A Model Driven Approach for Software Systems Reliability

Genaína Nunes Rodrigues
David Rosenblum and Wolfgang Emmerich (Supervisors)
Department of Computer Science
University College London
Gower Street, London UK WC1E 6BT
g.rodrigues@cs.ucl.ac.uk

1 Research Problem

Component-based development architectures (CBDA) are increasingly being adopted by software engineers. The Object Management Group (OMG) has focused on paving the way to provide CBDAs with, among other benefits, interoperability and portability standards through the Model Driven Architecture (MDA). The main idea of MDA is to use UML to specify both the static interfaces and the dynamic behavior of the components in platform-independent models(PIM). Additionally, it defines rules so that PIMs can be mapped into a number of platform-specific models(PSM).

Although MDA promises to overcome important unsolved problems in software engineering, it has not specified ways to represent software dependability yet. Particularly, there are no current means to design and assess system dependability, specially reliability, using the model-driven principles proposed by MDA. Issues such as reliability, safety, security and availability comprise software dependability [1, 10].

The lack of a representation for dependability in MDA models can consequently lead to undesirable situations during the software executions. We argue that the standard structure of MDA is suitable to address software dependability, for the MDA designates the system function as required by the stakeholders. Among other dependability features, our work focuses particularly on addressing reliability concerns into MDA. The syntax and semantics of MDA models are represented through profiles, a set of stereotypes, tagged values and constraints that extend the core UML using metamodeling techniques.

OMG has recently revised requests for proposals on Quality of Service and Fault-Tolerant profiles [12]. Nevertheless, those requests still require means to assess dependability by measurement and analysis methods to guarantee that the desired properties of the system are correctly represented in the models. Additionally, OMG needs to address how the PIMs will be consistent with the profile, once it is specified. In other words, how to apply transformation rules from profiles to PIMs and from PIMs to PSMs.

Existing research on dependability focuses on the early design phases of software development [3, 9]. They look at some of the problems we identify in our work related to addressing dependability concerns in these phases. They use analysis techniques based on architectural representation techniques, e.g. Architecture Description Languages (ADL), or modeling languages, e.g. UML, to validate qualities related to the design of the software system.

However, their work is tightly coupled with tools or platforms where these solutions are targeted for, which does not conform to the concepts of portability and interoperability of MDA. [2] 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 concerns is to provide a unified semantic representation 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 through testing techniques. Works such as [6] use those techniques to identify faults in the software that are likely to cause failures. Our purpose is to concentrate on the reliability assurance of the system from design to deployment level through transformation techniques. Concentrating on these levels, we believe that the desired reliability of software systems will be reported in the testing phase according to the required reliability property defined in the architecture level.

2 Hypothesis

Our hypothesis is that MDA is a feasible environment to systematically assess and express dependability by means

of profiles properly constructed. We propose to provide reliable software systems through a model driven approach. Once reliability mechanisms provided by current CBDAs are designed in a platform-independent way, platform-based design and implementation models must be therefore extended.

The UML meta-model defines the abstract syntax of UML, from which many concrete syntaxes can be derived [5]. This feature in UML allows us to express the design and analysis domains seamlessly, using the concepts inherent to these domains. Thus, this facility permits the mapping of 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 [11], which defines basic mechanisms to consistently structure the models and formally express the semantics of the model in a standard way. Moreover, the profiles define standard UML extensions to describe platform-based artifacts in a design and implemented model.

3 Our Proposed Solution

The main contribution of this research is to provide platform-independent means to support reliability design following the principles of a model driven approach. The contribution aims to systematically address dependability concerns from the early to the late stages of software development. MDA appears to be a suitable framework to assess these concerns and, therefore, semantically integrate analysis and design models into one environment.

We propose to overcome the lack of a model driven dependability concern by identifying the levels of abstraction following OMG's MDA principles. To achieve this goal, this work relies on reference models specifications such as [13] as well as extensions of the UML metamodels. The current dependability property on focus 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 is specified in the early stages of software architecture design.

The first step towards achieving reliability in MDA principles is to define a profile. In this regard, we have to represent software reliability in different levels of abstraction according to the MDA approach as well as the transitions from PIM to PSM and from PSM to code. Current CBDAs, such as Enterprise Java Beans, address a considerable range of features to support system reliability. There are some mechanisms these CBDAs provide in order to provide reliable services [4]:

- Replication transparency through clustering;
- Failure transparency through atomic transactions;
- Asynchronous communications through message oriented components;
- Persistency support through stateful and persistent component objects.

For each of these reliability mechanisms, we plan to build a profile for.

By means of a reliability profile, the architecture of an application can express both method invocations and deployment relationships between the application components. Actually, the reliability profile comprises three other sub-profiles: the design (where the reliability mechanism is modeled), the mapping (to map the desired reliability to the designed classes), and the deployment (to show how the components can be distributed in the network according to the required reliability support).

In the design profile, meta-modeling techniques are used to map out reliability mechanisms in a profile. This profile extends two main specifications:(1) the UML Profile for Schedulability, Performance and Real-Time Specification [13] and (2) the UML Specification [14]. The first step to accomplish this profile is to build a reference model that defines the classes, relationships, attributes and operations to represent the dynamic behaviour of the reliability mechanism.

In the mapping domain, where the elements in the design profile are mapped to the deployment profile, constraints rule how the desired reliability mechanisms are mapped to a designed application. The transformation rules defined by the OCL constraints are the core part of this domain. Finally, the deployment profile provides the configuration of how the components communicate and are distributed throughout the network.

4 Ongoing Work and Expected Contributions

We had previously concentrated on mapping out the area of interest, finding the problem to tackle, analyzing the contributions related to the problem we identified and defining the strategy to overcome the lack in the identified problem. The result of this work is in [8].

We decided to define a profile for each reliability property at a time. We have been currently working out the construction of reliability profile based on replication. In this regard, we have built a profile for transient systems and mapped the core elements of the profile to the EJB architecture as a platform-specific mapping of that reference model. The elements of the design profile were identified by extending the UML metamodeling language and mapping them into the deployment diagram of the components

following transformation rules to be automated in a future step of our work.

Expected Contributions - At the end of this process of mappings and refinements, it is therefore expected the identification of the elements required to design, implement and deploy reliability according to the OMG MDA. By these means, the main contributions we plan to achieve are:

1. Propose a way of specifying reliability in a platform-independent way;

2. Propose a set of rules to automatically map reliability mechanisms from PIM to PSMs in the context of the MDA.

In addition to these direct contributions, we expect to come up with ideas that contribute in the overall process of using a model driven approach in the software development so as to:(1) identify general procedure to build a reference model and (2) guarantee that the reference model and the profile are consistent with each other and with the generated code.

5 Future Work and Concluding Remarks

As soon as our reliability profile based on replication is built, our plans for the future contemplate:

1. Analysis - Identify those qualities that require formal analysis to determine and choose an appropriate technique for reliability analysis. Define a profile to represent the entities within the analysis domain.

2. Mapping - Define and verify the mapping between the design domain and the analysis domain from the previous step that correctly represents the semantics of each domain extending an MDA tool for code generation.

3. Another PSM - Choose another platform and extend the platform-specific profile (e.g. [7]) to represent the reliability mechanisms of interest.

Evaluation - The evaluation aims to test the maturity of the approach, its applicability, and the effectiveness of the reliability models. To provide such assessment, we plan to monitor the performance of the model using a case study. The rationale is to populate failure analysed from real life scenarios into a subject system. The evaluation aims to assess the ability of the models to detect the following qualities of interest: replication transparency, failure transparency, persistency and asynchronous communications.

Concluding Remarks - This work is expected to turn the provision of reliability for software systems into a more practical approach. There are currently mechanisms to provide reliability for software systems. However, techniques such as process algebras are generally considered time consuming, in regard to the software development. The model driven approach seems suitable to fill in this gap and we expect to provide a solution where reliability can be assured along the life cycle of software development.

References

[1] A. Avizienis, J. Laprie, and B. Randell. Fundamental Concepts of Dependability, Research Report N01145, LAAS-CNRS, Apr. 2001.

[2] 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.

[3] 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.

[4] W. Emmerich. *Engineering Distributed Objects*. John Wiley & Sons, Inc, 2000.

[5] D. S. Frankel. *Applying MDA to Enterprise Computing*. OMG Press and Wiley Publishing, Inc, 2003.

[6] 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.

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

[8] G.Rodrigues, G.Roberts, W.Emmerich, and J.Skene. Reliability Support for the Model Driven Architecture. In *Workshop on Software Architecture for Dependable Systems (ICSE/WADS 2003), Portland, OR.*, pages 7–12. ACM Press, May 2003.

[9] 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.

[10] 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.

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

[12] Object Management Group. UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms. Technical report, http://www.omg.org/docs/ad/02-01-07.pdf, December 2002.

[13] 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.

[14] 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.