

Can Model-driven Engineering solve Requirements Engineering problems?

Gustavo Cabral, gustavo@graco.c.u-tokyo.ac.jp

General System Studies - Graduate School of Arts and Sciences - The University of Tokyo - Japan

Abstract

The use of models in more concrete phases of the software development process is increasing. Most of current MDE researches concentrate on transformation and conformance checking of Computational Models. However, Computational Independent Models (CIM) and their relations to concrete models is still unclear. This paper explores Requirements Engineering problems in the light of MDE. Application domain, requirement and specification definitions are viewed based on MDE concepts, such as metameta-model, metamodel, transformation and ontologies. From this analysis improvements in MDE are proposed so it can handle requirements concepts, such as softgoals.

1 Problems and research goals

Model-driven Engineering (MDE) [2, 3] enables an interesting way to look at software artifacts, which are treated as models. In MDE the development process relies on execution of a sequence of transformations that link model concepts at different levels of abstraction. These transformations are maps between metamodels definitions and do not include any formal proof of refinement. In this case, the refinement correctness lies on the engineer's belief that the specified mapping represents a concept refinement rather than exploring the model's behavioral aspect. This flexibility is actually desired when handling abstract concepts with no yet specified behavior. But, how can we make use of MDE to guarantee that early artifacts present in the Requirements Engineering (RE) discipline are correctly translated into concrete behavioral models (specification)?

Michael Jackson's lexicon of RE [6] says that software development is like building a *machine*. He explains that the problem of building such a machine can be addressed through the understanding of the application domain and the machine's requirements in order to find a solution (specification). Afterwards, it is necessary to prove that the proposed solution actually satisfies the requirements and conforms to the domain constraints; requirement's terminology should consider the environment, the actions from the en-

vironment that control the machine and the effects of the machine actions onto the environment. The domain knowledge is used to support the refinement of requirements into an implementable specification. However, when we look at these artifacts how can they be formally related? Using MDE jargon, how are their metamodels related and how do we know the identified metamodel relations are sufficient for conducting refinement?

If we look at Platform Independent (PIM) and Specific models (PSM) in the Model-driven Architecture (MDA), we find extensive research related to the formalization of these models' behavioral semantics, such as of UML [12]. Some efforts have been done to better describe Computational Independent Models (CIM) in the meta-pyramid [3]. It includes artifacts such as domain models (describing concepts of domains and their interrelations), business models (describing company's rules of business) and requirements. In this context, ontologies promote standardization of concepts allowing the refinement process to be regarded as an engineering discipline. Still, abstract ideas from ontologies are loosely translated into computational concepts.

There is also formal approaches that use predicate logic to formalize CIMs and validate requirements [4]. In this type of approach a set of standard definitions (ontology) must be used so stakeholders and engineers can share conceptualization about the domain. Even though, such approach does not show that the specified solution will satisfy the stakeholders' expectation. When Zave and Jackson [15] discussed the RE problem using indicative descriptions (domain assumption), optative descriptions (requirement), designations and the distinction between definition and assertion, they covered a portion RE fundamental concepts. There are other notions [10], which are related to the communication process between stakeholder and engineers (such as preference) that need to be further analyzed.

Therefore, a revision of MDE foundations is necessary. It is not enough to model environments, requirements and specifications, it is necessary to create mechanisms to allow the evaluation of abstractions so that its quality (satisfaction of expectation) is perceived and measured within some quality space. Again, there remains a problem that such quality evaluation may not be shared; one person may give

a strong favorable evaluation and another may disagree. Because MDE offers a consistent engineering framework it is worthwhile to understand this approach and to verify its adequacy for solving RE problems.

This paper aims to explore the RE problem in the light of MDE and opens a discussion over the revision of MDE taking RE abstractions into consideration. The definition of metamodels and transformations is not sufficient to allow refinement of computational independent models. The use of ontologies in MDE aims the representation of abstract concepts but the RE problem have not been approached. It is necessary to review the current metamodel in MDE definition so *real world* concepts can be represent, such as those from *speech act* theory [13] (directive and commissive acts, for instance) and features from the *communication process*. A further analysis of the core metamodel of MDE will allow the definition of more concise methods to relate metamodels, reformulating the refinement notion in MDE.

2 RE and MDE backgrounds

First let us look at RE concepts and see how they have matured so far. In the paper [15] the RE problems amounts to finding the specification (S), for a given domain knowledge (K), that satisfies the given requirements (R). The problem is solved once the engineer finds S such that $K, S \vdash R$. In this strategy the RE core ontology is formed by the following concepts: environment, machine, designation, definition, indicative and optative descriptions, shared, unshared, environment-controlled and machine-controlled actions, requirement, domain knowledge and specification.

In [10] this characterization is revised and considered incomplete since any found solution (specification) cannot be sound and complete for any realistic system; we tend to continually encounter in practice new information that defeats previously valid conclusions (when requirements are revised or domain assumptions altered). In this view of the RE problem, the core ontology was structured with respect to *speech act* concepts (assertives, directives, commissives, expressives, declarations and representative declarations). It is believed that the *communication process* involving stakeholders and engineers holds essential information that supports the task of finding the correct specification. It proposed that the stakeholders' desires give reason to the system specification correctness; more than one solution may exist and the one that best fits the stakeholders' priorities should be specified. Therefore, this reviewed RE ontology consider nonfunctional requirements: goals (G), quality constraints (Q), softgoals (D) and attitudes (A), in the process of finding the best fitted specification (S), which conforms to domain knowledge constraints (K). Formalizing, $K, S \vdash G, Q, D, A$.

"Soft" requirements [14] may be vague and imprecise,

such as the condition that a system must be "secure", "reliable" or "easy to use", which is the reason why Zave and Jackson did not take them into account in their formalization. They argue that RE is about the satisfaction of goals, but goals do not make a good starting point; if a requirement and its goal are taken, we could see the goal as the "why" for the requirement necessity. And since this kind of analysis is too vague because it depends on the machine usage context it is avoided in their formalization.

Now let us look at MDE features and how it could support RE needs. Chapter 9 of [3] presents a strategy for handling descriptive models, such as domain knowledge and requirements; an important characteristic of models is that it can be descriptive or prescriptive. A CIM describes reality, but reality is not constructed from it. On the other hand, PIMs and PSMs prescribe the structure or behaviour of reality and reality is constructed according to the model. Because part of CIMs are descriptive concepts, *open-world* assumptions (shared conceptualization) defined through an ontology, these models do not allow a complete and final description of the machine: anything that has not been said explicitly is unknown. It is only possible to continue the refinement process when a model is created as a prescription for the systems, for which a *closed-world* assumption is required, allowing us to call it a specification model. Thus, ontologies use a form of partial description or underspecification as an important means of abstraction, however, they only support the definition of concepts and their interrelations; there is no mention of transformations (refinement) that handles nonfunctional requirements in MDE [1], specially taking *speech acts* into consideration.

Whether nonfunctional requirements are vague or not they have a fundamental role in the process of finding the correct specification. Any goal and any quality constraint is verifiable in the sense that the software engineer can check whether the system satisfies it. In the case of softgoals, the engineer have the attitude of evaluating definitions according to some degree of favor or disfavor. Definitions may have an *optionality* (compulsory or optional) or a *preference* role for the stakeholder. It is compulsory if a specification that does not account for that constraint; otherwise, it is optional. The preference role defines a priority order. Therefore, if we want MDE to support RE, model's elements should not only describe or prescribe reality, they should include a reason, a purpose for their existence.

3 Reviewing MDE to support RE

A common basis to explain a wide variety of programming paradigms is the unifying theories of programming (UTP) [5]. It uses an alphabetised relational calculus, presented in a predicative style, to define languages' semantics. It is based on other theories, such as set, relation and

logic. This bottom-up approach explain the design of languages, which allow us to specify and implement machines. In a top-down (from domain knowledge, passing through requirements, to specification) approach, the theory should explore *real-world* concepts and mechanisms to allow proof of assertions and refinements.

As explained in [3] in MDE metamodels are associated with a set of constraints and it is possible to verify assertions using transformations and its engines [8]. Models, which are *instances of* metamodels, are correct if they respect predefined constraints. In RE, domain assumptions and requirements describe *is-described-by* relations; only specifications are prescriptive models, since they constraints under the *closed-world* assumption. If we consider the latest RE core ontology [10], attitudes are not present in MDE's CI pointview. The notions of optionality (to be compulsory or optional) and preference are not used to define models constraints. Thus, MDE's core features, the metametamodel and the transformation mechanism, need to be reviewed to include these "new" abstract core concepts.

3.1 The core metamodel concepts

The definition of a class (that can extend other classes) with attributes and references and of datatypes is indeed clear and succinct. It has the characteristic of conforming to itself and, as a metalanguage, it is flexible enough at least for defining non attitude concepts. As explained, the RE task involves more abstract information, such as preferences, which are orthogonal to concepts. These desires concern all definitions allowing us to find more than one solution (specification, or plans) for the same set of requirements (goals, quality constraints and softgoals). That means a model is consistent or not depending on its metamodel's constraint and the defined stakeholders' attitude (level of satisfaction) over model elements.

Using current MDE features we model different solutions for the same problem when we make changes in the current version of a model. Refinement of models occurs with the tacit behavior of always caring about satisfaction criteria. The MDE itself lacks of support to express these satisfaction factors. In order words, in a model space defined by a metamodel and its constraints there is a subspace of models that are consistent according to the defined constraints. But which one should be used as a candidate model to be carried on in the MDE transformation (refinement) process? The addition of optionality and preference information to the metamodel would assist the selection of the most fitted solution. This "new" theory would still consider a model valid even if a softgoal is not satisfied; it is more important to obey a more prioritized softgoal when two or more softgoals are conflicting. This extra-information gives the model a conscious quality well-formation characteris-

tic regarding to stakeholders' satisfaction. The final model, after transformations, does not only follow its theory's constraints but has a reason (why) to be as it is.

This essential feature, in the eyes of RE, could be available through the introduction of new classes in the core metamodel of languages such as KM3 [9]. These satisfaction concepts (alternatives, arguments, claims, positions) can be reused from studies of Goal-oriented RE [11]. The formalization of these concepts inside the MDE is part of our on-going research. The introduction of new classes is a possibility, but there are other not so invasive options, such as the use of Aspect-oriented modeling [7]. However, such strategies have their own problems.

3.2 Transformation with satisfaction

The MDE development process is defined as a series of transformations over models where each step is a refinement that decreases the level of abstraction producing more concrete models. Other transformation scenarios in MDE are refactoring, reverse engineering and data translation between heterogeneous data sources. The issue here, according to the engineer's attitude view, is that when a refinement transformation yields a more concrete model there is no formal concern that backs up the veracity of the defined refinement, besides the fact that the transformation linked concepts from one metamodel to concepts of the more concrete metamodel. There is no proof that it is an actual refinement relation. When dealing with requirements this is specially visible, since the gaps between the metamodels concepts is wide and exposes a feeling of uncertainty regarding the correctness of the defined links.

Transformation rules should not only relate definition from different metamodels. They should also take into account the specified desires so the generation of the target model can fulfill softgoals. The execution of the "correct" transformation rule should be guided by the preferences specified at the metamodel level. The mapping defined by transformations would use preferences from source and target metamodels allowing the occurrence of optimizations when transformation rules are selected and executed by a transformation engine. This strategy changes MDE in the sense that transformations would contain explanations about the reason why the output model was generated that way. This mechanism could use the know-how of programming language semantics, and theories such as UTP, to establish the necessary formalization and enhancement in the MDE transformation metamodel. These improvements also affects the use of transformations with computational models; the preference factor (i.e. performance, modularity and security) supports choices at designing or coding stages.

4 Current research status

MDE is a variant of the refinement-based software development in which models are connected in a systematic way. However, the definition, manipulation and refinement of information (models) does not seem to currently solve RE problems. To understand an application domain and a problem and then write about the solution is complex. To come up with a solution (specification) involves not only creativity and experience, but it is necessary to give reason to the decision choice based on the stakeholders' preferences. If we are not sure that the proposed solution is ideal it is because it was not given enough effort to specify the "solution's why". To understand the creative process of making software is to comprehend the reason of the choices that led to the final solution. This is one of the reasons why the RE problem was revisited in first place.

Our initial research goal was to define a strategy to improve requirements definition through the detection of inconsistencies. This research involved the manipulation of requirements using a natural language parser, sentence pattern searching techniques and MDE tools. The task of reasoning over some types of sentences patterns, using MDE concepts, gave us some clues about the MDE lack of support. This led us to investigate the role of ontologies in MDE and current proposals for RE's core concepts. Recent researches about the Semantic Web have enhanced the understanding about the *open-world* and promotes the use of ontologies. This is a good starting point to improve the meta-pyramid; there is a good number of tools to assist in the manipulation and visualization of models [8, 9]. However, changes in the MDE core definitions will impact the use of these tools as it is.

At the current stage no complete case study was accomplish to validate the proposed changes in MDE. The effort was focused on developing a metamodel representation of natural language text and transformations to validate this metamodel instances. This validation, which is the actual detection of inconsistencies in requirements, is implemented through the recognition of phrasal structures and grammatical relations on sentences; these relations define the role of terms in a sentence. Since some types of sentences do not have an assertive or a declarative role, for instance, they could not be transformed into known MDE concepts. This suggested a better understanding of the problem of reasoning about *speech acts*, which could be adopted as RE's core ontology. The use of stakeholders' attitudes in the definition of *upper ontologies* and *domain ontologies* is reasonable since these artifacts are the initial input for the requirement writing task.

In the technical field, the use of MDE tools, such as ATL [8], was important to enhance the hands-on skills, which involved the study of OCL constructors, in order to specify

metamodel's constraint and transformation rules. This experience allowed us to realize how MDE concepts are unsuitable to define abstract concepts and refinement, and it is giving us some initial clues on how to improve it.

References

- [1] B. Baudry, C. Nebut, and Y. L. Traon. Model-driven engineering for requirements analysis. In *EDOC'07: Proceedings of the 11th IEEE Inter. Enterprise Distributed Object Computing Conf.*, page 459. IEEE Computer Society, 2007.
- [2] J. Bézivin. Model driven engineering: An emerging technical space. In *Generative and Transformational Techniques in Software Engineering*, pages 36–64. Springer, 2006.
- [3] C. Calero, F. Ruiz, and M. Piattini. *Ontologies for Software Engineering and Software Technology*. Springer, 2006.
- [4] V. Gervasi and D. Zowghi. Reasoning about inconsistencies in natural language requirements. *ACM Trans. Softw. Eng. Methodol.*, 14(3):277–330, 2005.
- [5] E. C. R. Hehner. Retrospective and prospective for unifying theories of programming. In *UTP*, pages 1–17, 2006.
- [6] M. Jackson. *Software Specifications and Requirements: a lexicon of practice, principles and prejudices*. Addison-Wesley, 1995.
- [7] J.-M. Jézéquel. Model driven design and aspect weaving. *Software and Systems Modeling*, 7(2):209–218, 2008.
- [8] F. Jouault, F. Allilaire, J. Bézivin, and I. Kurtev. ATL: A model transformation tool. *Science of Computer Programming*, 72(1-2):31–39, 2008. Special Issue on Second issue of experimental software and toolkits (EST).
- [9] F. Jouault and J. Bézivin. KM3: A DSL for metamodel specification. In *Formal Methods for Open Object-Based Distributed Systems*, pages 171–185. Springer, 2006.
- [10] I. Jureta, J. Mylopoulos, and S. Faulkner. Revisiting the core ontology and problem in requirements engineering. In *16th IEEE Int. Req. Eng. Conf.*, pages 71–80, 2008.
- [11] I. J. Jureta, S. Faulkner, and P.-Y. Schobbens. Clear justification of modeling decisions for goal-oriented requirements engineering. *Requir. Eng.*, 13(2):87–115, 2008.
- [12] Z. Liu, J. He, J. Liu, and X. Li. Unifying views of UML. *Electronic Notes in Theoretical Computer Science*, 101:95–127, 2004. Proceedings of the Workshop on the Compositional Verification of UML Models (CVUML).
- [13] J. R. Searle. Speech act theory and pragmatics. In *The background of meaning*, pages 221–232. Dordrecht, Netherlands: D. Reidel Publishing Co., 1980.
- [14] E. Yu, P. D. Bois, E. Dubois, and J. Mylopoulos. From organization models to system requirements: a cooperating agents approach. In *In Proceedings of the 3rd International Conference on Cooperative Information Systems*, pages 194–204, 1995.
- [15] P. Zave and M. Jackson. Four dark corners of requirements engineering. *ACM Trans. Softw. Eng. Methodol.*, 6(1):1–30, 1997.