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

J. Münch and M. Vierimaa (Eds.): PROFES 2006, LNCS 4034, pp. 177 – 191, 2006.

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