CHALMERS

Empirical assessment of Model-Driven Engineering in a middlesized e-commerce organization

Master of Science Thesis in Software Engineering and Technology

EMIL SELMERYD

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, May 2012

Empirical assessment of Model-Driven Engineering in a middlesized e-commerce organization

EMIL SELMERYD

Examiner: MIROSLAW STARON

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 May 2012

Abstract

In this thesis we strive to assess the feasibility of Model Driven Engineering (MDE) in a middlesized e-commerce organization. We intend to fill a gap of empirical research on the subject matter by providing our own case study. Using a qualitative data analysis approach, we base our research on data gathered from a project carried out at an internet auctions and valuations company. We raise the question of when to implement MDE and reflect on the idea that MDE is suitable only in situations when existing company methods are not sufficient to solve the problem at hand.

Acknowledgements

I would like to thank the people at Kvarndammen KVD for giving me the chance to conduct my research. Without their economic support and their openness to cutting edge technology this thesis would not have been possible. I would especially like to thank Jan Book who helped me in starting up this project. His leadership and critical eye have been outstanding. Also, sincere thanks to Calle Kabo for guiding me through the technical jungle at KVD and for his professional and creative feedback.

Also, I must thank my supervisor Lars Pareto for his support during the work of this thesis. His enthusiasm, academic expertise and experience in the field of Software Engineering have been invaluable.

Emil Selmeryd, Gothenburg 2011

Contents

Chapter 1

Introduction

Model-driven engineering (MDE) is a software development approach that builds on the use of models. The intention behind MDE is to raise the level of abstraction from just coding to also use models, thereby improving productivity, maintainability and portability. Promoters of MDE have long claimed this approach to provide the same level of improvement to developers as the transition from assembler to procedural programming languages offered in the 1960's [1].

MDE, however, is controversial. Results from recent empirical studies on MDE [2][3] contradict improvement claims such as the one mentioned above. In fact, the empirical support is scarce regarding actual benefits of MDE [4]. The few empirical studies in favour of MDE, explore large scale industries (such as Ivensys Rail [1] or Motorola [5]) which often have a history of using engineering methods similar to MDE. Little research on the benefits of MDE in smaller organizations has been carried out.

This controversy and the limited scope of success studies makes it difficult for companies to decide on whether or not to apply MDE and if it would be a feasible investment.

Kvarndammen (KVD) is one company with this problem. KVD is an actor in internet auctions and valuations. In 2010, they sold 33 000 objects online and are expecting an increased activity in coming years. Because of this increase, they are looking for ways to increase maintainability of their current auction system and to make new kinds of auctioning objects easier to implement. For this, MDE, as described by proponents, appeared to be a promising solution. What KVD did not know was whether or not MDE would be feasible and effective for their specific system.

Purpose statement

The purpose of this case study is to explore the feasibility of MDE for middle-sized organizations working in an e-commerce context (such as KVD). At this stage in the research, the feasibility of MDE will be generally described as whether or not an MDE approach is manageable and beneficial. Manageable, in the sense that the end product satisfies the requirements, and beneficial, in the sense that an MDE approach would decrease the costs and/or improve the value of the end product compared to an existing development approach.

Relevance of study

The relevance of this study is for organizations deciding on whether or not to adopt an MDE approach. This study aims to provide a real case scenario with lessons learned that could help to make such decisions. Also, this study is relevant as it adds to the knowledge-base of MDE research, particularly by providing empirical results.

Research questions

In order to state the research questions of the case study, we need to briefly describe the problem KVD had with their auction system. The problem was that their process of modifying the information displayed for a certain object type was slow and tedious. In order to make a change, something called object templates had to be modified. An object template defined what information could be displayed for a certain object type, for instance the object template for object type motorcycles required different information than the object template for object type boats. This process needed to be improved, in order for the organization to handle the increasing flow of objects going up for auction in an efficient and correct manner.

The case study states the following research questions in order to determine the feasibility of MDE for KVD's problem:

- RQ1: How does the introduction of MDE alter the object template creation and maintenance process?
- RQ2: What performance gain, in object template creation and maintenance, can be achieved by using MDE in the systems of the company?
- RQ3: Can MDE be recommended as a solution to KVDs current problem with object template maintenance?

To answer these research questions our case study applies mixed methods and triangulation. Methods include focus groups[6], observations[6][7], archival analysis[6][7], literature study[6], qualitative data analysis[6], iterative prototyping[8] and diagnostic evaluation[9].

The core of our study is a prototype MDE tool developed to speed up and reduce routine work in KVD's current processes. The prototype we developed, called OTM, supports a domain specific language [10] with object template management as problem domain. It uses concepts from the problem domain, such as Field and Template to maintain a high abstraction close to the problem space. As output, it generates code that can be used by other components in external systems.

Our results show that for middle-sized e-commerce companies, MDE is just one out of many approaches to raise the level abstraction and to increase productivity, maintainability and portability.

Chapter 2

Theoretical framework

This chapter will explain the theoretical framework used for this thesis, which in our case means the main technologies used in development of the prototype. We will also look at related research that is used as a basis for our reasoning around the topic of MDE.

2.1 Model Driven Engineering

A quick search online for “what is Model Driven Engineering” returns few results. There is a Wikipedia entry on the subject but the actual definition of MDE is in lack of a reference. Here are, however, four definitions from four different sources:

- “Model Driven Engineering (MDE) is a software development approach family based on the use of models in the software construction. It allows the exploitation of models to simulate, estimate, understand, communicate and produce code.” [11]
- “Model driven engineering is based on the systematic use of metamodels and transformations.” [12]
- “MDE is about the use of relevant abstractions that help people focus on key details of a complex problem or solution combined with automation to support the analysis of both the problem and solution, along with the mechanism for combining the information collected from the various abstractions to construct a system correctly.” [13]
- “[MDE is] the systematic use of models as primary artifacts during a software engineering process.” [14]

Although these quotes are formulated in different ways, they are not contradictory. The reason for their differences could be explained by MDE lacking a formal definition; there is no official organization behind the acronym, it is rather a buzzword that has become

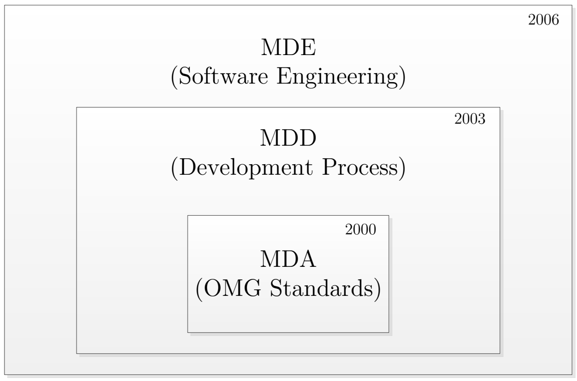


Figure 2.1: Figure showing the evolution of Model driven-families. Original picture by Ameller [15].

commonplace in recent years. Although a buzzword, it is still a valid software engineering process because of its widespread use and acceptance in the industry.

From the above definitions it is, however, possible to draw the conclusion that MDE is a software development approach that by a systematic use of models and transformations strive to help people focus on the key details of a problem. MDE concerns not only implementation but also analysis. After the construction of a model it is possible to verify it and receive feedback from stakeholders, before any implementation has been done.

A part of MDE is transformations and a common transformation is code generation. The vision of MDE is that after creating a domain specific model, the user should be able to generate code from it by the click of a button. Full code generation means that all code is generated without any additional code needed to be written manually. But even if full code generation[1] is not achievable, partial code generation can still be advantageous.

DSLs are common in the realms of MDE(see section 2.1.1 on page 5). They are developed in MDE by the use of metamodels (section 2.1.2, page 5).

Related terms in the Model Driven-family are Model Driven Development (MDD) and Model Driven Architecture (MDA). MDA is a trademarked acronym from the Object Management Group [16] to brand architectures developed using modeling standards such as UML. Ameller [15] explains that MDA is something that must follow a modelling language since it advocates the use of models as primary artefact. This tie to modeling languages makes it the strictest model-driven family. MDD is basically a term for development of software using models. What the models look like does not matter; they can be graphical, textual or other. MDE is the broadest term of the three and incorporates

[1]"Full code generation" is also a matter of definition. Code can be considered as fully generated in the application context, but at the same time only partially implemented in the system context.

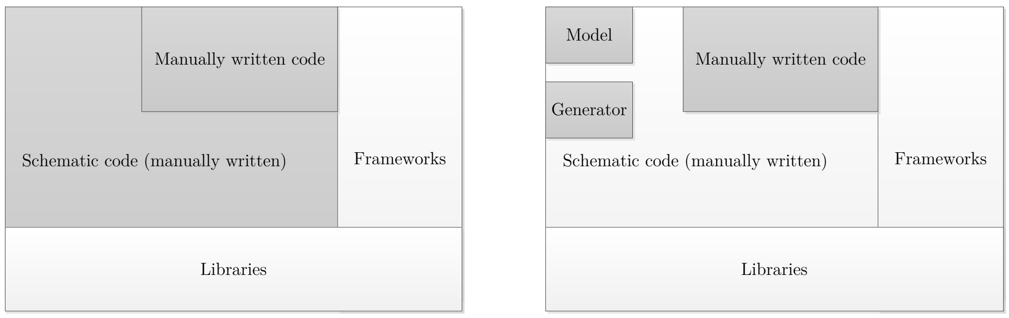


Figure 2.2: Left; Figure showing schematic code written manually. Right; Figure showing schematic code being generated. Original picture by Merks[17].

software engineering parts such as requirements, testing and management. Figure 2.1 shows the MD*-technologies relationship, type and history.

A key guideline for MDE is to separate schematic code from creative code. By schematic code we mean repetitive code that might reappear in project after project. By creative code we mean the challenging parts of programming that demands special attention from the developer. MDE aims to generate the schematic code as far as possible and then let the developer manually write the creative code (see figure 2.2).

Gherbi [11] makes no difference between MDE, MDD or MDSE. Mohagheghi and Dehlen [2] also suggests that MDE and MDD can be used interchangeably.

Through the course of this thesis, however, MDE means the development of MDD tools. We make a distinction between the two but as can be seen from above, sometimes there is not a clear difference.

There are currently many MDE technologies available on the market. Some common technologies are Microsoft DSL Tools, XSLT,the Eclipse Modeling Project and Metaedit+. We used the Eclipse Modeling Project and our hands on MDE experience comes from this perspective – we do not have sufficient experience of other modeling tools to make any comparisons.

2.1.1 Domain Specific Languages

Domain Specific Languages (DSLs) are languages developed to work in a small problem domain. The opposite to a DSL would be a general-purpose language such as Java or C. DSLs function on a higher abstraction level than general purpose languages and by doing so aims to (1) raise the level of abstraction beyond programming and (2) generate final products(in the form of code or other data) [10]. Because a DSL is specific to a problem it is possible to create generators and languages for that purpose and thereby achieving automation and increased productivity.

DSLs should not be confused with APIs, although they share similarities. Without going into a detailed discussion on the definition of a DSL we would like to suggest that an API requires a programmer's know-how but a DSL does not. Because of the raised abstraction level, knowledge of the problem domain is enough to operate a DSL. With that said, we would like to admit that the lines between DSLs, APIs and even general-purpose languages are blurry at times.

To make the distinction somewhat more clear, we can take the example of the system being developed in this thesis. The prototype we developed, called OTM, is a DSL that has the problem domain of object template management. It uses concepts from the problem domain, such as Field and Template to maintain a high abstraction close to the problem space. As output, it generates code that can be used by other components in external systems.

2.1.2 Metamodeling

Stahl and Völter [18] acknowledges metamodeling as one of the most important aspects of Model-Driven Engineering. They list the following activities of MDE to display the importance of a metamodel:

- Construction of domain-specific modeling languages: the metamodel describes the abstract syntax.
- Model validation: it depends on the constraints set in the metamodel.
- Model-to-model transformations: their complexity depends on the design of the two metamodels.
- Code generation: it refers to the metamodel.
- Tool integration: it is based on the metamodel.

What follows is that metamodelling affects the quality of all the above listed activities of MDE.

There is an important difference between abstract and concrete syntax, in regard to a domain-specific language. The abstract syntax is defined by the metamodel the concrete syntax by the interface generated from the metamodel. An example of a concrete syntax would be the syntax used by, for instance, a graphical or textual editor.

A meta relationship is something that should be seen as relative to a model. Take for example an instance of a Car. The car has a name "Audi" and a license plate "XYZ321". A metamodel to this instance would be its class Car. A Car class has two attributes "maker" and "licensePlate". A metamodel to the class would be a model describing the elements Class and Attribute. This sort of meta-relationship could go on forever to reach until one would reach the desired level of abstraction. In the case of MDE and EMF the levels of abstraction tend to stay between these three levels. The third level models the elements with which you design a DSL with (Ecore EClass, Ecore EReference and so on). The second level is where you would specify the DSL elements and at the first level is the instantiation of the 2nd level model (carName, licensePlate).

Furthermore, Stahl and Völter discusses that modeling only makes sense if the model ends up being used in software development. They state that "metamodels should be implementable and support the further development process" [18]. In order for metamodels to live up to that statement it is necessary to not only create the metamodel but to also adapt supporting development tools. This reasoning supports the findings of Kelly and Tolvanen's research [10] and the idea that DSLs increases productivity the highest. Modeling alone is not as efficient as creating a DSL. There is a cost factor in order to keep models updated and synchronized. Traditionally this has gone two ways, either:

- Once implementation starts, models are no longer maintained and becomes outdated.
- Once implementation starts, models are kept up to date at the cost of both money and development time.

Pitfalls in metamodeling

Stahl and Völker list a number of issues and common errors in metamodel design.

- Using interfaces the wrong way: interfaces should be used to model common attributes and operations among entities.
- Accidentally finding oneself on the wrong metalevel[2]: it is often hard to distinguish between metalevels and to decide whether an entity should be in the metamodel or the implementation model.
- Modelling an ID the wrong way: there is a need that an entity has exactly one attribute titled ID. A common mistake would be to add an attribute to an entity, call it ID and set the multiplicity to exactly one. The right way to do this would be to use a constraint.

An interesting part related to metamodeling is how to model behaviour. Metamodels can do this in two ways; either they implicitly hide the behaviour within or the behaviour is modelled explicitly by using concepts such as state diagrams.

EMF has support for modelling behaviour, in the form of EOperations. They work in an implicit way and tell the user where different behaviour (functions, methods etc.) should exist in the system. This is a matter of just marking the behaviour, you are not describing it.

2.2 Domain Specific Modeling

DSM is a superset of DSL and consists of (1) a DSL, (2) a domain specific code generator and (3) a domain framework [10]. DSL have a reported high productivity rate, between 500-1000% [10].

[2]M0,M1,M2,M3 etc.

> "Compared to creating the initial prototype, turning it into something that could be used in the real world is a much harder task. Building such a system without significant personal experience of industrial scale DSM appears to be near an impossibility. This is sad, but not surprising: if building a metamodel requires the top expert developer in a domain, building a meta-metamodel for all such top developers requires the very highest levels of experience as well as intelligence and skill. The only other hope is dumb luck, but then current authors probably used most of that up already, leaving little for newcomers!" [10]

The authors points out meta-metamodeling as a critical factor for the success of a DSL-tool, because if the meta-metamodel is updated in new releases it would mean too much work to adapt your current models. Compare this with updates such as notation and modeling tool functionality, from which modelers could benefit without much redesign, and one can see why the meta-metamodeling is such an integral part.

Kelly and Tolvanen [10] rates EMF as a level 3 DSL-tool for real word cases and level 4 for simpler examples. Level 3 is defined as "metamodel, generate a modeling tool skeleton over a framework, add code" while level 4 is defined as "metamodel, generate the full modeling tool over a framework"

Even though DSM takes away parts of traditional software product development, the important part of design is still present. It is only the error-prone, tedious and time-consuming task of turning design into finished product that is automated [10]. By removing those tasks developers will be given more time to spend on the creative part of development. At least that is the aim of DSM.

2.3 Eclipse Modeling Tools

Eclipse Modeling Tools is a package available for download on the official Eclipse site. It contains EMF, GMF and many other frameworks that are used to leverage models. Once installed it has its own IDE and from the Help menu one can easily install additional modeling components such as ATL and Acceleo. EMF, Acceleo and ATL were the primary frameworks used to design the OTM and when we refer to Eclipse modeling tools in this thesis, these three are what we are referring to. While the plug-ins and frameworks in the following text are highly developed and provide many features, we will focus on those features related to the OTM development.

EMF

EMF comes with the Ecore framework and can be used to create metamodels. The official documentation describes EMF as a Java framework and code generation facility for building tools and other applications based on a structured model. EMF supports code generation to Java from an Ecore model. It also comes with a graphical user interface for model creation that provides the user with an intuitive drag and drop interface but it also supports textual specification of models using XMI. The generated implementation code can be modified, for instance by adding methods and instance variables. Upon regeneration after a model change, this modified code is preserved; the user would only have to update the code if necessary to match the changes in the model.

Since EMF only supports generation to Java, custom code transformation had to be written for the OTM. This can be achieved by using Acceleo which is described below.

EMF uses a core metamodel called Ecore that is similar to MOF. Ecore consists of a multitude of classes that all starts with the letter E in order to signify their belonging to EMF. To name a few of the classes there are EClass, EReference, EEnum and EAttribute. These were the main classes used to create the OTM. They are similar to the elements in Java and can be understood straightforwardly if the user has an object oriented background. When designing a metamodel it is common to start with a primary class and think about how that class relates to other classes in the domain. For instance, a class for car is created with attributes brand and price. Then a car might have many bids, so a bid class is created with a one to many reference from car to bid. When modeling with Ecore the concept of relations becomes much more prevalent, in our experience. This is thanks to graphical representation of relations and the fact that we understand visible relations better than text relations.

Acceleo

Acceleo is a Model-to-text (M2T) plug-in for Eclipse that allows you to generate code from a model. It uses the Model to text language (MTL) originally create by the Object management group (OMG). MTL is a markup language that is similar to HTML – each start tag must have an end tag (or finish with /). Any text written outside of a tag pair is simple printed to file. Elements in a model are accessed by referencing a metamodel node, for instance [car.name/] would provide the name variable of the element car. We can say that dynamic code is written inside tags (for-loops, conditionals) and static code outside. Acceleo uses a concept called Template. A Template normally corresponds to a file to be generated. If a Java-class is to be generated, its class header for instance (public class [c.name/]) would be written as static code in the Template, while its variable value would be dynamic, inside MTL-tags. Acceleo comes with useful features such as syntax highlighting, content assistant and error detection.

ATL

ATL allows Model to model transformations. In the ATL documentation, we can read that model transformation, in the context of having models as first class entities, appears to be a central operation for model handling.

ATL supports two meta-metamodel technologies, Meta object facilities by the OMG and Ecore meta-metamodel defined by EMF, which was the technology used in the development of the OTM. In other words, ATL can handle metamodels specified according to either the MOF or Ecore semantics.

Model transformations can be useful to transform a PIM to a PSM. This can be done by adding platform specific elements to a model or creating new relations. ATL stands for Atlas transformation language and consists of three kinds of ATL units: ATL transformation modules, ATL queries and ATL libraries. These units can, according to their type, be composed of ATL helpers, attributes, rules, matched and called rules. The language is based on OMG OCL norm. Its syntax is similar to a functional language, such as Haskell.

OCL operations are supported by ATL and are used for standard commands, for instance car.oclType() will return the type of car. An ATL module holds many ATL helpers and rules. A module is normally created for a certain M2M transformation, for example car2invoice could be a module for transforming a car model into an invoice model. This module would consist of a rule called carname2invoicename that would simply use the car's name for the name of the invoice. A helper could be used to determine the price of the car. The helper could take the brand of the car and set a price accordingly (if Volvo then 20 000 €, if Lamborghini then 100 000 €).

```
helper context metamodel!Car def: determinePrice(): Int =
   if self.brand.oclIsTypeOf(Volvo) then
     20000
   else if self.brand.oclIsTypeOf(Lamborghini) then
     100000
   else
     -1
   endif;
```

Family to Person example

The basic example in the ATL documentation gives a good introduction to M2M-transformations. We want to transform a family into persons. This means using a metamodel for families and one for persons. A family, in this example, can consist of zero or one father and zero or one mother. Each family can have zero or more sons or daughters. A person consists of a first and last name and has a gender. The family metamodel consists of a Family class with a last name variable and a Member class with a firstname. Family roles are modeled as relations. The persons metamodel consists of a Person class with variable full name and genders represented as subclasses. An ATL transformation allows us to take an instance of a Family and recreate it as an instance of a Person. This means transforming relations into genders (if relation isFather then male subclass etc.) and combining a family name and a first name to create a full name. The families to person example provides an illustrative implementation of M2M. More commonly M2M could be used to transforms an abstract model to a certain technological domain. This was the case for the OTM, were we transformed a model into a SQL-domain. For further reading we refer to the implementation chapter.

2.4 Studies on productivity and MDE

Focused on qualitative research methods, a study by Hutchinson et. al [4] investigated social and technical factors that influence organizational responses to MDE.

They discuss claims that have been made about how MDE increases productivity, maintainability, portability and interoperability but that these claims have developed without the support of empirical data. Also there are conflicting reports of both successful and unsuccessful projects implementing MDE.

They also propose a table, which shows that MDE not only has positive effects on productivity, portability and maintenance, but also negative.

The conflicting results of MDE research are confirmed. Positive productivity was found to be anything between 20% and 800%, while negative productivity, received from anecdotal evidence, was quantified at 27%. Hutchinson et al. suggest that this conflict comes from that it is difficult to measure MDE productivity benefits and also, that a MDE benefit might come with a drawback. For example, they mention code generation, which is usually considered as having a positive effect on productivity, as something that could come with a drawback in the form of having to integrate generated code causing maintenance problems. It is concluded that the dependencies between such factors are unknown.

The respondents of their questionnaire were to a large extent experienced software engineers (40% had over 10 years experience in the field). Also, the majority of respondents were positive towards MDE as a method.

A majority (74%) thought that MDE required significant extra training for current staff. Regarding code generation, they discovered that there were more practitioners for whom code generation had a positive impact on their productivity than there were for those who had a problem integrating generated code. Two thirds thought that MDE helped understandability between stakeholders, while around 25% thought that MDE resulted in unexpected confusion/misunderstanding between stakeholders.

By having paired questions, they managed to demonstrate a couple of dependencies and conflicts, such as the following:

> For example, a company that believes that MDE will enable them to hire software engineers with less experience but doesn't prepare for the necessary training may well find itself in a difficult predicament and with a need for training time and costs that have not been planned for. Similarly, if for some legitimate reason a company finds that it must manually adapt code that is autogenerated, it will probably need to explicitly address how it should be done and what procedures are required to control the process if it is not to encounter difficulties with keeping the model and code synchronized.

This supports the idea that a benefit gained from applying MDE might come with a drawback. They also conclude that understandability improvements are essential to the perceived success of an MDE implementation.

From the interviews carried out it was clear that the selection of an initial project to try out MDE, is very important; the suitability of the project can make or break the successful implementation of MDE. The users of MDE must be motivated and the organization must be motivated to make the project a success, since the adoption of MDE is more turbulent than the adoption of, for instance, a new programming language. Often it will be difficult to introduce MDE in an organization that already has a satisfactory way of working because in such a case a radical change like MDE is not motivated. On the contrary, Hutchinson et al. suggests MDE should be more readily embraced in an environment where the current process is time-consuming, difficult and uninspiring.

Many adopters consider prototyping MDE on projects isolated from important deadlines (this was the premise of our OTM implementation). Unfortunately, this might lead to that not the best people are working on the project and that it will not receive top priority and, as have been discussed by Kelly and Tolvanen[10], top people and priority is necessary for successful creation of a DSL. Many MDE implementations have been successful thanks to its application on a critical project.

Organizations that have hardware as their main product are generally more welcoming to MDE compared to companies that have software as their main production artefact; “companies that don’t do software do MDE”.

Negative and positive influences of MDE

Hutchinson et al.[4] proposed a number of aspects of MDE that had both positive and negative influences on a software project. The idea was that a positive change might also come with a negative. For instance, code generation has a positive effect on speed, but can have the negative effect of time to integrate the generated code.

Table 2.1 on page 13 shows the suggested aspects. For this thesis, the table has been modified. The aspect regarding portability was dropped since there was no data coming from the case study related to that issue. When analysing the results from the case study this table will be used as a reference.

Impact Factor	**Illustrative Examples of MDE Influences**	
	Positive Influences	**Negative influences**
Productivity		
Time to develop code	Reduced by: automatic code generation.	Increased by: time to develop computer-readable models; implement model transformations, etc.
Time to test code	Reduced by: fewer silly mistakes in generated code; modelbased testing methods, etc.	Increased by: effort needed to test model transformations and validate models, etc.
ROI on modeling effort	Positive influences of modeling: more creative solutions; developers see the "bigger picture".	Negative influences of modeling: "model paralysis"; distracting influence of models.
Portability		
Time to migrate to a new platform	Reduced by: simply applying a new set of transformations.	Increased by: effort required to develop new transformations or to customize existing ones.
Maintenance		
Time for stakeholders to understand each other	Reduced since: easier for new staff to understand existing systems; code is "self-documenting".	Increased since: generated code may be difficult to understand.
Time needed to maintain software	Reduced since: maintenance done at the modeling level; traceability links automatically generated.	Increased since: need to keep models/code in sync, etc.

Table 2.1: Illustrative influences of MDE

Chapter 3

Method

The following chapter introduces the methods used to gather and analyse the data that was used as a foundation to answer the research questions. We describe how we gathered information, how it was analysed and how we conducted the practical development of the prototype tool.

3.1 Research strategy

Case studies are suitable for SE because they study a contemporary phenomenon in its natural context [7]. It was also an appropriate format for the purpose of this thesis because only one company was available for data collection. Another advantage of a case study is its possibility to go into depths on a specific topic. A deep understanding of a topic can be achieved through, for instance, interviews and observations.

The case study was carried out on site in the summer and fall of 2011 over a period of four months.

3.2 Research procedure

In order to answer the research questions, the problems were examined from multiple perspectives to provide a triangulation of facts. Data was collected from observations, interviews and focus groups during our time at the company. Also, a literature study was conducted to complement the empirical data. Finally, a diagnostic evaluation of the prototype developed was made to provide some additional empirical ground on which to formulate the conclusions. Figure 3.1 shows the activities and relationships of our research process. Boxes denote activities, curved boxes denote text documents, rhomboid denotes other kinds of artefacts (software products, application of theoretical models), whereas the ovals denote theoretical results. The activities are briefly described in the following sections.

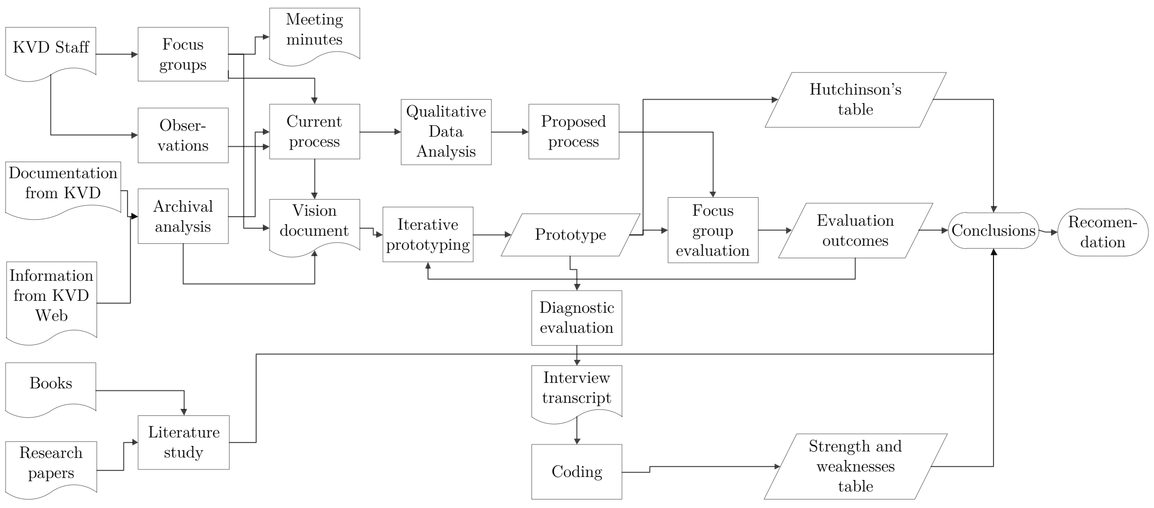


Figure 3.1: A figure showing the research process.

3.3 Research design

This research design section briefly describes how this study was carried out. Figure 3.1 on page 15 illustrates the research process and provides information on two things. First, it shows how the above described research methods are related. Secondly, it shows in what order the research was conducted (from left to right).

The first output of the research was the Vision document. This document summarized the requirements specified by KVD. The Vision document was created based on the initial archival analysis made and the focus groups held at KVD. The document was used to create a common understanding between stakeholders and developers.

After that we finished the Vision document, work began on the actual prototype. We used an iterative work method were the prototype was developed in weekly cycles. At the end of a cycle, we received an evaluation of the current status of the prototype from the focus group. This evaluation was then used to further develop the prototype.

The description of the current process was based on the archival analysis, focus groups and observations. The description can be found in chapter 5 and was also featured in the Vision document. A qualitative data analysis was made on the current process from which we extracted a basis for the proposed process.

The proposed process was evaluated in focus groups. These meetings resulted in the evaluation outcomes that lead to our final conclusions.

We scheduled the diagnostic evaluation near the end of the research, when the prototype was near completion. The diagnostic evaluation was used to improve the scientific level of the case study and to reduce the possible biased opinions that could influence our conclusions, as was mentioned in the above Case study research section. The evaluation was recorded as an mp3-file which was transcribed to a document. This document was then coded to finally be summarized in a strength and weaknesses table.

The use of Hutchinson's table was a way to relate our research to previous. We used a modified framework created by Hutchinson et al. [4] with the goal in mind to find out how the prototype affected areas of productivity and portability.

3.4 Research methods

The following section describes the steps involved in our research. Every step is featured in illustration 3.1 on page 15.

3.4.1 Observations

The data collection method of observation was used during group meetings. Often the project manager and the lead developer started discussions during the group meetings that generated valuable project data. These observations were summarized after each meeting in a Meeting minutes document – every meeting was briefly written down in-between 3-20 sentences. This document proved to be a major data source for the case study, containing the company's requests, problems and feedback.

3.4.2 Archival analysis

An archival analysis is a case study method that means looking at existing documentation. In our case, we analysed the documentation from KVD and put the results in the Vision document. Input was documentation from KVD, for instance a document describing how a car was received, registered and tested. Another input was information about the process through info texts on KVD's website.

3.4.3 Literature study

A method that helped us gain a greater understanding about MDE and to build on previous research. Books about MDE provided information how to develop the prototype and also helped us to make our final recommendation. A complement to the books was research papers. They were gathered from online databases such as IEEE Explore and ACM digital library.

3.4.4 Current process

The description of the current process was based on the archival analysis, focus groups and observations. The description of the current process can be found in chapter 4 and additional info about it can be found in the Vision document. A qualitative data analysis was made on the current process from which we extracted a basis for the proposed process.

3.4.5 Vision document

The first output of the research was the Vision document. This document summarized the requirements specified by KVD. The Vision document was created based on the initial archival analysis made and the focus groups held at KVD. The document was used to create a common understanding between stakeholders and developers.

3.4.6 Qualitative Data analysis

Qualitative data analysis methods were used for the data collected in the case study. Runesson and Höst explain that the objective of qualitative data analysis is to derive conclusions from the data, keeping a clear chain of evidence. In our case, this meant mapping ideas from the project meetings on to the research questions. By doing so, we were able to make the conclusions described in chapter 5. This form of analysis was a continuous process throughout the data collection. For instance, during a meeting the idea of a Template field group arose. Then this new insight would need further investigation through, for example, another meeting.

3.4.7 Iterative prototyping

The prototype was developed using an iterative method. A small set of requirements was implemented each week and then showcased in the focus groups. Then the prototype was modified based on the feedback from the focus group and new requirements were set for the next iteration.

Nielsen[19] stresses the fact that a usability interface should be developed iteratively because it is virtually impossible to design a user interface without any usability problems from start. We would like to suggest the same line of thinking for domain models. It is hard to design a domain model without any domain problems, such as missing domain elements, from start.

Iterative prototyping was a natural choice for our project because there were few upfront requirements. The iterative process helps with this problem because it is open to change. By maintaining short iterations and keeping weekly focus groups with the project team we managed to keep the model responsive to change. For instance, the first model developed was simple; it featured only a subset of the requirements specified by KVD. It was, however, useful as a meeting artefact on which we could base our discussion. We learnt that there is better to have something physical as a starting point for a discussion than to not have anything at all.

Iterate basically means re-do, which was the basic design approach for us in this project. We also added onto the model after each meeting which made the process an incremental method as well.

Cockburn[20] mentions two main rework strategies for iterative development.

- Develop the system as well as possible, in the thinking that if it is done sufficiently well, the changes will be relatively minor and can be incorporated quickly.

- Develop the least amount possible before sending out for evaluation, in the thinking that less work will be wasted when the new information arrives.

Of these two strategies, the one most closely resembling our work process is the latter. By having weekly meetings we only developed what was necessary. Iterative development helps to reduce time-waste in that sense.

The openness of iterative development can, however, also be a problem. By allowing change and new requirements, the development time can be drawn out. If requirements are given in their entirety upfront, as could be the case in a waterfall approach, and no change is allowed, than a more accurate time estimation may be given from developers to managers.

3.4.8 Proposed process

The MDE based process proposed to KVD that aimed to improve their business.

3.4.9 Prototype and Focus groups

After that we finished the Vision document, work began on the actual prototype. We used an iterative work method were the prototype was developed in weekly cycles. At the end of a cycle, we received an evaluation of the current status of the prototype from the focus group. This evaluation was then used to further develop the prototype. Output was meeting minutes. Meeting minutes is a written document that summarizes every meeting held over the course of a project. In our case, we wrote down 3 – 20 lines after each meeting held. Meeting minutes turned out to be an effective way to remember what was said at each meeting. Meeting minutes captured the desires of KVD staff, their opinions of the prototype and practical details.

3.4.10 Diagnostic evaluation

By recommendation of UsabilityNet [9], a diagnostic evaluation was chosen as a measurement of the developed tool. The method was selected because it was suitable for projects with limited time and resources. The evaluation was carried out as suggested by UsabilityNet except for a few limitations. These limitations had to be made due to the lack of time available from KVD staff:

- Evaluation was only performed with one user, not 3-5 as suggested.
- Only one administrator was present during the test.
- No developers (except for the user performing the tasks) were present during the evaluation.

We scheduled the diagnostic evaluation near the end of the research, when the prototype was near completion. The diagnostic evaluation was used to improve the scientific level of the case study and to reduce the possible biased opinions that could influence our conclusions, as was mentioned in the above Case study research section. The evaluation was recorded as an mp3-file which was transcribed to a document. This document, the interview transcript, was then coded to finally be summarized in a strength and weaknesses table.

3.4.11 Coding

The method of coding[21] was used to analyze the interview transcript. Common themes were found and coded in the transcript. The goal is to find specific categories for which the nouns and verbs of a text are instances of.

3.4.12 Focus groups evaluation

Focus group evaluations were used in order to make sure that the stakeholders were satisfied with the proposed process and also to involve them in the development of the prototype. KVD staff provided information about the current process by word of mouth.

3.4.13 Hutchinson's table

The use of Hutchinson's table was a way to relate our research to previous. We used a modified framework created by Hutchinson et al. [4] with the goal in mind to find out how the prototype affected areas of productivity and portability.

3.4.14 Evaluation outcomes

The outcomes were documented in the meeting minutes and provided results for us to base our conclusions on.

3.4.15 Strength and weaknesses table

A strength and weaknesses table was designed for the prototype. This provided a clear overview of the quality of the prototype.

3.4.16 Conclusions and recommendations

The final results of our research. The conclusions were the analysed and evaluated results that lead to our end recommendation.

Chapter 4

Results

In the following text we describe the results of our research carried out at KVD. We attempt to present the results in an neutral and objective fashion and leave conclusions and discussions for chapter 5.

4.1 Observations

The results from the observations method can be found in the below section titled Current process.

4.2 Archival analysis

The archival analysis resulted in a better understanding of MDE and provided a knowledgebase for this report. A concrete result is for example the quotes that occur frequently throughout this thesis.

A central discovery from the archival analysis was the following quote:

> [...] if the process already employed by a company is working 'sufficiently well', it will be very difficult to introduce as radical a change to that process as introducing MDE. However, if the existing process is itself a significant risk to a project, or if that process involves the developers in difficult, time-consuming and uninspiring activities, then process change in the form of MDE adoption will be more readily embraced.[4].

We discuss this further in chapter 5.

4.3 Current process

The current process at the company had been around since the release of their web based auctions[1]. In the beginning their domain area consisted solely of cars and the system was tailored for that purpose. With the passing of time, additional objects were incorporated into their range of products and the car tailored system did no longer fit these new objects. To address this issue, the system had to be modified, often with quick fixes. This has resulted in the system of today which is a large legacy system that is hard to maintain, slow and provides unsatisfactory description of objects.

The current process consisted of a practical part (figure 4.1) and a software part (figure 4.2). The practical part was for instance the handling of cars and other objects in the testing facilities and the contact with customers. The software part was the technical process of creating object templates that would suit the objects coming in for auction.

To create a new template with the current process a staff member of IT needs to be available. He/she needs to make changes in the FileMaker based system Kastor, create scripts that transport the template to SQL and to the web. He/she must make changes in 9 places which is a possible source of error. When having to make changes in so many places, developers witnessed that some scripts can be forgotten. Also, there was a performance issue because Kastor was used for other purposes, mainly financial, and had a lot of traffic at certain hours of the day. This resulted in sluggish response for the users of Kastor and meant that if Kastor somehow could be relieved[2] it would benefit by allowing better response times for the staff members, and in the end provide a faster workflow.

The current process has three main actors: the tester, the administrator and the client. The client is the organization that wants to sell something on the company's web page. The administrator is responsible for the contact with the client, the presentation of the object to be sold and follow up. The tester is responsible for testing the object to be sold and assuring its condition. The tester provides his/hers description of an object, the administrator his/hers. The version displayed on the web is the version corrected by the administrator.

At the start of a process a client contacts an administrator, they agree on what terms the object is to be sold. The administrator then collects the object and brings it to a testing station. There the tester follows a template which instructs the tester what aspects of the object are to be tested. This template depends on the type of the object (car, boat, motorcycle etc.). This so called test template, which is in paper form, is carried by the tester upon his/hers inspection of the object and is manually filled in during the course of the test. After the test is finished and the necessary fields of the test template have been filled in, the tester transcribes this information to a digital template on a computer. The test template itself is originally printed out from the computer

[1]There had been a recent transition to object oriented PHP, but many of the concepts from the legacy system remained the same

[2]There was a desire from management to completely remove object template management from Kastor.

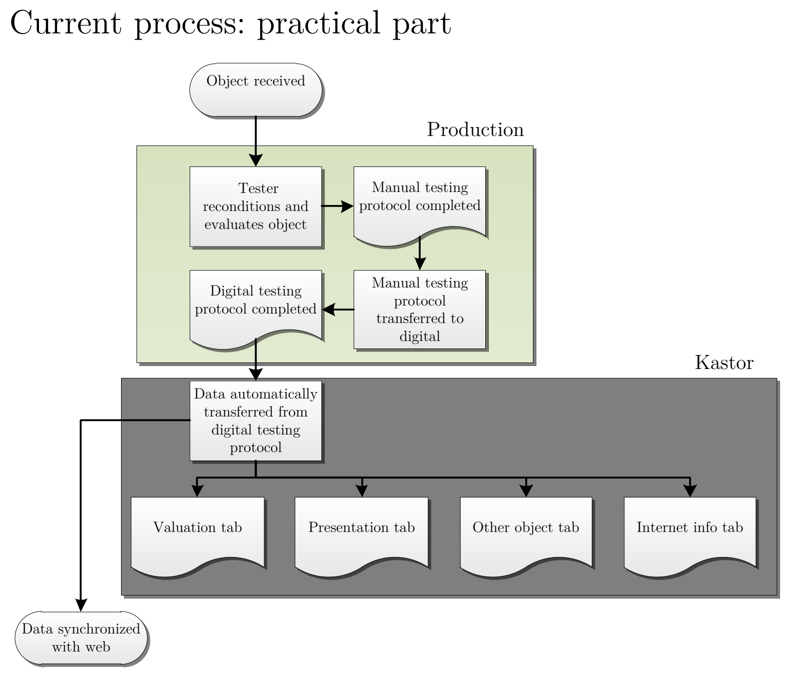

Current process: practical part
Object received
Production
Tester reconditions and evaluates object
Manual testing protocol completed
Digital testing protocol completed
Manual testing protocol transferred to digital
Kastor
Data automatically transferred from digital testing protocol
Valuation tab
Presentation tab
Other object tab
Internet info tab
Data synchronized with web

Figure 4.1: A figure showing the practical part of the current process: receiving objects, evaluating objects and the digitalization of data.

from this digital version. Once all information has been described by the tester, the description of the tester is available in Kastor.

After the tester is finished with his/hers test, an administrator reviews the information provided by the tester and makes corrections. Short forms are written out and any uncertainties are addressed so that the information is presentable to customers on the web. When the administrator is finished with this task, the presentation will be available on-line.

As mentioned above, there are two versions of an object description -– the description of the tester and the description of the administrator. This is important for KVD since it provides that the original test description is kept. If there are any misunderstandings between tester and administrator, it is possible to see who wrote what in the description. Also, if there are customer complaints regarding the description of an object, it is possible for the company to see where in the process an error appeared. Therefore, a requirement of KVD was, what they called "version control" (not to be confused with revision systems such as SVN) that meant that the above described security check was featured in the new system. Also, there was a demand for additional user type descriptions; for instance, it could be possible for the client to provide a separate description on the object. Potentially, there would be even more roles in the future and the system

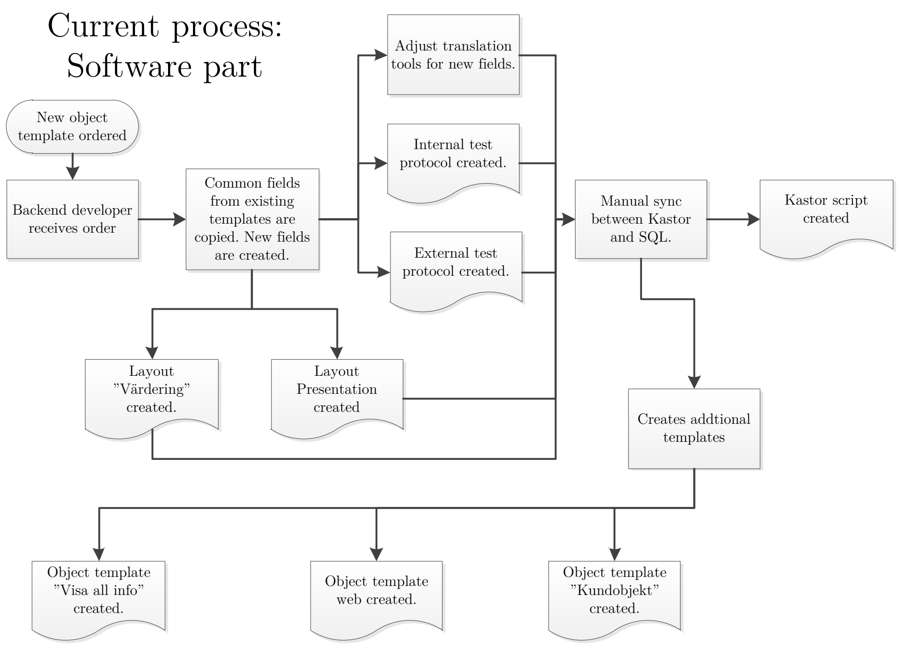

Current process: Software part
New object template ordered
Backend developer receives order
Common fields from existing templates are copied. New fields are created.
Adjust translation tools for new fields.
Internal test protocol created.
External test protocol created.
Layout "Värdering" created.
Layout Presentation created
Manual sync between Kastor and SQL.
Kastor script created
Creates addtional templates
Object template "Visa all info" created.
Object template web created.
Object template "Kundobjekt" created.

Figure 4.2: A figure showing the software part of the current process of managing object templates.

should be adoptable to that.

4.4 Vision document

The resulting Vision document can be found in the appendix.

4.5 Qualitative Data analysis

The result of the qualitative data analysis was a greater understanding of the problem at hand. It resulted in the understanding of the existing process (its lack of flexibility and its lack of speed in getting objects up for sale) and the need for a new one (the proposed process).

4.6 Iterative prototyping

Iterative prototyping resulted in a weekly build of the prototype. The method also resulted in weekly feedback that was gathered in meeting minutes.

We developed an initial skinny model, inspired by Jacobson's concept of skinny systems [22], with basic features. The model was then presented to the project team from whom we received feedback. In an iterative fashion the model was then redesigned and changed after each weekly project focus group.

4.7 Proposed process

To address the current problems of the system a MDE approach was proposed. The vision was to make a model based system that, at the push of a button, would automatically spread out the generated files to the right place in the system. Because KVD had common information (object templates) that they wanted to distribute across multiple platforms (web, mobile and SQL), MDE seemed as a feasible solution to the problem.

The process suggested was initially a web interface from which any staff member at the company could work from and manage object templates. The interface was to be generated from the meta-model. However, since we had previous experience of GMF (the framework for creating graphical model interfaces) we knew that there was a tedious process to create any graphical interface and how to create a web interface simply was not known. Because of these reasons the idea of a custom graphical interface was dropped, and instead an interface based on the Eclipse tree editor was developed. The project team was, however, satisfied with this compromise.

From the Eclipse editor a developer should create an object template and then generate files from it. The transformations would handle all the necessary creations of files for the MVC structure that KVD's web page consisted of. It would generate the database structure, PHP-view files and PHP-ORM-objects – anything necessary for the display of an object on-line. The introduction of this new process would cut down the number of places that needed to be modified each time an object template was to be changed, resulting in reduced lead time, better descriptions of objects and faster sales.

In order to make this process possible Eclipse Modeling Framework was to be used, because of our prior experience with it. The metamodel were to be developed with EMF, Model-to-text transformations could be handled with Acceleo and potential model-to-model transformations could be handled by ATL.

4.8 Prototype and Focus groups

The discussion from the focus groups was summarized in meeting minutes. A prototype was developed to showcase the general functionality of the MDE solution.

A metamodel (figure 4.3) was created for the prototype. The metamodel allowed creation of Templates, which was an abstraction of KVDs object templates. Each template consisted of a number of Fields. A Template could be created for, for example, a car. The car could then have the Fields “motor”, “horsepower” and “model” etc.

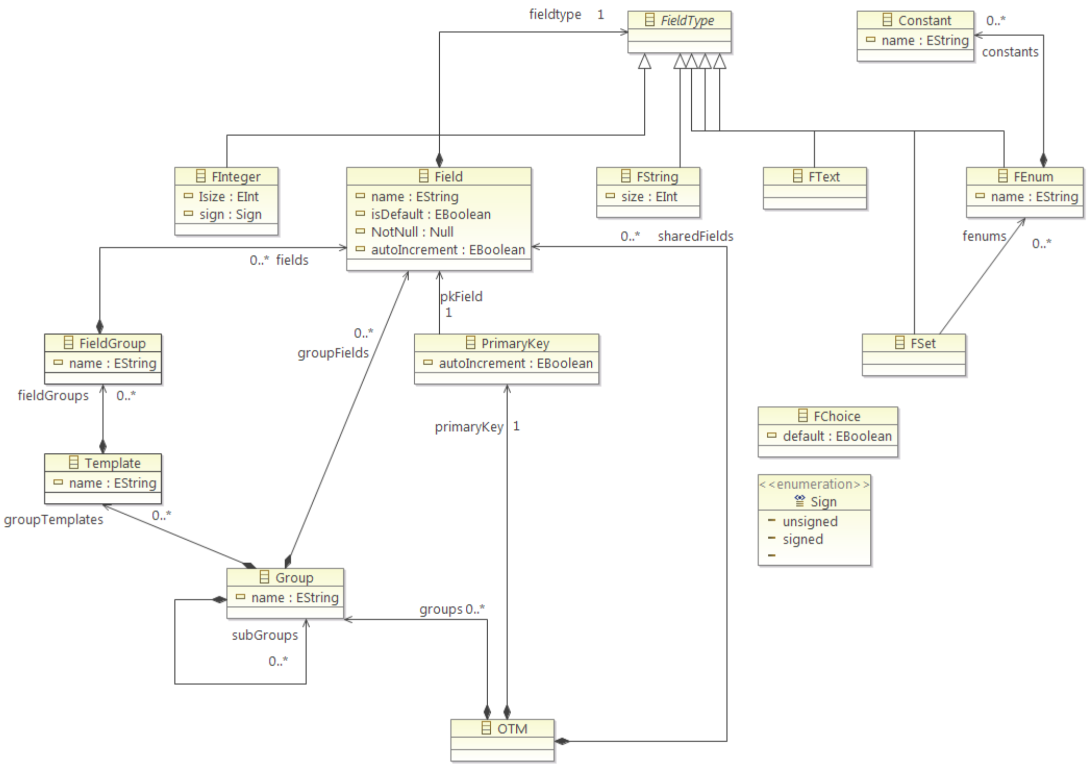


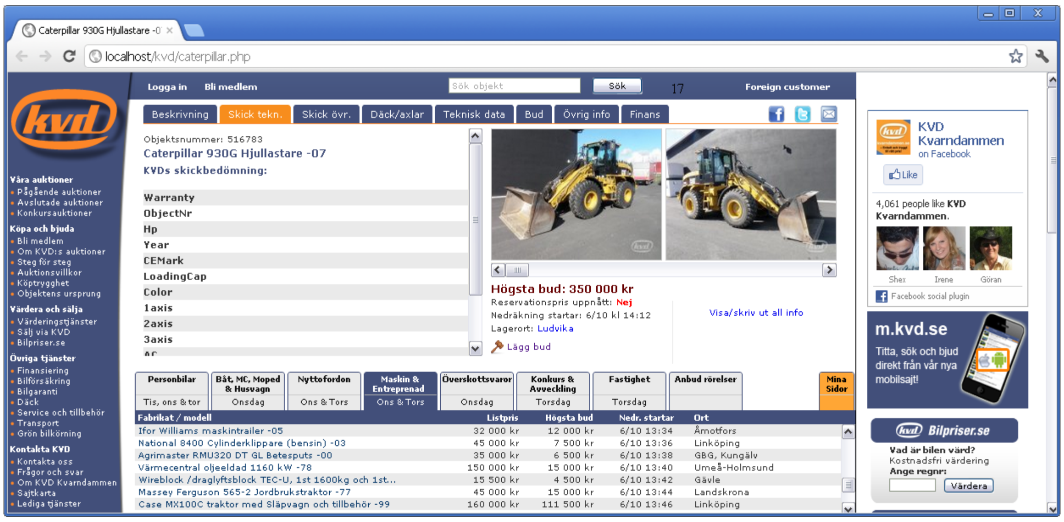
Figure 4.3: The ecore metamodel developed for the prototype.

4.8.1 Implementation of prototype

To demonstrate the MDE approach a tool called OTM (Object Template Manager) was developed as part of the case study. The tool was delivered iteratively and evolved through many versions through the course of the project. Focus of the tool was mainly on the editor and to gain an as accurate representation of the current business model at KVD as possible, but the tool also had automatic partial code generation to display the possibilities of MDE.

The OTM was developed from an Ecore model (figure 4.3). Elements were taken from business domain of KVD and were implemented into the metamodel. After the metamodel was finished, transformations were written for Acceleo. The initial transformations only generated SQL-code and the editor provided no possibilities to set SQL-properties. After weighing pros and cons of having a general editor vs a SQL-specific one, a SQL-specific one was created with the justification that it would be handled only by developers of KVD and because of that there would not be any problems with SQL-syntax.

After the initial model was done, it was demoed for the project team. Based on



Objektsnummer: 516783
Caterpillar 930G Hjullastare -07
KVDs skickbedömning:

Warranty
ObjectNr
Hp
Year
CEMark
LoadingCap
Color
1axis
2axis
3axis
AC

Högsta bud: 350 000 kr
Reservationspris uppnått: Nej
Nedräkning startar: 6/10 kl 14:12
Lagerort: Ludvika
Lägg bud
Visa/skriv ut all info

Personbilar	Båt, MC, Moped & Husvagn	Nyttofordon	Maskin & Entreprenad	Överskottsvaror	Konkurs & Avveckling	Fastighet	Anbud rörelser	Mina Sidor
Tis, ons & tor	Onsdag	Ons & Tors	Ons & Tors	Onsdag	Torsdag	Torsdag		

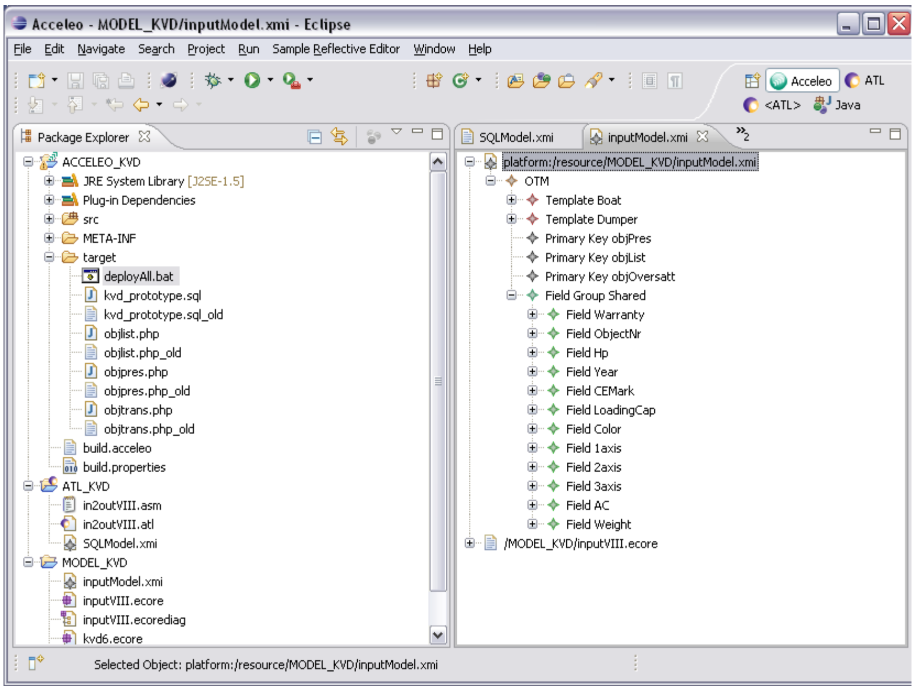
Fabrikat / modell	Listpris	Högsta bud	Nedr. startar	Ort
Ifor Williams maskintrailer -05	32 000 kr	12 000 kr	6/10 13:34	Åmotfors
National 8400 Cylinderklippare (bensin) -03	45 000 kr	7 500 kr	6/10 13:36	Linköping
Agrimaster RMU320 DT GL Betesputs -00	35 000 kr	6 500 kr	6/10 13:38	GBG, Kungälv
Värmecentral oljeeldad 1160 kW -78	150 000 kr	15 000 kr	6/10 13:40	Umeå-Holmsund
Wireblock /draglyftsblock TEC-U, 1st 1600kg och 1st...	15 500 kr	4 500 kr	6/10 13:42	Gävle
Massey Ferguson 565-2 Jordbrukstraktor -77	45 000 kr	15 000 kr	6/10 13:44	Landskrona
Case MX100C traktor med Släpvagn och tillbehör -99	160 000 kr	111 500 kr	6/10 13:46	Linköping

Figure 4.4: A figure showing the dummy HTML-page with a generated table ("KVDs skickbedömning") by the OTM.

the feedback received we decided to re-design the tool. The concept of Model-to-model transformations was introduced and the pros and cons of another technology (ATL) were evaluated. We reasoned that a M2M-transformation would provide for a more elegant MDE-solution but on the other hand it would require another transformation and possibly more effort put on the end-user. The vision of "one-click-everything-is-generated" became one step further away.

In order to improve the automation, we investigated the possibility of a script solution. Apache Ant[23] is used to build Java applications, and was considered as a script solution but when searching for help on how to use it with modelling we found little documentation. Apache Ant-scripting for ATL was well documented and proved successful, alas that help little as long as there were no control for Acceleo or EMF. Eventually, the ANT-solution was abandoned due to lack of time and our failure to find documentation on the topic. Instead a Windows batch-solution was created, that automatically copied the generated files to their respective folders and started a dummy HTML-page (figure 4.4), where the user could see the generated PHP-arrays and SQL-tables displayed as actual KVD tables. After several iterations of the OTM, it finally generated three PHP-files and a SQL-create script. These files had been received from KVD and were reverse engineered. The static code parts were separated from the dynamic parts. Certain dynamic areas were, however, too complex to be generated so limitations were made. In the PHP-files the code being generated was an array that contained the names of the field in the SQL. The SQL generation on the other hand allowed for more variability. Primary keys could be set by adding a primary key node in the editor. Default, auto_increment was set by Boolean properties.

The design of the metamodel passed several phases during development. In the initial design, a Template was represented by a node and under that node each field



Selected Object: platform:/resource/MODEL_KVD/inputModel.xmi

Figure 4.5: A figure showing a inputModel xmi with Template (Boat and Dumper) and Field (Warranty, ObjectNr, Hp etc.) instances.

belonging to that Template was represented as a child node. This solution provided, according to the project team, a good overview of Templates. It was possible to see which Templates were available in the system and what their Fields were thanks to the graphical tree-view. One problem with this solution was the duplication of fields. KVD's current database structure only allowed one instantiation of each field (i.e. one field shared many Templates). With that in mind, we proposed another design that only allowed one instance of each Field. Instead of having fields instantiated for each Template, Fields where instantiated in one place and then referenced by Templates (figure 4.5). This created another way of interacting with the OTM. To an extent, the overview became less obvious since the tree structure for fields was lost. Instead a Template was represented by a node and its fields were visually represented as an array in the properties view. On the other, adding and removing fields became quicker for the user since all fields could be selected from a list view in eclipse. Compared to the previous version, in which the user needed to create a new field for each Template, this made re-use of Templates faster.

But a problem for the list view was that, while it worked fine for 10-20 fields, it would become less viewable once the number of fields got to 200 and above, and the company at that time was using around 270 fields. This was a limitation in the tool. A workaround for the problem was, however, to use the search feature of the Eclipse list view. That combined with the introduction of naming conventions of tools would reduce the problem.

The default setting for the Eclipse editor does not allow for setting properties for multiple nodes at the same time. This could be a serious usability problem, but was not addressed in the prototype version. A question was posted on the eclipse forums for a solution and it was answered within days, but the solution was not trivial to implement and we failed to solve the issue during the time of the project. But to summarize, this issue is known in the Eclipse modelling community and there is a plug-in fix available for the Eclipse-crafty.

There was a clear requirement from the company that they needed multilingual support. They currently used English and Swedish but since they were expanding in Northern Europe, they were going to need support for additional languages. This requirement was however not addressed in the OTM.

The requirement to automatically spread out all files into the right of the system was demonstrated by the way the OTM generated PHP and SQL-files. With the additional batch-script the project team got a sense of what a future system could look like, and they were happy with the result. Of course, since the OTM was only a prototype there were many features missing from a ready-to-be-released version. Some of these features were for instance indexing, which was not present at all in the OTM. However, it is something that could be implemented using enums.

4.9 Diagnostic evaluation

In the following text, the result of each scenario will be described in detail and usability problems will be discussed. The scenarios, that were tested with a developer from KVD, can be found in appendix A.1. They will, however, be given by name here for easy reference: Scenario 1 – Object template from scratch, Scenario 2 – Object template with similar fields and Scenario 3 – SQL settings.

The user started exploring the properties right away. It was apparent that he was used to the eclipse environment; there was no need for a general introduction to Eclipse. When naming elements that had "-" characters he naturally removed them. We believe this was because he was used to SQL-syntax and understood that that is an illegal character for SQL. Once the test had started it became clear that the instructions were lacking – there was no explanation about the concept of Templates. This was blurry to the user.

Without being instructed to do so the user created subtemplates in order simulate inheritance. The user was instructed to create a Minidumper based on the earlier created Dumper, and thought that this was something that should be done using inheritance.

There was confusion about the types. A reason for this was because the instructions for adding types were located at the end of the manual. Because of this the SQL code generated contained errors. The user managed to create primary keys by following the manual.

After all files had been generated the user discovered that there was a bug in the system. All PHP-files were identical except for the array. This error could be traced to a copy and paste error and was addressed in the post analysis.

There were additional properties that the user requested. Among these was the ability to set indexes on fields. In future releases this is something that could be addressed using enums in a similar fashion that Tables are selected for primary keys.

Scenario 1 was completed in 4 min. The user got confused about Allfields. The user did not grasp that concept from the instructions given. Here the user received help in order to finish the scenario. Also there was some confusion with siblings and children. It was not apparent to the user that he needed to right click and select node under which he wanted to create a child. Also it was not obvious that a sibling creates a parallel element to the selected node. The user created a group under the wrong node but solved that problem by moving (clicking and dragging) the node to the right place. The error was that the user created a Template group instead of a Template for Dumper. The user needed instructions to correct this error because the instructions given in the manual was not enough.

Scenario 2 was completed in 2 min 25 s. Here, inheritance i.e. a sub-template, was created under a template (Dumper/Minidumper) and used to complete the task. It was not the original solution to the scenario but was acceptable as an alternative solution. A source to the idea behind inheritance could have been that the instructions stated that a Dumper should be "based on" a Minidumper. This led the user to interpret that Minidumper should inherit from Dumper but there was not supposed to be an actual inheritance because Minidumper did not have all fields that Dumper had. Therefore, the solution was erroneous; a better solution would have been to create a new Template for Minidumper, instead of a sub-template to Dumper.

Scenario 3 took 13 min 40 s to complete. This was the scenario that took the longest time to complete but it was also the test that required the most configurations. The user was questioning why Object number was to be used as a primary key but accepted it as a when explained to that it was used as such in the test environment. The user tried to generate code in this scenario (which was not part of the scenario) and that added to the resulting test time. The user had problems navigating between the folders in the Eclipse workspace, especially the Acceleo-files that were located in a separate package in the Acceleo project. The user tried to generate files before having set any types and because of that the dummy page displayed errors. The user got instructions that he needed to set types and was directed to the help section covering that area. After leaving the input.xmi-file to view the generated files, the user had problems navigating back to the input-model and needed instructions to find it. There was confusion about FText type, but the user was told that it was a long string which was enough for him to comprehend the concept. Also, the user found it odd that there was no bool. Primary keys were created without errors.

4.10 Coding and Strength and weaknesses table

The interview transcript from the diagnostic evaluation was coded into categories (such as productivity and maintainability). The transcript was colour coded with a colour representing a category. The aim was to find common themes and issues and an in depth coding analysis is found on page 36 chapter 5.6.

	Strength	**Impact**
Features	1. Creating sub-templates	Mid
Familiarity	2. Eclipse environment	High
Learnability	3. Primary keys were created without errors	High
Tool support	4. Drag and drop	Low

Table 4.1: Strengths gathered from the diagnostic evaluation of the OTM.

	Weakness	**Impact**	**Workload**
Learnability	1. Concept of Templates	Low	High
	2. Confusion about types.	Mid	Mid
	3. Confusion about All fields	Low	Low
	4. Created a Template group instead of a Template	Low	High
	5. Lots of configurations needed for sql	High	High
	6. Difficult navigating between folders	Low	High
	7. Different structure in diff. technologies	Mid	High
Authenticity	8. Object number should not be used as a primary key	Low	Low
Features	9. No boolean types	Mid	Mid
	10. No indexes	Mid	Mid
Vulnerability	11. SQL generated contained errors	High	High
Familiarity	12. FText type unfamiliar	Low	Low
Expressiveness	13. Created subtemplates in order to simulate Inheritance	Low	High
Tool support	14. Right click to select node	Low	High
	15. Difficulties navigating between models and files	Low	High

Table 4.2: Weaknesses gathered from the diagnostic evaluation of the OTM.

Tables 4.1 and 4.2 shows the, from the diagnostic evaluation, encountered strengths and weaknesses. Impact means the issue's estimated effect on the quality of the finished application. Workload is an estimate of how difficult we believe it would be to address the issue.

The result of the qualitative analysis was a total of 15 weaknesses compared to 4 strengths. These weaknesses would need to be addressed for another iteration of the OTM. Further discussion about each specific weakness will reveal its seriousness and whether it could be fixed or not.

4.11 Focus groups evaluation

This method led to evaluation outcomes that were used to draw conclusions. In practice, the focus groups resulted in written notes concerning feedback of the prototype (for example, the prototype tree structure needs to have a separate parent node for “groups”), new requirements (for example, multilingual support) or technical knowledge (for example, information about KVD’s database structure from the lead programmer).

Chapter 5

Discussion

Based on the single case study performed in this thesis, general conclusions cannot be made. We would still like to discuss some issues and lessons learned that we believe could be common for SMEs that are thinking about implementing an MDE approach. Based on these assumptions we find it reasonable to believe that the obstacles present at KVD could be present in other SMEs:

- No previous experience of MDE.
- Works with legacy system that has maintenance issues.
- Small IT team (around 10 employees)
- Web based E-commerce architecture.
- Works according to a MVC pattern (figure 5.2).

5.1 Why KVD is a representative company for SMEs

They are working according to MVC. This is a very common development pattern in software engineering. They are also using object-oriented PHP which is a widespread web programming language. Their business model of online auctions is similar to other businesses such as Tradera, Ebay and uBid.

5.2 What productivity means

An area commonly claimed to be improved by MDE is productivity. But what is actually meant by productivity? Because MDE is a rather extensive term, it is important to specify just what parameters are improved when talking about productivity. There is the possibility of improving productivity in a process sense; through raised abstractions faster implementations due to models and code generation. There is also the matter of improved productivity in the sense that the resulting DSL, i.e. the application developed by an MDE approach, makes tasks easier to perform.

It is difficult to separate these two situations. We would like to make the distinction of a process improvement and product improvement.

The process improvement is perhaps the most interesting area of productivity. Can MDE shorten the time it takes to develop software? As an example, we could look at a banking system. How much quicker would it be to develop this system using an MDE approach compared to a traditional development method? In this case, productivity concerns the efficiency of the team to deliver the finished software.

Product improvement, on the other hand, is the finished software product and to what extent that product improves tasks at a company. Following the banking example, the product would be the finished banking system. Productivity in this case refers to how much faster clerks can handle customer services compared to the banks old way of working (perhaps they were using a completely analogue system, to make an extreme example). In this case we could expect rather drastic productivity improvements. We suspect that these are the productivity improvements that DSLs often refer to, when claiming that productivity was increased with 500-1000%.

We would like to argue, however, that such drastic productivity improvements are not unique to MDE. Take the banking example; the bank would be likely to expect a drastic product productivity improvement regardless of what software development methodology was used to deliver their software. Because these product productivity improvements are not unique to MDE, we believe that they therefore cannot be used as an argument to motivate an MDE implementation.

In this thesis, we examine both process and product improvement but would like to make clear that it is the process improvement that is the deciding factor when it comes to adopting MDE. The product improvement is still necessary to examine, however, because if there is no product improvement then most companies would not likely want to use the MDE approach. If there is a possible product improvement then that is not enough to justify an MDE adoption, because a product improvement would be likely with another software development methodology as well.

Mohagheghi and Dehlen[2] talks mainly about process improvements, how much quicker is an application to develop with MDE, although they argue that these measurements are difficult to make. To make a fair comparison one would have to develop two applications, one with the current process and one with MDE.

Kelly and Tolvanen[10] writes that a development task that earlier took days, using code concepts, can be done in minutes with DSM. We would argue that this is an unfair comparison. Of course a DSL will be more efficient to a third generation programming language. The DSL is tailored to the domain and should be more efficient, the 3rd generation programming language can be used in ANY domain. Is it reasonable two make a comparison between the two at all? We would like to suggest that a fairer way of comparing would be to take the DSL and put it up against an application developed by a 3rd generation language. As an example, compare a DSL. It would be interesting to see if the productivity gains would still be as drastic. For instance they have an example of a mobile phone application. Instead of comparing the DSL with a 3rd generation language, the application should be compared with an editor created with a 3rd generation language. This editor should contain the same concepts as the DSL.

With this reasoning in mind we further suggest that it is the process that is the interesting area of improvement for MDE. Product improvements can be interesting to look at if the comparison is fair, i.e. the two parts being compared share a similar domain. Comparing a DSL to a 3rd generation language is like comparing a samurai sword to a Swiss army knife. The former is a highly specialized tool and the latter a multipurpose tool.

Since we are talking about two different things, it would perhaps be appropriate to make a silly metaphor – is the trip shorter or is the destination more pleasant? The metaphor as stated is not mutually exclusive, but the trip should be shorter and the destination more pleasant to be able to recommend MDE.

5.3 Has MDE missed the boat?

In October 2011 a presentation with the title “Why did MDE miss the boat?” appeared online. The presentation was unfortunately not available at the time of writing, but summaries were and the lively discussions followed in the modelling community[24]. From these summaries and discussions the following text was created. We reasoned that it would be of interest to the thesis to bring up these current reflections on MDE.

The main conclusion from this presentation is that MDE in terms of adoption in industry and application on large scale projects has reached a halt. Bezivin argues that in that sense MDE has failed and the promises made 10-20 years ago has not been fulfilled.

One of the main arguments was that MDE has not had a killer app, an application that fully showcases the capabilities of a modelling approach. As a comparison, the first object-oriented “killer app” by Tom Love was mentioned as a landmark for the widespread adoption of object-oriented programming. He created a 220 line object oriented Smalltalk-80 prototype, that was estimated to have needed 10000 lines of FORTRAN code[25]. A similar killer app would help to bring out MDE in the light and promote widespread adoption.

There are too many definitions and theories (MDE, MDSD, MDA, MDD). We find it difficult at times to distinguish between these definitions and we find it reasonable to believe that MDE-outsiders are confused by the numerous acronyms.

UML is often used as a foundation for MDE but UML is a loosely defined language that is too complex and too big. We did not experience this as a major problem in the OTM mainly because Ecore uses its own language based on the MOF and not UML.

Drastic conclusions should not be drawn from a single unpublished presentation, but we find the discussion that it spurs interesting. Many MDE practitioners admit that they recognize the criticism from the presentation and there is an unscientific but unmistakable sense of resignation in these comments. Tolvanen[10] adds to the debate and agrees that “the idea of models transformations between CIM, PIM, PSM and code

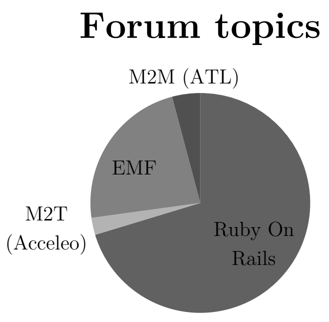


Figure 5.1: Topics distribution per technology.

has not happened. ” while he does not agree that MDE has failed.

Developers & MDE

After the presentation of MDE fundamentals, the reaction from the developers was curious albeit somewhat questioning towards how to actually use it. In that sense MDE is very different from a new attractive programming language. MDE does not appeal to the developers imagination in the same way, the concept is more abstract and does not spur the same “i-got-to-try-that-when-I-get-home” reaction that perhaps a Ruby-on-rails presentation. If we look at the number of topics started on the Eclipse modelling forums and the railsforum.com we can see that, as of the 3rd of November 2011, 1256 topics were started related to EMF, 1373 related to M2T, 2255 related to M2M. On the railsforum.com 38340 topics in total were related to Ruby on Rails development (see figure 5.1). This is an unfair comparison in many ways (perhaps there are other forums having more traffic) but what we like to suggest with this simple example is that MDE is not receiving a fan base online in the same way as Ruby on Rails. But parameters as number of members that the forums attract and number of topics are interesting to prove our point. A more active forum signals a more active technology.

We suggest that MDE is not developer-centric. MDE appeals more to those that want to achieve engineering discipline. Is that necessary a bad thing? No, but MDE lack of appeal to developers is a major liability to its widespread adoption.

5.4 Does MDE improve understandability?

An interesting discovery made, was that of understandability. According to the IT manager at KVD, the model would not be helpful in discussions with stakeholders outside of IT. He himself had troubles understanding the class diagrams. The IT lead, on the other hand found the model understandable and helpful. They both agreed, however, that a model could help improve the understandability of a system among IT developers at the company. This is contrary to what most other papers about MDE report concerning understandability[14][2].

There may be a potential issue here. MDE might not be exposed to many other stakeholders than developers or people with modelling know how. If that is the case then the benefit of improved understandability among stakeholders might not be a benefit at all. But this is just a speculation based on the experience from this case study; it would require more investigation before any conclusions could be drawn.

5.5 Advantages and disadvantages of MDE at KVD

Advantages

Platform independence would be achieved by keeping templates in a platform independent model. In other words, the object template domain would be separate from the technical platform. This would support future technology migrations.

Disadvantages

There would have to be MDE training of the project members. This comes with costs and time delays. A 2-day Metaedit Proof of concept workshop for 5 attendees costs 3000€[26]. A 2-day eclipse modelling course costs 1200€ per person[27].

5.6 Coding analysis of diagnostic evaluation

The following text refers to table 4.1 and 4.2 in the results chapter.

Strengths

S1: Although, not fully implemented in the OTM prototype, subtemplates are represented in the editor and were used by the user during the diagnostic evaluation. While it is a strength that it is possible to create subtemplates in the editor, this, however, led to some misunderstandings which are described in W13.

S2: The fact that the intended user group is familiar with the Eclipse environment is an important strength in favour of the OTM. The advantages of this could be observed in the diagnostic evaluation in means that the user got started quickly and finished the scenarios without much help from the instructor.

S3: The user created primary keys without any errors. This was an unexpected but positive result because we had estimated this part to be one of the more complex and unintuitive, since primary keys require creating a separate node that points to a table.

S4: Drag and drop interfaces are common place today, and thus it is appropriate that Eclipse supports it. Template, sub-templates and Template groups can be rearranged and duplicated by drag and drop actions. The user benefited from this feature in scenario 1 when a node, misplaced by the user, needed to be moved.

Weaknesses

W1 is a learnability issue that will reduce with time. The more the user gets accustomed to the tool, the more natural the concept of Templates will become. Perhaps this issue could be overcome with a WYSIWYG-editor, i.e. if the editor was further developed, for instance if the user could see the Template being edited instead of an abstract tree view, it would be easier for the user to understand the concept of Templates.

W2: Type definition is not intuitive and because of that the user had problems setting types. Perhaps this could be addressed by having types as a reference (similar to All fields) instead of containment. That way a type would be represented in the properties view instead of as an individual node. The change would require modifying the metamodel and updating transformations.

W3: The user got confused by template group All fields. The way All fields work is as a Template group that points to the fields that should be in a Template. We see this as having low impact on the OTM and expect the issue to decrease with increased experience of the tool.

W4: The user created a template group instead of a template which lead to an erroneous result. We could argue that this was a result of unclear instructions but on the other hand it could be seen as a usability problem. Well designed interfaces today does not need documentation to explain how they should be used, they are often intuitive enough for the user to start using them right away. If this issue were to be solved in that sense the editor would need to be redesigned, which we estimate to have a high workload. Finally, it was out of the scope for this prototype to have that level of intuitiveness, so the impact for this issue was estimated to be low.

W5: SQL configurations needed many parameters to be set. This was an expected issue and the user needed help finishing scenario 3. A way to overcome the issue would be to somehow transfer some of the manual settings to the transformations. For instance, in ATL there could be a type check on the input of a field that would automatically set an appropriate type. This, however, is not possible in the current version of the OTM because input is handled outside in another web based system. Also, other SQL settings such as "auto_increment" and "default" are difficult to automatically set in ATL because there are no general rules to when such settings are set and when they are not. Alas, it will be difficult to remove the manual setting of SQL parameters and we estimate a high workload (if possible) in fixing the issue. We also believe that this issue is a major disadvantage for the OTM, because it is doing things that can be done in a more controlled and dedicated way with other tools, namely SQL GUIs.

W6: Difficult navigating between project folders in Eclipse. As the OTM looks today it consists of a workspace with an EMF project, an ATL project, and an Acceleo project. This led to the user having problems navigating between the three projects. A possible fix for this would be to distribute the OTM as a coherent plug-in for Eclipse with folders organized for models, transformations, and output.

W7: There were different structures in different technologies (ATL, Acceleo and EMF). This is first of all a problem on the developer side but the results from the diagnostic evaluation shows that it also has an impact on the user, who had problems navigating in the Acceleo project. For possible fix, see W6.

W8: Object number as primary key was not an authentic representation of the company's current system. The reason that Object number was used as primary key was to make the scenario simpler to complete. Because the test was mainly concerned with usability aspects, we reasoned that this was a reasonable simplification but since it lead to misunderstandings, it would be relevant to create future diagnostic evaluations in a more authentic (i.e. make sure concepts in a scenario are the same as the ones in existing systems) way. A fix to the problem would be to create another primary key, for instance objListaNr, that is authentic to the KVD domain.

W9: The user requested boolean support. This is a valid request, however, in the target domain of the company there was no Boolean types in the SQL tables. Therefore Boolean types were left out of the implementation. The workload to implement this feature would be of medium effort, requiring the adding of a Boolean class in the metamodel and the creation of related transformations.

W10: There is no support for indexing in the OTM, which was something the user requested. Indexing was never discussed during the requirements meetings but should be considered for future versions of the OTM. A possible implementation of indexes would be using an enum in a similar way that primary key tables are selected.

W11: The Acceleo generated SQL contained errors which lead to the dummy-page displaying error messages. This is a serious issue and a possible vulnerability threat to the system. As it is, there is no error checking or validation on the input side. We thought about using OCL constraints in order to overcome the issue but we decided to leave that technology out of the prototype in order to be able to finish on time, but constraints could be a solution in coming iterations. For instance, constraints could be written to force a field to have a type or to assure that a primary key must exist for every table.

W12: The user was unfamiliar with a type used by the OTM. The FText type, which represents the SQL type Text, was not recognized by the user but did not cause much confusion after it had been described as a long String. We see this issue as having low impact on the OTM. It could possibly be addressed by naming OTM types exactly as SQL types, but we believe that it is beneficial to distinguish OTM types from other domains (by starting each type name with an "F", for field).

W13: The user created a subtemplate to simulate inheritance, in a case where it was inappropriate. This resulted in an unwanted result, because the subtemplate inherited fields that it was not supposed to have. To make this clearer to the user perhaps what fields belonging to a subtemplate could be visualized in more intuitive way, for instance by colour-coding subtemplates and their associated fields. There is, however, no tool support for such a feature so the expected workload would be high. Also, an actual inheritance feature is yet to be implemented. As it is now, inheritance is merely represented in the editor (in the form of subtemplates) – the generated files are unaffected by inheritance.

W14: The user had problems right clicking on the node for which to create a child node but we expect this problem to decrease with increased experience of the OTM. A high workload is estimated in order to overcome the issue because it would mean creating a custom editor or changing the Eclipse source code.

W15: After leaving the inputModel.xmi view, the user had problems navigating back to it. This could possibly be addressed the same way as for W6.

It is possible that this form of diagnostic evaluation is more likely to find flaws than it is to find positive aspects.

5.7 Validity of findings and case study research

A disadvantage of case studies is that there is a lack of comparative sources. When only one source is investigated, as the case was for our study, we may not draw any general conclusions. For instance, if the source is a company then we cannot be certain that the company is representative for all companies of its field. This is one reason why case studies are sometimes criticized for being of less value than studies that uses multiple sources.

This issue can, however, be addressed by applying proper research methods [7]. If we use research methods such as interviews or diagnostic evaluations, then we can assure, given that the methods are carried out in a scientific manner, a certain validity of the data gathered and reduce the risk of biased opinions.

Benbasat et al. [28] mentions that a case study is information gathering from few entities and a lack of experimental control and that description is appropriate for this thesis. The few entities we had to gather information from were the staff at KVD and there were few possibilities for experimental control.

A case study often incorporates research methods such as surveys and action research. In our case, the research method of literature study was much used and also an archival analysis of KVDs current process was made at the start of the project. The archival analysis meant going through the documentation that KVD had available. The documentation consisted of internal documents and the information available on their web page.

When conducting case studies, trade-offs between realism and level of control must be balanced. It was necessary to reduce the level of realism, i.e. leave out features such as internationalization, when we created the prototype tool because the complexity of KVD's system would not allow a full prototype to be developed in the limited time-frame.

Data was collected in what Lethbridge et al. calls the 1st degree [29], meaning that we interacted directly with the employees of the company and gathered the data through meetings, interviews and observations. There was also data gathered in the third degree meaning that KVD's existing system was analysed based on existing documentation, as described above.

5.8 Hutchinson's table

From Hutchinson's table (2.1) we have three areas of interest; productivity, portability and maintenance. For each of these there are both positive and negative consequences. In the following section we will discuss how these areas of interest relate to our case study.

5.8.1 Productivity

Time to develop code The amount of code generation implemented in the prototype was the code for three PHP-files and a SQL-script. Here we do have increased productivity. The code generation took around 40 s to be finished. We do not have any data on how long it took to write those files manually but we assume that it was greater than 40 s.

On the negative side we have the time it took to implement the transformations. This took approximately 25% (30 days) of the development time and we assume that the creation of the transformation took longer time than the manual writing of the original files.

This negative effect reduces over time, the more often the files are generated. These files were, however, seldom changed and because of that the benefits of code generation in this case were small.

Time to test code There was no explicit testing as part of the case study, but we still made the following observations. Hutchinson et al. suggest that the amount of silly mistakes reduces through code generation. In our experience, silly mistakes were still prevalent in the code even though code generation was used. These mistakes were, however in the transformations and once traced and fixed it was only a matter of regenerating the code to get rid of them. The models needed to be tested through proof of concept, i.e. we had to find out if the model could generate the kind of code that existed in the exemplars.

ROI on modelling effort We did experience the positive effects; it was easy to see the whole picture and to come up with creative solutions. The negative effect of model paralysis was on the other hand also experienced, especially after a first iteration of the system was up and running. We experienced an unwillingness to change the model that was working, in order to improve it. If this is an issue related to modelling or development in general cannot be said.

5.8.2 Portability

Time to migrate to a new platform. In the short development time (roughly four months) that existed we still had the opportunity to experience this parameter. The initial system had only one model that generated code directly. Eventually, another PSM was added

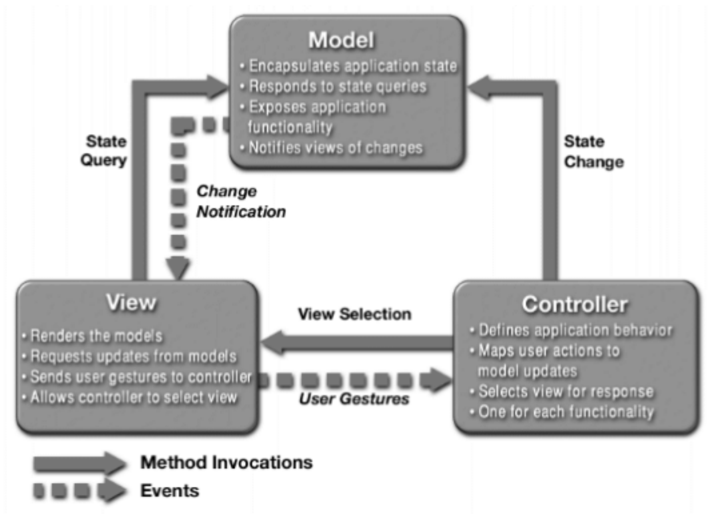

Model
• Encapsulates application state
• Responds to state queries
• Exposes application functionality
• Notifies views of changes
State Query
State Change
Change Notification
View
• Renders the models
• Requests updates from models
• Sends user gestures to controller
• Allows controller to select view
View Selection
User Gestures
Controller
• Defines application behavior
• Maps user actions to model updates
• Selects view for response
• One for each functionality
Method Invocations
Events

Figure 5.2: Figure of a MVC architecture [30].

for SQL and a PIM was created that only handled the domain of KVD. This was a small portability problem. We had to move some of the SQL properties to the PSM from the PIM, but a large part of the structure (concepts such as Fields and Groups) of the initial model could be preserved showcasing some of the benefits of MDE portability. Unfortunately, PIM was not as platform independent as preferred: it was cluttered with SQL-syntax. The reason to do so was because we wanted to have a lot of control over the SQL in the editor. The editor was created from the PIM. Removing the SQL-syntax from the PIM would result in an editor that was not able to set properties such as primary key and auto_increment.

5.8.3 Maintenance

Unfortunately we never experienced the prototype's effect on maintenance due to the limited time frame of the thesis. The prototype would need to be used in production for a period of time before such data could be gathered.

5.9 Reflections on the impact of the study at KVD

A main aspect of MDE that remain critical is that you as a developer spend a lot of time thinking is this feasible/possible to do instead of just doing it. But is this just a matter of inexperience with modelling or is it really a shortcoming?

During everyday meetings in different workplaces whiteboards are often used to collect and share ideas. The sketches drawn on these whiteboards are often abstract and can be incomprehensible for anyone who did not take part of the meeting. However, it is not the whiteboard sketch that matters, it is the collaborative work that led up to it that matters, and that is valid even for MDE. During the development of the OTM the very fruitful discussions that were triggered by the method itself was perhaps the greatest achievement. It cannot be said on the other hand, that by using a non-model driven approach these discussion would have been lacking.

The developers did not find that using models (in terms of Ecore metamodels) would ease understanding among stakeholders. But as a mean of improving understandability internally, among developers, they (KVD) were positive of the effect.

Chapter 6

Related Work

6.1 Other applications of MDE to e-commerce systems.

WebML is an organization that has developed an UML-like language for development of data intensive web pages. They have successfully applied model-driven techniques in web development but are mainly concerned with the construction productivity of the web page, not the resulting process. The annual Model-Driven Web Engineering Workshop has been running for seven years and features speeches and workshops related to web modelling. At the 2011 event Eickhoff et al. [31] wrote an article on how to develop enterprise web applications proposing a model-driven method. In that research paper the focus was also on construction rather than process.

Stahl and Völter writes about a Web Application project, developed using MDE(or rather MDSD, as their definition stands).Infrastructure code requires a lot of time but is generally repetitive and not very complicated, i.e. it could be generated. 60% to 70% of modern e-business applications consists of infrastructure code. If some technical reference implementation has been established the rest of the development will be mostly copy and paste programming, says Stahl and Völter. This leads to a generative software architecture, which would require a domain model of the application. Reverse engineering results in a model that is as abstract as the code, it is hard to create a PIM this way. Using the PIM, PSM approach there is a risk of inconsistency problems because the manual change of an intermediate model might not change automatically in a higher abstraction model. They avoid round-trip engineering instead they use forward engineering – changes must always be done to the model. They use a PIM that is as abstract as possible but still transformable into source code. No 100% code generation. While not impossible, 60% to 80% code generation is more reasonable for AC-MDSD. In the web application project they create a car-sharing system based on the J2EE framework. The architectural concepts consisted of presentation, activityController, ValueObject among other. The generator output was JSP,Struts-Config and Entity Beans.

Chapter 7

Conclusions and Future works

7.1 Conclusions

7.1.1 How does the introduction of MDE alter the object template creation and maintenance process?

The introduction of MDE alters the process in the sense that a change in an object template now only needs to be done in one place compared to the earlier nine. Templates are created using a tree structure in Eclipse and not with the existing Filemaker interface.

7.1.2 What performance gain, in record definition and maintenance, can be achieved by model transformation in e-commerce systems?

This question remains unanswered since the only test carried out was on the prototype which was not comparable to the old process. No full tests were possible to be made and therefore no performance gains could be measured.

7.1.3 Can MDE be recommended as a solution to KVDs current problem with object template maintenance?

The development of the prototype shows that KVD's current process could be improved by using an MDE approach. That does not, however, mean that the change to MDE is worth the risk. In the case of KVD it was too much of a risk to be recommended.

Our final recommendation to KVD was for them to not adopt an MDE approach. The main argument for this conclusion was that they could solve their problem with their current work techniques; MDE was not necessary to solve their problem. This argument follows the same thinking as the lessons learned presented by Hutchinson et. al:

> [...] if the process already employed by a company is working 'sufficiently well', it will be very difficult to introduce as radical a change to that process

> as introducing MDE. However, if the existing process is itself a significant risk to a project, or if that process involves the developers in difficult, time-consuming and uninspiring activities, then process change in the form of MDE adoption will be more readily embraced.[4].

During our stay at the company, we did not find any indications that their existing process[1] would be a risk to their projects. Quite the opposite; their SCRUM-based project process had just recently worked well for their overhaul of the web page and was thus tried and proven. It could be used, the prototype shows how it could be implemented, but the actual benefits associated with that is hard to estimate and there are many risks concerning the inexperience with MDE at the company.

7.2 Future works

Questions remain unanswered: would an MDE implementation go smoother with another tool (Metaedit+, Microsoft DSL tools) compared with the Eclipse Modeling Tools? Was the tool we used in itself a reason that led to the fact that an MDE solution could not be recommended? We have seen research on MDE's adoption in industry and through the sheer number of research articles available on the topic we can confirm that there is an adoption in academia as well. But what would be interesting to see is research on how popular MDE is outside of these two areas; how big is the user community online? Does anyone outside industry and academia use MDE? Does MDE appeal to software development enthusiasts, whose interest is necessary to build a supportive community?

One aspect that interested KVD and that they ensured that they would implement in coming projects was models. They were referring to the traditional idea of model-based development but did not fully embrace the MDE approach of developing DSLs and code generation. Model-based development was already, to an extent, part of their development process. Many of their projects started on the whiteboard where simple models were drawn. These models were used to come to agreements on new ideas and to illustrate points that were hard to express verbally.

The transition is, however, still long to a fully realized MDE approach. Future MDE tools should investigate the developments process used at businesses today. Since many already worked with some form of models it is only a matter of getting the model into the project as a primary artefact. A useful tool would be a whiteboard that transforms drawings into an initial Ecore-model. A tool is needed that bridges the gap between simple sketch models and detailed system models. There have been attempts at developing such tools and we believe that is a step in the right direction. The question to ask is, if we are using a model in the development why not generate code from it? Well, it is hard to generate code from a whiteboard, therefore a tool is needed.

[1]We would like to clarify that existing process in this case means the process of working at KVD (MVC, object-oriented, SCRUM) not their process of object template management which indeed was a risk, as explained in section 4.3.

Bibliography

[1] A. MacDonald, D. Russell, and B. Atchison, “Model-driven development within a legacy system: An industry experience report,” in *Proceedings of the 2005 Australian conference on Software Engineering*, (Washington, DC, USA), pp. 14–22, IEEE Computer Society, 2005.

[2] P. Mohagheghi and V. Dehlen, “Where is the proof? - a review of experiences from applying MDE in industry,” in *Proceedings of the 4th European conference on Model Driven Architecture: Foundations and Applications*, ECMDA-FA ’08, (Berlin, Heidelberg), pp. 432–443, Springer-Verlag, 2008.

[3] M. Staron, “Adopting Model Driven Software Development in industry – a case study at two companies,” in *Model Driven Engineering Languages and Systems* (O. Nierstrasz, J. Whittle, D. Harel, and G. Reggio, eds.), vol. 4199 of *Lecture Notes in Computer Science*, pp. 57–72, Springer Berlin / Heidelberg, 2006. 10.1007/11880240_5.

[4] J. Hutchinson, J. Whittle, M. Rouncefield, and S. Kristoffersen, “Empirical assessment of MDE in industry,” in *Proceeding of the 33rd international conference on Software engineering*, ICSE ’11, (New York, NY, USA), pp. 471–480, ACM, 2011.

[5] P. Baker, S. Loh, and F. Weil, “Model-driven engineering in a large industrial context — Motorola case study,” in *Model Driven Engineering Languages and Systems* (L. Briand and C. Williams, eds.), vol. 3713 of *Lecture Notes in Computer Science*, pp. 476–491, Springer Berlin / Heidelberg, 2005. 10.1007/11557432_36.

[6] I. M. Holme and B. K. Solvang, *Forskningsmetodik - Om kvalitativa och kvantitativa metoder*. Lund, Sweden: Studentlitteratur, 2nd ed., 1997.

[7] P. Runeson and M. Höst, “Guidelines for conducting and reporting case study research in software engineering,” *Empirical Software Engineering*, vol. 14, pp. 131–164, 2009. 10.1007/s10664-008-9102-8.

[8] E. M. Chocano, "A comparative study of iterative prototyping vs. waterfall process applied to small and medium sized software projects," Master's thesis, Massachusetts Institute of Technology, 2004.

[9] N. Bevan, "UsabilityNet." Retrieved October 25, 2011, from UsabilityNet: Diagnostic Evaluation `http://usabilitynet.org/tools/diagnostic`.

[10] S. Kelly and J.-P. Tolvanen, *Domain-Specific Modeling: Enabling Full Code Generation.* Hoboken, NJ, USA: John Wiley & Sons, Ltd, 2008.

[11] T. Gherbi, D. Meslati, and I. Borne, "MDE between promises and challenges," in *Computer Modelling and Simulation, 2009. UKSIM '09. 11th International Conference on*, pp. 152 –155, march 2009.

[12] R. I. Bull and J.-M. Favre, "Visualization in the context of model driven engineering," in *MDDAUI 05, Model Driven Development of Advanced User Interfaces 2005, Proceedings of the MoDELS 05 Workshop on Model Driven Development of Advanced User Interfaces, Montego Bay, Jamaica, October 2, 2005* (A. Pleuß, J. V. den Bergh, H. Hußmann, and S. Sauer, eds.), vol. 159 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2005.

[13] M. R. Blackburn, "What's Model Driven Engineering(MDE) and how can it impact process, people, tools and productivity," tech. rep., Systems and Software Consortium, Inc., 2008.

[14] J. Hutchinson, M. Rouncefield, and J. Whittle, "Model-driven engineering practices in industry," in *Proceeding of the 33rd international conference on Software engineering*, ICSE '11, (New York, NY, USA), pp. 633–642, ACM, 2011.

[15] D. Ameller, "Considering non-functional requirements in model-driven engineering," Master's thesis, Universitat Politècnica de Catalunya, 2009.

[16] OMG. Retrieved November 9, 2011, from `http://www.omg.org/`.

[17] E. Merks, "The unbearable stupidity of modeling." Retrieved October 25, 2011, from `http://www.slideshare.net/peterfriese/the-unbearable-stupidity-of-modeling`. Presentation.

[18] T. Stahl and M. Völter, *Model-Driven Software Development: Technology, Engineering, Management.* Hoboken, NJ, USA: John Wiley & Sons, Ltd, 2006.

[19] J. Nielsen, "Iterative user interface design." Retrieved October 25, 2011, from `http://www.useit.com/papers/iterative_design/`, 1993.

[20] A. Cockburn, "Using both incremental and iterative development." Retrieved April 25, 2012, from alistair.cockburn.us `http://alistair.cockburn.us/Using+both+incremental+and+iterative+development`.

[21] J. M. Corbin and A. L. Strauss, *Basics of qualitative research.* Sage Publ., 3 ed., 2008.

[22] I. Jacobson and P.-W. Ng, *Aspect-Oriented Software Development with Use Cases (Addison-Wesley Object Technology Series).* Addison-Wesley Professional, 2004.

[23] A. Ant. Retrieved October 27, 2011, from `http://ant.apache.org/`.

[24] J.-J. Dubray and J. Bezivin, "Why did MDE miss the boat?." Retrieved November 02, 2011, from `http://www.infoq.com/news/2011/10/mde-missed-the-boat`, 2011.

[25] T. Love, *Object Lessons: Lessons Learned in Object-Oriented Development Projects.* Cambridge University Press, 1997.

[26] M. Consulting and Training. Retrieved November 01, 2011, from `http://www.metacase.com/papers/Service_Descriptions_and_Pricing.pdf`.

[27] E. modeling training. Retrieved November 01, 2011, from `http://eclipsesource.com/fileadmin/doc/2009_course_reg/Anmeldung_EMF2Tage.pdf`.

[28] I. Benbasat, David, and M. Mead, "The Case Research Strategy in Studies of Information Systems," *MIS Quarterly*, vol. 11, pp. 369–386, Sept. 1987.

[29] T. C. Lethbridge, S. E. Sim, and J. Singer, "Studying software engineers: Data collection techniques for software field studies," *Empirical Software Engineering*, vol. 10, pp. 311–341, 2005.

[30] Kohana. Retrieved November 01, 2011, from `http://dev.kohanaframework.org/projects/kohana2/wiki/Kohana101`.

[31] C. Eickhoff, N. Geiger, M. Hahn, and A. Zündorf, "Developing enterprise web applications using the story driven modeling approach," (Kassel, Germany), University of Kassel, 2011.

Chapter A

Appendix

A.1 Scenarios

Scenario 1 – Object template from scratch

You have been asked to create a new object template for the object type Dumper. Since a Dumper is very different from other object templates, no existing template can be re-used and therefore the "Dumper"-template must be created from scratch. A Dumper consists of the following:

- Object number
- Hp
- Year
- CE mark
- Loading cap.
- Color
- 1st wheelpair
- 2nd wheelpair
- 3rd wheelpair

Create a new Template that contains the above fields. For simplicity put them all in a template group called "All fields". Refer to the Help section for guidance.

Scenario 2 – Object template with similar fields

You are asked to create a new template for Minidumpers. It will be based on the Dumper template but will have some modifications. The Minidumper consists of the following fields:

- Object number
- Hp
- Year
- CE mark
- Color
- 1st wheelpair
- 2nd wheelpair
- 3rd wheelpair
- AC
- Weight

Hint! The Minidumper should reuse common fields from the Dumper template.

Scenario 3 – SQL settings

The goal of this test is to generate correct SQL syntax from the model. Use the fields created in the previous scenarios and set a suitable type for each of them. Use the default properties for the types. Use Object number as a primary key and set its properties Presentation, List, and Translate to true. Also set its type to FInteger. Create three primary keys, one for objpres, one for objlista and one for objovers. Set the rest of its properties as instructed in Set primary key in the Help section. When the SQL-properties have been set, generate code from the model. The test is successful if at least one field is generated for objpres, objovers and objlista and each table has Object number as primary key.

A.2 Vision document

OTM - Object template manager
Vision

Version 1.1

Revision History

Date	Version	Description	Author
2011-08-15	1.0	Document created.	Emil Selmeryd
2011-08-31	1.1	Minor corrections and extended introduction. Added SQL metamodel. Added new process figures in 2.3 and 3.3. New product features. Added chapter 7.	Emil Selmeryd

OTM - Object template manager	Version: 1.1
Vision	Date: 2011-08-31
Draft	

Table of Contents

OTM - Object template manager	Version: 1.1
Vision	Date: 2011-08-31
Draft	

Vision

1. Introduction

The purpose of this document is to collect, analyze and define high-level needs and features of the object template manager (OTM). It focuses on the capabilities needed by the stakeholders, and the target users, and why these needs exist. The details of how the OTM fulfils these needs are detailed in the use-case and supplementary specifications.

The OTM is a system that will simplify the process of creating object templates. In the current process there are several tedious tasks associated with modifying. It is the ambition of the OTM to reduce these steps.

1.1 Purpose

The purpose of this document is to introduce the OTM by specifying its features and its intended usage.

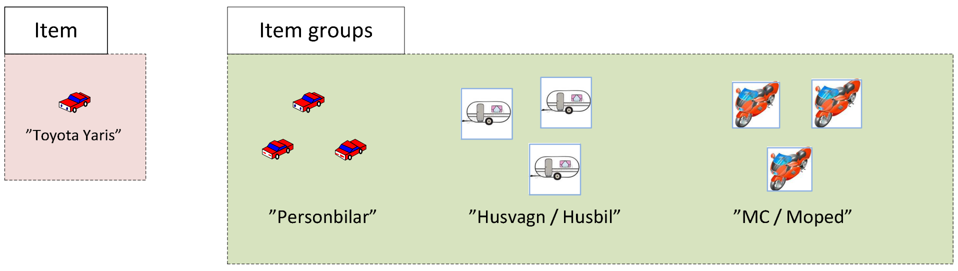
1.2 Scope

This vision document is associated with the general handling of items and object templates and also the project concerning customer objects (“Kundobjekt”).

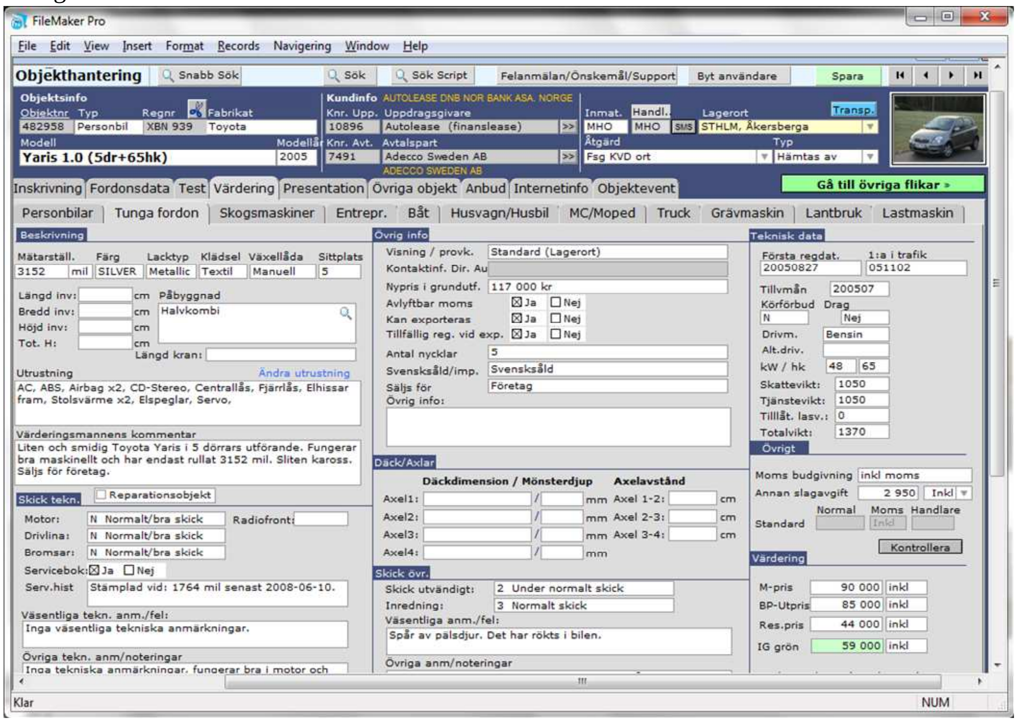
1.3 Definitions, Acronyms and Abbreviations

- Items: The items for sale on KVD’s web page.
- Item group: The groups that the items are sorted in. For example: “Personbilar”, “Entreprenad” and “MC/Moped”.
- Object template: The templates currently used in Kastor for different item groups. The templates are forms with different sets of fields depending on their item group.
- Kastor: The Filemaker based database that handles all items.
- OTM: Object template manager.
- M.V.C.: Model View Controller, a design pattern in systems engineering that separates logic, GUI and data storage.
- KVD Web: The collected web business of KVD including kvd.se, kvdauctions.com, m.kvd.se and “Kundobjekt”.

The picture below describes the definition of Item and Item groups:



In the following picture we can see a screenshot of an Object template from Kastor of the Item group “Tunga fordon”:


FileMaker Pro
File Edit View Insert Format Records Navigering Window Help
Objekthantering
Snabb Sök
Sök
Sök Script
Felanmälan/Önskemål/Support
Byt användare
Spara
Transp.
SMS
Gå till övriga flikar »
Inskrivning Fordonsdata Test Värdering Presentation Övriga objekt Anbud Internetinfo Objektevent
Personbilar Tunga fordon Skogsmaskiner Entrepr. Båt Husvagn/Husbil MC/Moped Truck Grävmaskin Lantbruk Lastmaskin
Ändra utrustning
Kontrollera
Klar
NUM

Objektsinfo
Objektnr 482958 · Typ Personbil · Regnr XBN 939 · Fabrikat Toyota
Modell Yaris 1.0 (5dr+65hk) · Modellår 2005

Kundinfo AUTOLEASE DNB NOR BANK ASA. NORGE
Knr. Upp. 10896 · Uppdragsgivare Autolease (finanslease)
Knr. Avt. 7491 · Avtalspart Adecco Sweden AB
ADECCO SWEDEN AB

Inmat. MHO · Handl. MHO · Lagerort STHLM, Åkersberga
Åtgärd Fsg KVD ort · Typ Hämtas av

Beskrivning
Mätarställ. 3152 mil · Färg SILVER · Lacktyp Metallic · Klädsel Textil · Växellåda Manuell · Sittplats 5
Längd inv: cm
Bredd inv: cm
Höjd inv: cm
Tot. H: cm
Påbyggnad: Halvkombi
Längd kran:
Utrustning: AC, ABS, Airbag x2, CD-Stereo, Centrallås, Fjärrlås, Elhissar fram, Stolsvärme x2, Elspeglar, Servo,
Värderingsmannens kommentar: Liten och smidig Toyota Yaris i 5 dörrars utförande. Fungerar bra maskinellt och har endast rullat 3152 mil. Sliten kaross. Säljs för företag.

Skick tekn. ☐ Reparationsobjekt
Motor: N Normalt/bra skick · Radiofront:
Drivlina: N Normalt/bra skick
Bromsar: N Normalt/bra skick
Servicebok: [✓ Ja] Nej
Serv.hist: Stämplad vid: 1764 mil senast 2008-06-10.
Väsentliga tekn. anm./fel: Inga väsentliga tekniska anmärkningar.
Övriga tekn. anm/noteringar: Inga tekniska anmärkningar, fungerar bra i motor och

Övrig info
Visning / provk.: Standard (Lagerort)
Kontaktinf. Dir. Au:
Nypris i grundutf.: 117 000 kr
Avlyftbar moms: [✓ Ja] Nej
Kan exporteras: [✓ Ja] Nej
Tillfällig reg. vid exp.: [✓ Ja] Nej
Antal nycklar: 5
Svensksåld/imp.: Svensksåld
Säljs för: Företag
Övrig info:

Däck/Axlar
Däckdimension / Mönsterdjup · Axelavstånd
Axel1: / mm · Axel 1-2: cm
Axel2: / mm · Axel 2-3: cm
Axel3: / mm · Axel 3-4: cm
Axel4: / mm

Skick övr.
Skick utvändigt: 2 Under normalt skick
Inredning: 3 Normalt skick
Väsentliga anm./fel: Spår av pälsdjur. Det har rökts i bilen.
Övriga anm/noteringar

Teknisk data
Första regdat. 20050827 · 1:a i trafik 051102
Tillvmån 200507
Körförbud N · Drag Nej
Drivm. Bensin
Alt.driv.
kW / hk 48 65
Skattevikt: 1050
Tjänstevikt: 1050
Tilllåt. lasv.: 0
Totalvikt: 1370

Övrigt
Moms budgivning: inkl moms
Annan slagavgift: 2 950 Inkl
Standard: Normal · Moms Inkl · Handlare

Värdering
M-pris 90 000 inkl
BP-Utpris 85 000 inkl
Res.pris 44 000 inkl
IG grön 59 000 inkl

1.4 Overview

The Vision document will explain the positioning of the OTM, describe the stakeholders and product features.

In the *Appendix* there are examples of actual code that the company currently is using. The code examples are in this Vision document to give a clear picture of the desired output from the OTM. There is also a simple metamodel in the appendix that serves to give stakeholders an idea of what the final metamodel could look like. What the OTM’s interface might end up looking like can be seen in the screenshots of the tree based editor and the GMF editor.

Finally, there is a detailed use case in the appendix describing the process of managing object templates in the OTM.

2. Positioning

2.1 Business Opportunity

The business opportunity in this case comes from improving the old process for handling object templates. This will help reduce maintenance costs and lessen the workload of IT personnel. Also, by moving the process out of Kastor it will reduce some of the traffic and time waste therein.

The main economic benefit comes from the reduced lead time of getting items up for auction. The auction business requires rapid processes because there is constant flow of incoming and outgoing items. Also, by improving the current process the risk for auctioning out items under the wrong object template (i.e. giving an incorrect presentation of items for sale) will be lower.

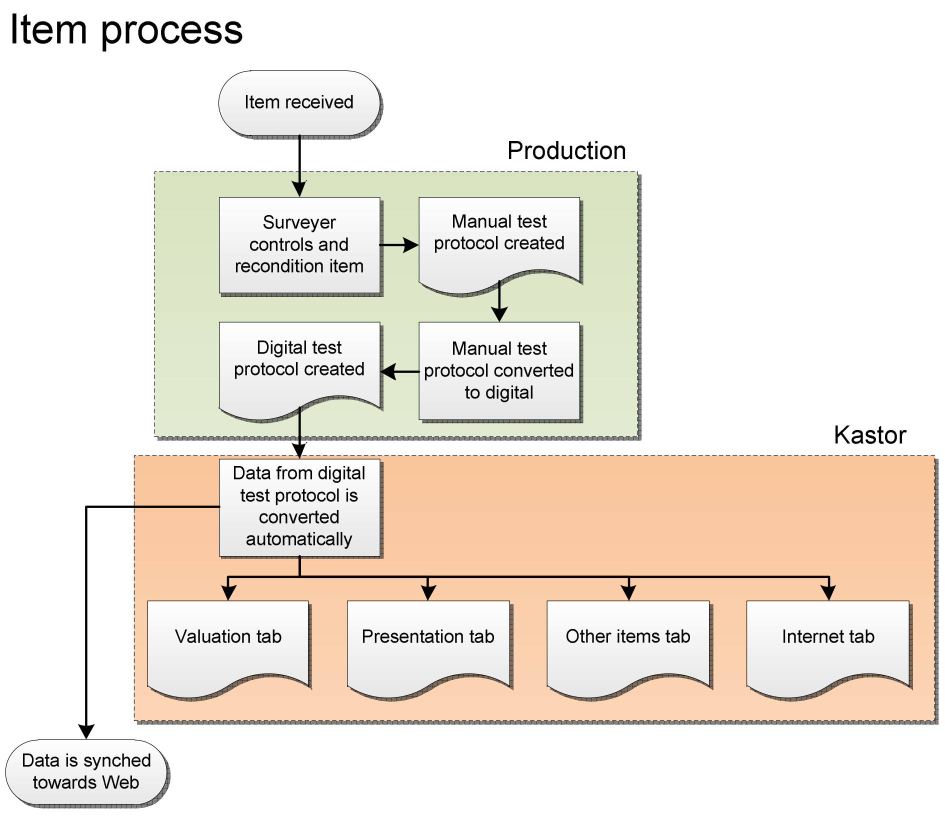
2.2 Problem Statement

The problem of	Changing an object template.
affects	IT Personnel, administrators
The impact of which is	Increased lead time.
A successful solution would	• Reduce lead time • Improve maintainability • Shorten the time for getting an item up for auction

2.3 Product Position Statement

The OTM will help automate the existing process of managing object templates. It will also be developed by using a MDE approach and can be seen as an opportunity to showcase this development methodology for future projects.

The current process of getting items up for auction on the Web:


Item process
Item received
Production
Surveyer controls and recondition item
Manual test protocol created
Manual test protocol converted to digital
Digital test protocol created
Kastor
Data from digital test protocol is converted automatically
Valuation tab
Presentation tab
Other items tab
Internet tab
Data is synched towards Web

3. Stakeholder and User Descriptions

3.1 Stakeholder Summary

Name	Represents	Role
IT Personnel	The in-house developers at KVD.	Provides expertise regarding technical issues and questions concerning KVD’s existing technologies. Also they are to some extent involved in the OTM development.
Buyer	The buyer is the person who ordered the OTM.	Ensure that the project is on budget.
IT Manager	Development and managing.	Link between all stakeholders. Wants the project to be a balance between satisfying the buyer’s demand, improving existing process and exploring new technology.
Client	The customers who want to auction items on KVD web.	Different clients might want to have different object templates, therefore clients affect the OTM’s changeability.

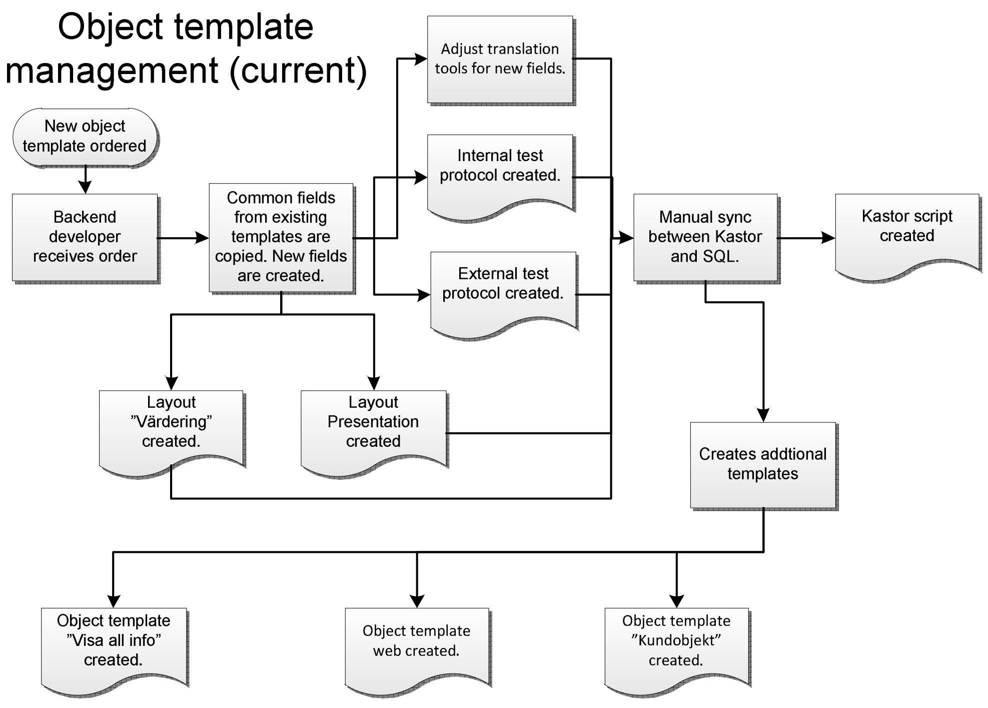
3.2 User Summary

Name	Description	Stakeholder
Backend developers	The personnel managing the backend at KVD. They are the main users of the OTM.	The IT Personnel are the stakeholders for this user type.

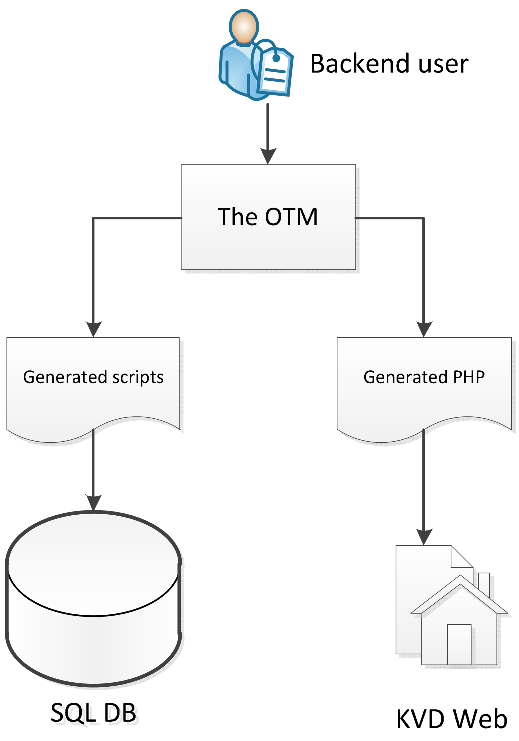
In future versions there are plans on having users who are less capable and to also support multiple role privileges (admin, user etc.). As for now, having only Backend developers as users is sufficient.

3.3 User environment

The main environment for the OTM is Eclipse. There are users running Eclipse on OS X and Windows. The way that users interact with object templates in the current system and in the envisioned system is illustrated below:


Object template management (current)
New object template ordered
Backend developer receives order
Common fields from existing templates are copied. New fields are created.
Adjust translation tools for new fields.
Internal test protocol created.
External test protocol created.
Layout "Värdering" created.
Layout Presentation created
Manual sync between Kastor and SQL.
Kastor script created
Creates addtional templates
Object template "Visa all info" created.
Object template web created.
Object template "Kundobjekt" created.

Envisioned System Overview



3.4 Key Stakeholder / User Needs

Need	Priority	Concerns	Current Solution	Proposed Solutions
Adoptable object templates	High	It is time consuming to change object templates.	Creating and changing templates in Filemaker Pro and writing corresponding scripts.	An Eclipse based editor that generates O.R.M.-objects and SQL-scripts.
Low development cost	High	The buyer has asked for that the development cost of the OTM is low in comparison with other projects.	The company is currently working in an agile fashion using the SCRUM methodology. The technology is object-oriented PHP following the M.V.C. design pattern.	Develop the product using an agile methodology and Model Driven Engineering.

Internationalization	Medium	The company is broadening their activity into new countries. Object templates must be multilingual.	Dynamic PHP	Model-2-Text transformation in multiple languages.
Reduced lead time for new items	High	The company wants to improve process of getting items up for auction.	See Adoptable object templates.	New working environment (the OTM) for backend developers reducing number of tasks such as script writing.

3.5 Alternatives and Competition

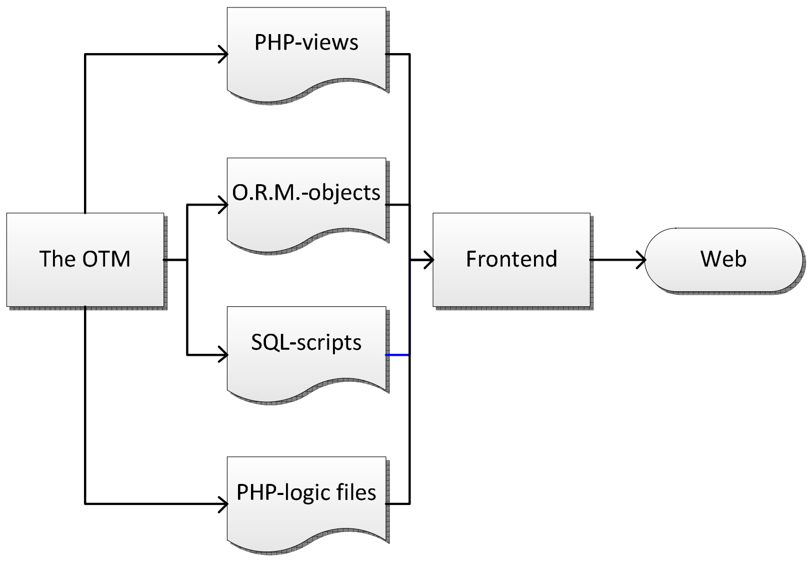
An alternative to the OTM would be to continue developing Kastor to overcome the problems but the company wants to move the object template handling out of there. Another alternative would be to replace the entire infrastructure.

4. Product Overview

The OTM is used as an editor in Eclipse. IT Personnel will be able to use it to construct new object templates and to make changes to existing ones. Also, database schemas and web interface components will be generated from these templates.

4.1 Product Perspective

The OTM is a component working with other systems in order to generate content on the web. Communication with the other systems is handled via O.R.M-objects and SQL. It also generates PHP-views and logic for display on the web. An overview of the product perspective can be seen in the picture below:



4.2 Summary of Capabilities

Object Template Manager

Customer Benefit	**Supporting Features**
IT Personnel will spend less time on managing object templates .	5.1-5.5
Reduced time to get items up for auction.	See above.

4.3 Assumptions and Dependencies

The OTM is dependent on the communication with the frontend web page. A solution for this is to have the OTM generate O.R.M-objects that later can be handled by PHP-code.

5. Product Features

5.1 Create object templates

Via the Eclipse editor it will be possible to create, change and delete object templates.

5.2 Change existing object templates

See above. When changing a template in the editor, a new SQL-script will be created. This will then be compared with the existing database and only the new changes will be updated. In this way no existing data will be lost.

5.3 Delete existing object templates

See above.

5.4 Generation of O.R.M.-objects

The OTM will generate O.R.M.-objects from the model to make it easy to display the info online via PHP.

5.5 Generation of SQL-scripts

Since the main storage of data is MySQL, the OTM will generate SQL-scripts that can be run to save a created object template in the database. The SQL-script is a file with a series of CREATE-statements for the tables to be created.

5.6 M.V.C. Support

The OTM will generate files conforming to the M.V.C. design pattern. This means the aforementioned O.R.M.-objects and SQL-scripts that represent the Model part, plus the generation of PHP-files for the View part and for the logic connecting the Model and the View known as the controller.

5.7 Tree-based EMF editor

The Eclipse Modeling Framework (EMF) supports the generation of a tree-based editor (see Appendix). This will be the user interface for the OTM. (There is also a possibility to create graphical interface using the Graphical Modeling Framework.).

5.8 Multilingual

The OTM will have support for multiple languages and generate necessary SQL-scripts and O.R.M.-objects.

5.9 Automated installation

By the use of additional scripts, the OTM will be able to automatically install required PHP-files onto the server and execute necessary SQL-scripts for persistency control. Apart from creating the templates in the editor, the user will not have to make additional configurations.

5.10 View-control over fields

The OTM has support for controlling which fields are displayed in which web pages. For instance, a field might not be displayed under some circumstances due to dependencies on language or region. In the editor, the user can select the view properties of a field.

5.11 Grouping

It is possible to group fields into certain sets of common properties. There is one Shared group that is a set of fields that are used in all object templates. There is also the possibility to group fields into subgroups such as “Entreprenad\Väggbyggnadsmaskiner” or “Båt\Segelbåt” that in turn have sub-shared fields that are used in all instances of a certain subgroup.

6. Constraints

- The OTM should be developed using the Eclipse platform.
- The editor will be created for the Eclipse environment.

7. Precedence and priority

Considering the complexity of the OTM it is necessary to divide the development into three major milestones:

7.1 Milestone 1 – Basic functionality

- Generation of SQL-scripts
- Generation of PHP code for O.R.M.-objects
- Tree-based editor

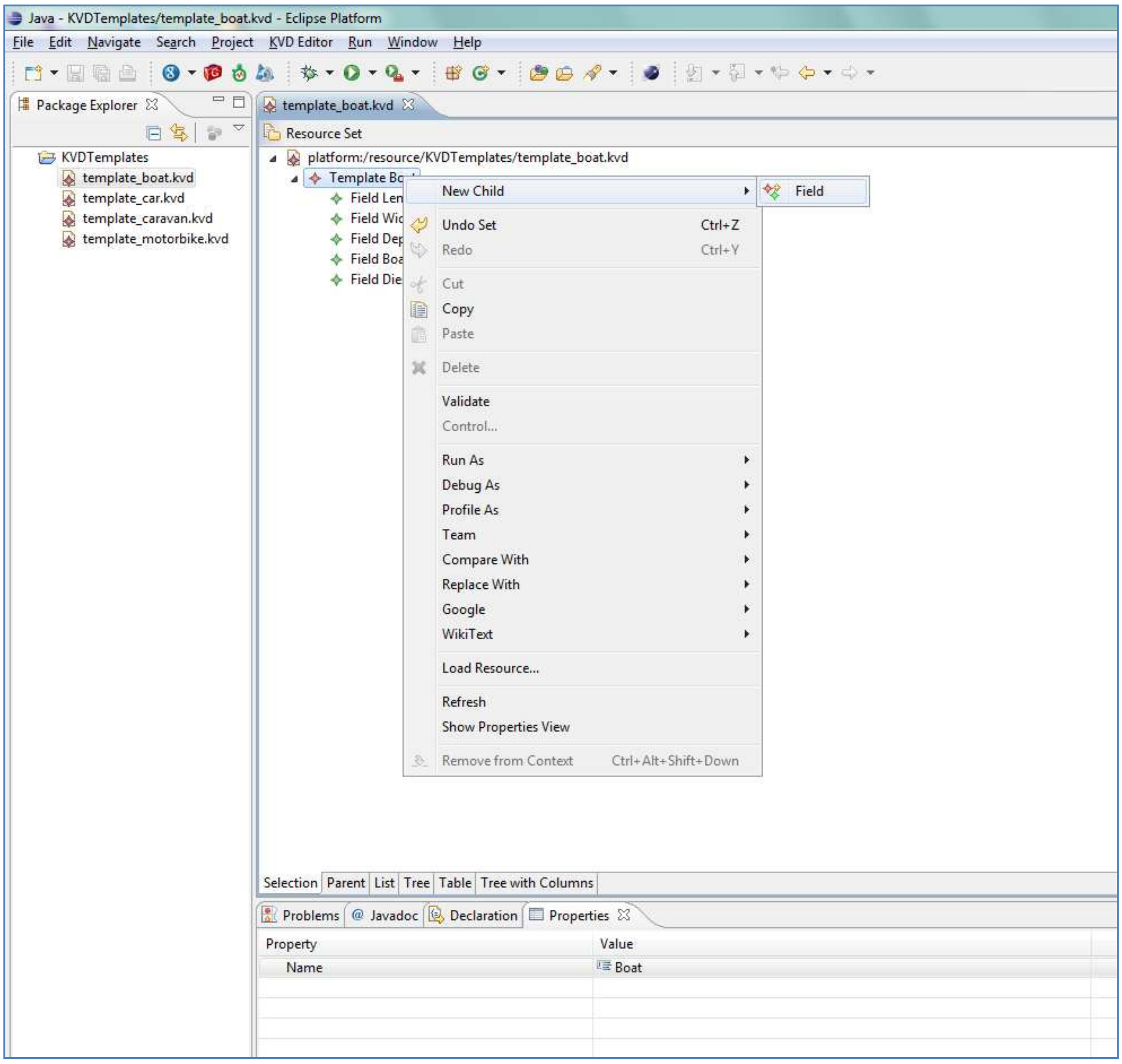
7.2 Milestone 2 – Views and logic

- PHP-code for views and logic

7.3 Milestone 3 – Improved automatization and usability

- Fully automated scripts (the OTM will automatically upload PHP-code and execute SQL-scripts to the appropriate nodes of the system).
- Graphical user interface

8. Appendix



Property	Value
Name	Boat

Figure 1: Example of a tree-based editor.

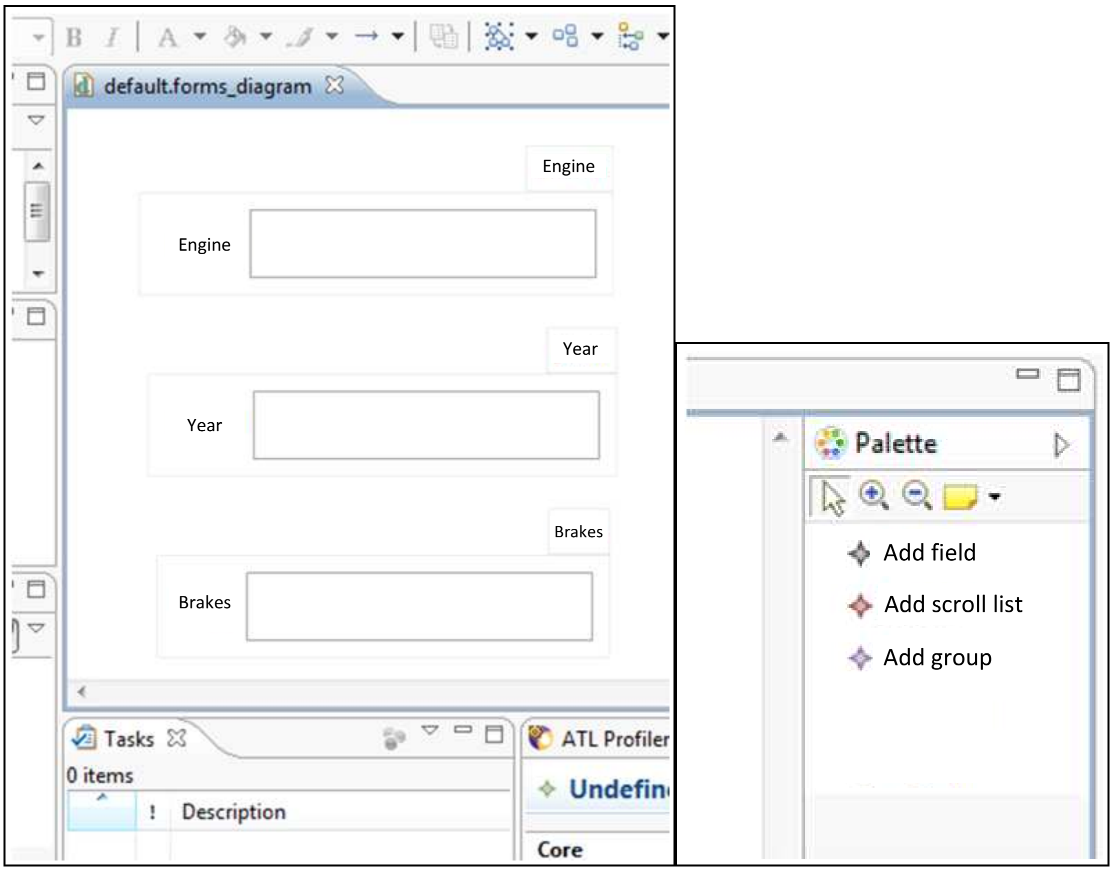


Figure 2: A GMF based editor showing the workspace and a tool palette.

8.1 Example of an SQL-script (Model). Extract from kvd_web_utv_2011-08-18.sql:

```
# Dump of table objlista
# ------------------------------------------------------------

CREATE TABLE `objlista` (
  `ObjListaNr` int(10) unsigned NOT NULL auto_increment,
  `ObjektNr` int(10) unsigned default NULL,
  `ListNr` int(10) unsigned default NULL,
  `AuktionsTyp` char(60) default NULL,
  `Lagerort` char(60) default NULL,
  `Fabrikat` char(30) default NULL,
  `Modell` char(75) default NULL,
  `Arsmodell` int(11) default NULL,
  `Reservationspris` int(11) default NULL,
  `WebBud` char(5) default NULL,
  `Marknadspris` int(11) default NULL,
  `Status` tinyint(3) unsigned default NULL,
  `HogstaBudBelopp` int(11) default NULL,
  `HogstaBudAlias` char(30) default NULL,
  `Handlaggare` char(5) default NULL,
  `LagstaPris` int(11) default NULL,
  `HogstaPris` int(11) default NULL,
  `LagstaHojning` int(11) default NULL,
  `Visas` tinyint(3) unsigned default NULL,
  `SlagAvgift` int(11) default NULL,
  `SlagAvgiftHandlare` int(11) default NULL,
  `SlagAvgiftMoms` char(20) default NULL,
  `MomsBudgivning` varchar(95) default NULL,
  `UppdatKastor` tinyint(3) unsigned default NULL,
  `UppdatKastorVad` char(80) default NULL,
  `MomsText` varchar(95) default NULL,
  `AuktionStanger` datetime default NULL,
  `AuktionForlangSek` int(10) unsigned default NULL,
  `AuktionNedrakning` datetime default NULL,
  `HogstaBudTid` datetime default NULL,
  `AuktionStangdTid` datetime default NULL,
  `SlagavgiftText` varchar(255) default NULL,
  `Export` varchar(45) default NULL,
  `Losenord` varchar(45) default NULL,
  `LosenordKrav` tinyint(3) unsigned default NULL,
  `Grupp` varchar(8) default NULL,
  `TimeStamp` timestamp NOT NULL default CURRENT_TIMESTAMP on update
CURRENT_TIMESTAMP,
  `Ikon` smallint(5) unsigned default NULL,
  `Villkor` varchar(90) default NULL,
  `Valuta` varchar(45) default NULL,
  `ObjektUrl` varchar(255) default NULL,
  `ListBenamning` varchar(90) default NULL,
  `VisaObjektKVD` int(10) unsigned default NULL,
  `VisaObjektExport` int(10) unsigned default NULL,
  `ArendeTyp` varchar(45) default NULL,
  `Finansiering` tinyint(3) unsigned default NULL COMMENT 'Erbjud ej
finansiering när värde = 0. (False)',
  PRIMARY KEY  (`ObjListaNr`),
  UNIQUE KEY `unique_ObjektNr` (`ObjektNr`),
  KEY `FabrikatModell` (`Fabrikat`,`Modell`),
  KEY `Reservationspris` (`Reservationspris`),
  KEY `Lagerort` (`Lagerort`),
  KEY `Marknadspris` (`Marknadspris`),
  KEY `Nedrakning` (`AuktionNedrakning`),
  KEY `AuktionStanger` (`AuktionStanger`),
  KEY `ListaSok` USING BTREE
(`ListNr`,`Visas`,`Status`,`AuktionNedrakning`,`Grupp`),
  KEY `Fabrikat` (`Fabrikat`),
  KEY `Modell` (`Modell`),
  KEY `Visas` (`Visas`),
  KEY `Grupp` (`Grupp`),
  KEY `Status` (`Status`),
  KEY `AuktionStangdTid` (`AuktionStangdTid`),
  KEY `ListNr` (`ListNr`),
  KEY `TimeStamp` (`TimeStamp`),
  KEY `AuktionsNedrakning` (`AuktionNedrakning`)
) ENGINE=InnoDB AUTO_INCREMENT=96275 DEFAULT CHARSET=latin1
ROW_FORMAT=DYNAMIC;
```

8.2 Example of O.R.M.-object (Model). Objlist.php:

```
<?php
/**
 * Defines the Kvd_Model_Objlist class
 *
 * PHP Version 5
 *
 * @category KvdCore
 * @package  KvdCore.class
 * @author   Erik Hantelius <erik@kvd.se>
 * @license  http://trac:8080/kvdcore/license KVD License
 * @link     http://trac:8080/kvdcore
 */

/**
 * A Kvd_ORM class
 */
class Kvd_Model_Objlist extends Kvd_ORM_I18n
{
      public $db = 'modify';
      protected $table_name = 'objlista';
      protected $table_columns = array(
            'ObjListaNr',
            'ObjektNr',
            'ListNr',
            'AuktionsTyp',
            'Lagerort',
            'Fabrikat',
            'Modell',
            'Arsmodell',
            'Reservationspris',
            'WebBud',
            'Marknadspris',
            'Status',
            'HogstaBudBelopp',
            'HogstaBudAlias',
            'Handlaggare',
            'LagstaPris',
            'HogstaPris',
            'LagstaHojning',
            'Visas',
            'SlagAvgift',
            'SlagAvgiftHandlare',
            'SlagAvgiftMoms',
            'MomsBudgivning',
            'UppdatKastor',
            'UppdatKastorVad',
            'MomsText',
            'AuktionStanger',
            'AuktionForlangSek',
            'AuktionNedrakning',
            'HogstaBudTid',
            'AuktionStangdTid',
            'SlagavgiftText',
            'Export',
            'Losenord',
            'LosenordKrav',
            'Grupp',
            'TimeStamp',
            'Ikon',
            'Villkor',
            'Valuta',
            'ObjektUrl',
            //'ListBenamning',
            'VisaObjektKVD',
            'VisaObjektExport'
      );
      protected $primary_key = 'ObjektNr';
      protected $has_many = array(
            'bids' => array('model' => 'bid', 'foreign_key' => 'ObjektNr'),
      );
      protected $belongs_to = array(
                                                      'objpres' => array(
                                                            'model' =>
'objpres',
                                                            'foreign_key' =>
'ObjektNr'
                                                      )
                                                );
      protected $trans_table_name = 'obj_oversatt';
      protected $trans_columns = array(
            'Objektstyp'
      );
      protected $language_key = 'KeySprak';

}
```

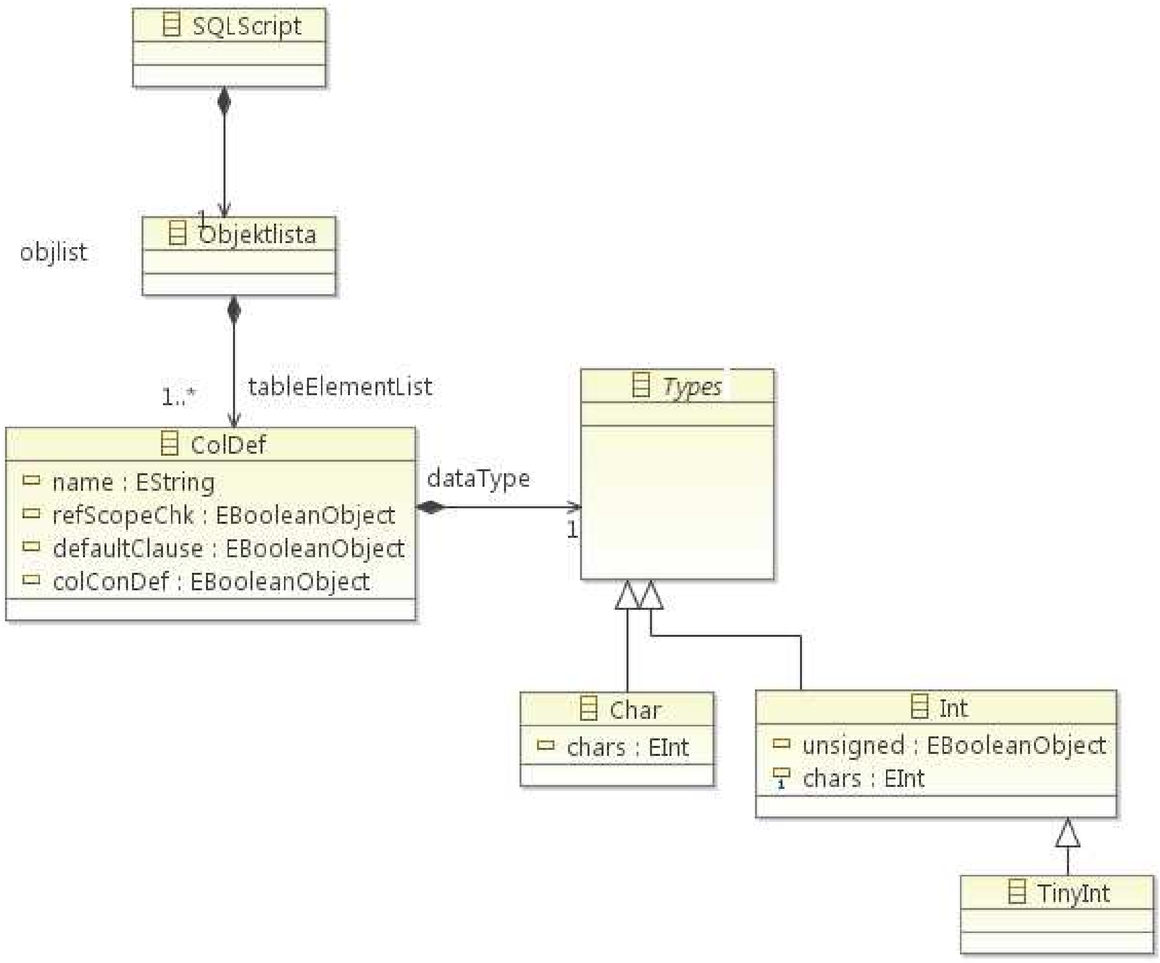
8.3 Example of PHP logic (Controller). Switch statement from objectmodule.php:

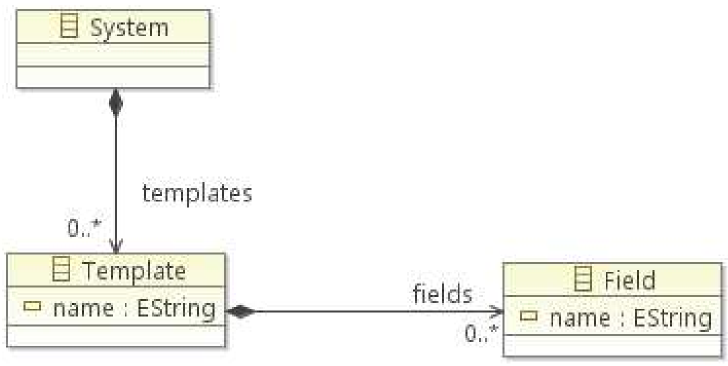
```
switch ($objpres->Mall) {

                  case 1: //01: Personbil

                        $view = View::factory(Site::instance()-
>view_path('objectmodules/facts_car'));

                        $view->facts = self::getCarFactFields($objpres);

                        $view->bp_link = self::getBPLink($objpres);

                        break;

                  case 17: // 17: Tunga fordon

                        $view = View::factory(Site::instance()-
>view_path('objectmodules/facts_heavy'));

                        $view->facts = self::getHeavyFactFields($objpres);

                        break;



                  case 21: // 21: Båtar

                        $view = View::factory(Site::instance()-
>view_path('objectmodules/facts_boat'));

                        $view->facts = self::getBoatFactFields($objpres);

                        break;



                  case 2: //02: övrigt

                  case 19: //19: entreprenad

                  case 23: //23: Husvagn
                  default:

                        $view = View::factory(Site::instance()-
>view_path('objectmodules/facts'));

                        // TODO Assign facts!

                        break;

                  }

                  $view->objpres = $objpres;

                  Kvd_TTL::add('object');

                  Kvd_Apc::store($apcName, $view->render(), 15);

                  $module = $view;

            }
```

8.4 Example of PHP view (View). Description.html.php-file.

```
<?php if($site == "mobile") { ?><span class="graytitle2">
<?php echo ___('facts.description'); ?></span><br><?php } ?>
<?php if(isset($_GET['allinfo'])) { ?><span class="allinfo_tab">
<?php echo ___('facts.facts'); ?></span><br><?php } ?>
<?php
echo '<p>'.implode(', ', $fields['info']).'</p>';

if(implode(', ', $fields['equipment']) != "")
{
   echo '<h4>'.__('object.equipment', NULL, 'tag').'</h4>';
   echo '<p>'.implode(', ', $fields['equipment']).'</p>';
}

if(implode(', ', $fields['truck_info']) != "")
{
   echo '<p>'.implode(', ', $fields['truck_info']).'</p>';
}

if(implode(', ', $fields['entrerpenad_info']) != "")
{
   echo '<p>'.implode(', ', $fields['entrerpenad_info']).'</p>';
}

if(implode(', ', $fields['Batteri']) != "")
{
   echo '<h4>'.___('object.battery').'</h4>';
   echo '<p>'.implode(', ', $fields['Batteri']).'</p>';
}

if(implode(', ', $fields['Rototilt']) != "")
{
   echo '<h4>'.___('object.roto_tilt').'</h4>';
   echo '<p>'.implode(', ', $fields['Rototilt']).'</p>';
}

if(implode(', ', $fields['stated_dimensions']) != "")
{
   echo '<h4>'.___('object.transport_dimensions').'</h4>';
   echo '<p>'.implode(', ', $fields['stated_dimensions']).'</p>';
}

if(implode(', ', $fields['comments']) != "")
{
      if(Site::instance()->name == 'main'){
                  echo '<span 
class="graytitle2">'.___('field.object.VarderingsmanKommentar_facts').'</span
>';
                  if (!isset($_GET['allinfo'])) {
                        echo '<img 
src="'.URL::site('assets/images/main/pointer_1.gif').'" 
id="ShowAllInfoArrowImg" /><span><a class="iframe iframe_allinfo" 
href="?allinfo=1">'.__('object.allinfo').'</a></span>';
                  }
      } else {
            echo '<span 
class="graytitle2">'.___('field.object.VarderingsmanKommentar_facts').'</span
>';
      }
   echo '<p>'.implode(', ', $fields['comments']).'</p>';
}
```

8.5 Metamodel example


SQLScript
1
Objektlista
objlist
1..*
tableElementList
ColDef
name : EString
refScopeChk : EBooleanObject
defaultClause : EBooleanObject
colConDef : EBooleanObject
dataType
1
Types
Char
chars : EInt
Int
unsigned : EBooleanObject
chars : EInt
TinyInt


System
templates
0..*
Template
name : EString
fields
0..*
Field
name : EString