M. Staron and C. Wohlin, "An Industrial Case Study on the Choice between Language Customization Mechanisms", Proceedings of International Conference on Product Focused Software Process Improvement (PROFES06), pp. 177-191, LNCS-series, Springer Verlag, Amsterdam, The Netherlands, 2006. Best paper award.

An Industrial Case Study on the Choice between Language Customization Mechanisms

Miroslaw Staron[1)2)] and Claes Wohlin[1)]

[1)] Department of Systems and Software Engineering
School of Engineering
Blekinge Institute of Technology
(miroslaw.staron, claes.wohlin)@bth.se

[2)] Department of Applied IT
IT University in Gothenburg
miroslaw.staron@ituniv.se

Abstract. Effective usage of a general purpose modeling language in software engineering poses a need for language *customization* – adaptation of the language for a specific purpose. In the context of the Unified Modeling Language (UML) the customization could be done using two mechanisms: developing profiles and extending the metamodel of UML. This paper presents an industrial case study on the choice between metamodel extensions and profiles as well as the influence of the choice on the quality of products based on the extensions. The results consist of a set of nine prioritized industrial criteria which complement six theoretical criteria previously identified in the literature. The theoretical criteria are focused on the differences between the extension mechanisms of UML while the industrial criteria are focused on development of products based on these extensions. The case study reveals that there are considerable differences in effort required to develop comparable products using each mechanism and that the quality (measured as correctness of a product) is different for these comparable products by an order of magnitude.

1. Introduction

Effective usage of a general-purpose modeling language (like the Unified Modeling Language – UML, [1]) in the course of software development strives for a customization of the language – i.e. its fine-tuning and adaptation for a specific purpose. For example in the context of code generation, customizing the language could be done by defining additional modeling constructs that would enable generating a complete source code. The customized languages can be primary assets of MDA (Model Driven Architecture [2]) frameworks for automating software construction through model transformations – for example as presented in our previous case study [3]. UML has two extension mechanisms which can be used for its customization – profiles and metamodel extensions [4]. The option of creating a metamodel extension (as opposed to creating profiles) was usually not supported in UML modeling tools and thus not considered so far in the enterprises customizing

UML. The situation, however, changes as modern modeling tools expose mechanisms for extending the metamodel of UML. Examples of this kind of tools are Telelogic Tau G2 and Coral Modeling Framework [5]. These tools are modeling tools with metamodeling capabilities – i.e. tools primarily dedicated for creating UML models with possibility of altering the metamodel of UML. It should be noted that these tools are dedicated for the modeling of software and not for the development of modeling tools – which makes them representative for modeling tools used in software development companies. As the new customization possibilities emerge in modeling tools, companies willing to use the modeling language more effectively consider customizing the language and hence need to choose the appropriate extension mechanism.

The case study presented in this paper is performed at a UML modeling tool vendor which regularly develops language extensions, with significant experience in this area – Telelogic AB in Malmö, Sweden (referred to as Telelogic hereafter). The extensions are the basis for some of the products developed by the company. Examples of these products are domain specific modeling tool extensions for modeling real-time software, full source code generation for embedded systems or architecture frameworks. The products are based on the extensions of the modeling language but they also require supporting software components to provide additional functionality defined by the extension. These products are dedicated mostly for companies developing software for the domains of real-time and embedded software. The domains pose strict requirements, e.g. that the products should allow for early verification based on models – thus the models created with Telelogic's modeling tool – Tau G2 – can be executable which in turn leads to strict requirements on the quality of the customizations of the tool. The language customization endeavors at Telelogic are conducted within a UML modeling tool which is a similar situation to companies customizing the modeling language while not being tool vendors themselves. The large number of developed extensions by Telelogic provided us with a unique opportunity of studying products which are based on each of the extension mechanisms thus allowing comparison of products which are very similar yet based on different extension mechanisms. This unique opportunity provides an evidence of influences of choosing the mechanisms in industrial context.

As a starting point in designing our case study we used the theoretical differences between the extension mechanisms identified previously in the literature [4]. The initial set of criteria based on these differences does not contain considerations on the implications on the products based on the customized notation, for example in terms of the quality. The products based on the customized language are adaptations of tools used in software development as presented in Section 3. In addition to identifying the industrial criteria focused on developing complete products based on the extensions, studying comparable metamodel extensions and profiles provided us with differences in quality (measured as correctness) of the products. In addition to the quality we also studied effort required to develop the language extensions in order to investigate whether there are differences between profiles and metamodel extensions in this aspect. We discuss the relevance of the results of this case study in the context of the results of our previous industrial case study at an enterprise working with customizing UML in order to enable automating part of their software development process [3].

The paper starts with the presentation of the related work in the field in Section 2. The two compared techniques and the differences between them are presented in Section 3 followed by the description of the design of the case study performed and its operation in Section 4. The results of the study and their analysis are presented in Section 5 with the evaluation of the validity of the study in Section 6. The conclusions of the paper are presented in Section 7.

2. Related work

Language customization is one of the core elements of MDA. The MDA related literature often discusses the use of both profiles and metamodels for language customization (for example [6, 7]), but does neither give guidelines when to use one over the other nor criteria for choosing between the extension mechanisms. The discussions presented in these papers are taken into account while considering the differences between metamodels and profiles in Section 3. The authors of [8] in the context of MDA conclude that profiles should always be used over metamodels due to their simplicity. The suggestion, however, is based on an analytical evaluation of the expressiveness of profiles and metamodel extensions without considering industrial applications of profiles or products based on the customizations.

The issues of creating custom modeling notations are similar to problems studied while creating Domain Specific Modeling Languages (DSLs). In particular, while considering such initiatives as software factories or generative programming [9]. As opposed to that work this paper considers creating DSLs based on a general-purpose language, not creating DSLs from scratch.

In this paper we discuss the issues of choosing between metamodel extensions and profiles in the context of language customization. The issues related to extending metamodels, however, are a part of a bigger aspect – language engineering. Details on language engineering – in particular creating a language from scratch – can be found in [10]. Furthermore, it is not always the case that metamodels and profiles can both be used for the same purpose. Considerations on this matter are discussed in [11]. The authors classify metamodel extensions in a similar way as in [12] and draw conclusions in the context of domain modeling and code generation. Despite the similarity of the purpose of that study to ours, the context (specific domain of extensions) does not allow drawing conclusions from the study in [11] in other cases than code generation or domain modeling.

3. Language customization

Usually using an off-the-shelf modeling language like UML is dictated by the tool support and the knowledge base. Benefits from using a general-purpose language can be increased if a company is willing to *customize* it for the specific needs. This means that the language is adapted for a specific purpose and a tool that takes advantage of the customization is created. In the context of code generation, customizing the language could be done by defining dedicated constructs that would provide (along with customizing the code generators) means for generating a complete source code for the developed system.

Advanced industrial usage of UML (as studied at Volvo IT [3] and ABB Robotics in Sweden) showed that creating the language customization can (and should) be handled by modelers within the enterprise since they possess the necessary domain knowledge. The motivation behind the research presented in this paper is that practitioners improving the practice of modeling in their enterprises need support in choosing between costly (but powerful) metamodel extensions and cheaper (but limited) profiles since these practitioners are usually not language engineers. They also need to be able to assess potential consequences of their decision in terms of quality of the customization and the effort required to develop them. The scenario in which a language is customized in a company using a modeling tool with metamodeling capabilities for the development of models is presented in Figure 1.

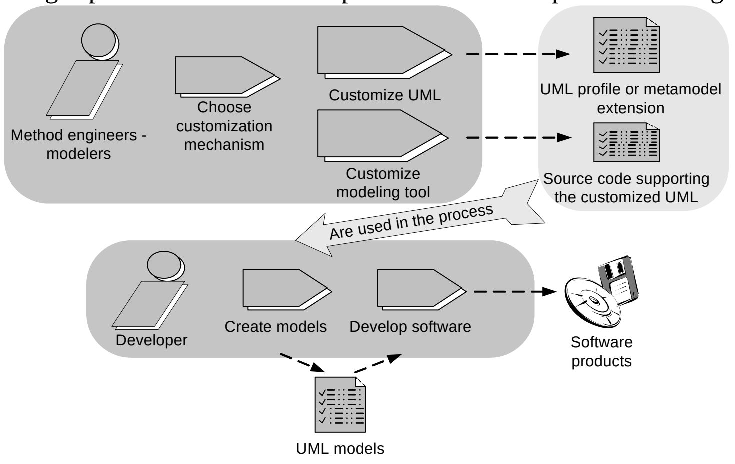
Method engineers - modelers
Choose customization mechanism
Customize UML
Customize modeling tool
UML profile or metamodel extension
Source code supporting the customized UML
Are used in the process
Developer
Create models
Develop software
Software products
UML models

Fig. 1. Language customization in companies – an abstract view

Method engineers (who are also modelers) are specialists possessing the necessary skills to develop language extensions and to customize the appropriate tools. They customize the modeling language by creating language extensions and they customize the tools by developing add-ins extending the functionality of the tools. The add-ins are required so that the developers (at the bottom of Figure 1) can use the extensions in an effective way. In their work the method engineers need to choose the appropriate customization mechanism for a particular task. It was found in this study and it was observed in the study presented in [3] that there are usually a handful of people (a dedicated small team with a designated architect) in a company who can play this role. The results of the work of the method engineers are the customized modeling language and the customized tool which is used by developers in the company to create models.

In the case of Telelogic the products based on the customized language are add-ins for the tool, which provide such functionality as, for example code generation for C++, architecture modeling with DoDAF (US Department of Defense Architecture Framework) or providing new diagrams for the tool. In the case of other companies, the products are additional components which can take advantage of the extensions to the base language. For example in the case of a company developing an MDA framework, the product was a framework for model driven software development, including add-ins for software development and project management. This company also regarded extensions of UML to be of primary importance in this kind of products.

3.1. Theoretical differences between profiles and metamodel extensions

As a starting point for elaborating criteria in our case study we adopted theoretical differences based on the analysis of criteria for choosing techniques for building language families [4]. The differences are numbered T1 – T6 (where the letter *T* indicates that they arise from theoretical analysis). The differences are applicable to the issue of language customization and as such are evaluated in this study.

T1: **Profiles are easier to construct.** Not as much knowledge is required to develop a profile as to develop a metamodel extension.

T2: **Profiles promote reusability.** Profiles might be applied (thus reused) to different metamodels provided the extended elements (and the mechanisms of profiles) are presented in both metamodels.

T3: **Profiles are better supported in tools.** Metamodeling tools are not yet common on the market (with the exception of [10]), which is quite the opposite to UML modeling tools. Defining profiles is specified in the UML specification and thus is supported in the growing number of UML tools, not requiring special tools.

T4: **Profiles are better for smaller solutions.** For a small language customization (i.e. when the customization requires only making a single certain language construct more precise in a given context), the profiles are possibly better since they provide means of refining single existing constructs.

T5: **Metamodels are more expressive.** Profiles are a restricted way of metamodeling, and as such are less expressive than metamodels. The mentioned lack of ability of defining new associations between extended model elements is only one of the restrictions of profiles, which makes them suitable only for some of the purposes for which metamodels can be used.

T6: **Metamodels are harder to integrate.** Due to the strict metamodeling principle [13] each element in the model can be an instance of only one element in the language definition. While this limits the integration possibilities of metamodels, it does not affect profiles since several stereotypes can be applied to the same element in the model.

Evaluation of whether these differences are used in the choice between the two language customization mechanisms is one of the goals of our case study.

4. Case study design

Telelogic is a vendor of multiple software development tools, among others a modeling tool for UML – Tau G2. The tool is constructed in such a way that it supports the mechanisms of profiles in the way defined in the UML 2.0 specification and allows extending the metamodel in the tool. Creators of language extensions at

Telelogic often (several times per release, which is usually every other month) need to make the decisions of which is the preferred extension mechanism in a certain situation. The modelers perceive UML as a framework for language definition. Such a perception is similar to the vision of UML as a family of languages as presented in [4] and it is also known as the language product line, since the main – core – language elements are reused by all languages. The existence of the common core allows using both profiles and metamodels for language extension.

Three employees were interviewed in the case study: (i) chief architect of the tool responsible for the decision process; (ii) two modelers responsible for development of profiles and metamodel extensions. The chief architect and the two modelers are actually the whole population of people involved in choosing between profiles and metamodel extensions hence no sampling was conducted. The chief architect is the person who makes the major decisions whether a given product is to be based on a metamodel extension or a profile. His experience in the field of UML and tool development allows us to regard his opinion as the expert.

We facilitated data source triangulation [14] to cross-validate the data from different sources, which were:

- Comparable profiles and metamodel extensions – profiles for interaction overview diagrams, component diagrams, and deployment diagrams; metamodel extensions for activity diagrams and Java add-in. They were used to obtain data on their size and to verify whether the products can be directly compared.
- The specification of requirements and the high level designs of the products which (i) provided the necessary information to characterize the products, (ii) ensure compatibility in the scope, purpose and size, and (iii) were used to explore purposes of customizing UML at Telelogic.
- Fault database – to investigate the quality of the products. The faults were obtained by querying the database. The data available in the fault database consists of the faults found in the software, their severity and history of their fixes.
- Decision process – obtained during the interviews – was used to elaborate industrial criteria considered while choosing between profiles and metamodel extensions.

It should be noted that diagrams are treated uniformly with other modeling elements in Tau G2, which allowed us to use the diagrams as objects of the study without any threat to generality. In this study we studied metamodels and profiles which were close to each other as they were: (i) developed by the same modelers in the same tool; (ii) found to provide similar functionality; (iii) were of similar complexity. Due to the above we were able to compare the quality of the different products in spite of the fact that they are not created to provide identical functionality. With this respect we see these products as *sister projects* [15] typical for the studied company and representative for other companies.

In our study we pose the following research questions:

RQ$_1$: *What are the criteria used in choosing between profiles and metamodel extensions in order to customize a modeling language in industry and how important is each criterion?*

RQ$_2$: *Is there a difference in the quality of products based on profiles and metamodel extensions?*

We pose the first research question – RQ$_1$ – in order to get insight into the decision process from the software engineering perspective. The differences between metamodel extensions and profiles in the expressive power and the degree to which they affect the other elements of the language – i.e. how deeply they alter the base language (which is important for metamodel extensions) – can have implications on the quality of the products based on these extensions. Therefore, the second question – RQ_2 – is posed. The quality was measured as the number of reported faults (along with severity) in the behavior of the product as observed during the testing phase and by users of the tool.

We used several methods for collecting the data, thus facilitating method triangulation [14]. The methods are: (i) interviews with chief architect and modelers, (ii) investigations of documents and fault database, and (iii) a prioritization technique – one hundred dollar test [16]. The interviews were in the form of a structured questionnaire (to verify the criteria) with additional open questions (to complement the criteria). In particular the questions to the chief architect were about: the process of choosing between profiles and metamodel extensions, resources required and available for development of the extensions, and differences between processes of developing profiles and metamodel extensions. The questions for the modelers contain aspects related to products development: the purposes of customization of UML at the company, the process of developing profiles and metamodel extensions, quality assurance of the developed products, and development effort required to develop profiles and metamodel extensions.

Data was collected sequentially during the case study during three months. The interviews with the chief architect and one modeler as well as the fault database investigation were conducted during a one day visit to the company headquarters. The prioritization of the industrial criteria was done by the chief architect a week later via e-mail. The interview with the second modeler was performed after one and a half months (due to summer holidays). During the interviews, documentation of the requirements for extensions and specification of the metamodel and profiles to be developed were provided by the interviewees. The fault database was studied during the first visit and it was later checked (during the last visit) that there were no additional faults reported. After the study, the results were discussed during a half-day workshop in order to verify whether the conclusions drawn and the obtained data were valid. During that workshop we performed a simple experiment to check whether the views on language customization are consistent among all subjects. The results were positive – the subjects had uniform views.

5. Results

The results of the study are presented according to the research questions posed in the case study. The results are analyzed in a qualitative way.

5.1. Criteria and prioritization

In the course of the interviews and the subsequent data analysis it appeared that there are two levels at which the criteria are considered: high level (business level) and low level (technical level). The two levels of criteria introduced in the interview are equally important while choosing between the two ways of customizing the language. The business value, i.e. market related elements, are considered at the business level. They relate to the following question – "Which way of customizing the language would be more profitable?" The technical criteria are considered at the lower level and they relate to the following question – "Can we do it in a certain way?"

The criteria presented in this section are numbered I1 to I9 (the letter *I* indicates that they were identified in the course of the industrial case study). The process of making the decision consists of two phases – first, the business level criteria are considered and then the technical level criteria are considered.

I1: **User expectations**. The expectations can explicitly state that one of the mechanisms should be used or implicitly require that a specific one should be used. The expectations might be altered if other criteria point to the alternative extension mechanism.

I2: **Cost** is virtually the most important from the perspective of profits that the company provisions from the customization. The cost is measured in person-hours. The considered aspect of cost is how much the development of the product would cost if it was to be included in the next release.

I3: **UML compliance** is considered since it affects the level of compatibility with the UML 2.0 language and other tools that implement it. The level of compatibility with the UML standard is taken into considerations by customers of Telelogic[1].

The set of technical criteria consists of criteria I4 – I9.

I4: **Limitations of Tau** are considered, although the company is itself the manufacturer of Tau, since there is a constant non-functional requirement – the tool's extendable architecture. The decision taken should aim at increasing extensibility of the architecture.

I5: **Expression power of the approach (includes the limitations of the approaches)**. This criterion corresponds to difference T5 which states that metamodels are more expressive than profiles.

I6: **Possibility of integration of a profile/metamodel with other profiles/metamodels**. This criterion corresponds directly to the difference T6.

I7: **Versioning of the product to be developed** – is considered since according to the company's configuration management practices metamodels' versions need to be controlled while the profiles required less strict control over versions.

I8: **Effort required so that the product can be included in the next release** – is considered on this level although it is not strictly a technical criterion. Nevertheless, since each product influences the components of the tool to a different extent, the effort required to include the product (including the effort of integration into the architecture) is considered. The question that is asked based on this criterion is how much effort the product would require if it was to be included in the next release and whether such resources are available. This criterion is derived from difference T1 which states that profiles are easier to construct, but it complements it with a software engineering perspective – the

[1] The metamodel of UML used in the tool differs at some places from the standard UML metamodel which might be an effect of choosing different extension mechanisms in particular situations.

effort required to include the product in the next release. It also relates to the difference in reusability T2 as profiles reuse more than metamodels.

I9: Knowledge known by the persons who potentially can make the work. The availability of staff that can actually develop each extension determines whether it is feasible to include the product in the desired release (c.f. section 5.3).

As it can be observed in the criteria, they are focused on the complete product based on the language customizations. The aspects of development effort, knowledge required to develop the products, versioning and the business criteria have not been identified previously in the literature.

Although it seems that T3 is similar to I4, there are differences between them. While T3 relates to the way in which the metamodels are defined (and the lack of support for it) I4 (limitations of Tau) relates to the fact that the architecture should be extendable. The theoretical difference is very strict – either there is a support for metamodel extensions or not. The criterion I4 is a "relaxed" version of criterion T3. For companies adopting effective usage of models, however, it is a significant criterion since one of the factors determining the success of the introduction of customized way of modeling is the evolution in company's way of working (c.f. [3, Factors 13-14]) which can be seen as a limitation on the introduction of the extension (or on the architecture of enterprise tools).

Criterion I7 is a new one and relates to configuration management, which is part of software engineering, but it is not a theoretical difference between profiles and metamodels. The last new criterion is the knowledge of the employees who potentially can do the work (I9). Once again this criterion is important for software development companies which usually do not possess extensive expertise in language engineering. This criterion seems to be related to the overall quality of the products developed based on the language extensions.

It should also be noted that the interviewee regarded some of the theoretical criteria as not important. In particular the interviewee has not considered size of the customization as one of the criteria in making the choice although it was identified in literature (c.f. difference T4). The prioritizations at both levels (Table 1) were done with the same technique. The prioritization was conducted by the chief architect as he is the person making choices between metamodel extensions and profiles in most cases. Modelers were consulted after the study whether the percentages reflect their opinions.

Level	Importance [%]	Criterion
Business	40	I1: User expectations
	40	I2: Cost
	20	I3: UML compliance
Technical	25	I5: Expression power of the approach
	25	I9: Knowledge known by the persons who potentially can perform the work
	20	I4: Limitations of Tau
	20	I8: Effort required so that the product can be included in the next release
	5	I6: Possibility of integration
	5	I7: Versioning of the product to be developed

Table 1. Prioritization results

The low importance of the UML compliance indicates that in industry the UML compliance is not as important as the business value. The low importance of the criterion shows that it does not have a decisive influence on the decision although it is considered due to the market-driven nature of software development at Telelogic. The equal importance of user expectations and cost indicate a constant trade-off between the expectations of the product and the cost that is required to implement it in the next release. High expectations can be reconsidered if the cost of their fulfillment is significant.

The technical criteria prioritization reflects the importance of technology related issues. There is no single decisive factor whether to choose profiles or metamodels but the most important criteria are the expressiveness of the approaches and the knowledge required for development. These criteria are related to both the difference between products and the software engineering perspective on the difference – knowledge required to develop them. The issues of knowledge required to customize the language are related to several factors determining successful adoption of customized modeling language [3].

Slightly lower importance (20%) characterizes the limitations of Tau (I4) and the effort required for development and integration of the product into the next release (I8). The extensible architecture requirement limits to some extent the possibilities of extending the metamodel. An equally important aspect is the effort required to develop the product in a given time – since no delays are allowed. Any potential delays of product development are compensated in decreasing the size of the product – i.e. if there is not enough time, the time is not extended, but functionality is being restricted or if it is not possible, the integration is postponed until a later release. Issues related to the delivery of the product were also observed in the previous study – in the course of the studied industrial MDA realization there was an internal deadline: the profiles were to be finished before developing transformations.

The low priority of the criterion related to integration issues (I6, 5%) is in contradiction to the theoretical analysis, in which the integration issues were indicated as one of the main drawbacks of using different metamodels in the course of model driven software development. From the practical perspective in the company the integration issue is not as important as there is always only one metamodel used at a time in the tool. So the only integration issue is how the new elements are incorporated in the existing metamodel and what the effects of this activity are, which are reflected in criteria I4 and I5 (together 45%).

The names of these industrial criteria seem to be specific for the studied company, but they represent issues that are not specific for Telelogic. Three of the criteria need generalization as presented in Table 2.

Criterion	Telelogic specific name	General name
I1	User expectations	Requirements for customization
I4	Limitations of Tau	Limitations of the tool to support each mechanism
I8	Effort required so that the product is included in the next release	Effort required so that the customization is finished before it is needed

Table 2. Generalization of the names of the criteria

Criterion I1 can be renamed as the requirements for the customization are essentially the user expectations for the extension. Criterion I4 is renamed since it is important to keep in mind that the customization is done in the context of modeling tools (not metamodeling) and these – even though they can be metamodel driven – have some limitations (c.f. [17]). Thus we advocate for making the criterion closer to the original understanding stemming from theoretical difference T3. Criterion I8 can be renamed as the given deadline for the customization of the language in a company is defined by the time in which the customization is to be used in the course of software development (c.f. [18]).

5.2. Quality

The quality of the products is measured as the number of faults reported in the fault database, as presented in Table 3.

Severity	**Metamodels**		**Profiles**		
	Activity	Java	Deployment	Interaction overview	Component
1	1	3	1	0	0
2	12	10	0	1	1
3	58	7	2	4	0
4	6	2	1	0	0
5	9	4	0	4	2
6	2	2	0	0	0
Non-classified	3	1	1	2	0
Total number of faults	91	29	5	11	3
Avg. Severity	3.2	3	2.8	3.8	4
Size	21	37	11	7	13

Table 3. Normalized number and severities of faults and sizes of the extensions

The faults correspond to incompliance with user requirements and erroneous behavior of the products based on the extensions which can be regarded as the measure of quality as defined in [19]. The extensions included which are analyzed in this section are comparable in terms of the implemented functionality (c.f. section 4.2) and therefore analyzed in this paper. The normalized number of faults for the studied extensions are presented in Table 3, grouped according to their severity (1 – catastrophic, 2 – critical, 3 – non-critical, 4 – minor, 5 – suggestion, 6 – question).

In total it seems that the products based on profiles are of better quality (less faults reported in the fault database). We found (and it was confirmed by the interviewees) that the elements potentially contributing to the differences are: (i) high level of reusability that is facilitated by profiles; (ii) technical differences in introducing profiles and changes in the metamodel in the tool. The technical differences in the way the extensions are introduce stem from the fact that when developing a profile certain support is already provided (e.g. adding graphical icons to stereotypes). In the case of metamodel extensions, these mechanisms are not present and therefore, similar functionality (e.g. representing a model element on a diagram) needs to be implemented in the final product. One of the main technical differences seems to be the way in which the extensions are integrated into the tool. In the interview with the

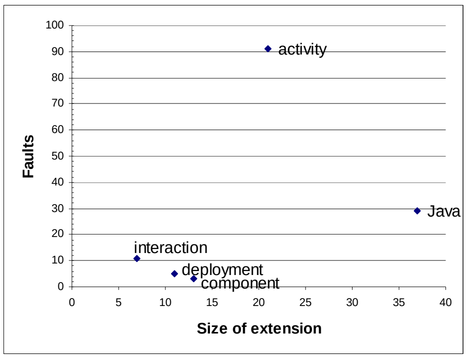

Fig. 3. Scatter plot of size and number of faults

modelers on the process of developing profiles and metamodels it appeared that there are minor differences which do not have an influence on the quality directly. It is the fundamental integration issues that potentially affect the quality. However, the fault data in the fault database does not allow distinguishing whether the fault is caused by the implementation (integration) or by an extension to the metamodel itself (i.e. a modeling error). Although it might be initially perceived as a confounding factor in our study it is not since the additional implementation is expected to be present while customizing a language (as it was the case of Volvo IT [3]). The source code often supports the extensions.

The scatter graph for size and number of faults is shown in Figure 2. It seems that the faults are not related to size (wide spread of the points), for example the largest product – the Java add-in contains fewer faults than the smaller activity diagrams. This in turn indicates that the quality of the product is not dependent on the size of the extension measured as the number of elements in it. This might explain why the size is not considered during the decision process.

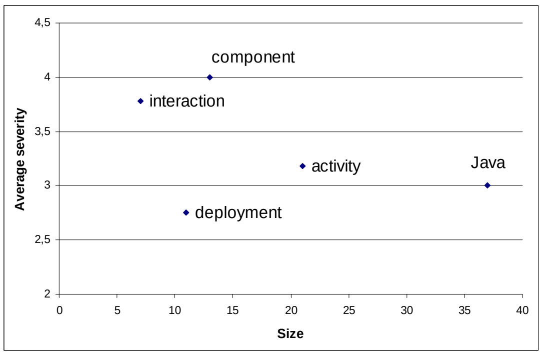

Fig. 2. Scatter plot of size and average severities

Another important aspect in the analysis is the relationship between the size and the average severity, which is shown in Figure 3. The small number of data points (only five products) does not allow calculating the correlations coefficients. Although it would be possible to include several more profiles or metamodel extensions, it was our intention to include only the products which are directly comparable. Furthermore, the lack of visible relationship between the size of the extension and the quality of the products could be caused by: (i) measures of size which are not done with stereotype and metamodel specific metrics (these are part of our current work); (ii) implementation of an additional functionality specific for the product – e.g. support for graphical representation of new model elements in case of metamodel extensions. In order to minimize the threat of drawing conclusions based on descriptive statistics while evaluating a very small set of profiles and metamodel extensions our respondents were consulted after the study. Our interpretation of the results is consistent with their expert opinion.

5.3. Effort and resources

The development effort for creating profiles and metamodel extensions determines the cost of the product and has a potential influence on quality. In the course of the interviews it was found that developing an extension to the metamodel takes approximately three to four weeks depending on the size of the extension and usually involves several engineers (one modeler creating the metamodel extension and several developers integrating the extension with affected components of the tool). The development of a profile takes approximately one week and usually is done by a single modeler. According to the chief architect's experience extending the metamodel requires ten times more effort than developing a profile. This impacts the development effort. In the project (or release) planning phase, the potential influences of the metamodel extension on other components need to be considered.

	Metamodel extension	**Profile**
Development time (calendar time)	3-4 weeks	1 week
Resources available – who can potentially do the work	2 modelers	5 modelers

Table 4. Development time and resources available for developing profiles and metamodel extensions

There is a difference between the metamodels and profiles in terms of resources available for their development. There are five modelers who have the knowledge and expertise to develop profiles although only two of them have enough expertise to fully develop metamodel extensions. The two modelers who develop both profiles and metamodels were interviewed. The remaining three modelers develop very small profiles on irregular basis and thus were not included in the study.

6. Validity evaluation

In evaluating the validity of the presented study we follow the schema presented in [20], describing four kinds of validity threats to empirical studies.

The main threat to the *external validity* is related to the fact that the company might not be representative for the population of companies in which decisions on the choice between profiles and metamodels are taken. The results can be transferred to other companies customizing their modeling language (e.g. Volvo IT, ABB Robotics); particularly as the case study was performed based on the needs identified in these enterprises. The sizes of products studied in this paper were similar to the sizes of products used in the previous case study (c.f. [3]) which indicates that the objects of this case study are representative not only for Telelogic.

The main *internal validity* threat to the study is there is a risk that while measuring the quality of products the metamodel and the profiles under study were specific and the result is only due to chance factors. To minimize this threat, the most representative sister profiles and metamodel extensions were chosen. All interviewees provided the same estimations of differences in development effort between profiles and metamodel extensions. This similarity increases the validity of their claims.

We have identified a *construct validity* threat. Measuring of the quality can be confounded by the lack of distinction whether the problems reported arise from the modeling issues or the code that accompanies the extensions. Nevertheless, we have found that the accompanying code is often required to introduce changes into the metamodel so the influence of the code on quality is expected to be similar for other metamodel extensions and profiles (even in other companies).

As a *conclusion validity* threat we see the lack of inferential statistics used in the analysis of results caused by a small number of data points even though we have interviewed the whole population of modelers at Telelogic who develop profiles and metamodels in their daily work. The conclusions are similar to observations in our previous case study [3] which increases their validity. The small number of persons involved in the decision process is a representative situation, which was also observed in our previous case study.

7. Conclusions

In the case study presented in the paper we investigated a company with extensive expertise on language customization. The case study was stimulated by the need identified in our previous case studies on issues related to industrial adoption of model-driven software development [3] in which the language customization is a prerequisite for automation of software development. The customizations are the basis for the development of products, which usually are customized modeling tools. In choosing the appropriate mechanism, the criteria should consider the whole products and not only the differences between the mechanisms, which was the case so far. In this case study the identified set of nine criteria for choosing between these mechanisms complement an existing set of theoretically elaborated criteria existing in literature. The criteria allow making the decision in considering two different levels – business and technical. Three industrial criteria at the business level relate to the issues of profitability of using the language customization mechanism (which is crucial in industrial software development). The remaining industrial criteria at the technical level allow assessing whether the chosen extension is technically sound, feasible, and can be the basis of a product.

In addition to identifying the criteria, in the case study the quality of a set of sister products was investigated. The results show that the products based on metamodel extensions usually have more faults and they require up to ten times more effort to develop than profiles. Together with the criteria the quality investigations provide a basis for taking informed decisions on the way in which the modeling practices in enterprises can be improved based on language customization.

During the study we have also observed that the quality of profiles depends on the quality of the base metamodel. If the metamodel is well-suited for the customization at hand, then the profiles are easier to construct. This observation has not been described in the paper, and its further investigation will be done in our upcoming research.

Acknowledgments
We would like to thank Telelogic AB, Sweden for letting us perform the study. We would like to thank Dr. Ludwik Kuzniarz for his valuable comments on the paper.

References

1. Object Management Group: Unified Modeling Language Specification: Infrastructure version 2.0, Object Management Group (2003).
2. Miller, J., Mukerji, J.: MDA Guide, Object Management Group (2003).
3. Staron, M., Kuzniarz, L., Wallin, L.: A Case Study on Industrial MDA Realization - Determinants of Effectiveness, Nordic Journal of Computing **11** (2004) 254-278.
4. Evans, A., Maskeri, G., Sammut, P., Willians, J.S.: Building Families of Languages for Model-Driven System Dev, Workshop in Sw. Model Eng., San Francisco, CA (2003).
5. Centre for Reliable Software Technology: Coral Modeling Framework, CREST (2004).
6. Kleppe, A.G., Warmer, J.B., Bast, W.: MDA explained: the model driven architecture: practice and promise, Addison-Wesley, Boston (2003).
7. Mellor, S.J.: Make models be assets, Communications of the ACM **45** (2002) 76-78.
8. De Miguel, M., Jourdan, J., Salicki, S.: Practical Experiences in the Application of MDA. In: Stevens, P., Whittle, J., Booch, G. (eds.): The 6th Int. Conf. on UML, Vol. 2460, Springer-Verlag (2002) 128-139.
9. Greenfield, J., Short, K.: Software factories: assembling applications with patterns, models, frameworks, and tools, Wiley, Indianapolis, IN, USA (2004).
10. Clark, T., Evans, A., Sammut, P., Willans, J.: Applied Metamodeling - A Foundation for Language Driven Development, Xactium (2004).
11. Schleicher, A., Westfechtel, B.: Beyond stereotyping: metamodeling approaches for the UML. Hawaii Int. Conf. on System Sc., IEEE Comp. Soc, Maui, HI, USA (2001) 10-17.
12. Berner, S., Glinz, M., Joos, S.: A classification of stereotypes for object-oriented modeling languages, 2nd Int. Conf. on UML, Fort Collins, CO, USA (1999) 249-264.
13. Atkinson, C., Kühne, T.: Profiles in a strict metamodeling framework, Science of Comp. Programming **44** (2002) 5-22.
14. Martella, R.C., Nelson, R., Marchand-Martella, N.E.: Research methods: learning to become a critical research consumer, Allyn & Bacon, Boston (1999).
15. Fenton, N.E., Pfleeger, S.L.: Software metrics: a rigorous and practical approach, International Thomson Computer Press, London (1996).
16. Leffingwell, D., Widrig, D.: Managing software requirements: a unified approach, Addison-Wesley, Reading, MA (2000).
17. Alanen, M., Porres, I.: The Coral Modeling Framework, In: Koskimies, K., Kuzniarz, L., Lilius, J., Porres, I. (eds.): 2nd Nordic Workshop on UML, Turku (2004) 93-98.
18. Staron, M., Kuzniarz, L., Wallin, L.: Factors Determining Effective Realization of MDA in Industry, In: Koskimies, K., Kuzniarz , L., Lilius, J., Porres, I. (eds.): 2nd Nordic Workshop on UML, Turku, Finland (2004) 79-91.
19. IEEE: Standard glossary of sw. eng. terminology, Std 610.12-1990, New York (1990) 84.
20. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslèn, A.: Experimentation in Sw. Eng.: An Introduction, Kluwer, Boston MA (2000).