    - Power point is used for drawing architecture view in presentation.
  - In the diagram drawn by architects themselves, it's difficult for others to understand how data is linked.
    - Relationship information of different adapter models is missing when shown in the tree view.
    - User who see the diagram should have the knowledge of OSLC.
    - The meaning of arrows in current diagram is hard for people to understand.
    - It takes some time to explain how data is transferred, where data is from and the concept of linked data.
  - I want a standardized domain specific language for the tool chain modelling.
    - A formal or standardized tool chain model is necessary when showing to other people who are not familiar with the tool chain.
    - An domain specific language is necessary.
    - A domain specific language and components is required to model the tool chain.
    - UML is not used for tool chain modelling because there is no special components to represent tool chain components like tools, relationships.
    - There is no standardize component for showing the component in OSLC or linked data diagram, the appearance of components is not consistent, people design them by themselves.
- **People prefer the simple modelling diagram with necessary information and avoid complication & data overwhelming. In this situation, they have to use other modelling diagrams to make up the missing info.**
  - I don't want a complicated diagram which is full of information immediately.
    - The diagram need to be kept simple, so much detailed information are not shown, and no other artifacts are used to record those information.
    - The basic view shows only rough connections are necessary, for example, only show adapters and connections between adapters. People don't want too much details immediately.
  - I use the modelling diagram with other kinds of diagrams, since some information like detailed information, architecture information is not included in the modelling diagram.
    - EA is also used to model data structure, since rules, constrains, mapping condition can be applied to the model. These can't be done in yEd, since it's not a modelling tool.
    - Data model diagrams are also used together with the other information architecture model diagram, since there are related.
    - Sequence diagram is helpful to show how adapters/tools talk with each other, but it has limitation. It can't tell data transfer in general case, only in a specific case.
  - I usually only focus on part of information in the diagram, and I don't want to be disturbed by other irrelevant information.
- Adaptor developer might be interested in showing adaptor model by navigating the architecture modelling diagram.
- User wants to find where data is from only when they want to.
- If some architecture information like tools, adapter, service providers are displayed in the modelling diagram, this information might bring confusion to user to understand data relationships.
- When designing how data are linked, architect prefer to focus on data/resources without disturb from adapters or tools.

- **People do discussion and analysis based on the modelling diagram together.**
  - I will do some analysis on the diagram to know how things affect each other.
    - People want to know how components affects each other when one components changes.
    - In addition to modelling data integration, a sequence diagram also allow people to make further analysis basing on it.
  - People often sit together to discuss the design with the modelling diagram.
    - Architect and people who give the requirements sit together to do validation.
    - Whether the design meets the requirements is verified until implementation is done or during the implementation.

- **Tool chain is designed to support scenarios and the design keeps being updated until it is finished.**
  - Tool chain design keeps being updated until it is finished.
    - The tool-chain design is always changed in small steps continuously.
    - The modelling diagram is always updating and growing to meet the new requirements.
    - Discuss and change the data modelling diagram, and implement change, and then do validation.
    - The scenario will change, so the design will also be updated and changed.
  - Tool chain is designed to support some scenarios. Data need to be provided in this scenario is considered at first.
    - Which data is required is considered at first.
    - The design of how data are linked and transferred is discussed basing on the scenario.
    - Architecture design starts from a scenario case, the scenario is described in text.
    - User don't link all data in a tool, only part of data is concerned.
The affinity diagram shows both users’ problems and requirements. From the perspective of users’ problems, it is hard for user to model a tool chain without usable graphical user interfaces. In addition, there is no a unified representation specification for the tool-chain model. A user has to create their own diagrams. Most of these diagrams can’t be understood easily. From the perspective of users’ requirements, users want the easily understandable model diagrams and a usable modeling tool. In the diagrams, resources, relations and dataflows are the most important information. And the diagrams should avoid information overwhelming and keep simple.

4.1.2 Design visioning and Prototypes

Based on the users’ problems and requirements exposed by the affinity diagram, possible solutions were created and visioned in this step. The first prototype is mainly about the modeling diagrams. See picture below,
In this prototype, resources, resource properties and semantic relations were all represented in the diagram. Tool adapters and adapter architecture were represented, too. This prototype referred entity-relationship diagram representation. After 2 prototype review interviews, information overwhelming issue was complained by users. Too much information was displayed together. It brought confusions and made it hard to locate important information quickly. In addition, the table view for resources and properties failed to show data flows.

The second prototype including both the modeling diagrams and the tool UI layout was created according to users’ feedback. See pictures below,
In this prototype, the entity-relation diagram representation was still referred. Resource properties were shown in a separated view. The tool adapter architecture was shown in a pop-up window when clicked. Besides, a tree view was added to enable user to quickly locate information. And a toolbox view was added, which contained the tools used for the tool-chain modeling.

After 2 more prototype review interviews, 2 main usability problems were found. One was about semantic relation representation. Linked data is based on RDF, the RDF model representation should be referred rather than the ER diagram one. The other was user’s workflow of the tool-chain modeling wasn’t supported. The modeling workflow
contains 2 phases. In the first phase, the semantic context of the linked data, including resources, properties, domains, is modeled. In the second phase, tool adapters and relations between resources and adapters are modeled based on the semantic context model of the last phase.

Since the tool chain data model is the meta model of RDF model, the RDF model representation can’t be applied directly. A new tool chain data model is created based on the RDF model. The picture below shows the difference between the RDF model and the tool chain data model.

**Figure 4.4: Tool-chain Model Based on RDF Model**

As same as the RDF model representation, the tool chain data model uses ellipse representing resource, rectangle representing literal data and arrow line representing property. But the ellipse represents the meta data of resources rather than one resource. It is the same as literal data representation. Instead of a solid line rectangle, a dashed line rectangle is used, because it represents the meta data of literal data. The literal value is replaced by the value type in the middle of rectangle. To match users’ workflow of the tool chain modeling, 3 diagrams are used to represent the tool chain from 3 different perspectives. One is the semantic context model diagram. Another one is the diagram showing how these well-defined resources are managed by tool adapters. The last one is the tool adapter architecture model diagram.
The details of diagram elements like size, color and shapes vary a lot when different visualization techniques are used, they are not included in this prototype. But the prototype shows the gist of the modeling diagram design.

4.2 Implementation Phase

4.2.1 EMF structured data model

Sirius is a UI framework based on EMF structured data model. The tool chain model isn’t able to be visualized until the structured data model of the tool chain is created.

In EMF, a meta model is developed to generate the structured data model. Picture below shows the meta model for the tool chain structured data model.
Figure 4.5: Meta model of EMF model for tool chain data
In the model, ToolChain is the root element of the structured data model. It contains domains and tool adapters. ServiceProviderCatalog, ServiceProvider and Service are elements of the tool adapter architecture. Resource and ResourceProperty are contained by the Domain element. The relation called properties between resource and resourceproperty represents connections between resources and properties. The relation named manageResource represents the management relation between tool adapters and resources. ResourceProperty’s attributes, such as occurs, valueType and the relation called range, are defined according to the OSLC specifications. Picture below shows an example of the structured data model for tool chain built on this meta model.

![Structured Data Model for Tool-chain](image)

**Figure 4.6: Structured Data Model for Tool-chain**

### 4.2.2 User Interface Implementation

**Representations For Model Elements**

Each model element of the structured data model should be visualized as the graphical object of the modeling diagram. The graphical representations for the elements are illustrated in this part.

Picture below shows the example of domain representation. It is a container with name and icon on the top. There is a sub-container inside. It contains the properties belongs to this domain. All resources belong to this domain will be placed in the container.
A resource is represented as a circle. Resource name and icon are placed in the center. There are 2 types of property. One is the property connecting the resource. It is represented as an arrow line pointing to another resource. Property name is placed in the middle of the line. The other is the property for literal values. It is the same arrow line but pointing to a dashed line rectangle that represents the literal value. And value type name is shown inside. There is an attribute of the property called occurs. Its values includes exactlyOne, zeroOrOne, zeroOrMany and oneOrMany. And they are represented as the annotations including 1..1, 0..1, 0..* and 1..*. The annotations are placed on the end of arrow line. See the example below,
In the example below, resources are placed in the domain container, which represents these resources belong to the domain. Property1 and Property2 are contained by the upper domain, and they are in the sub-container. However, one resource can be assigned with properties from other domains, so the arrow line representing property crosses domains. See picture below,

![Figure 4.9: Resource and Resource Property Representation 2](image)

Tool adapter is represented as the container which is same as domain but with a different icon. Resource contained by a tool adapter container means that this resource is managed by the adapter.

![Figure 4.10: Tool Adaptor Representation](image)
Views

The modeling tool contains 3 main views. One is called domain view. It contains the diagram showing the semantic context of linked data. See picture below.

![Figure 4.11: Domain View](image1)

In the diagram, there are domain, resource and property. Both literal value property and resource connecting property are included.

The second view is the tool chain view. It contains the modeling diagram including tool adapters and their managed resources. See the example below.

![Figure 4.12: Tool Chain View](image2)
In the diagram, only resource connecting properties are displayed. All resources and properties are from the domain view and reused here. And only tool adapter are created. Sync is another data relation between resources. It represents that the 2 linked resources are synchronized. And this relation is created in this view. It is a green dashed line between 2 resources in the diagram.

The third view is the tree view. It displays elements from both the domain view and the tool chain view in a hierarchy structure. It helps user to quickly locate the information. See the example below,
Results

Layout

Picture below shows the layout of the modeling tool. The tree view is placed on the left side. The tool box containing tools for drawing diagrams is on the right side. The main view and tool chain view are tab views in the center. They could be switched by clicking. When the tab is switched, tools in the tool box are updated, too. The bottom part is used for showing semantic attributes of selected objects in the diagram.

![Modeling Tool Layout](image)

**Figure 4.14**: Modeling Tool Layout

4.3 Post-implementation Phase

After the prototype of the modeling tool had been implemented, a usability test was conducted on it. There were 6 participants in total. All participants had the knowledge of linked data and OSLC specifications. And all of them were completely new to the modeling tool.

Task completion rate and SUS ratings were measured and calculated as the quantitative data. Users’ comments and suggestions in the post interviews were record as the qualitative data.
4.3.1 Task Completion Result

In the usability test, there was only one open task. A same task description was given to all participants. Each participant was required to model the tool chain all by himself/herself. All participants were able to completed the task. The task completion rate was 100%.

According to observations, overall, all participants followed the similar workflow, which is modeling the semantic context of the linked data at first and then the tool chain. But in some steps, they used different approaches. For example, some participants connected properties with resources until all resources and properties had been created. The other participants did it as soon as properties were created. In addition, assigning properties to resources by dragging and dropping was not intuitive for most of participants at the beginning. But after they had tried it for several times, they used it more frequently.

4.3.2 SUS Questionnaire

At the end of the test, every participant filled in the SUS questionnaires. The average score is 79.2. See details in the table below,

<table>
<thead>
<tr>
<th>Participant</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
<th>Q6</th>
<th>Q7</th>
<th>Q8</th>
<th>Q9</th>
<th>Q10</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>85</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>75</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
<td>95</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>60</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td></td>
<td>85</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>75</td>
</tr>
</tbody>
</table>

According to a study of 500 cases of SUS usages, a SUS score above 68 would be considered above average and anything below 68 is below average[22]. Even though all tests were performed on the prototype of the modeling tool, 79.2, the SUS score, is higher than the average. In the scope of the study case on which the thesis project is based, the modeling tool has a good usability.

4.3.3 Users’ comments and suggestions

Some common things were mentioned by users in the post-interviews.

The modeling diagrams are easily understood
In the interviews, all participants said, more or less, that they were able to understand the diagrams easily. They knew nothing about the tool chain they modeled before the test. But, after they had finished the test seen the diagrams, all participants were able to tell what the tool chain it is. The representative comments are...

"I like the resources and properties, because they are very obvious."

"I haven’t found something hard to understand. It’s easy to understand."

"It is quite straight forward. It’s not like you have to go across several places to find what you want to do...”

**Scalability**

Some participants showed concerns about scaling problem. If a more complex tool chain were modeled by this tool, some scaling problems might appear. For example, the domain diagram can’t be completely shown if it contains many elements. A user has to scroll and navigate. In the tree view, a user might need to scroll for several times to find a property if there are more than 100 properties. In addition, some participants suggested that multiple properties should be allowed to be selected and assigned to one resource together. And extra structured data models are allowed to be loaded and added into the existing one. These suggestions were raised to solve potential efficiency problems when the tool chain model becomes more and more complicated. The representative comments are...

"If I model a tool chain having 10 tools and 50 resources, I can’t have a overview from this diagram. And it will become crowded."

"When all items are expanded, the list would be too long to find the property."

**Little Learning Effort**

Most of participants said that it was easy to learn this tool. The operations of this tool and layout looked intuitive and straight forward. They were confident to use the tool by themselves. The representative comments are...

"The layout looks quite intuitive. I can find what I want.”
"It’s easy to learn, nothing difficult..."

"No, I have no question. The operations are simple."
Chapter 5

Discussion

In general, the result of usability test was positive. All participants were able to complete the task. And their feedback after the test was positive, too. In the meanwhile, the usability test also exposed some usability problems. In this section, they will be discussed in details.

According to the observations in the test and users’ feedback in the post-interviews, all participants could finish the task smoothly. None of them had been blocked seriously and felt frustrated. One reason should be the valid results from the UCD process in the pre-implementation phase. It was able to expose users’ real problems in the tool chain modeling. What’s more, the prototype review interviews were very helpful to make right design decisions which got users’ problems solved. For example, the decision to separate the semantic context model and the tool chain architecture model as two diagrams. It matches users’ workflow. Compared with a combined diagram, it looks more intuitive to users, because it enables the tool to support users’ workflow rather than creating a new one. It also explains why most of participants said that the tool was easy to learn. The core design thinking of the UCD method is to create the system that can solve users’ problems and support them to do their work more quickly and easily. Accordingly, the participants’ knowledge got reused. It enabled them to use the modeling tool with little learning effort.

In the post interviews, several participants talked about potential usability problems of the tool if the tool chain were more and more complicated. For example, the modeling diagram in the domain view might become too crowded to display detailed information. The same issue might happen in the tree view, too. The reason for users’ concerns should be the problems or requirements were missed in the UCD phase. In the contextual interviews, the interviewer haven’t sensed and investigate them. In addition, in the scope of the study case on which the project is based. The study case, which is also
used in the usability test, only involved 3 tools. The problems were not able to be exposed. But this issue can be fixed by iterating the UCD cycle. The problems missed are able to be found and solved in the next round UCD process.

The gap between design and implementation should be considered, too. It is a common case that some part of design can’t be implemented due to technical limitation. It usually brings some usability problems. For example, when one item in the tree view is selected, the same item in other views is designed to be highlighted. But the interaction is not supported by Sirius. A user had to spend more time to find the item. For another example, the domain container can’t be resized automatically when a resource or property is placed out of it. And the item out of container space was not shown, either. A user had to resize the container by themselves to find the hidden item. In most cases, the gap can’t be eliminated but can be minimized. In general, a good communication between designer and developer is very helpful to make a good balance between design and implementation. Being Familiar with the techniques used in the implementation helps a designer to make the design implementable instead of throwing the part of design completely.
Chapter 6

Conclusion

In the project, the UCD approach was used in the process of making a usable modeling tool for the OSLC tool chain. Based on the results from the UCD approach, the modeling tool was designed and implemented. At last, the usability test was performed to verify the usability of the modeling tool. Considering it is a case study research, the conclusions are valid in the scope of the case on which this study is based. They are not verified in the bigger scope.

The results of the usability test were quite positive. 100% task completion rate and 79.2 SUS score showed that the system had a good usability. Users’ feedback from the post interviews also showed that they were satisfied with the system in the scope of the study case.

In general, it could be concluded that the modeling tool should provide easily understandable modeling diagram representations. It should enable users to model a tool chain graphically. And the tool should support user’s workflow of the tool-chain modeling. The semantic context of data should be modeled at first. And then the tool chain architecture model is created. In the diagram of the domain view, resources and properties should be included and displayed, as well as relations between them. In the diagram of the tool chain view, tool adapters, resources tools manage and resource connecting property should be included and displayed.

All participants were able to model the tool chain by using the modeling tool. It shows that the solution for implementing the tool is valid.

In the scope of the study case, the UCD approach was considered to be valid to make the usable modeling tool for the OSLC tool chain. During the tool-chain development, a variety of people would be involved in the design and development. In this situation,
the modeling tool will be used to support various people’s work. The high usability will be one of the most important goals for the tool. The UCD approach should be used.
Chapter 7

Future Work

According to the usability testing results, efficiency problems probably appear when the model becomes more and more complex. A new round UCD process need to be executed. In the new round, problems caused by the complex tool chain model could be addressed. The solutions to the problems could be visioned, created and implemented, too. Finally, a usability test will be performed to verify whether the problems have been solved completely.

In addition, it is possible to use the structured model generated by the modeling tool for the automatic code generation of the tool chain. It can save much effort for developers in the tool chain development. And it is also easier to maintain the tool chain when changes happen.

Extra functions can be added to improve efficiency. For example, adding shortcuts, search in the tree view, highlighting selected item, filter and so on. Improving efficiency will increase the usability as well. The more usable the modeling tool is, the better the model-based development of the tool chain is supported.
Appendix A

Moderator Script
This tool is aiming to help user graphically model data in the tool chain which uses linked-data approach.

Before introducing the tool, let me briefly introduce linked-data and OSLC.

In the real world, we often defined many different domains for information grouping. Each domain contains different resources and relations. For example, engine, chassis, truck are all resources belong to truck manufacturing domain. Truck is drove by engine is one of the relations in this domain.

Now we use more and more software systems and tools to manage the data from different domains. Linked data is about linking these data according to the semantic relations. And OSLC is the techniques used for integrating these systems and tools according to linked data concept. In OSLC specification, tools are integrated and linked as a tool chain. In this tool chain, all tools can share and manage data together.

(It is just prototype to demonstrate core idea like how the tool looks like and how it works. So several known technical bugs are not solved completely, since they don’t influence the usability test. And I’ll show you what they are later. )

So, there are 3 views in total in this modelling tool.

One is the Domain view, this view is used for modelling the data from perspective of domain. The second view is tool chain view, it is used for modelling the OSLC tool chain architecture from the perspective of tool chain. The third view is a tree view, it shows both domain and tool chain model as a hierarchy tree. It gives an overview of the whole tool chain data model and helps to quickly locate the information.

To understand how to use this tool, I’ll let you go through a tiny tutorial. In this tutorial, you’ll create a tool chain data model for a simple use case.

The case is now a university has 2 software systems. One system is responsible for managing student information. The other system is responsible for managing course information. The student has name and studentID, course has name and course description. The student takes courses.

In domain view

- Create domains and introduce domain container and property sub container.
- Introduce “Property” tab
- Edit name in 2 ways
- Add resource
- Introduce “Property” tab and rename.
- Introduce Tree View
- Add property StudentID to Learner domain, then edit name and value type. (Mention the container bug)
- Add property Description to Teaching Domain from tree view and edit attributes from property tab. (Mention the bug)
- Add new domain Dublin.
- Introduce zoom in&out
• Add “name” to Dublin
• Add “take” to learner and introduce Occurs and Range. (Explain why and when a resource won’t be not listed in the range dropdown list.)
• Assign properties to resources in 2 ways. Explain the arrow.
• Explain RELEASE property from resources and difference between DELELE
• Mention autosize and related bug
• Literal property filter
• Click save and go to tool chain view.
• Check tree view in tool adaptors category.
• Explain available resources.
• Create 2 adaptors and rename as Student System Adaptor and Course System Adaptor.
• Assign resource to tool adaptor in 2 ways. (Name not shown bug)
• Explain literal properties are not shown in this view.
• Release resource, change resource.
• Sync resource.

Any question?

In the test, you’ll model another similar use case but a bit more complicated. You might not have the background knowledge of this use case, so you might meet some terms you are not familiar with. But it won’t block the modelling task, you can just see them as some particular names.

During the test, you are encouraged to think aloud, which is saying whatever you are thinking, doing and feeling. When you meet some problems, please find a way to solve it. I’ll only help you solve issues caused by known technical bugs during the test. But if there is any place you don’t understand in the task description. Please feel free to ask me.

It’s worth to mention that this test is to test the usability of this prototype tool. It doesn’t test anything about you. So please take it at ease and have fun!
Appendix B

Test Task
Background

There is a tool chain consisting 3 tools called Requirement Management System (RMS), SESAMM Tool and Message Sequence Chart Drawer.

In this tool chain, there are 4 resources and 9 properties in total from 4 difference domains.

Please model this tool chain according to the following description.

From the perspective of domain...

There are 4 domains, Message Sequence Chart, Requirement Management, Electrical System Architecture and Dublin.

- In the domain of Dublin, there are 2 properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Name</th>
<th>Occurs</th>
<th>Value Type</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>name</td>
<td>exactlyOne</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>identifier</td>
<td>exactlyOne</td>
<td>String</td>
<td></td>
</tr>
</tbody>
</table>

- In the domain of Requirement Management, there are 1 resource and 1 property.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Name</th>
<th>Occurs</th>
<th>Value Type</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource</td>
<td>Requirement</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>name</th>
<th>XMLLiteral</th>
<th></th>
</tr>
</thead>
</table>

Requirement, the resource listed in the table above, contains 2 properties. See details below,

Resource: Requirement
Properties: name – from domain of Dublin
description – from domain of Requirement Management

- In the domain of Electrical System Architecture, there are 2 resource and 3 properties.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Name</th>
<th>Occurs</th>
<th>Value Type</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource</td>
<td>Component</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Signal</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>family</th>
<th>exactlyOne</th>
<th>String</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>componentName</td>
<td>exactlyOne</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>sendSignal</td>
<td>zeroOrMany</td>
<td>Resource</td>
<td>Signal</td>
</tr>
</tbody>
</table>

Component, one of the resources listed in the table above, contains 4 properties.
Signal, one of the resources listed in the table above, contains 1 properties.
See details below,
**Resource**: Component

**Properties**: name – from domain of Dublin
  - sendSignal – from domain of Electrical System Architecture
  - family – from domain of Electrical System Architecture
  - componentCode – from domain of Electrical System Architecture

**Resource**: Signal

**Properties**: name – from domain of Dublin

- In the domain of **Message Sequence Chart**, there are 1 resource and 3 properties.

<table>
<thead>
<tr>
<th>Name</th>
<th>Occurs</th>
<th>Value Type</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource</td>
<td>Sequence Chart</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Property</td>
<td>hasComponent</td>
<td>zeroOrMany</td>
<td>Resource</td>
</tr>
<tr>
<td></td>
<td>hasSignal</td>
<td>zeroOrMany</td>
<td>Resource</td>
</tr>
<tr>
<td></td>
<td>hasRequirement</td>
<td>oneOrMany</td>
<td>Requirement</td>
</tr>
</tbody>
</table>

**Sequence Chart**, the resource listed in the table above, contains 4 properties.
See details below,

**Resource**: Sequence Chart

**Properties**: name – from domain of Dublin
  - hasComponent – from domain of Message Sequence Chart
  - hasSignal – from domain of Message Sequence Chart
  - hasRequirement – from domain of Message Sequence Chart

**From the perspective of tool chain architecture...**

There are 3 tool adaptors in the tool chain. They are called **MSC adaptor, SESAMM adaptor and RMS adaptor**. They manage 4 resources as described below,

- **MSC adaptor** manages 1 resource called **Sequence Chart**.
- **SESAMM adaptor** manages 2 resources called **Component** and **Signal**.
- **RMS adaptor** manages 1 resource called **Requirement**.

In the tool chain, resource called **Requirement** is synced with resource called **Signal**.
Appendix C

System Usability Scale Form
### System Usability Scale


<table>
<thead>
<tr>
<th>Statement</th>
<th>Strongly disagree</th>
<th>Strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. I think that I would like to use this system frequently</td>
<td>1 2 3 4 5</td>
<td></td>
</tr>
<tr>
<td>2. I found the system unnecessarily complex</td>
<td>1 2 3 4 5</td>
<td></td>
</tr>
<tr>
<td>3. I thought the system was easy to use</td>
<td>1 2 3 4 5</td>
<td></td>
</tr>
<tr>
<td>4. I think that I would need the support of a technical person to be able to use this system</td>
<td>1 2 3 4 5</td>
<td></td>
</tr>
<tr>
<td>5. I found the various functions in this system were well integrated</td>
<td>1 2 3 4 5</td>
<td></td>
</tr>
<tr>
<td>6. I thought there was too much inconsistency in this system</td>
<td>1 2 3 4 5</td>
<td></td>
</tr>
<tr>
<td>7. I would imagine that most people would learn to use this system very quickly</td>
<td>1 2 3 4 5</td>
<td></td>
</tr>
<tr>
<td>8. I found the system very cumbersome to use</td>
<td>1 2 3 4 5</td>
<td></td>
</tr>
<tr>
<td>9. I felt very confident using the system</td>
<td>1 2 3 4 5</td>
<td></td>
</tr>
<tr>
<td>10. I needed to learn a lot of things before I could get going with this system</td>
<td>1 2 3 4 5</td>
<td></td>
</tr>
</tbody>
</table>
Bibliography


[12] Lodlive. . URL http://www.w3.org/.


