Reliability Support for the Model Driven Architecture*

Genaína Nunes Rodrigues, Graham Roberts, Wolfgang Emmerich and James Skene
Dept. of Computer Science
University College London
Gower Street, London WC1E 6BT, UK
{G.Rodrigues|G.Roberts|W.Emmerich|J.Skene}@cs.ucl.ac.uk

Abstract

Reliability is an important concern for software dependability. Quantifying dependability in terms of reliability can be carried out by measuring the continuous delivery of a correct service or, equivalently, of the mean time to failure. The novel contribution of this paper is to provide a means to support reliability design following the principles of the Model Driven Architecture(MDA). By doing this, we hope to contribute to the task of consistently addressing dependability concerns from the early to late stages of software engineering. Additionally, we believe MDA can be a suitable framework to realize the assessment of those concerns and therefore, semantically integrate analysis and design models into one environment.

1. Introduction

Component-based development architectures (CBDA) are increasingly being adopted by software engineers. These architectures support distributed execution across machines running on different platforms (e.g. Unix, Windows). Examples of component models include Sun's Enterprise Java Beans (EJB), OMG's CORBA Component Model (CCM) and Microsoft's .NET. Additionally, CBDAs rely on the construction and deployment of software systems that have been assembled from components [3].

One of the advantages of applying a component-based approach is reusability. It is easier to integrate classes into coarse-grained units that provide one or more clearly defined interfaces. However, the lack of interoperability among diverse CBDAs may be one of the major problems that hinders the adoption of distributed component technologies. Once a platform has been chosen and the system has been developed, porting to a different platform becomes troublesome.

*This research is partially supported by CAPES and the European Union under grant IST-2001-34069

To fill the gap, the OMG has focused on paving the way to provide CBDAs interoperability standards through the Model Driven Architecture (MDA). Essentially, "the MDA defines an approach to IT system specification that separates the specification of system functionality from the specification of the implementation of that functionality on a specific technology platform" [9]. To accomplish this approach, MDA structures the system into key models: the Platform Independent Models (PIMs) and the Platform Specific Models (PSMs). While a PIM provides a formal specification of the structure and function of the system that abstracts away technical details, a PSM expresses that specification in terms of the model of the target platform. Basically, PIMs are mapped to PSMs when the desired level of refinement of PIMs is achieved.

The Unified Modeling Language (UML) is the core element to represent those models. According to the OMG, UML supports the formalization of an abstract, though precise, models of the state of an object, with functions and parameters provided through a predefined interface [9]. Furthermore, UML models facilitate the assessment of a design in the early stages of software development, when it is easier and cheaper to make changes.

The defined and standard structure of MDA would seem suitable to address software dependability, in that the MDA designates the system function as required by the stakeholders. Issues such as reliability, safety, security and availability comprise software dependability [8, 12]. However, there is no standard representation for dependability in MDA models. During the software execution this can lead to situations not foreseen in the platform models.

Reliability assurance is an important concern in software dependability. Quantifying dependability in terms of reliability can be carried out by measuring the continuous delivery of correct services or equivalently, of the mean time to failure [4]. A system can be considered reliable if it performs at a constant level, as the stresses on that system change. For example, if a0 request takes 10 ms to complete with one user, then it takes the same time to process

the same request with 1000 concurrent users.

To overcome the lack of dependability concern in the current MDA specification, we propose to explicitly tackle this problem in the levels of abstraction suggested by OMG's MDA. We believe it is feasible to accomplish this task using the standard meta-modeling approach of MDA and specifications, such as the work in [10], as sources to achieve this goal. Our focus on dependability at the moment is reliability. To guarantee and assess reliability properties of software systems using the MDA approach, we plan to achieve reliability in such a way that it would be specified in the early stages of software architecture design. In this way, we aim to provide reliability in a platform-independent way. In the context of MDA and current distributed component-based architectures, early reliability assessment is important as the software architecture design is specified in the context of software development.

The novel contribution of this paper is to provide a means to support reliability design following the principles of MDA. By doing this, we hope to contribute to the task of consistently carrying out dependability concerns from the early to the late stages of software engineering. Besides, MDA appears to be a suitable framework to realize the assessment of those concerns and therefore, semantically integrate analysis and design models into one environment.

In this position paper, we elaborate our approach on how the provision of reliability can be suitably realized through a standard model-driven architecture approach. In Section 2, we present the related work targeting reliability support and analysis in the CBDA scenario. In Section 3, we show how we plan to provide reliability modeling in MDA and the steps to be followed to accomplish this goal. In Section 4, we provide a sample scenario of how our approach addresses reliability support in MDA. Finally, Section 5 summarizes our contribution and discusses future work towards achieving standard reliability support from designing models to programatic interfaces.

2. Related Work

The work described in [1, 2, 7] looks at part of the problems we identify in our work, in terms of addressing dependability concerns in the early stages of software development. We can basically find in these works analysis techniques to validate design tools based on UML.

However, they differ from our approach in some important aspects. Primarily, they do not conform to the principles stated by MDA. MDA uses the straightforward approach through the concepts of mapping models among different platforms. Therefore, we believe that MDA offers a suitable environment to consistently integrate the analysis and design of dependability issues, and from design to implementation. [1] provides a useful transformation technique to automate dependability analysis of systems designed using UML. Nevertheless, to properly contemplate dependability in all stages of the software engineering process, we believe that one of the main concern is to provide a unified semantic between the analysis and the design models.

Another approach to address software dependability is to provide mechanisms to improve reliability of software after it has been implemented. Works such as [5] use testing techniques to identify faults in the software that are likely to cause failures. Although they carry out an important research agenda, we believe that is cheaper to design and evaluate dependability concerns in the early stages of software engineering processes . Besides, levels of abstraction like the one expressed by the PIM and PSM models of MDA seems to be necessary in a scenario where each of the existing CBDA holds distinct mechanisms to support dependability.

A meta-model is a model of a language expressed using a modeling technique. This feature in UML allow us to express the design and analysis domains naturally, using the concepts inherent to these domains. Moreover, this facility permits to map the behavior of distributed component architectures into a domain knowledge keeping the semantics of the modeling requirements of UML. Following this principle, our approach to meta-modeling using the UML lightweight extension mechanisms, i.e. profiles, is consistent with the official MDA white paper [9], which defines basic mechanisms to consistently structure the models and formally express the semantics of the model in a standardized way. Moreover, the profiles define standard UML extensions to describe platform-based artifacts in a design and implemented model. For example, the UML Profile for EJB [6] supports the capture of semantics expressible with EJB through the EJB Design Model and the EJB Implementation Model. Although currently cannot be found UML profiles to thoroughly address dependability, MDA seems to be a feasible environment to consistently assess and express dependability by means of profiles properly constructed.

Another benefit that arises from this consistent integration is the facility to realize reverse engineering. However, it is out of scope of our current work to cope with this topic.

3. A Profile for Reliability

To provide a reliability profile for MDA, we follow a bottom-up approach as MDA allows this flexibility (see Figure 2). Having J2EE as a first target to realize a reliability profile, we plan to extend the UML Profile for EJB [6] to express reliability primitives available in J2EE in a standard way. By standard way, we mean to specify a subset of UML meta-model that describes the semantics of mechanisms in J2EE to achieve reliability. This subset contains

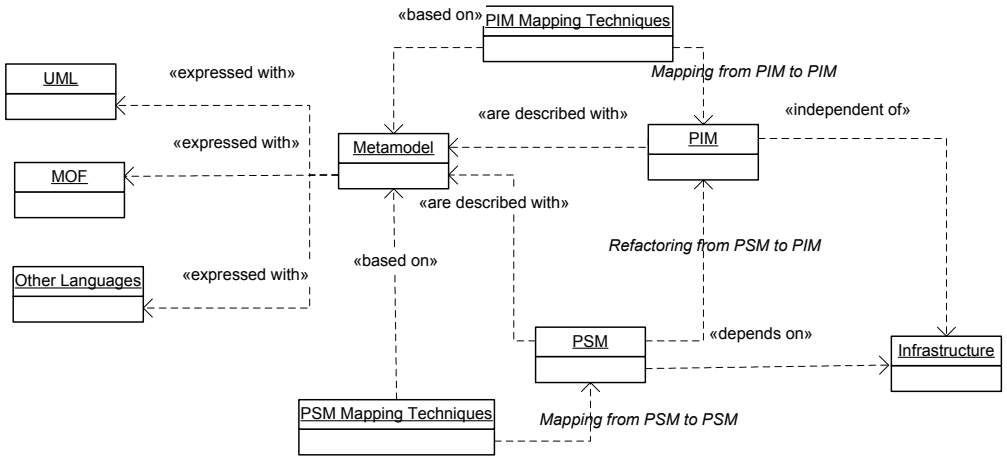

Figure 1. MDA Metamodel Description [9]

stereotypes, tagged values and OCL constraints.

To assure reliability, the J2EE platform has several mechanisms [15]:

- Fail-Over through clustering of containers[1]
- Asynchronous communication with persistent JMS and Message Beans;
- Persistent Entity Beans;
- Transactions through the two-phase commit protocol;
- Security.

For the sake of achieving abstraction, these mechanisms should be supported by the UML/EJB profile. In order to realize this task, UML meta-models must be designed to reflect each of those mechanisms, relying on the current UML Specification [11]. By doing this, it will be possible to express semantics and notations that adequately address those reliability mechanisms in a standard way.

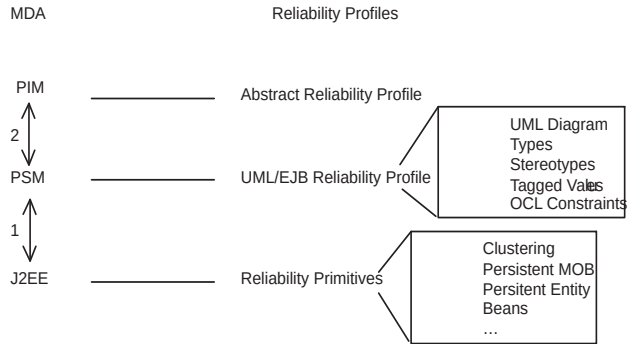

Figure 2. Reliability Profiles in MDA

[1]Interested readers may refer to [13], chapter 14

UML provides a wide set of modeling concepts and notations to meet current software modeling techiniques. The various aspects of software systems can be represented through UML, what makes it suitable for architecture modeling. Besides, it is widely used for software analysis and development. To model complex domains in UML, new semantics can be added through extension mechanisms packages that specify how specific model elements are customized and extended with those semantics. We adopt profile as our extension mechanism, which comprises model elements customized for a specific domain or purpose (e.g. designing reliability) by extending the meta-model using stereotypes, tagged definitions and constraints [11]. In order to design and formalize J2EE reliability mechanisms, we will first map them into a profile, the Refactoring mechanism in Figure 1.

In MDA, a mapping is a set of rules and techniques used to modify one model in order to get another model. The following step is to design reliability in the highest abstract level stated by the architecture of MDA, which is the PIM. Achieving a platform-independent reliability model, the task of designing dependability concerns can be carried out in the early stages of software engineering where the software architecture is designed. The steps to accomplish this goal are as follows:

1. Determine the reliability properties of interest.
2. Create a set of stereotypes, tagged values and constraints to build the UML/EJB Profile for Reliability.
3. Provide the design domain mappings between each reliability profile instances and the UML/EJB Profile.
4. Define a mapping between the design domain achieved in the previous step and a platform-independent design domain that correctly represents the semantics of each

reliability mechanism. A preliminary PIM version is expected at the end of this step.

5. Identify those qualities that are of interest but require formal analysis to determine. Choose an appropriate analysis technique for reliability analysis (e.g Bayesian Networks) and define a profile to represent the entities within the analysis domain.

6. Define a mapping between the design domain and the analysis domain that correctly represents the semantics of each.

7. Choose a commercial existing component model other than EJB for J2EE platform to make the PIM to PSM mapping, providing the PIM reliability profile.

8. Automate the mapping.

9. Provide scenarios to monitor and assess the models in a real-life case study.

It should be noticed that in Step 4, the mapping from PSM to PIM will be carried out in order to reach the highest abstract level of reliability mechanism. The MDA principles allows this bottom-up approach and we decided to follow this step in order to raise the kind of resources needed in a reliability UML profile. This mapping can be formalized using the Object Constraint Language (OCL), which is the formal language used to specify well-formedness rules of the meta-classes comprising the UML meta-model [11]. However, this formalization would require an assessment of the designed properties. That is the purpose of Step 6. Achieving this level of abstraction is not the whole plan however. In Step 8, mechanisms to automate the target reliability primitive may be desired by those who want to apply the reliability profile attained in our work. Finally in Step 9, we apply and evaluate our approach using a real-life case study.

4. A Scenario of Reliability Support in MDA

This section shows how we plan to achieve the previously stated goals through an example. We highlight how one of the reliability mechanisms can be mapped to a UML profile in a standard way and how it would reflect on the deployment of the components. To make it concrete, we plan to first build a reliability profile for a target platform, which is the J2EE platform. By doing this, it will be easier to identify the kind of resources for reliability that should be comprised in a platform-specific model and therefore those to be comprised in a platform-independent model through MDA mappings.

In this scenario, we exploit one of the mechanisms of EJB to provide reliability, the fail-over through clustering. The fail-over mechanism redirects a single request to another node in the cluster because the original node could not process the request. This implies another concept, which is clustering. The overall goal of employing a cluster is to increase the availability or reliability of the system by joining services into groups that provide services to their clients in a loosely coupled way. The number of nodes comprising a cluster will vary according to the degree of reliability we want to assure.

The first step towards achieving reliability in MDA principles, is to define the architecture of the application by means of the UML Profile for EJB [6]. To express how reliable the method invocations will be and the deployment relationships between the application components, a reliability profile is needed. Figure 3 shows what the overall scenario would look like. Basically, there are three main profiles: the design (where the reliability mechanism is modeled), the mapping (to map the desired reliability to the designed classes), and the deployment (to provide how the components will be distributed in the network according to the required reliability support).

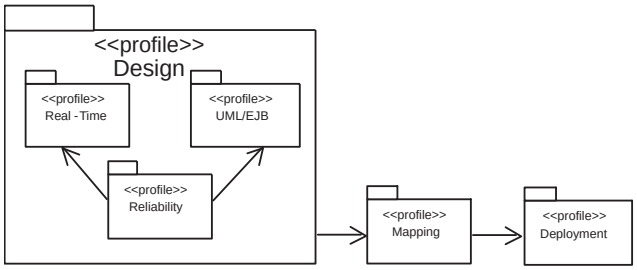

Figure 3. Profiles to model reliability in EJB applications

In the design profile, meta-modeling techniques will be used to map out reliability mechanisms in a profile. This profile is composed of three main specifications:

1. UML/EJB Profile - which expresses the basic semantics of EJB in the UML notation.

2. UML Profile for Schedulability, Performance and Real-Time Specification (briefly, Real-Time Profile) [10] - for the reason that it specifies how applications requiring a quantitative measure of time can be modelled in a MDA standard way.

3. UML Specification [11] - to realize what is lacking in the above specifications to carry out the reliability profile following standardized UML notations, definitions and semantics.

In the mapping domain, where the mapping profile will be realized, constraints that rule the desired reliability mechanism are mapped to a designed application. For

example, all the stateful session beans to be replicated throughout the clusters should be idempotent (i.e. they can be called repeatedly without worrying about altering the system so that it becomes unusable or provides errant results). Finally, the deployment profile will provide the configuration of how the components will communicate and be distributed throughout the network.

In order to map the clustering mechanism proposed in this scenario, we should know what is the desired reliability assurance of the system. By this means, it is possible to know how many replicated components there should be in each cluster to guarantee the desired reliability level.

The functional formula for this assurance is:

$$1 - (1 - c)^n > a \quad (1)$$

where c is the reliability of each component, a is the required reliability of the system and n is the number of components that should be comprised in each cluster. Suppose a is 95% and c is 75%. Then, according to Formula 1 the value of n is 3. Therefore, each cluster of the deployment diagram should have at least 3 copies of the component to be replicated.

To reflect this scenario, the classes of the design profile to be replicated should be mapped to the deployment profile through the mapping profile. A fragment of the mapping profile to assure the reliability property above is described in OCL as follows:

```
context mapping inv:
self.supplier.ownedElements->select(
  m : ModelElement | m.oclIsType(Class) and
  m.stereotype->exists(name =
  "replicatedComponent"))-> forAll(
     (1 - (1 - m.taggedValue->any(type =
     "componentReliability").dataValue)^
      self.consumer.ownedElements->
         select(n : ModelElement |
           n.oclIsType(Component) and
           n.name = m.name))->
           size())> m.taggedValue->any(type =
           "aggregateReliability").dataValue)
```

where `self.supplier` refers to the classes in the designed profile and `self.consumer` refers to the components in the deployment profile.

There is, however, one important step that is not described here but must be accomplished, which is the support in MDA for formal analysis. In this regard, it is required a formal analysis profile to separately express dependability in an analysis model. This accomplishment might follow the approach in [14], which is under development here at UCL. That work aims at providing a MDA performance analysis to enterprise applications and has shown that worthwhile benefits arise, such as:

- Flexible application of analysis techniques to design domains by defining new mappings;
- Use of model checking tools to check the semantic validity of the analytic model against the design model.

5. Conclusions And Future Work

In this paper we have presented the idea on how we plan to tackle the problem of reliability assurance in MDA. The motivation to achieve this purpose is the identified importance and benefits arising from addressing dependability concern in the stage of software engineering where the architecture is designed.

There are many steps, however, to accomplish that task. First of all, a reliability profile should be carried out. In order to achieve a consistent and integrated environment, all the steps should be expressed within the available mechanisms of MDA. Exploiting the standard UML and the profiles already created constitutes the basis of our work. However, there are complementary mechanisms that the current MDA does not provide. For example, ways to assess the designed dependability issues. Therefore, a profile to carry out this assessment should be created by means of metamodeling techniques, following the same approach of [14].

Immediate future challenges include determining precisely a profile to translate reliability in terms of a valid MDA profile. To achieve this goal within a more concrete approach, we plan to map into that profile the reliability mechanisms available in the J2EE platform. This bottom-up approach is expected to aid in identifying the required resources that should be mapped in a reliability-aware PSM. Following all the steps presented in Section 3, we finally wish to enhance the level of automation for mappings and evaluate the practicality of our approach using real-life case studies with realistic complexity.

Acknowledgments

We would like to thank Licia Capra, Rami Bahsoon and Philip Cook for their assistance with this document.

References

[1] A. Bondavalli, I. Majzik, and I. Mura. Automatic Dependability Analysis for Supporting Design Decisions in UML. In R. Paul and C. Meadows, editors, *Proc. of the 4^{th} IEEE International Symposium on High Assurance Systems Engineering*. IEEE, 1999.

[2] V. Cortellessa, H. Singh, and B. Cukic. Early reliability assessment of uml based software models. In *Proceedings of the third international workshop on Software and performance*, pages 302–309. ACM Press, 2002.

[3] W. Emmerich. Distributed Component Technologies and Their Software Engineering Implications. In *Proc. of the* 24^{th} *Int. Conference on Software Engineering, Orlando, Florida*, pages 537–546. ACM Press, May 2002.

[4] J. C. L. et.al. *Dependability: Basic Concepts and Terminology*. Springer–Verlag, 1992.

[5] P. Frankl, R. Hamlet, B. Littlewood, and L. Strigini. Choosing a Testing Method to Deliver Reliability. In *International Conference on Software Engineering*, pages 68–78, 1997.

[6] J. Greenfield. UML Profile for EJB. Technical report, http://www.jcp.org/jsr/detail/26.jsp, May 2001.

[7] G. Huszerl and I. Majzik. Modeling and analysis of redundancy management in distributed object–oriented systems by using UML statecharts. In *Proc. of the* 27^{th} *EuroMicro Conference, Workshop on Software Process and Product Improvement, Poland*, pages 200–207, 2001.

[8] B. Littlewood and L. Strigini. Software Reliability and Dependability: A Roadmap. In A. Finkelstein, editor, *The Future of Software Engineering*, pages 177–188. ACM Press, Apr. 2000.

[9] Object Management Group. Model Driven Architecture. Technical report, http://cgi.omg.org/docs/ormsc/01-07-01.pdf, July 2001.

[10] Object Management Group. UML Profile for Schedulability, Performance and Real-Time Specification. Technical report, http://www.omg.org/cgi-bin/doc?ptc/02-03-02.pdf, March 2002.

[11] Object Management Group. Unified Modeling Language (UML), version 1.4. Technical report, http://www.omg.org/cgi-bin/doc?formal/01-09-67.pdf, January 2002.

[12] B. Randell. Software Dependability: A Personal View (Invited Paper). In *Proc. 25th Int. Symp. Fault-Tolerant Computing (FTCS-25, Pasadena)*, pages 35–41. IEEE Computer Society Press, 1995.

[13] E. Roman. *Mastering Enterprise Java Beans*. John Wiley & Sons, Inc, 2002.

[14] J. Skene and W. Emmerich. Model Driven Performance Analysis of Enterprise Information Systems. *Electronical Notes in Theoretical Computer Science*, March 2003. To appear.

[15] Sun MicroSystems. Enterprise JavaBeans Specification, version 2.1. Technical report, http://java.sun.com/j2ee/docs.html, August 2002.