A Model-Driven Approach to Non-Functional Analysis of Software Architectures*

James Skene and Wolfgang Emmerich
Department of Computer Science, University College London

E-mail: {j.skene|w.emmerich}@cs.ucl.ac.uk

Abstract

We present an approach to managing formal models using Model Driven Architecture (MDA) technologies that delivers analysis techniques through integration with the design tools and repositories that practitioners use. Expert modelling knowledge is captured in domain-specific languages and meta-model constraints. These are represented using UML and colocated with designs and analysis models, providing a flexible and visible approach to managing semantic associations. The approach relies on standards to permit deployment in multiple tools. We demonstrate our approach with an example in which queuing-network models are associated with UML design models to predict average case performance.

1 Introduction

Formal analysis is often the only way to determine whether an architectural design will meet its non-functional requirements, such as performance and reliability. Formal analysis is rarely performed partly because of difficulties inherent in its application, such as the need to employ unusual high-level languages, to combat state space explosion and a lack of integrated tool support. This has motivated research into the automated derivation of analysis models from design models, most commonly based on the Unified Modelling Language (UML), a widely adopted design notation capable of capturing details pertinent to a broad range of analyses. Extensions to UML have also been defined to improve its amenity to analysis, most notably [6].

This work suffers in several respects. Firstly, UML and its extensions do not have a formally defined semantic, meaning that there can be no strong proof of the validity of a particular derivation, and this is unlikely to change as it has thus far proved impossible to build a consensus for strong formality for UML. Secondly, the derivations proposed in the literature are defined using a number of ad-hoc techniques ranging from graph-grammars to natural language descriptions. The lack of a standard representation hinders their deployment, and when deployed the technique becomes coupled to a particular tool. Thirdly, a completely encapsulated derivation is unlikely to produce successful analyses every time because of the difficulty in deriving a feasible and valid model for all designs. This implies that tools are likely to be brittle if they cannot be adapted.

In [10] we sketched an approach to analysing non-functional properties of distributed software architectures based on Model Driven Architecture (MDA) technologies. MDA is a development approach based on UML models, in which business knowledge (Platform Independent Models - PIMs) is maintained seperately from technical artefacts, such as design models (Platform Specific Models - PSMs) and source code. The successful application of the MDA approach depends on technologies and tools supporting flexible modelling of diverse semantic domains (PIMs and PSMs), and relationships and transformations between them (deployment of PIMs to PSMs). We use the UML profile mechanism to define classes of analysis models, design models and the mapping between the two. Profiles are denoted using UML and may be injected into any conforming tool, reducing tool tie-in. The derivation is visible and modifiable within a tool, adding flexability, and its colocation with design and analysis models reduces ambiguity due to the relatively informal semantics of UML.

The main contribution of this paper is a substantiation and evaluation of the ideas in [10], using an example in which queuing networks are associated with architectural designs.

The paper proceeds as follows: Section 2 describes our approach; Section 3 gives an example of the approach applied to associate queuing networks with architectural designs; Section 4 evaluates the approach; Section 5 describes related work and Section 6 concludes.

*This research is partly funded through the EU project TAPAS (IST-2001-34069)

2 Approach

Our approach to delivering expert analysis techniques is to specify the derivation of a formal analysis model from a UML design using logical constraints. This is similar to the derivation of a PSM from a PIM in the MDA development approach.

The meaning and structure of UML is defined by a meta-model (the 'abstract syntax') and a set of constraints over the meta-model. The meta-model is object-oriented, containing classes that informally define the meaning of model elements. This informal semantic is reinforced by constraints expressed using the Object Constraint Language (OCL). These prevent the production of models that would represent an illogical situation in the real-world.

A profile is a collection of extensions to the meta-model. It contains 'stereotypes', which are labels for model elements that indicate membership of a 'virtual meta-class'. Virtual meta-classes extend meta-classes to provide a refined meaning. This meaning is reinforced using constraints attached to the stereotype. Model elements may also be adorned with 'tagged values' to specify new properties.

In our approach we define a profile which extends UML to model specifications in some formal language. This language can be used for analysis, so these specification models can be operated upon to generate results. The MDA technologies include standard interfaces for operating on model data [4, 5], so there is the potential to closely integrate model solvers with design tools.

We use a profile for the design domain to direct the derivation of analysis models. Constraints ensure that design models are reasonable and contain sufficient detail to permit a derivation.

The derivation itself is specified in a third profile, allowing reuse of design and analysis profiles. The mapping profile provides a stereotype on associations. The refined type of association is constrained to be between two sub-models, one representing the design and the other the analysis model. Additional constraints on the contents of the sub-models, ensuring that the analysis model correctly represents the design.

Profiles are repesented using UML and may be used wherever necessary by including them in a model. Therefore, profiles specified according to our scheme can be imported into standards-compliant tools to enable formal modelling and the consistent association of formal models with designs.

To provide a completely automated derivation of formal models it is necessary to implement an additional model transformation algorithm, perhaps using a tool-specific scripting language. The mapping constraints provide the contract for such an algorithm, and can be used to test its operation. Figure 1 shows the overall approach.

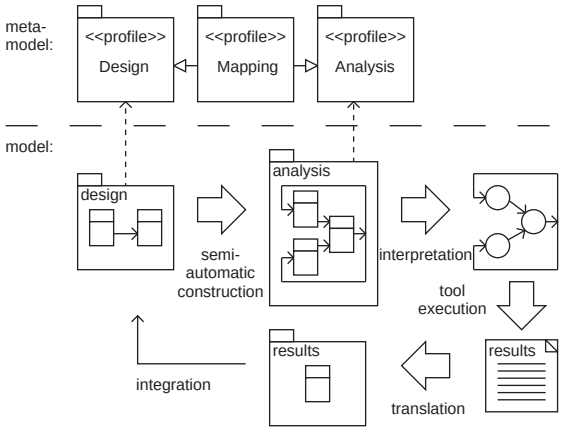

Figure 1. Approach

Results can also be reintegrated into design models by applying a mapping from a results domain to tagged-values or notes in the design domain. We do not consider this in this paper.

The approach is applicable to analyses relying on graphical formalisms, as UML can easily be extended to resemble these. For example, performance evaluation and functional analysis using Petri nets or Markov chains. [8] proposes to use the technique for reliability modelling, and Baysian networks or fault trees would be an appropriate formalism. Our example in the next section uses queuing networks to forecast performance and resource utilisation.

3 Example: Analysing real-time UML using queuing networks

In this section we provide an example of our approach applied to define a derivation from a class of architectural models to queuing networks. To demonstrate the way that this derivation would be used in practice, we present a running example based on a hypothetical website content-management system. The profiles and example packages are shown in Figure 2.

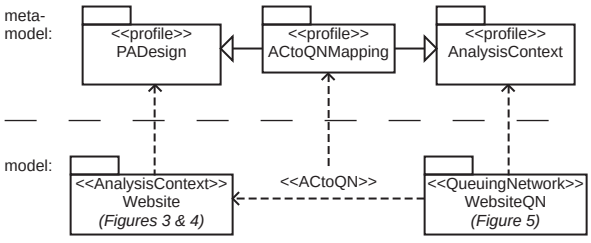

Figure 2. Example profiles

The design model profile closely resembles the perfor-

mance subprofile of the standard Profile for Schedulability, Performance and Time Specification (the 'real-time profile') [6], which permits the modelling of systems containing contended resources. Reuse of such standard profiles enhances the applicability of our approach.

To permit analysis the designer must model the environment of the system in terms of populations engaging in use-cases, identified by a stereotype on an Actor. In our example there is a large `<<OpenPopulation>>` of site users, characterised by an exponential arrival rate, and also a a small `<<ClosedPopulation>>` of editors who interact after a think-time.

The behaviour of the system is modelled using a sequence diagram for each use-case, including `<<step>>` actions with resource demands. The 'user' use-case is shown in Figure 3.

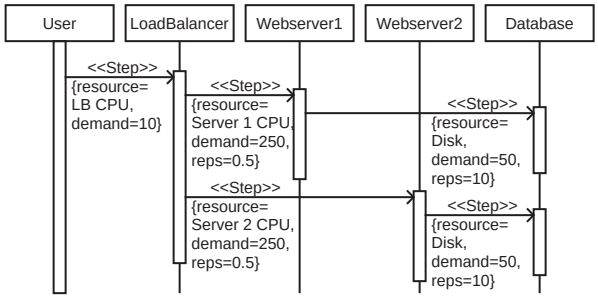

Figure 3. Sequence diagram, describing demands associated with a use-case

The structure and `<<resource>>`s of the system are modelled using a deployment diagram, as shown in figure 4

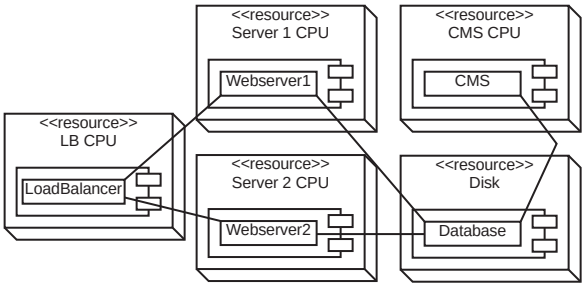

Figure 4. Deployment diagram, describing system resources

Profile constraints are presented here in natural language, but all have a counterpart defined formally using OCL. An example is provided later in the section. The full OCL constraints are available on the web [9].The design profile constraints are:

1. An analysis context must contain at least one population.
2. A population (stereotype on an actor) must be associated with a use-case.
3. Every use-case must be realised by exactly one associated interaction (the dynamic part of a collaboration).
4. Each interaction should include at least one message with a resource demand.
5. Resources must be uniquely named.
6. Every message with a resource demand must be sent to a role deployed in a context where the resource is available.

Analysis models are queuing networks. Figure 5 shows the queuing network derived from our example design, including the demands due to editors.

Analysis model constraints:

1. The network must contain at least one workload class
2. Classifiers can be either queues or workloads, not both
3. All associations stereotyped as demands must start from a classifier stereotyped with a workload and end in one stereotyped as a queue.
4. All queues must be demanded by at least one workload class.
5. All workloads must demand at least one queue.

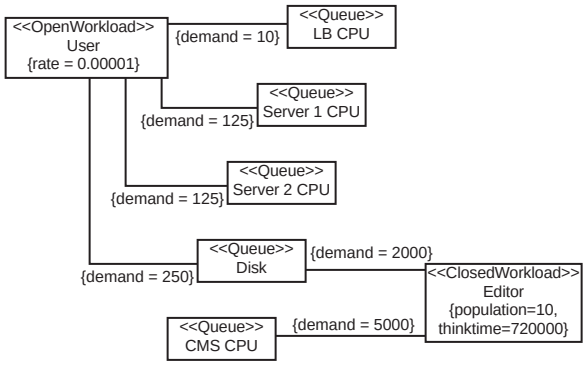

Figure 5. Queuing network model

The mapping is defined in the context of a stereotype `ACtoQN` defined on an association. Mapping constraints are:

1. The mapping must be between an analysis context model and a queuing network model.
2. Every population must correspond to exactly one workload class in name, type and tagged values.
3. Every workload must have exactly one corresponding population.
4. All resources with resource demands in a use-case must be represented by queues.

5. All resource demands present in an interaction must be represented by demands on associations between the corresponding workload and queue. The demands must equal the sum of the products of action demands and action repetitions within the workload.

Constraint 3 above is expressed in OCL as follows (this constraint relies on the previously defined functions 'populations' and 'workloads' that return the sets of these elements in the associated models):

```
package Foundation::Core
 context Abstraction inv:
  self.stereotype→exists(name = "ACtoQN")
  implies
  self.populations→forAll(w : ModelElement |
   self.workloads→one(p : ModelElement |
    p.name = w.name and
    (w.stereotype→exists(name = "OpenWorkload") implies
    p.stereotype→exists(name = "OpenPopulation")) and
    (w.stereotype→exists(name = "ClosedWorkload") implies
    (p.stereotype→exists(name = "ClosedPopulation"))))
```

4 Evaluation

A lack of tool support for OCL currently hinders widespread use of our approach. We have shown it to be workable in our example, which we developed using a commercial UML editor, the Rocase OCL Evaluator [3], a research prototyped developed as part of the IST Neptune project, and an external queuing network evaluator. The need to develop models and then export them to an external tools compromises the goal of tight integration of analysis and design tools. We have reason to be hopeful that the situation will improve. The reliance of the MDA approach on model transformations, the use of OCL to describe domain rules, and an increased emphasis on the meta-modelling of disparate domains will lead to more standards-based tools compatible with our approach.

5 Related work

Our work identifies a standard means to describe the derivation of formal models from UML, so is related to and we believe compatible with the large amount of work that covers particular derivations [2]. Our example uses a formalisation of part of the real-time profile and derives a simple queuing network. It resembles the real-time profile to layered queuing network derivation presented using graph grammars in [7].

The Precise UML group [1] have proposed the use of constraints between language and domain meta-models to describe the semantics of UML. We apply this approach to mappings, but using profiles instead, to enable tool support.

6 Conclusions

We have presented an approach to associating analysis models with designs using MDA technologies, including UML and OCL, and based on model-transformations, a concept intrinsic to the MDA. The approach addresses several problems with previous work that derives analysis models from UML diagrams: It relies on standards so is not coupled to a particular modelling tool; mappings are visible and modifiable, increasing the flexibility of tools; mappings and analysis models are visible in the same context as designs, reducing ambiguity due to the lack of a strict semantic for the design notation.

Despite an existing lack of tool support, we have confidence that the emerging popularity of the MDA will lead to an increasing adoption of standards-compliant integrated development environments capable of supporting the kind of re-deployable automated analysis modelling that we describe.

References

[1] The precise UML group. `http://www.puml.org/`.

[2] *WOSP 2002, Third International Workshop on Software and Performance*. ACM Press, 2002.

[3] Computer Science Research Laboratory, "BABES-BOLYAI" University, Romania. *OCL-Evaluator*. `http://lci.cs.ubbcluj.ro/ ocle/`.

[4] OMG document formal/02-01-01. *XML Metadata Interchange (XMI), version 1.2*, 2002.

[5] OMG document formal/02-04-03. *Meta Object Facility (MOF), version 1.4*, April 2002.

[6] OMG document ptc/03-03-02. *UML Profile for Schedulability, Performance, and Time Specification*, 2003.

[7] D. Petriu and H. Shen. Applying the UML performance profile: Graph grammar-based derivation of LQN models from UML specifications. In *Proceedings of Performance TOOLS 2002*, 2002.

[8] G. N. Rodrigues, G. Roberts, W. Emmerich, and J. Skene. Reliability support for the model driven architecture. In *Workshop on Software Architectures for Dependable Systems (ICSE-WADS)*. ACM Press, 2003.

[9] J. Skene and W. Emmerich. Full example profiles. `http://www.cs.ucl.ac.uk/ staff/ j.skene/ FSE-2003-profiles`.

[10] J. Skene and W. Emmerich. Model driven performance analysis of enterprise information systems. In *Workshop on Test and Analysis of Component Based Systems (ETAPS-TACoS)*, Electronic Notes in Theoretical Computer Science (ENTCS). Elsevier Science B. V., 2003.