Model Driven Development - TDT4250
A Summary

Alexander Nossum

November 27, 2007

Abstract

Most important:

- SEQUAL elements
- SEQUAL relationships
- DSM/DSL concept

Contents

1 SEQUAL - Quality evaluation framework

SEQUAL is a framework for evaluating the quality of both models and languages in the context of software development. The framework base itself on a set of elements involved in a model and in a language. Key concept for the framework is to evaluate the relationship between these elements, which typically includes some form of perception of knowledge. The elements are:

- Social actor explicit knowledge
- Social actor interpretation
- Language extension
- Model externalization
- Technical actor (tools) interpretation
- Modeling domain
- Goals of modeling

Further on, we discuss the quality evaluations based on relationship between these elements. With the evaluation of models and evaluation of languages.

1.1 Quality of models

1.1.1 Physical quality

The physical quality of a model is in essence how the model is externalized or in a sense manifested. Whether it is on paper, electronic ... and how this affects attributes such as; ability of distribution, possibility for interpretation, change management, backup etc. These attributes can be grouped into two key components: *Persistence* and *Availability*.

Persistence is in essence how well the physical model is saved. Including different versions, changes and so on.

Availability describes how easy it is for the audience to get hold of the model. An important factor is the ease of distribution - which is heavily dependant on the externalization of the model.

Physical quality of a model relates directly to the *Social actor, explicit knowledge* element of the framework.

1.1.2 Empirical quality

Empirical quality is the evaluation of how easy it is to gain knowledge (externalize) from the model. The externalization of knowledge is dependant on how the actual model is designed. Several factors are relevant for this, symbology, use of colors, layout of symbols (crossing lines, squared vs. wobbly boxes) and so on. The empirical quality is heavily dependant on subjective factors hence, there are no formal rules only guidelines for evaluating the quality.

As implied, the empirical quality relates to the *model externalization*. Other attributes relates the externalized model to the actual perception value for the audience. Social quality (1.1.6) and Pragmatic quality (1.1.5) handles the concrete perception and values that are obtained by the model audience.

1.1.3 Syntactic quality

Syntactical quality of a model relates to the relationship between; *Language extension* and *Model externalisation*, and is as such, the ability the model has for externalization (manifestation/capturing).

Since the syntactic quality derives from the language extension it can be formalized from the syntactic rules of the language in use. The overall, *and only*, goal is: **syntactic correctness**. If the model is in complete compliance with the syntactic rules of the language - the model obtains a high syntactic quality.

1.1.4 Semantic and perceived semantic quality

Semantic quality is the completeness between *the domain* and *the model* in a sense;

> The completeness of the captured knowledge between domain and model.

There are mainly two goals for semantic quality; *Validity* and *completeness*. Since a model can capture knowledge that actually does not exist in the domain, we need *validity between the domain and model*. On the other hand the main interest for a model is to capture the whole, correct, knowledge area of the domain - ensuring that *the completeness between domain and model are fulfilled*.

Semantics are not easy to formalize since it base itself mostly on subjective factors. The introduction of perception and thus ***perceived semantic*** *validity and completeness* is necessary for the evaluation of the model from the audience. Perceived semantic quality is hence not related directly to the domain⇔model but *between the Social actor explicit* ***knowledge*** *and the Social actor* ***interpretation***.

1.1.5 Pragmatic quality

In the evaluation of pragmatic quality - the practical considerations are taken into account. A main goal for achieving pragmatic quality is **comprehension** from the different actors. The actors are of different kind, both human and abstract. This relates the pragmatic quality from the **model externalization** to; **modeling domain, social actor explicit knowledge, social actor interpretation** and **technical actor interpretation**. Induced from this we see that *interpretation* is a key concept in pragmatic quality and hence the *concrete comprehension*. The learning aspect is therefore an important factor for evaluating and improving the pragmatic quality.

Focusing on the elements, mentioned earlier, that are related with the pragmatic quality, we can introduce some concepts that are necessary in obtaining pragmatic quality.

Social actor interpretation is essentially human understanding of the model. How well is the human actor able to understand the model.

Technical actor interpretation is the ability for tools to interpret the model. Typical the correctness between the tool interpretation and the actual meaning described in the model.

Modeling domain is the models ability for changing/affecting the domain it models. Both positive and negative directions are included in the term change.

1.1.6 Social quality

Social quality is related to the *social actor interpretation* by itself. The overall goal for obtaining social quality is **agreement**. Agreement in the sense that social actors agree on their interpretations on different levels of agreement.

1.1.7 Organizational quality

The organizational quality is related to the element; **goals of modeling**. Evaluating whether the overall goals for the modeling performed is achieved. Overall goals, including organizational goals and concrete modeling goals.

In real life, the ability in obtaining all these goals perfectly in performing the modeling task is *unrealistic*. Taken this into account, the term **feasibility** is introduced. Feasibility, used as a realistic level of "perfectness" in obtaining **enough perfection** of the goals.

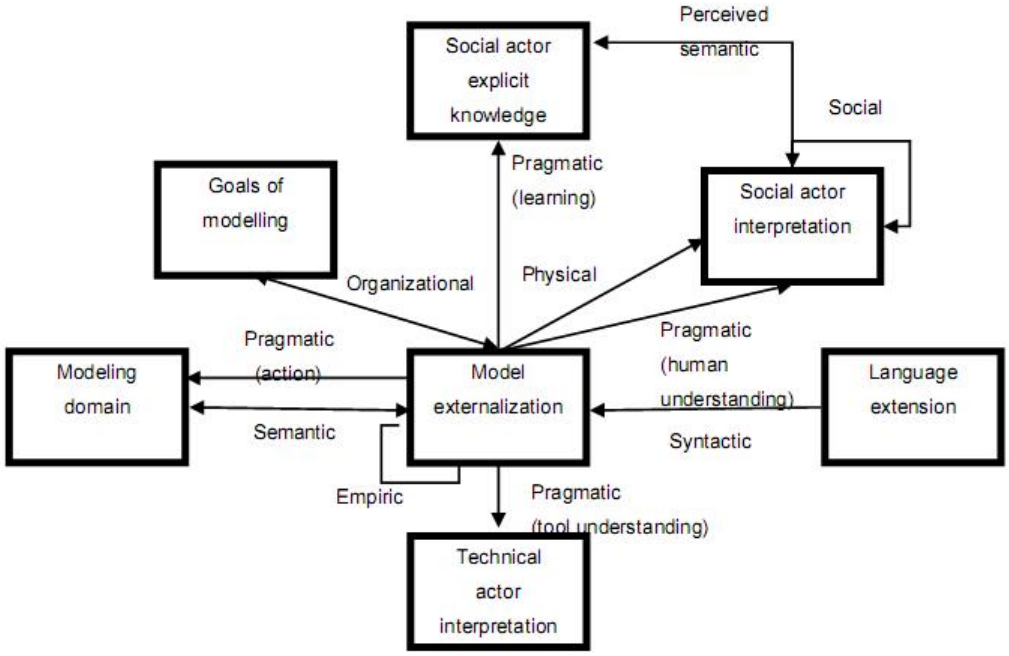

Figure 1: SEQUAL framework for evaluating model quality

1.2 Quality of languages

The language is the main basis for any model and it's important that one can evaluate the language in terms of quality. SEQUAL proposes a framework for this. Using the same approach as for the evaluation of models, with qualities related to the relationship between elements. The relationships are of some greater intuitivity when it comes to languages, with less abstract evaluation terms. The proposed evaluating factors are:

Organizational appropriateness

Tool appropriateness

Participant and modeler appropriateness

Comprehensibility appropriateness

Domain appropriateness

All factors are spun from the *language extension* element in the framework. This is very natural since it is the language itself that is under evaluation. Further on we see that all factors are based on *appropriateness*. From this we can derive that

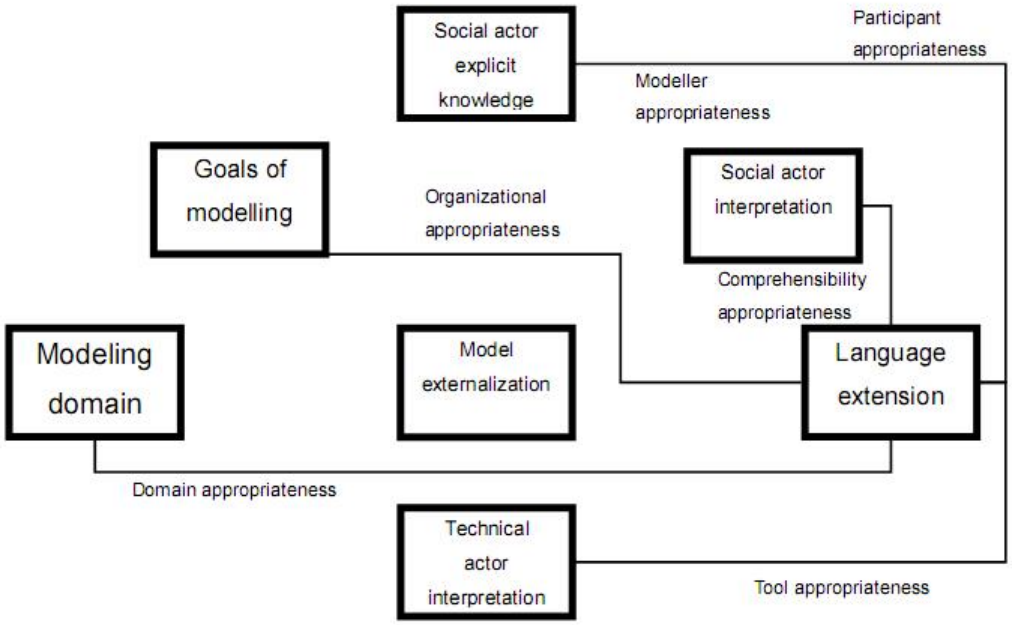

Figure 2: SEQUAL framework for evaluating language quality

the main driver for quality in a language, is how well it is suited for the task. The framework only elaborates more specifically this question.

2 MDD and MDA

2.1 MDD - Model Driven Development

Model Driven Development aim is to model a situation with two main purposes; *communication* and *development of a system*. Often a model describes a process of some sort and the method used for execution of the process.

In general MDD exhibits the tradition of *seperation of specifications* in different abstraction levels. A typical seperation of abstraction levels are (listed in decreasing degree of abstraction):

CIM - Computational Independant Models Which are indedependant from any execution and computation of the modeled process. This is typically called *Enterprise model* since it's main use is for the management of the modeled process.

PIM - Plattform Independant Model is a model with such a high level that it is independant of the plattform it will execute on. The model takes com-

putation aspects into consideration, often in a very general and abstract matter.

PSM - Plattform Specific Model Extends the PIM further on, elaborating more specifically on the computational part of the model. The trend in MDD is to try to generate the PSM automatically. This is fairly difficult and approaches made are not very successful, yet.

Code is the final level of abstraction - and in a sense is not under the model domain. It is mentioned since models often culminates into code of some sort. Traditionally very specific code either generated or coded manually. A trend seen in MDD is not to generate code at all, but execute the model as-is. Further discussed under DSL/DSM.

2.2 MDA - Model Driven Architecture

MDA is an approach for combining different standards, methods and approaches for MDD. Leading to a general framework for model driven development process. In a sense an instantiation of MDD. The OMG coalition is the main driver behind MDA which leads to quite *heavy use of UML* as the standard notation, nevertheless MDA is *not limited* to UML.

As an approach for combining several different notations and development of specific notations MDA introduces a standard called *MOF - Meta Object Facility.* MOF is an abstract modeling language which intention is to describe a common meta-model for a language. With a common meta-model, languages can easily be transformed between themselves, since the main structur (meta) is shared.

The MDA approach has several goals, the main goal is to be able to *create PIMs describing specifications* then transform the PIMs into an implementation supported by transformation knowledge. Often *generating executable PSMs* or *standard plattform specific code.* Increasing the degree of reusability and the usefulness of the model.

The approach is heavily discussed, especially the heavy focus on UML as a notation. UML is extremely general which has lead to a *rich symbology and thousands of pages with specification.* Almost all modeling requires just the use of a small subset of the UML specification which leads to *much work in just finding the right concepts of UML to use.* This problem is refered to as **navigating the meta-muddle**. In addition a MDD-project (almost) always needs to *tailor (tweek) the language* or framework - leading to the same issues as mentioned.

3 DSM/DSL - Domain Specific Modeling

Important terms and concepts:

Meta- and Meta-Meta-model Abstraction levels, meta-meta is very abstract (nodes, edges etc) while meta-model is the "allowed" elements and concepts of a language.

Product line DSM is often used in product lines. Example of a product line is a Nokia cellphone - each model is slightly altered but share the main concept (meta-model).

Model is system Not generating code, but compile the model to execute itself. Typical approach is describing model in XML which can be executed **and** presented as model. Typically using a framework which supports the meta-model very well.

Limited development time DSM, when used correctly, is fast, very fast!

4 BPM - Business Process Modeling

Important concepts

B2B and VE Business2Business and Virtuell Enterprise. Creating "ad-hoc", temporary, business cooperation. Providing needed (requested) *just-in-time* services for customers.

Managing processes Improve, administrate, analyse ...

BPMN Notation specifically made for BPM. Relatively small symbology. Handling the bloated symbology-issue in UML.

BPEL Executable language for BPMN, XML-based.

SOA and BPMN BPMN embraces SOA; use of external services (subprocesses), orchestration, choreography, trust (contracts) ...

WS-* stack Extreme numbers of Web-Service standards. (SOA doesn't kill people. The WS-* stack does.)

WSDL Describing services

WSMO Ontology, Semantic web, heavy use of mediators as connectors, choreography, orchestration ...