CHALMERS

Software Process Improvement using Language Workbench Technology

Master of Science Thesis in the Programme Software Engineering

Xi Zhu
Congchi Phung

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, June 2013

Software Process Improvement using Language Workbench Technology

Xi Zhu
Congchi Phung

Examiner: Michel Chaudron
Supervisor: Lars Pareto

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden June 2013

Abstract. Model driven engineering (MDE) is a proven approach to improve software development processes by automation. However, traditional development of MDE tooling requires a high upfront cost. Recent developments in language workbench technologies promise to significantly reduce these investment costs. By providing domain experts with targeted projections, the speed and quality of delivering customer value is improved. This paper provides results from an industrial case study in the telecommunications domain and compares the value of using a language workbench to traditional MDE technologies. Our results, using the Intentional Domain Workbench, indicate that applying a language workbench promises significant improvements in several aspects of MDE based software development. Most notably in this paper: (1) improved speed in development of domain specific tooling and (2) improved speed in software development process re-engineering.

Keywords: language workbench, domain-specific language, model-driven engineering, case study

Acronyms, Abbreviations and Terms

CPI	Customer Product Information
EMF	Eclipse Modeling Framework
DSL	Domain-specific language
IDE	Integrated Development Environment
IDW	Intentional Domain Workbench
KWSID	Knowledge Workbench for Software Interface Definition
LOP	Language Oriented Programming
MOM	Managed Object Model
MDE	Model-driven engineering
NMS	Network Management Systems
UIO	User interface objects
UPT	Usability Problem Taxonomy
RBS	Radio Base Station

Table of Contents

Software Process Improvement Using Language Workbench Technology

Xi Zhu[1], Congchi Phung[1]

[1]Chalmers University of Technology, Gothenburg, Sweden

Abstract. Model driven engineering (MDE) is a proven approach to improve software development processes by automation. However, traditional development of MDE tooling requires a high upfront cost. Recent developments in language workbench technologies promise to significantly reduce these investment costs. By providing domain experts with targeted projections, the speed and quality of delivering customer value is improved. This paper provides results from an industrial case study in the telecommunications domain and compares the value of using a language workbench to traditional MDE technologies. Our results, using the Intentional Domain Workbench, indicate that applying a language workbench promises significant improvements in several aspects of MDE based software development. Most notably in this paper: (1) improved speed in development of domain specific tooling and (2) improved speed in software development process re-engineering.

Keywords: language workbench, domain-specific language, model-driven engineering, case study

1 Introduction

Model-driven engineering (MDE) is a software engineering paradigm that addresses the problem of increasing complexity of software by abstraction and transformation. With MDE, domain experts use modeling languages which express domain notations in order to model abstractions for specific problems. As MDE received wider recognition in the field of software engineering, a plethora of modeling tools were introduced.

First generation modeling tools were characterized by MDE through domain specific model driven development tools, and realized by an external tool vendor using conventional programming languages, e.g., Simulink [1], Rational Rose Realtime [2] and Rhapsody [3]. In first generation modeling tools, meta-models, editors, and transformations were typically concealed, data formats typically proprietary, and platform adaptations typically provided by the vendor.

Second generation modeling tools made meta-models and transformations first class artifacts. Modeling tools of this generation followed standards to an increasing degree, and users of these tools could define their own model transformations. The

second generation modeling tools were characterized by the Eclipse Modeling Framework [4].

Third generation modeling tools addressed the high development cost of implementing DSLs and were characterized by complete IDE solutions in which modeling languages can be realized "in a day or two". Examples of this generation are Microsoft Visual Studio DSL Toolkit [5] and Metaedit [6].

Recently, a new type of tool has emerged which is an evolution of third generation modeling tools. Language workbenches with projectional editor provide editable and synchronized views of models, specifically tailored for users in specific domains [7][8]. Language workbenches promise to significantly reduce the development effort of constructing DSL applications and improving the speed in software development through tailored projections for domain experts. To our knowledge, there are no published studies that, in an industrial context, investigate the values that language workbench technology can provide to MDE based software development processes such as cost, usability, end-to-end speed, error prevention and so on, compared to existing MDE solutions.

This thesis presents an industrial case study which investigates how language workbench technology can improve MDE based software development processes, in telecommunication systems development.

The research problem and related research questions are the following:

RP: How can language workbenches improve MDE based software development processes?

- **RQ1**: *What process qualities (e.g. speed, cost, usability) may language workbenches improve in the context of interface modeling within large scale embedded system development?*
- **RQ2**: *How do X compare between traditional MDE solutions and language workbench solutions., with X ranging over development cost, end-to-end speed for change requests, usability, and other factors found in RQ1, in the context of interface modeling within large scale embedded system development?*

The case study applied a language workbench (the Intentional Domain Workbench from Intentional Software) to re-engineer an existing development process for interface definitions. To evaluate the language workbench approach, the study also compared the development effort of creating a DSL application for interface definition development using a language workbench, with that of a development process based on the Eclipse Modeling Framework. In addition, a usability study was conducted to assess the values gained from using the language workbench approach of users in the development process.

The paper is structured as follows: Chapter 2 lays the theoretical foundation of the concepts used in this thesis including software interface development and language workbench technology in particular the Intentional Domain Workbench; Chapter 3 presents the research methodology including the design of the case study at Ericsson AB; Chapter 4, outlines the results of the studies; finally, chapter 5 and 6 discuss the results, recommendations and conclusion drawn from the study.

2 Background Theory

This chapter covers the relevant theory of the concepts used in subsequent chapters of this thesis report.

2.1 Software Interfaces In Telecom Management Network

In a telecom management network, network management systems (NMS) are used for monitoring and controlling network resources, for example radio base stations [9]. In current practices, NMS are realized using an object oriented approach where an object information model provides abstract representations for the entities in a network [10]. These abstract representations, managed objects, encapsulate the underlying network resources and expose software interfaces which NMS require in order to handle operations requested by an operator. Figure 1 illustrates an NMS and several radio base stations as managed objects in a network. An operator terminal is used to control and monitor the network resources through an NMS.

The different interface development environments address two different types of software interfaces: external interfaces which specify the interaction between radio base stations and the NMS, and Internal interfaces which specify the interaction between the software components within the radio base station. When new features are requested or changes are made to the underlying network resource, the external and/or internal software interfaces might need to be updated to reflect these changes.

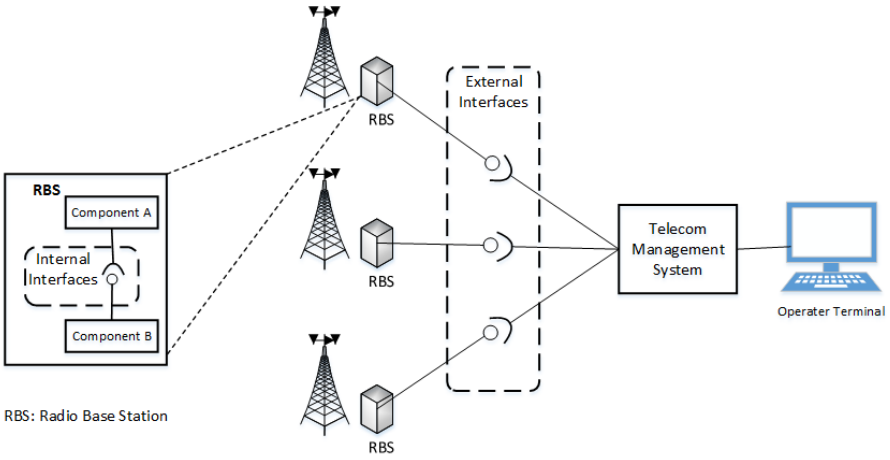

Fig. 1. Software interfaces in a telecom management network

2.2 Language Workbenches

Language workbenches denote a category of tools that according to Fowler [11] "implement language oriented programming (LOP)". Language oriented programming is based on the concept of allowing developers to easily define reusable and interoperable domain-specific languages (DSLs) [12]. Fowler, who coined the term language

workbench, defined the required characteristics that language workbenches shall exhibit [11]: “

- *Users can freely define new languages which are fully integrated with each other.*
- *The primary source of information is a persistent abstract representation.*
- *Language designers define a DSL in three main parts: schema, editor(s), and generator(s).*
- *Language users manipulate a DSL through a projectional editor.*
- *A language workbench can persist incomplete or contradictory information in its abstract representation.* “

Voelter [13] further extended these characteristics with the ability to develop complete programs and the addition of tool support such as code completion, syntax highlighting and debugger.

In essence a language workbench is a platform where interoperable DSLs can be specified and used to create domain specific encodings which are then generated to artifacts. An overview of language workbench technology is shown in Figure 2. As MDE tools [14] are based on the similar idea of using DSLs as modeling language and transformation to generate artifacts, language workbenches can be applied in the context of model-driven software development. The key advantages of using language workbenches in such a context are the benefits provided by a projectional editor. A projectional editor enables the creation of editable views of a user defined model. These views can be tailored for specific domains allowing domain users to encode their solution in notations they find most suitable.

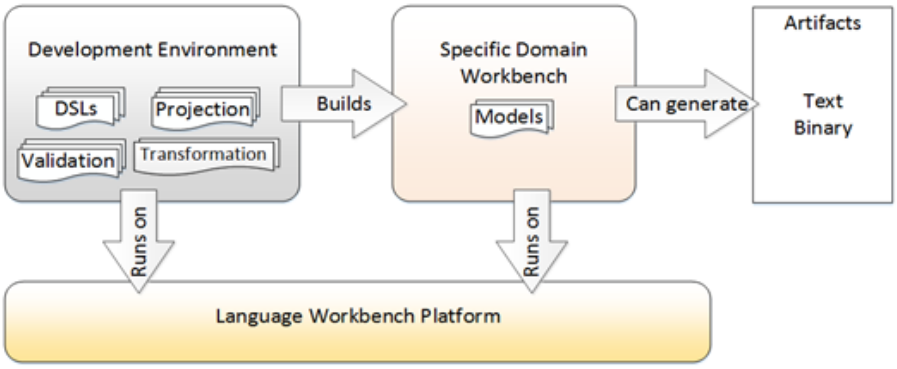

Fig. 2. Overview of language workbench technology

Intentional Domain Workbench.

The Intentional Domain Workbench (IDW) is a commercial language workbench developed by Intentional Software. The Intentional Domain workbench is targeted towards business users by providing projectional editors which allow manipulation of models described with DSLs in textual, tabular and graphical notation [15]. The core elements of a DSLs application, Knowledge Workbench, developed using IDW con-

sists of: domain schemas, corresponding to the abstract syntax (meta-model) of DSLs; domain code, models described using DSLs; projections, the editable views provided by projectional editors; validation rules, which express the constraints of DSLs; and generators, which given domain code (model) produces code for specific target platforms.

2.3 Usability Problem Taxonomy

The Usability Problem Taxonomy (UPT) is a framework for classifying and analyzing usability problems of graphical user interface applications [16]. UPT facilitates the process of analyzing usability problems by categorizing in problem clusters. Problem clusters can then be used to identify global problems and outliers, enable problem analysis in different abstraction levels to identify contradictions and tradeoffs and finally, provide problem prioritization based on UPT distributions and severity. The framework includes a problem categorization which consists of two components: an Artifact component and a Task component. The artifact component is related to issues when "users interacts with individual user interface objects" while the task component is related to issues of "how a user task is structured on the system (task mapping) and the system's ability to help the user follow the task structure and return to the task when a deviant path has been taken". Each component is structured as a hierarchy containing primary categories and subcategories. Figure 3 [16] shows an overview of the categories of the UPT. Descriptions of the primary categories are listed below. For more details regarding the categories, see [16] [17].

Artifact Component.

Visualness

The visualness category is a related to the user interface; how users view user interface objects (UIO). This includes the appearance and layout of UIOs, how information and feedback of user queries are presented.

Language

The language category is related to problems which users have with the understanding of the terminology that is presented in the user interface.

Manipulation

The manipulation category is related to problems when users are manipulating UIOs. This category includes problems with direct manipulation, viewing and understanding of visual cues.

Task Component.

Task-mapping

The task-mapping category is related to "*how well user tasks are mapped to the system*" [16]. This category includes problems concerning the functionality for supporting a user task, how well the user can navigate and interact with the system.

Task-facilitation

The task-facilitation category is related to how well the system supports the user when the user is performing task on the system. This category includes among others: error recovery, task/function automation, keeping the user on track and error prevention.

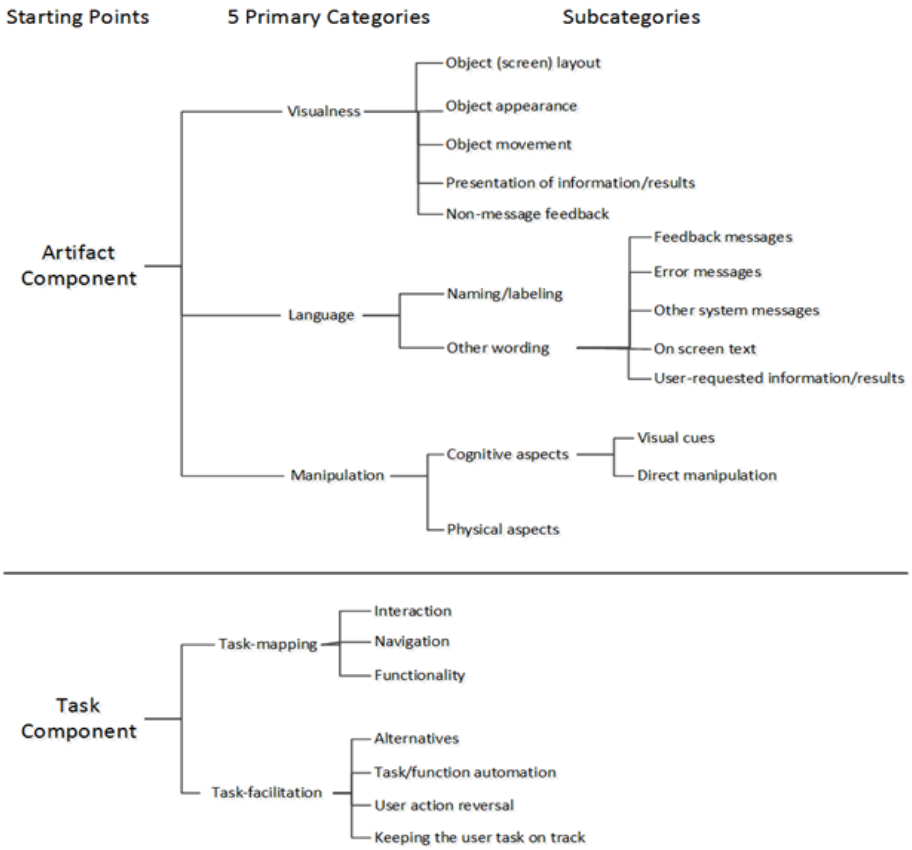

Fig. 3. The Usability Problem Taxonomy [16]

2.4 Semantic Gap

In language processing theory the semantic gap refers to [18] "*the difference in meaning between constructs formed within different representation systems*". In a software engineering context, semantic gaps occur in the mapping of high level domain knowledge to machine processable construct expressed in some proper programming language. Figure 4 [18] shows an example of the manifestation of semantic gaps between the constructs to express a mathematical formula in a machine processable construct. Problems caused by semantic gaps consist of increased development effort

[19] and reduced software quality [19] due to communication issues between domain experts and software developers [18].

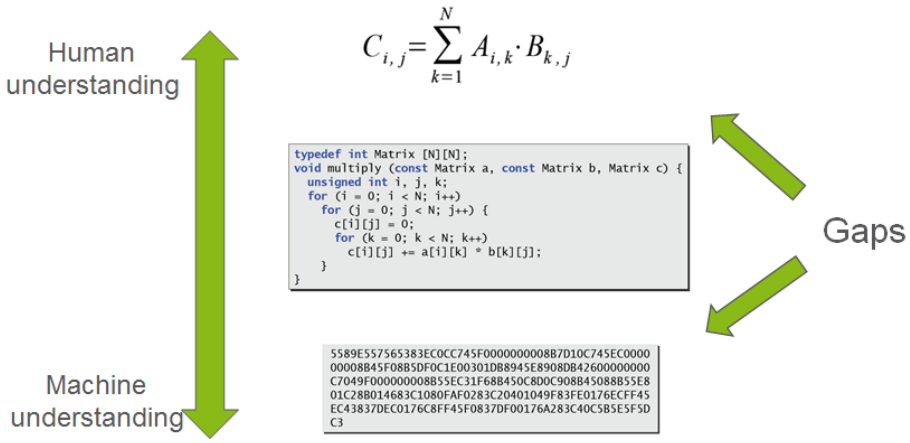

Fig. 4. Semantic gaps manifest in the difference of the constructs expressed in different representation systems [18].

3 Research Methods

This chapter presents the research methods used in this study. An overview of the research methods is given in Figure 5. The research strategy in this study is case study research, with software processes for model based interface specifications being the unit of analysis. Research methods employed were semi-structured interviews for data collection on needs; qualitative analysis for identification of desirable qualities of interface modeling processes and tools; proof of concept implementation of and IDW-based solution; usability study to see to which degree language workbenches can fulfill the desirable needs and possess desired qualities; qualitative data collection and qualitative analysis to estimate and compare the efforts of using traditional MDE versus using language workbenches —- efforts for implementing the tools as well as using them.

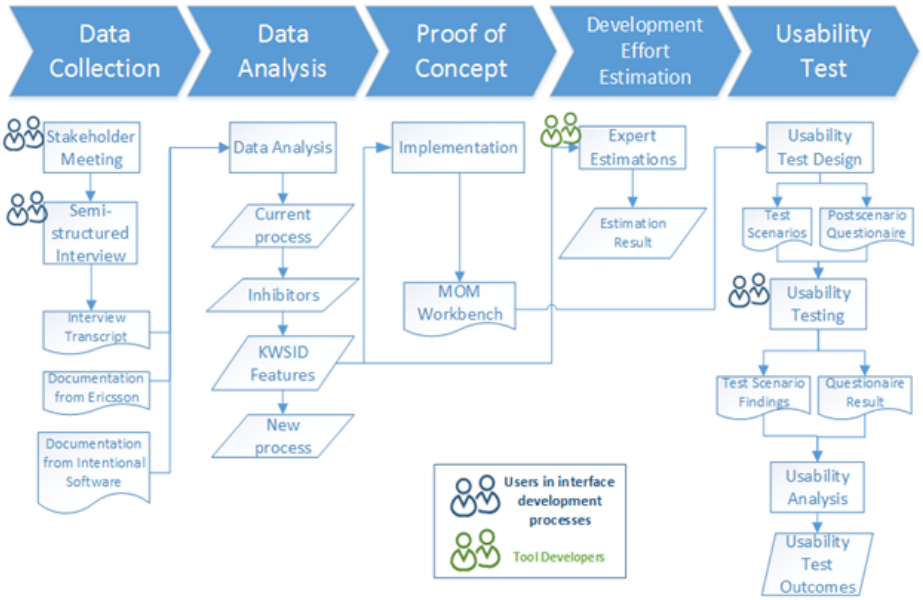

Fig. 5. Overview of the Research Methods

3.1 Research Site and Informants

The case study was conducted at Ericsson AB, a worldwide corporation which provides telecommunication solutions for network operators. Ericsson AB is divided into business units targeting different areas within the telecommunication domain. Our case was a particular MDE based software development process for interface definitions used within the business unit Networks. The process is widely used within the unit, and utilizes a flora of second generation MDE tools and technologies. The study focused mainly on two specific software interface domains and its associated tooling. Both use an MDE approach to automate the transformation of the interfaces to deployable artifacts. Although users of the current environments find them useful, there are several opportunities to increase speed and quality to strengthen the business units' competitive advantage on the market.

The roles of informants in the case study include tool developers and users of both the EMF-based and IDW-based environments: a tool developer and a domain expert from one domain; a tool developer from the other. One domain expert involved with both domains. The informants are well-versed in the field of modeling while only developers have practical experience of using the specific tools of the studied MDE based development processes. None of the informant had previous experience with IDW.

3.2 Data Collection

Data was primarily collected from archival data and through qualitative enquiry from stakeholder needs. Semi-structured stakeholder meetings were conducted to understand the domain and the context in which MDE is applied in their current development process. Stakeholder meetings were held separately for each interface domain with at least one person. The meetings were conducted during the period January-May 2013 and the duration of the meetings varied from 40 minutes up to 1 hour.

Archival data was included as data source in the form of guideline documents, software design documents and work artifacts produced by stakeholders involved in the studied context. Furthermore, semi-structured interviews with the informants of different roles were conducted in order to gain a better understanding of the specific aspects mentioned in the stakeholder meetings. The duration of an interview lasted for approximately 1 hour and was held during the same time period as the stakeholder meetings. Interviews were audio recorded and field notes were taken.

3.3 Data Analysis

The analysis started with transcription of the recordings and field notes taken during interviews and stakeholder meetings, as well as documentation from both Ericsson and Intentional Software. From the transcripts, we identified keywords and phrases, mentioned by the informants that caused inhibiting effects on the studied MDE based development process. These keywords and phrases were then categorized based on their effect on different aspects on the studied development process such as speed and quality. Based on the result of the categorization, we identified different types of inhibitors, the reason for these inhibitors and how the inhibitors mapped to the different roles involved in the studied process. From archival data, we extracted additional information regarding the roles, artifacts and activities that are part of the studied development process. This data was used to validate and complement the information gathered from the interviews and stakeholder meetings.

We then identified features and concepts of language workbenches that would address the possible causes found in the analysis of the interview. This mapping, between the causes for the inhibitors and the features of language workbench technology, was used as specification for a demonstrator which we iteratively developed using the Intentional Domain Workbench (see 3.4 Proof of Concept Implementation). Based on the features provided by the demonstrator, a new process for software interface development was designed.

3.4 Proof of Concept Implementation

A demonstrator for external interface definition for the studied development process was developed using the Intentional Domain Workbench. The mapping between inhibitors of process qualities and features of language workbench technology (see section 3.3) were used as specification for the demonstrator which was developed by two research students with no prior experience of IDW. The demonstrator was, for each

activity of the development process, compared in a step-by-step fashion to the studied MDE based software development process. The output artifacts of the demonstrator were compared to corresponding artifacts produced by the current tooling environment.

3.5 Development Effort Estimation

A qualitative comparison of the development effort of constructing the demonstrator was made between the Intentional Domain Workbench (IDW) and the current tooling environment based on Eclipse Modeling Framework (EMF). The comparison was based on expert estimations [20] for the EMF-based approach and actual development effort for the IDW approach. The estimates for the EMF-based approach with additional customized plugins were given by three tool developers in the interface domains in three separate sessions with duration of one hour per session. The tool developers were asked to use a bottom-up approach [21] to fulfill the values provided by the demonstrator. They would then proceed with breaking down the value to concrete tasks and provide an estimate in person weeks.

The estimates were subject to a number of constraints. First, estimators were instructed to give estimates for a tool developer with basic knowledge in EMF, Eclipse plugin development and interface definition development. Second, in case a EMF-plugin was used, they would need to include the time it would take to familiarize with the plugin. A template listing the constraints and instructions were used to aid the estimators (see Appendix A). Furthermore, in order to maximize the accuracy of the provided estimates, a subset of Jørgensen's [21] expert estimation guidelines were applied on the design of the estimation. Specifically, to reduce situational and human biases the estimation sessions were designed to address:

- **Avoidance of high evaluation pressure**

Monetary awards and other incentive were not given to the estimators in order to avoid evaluation pressure. In addition, the estimators were selected from an interface definition domain with no relation to the one in the studied context. Furthermore, the demonstrator was presented as a proof of concept implementation not for actual deployment.

- **Estimators were asked to justify and criticize their estimates**

At the end of the estimate session, the estimators were asked to criticize the estimates. Afterwards, they were presented with the opportunity to change their estimates.

- **Use documented data from previous development tasks**

During the estimation session, the estimators were asked to relate to any existing data of development effort from previous development tasks.

- **Estimate Experts with Domain Background**

The estimators were tool developers with significant experience of the current tooling environment, specifically EMF and Eclipse plugin development. In addition, the developers also possessed extensive domain knowledge of interface definition development.

3.6 Usability Study

The objectives for the usability study are threefold: first, to collect quantitative and qualitative data to determine the influences of using the demonstrator, MOM Workbench, on process qualities of an interface development process, RBS MOM process; second, to identify usability issues related to the MOM Workbench and the Intentional Domain Workbench; finally, to provide recommendations for improvement of the identified usability issues.

From the objectives, the usability test was designed to investigate the following questions:

- How easily do users get started on defining a delta MOM in the MOM workbench?
- How easily and successfully do users complete the task of defining changes in a delta MOM?
- How well does the MOM workbench support the goals of users? i.e. Does the MOM workbench match their mental model of how a tool should behave?
- What are the users' experiences of the MOM Workbench with respect to viewing, defining and assessing changes to MOM, compared to the current way of working?

Usability Measures.
The usability evaluation measures are a combination of quantitative and qualitative measurements consisting of user comments during testing session and from interviews, post-scenario questionnaires and observations. The measurements are listed as following:

- Number of participants completing a scenario.
- Number of participants failing to complete a scenario.
- Rating of level of difficulties of scenario in the post-scenario questionnaire.
- Observation of behavior during test session.
- Verbal comments when think out loud.
- Post-scenario questionnaire answers.
- Post-briefing interview answers.

Usability Analysis.
From the data collected in the usability sessions, usability problems were identified based on issues the participant had while interacting with the MOM Workbench. The identified usability problems were then categorized according to the Usability Problem Taxonomy [16], severity and origin [22]. Origin ratings were given based on the experience of the MOM Workbench developers and were in addition, confirmed with

a developer of the tool platform vendor, Intentional Software. Severity ratings were based on user observations, comments and the outcome of the scenarios in the usability test session.

Test Environment and Equipment.
The usability test was conducted both locally at the site of the participants at Ericsson Lindholmen and remotely through video-conference with users located in Ericsson's offices in Sweden. In the case where the usability tests were conducted remotely, the participants remote controlled the demonstrator which was running on the same laptop that was used in the local test environment.

Local Test Site at Ericsson Lindholmen:

- Laptop configuration (Windows 7, LAN network connection)
- Keyboard and mouse
- 19" Monitor, 1280x800 resolution
- Audio recorder

Remote Test Site:

- Laptop configuration (Windows 7, LAN network connection)
- Mouse
- 15" Monitor, 1280x800 resolution
- Audio recorder
- Video-conference equipment (40" Display, HD camera, microphone)
- Microsoft Lync for remote control of the MOM workbench

User Profiles.
User profiles were defined based on the analysis of the interview data from users in the RBS MOM process, a software interface development process (see section 4.1). The user profiles are listed in Table 1. The characteristics of the profiles were used for user recruitment in order to ensure that the target group is representative of the actual users.

Table 1. User profiles for the usability test of the MOM Workbench.

Role	Characteristics
Feature developer	• No or minimal experience of using modeling tools. • Must have significant domain knowledge of MOM. • Must have experience of defining a delta MOM. • No experience of language workbenches.
Modeling expert (part of review group)	• Some experience of using modeling tools. • No experience of language workbenches.

	• Optional: Domain knowledge of MOM. • Optional: Experience of defining delta MOM.
Domain expert (part of review group)	• No or minimal experience of using modeling tools. • No experience of language workbenches. • Must have significant domain knowledge of MOM. • Must have significant experience of defining delta MOM.
Model developer	• Must have experience of using modeling tools. • No experience of language workbenches. • Optional: Domain knowledge of MOM. • Optional: Experience of defining delta MOM.

Test Scenarios.
The participants were given six scenarios to complete on the MOM Workbench. The scenarios are listed below. The scenarios were given one at a time to the participant who was asked to think-out aloud while performing the scenario.

Participants

Five users of the RBS MOM process participated in the study. A user may have tasks that spans across several roles in the process. However, users were asked to rank their primary and secondary roles. Table 2 describes the participants and their roles in the RBS MOM Process.

Table 2. Participants and their roles.

Participant	**Roles (Rank is given from left to right)**
P1	Feature Developer, Domain Expert
P2	Feature Developer, Domain Expert
P3	Model Developer, Modeling Expert
P4	Modeling Expert
P5	Feature Developer, Domain Expert

Session Outline
A test session lasted for approximately 85 minutes including the following:

- Introduction: 5 minutes
- Training: 25 minutes

- Task scenarios with post-scenario questionnaires: 45 minutes
- Post-test debriefing: 10 minutes

Introduction (5 minutes)

The moderator welcomes the participants and explains the purpose of the usability study and how it relates to thesis study. Next, the moderator informs the user about the test environment outlines the session and asks the user to think-out aloud when performing task scenarios. After answering questions, the moderator proceeds with the training session.

Training (25 minutes)

The training session consists of introduction to the concepts related to navigation and selection using the Intentional Domain Workbench. The training is performed on an example workbench, Entity Workbench, unrelated to the MOM Workbench. Each concept is followed by an exercise where the concept is exemplified.

Post-scenario questionnaire

The participant was asked to complete a questionnaire after each scenario. The questionnaire was done through an online survey. The questionnaire was designed to assess the user satisfaction among usability aspects of the MOM Workbench according to categories in the UPT. Table 3 shows the mapping between the questions in the questionnaire and UPT categories.

Table 3. Categorization of post-scenario questionnaire.

	Question	1st Category	2nd Category
S1.1	Understanding the workbench layout	Visualness	Task-mapping
S1.2	Understanding the workbench terminology (words, name of element, icons, etc)	Visualness	Language
S1.3	Understanding the first step you need to do to define a delta MOM	Manipulation	Task-mapping
S2.1	Task (Add a new delta MOM and set it as current)	Manipulation	Task-mapping
S3.1	Task (Define changes in a delta MOM)	Manipulation	Task-mapping
S3.2	Understanding the procedures of defining a delta MOM	Visualness	Task-mapping

S3.3	Understanding the presentation of changes in MOM	Visualness	Task-facilitation
S3.4	The projections are helpful in viewing the changes I defined in delta MOM	Visualness	Task-facilitation
S4.1	Task (View different delta MOMs)	Manipulation	Task-mapping
S4.2	Overall ease or difficulty of comparing changes in different delta MOMs	Visualness	Task-facilitation
S4.3	This feature of MOM workbench is helpful to preview changes in different delta MOMs?	Visualness	Task-facilitation

Scenario 1: First Impressions

You want to add a new feature to RBS. In order to implement the feature, you need to define a delta MOM which will contain a set of changes to the current MOM model. Now you will use MOM workbench to create an empty delta MOM.

What you see in front of you is a model file ”Test Model” containing a subset of the APC MOM and a deltaMOM “DMOM SV2310”

Take a moment to look at the layout of the MOM workbench without changing anything (You may click and navigate around). After you have familiarized yourself with the workbench, please tell us: What do you think you would do on the MOM workbench to create an empty delta MOM?

Scenario 2: Creating a new delta MOM

You are going to define a delta document for the new feature. You want to 1) create a delta MOM in the model, 2) give its name as dmom_20130417, and 3) set it as current delta MOM. Then save the document.

Scenario 3: Specifying changes and editing experiences

Now you want to define changes to the newly created delta MOM. Your delta MOM contains the following changes:

1. You want to add a new Action with following data:

   Class: **SSysConfiguration**
   Action Name: **configureSupportSystem**
   Description: **Configures the support system.**
   ReturnType: **void**
   Raised Exceptions: **SupportSystemConfigurationFailed**
   IsTransactionRequired: **True**

2. Now you want to change the type of a Struct Member, '**measured ElapsedTime**' to:

   Datatype: **MoString**
   Range: min: **1**
   max: **99**

3. In the end, you want to delete an enum member, **UNDEFINED**, from enum **BatteryTestState**.

Scenario 4: Comparing delta MOMs

You find that there exists another delta MOM that realizes the same feature with different a solution. Now you want view the changes in that delta MOM applied to the model "Test Model". Set DMOM_SV2310 to current delta MOM and display the changes in the model "Test Model".

Scenario 5: Comparison between delta MOM projection and current way of viewing changes using Excel spreadsheet

Open the Excel template for delta MOM without clicking on anything. Tell us what you think of this template compared to the MOM workbench

Scenario 6: Comparison between MOM projection and current way of viewing MOM using HTML

Open the HTML-file of the APC MOM and navigate around. Tell us what you think of this view compared to the MOM workbenches

Post-test debriefing (10 minutes)

The post-test debriefing was conducted with the goal of collecting the participant's overall impression of using the demonstrator. The participant was also asked about the usefulness of the MOM Workbench.

4 Result

This chapter presents results from the analysis of the conducted case- and usability study. First, the current process of software interface development is presented together with identified inhibitors. Then, we describe how a demonstrator based on IDW, addresses the identified inhibitors. We also present a comparison of development effort of constructing a technical equivalent of the demonstrator based on the current tooling environment in the studied case. Quotes have been taken from the interviews, stakeholder meetings and usability testing sessions in order to strengthen our claims presented in subsequent sections. Minor changes have been made to the quotes in order to make them more readable.

4.1 Current Process

Roles.

The development of software interfaces involves mainly the roles listed below.

Feature developers are responsible for defining requirements on interface model which fulfill requested features. Feature developers have knowledge on solving problems in the telecom domain. Although many of them are familiar with modeling, few have knowledge in using MDE tools.

The Review group consists of two types of reviewers: domain experts and modeling experts. Domain experts validates that the proposed changes satisfy requested features, while the modeling experts make sure that the proposed changes follow the principles of the design of interface model. The group reviews a set of delta documents at weekly meetings and can reject the change request with feedback.

Model developers integrate the changes in delta documents to the interface model using an EMF-based modeling tool. Contrary to feature developers, model developers have a stronger background in model driven engineering with knowledge in using MDE tools, while less knowledgeable about the problem domain.

Artifacts.

An Interface Model is a model describing the software interfaces in radio base stations. The interface model is defined using an UML-profile based meta-model in an EMF-based modeling tool. All entities in the interface model need to follow the design rules which are constraints from the problem domain.

Delta document contains a set of proposed changes to the interface model. The delta document describes what to be changed in the software interfaces. Each change refers to requirements of a specific feature. Thus one delta document represents one possible solution for realizing the requested feature. Several delta documents can be proposed as solutions to realize a certain feature. The delta documents are stored as spreadsheets or text documents which are not interpretable by the current interface development environment.

Deliverables are automatically transformed from the interface model(s) using an EMF-based modeling tool. Deliverables are stored as structured text or binary files, which are input to different deployment processes.

Development Process.

Figure 6 presents the current software interface development process in the case study. As shown in the figure, an MDE based approach is adopted to automate the transformation from the interface model to deliverables ready for deployment.

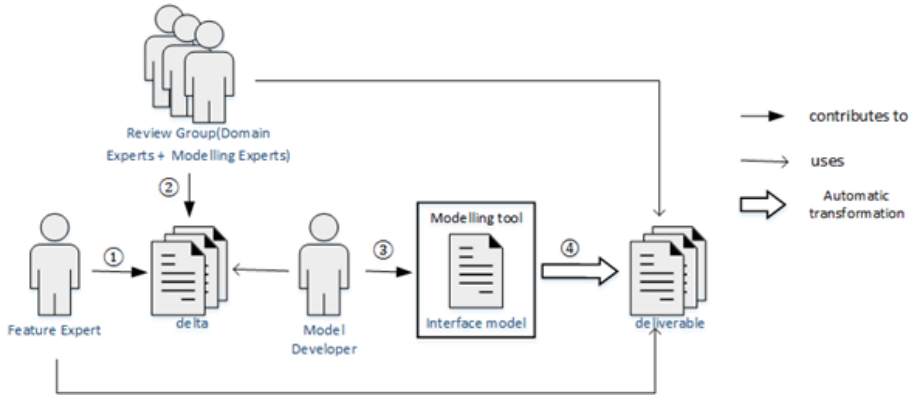

Fig. 6. Software interface development process for radio base stations

We illustrate the current development process with an example in order to explain the roles, the interactions between the roles and the activities in the process. Consider the development of software components in radio base stations in the context of network management. Usually, during the development of software components, changes are requested for reasons such as changing needs in the market. Once change requests are accepted for the next release of the software components, the change requests are analyzed for feasibility from a technical point of view. In this specific example, let us consider a request for a new feature.

First, feature developers who are responsible for the particular feature analyze the changes that are required to the existing software component. If there is a need to make changes to the component, the component's software interface must also be changed in order to support the feature (1). This is done by the feature developers who define a set of changes in a delta document. In the specific context of the study, the software interfaces of the components are defined as models using a UML based modeling tool.

Once the feature developers are satisfied with their solution, the delta document is evaluated by a review group responsible for the affected software component interfaces. The review group evaluates a certain number of delta documents in a review meeting (2). In a review meeting, the review group validates the proposed changes according to predefined design rules and assesses the maturity level of the delta documents. At the end of the review meeting, the review group makes the decision whether to approve the reviewed delta documents or not. If the delta document did not pass the review, feedback is sent to the responsible feature developers who may decide to refine the delta document to be considered in the next review meeting.

After a delta document gets approved, the document will be handed over to model developers. The model developers are responsible for manually integrating the changes in the delta documents to the interface model using specific modeling tools (3). Once the delta documents are integrated to the model, automatic transformations (4) can be invoked to obtain the deliverables which are then used in the deployment of the new version of the software component.

4.2 Inhibitors in the Software Interface Process Development

From analysis of the interview data, inhibitors (see Table 4) were identified in the current development process.

Table 4. Inhibitors in the Software Interface Process Development

	Inhibitor
IH1	Semantic gap between delta document and the interface model
IH2	Manual transformations
IH3	Assess impact of change requests to the interface model
IH4	No traceability between interface model and requirements
IH5	Inhibitor: Dependency on modeling tooling expertise

(IH1) Inhibitor: Semantic gap between delta document and the interface model

Feature developers specify changes to the interface model through delta documents. The specification of changes is expressed using concepts in the interface domain which is represented as natural language in a delta document. To implement the changes to the actual model, model developers need to translate these changes to concepts in the modeling tool. This semantic gap causes communication problems between feature developers and model developers which increase the development time. A feature developer expressed the following:

> *"The persons creating delta MOM are not working with the actual models. So maybe they can explain in text what they want to be changed. Then there is another person who is supposed to interpret the change. It can happen that the model developer goes back to the feature developers and say: What do you mean by this?"*.

This is further confirmed by a model developer:

> *"Not everyone is used to the delta document. Perhaps we need some kind of intermediate format to make discussions easier."*

(IH2) Inhibitor: Manual transformations

In the current development process, artifacts, namely the delta document and the interface document are stored in different data formats. Currently, no automatic transformation exists between the data formats. For example, when changes defined in a delta document are to be integrated to an interface model, the integration is done manually by model developers. Both model developers and domain experts find the process of manual integration tedious and error prone. One review group member said:

"*The quality of delta document is a problem. One of our task is to check spelling mistakes. [...] There are several spreadsheets in a delta document. It is so easy to make mistakes during implementation to the interface model*".

(IH3) Inhibitor: Assess impact of change requests to the interface model

In order to assess the changes, feature developers and the review group rely on information that is stored in two separate files: the delta document and the interface model. Typically, a feature developer or a review group member needs to create a mental model and then apply the changes to this mental model to assess the impact of the changes. One of the domain experts explains the process as the follows:

"*For example, a proposed change is to add a new attribute to a class. When the review group assess this proposed change, they need to check if the attribute is already visible somewhere else, whether it is proper to do that. In order to assess that, people need to remember the interface model in mind*".

This is an activity that requires experience and becomes even more difficult if the interface model is large and complex. The same domain expert said:

"For someone not familiar with the interface model, it is difficult to navigate in the model". Domain experts also expressed difficulties in assessing which elements of an interface model are affected by a certain change:

"*A good idea would be...for a certain change, which elements in the interface model are affected.*"

(IH4) Inhibitor: No traceability between interface model and requirements

In the current development process, requirements of features are not modeled in the interface model. Instead a change in a delta document contains references by name stored as a string, to requirements. As a consequence, when changes of a delta document are integrated into an interface model, references to requirements are lost. Traceability of requirements is import in the review of delta documents, especially in cases where the review group compares a specific delta document with alternative delta documents:

"*It is interesting to keep requirement and feature information. For example, when the review group assess a delta document, they want to know if this solution had been proposed before and its alternative solutions to the same problem*".

Currently, a review group member needs to rely on memory to find changes in alternative and previous delta documents that are related to certain requirements.

(IH5) Inhibitor: Dependency on modeling tooling expertise

In the current development process, the integration of changes in a delta document is done by model developers with expertise in a certain modeling tool. As several delta documents can be reviewed at the same time, the number of model developers may become a bottleneck in situations where the rate of processed delta documents are higher than the rate with which model developers can integrate delta document changes. A domain expert in a review group phrased it as:

"[The number of] Model developers would be a bottleneck in the process if the workload is high".

A wider adoption of the current tooling among domain experts is also not likely due to the cost of training and deployment of the tooling.

4.3 A Knowledge Workbench for Software Interface Development(KWSID)

Our solution to address the inhibitors in the current development process is a Knowledge Workbench, for definition of software interfaces (KWSID) with following features (Table 5):

Table 5. Features of a KWSID

	Features
KWF1	DSLs for Software Interface Development
KWF2	Defining delta in interface model
KWF3	Tailored Projections
KWF4	Previewing changes in delta on interface model
KWF5	Validation of domain code
KWF6	Merge delta to interface model
KWF7	Generator for deliverables

(KWF1) DSLs for Software Interface Development

The KWSID implements DSLs for specifying software interfaces, in particular DSLs the interface model and delta document specification. These DSLs also support additional information used by the roles in the development process such as requirement traceability in interface models.

(KWF2) Defining delta in interface model

The KWSID defines a domain schema for containing both interface models and delta documents in a single file format. In KWSID, domain code for interface models and deltas can be mixed in the same document. The deltas are synchronized with the interface models by direct references, thereby providing the ability to directly specify changes that are consistent with an interface model.

(KWF3) Tailored Projections

Each role in the current development process is provided with editable projections synchronized with the underlying domain code which means changes made to the

domain code through one projection will also be reflected in other projections. Modern IDE features to aid the editing experience such as code completion and syntax highlighting are also provided by KWSID.

(KWF4) Previewing changes in delta on interface model

While feature developers are specifying changes in a delta, the changes are previewed on the interface model. Change markers are supported to indicate the type of change, the delta for which the change belongs to and the previous value before the change. Currently, changes are only color coded to mark the type of change. For feature developers, domain experts and modeling developers, the preview capability facilitates the process of assessing impact of changes to the resulting interface model. Furthermore, KWSID provides the functionality of comparing deltas by selecting which delta to preview on an interface model.

(KWF5) Validation of domain code

Constraints for some of the design rules used by the review group are implemented as validation rules in the KWSID. Validation rules are checked continuously while the user is editing the KWSID document.

(KWF6) Merge delta to interface model

In KWSID, model-to-model transformations exist to allow changes in a delta to be integrated to the affected interface models.

(KWF7) Generator for deliverables

KWSID defines generators which specify automatic model-to-text transformations from the interface model to the required format of the deliverables.

4.4 A New Process for Software Interface Development with KWSID

The KWSID can be applied to the current software interface development process in order to address the identified inhibitors. Figure 7 illustrates the new process.

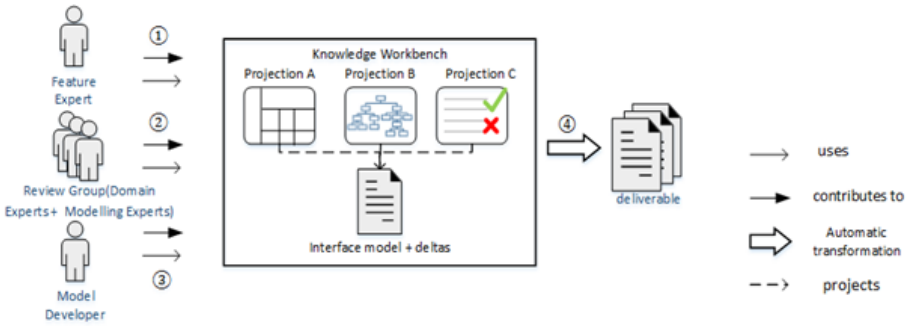

Fig. 7. A new development process enabled by KWSID for software interface development

The inhibitor related to manual integration (IH2) is addressed by introducing automation by (KWF1) and (KWF2). As interface models and changes in a delta document are combined in one KWSID document, the changes in a delta document are integrated to the interface model by transformations defined in the KWSID. As a result, the activity of manual transformation performed by the model developer is replaced by automatic transformation. Similarly, the manual transformation done by feature developers when defining delta documents is replaced by directly editing the KWSID document which may contain domain code for delta documents and the interface model.

For every role in the development process, KWSID provides tailored projections (KWF3) with domain-specific views containing information and commands specifically targeted to the role. Having tailored projections with editing aid may reduce user errors which are related to (IH2). For example, when a feature developer is creating a delta document, KWSID provides continues validation (KWF5) in order to restrict the feature developer from violating design rules of a certain interface model.

However, tailored projections are primarily used to address (IH3). The review group reviews delta documents with a projection providing preview capability to assess the impact of changes (KWF4). For the review group and feature developers, rather than creating a mental model and imagine the applied change, they are given graphical and textual visualizations of the changes previewed on the interface model. In addition, the review group is provided with the ability to comment changes, trace requirements to elements in interface model and set the maturity level of delta documents. This type of information is preserved for later use in the discussions of the review group (IH4).

The KWSID reduces the workload of model developers (IH5) due to effects of (KWF3, KWF6, and KWF7). The model developer role has changed from manually integrating delta document to maintaining the KWSID. KWSID provides a targeted projection (KWF3) which summarizes the changes in a delta document (KWF2). In the same projection, automatic transformation (KWF6) can be invoked to integrate the changes to the model. The integrated model can then be transformed into deliverables of chosen data formats. In effect, the features related to the model developer eliminates the need for a using a specialized modeling tool for integration of delta to the interface model.

Table 6. *Comparison between the current development process and the new development process enabled by KWSID*

Role	Current Software Interface Development Process	New Software Interface Development Process with KWSID
Feature Developer	• Specifies changes in structured text with no reference to actual interface model • Assesses impact of changes on interface model using a mental model.	• Specifies changes in a preview mode which shows how the changes will affect the interface model, and with automatic validation. • Assesses changes through pro-

		jections showing previews of how the changes will affect the interface model.
Review Group	• Manually validate design rules. • Assess impact of changes with mental model. • Has no traceability support from requirements to the interface model.	• Assess changes through projections previewing how the changes will affect the interface model. • Has traceability of requirements. • Adds information which review group is preserved in the interface model. • Validates input
Model Developer	• Manually integrates changes using modeling tool. • Generates deliverables from models by automatic transformation	• Merges changes to interface models by invoking automatic transformations. • Views summaries of changes in delta document. • Generates deliverable from interface models by invoking automatic transformations. • Maintains the KWSID.

4.5 MOM Workbench- A Prototype of KWSID

A KWSID, MOM Workbench, was developed using the Intentional Domain Workbench (IDW).

The user interface of the MOM Workbench is presented in Figure 8 and Figure 9. Below follows descriptions of the user interface objects shown in the figures.

User Interface Objects

① Table of Contents (TOC)

The Table of Contents (TOC) is a projection showing the tree structure of the model entities and their relations.

② MOM-Projection

The MOM-Projection is the user interface object for presenting and interacting with the model. The MOM-Projection provides the capability of directly specifying changes to the MOM model. Made changes are automatically recorded to the delta MOM instance set as current. Preview is another feature in the MOM-projection.

When changes are made to a MOM entity, the MOM entity is actually not changed, instead the changes are overlaid on top of the MOM. Thus, preview of changes is possible by setting a delta MOM as current.

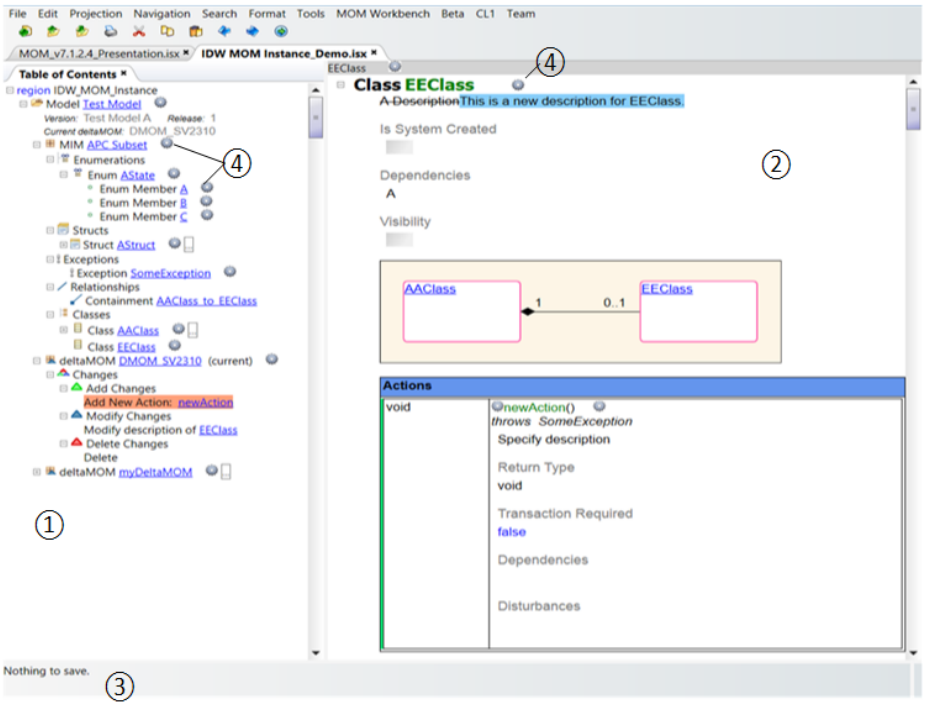

Fig. 8. The user interface of the MOM Workbench

③ IDW Status Bar

The IDW Status Bar shows messages that are defined by the MOM Workbench and the IDW.

Messages include system error messages, error and feedback messages of operations performed by the MOM Workbench.

④ The Operations Button

The Operations Button is a user interface object that upon activation displays a context menu with commands that are related to the model entity the button belongs to. For model entities displayed in the MOM-projection and TOC, Operation Buttons are depicted as cogs.

⑤ Delta MOM Projection (Figure 9)

The Delta MOM Projection presents a summary of the changes made to a MOM entity in the model. It also functions as a specification of how to define changes to a

MOM entity. Users can in addition specify changes to a MOM entity by using the Delta MOM Projection. Each change corresponds to an element in the list of changes which are categorized in sections related to their type: Add Changes, Modify Changes and Delete Changes.

⑥ Change Row (Figure 9)

A Change Row in a delta MOM corresponds to a single change request to a MOM.

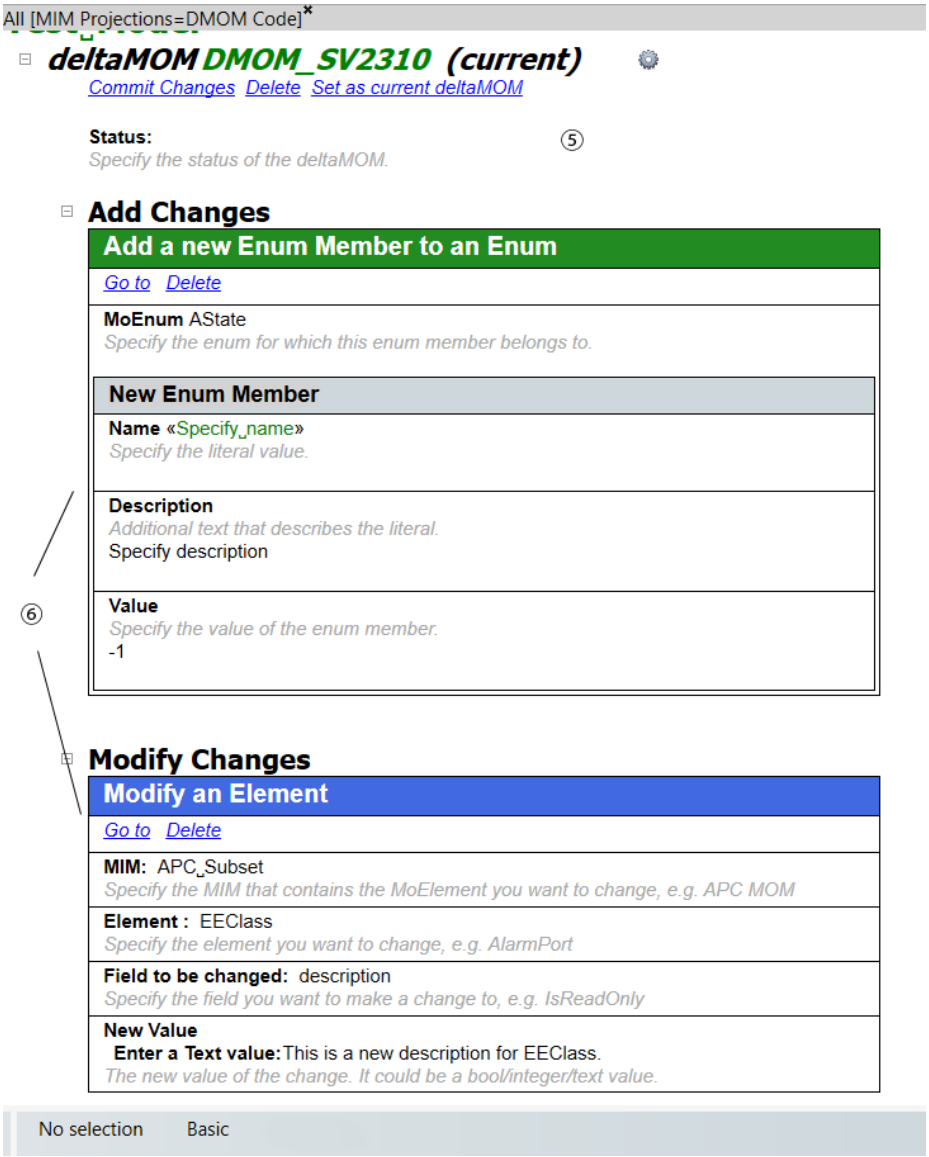

Fig. 9. Delta MOM Projection showing a summary of specified changes

4.6 Development Effort Comparison between Intentional Domain Workbench and Current Modeling Tools

A comparison of development effort between IDW and the current tooling in the studied case was made in order to estimate the costs of developing a DSL application providing the same benefits as the KWSID.

The estimates given for the EMF-based approaches were based on the realization of the features of the KWSID. From the features, three values provided by the KWSID were identified: ability to specify changes to interface models enabled by meta-models of the interface- and delta model; automation through integration of delta model to interface model; tailored projections with specific commands such as previewing changes for feature developers, review group and model developers.

The results of the estimations are listed in Table 7 and illustrated in Figure 10. Three estimates were provided, showing that IDW decreases the development effort of building a DSL application in average with three times compared to the EMF-based approaches. For all estimates the effort of implementing the domain for interface model and delta model is approximately the same with a slight advantage to IDW. The effort for introducing automation of integrating delta model to interface model takes in average two times more effort for the EMF-based approach. The main difference in effort is from the implementation of projections where the EMF-based approaches take in average 3.5 times more effort than using IDW.

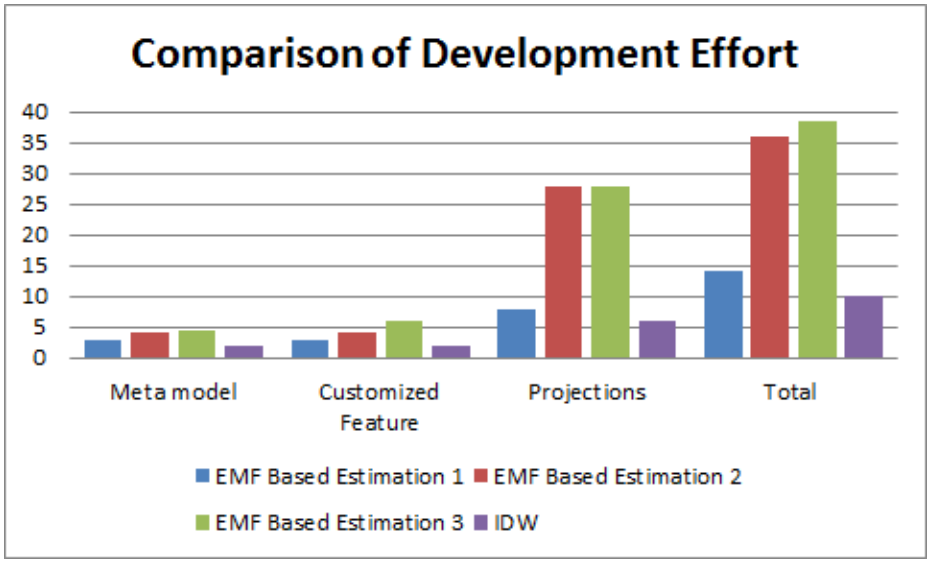

Fig. 10. Estimations of development effort for a DSL application providing same value as KWSID

Table 7. Estimations of development effort for a DSL application providing same value as KWSID. The development effort using Intentional Domain Workbench is based on actual data from the implementation of the MOM Workbench. The unit "x" denotes the development effort of a person per time unit.

Value	**EMF Estimation 1**	**EMF Estimation 2**	**EMF Estimation 3**	**IDW (based on MOM Workbench)**
Meta-model for delta model and interface model	3x	4x	4.5x	2x
Automation (merge delta to interface model)	3x	4x	6x	2x
Projections for feature developers, review group and model developer	8x	28x	28x	6x
Total	14x	36x	38.5x	10x

4.7 Usability Study Outcomes

This chapter presents the findings from the test scenarios, post-scenario questionnaires and post-test interviews.

Findings for Scenarios.

Findings for Scenario 1: First Impressions

Four out of five participants found the terminology of MOM workbench (name of elements, icons, etc.), easy to understand, whereas one had no opinion. All participants understood the names of the element types. Although none of the participants recognized the icons used for the elements, they did not found this conflicting nor unpleasant. Three participants thought that the navigation of the MOM model was easy to understand, while one (P1) found it difficult due to lack of understanding the model structure. In general, participants understood the layout and the function of the

delta MOM and MOM-projections; however, they had different opinions of how the presentation of the information and commands should be done.

Regarding participants expectations of the first step to create a new delta MOM, two participants found it easy (P2, P5); two (P3, P4) had no opinion and one (P1) found it difficult. Participants who found it *Easy* and *Neither Easy or Difficult*, expressed that the first step corresponded to their expectations of creating the delta MOM through a menu item in the menu bar. However, all participants expected the menu item to be located in the File menu. The participant who found it *Difficult*, located the correct menu item after a considerably amount of time due to confusion caused by the available menu items. The participant would have preferred to create a delta MOM through right-clicking on the root element. During scenario 1, observations indicated that the large number of menu items unrelated to the MOM Workbench caused minor confusion for all participants: they scanned through all menus and submenus in sequence to locate the correct menu item. In all cases the participants found the correct menu item, however, the large number of irrelevant menu items delayed the completion of the scenario.

All participants expected that the first step of creating a delta MOM would be through a menu option in the top menu bar. Three participants (P1, P3, P5) also tried to right-click on the whites space area in the Table of Contents or the root element. The order for which the participants (P) followed in order to find a way to create a delta MOM can be described as follows:

1. P tries to find menu item by selecting the File menu.
2. P right-clicks on the root element or a white-space area in a tab to find the menu item for creating a delta MOM.
3. P left-clicks on icons and symbols to find a menu item.

General Impression

- "*This is more about getting used to the icons and symbols*" — P1 about the icon and symbols used.
- "*When it comes to understanding the first step of defining a new delta MOM, it is neither easy nor difficult. It is just somewhat unfamiliar in a unnecessary complex way. Just because the layout is complex it doesn't make it difficult. It is just not what I would like to have basically*" — P4
- " *I am not familiar with the icons but I understand the names of the elements*" — P2
- "*When I do(define) a delta MOM I would like to have an easy way to see what is the current MOM that I am applying my delta MOM to and we see that in this view over here.*" — P5
- "*[...] Then we have a specification here of what to change and how*." — P5 about the dMOM-projection.

Positive Comments

- *"I like that the colors on the delta MOM changes [in the Table of Contents] are green for new, blue for changed and red for deleted."* — P1 about the icons used for signifying the type of change in a delta MOM.
- *"The menu alternatives were quite expressive so I understand what they are doing."* — P3
- *"The layout is good"* — P2

Negative Comments

- *"By looking at it in the beginning, I didn't see the structure, so I didn't understand what to find. I saw the MIM structure, it was quite easy. Then I didn't understand that the delta MOM that was there was what I was actually not supposed to use. And I was supposed to create a new one"* — P1
- *"Why is there a big red cross...I get the impression that I have done something wrong"* — P1
- *"When I open up the thing, the first thing I expect to see in the middle is not adding a new enum in a delta MOM. Then you are in the middle of something. If I open up a delta MOM document and end up in the middle of the structure somehow."* —P4

Suggestions

- *"I would like to start by right-clicking on the top package [root element]"* — P1 about the first step of adding a new delta MOM.
- *"I want to have the big picture.[...] I want to have instruction or introduction telling why did you add this enum."* — P4 about the presentation of the delta MOM projection.

Scenario 1: Post-task Questions with Responses
1. Understanding the workbench navigation

	Very Easy	*Easy*	*Neither Easy nor Difficult*	*Difficult*	*Very Difficult*
Participant		P2, P3, P5	P4	P1	

2. Understanding the workbench terminology (words, name of element, icons, etc.)

	Very Easy	*Easy*	*Neither Easy nor Difficult*	*Difficult*	*Very Difficult*
Participant		P1, P2, P3, P5	P4		

3. Understanding the first step you need to do to define a delta MOM

	Very Easy	*Easy*	*Neither Easy nor Difficult*	*Difficult*	*Very Difficult*

Participant		P2, P5	P3, P4	P1	

Findings for Scenario 2: Creating a new delta MOM

All participants completed the scenario. Participants found the task *Very Easy* (P1, P3, and P5) and *Easy* (P2, P4). The ease of this task is related to the minimal and straightforward interaction with the Mom Workbench. Overall, the task corresponded to the participants expectations of how to edit the name of an element and invoke the creation of a new delta MOM. However, some participants (P3,P4,P5) would prefer to right-click on an element or the white space area of the tab to get a context menu with operations that can be performed on the element. For instance, right-click on a delta MOM(Figure 11) and find menu items for save, delete, set as current and so on. Participants were observed to interact with the Table of Contents by right-clicking the delta MOM to set it as current. This is explained by their previous experience of window systems where they expect the TOC to behave as other file browsers where they can right-click an element to get a context menu.

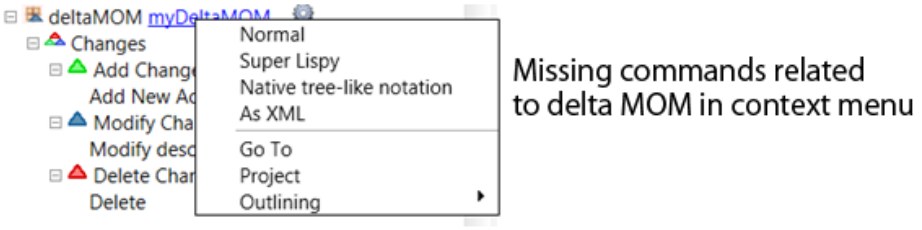

Fig. 11. When participants right-clicked on the delta MOM element, they expected to find commands to operate on the delta MOM, e.g. Set as current, save and delete.

Positive Comments

- *"This was very intuitive and self-explaining... Mark the name, type what you want and then set it to current by a button."* — P1
- *"Everything was visible and the short training I received before. A combination of everything made it easy."* — P2
- *"I think it was quite easy. Almost like a normal tool, you can left-click to select and right-click to get options and everything."* — P3
- *"The fact is, it is a single click action. You don't have any unnecessary interaction with the tool."* — P5

Negative Comments

- *"I expected some options to save and other things"* — P2 tried to right-click on the delta MOM in the TOC.

Suggestions

- *"In most programs that I am used to, when I right-click I get the menu alternatives for the specific item. Like in windows, you get all the fancy stuff by right-clicking."* —P3
- *"My intuitive feeling is to go to the file explorer thing (Table of Contents) to change the name."* — P4
- *"The tool should preferably place the insertion cursor in the name so you don't have to click on it and then type the name."* — P5

Scenario 2: Post-task Questions with Responses

1. This task is...?

	Very Easy	*Easy*	*Neither Easy nor Difficult*	*Difficult*	*Very Difficult*
Participant	P1,P3, P5	P2, P4			

Findings for Scenario 3; Specifying changes and editing experiences

Four out of five participants completed the scenario. Two participants (P1, P2) did not meet any major issues that hindered their tasks. Three participants (P3,P4,P5) encountered issues that delayed or prevented the completion of their tasks. The issues are related to problems with manipulation, selection and presentation of user interface objects.

Four participants did not understand the meaning of the Operations Button presented as a cog; the preferred way of accessing commands is through context menu by right-click or top menu bar. Participants also preferred to only display a single delta MOM in a separate tab while defining changes however, they would like to have a list of delta MOMs to get a summary changes made to MOM entities.

Concerning global problems of presentation and manipulation, participants did not know which user interface objects can be selected or manipulated due to missing visual cues. When participants tried to edit fields of new elements that were created from the MOM-projection, they were confused of where to find the input field. For instance, participants (P2, P4) did not understand that the grey area under a property was a visual cue for inputting values. Furthermore, input fields did not display the latest input character, which caused confusion of whether the participant had entered a character or not. Figure 12 and 13 shows a selection of the mentioned problems. In addition, all participants had problems of viewing content of a tab due to missing horizontal scrollbar.

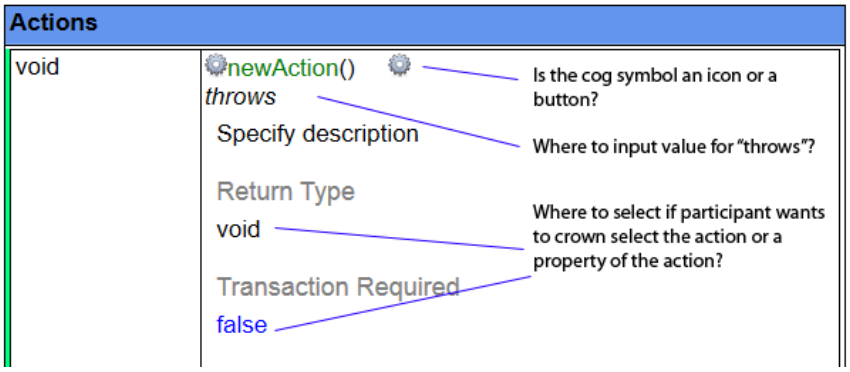

Fig. 12. Missing visual cues cause difficulties with understanding of which user interface object can be interacted with

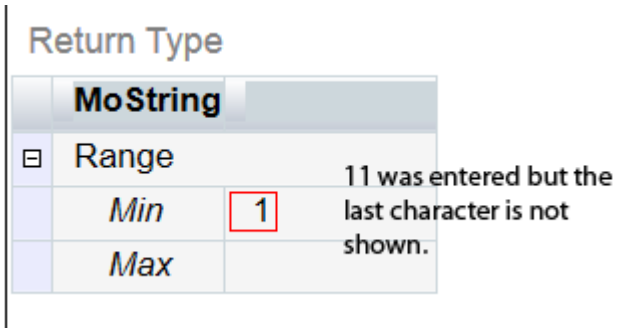

Fig. 13. Problems with manipulation of user interface object when participant input values

Participant P4 did not complete the scenario due to problems related to selection and editing of values.

First, when the participant tried to select a property by left-clicking within a change row in the delta MOM-projection, the participant was confused as the selection did not correspond to the participant's expected behavior. The selection would result in three outcomes:

1. The tab will scroll down half a page.
2. The top element of the selection will be tree selected which will cause the tab to scroll down a few rows.
3. Selecting the property element.

Second, the participant had problems with selecting text values using click-and-drag with the mouse. If the mouse selection moves past the selected word, IDW will instead select the parent of the current node and shift the focus of the tab to display the parent, which disoriented the participant. This problem with text selection is illustrated in Figure 14 and Figure 15.

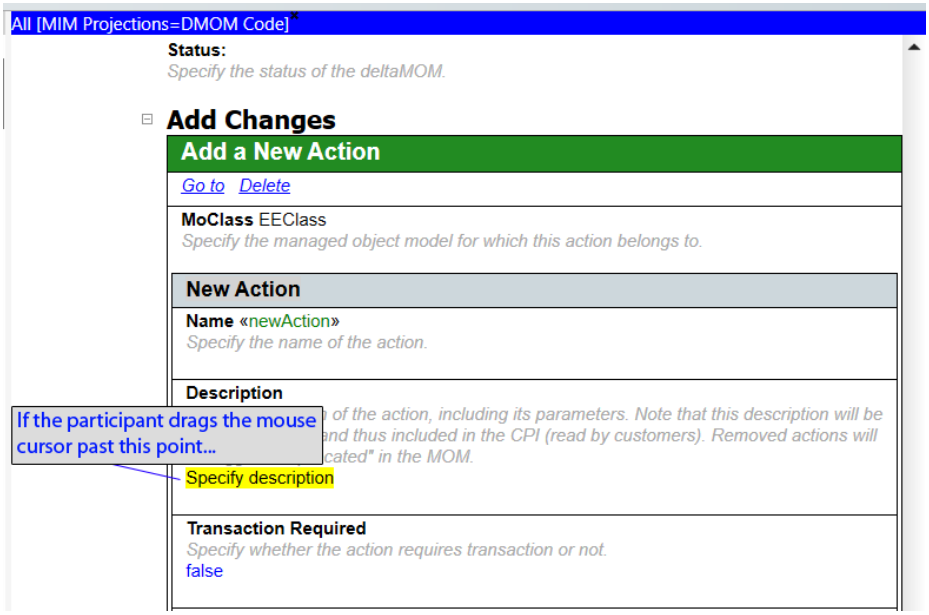

Fig. 14. Problem when selection of a text value moves past the word to be selected

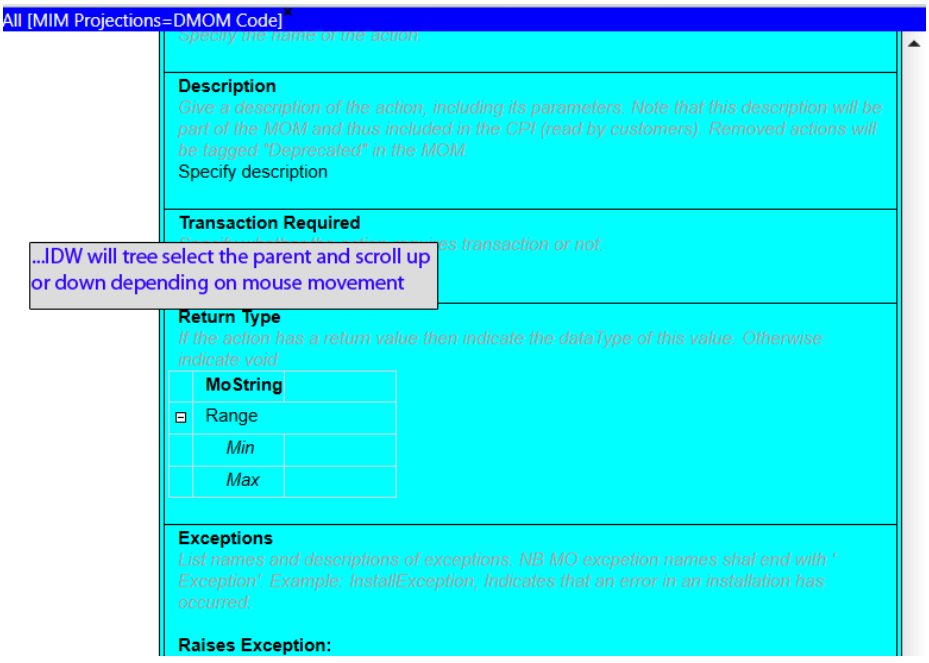

Fig. 15. The resulting behavior of IDW is tree selection of parent which will change focus and disorients users

Finally, P4 also experienced problems with the copy-paste functionality in IDW. In the scenario, the participant was supposed to specify a new action which throws an existing exception, *SupportSystemConfigurationFailed* defined in the MOM. The participant first specified the name of the exception in a text field, then copied the text value in the text field and pasted in the field for exception which expects a value of reference type. The result was that the newly created action was copied instead, and subsequently pasted into the field for reference. This behavior clearly confused the participant who gave up on completing the task.

Regarding the usefulness of the projections for presenting changes defined in the delta MOM, feature developers and domain experts (P1, P2, P5) found the ability of previewing changes to the MOM more useful than modeling experts (P3, P4). In their daily tasks, Model Developers primarily use the current modeling tool to understand how the changes specified in delta MOMs affect the models representing MOMs.

General Impression

- *"Shall I use this window to the right? Is there a search function?"* —P2
- *"It took some time before I saw it. I thought I need a bunch of options to make the changes so I clicked on it. I think I have seen it somewhere else."* —P2
- *"What is the thinking here. If I am creating something new deltaMOM and set it to current. I still have the same background information."* —P4 , did not notice that right tab updated
- *"Why would I have a form specify how to change? Because I just want to check the current state of my MOM and I would like to just to go in and start editing directly just like a Word Document. The tool would then pick up my changes and summarize it."* —P5

Positive Comments

- *"I thought it was easy to understand what I was supposed to click. Because if I want to change the enum or enum member there is the wheel icon."* —P1
- *"I think it was easy to understand the changes that I actually did to the model while doing the changes. I think that the projections that I could see are helpful as well."* — P1
- *"It is easy to use Mom Workbench because similar to HTML. To do the changes in the actual MOM is easy. For example, you don't need to know which project in RSA this MOM is in so it was easy and I liked it."* —P2
- *"It is easy to see what kind of changes I have done. It is useful. Usually it is red for deleted, blue for modified and green for new."* —P2
- *"I think it was good to see here a summary of the changes."* —P3
- "*That would be really useful because that RSA/RTE is not good in this.*" —P3
- *"Definitely I am not used to see the HTML pages. It would be much better if I can see what my colleagues sees, that I have the same picture as them. It would much easier to talk about things."* —P3
- *"That is really useful. You have started to have something that looks like a word processor"* —P5 about directly changing the MOM-view and changes recorded.

- "*This is nice. The fact that you can edit it here(dMOM) and you get the presentation here(MOM/ projection)*" —P5
- *"Well, the immediate visual feedback is what makes it easy."* —P5 what aspect made it easy for you

Negative Comments

- *"How do I mark just the description field..."* —P2
- *"It is hard to see..."* —P2 cannot see entire tab due to missing horizontal scrollbar
- *"I am used to RSA/RTE, this is almost completely text based. I don't really see the DMOM changes."* —P3
- *"So here we have a presentation that I definitely think that I dont want to see multiple delta MOMs at the same time."* —P5
- *"The wheel-icon could mean anything. It is fine if you in the beginning know where to look for commands. But that wasn't my first intuitive way of thinking."* —P5
- *"I don't have any idea what this wheel symbolizes"* — P4
- *"What is going on?"*—P4 confused about selection
- *"I don't think that this is usable. In all programs you can double click and copy the stuff."* —P4 about copy paste text to resolve to reference
- *"I didn't like it at all. Maybe because this is the first time. In my intuitive way to edit something it is not in line what how this tool behaves. I don't think this tool is very useful. I don't like it how it behaves. I only get irritated...I don't think it is difficult but I really don't like it at all."* —P4

Suggestions

- *"I would like to start by right-click to see where I can add a change"* — P1
- *"I want to press a button here so I can quickly see what has been changed or not (click a button to highlight the changes in previewed MOM tab)"* —P3
- *"I would actually expect empty instances of add changes, modify changes and so on. From the start when the delta MOM was created. Then I can right click on the instances and add the changes."* —P5
- *"There should be a graphical thing to show that there is a button here so I can add things to that class."* —P5
- *"Here is a question about scoping. At different times you would like to have types you have defined in your delta MOM, those you would like to see first. Those you are first choice. Then you would like to see the data types and work your way out."* —P5
- *"A list of delta MOM, I want to select which ones are presenting so I can also see them when I am editing my current delta MOM. The fact that you can present information from multiple delta MOMs on multiple edits is extremely valuable."* — P5

Scenario 3: Post-task Questions with Responses

1. Overall ease or difficulty of this task

	Very Easy	*Easy*	*Neither Easy nor Difficult*	*Somewhat Difficult*	*Difficult*
Participant		P1, P2, P5		P3, P4	

2. Understanding the procedure for defining a delta MOM

	Very Easy	*Easy*	*Neither Easy nor Difficult*	*Somewhat Difficult*	*Difficult*
Participant		P1, P2	P3, P5	P4	

3. Understanding the presentation of changes in MOM

	Very Easy	*Easy*	*Neither Easy nor Difficult*	*Somewhat Difficult*	*Difficult*
Participant		P1, P2, P5		P3, P4	

4. The projections are helpful in viewing the changes I defined in delta MOM

	Strongly Agree	*Agree*	*Neutral*	*Disagree*	*Strongly Disagree*
Participant	P2, P5	P1		P3, P4	

Findings for Scenario 4: Comparing delta MOMs

All participants completed the scenario. Three participants found it *Very Easy*; two found it *Easy*; one found it *Neither Easy nor Difficult*. The ease was related to lesson learnt from previous scenarios of how to access commands related to the delta MOM. The majority of the participants found the capability of previewing changes defined in delta MOM to the MOM useful. For Feature Developers and Domain Experts, the preview capability aided them in assessing how the MOM will be affected by defined changes. However, participants (P1, P5) believed that the navigation and presentation of changes could be improved to match the actual task of comparing changes. When comparing changes, P1 and P5 wanted to have a high level overview of changed entities and their parents, in the MOM with graphical markers that specify which delta MOM the change belongs to. Furthermore, to quickly find the affected entity and display it in the MOM, a "go to" functionality would further facilitate the task of comparing changes.

Positive Comments

- *"It was easy. It was good to see the change or good to see where I made the change."* —P1
- *"Interesting. It would be useful for reviewing. It is really useful to see in the MOM. How the changes would affect the MOM."* — P2 switch delta MOM and changes are previewed on MOM
- *"When you have these changes there, it is easy to compare."* —P2 about ease of comparing changes in different delta MOMs.
- *"It was intuitive. This one was quite easy to find the changes in the delta MOM."* —P3
- *"The capability of presenting the edits from multiple delta MOMs in the same view and overlaying that in the MOM. It makes it really easy to see and how the result will be and how the changes are."* —P5 about what aspect in MOM workbench made it easy for completing the task

Negative Comments

- *"The task is somewhat difficult since I need to (manually) track what changes made and where."* —P1
- *"I haven't seen the changes over here. I set the delta MOM to current but I don't know what has changed."* —P4

Suggestions

- *"It would be good if I could go to the change and see it on the right tab (preview on MOM)"* — P1 when trying to view how a change would be previewed on the MOM
- *"You could start with collapse everything and highlight where (element) it has been changed. I would say, highlight on top level and drill down."* — P1 about the preferred way of presenting changes to the MOM
- *"Is it possible to see both this (existing delta MOM) and that (newly created delta MOM) at the same time?"* —P3
- *"To compare delta MOMs would be useful. This could happen if different teams are working on the same MOM."* —P3
- *"If people in the different teams are doing contradictory changes. It would be nice to see if their delta MOM changes are affecting my delta MOM changes"* —P3

Scenario 4: Post-task Questions with Responses

1. This task was...

	Very Easy	*Easy*	*Neither Easy nor Difficult*	*Somewhat Difficult*	*Difficult*
Participant	P5	P1, P2, P3	P4		

2. Overall ease or difficulty of comparing changes in different delta MOMs.

	Very Easy	*Easy*	*Neither Easy nor Difficult*	*Somewhat Difficult*	*Difficult*
Participant	P5	P2	P3	P1, P4	

3. This feature of MOM Workbench is helpful to preview changes in different delta MOMs?

	Strongly Agree	*Agree*	*Neutral*	*Disagree*	*Strongly Disagree*
Participant	P1, P5	P2, P3		P4	

Findings for Scenario 5: Comparison between delta MOM projection and current way of viewing changes using Excel spreadsheet

When asked to compare the existing delta document with the Mom Workbench, the majority of participants found the Mom Workbench more intuitive, useful and easier in specifying and viewing changes. The main reasons often stated by participants are related to the capabilities enabled by IDW's projectional editor: projections that allow specifying changes in a notation familiar to the users (which reduces the need for special knowledge of how to use the tool); immediate feedback of changes previewed on the MOM; validation of model while editing.

Comments

- *"The workbench is much easier for everyone to understand. We have a lot of columns here (delta document) which are not directly reflected in the MOM".* —P1
- *"With the workbench it is much easier: you start from what you know and have a rough idea of what you would like to change."* —P1
- *"In the current delta MOM, there are many fields and a lot of information. It is hard to see. But with the MOM Workbench it is easier to see the changes because of the structure and that you can see the changes previewed in the MOM. It will be easier to understand the consequences of the changes."* —P2 when asked whether
- *"That is really good."* — P3 about validation of constraints on the model while editing.
- *"The strength here is that you can see directly what you changes will look like as the final output. In that sense, the tool makes it more convenient for the user."* — P4
- *"If you use these views [projections] then I think it becomes easier for those who are not familiar with the current tool and those details."* —P4

- *"Every aspect made this easier than the delta document. You can't really compare something showing you different layers of transparency while you see the changes merged together and you still see where they come from"* —P5

Findings for Scenario 6: Comparison between MOM projection and current way of viewing MOM using HTML

Compared to the current MOM documentation in HTML-format, participants expressed the need of obtaining an overview of the MOM and its entities in a graphical tree structure. This type of graphical tree structure shows the relationship between the different MOMs and is useful for users to get an overall understanding of a MOM. Currently, the MOM-projection is not accurate enough and lacks functionality to match the HTML-version when it comes to browsing the MOM.

Comments

- *"What I can't see right now which is also not part of our MOM is a diagram, showing the complete MIM fracture. I would like to see the diagram of the MIM fragment and only those MO that are connected to the MOM."* —P1
- *"I think that diagrams are mostly used to get some kind of overview. To get some feel for it. I think that the first way of navigating through the model would be through a diagram then you can click on a fragment to show all those different classes and details"* —P4
- *"The MOM-projection is not as close to the final CPI view that users are used to for it to be good enough."* —P5

Result from Post-Scenario Questionnaires.

Figure 16 illustrates user satisfaction of usability aspects of MOM Workbench according to categories in the UPT (See section 2.3). The X axis shows the distribution of satisfaction level while the Y axis lists UPT categories. From Figure 16, we conclude that the majority of subjects are satisfied with the overall usability: 60% of the participants were satisfied with the visualness. Similarly, approximately 60% of the participants were satisfied with how their tasks were mapped to the MOM Workbench although more functionality was asked for. As previously mentioned in the findings from scenarios above, dissatisfaction was mostly related to issues with fine tuning the appearance and interactions of user interface objects and implement additional features for presentation of information and model manipulation.

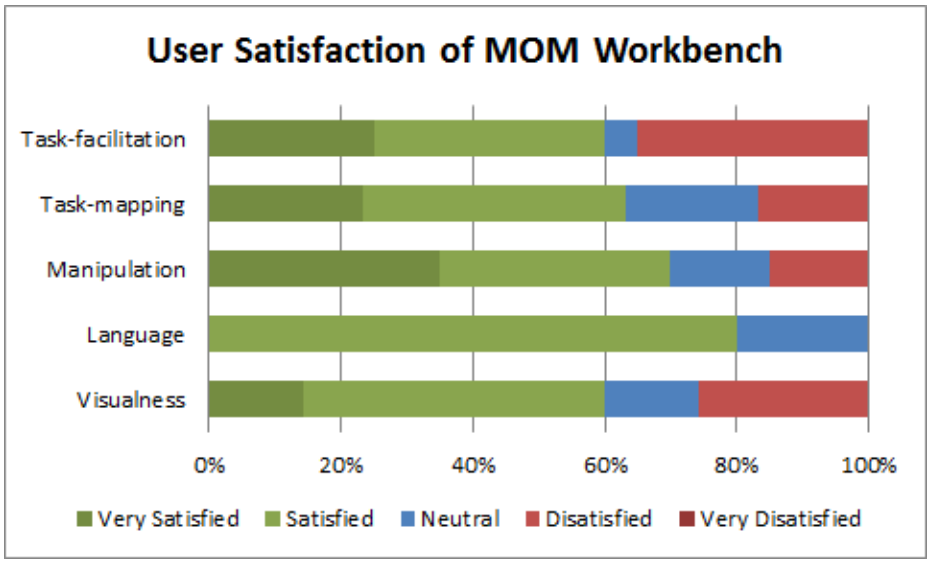

Fig. 16. Shows user satisfaction of different usability aspects of the MOM Workbench based on the result from usability survey

Collated Findings.
Table 8 lists the usability problems, their severity(as defined below), their origin on the usability and the number of participants who experienced each problem.

Severity

Usability problems encountered by the participants were categorized using the following severity levels:

- Level 1: Problems which caused difficulties for the participant to complete the task.
- Level 2: Problems which caused frustration and delay of task.
- Level 3: Problems that could be confusing at first but not to such a degree that it considerably delayed the completion of the task.

Origin

The origin of a usability problem defines whether the problem is present specifically to the MOM workbench (local origins) or in general for the Intentional Domain Workbench (global origins).

1st Category and 2nd Category

The usability problems were categorized using the Usability Problem Taxonomy described in section 2.3. Note that some usability problems could be distributed to different categories (1st category and 2nd category are different), while some usabil-

ity problem could be distributed to one category (1st category and 2nd category are same).

Number of Participants

The number of participants that experienced the problem.

Table 8. Usability problems categorized according to UPT with additional severity rating, origin and number of affected partipants

	Usability Problem	**1st Category**	**2nd Category**	**Origin**	**Severity**	**Number of Partici-pant(N=5)**
UP1	Users are confused with the meaning of the "cog"-icon.	Object appearance	Object appearance	Local	1	4
UP2	Users want to right-click with mouse button anywhere in a tab to access popup menu with relevant commands	Direct manipulation	Interaction	Global	1	5
UP3	Users could not see all content of tabs due to missing horizontal scrollbars.	Object appearance	Object appearance	Global	1	5
UP4	Dragging the vertical scrollbar of a tab do not properly scroll the content of the tab.	Navigation	Navigation	Global	2	5
UP5	Users overwhelmed by an excessive amount of unrelated menu options on the menu bar	Object(screen) Layout	Interaction	Local	1	5
UP6	Users did not notice when focus shifted to a the element selected from the Table of Contents projection.	Navigation	Keeping the user task on track	Global	1	4

UP7	Users did not notice messages displayed on the status bar on the left-bottom corner.	Object appearance	Object appearance	Global	1	4
UP8	Users want to be able to display tabs in separate detached windows	Object(screen) Layout	Interaction	Global	2	2
UP9	Users find the close button on tab not obvious.	Object appearance	Object appearance	Global	3	4
UP10	Users want to view changes in a delta MOM on a higher level by having color coding of the element and its ancestors in tree view of the affected MOM	Presentation of information	Presentation of information	Local	1	4
UP11	Users want to display a single delta MOM.	Presentation of information	Presentation of information	Local	1	2
UP12	Users want the Table of Contents to be updated(including color coding) to reflect the changes made in the current delta MOM	Presentation of information	Functionality	Global	1	2
UP13	Users want to make changes to classes by interacting with a graphical diagram similar to a class diagram.	Functionality	Functionality	Global	3	2
UP14	Inconsistent usage of cog icon. The cog	Object appearance	Object appearance	Local	1	1

	icon is used as icon for an action as well as a button for displaying popup menus.					
UP15	Users want to have different tabs in the editing basket showing the scope of elements that users are interested in selecting	Object(screen) Layout	Interaction	Local	3	1
UP16	Making changes directly by editing the MOM-projection	Interaction	Interaction	Global	1	5
UP17	User wants to copy a plain text to a field for reference	Interaction	Direct manipulation	Global	1	1
UP18	Users want to preview multiple delta MOMs at the same time	Presentation of information	Functionality	Local	2	2
UP19	Users want to merge different versions of MOM	Functionality	Functionality	Local	3	1
UP20	Users do not notice that newly created changes in delta MOM-projection are previewed on MOM in the MOM-projection	Keeping the user task on track	Keeping the user task on track	Local	1	3
UP21	Users do not notice that newly created changes in MOM-projection are displayed in delta MOM-projection.	Keeping the user task on track	Keeping the user task on track	Local	1	3
UP22	Users wants to dis-	Presentation of	Functionality	Local	3	1

	play and navigate to conflicts caused by changes in delta MOMs	information				
UP23	Graphical tree structured view of MOMs.	Presentation of information	Presentation of information	Local	2	2
UP24	Overview of MOM overlaid with changes marking which nodes are changed in a delta MOM	Functionality	Functionality	Local	2	2
UP25	Status of delta MOM not changed when committed.	Functionality	Functionality	Local	3	1
UP26	Input field for exceptions is not clear in the MOM-projection	Visual cues	Visual cues	Local	3	1
UP27	Users find input field for values in delta MOM-projection confusing at first.	Visual cues	Visual cues	Local	3	2
UP28	Users think that selection of table cells in the delta MOM projection is unpredictable.	Direct manipulation	Interaction	Global	2	1
UP29	Users want to crown select entity (attribute, action, enum member and struct member) but do not know what to click.	Visual cues	Interaction	Global	1	2
UP30	Users want to export delta MOM.	Functionality	Functionality	Local	3	1
UP31	When adding a	Keeping the	Navigation	Local	1	4

	change and creating a delta MOM users confused if anything happened.	user task on track				
UP32	Drag and drop.	Direct manipulation	Interaction	Global	3	2
UP33	When creating a new delta MOM the sections for Add-, Modify- and Delete Changes should be displayed. Users may then add changes by right-clicking on a category to add a change.	Presentation of information	Keeping the user task on track	Local	3	1
UP34	User confused about the meaning when mouse cursor changes to a “red cross”.	Visual cues	Visual cues	Global	3	1
UP35	Input box not showing last input character	Object appearance	Object appearance	Global and Local	1	5

Problem Categorization

As shown in Figure 17 which illustrates the distribution of usability problems, 38.57% of the problems are from task-mapping, 35.71% from visualness, 15.71% from manipulation, and 10% from task facilitation. No problems are categorized in the language category which shows that the terminology used in the Mom Workbench is consistent with the terminology in the MOM domain.

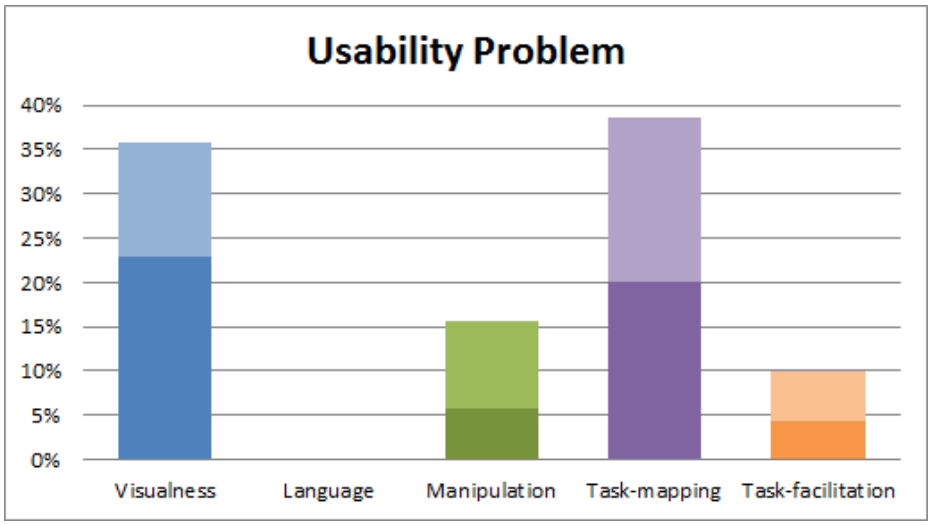

Fig. 17. The distribution of usability problems. The light shades in each bar represent the global problems in the category, while the bars in deep color show local problems

Visualness

Figure 18 shows the distribution of problems over the subcategories in visualness. The problem category with the largest percentage is object appearance, which attributes to 48%. Apart from object appearance, presentation of information constitutes to 40% of the visualness problems, followed by object layout with 12%. There are no usability problems categorized in object movement and non-message feedback categories.

Among identified problem in object appearance, 58.33% constitutes to global problems. The problems are related to limitations in the implementation of user interface objects of IDW, examples of such issues are manipulation of scrollbars (UP3) and issues with unclear appearance of the close button in tabs (UP9). Among the object appearance problem, 33% can be handled locally in the MOM Workbench such as using icons that are familiar to users (UP1, UP14).

In addition, 90% of the problems in presentation of information category are related to the MOM Workbench since these problems mainly reflect domain concerns of MOM. For instance, in their current way of working, users view MOMs through a graphical tree view. Users want the Mom Workbench to provide a similar graphical tree structured view of MOMs (UP23).

Among object layout problems, 66.67% are related to the implementation of MOM workbench (UP 5 and UP15) and can be addressed by improving the MOM domain projections. Other object layout problems (33.33%) are global issues related to limitations in IDW (UP8).

There is no problem categorized as object movement problems or non-message feedback since behaviors in these categories are not supported in the IDW and thus not present in the MOM Workbench.

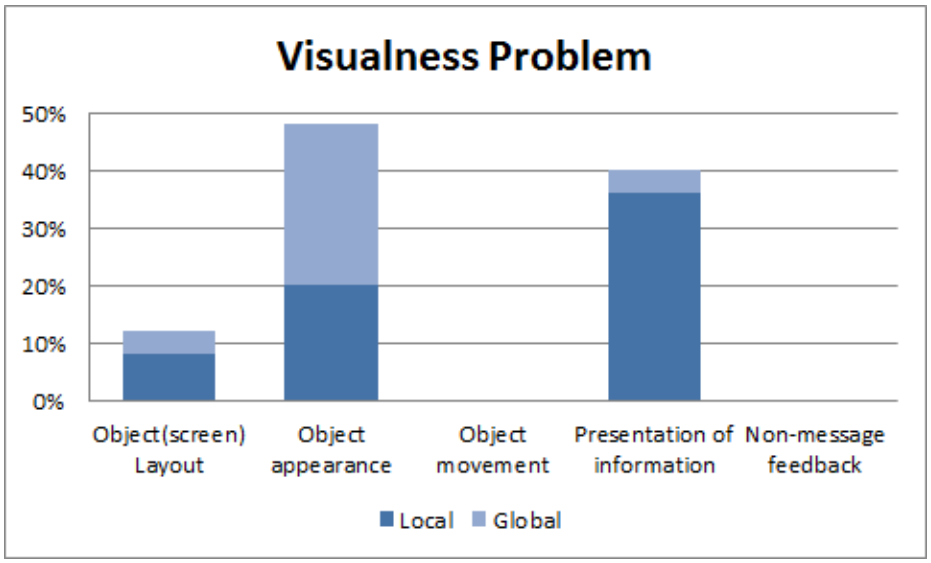

Fig. 18. The distribution of visualness problems. The bars in light color present global problems in the category, while the bars in deep color show local problems.

Manipulation

Figure 19 shows the distribution of identified usability problems related to manipulation. Among problems in this category, 63.64% are visual cues problems, which constitute the majority of the manipulation subcategories. The remaining is direct manipulation problems (36.36%). There is no usability problems related to physical manipulation.

More than half (57.13%) of the visual cues problems are local to MOM Workbench. Local visual cues problems are mainly about issues with locating and recognizing input fields (UP26, UP27). Global visual cues problems include missing visual cues for selections (UP29) and mouse behaviors (UP34). All reported usability problems in direct manipulation subcategory (UP2, UP28, and UP32) are global due to limitations in the IDW. UP2 is related to limitation of how and where to invoke context menu. UP28 is related to issues with interacting with tree based documents in IDW (see 4.2.3) and finally UP32 is related to missing feature in IDW to support drag and drop.

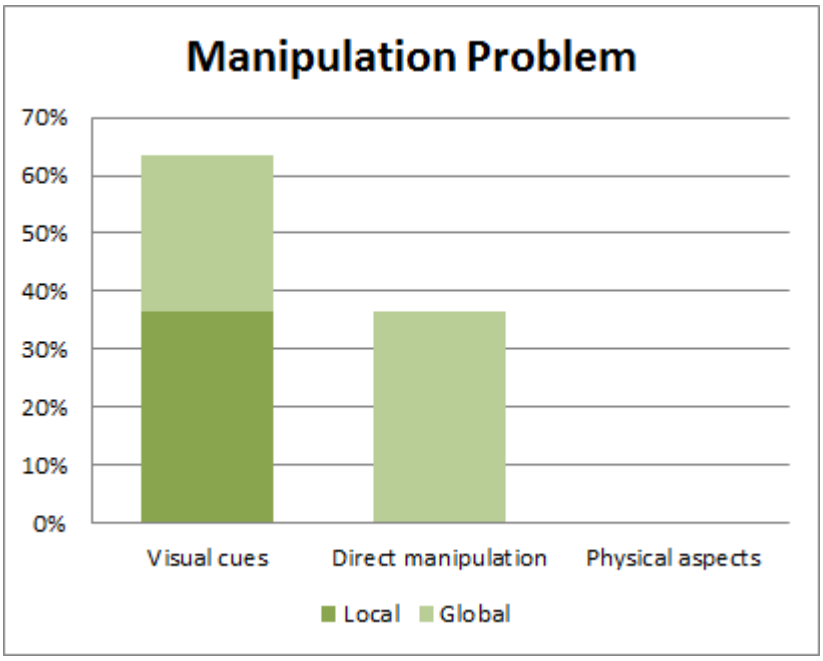

Fig. 19. The distribution of manipulation problems. The bars in light color present global problems in the category, while the bars in deep color show local problems

Task-mapping

Figure 20 shows the distribution of task-mapping problems among the subcategories: interaction, navigation and functionality. The majority of task-mapping issues are related to functionality which constitutes 48% of the problems. The functionality category consists of mainly user requests for additional features of the Mom Workbench to support user tasks. For example, Feature Developers are requesting a graphical notation which appearance should be similar to UML class diagram to specify changes (UP13). Model Developers want to be able to merge different versions of a MOM (UP19). These features requests are primarily local problems that are specific to the Mom Workbench. The high amount of functionality problems indicates a more thorough study is needed to understand how users in the RBS MOM process perform their task of defining and viewing changes.

Concerning global functionality problems, an issue with updating the Table of Contents to reflect changes made to a MOM (UP13) is due to limitations of the IDW.

In the interaction subcategory, 80% of the problems are issues related to the IDW. These problems are primarily related to users' expectations of the interaction with user interface objects do not correspond to the actual system behavior. For example, selections of elements in a tree structured document in IDW will always result in selection of a node in the document. When a user selects a white space area in the document, IDW will instead select a node which in some cases will result with the page scrolling down due to shifting focus to the selected node. This behavior is con-

trary to the user's expectation of how selection should work according to their previous experience when working with text-based documents (UP28).

Problems with navigation are global concerns related to issues in IDW with navigation in tabs (UP4) and navigating (shifting focus) to elements (UP31).

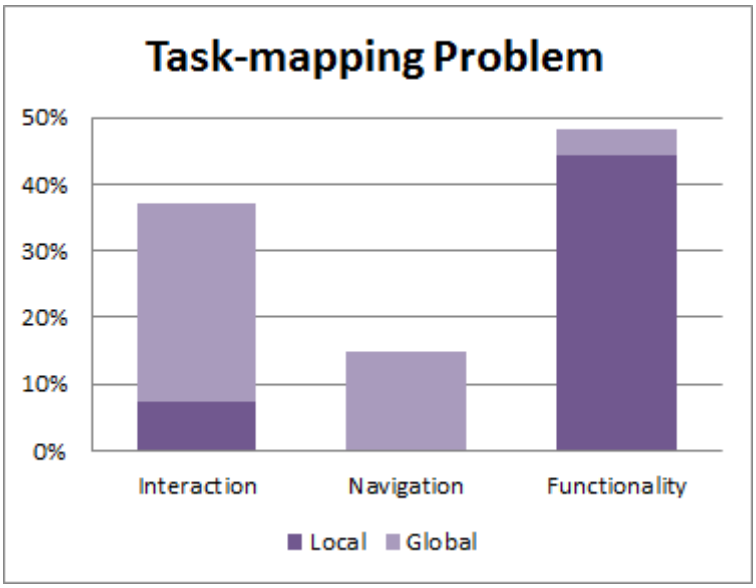

Fig. 20. shows the distribution of task-mapping problems. The bars in light color present global problems in the category, while the bars in deep color show local problems.

Task-facilitation

The distribution of the problems categorized as task-facilitation is shown in Figure 21. The categories Alternative, User Action Reversal and Task/function Automation are not applicable in the Mom Workbench. All task-facilitation problems (UP20, UP21, and UP31) are located in the subcategory, Keeping the user task on track.

Problems UP20 and UP21 are both related to the user losing track in their task when Mom Workbench did not provide adequate feedback of the resulting output.

UP20 and UP21 are considered as both local and global problems since a feedback message system is provided in IDW but not adequate for users to notice it (UP7). UP31 is in addition related to navigation since the Mom Workbench should directly move/shift focus to the created element.

Severity of Local and Global Problems

Figure 22 illustrates the how problems with different severity ratings are distributed among local and global problems. As shown in the figure, the majority of the problems with severity S1 are global problems due to limitations with the IDW described in section 4.2. Problems with severity S2 have approximately an even distribution of

both local and global problems. Most of the problems with severity S3 are local to the Mom Workbench.

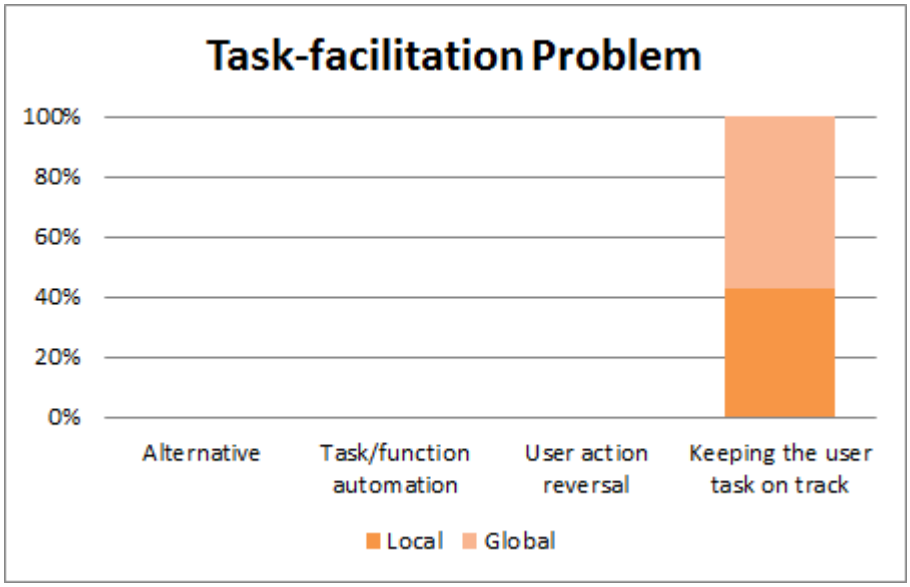

Fig. 21. The distribution of task-facilitation problems

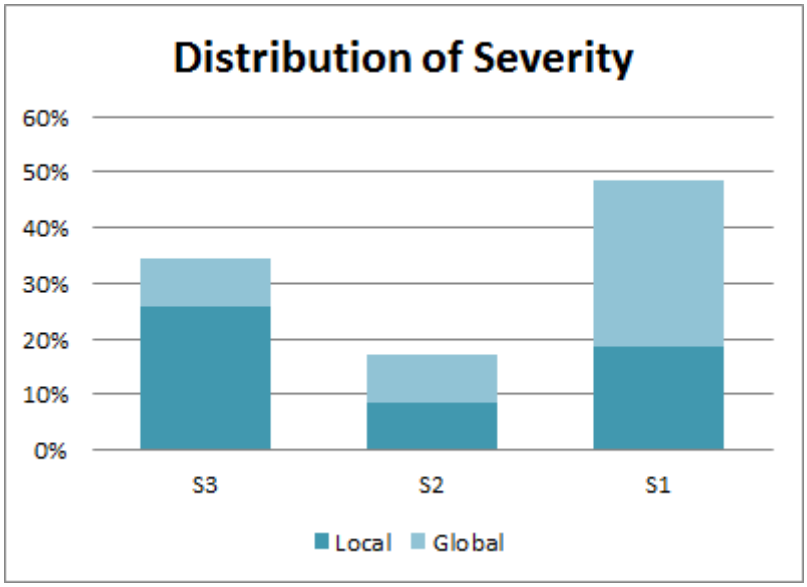

Fig. 22. Severity ratings of local and global problems

Severity Distribution of Local Problems

The severity of the local usability problems are shown in Figure 23. As shown in the figure, the problems with the highest severity ratings, S1 and S2, are primarily located in the visualness category. These problems are related to feature requests of how information should be presented that will further ease the users' work of viewing and comparing changes defined in delta MOMs. The problems can be resolved by providing the requested features in the implementation of the IDW. In the task-mapping category, the majority of problems have severity rating S3 which shows that although the problems do not correspond to their expectations, the users can still tolerate the issues and perform their given tasks. The large amount of S1 problems in task-mapping are mainly from functionality (see figure 20) which is expected due to the Mom Workbench is still in its the early phases of development.

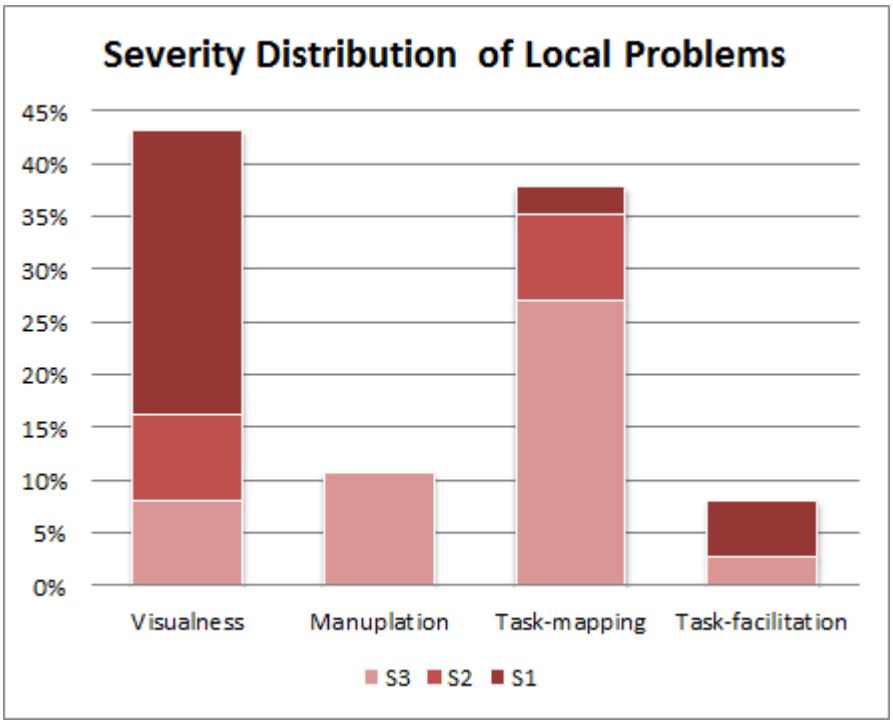

Fig. 23. Local usability problems with severity ratings

Severity Distribution of Global Problems

Figure 24 illustrates the distribution of global usability problems annotated with severity ratings. The majority of global problems with high severity primarily belong to task-mapping, visualness and task-facilitation. The large amount of severe problems located in task-mapping is critical problems related to defining changes (UP16, UP17) and navigation (UP6). Problems with severity S2 in the same category are concerned with navigation (UP4) and interaction (UP 8, UP28). In addition UP8 is

dealing with Object Layout while UP28 is dealing with direct manipulation. The problems can be resolved by improving current interaction features. The majority of problems have severity rating S3 lay in direct manipulation (UP32) and visual cues(UP34), which shows that unfamiliar manipulation and visualness in IDW may confuse users but it would not discourage user from their tasks.

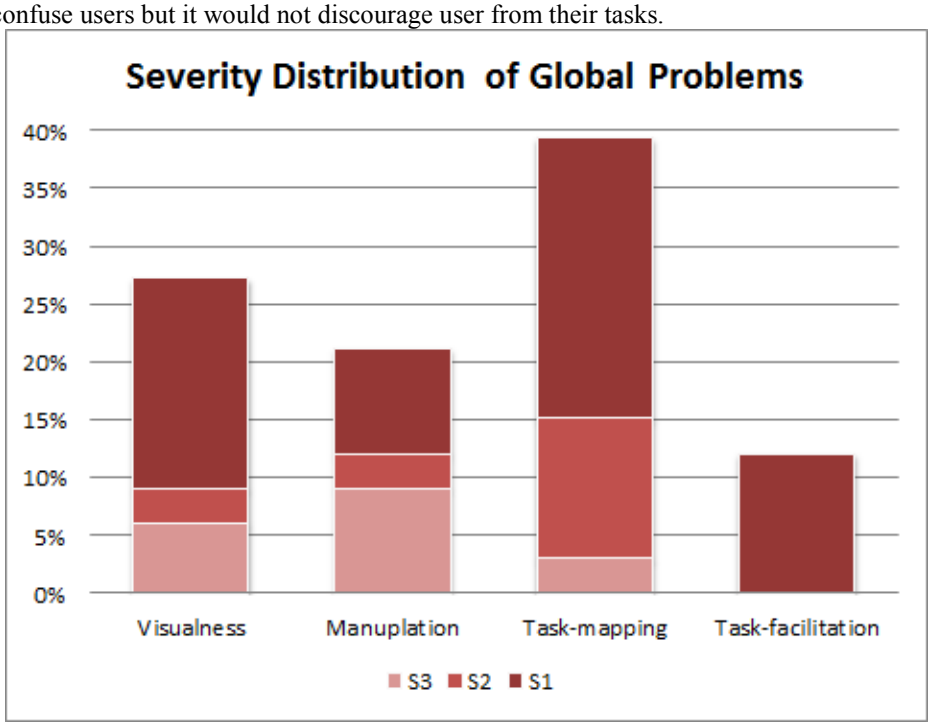

Fig. 24. Global usability problems with severity ratings

5 Discussion

5.1 Process Quality Improvements using Language Workbench Technology

The result of this study shows that process qualities of the current interface development process have improved by using language workbench technology.

The identified inhibitors on the current process are inhibiting the speed for which the users are performing their tasks and the quality of the resulting output artifacts. The inhibiting effects are primarily caused by the semantic gap between the delta document and the interface model i.e. different constructs expressed in different representation systems. By only using constructs in one representation system expressed in different projections, the need for separate delta documents is eliminated and thus the semantic gap is closed.

As a result, there are several process quality improvements with respect to speed and artifact quality. Furthermore, improvements in the support for the user roles in the process have been observed. First, the need for manually translating the changes in

delta documents is replaced by automatic transformations which merge the delta documents with the interface models in the MOM Workbench. Eliminating the step with manual translation increases the end-to-end speed of the development process. Second, communication and understanding among user roles are increased due to tailored projections. In the usability test several users of different roles of the current process stated that having different but consistent views of the interface model would allow them to "make discussion easier" and "understand the consequences of the changes" defined in the delta documents.

Third, modern IDE features in projectional editors combined with tailored projections eases the tasks of viewing, defining, and comparing changes to the interface models. The majority of the users found it both easier and more useful to work with the MOM Workbench compared to the current delta documents and tooling environment. Continues validation ensures those delta documents which do not fulfill specified design rules in the domain will be passed on to the next stage of the development process. As a result, the possibilities of introducing common errors caused by mistakes and logical errors are captured in the early phases of the interface development process.

However, uncertainties in study's findings cannot be disregarded without actual deployment of the MOM Workbench in a real life setting. In such a case, process qualities may initially decrease due to unfamiliarity of the MOM Workbench but later to increase due to the ease of learning and using the workbench. The ease of use was shown in the usability tests, learning the DSLs, navigation, interaction and presentation of information required minimal training.

As shown in previous research of the benefits of DSLs (see 6), we believe that an actual deployment will improve productivity for the roles in the process. Overall, the benefits from improved process qualities, perception, communication and understanding of the MOM Workbench, outweigh the approach and tooling used in the current process.

5.2 Comparison of Development Effort between IDW and current MDE tooling

The result of the estimation on development effort between IDW and the current MDE tooling shows a considerable decrease in effort using IDW. The effort for realizing the domain, validation rules and introducing transformation is roughly the same for both approaches. This is due to the already mature support for specifying meta-models, validation (e.g. OCL, VF) model transformations (e.g. ATL, QVT) in EMF. The effort is instead in the design of the meta-models and additional constructs to support the visualization and specification of changes to an interface model. This is shown in the difference of effort it would take for realizing projections for the roles in the interface development process. In an EMF-based approach the construction of concrete syntax is mainly divided into plugins which support textual syntax (Xtext, TCS) or graphical syntax (GMF, Graphiti, GMP). In order to provide the support of both graphical and textual syntax, considerable effort is required to extend the plugins to either support both forms of notation or make the additional plugins interoperable.

Compared with IDW which supports interoperable DSLs for both textual and graphical syntax, no additional effort is required. IDW provides a set of graphical constructs which support common constructs found in typical word processors such as tables, headers, lines and boxes. In this aspect, IDW offers a more flexible and faster approach to construct domain-specific editors which match the presentation and notation to domain users than solutions based on EMF. However, the question rises concerning the limitations of IDW's capabilities of constructing projections. In other domains which require more advanced graphical constructs involving 3D-graphics and animation effects such as zooming, would require development of new DSLs which integrate to a target graphics engine. The initial effort of such an implementation would be equal to an EMF-based approach but once implemented the DSLs are reusable and interoperable with other DSLs, therefore subsequent adaptation to other domains is minimal or unnecessary.

5.3 Experience of Using the Intentional Domain Workbench

From our experiences of using the Intentional Domain Workbench for the development of the MOM Workbench, we believe that IDW are advantageous in several situations compared to an approach using Eclipse Modeling Framework (EMF).

IDW is preferable when there is a need to mix multiple domains. A model in IDW can contain domain code from different DSLs while an instance model in an EMF-based approach can only contain data from a single DSL. In our case, a MOM Workbench document can mix domain code using different DSLs such as interface model, delta documents and requirements. This integration of DSLs is supported by IDW, so no additional glue code is required. However, to obtain a similar ability in an EMF-based approach would require considerable development effort to extend the meta-model and dependent constructs such as transformation and constraints.

In the case of IDW, less effort is required when it comes to changing a meta-model. Since documents in a knowledge workbench are tree based documents, entities in the document refer to each other by identity rather than text. Once a change is made in one entity in the domain schema, the reference of the entity is retained and synchronized with the domain code. Thus, the consistency in the domain schema with documents using the concepts in the domain schema is preserved. For example, in IDW, when we update a name of an entity in the domain schema, other parts of the program which refers to this name will also be updated to reflect the change. However in an EMF-based approach, if we make a corresponding change in the meta-model, we need to refine other dependent artifacts in EMF such as validation rules and transformations to support the change we made in the meta-model. Furthermore, additional effort is required to make the new instance model and old instance model consistent.

Compared to EMF-based editors, projections in IDW are more flexible due to the possibility to combine graphical, textual and symbolic notations in a single editable view. Thus projections could be tailored for domain users which would reduce the training costs for using such a tool. As discussed above, the possibility to mix DSLs, the ease of making changes and flexible projections are features which we think make IDW a powerful tool when it comes to rapid prototyping in order to explore unknown

problem domains and re-engineer existing MDE based software development processes. Changing an existing process by introducing a new role and activities could be done in a short time by defining a tailored projection. Depending on the information that the role requires, minimal or no changes are required to the domain schema.

5.4 Threats to Validity

This study is subject to several threats of validity. This section will discuss the about external validity, internal validity, reliability and construct validity.

External Validity

Subjects engaged both in study are not representative to a larger population outside of the studied context at Ericsson AB. However, interfaces of the kind studied in the thesis are very common in large scale embedded software development. For example in automotive, aerospace, industrial automation and other interface intensive domains. Therefore the comparison may not be widely generalizable but still replicable.

Construct Validity

The implementation is done by two research students with basic experience of using EMF in student projects. The data confidence of comparison could be improved if the implementation is done by professional developers with background in software interface development using EMF.

Internal Validity

Threats that will challenge the internal validity of the study are the estimations made by tool developers with experiences from a different sub-domain of software interface modeling. Although these developers are responsible for the same domain of software interface development, minor differences in their tasks and context may affect their estimation. To decrease the human and situational bias, guidelines for expert estimations were followed (see section 3.5). However, the validity of the comparison will increase if the data is based on actual implementation of a prototype using an EMF-based approach.

Concerning data collection from stakeholder meetings and semi-structured interviews, interviewer bias may have affected the subjects.

In the usability testing, the number of participants may affect the validity of the findings. Although the coverage of usability problems increases with the number of participants, previous studies [23] point out that five users in one session are sufficient to cover approximately 80 percent of all usability problems. In order to further improve the validity of the findings, it is recommended to use an iterative approach with several usability test iterations.

Content Validity

Content validity is related to the design of the test scenarios in the usability test. To ensure the validity of the test scenarios, a pretest was made with a domain expert with

extensive knowledge about the studied interface development process. Feedback from the domain expert was incorporated in the design of the usability test.

5.5 Recommendations on the basis of the usability study

In this section, recommendations are presented to improve the usability issues identified in this study. Recommendations are given to both tool developers of the MOM Workbench and the tool vendor.

Recommendations to the tool vendor.

Based on the usability test result (see section 4.7), recommendations are given in the order of severity and frequency. The prioritization of problems should be done in the following order: task-mapping, visualness, task-facilitation and manipulation. Below follows a listing with recommendation of resolving the identified problems.

1. Visualness: Window system with detachable tabs (UP7, UP8, UP31)

The presence of a window system with detachable tabs is a feature that is common in most development tools. By providing this feature, users can customize the layout of the tabs for which the user find suitable and satisfactory. In the MOM Workbench, a user may display two tabs on separate screens whereas one screen displaying a tab with several delta MOMs side by side; and the other screen showing the changes preview on one or several MOMs. This feature would apart from resolving UP8 also provide values which increase the perception for the user. Such a window system also enables the possibility of defining groups of tabs displaying for instance system related messages, validation results and similar content. Users would then know that system messages will appear in one of the tabs in the a certain grouping, in contrast to how it is currently implemented where messages are displayed in the left corner of the user interface which users found unclear (UP7, UP31).

2. Visualness and Manipulation: Visual Cues (UP26, UP27, UP29)

Visual cues are graphical indicators specifying whether a user interface object can be manipulated or not. By providing visual cues, indicators can be placed which will change appearance upon mouse-hover. For instance, in the case of the MOM Workbench, when a user may put the mouse over an element type which will then change the icon of the mouse cursor or the area of the element will be highlighted, to show that it can be crown selected (UP29). Similarly, the input field of a value would popup a message with description of how to specify a change according to MOM design rules (UP26, UP27).

3. Task-mapping: Multiple paste options to resolve ambiguity (UP17)

When the user is copying a node, IDW should be able to support different paste choices, such as paste as a sub tree, a node, a reference, or plain text. In the MOM Workbench, a user would be able to copy a text value of a name to an existing ele-

ment and then select to paste it as a reference in a field accepting references to elements (UP17).

4. Manipulation: Right-click on whitespace area to invoke context menu (UP2)

Currently, right-clicking on a white space area of a tab will select the closest node and display the context menu for the selected element. As described in the findings, providing the possibility of right-clicking on a white space area with customizable context menu would correspond more correctly to the user's mental model of where to access relevant commands (UP2).

5. Task-mapping: Resolve issue with multiple editable projections of a single element (UP16)

Known issues with IDW editing multiple projections cause severe problems for users in the MOM Workbench (UP16). Currently in the Mom Workbench, when a user is editing a new change in the MOM-projection, there are severe interaction issues which hinder the user from completing the task. Resolving these issues should take priority in order to increase the usefulness of the Mom Workbench.

6. Task-mapping: Support drag and drop (UP32)

For the users in the usability study with experience of interacting with graphical user interfaces, the feature of drag and drop is part of their mental model of such an interface should work. It is recommended that the drag and drop feature should be supported to further ease the work for the users (UP32).

7. Task-facilitation: Resolve issue with focus (UP20, UP21)

When a user follows a reference or selects an element in the TOC, IDW will shift focus to the selected element by scrolling the content of the tab to the element and then crown select it. In cases where the element is visible in the bottom of a tab, scrolling will not happen, instead the element is crown selected. In most of these cases, users will not notice that they have selected anything (UP20, UP21). To keep the user on track on their task and thus improve task-facilitation, it is recommended to fix this issue so the focus is centered on screen when an element is selected.

8. Visualness: Support for Horizontal Scrollbar (UP3) and Resolve Bugs (UP9, UP12, UP35)

Global usability problems related to the appearance of user interface objects such as issues with the input box (UP35) and close button on tab bars (UP9) are bugs in the IDW. Furthermore, by providing a horizontal scrollbar, resolves the issues with users not seeing the entire document and navigate horizontally (UP3). In the Mom Workbench, there is currently an issue with the Table of Content not displaying elements that have been added through add changes in a delta MOM. The problem is due to issues with the IDW (UP12). The above mentioned problems are considered as

smaller issues and bugs, if resolved would significantly improve usability with respect to visualness i the Mom Workbench.

Recommendations to MOM Workbench developers.
Based on the findings in section 4.7, recommendations are given in the order of severity and frequency. The prioritization of problems should be done in the following order: visualness, task-mapping, task-facilitation and manipulation. Below follows a listing with recommendation of resolving the identified problems.

1. Visualness: Standardized set of icons and design consistency

   The icons used for the user interface object in the Mom Workbench should be redesigned to have a consist design indicating the type of object and thus if it can be interacted. Icons for buttons and element type should be designed in a way that the user will notice the difference. For instance, in the Mom Workbench, the Operations Button is illustrated with a cog-wheel which in its current design can be interpreted as any kind of icon, not as a button.

2. Task-mapping: Functionality for Model Developers

   A mapping of the task performed by the Model Developers should be performed in order to refine the projection for the role. As mentioned in the findings, functionality for merging models, displaying conflicting change requests and export and import delta MOMs are some features that should be supported in order to improve usability for Model Developers.

3. Visualness: Context Menu through right-clicking on model elements

   In users' mental model, it is given that right-clicking on an element will display a context menu with commands for that specific element. Implementing this feature would better match the mental model of users and thus improve usability for the Mom Workbench.

4. Visualness and Manipulation: Selection and Displaying Elements in the Table of Contents and MOM-projection adopt behavior of current CPI HTML

   In the studied process, users mainly navigate and access the MOM through a website representing the MOM. By adopting some of the behavior of the web site, especially selection and navigation, users will find it less unfamiliar and easier to learn how to use the Mom Workbench. Currently, it is possible to right-click on separate parts of a line in the TOC and in the MOM- and delta MOM-projection left clicking can be done on the element type and the name of the element.

   One of the features that will be helpful is that elements, when selected will display the whole line as a crown selection. The element can then only be left-clicked and right-clicked on the entire line. This can then be used to exhibit distinct behavior when performing mouse clicks. For example, whenever the user single-click with the left mouse button on an element, the whole line is selected, in the TOC, the element

will then be displayed on a fixed tab (related to feature 1 window system with detachable tabs described in section 5.5). The user may also be able to right-click the whole line to access the context menu and display the element to a separate tab. Another example is when the user clicks on the text value or the element type in the MOM- and delta MOM-projection, the whole line will be selected instead of placing the cursor to edit the text. To change the name the user can right-click and access the corresponding menu item or the user can left-click twice with a delay between each click to place the cursor to change the name. Users are familiar with this type of selection which they know from Windows operating systems.

5. Visualness: Presentation of Changes in MOM-Projection

In the Mom Workbench, it is currently difficult to navigate to changed elements of the MOM. The MOMI-projection shows the entire MOM with previewed changes which in larger models would be not feasible to display. One way to improve the display of changes is to present the entire MOM with all children collapsed and highlight with color coding those elements and their ancestors that are affected by the changes. The user can then expand the tree structure of the MOM and go to each change that interests the user. In addition, the color coding should also be presented in the TOC.

6. Visualness and Task-mapping: Present changes to MOM from Multiple delta Moms

A feature that would increase the usefulness of the Mom Workbench is the possibility to preview changes of multiple delta MOMs. This feature would further facilitate the tasks of feature developers, domain experts and modeling experts to compare and assess changes in delta MOMs.

7. Task-mapping: Graphical diagram showing an overview of the MOM

In the current CPI HTML, graphical diagrams are displayed to illustrate MOMs. These static diagrams are used to obtain an overview of a MOM and its relationship to other MOMs. The Mom Workbench should provide similar but dynamic diagrams with additional functionality. For example automatic update of diagram due to changes made by users, color coding to elements to show changed elements, specifying changes to diagram elements and build a custom view by drag and drop elements on a diagram where the changes made to elements will also change the underlying model.

6 Related Work

To our knowledge there are no published studies on software process quality improvements using language workbench technology in an industrial context. Several studies exist on the concept of language-oriented programming describing possible benefits and disadvantages. Ward [12] established the concept of language-oriented programming and how it was designed to enable rapid-prototyping and handle chal-

lenges in large-scale software systems such as complexity, change and conformity. Fowler [11] coined the term and characteristics of language workbenches which implement the concept of language-oriented programming. End-user programmability and ease of constructing interoperable DSLs are mentioned as benefits. Voelter et. al. further extended the characteristics [24][25] and compared the ease of extending and composing domain-specific languages for embedded systems with a code-centric approach [26]. The results in Voelter's study indicate significant improvements in development effort. However, the study is based on an example with limited scope. Simonyi et.al [15] introduced Intentional Software, a language workbench evolving the ideas of Intentional Programming [27]. An evaluation of the maturity of language workbenches was conducted by Stoffel [28]. Stoffel listed issues of language workbenches involving integration of language workbenches with existing tool chains, refactoring DSLs, support for debugging and unit-testing.

The benefits of DSLs are a well-known area of the subject. Kärnä et. al. [29] evaluated the use of DSLs in industrial context, which showed improvement in productivity, usability, quality and error prevention compared to a non-DSL approach. Further studies in DSLs using graphical notation, domain-specific modeling languages, such as Caprio [30], Tolvanen et. al. [31] [32] and textual notation such as Hermans et. al. [33] confirms the findings to a varying degree.

The high costs of constructing DSLs have been covered by several studies. Mernik et. al. [34] identified problems in current language systems to support the creation of DSLs and concludes that process of creating DSLs is still complex and costly. A similar conclusion was made by Wu et. al. [35] stated that although maintainability of DSLs is improved using DSLs tools, the development of DSLs is still complex.

Usability of language workbenches based on projectional editing is an issue recognized by Voelter et. al. [13]. To our knowledge there are no published studies on the area of usability.

7 Conclusion

MDE has shown to increase productivity. However the high cost of implementation, maintenance as well as training of modeling tools have been hurdles from widespread adoption of MDE. In this study, we investigated the influences on software process quality (end-to-end speed, development effort, error prevention) for which the latest generation of MDE technology, language workbenches, has on MDE-based software interface definition processes in the context of large-scale embedded systems. This study was conducted as a single-case case study at Ericsson AB where we identified inhibitor of speed and quality in a certain interface definition process, implemented a proof of concept using Intentional Domain Workbench to address the identified inhibitors, re-engineered the interface definition process to support the proof of concept, conducted a usability study for evaluation and compared the development effort using 2nd-generation modeling tool based on estimations.

Our results show that language workbench technology has positive impact on several aspects compared to the current tooling environment:

- The speed in development of domain specific tooling, increased significantly due to its flexible projections, agility to change and mixing DSLs. These benefits of language workbenches facilitate rapid software development process re-engineering.
- The end-to-end speed for defining interface definitions improved due to tailored projections and the introduction of automation which eliminates manual tasks in the process. Feature developers get faster turnaround for requested changes and model developers get fewer intermediate steps. Product owners get increased end-to-end speed and information quality in the development of new product features.
- Improved communication, understanding and perception among the different roles in the process due to flexible projections can be tailored for different needs.
- Usability of projectional editing is considered satisfactory for the majority of users.

Furthermore, for modeling researchers, this study is an empirically example on the benefits of a multiple viewpoint based MDE solution compared to a classic transformation based solution.

Further studies are necessary to strengthen our conclusions such as formal experiments involving actual deployment.

Reference.

1. Simulink, http://www.mathworks.se/products/simulink/

2. Bran, Selic, Using UML for Modeling Complex Real-Time Systems, Languages, *Compilers, and Tools for Embedded Systems*, LNCS 1474 (1998)
3. Rational Rhapsody Developer, http://www-142.ibm.com/software/products/us/en/ratirhap
4. Eclipse Modeling Framework Project (EMF), http://www.eclipse.org/modeling/emf/
5. Cook, J., Kent, W, Domain specific development with visual studio DSL tools, Addison Wesley, (2007)
6. MetaEdit, http://www.metacase.com/products.html
7. Dmitriev, S, Language oriented programming: The next programming paradigm. *JetBrains onBoard,* 1(2). (2004)
8. Intentional Software: Technology, http://www.intentsoft.com/intentional-technology/
9. ObjectStore; Progress Software Corporation, Object Data Management for Network Management Systems, (2003)
10. Breugst, M., Marino, G., Chatzipapadopoulos, F., Choy, S., De Zen, G., Faglia, L., Magedanz, T. Object Oriented Software Technologies in Telecommunications: From theory to practice. Chichester, West Sussex, England: John Wiley & Sons Ltd. (2000).
11. Fowler, M. Domain Specific Languages, Addison-Wesley Professional. (2010).
12. Ward, M. P. Language-oriented programming. *Software - Concepts and Tools*, pp. 147-161. (1994).
13. Voelter, M., Benz, S., Dietrich, C., Engelmann, B., Helander, M., Kats, L., Visser, E., Wachsmuth, G.: DSL Engineering - Designing, Implementing and Using Domain-Specific Languages. *CreateSpace Independent Publishing Platform*, (2013).
14. Brambilla, M., Cabot, J., & Wimmer, M. Model-Driven Software Engineering in Practice. Morgan & Claypool Publishers. (2012).
15. Simonyi, C., Christerson, M., Clifford, S.: Intentional Software, 451–463. (2006)
16. Keenan, S. L., Hartson, H. R., Kafura, D. G., & Schulman, R. S. The usability problem taxonomy: A framework for classification and analysis. *Empirical Software Engineering*, 4(1), 71-104. (1999).
17. Keenan, S. L. Product Usability and Process Improvement Based on Usability Problem Classification. Ph.D. Dissertation. Department of Computer Science. Virginia Polytechnic Institute and State University. (1996).
18. A. Hein, Identification and Bridging of Semantic Gaps in the Context of Multi-Domain Engineering. *Proceedings 2010 Forum on Philosophy, Engineering & Technology*, (2010).
19. Dhamdhere, Systems Programming and Operating Systems, Tata McGraw-Hill Education, (1999).
20. Jørgensen, M. Forecasting of software development work effort: evidence on expert judgement and formal models. *International Journal of Forecasting*, *23*(3), 449-462. (2007).
21. Jørgensen, M. A review of studies on expert estimation of software development effort. *Journal of Systems and Software*, *70*(1), 37-60. (2004).
22. C. M. Barnum, Usability Testing Essentials - ready, set...test!, Burlington: Elsevier Inc. (2011).
23. Nielsen, J, and Landauer, T. K. A mathematical model of the finding of usability problems, *Proceedings of ACM INTERCHI'93 Conference* (Amsterdam, The Netherlands, 24-29 April 1993), pp. 206-213.
24. Voelter, M., & Visser, E. Language extension and composition with language workbenches. In *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion* (pp. 301-304). ACM. (2010, October).

25. Voelter, M., & Pech, V. Language modularity with the MPS language workbench. In *Software Engineering (ICSE), 2012 34th International Conference on* (pp. 1449-1450). IEEE. (2012, June).
26. Voelter, M. Embedded software development with projectional language workbenches. In *Model Driven Engineering Languages and Systems* (pp. 32-46). Springer Berlin Heidelberg. (2010).
27. Simonyi, C. The Death of Computer Languages, The Birth of Intentional Programming The Death of Computer Languages, The Birth of Intentional Programming, (1995).
28. Stoffel, R. Comparing Language Workbenches. MSE-seminar: Program Analysis and Transformation. University of Applied Sciences Rapperswil (HSR), Switzerland, (2010).
29. Kärnä, J., Tolvanen, J. P., & Kelly, S. Evaluating the use of domain-specific modeling in practice. In *Proceedings of the 9th OOPSLA Workshop on Domain-Specific Modeling, DSM*. (2009, October).
30. Caprio, G. Domain-Specific Languages & DSL Workbench. *Dr. Dobbs*. (2006).
31. Tolvanen, J. P., & Kelly, S. Defining domain-specific modeling languages to automate product derivation: Collected experiences. In *Software Product Lines* (pp. 198-209). Springer Berlin Heidelberg. (2005).
32. Kelly, S., & Tolvanen, J. P. Visual domain-specific modeling: Benefits and experiences of using metaCASE tools. In *International Workshop on Model Engineering, at ECOOP* (Vol. 2000). (2000).
33. Hermans, F., Pinzger, M., & Van Deursen, A. Domain-specific languages in practice: A user study on the success factors. In *Model Driven Engineering Languages and Systems* (pp. 423-437). Springer Berlin Heidelberg. (2009).
34. Mernik, M., Heering, J., & Sloane, A. M. When and how to develop domain-specific languages. *ACM computing surveys (CSUR),* 37(4), 316-344. (2005).
35. Wu, Y., Hernandez, F., Ortega, F., Clarke, P. J., & France, R. Measuring the effort for creating and using domain-specific models. In Proceedings of the 10th Workshop on Domain-Specific Modeling (p. 14). ACM. (2010).

Appendix A. Development Effort Estimation

Goal
Provide rough estimates of development effort for a technical equivalent developed using 2nd generation modeling tools (EMF based approach) providing the same value (see table below) as the MOM Workbench.

- **Rough estimates will be used as index:** "Using approach K takes X more/less effort than approach M…"

Guidelines and Constraints for Estimations

- **Developer background:** A team member with Ericsson experience of the current tooling environment (EMF-based).

- **Plugins/Technologies**
  If you use EMF-based plugins not part of the current tooling environment as estimates:
  - Include the plugin if you believe that it would be feasible and realistic to use in a development team at Ericsson.
  - Also, include the training effort it would take for the developer to be able to use the plugin.

- **Relate to development effort from previous development tasks**
  How long time did it take to for previous development tasks with similar conditions?

- **Assumptions**
  Tell us about the assumptions that you made to arrive to the estimates.

<table>
<tr><th></th><th>Value of MOM Workbench</th><th>Current tooling</th></tr>
<tr><td></td><td>Meta model for MOM and delta MOM with validation rules

Delta model consistent with MOM model (Delta model reference actual MOM model)</td><td>Model with delta info
<table>
<tr><th>Activity/Value</th><th>Effort (person week)</th></tr>
<tr><td>Define the domain (MOM, deltaMOM)</td><td></td></tr>
<tr><td>Modify the meta model so delta information can be added to MOM model.

(One MOM model</td><td></td></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td></td><td>
<table>
<tr><td>with many delta information)</td><td></td></tr>
<tr><td>Implement validation rules</td><td></td></tr>
</table>
Total Effort:
</td></tr>
<tr><td></td><td>Remove manual transformation from delta model to model
(i.e. Merge information in delta model to MOM model)</td><td>
<table>
<tr><td>Activity/Value</td><td>Effort (person week)</td></tr>
<tr><td>M2M transformations to “merge” changes in delta model to MOM model</td><td></td></tr>
</table>
Total Effort:
</td></tr>
<tr><td></td><td>Projection for Feature Expert</td><td>
“CPI-view”.
<table>
<tr><td>Activity/Value</td><td>Effort (person week)</td></tr>
<tr><td>Present same or similar information as the CPI of MOM</td><td></td></tr>
<tr><td>Present same of similar information as delta MOM</td><td></td></tr>
<tr><td>Define delta information to MOM model.</td><td></td></tr>
</table>
Total Effort:
</td></tr>
<tr><td></td><td>Projection for Review Group</td><td>
<table>
<tr><td>Activity/Value</td><td>Effort (person week)</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td></td><td>
<table>
<tr><td>Present summary of changes in deltaMOM</td><td></td></tr>
<tr><td>Present comparison of deltaMOMs.</td><td></td></tr>
<tr><td>Add additional information to deltaMOM such as comments and status(Created, Committed, Closed, etc)</td><td></td></tr>
<tr><td>Validate model and present validation result.</td><td>.</td></tr>
</table>
Total Effort:
</td></tr>
<tr><td></td><td>Projection for Model Developer</td><td>
<table>
<tr><th>Activity/Value</th><th>Effort (person week)</th></tr>
<tr><td>Preview how delta information would be merged to MOM model.
M2M transformation to an “intermediate model”</td><td></td></tr>
<tr><td>Select delta model to merge with MOM model. (GUI and commands to invoke transformations)</td><td></td></tr>
</table>
Total Effort:
</td></tr>
</table>