A Model-Driven Architecture Approach Using Explicit Stakeholder Quality Requirement Models for Building Dependable Information Systems

Stefan Biffl[1] Richard Mordinyi[2] Alexander Schatten[1]
biffl@ifs.tuwien.ac.at *richard@complang.tuwien.ac.at* *schatten@ifs.tuwien.ac.at*
[1]*Inst. of Software Technology and Interactive Systems, Vienna University of Technology*
[2]*Space-Based Computing Group, Inst. of Computer Languages, Vienna University of Technology*

Abstract

Decision makers in safety-critical domains rely on data from dependable information systems. Traditional time- and safety-critical information systems, such as traffic management systems, have been using proprietary point-to-point data links, which are very dependable, but also time-consuming and costly to change due to the need to manually adapt a multitude of deployed systems.

In this paper we introduce a model-driven architecture (MDA) system approach that describes explicitly stakeholder quality requirements on dependable data links between systems for decision support and generates new system versions that implement these requirements. The MDA approach is expected to a) improve the quality (assurance) of system requirements; b) support more explicit feedback on the quality of intermediate models during systems development; and c) provide better auditing capabilities of the systems development process.

Based on an industry case study we describe the MDA concept of the system, the development process, and how software quality can be measured and improved.

1. Introduction

Much work of requirements engineering aims at capturing the value of a software system from eliciting and reconciling the value propositions of success-critical stakeholders [1], [2]. Based on these requirements project management and quality assurance derive internal and external measures for guiding software development. While considerable research has been done on internal views of quality [3], [4], the customer view of quality seems to be more elusive [5]. One challenge is that elements of customer quality need to be transformed and implemented properly in many parts of the system (and the organization that runs the system) in order to achieve these customer quality goals.

In traditional software development, e.g., following a waterfall or RUP approach, the many artifacts in the software development are linked by the software process and responsibilities of the software development roles. However, these artifacts typically change concurrently as requirements change, feedback from software testing comes in, etc. Thus, a major challenge of quality management is to ensure that a consistent picture of stakeholder value propositions gets propagated throughout the software development process [6].

Model-driven architecture (MDA) development [7] aims at generating systems from high-level system models and requirements models, taking away much of the concurrent manual changing of artifacts at the different stages of software development. Such an approach promises better leverage on building quality (i.e., stakeholder value) into the software products and should support the measurement of software quality at different stages of the development life cycle [8], [9]. However, demonstrating effective quality assurance for a) the high-level source models that contain the key requirements and b) realistic solution scenarios for the target architecture becomes more critical [10], [11].

In this paper we report on work-in progress from an industry case study that a) introduces a MDA approach for dependable systems' information sharing middleware, b) discusses the expected benefits and risks for building and assuring stakeholder-related quality compared to a traditional development approach.

The goal of the case study project, *information sharing network* (ISN), is to provide decision makers in safety-critical domains, such as real-time logistics management or traffic control, with the most relevant information in a timely manner. The *ISN* connects business applications that provide and/or consume data with defined quality levels, e.g., for accuracy, validity, and refresh rates. Background information on the needs to introduce new technology can be found in [17].

In the case study context, there are several views on stakeholder quality: 1. Support for the operational *decision maker*, i.e., provide the best currently possible data quality without confusing the operator; 2. Adjust the information supply to *changing business needs*, i.e. include new data sources both flexibly and safely; and 3. Reconcile the *business interests of the various stakeholders* in one organization or several organizations that provide, refine, transport and consume a wide range of data, from logistics plans to vehicle position information, weather forecasts and fleet capacity utilization reports.

Core idea of the ISN project is to describe stakeholder data access quality requirements and system infrastructure capabilities in explicit models (using semantic web technology that allows to discover matching data consumer needs and providers regarding data content and service level requirements); then try to generate a systems configuration plan that satisfies the stakeholder quality requirements (see Figure 3). The semantic web technology supports the system design, but is not used at run-time as rather strict timeliness and safety requirements have to be met in an auditable way.

In the context of the industry case study we propose a model-driven architecture (MDA) system approach that allows to describe dependable data links between systems for decision support and to generate new system versions. With the approach we want to address the following issues:

- *Explicit models* of requirements and target infrastructure as well as tool-supported systems generations are expected to allow exploring design trade-offs in different system variants [3], [4].
- *Software Quality Management:* Explicitly link stakeholder value to development process for role-specific access that supports the quality assurance of requirements models [10] to strengthen the traditionally rather loose connection between the multitude of developer models and in-process products to stakeholder value.
- *System verification and feedback* to explicit models of system infrastructure capabilities: Measure actual system capabilities in lab simulation, test bed, and field [4], [12].

Research contributions of this paper are: a) to provide a real-world prototype study of an approach to explicitly capture stakeholder value propositions and carry them through development, test and operation in an auditable way (as mandated in a safety-critical domain); and b) to discuss benefits and limitations of the proposed MDA approach as an example of a software process improvement initiative.

The remainder of this paper is structured as follows: Section 2 summarizes related work on software development using a model-driven architecture approach. Section 3 introduces the research issues. Section 4 describes the industry case study and Section 5 discusses results of the case study and suggests directions for future work.

2. Model-Driven Architecture Background

The aim of the *Model Driven Architecture* (MDA) is to separate the specifications of system functionality and implementation [13]. The promise of MDA is to create application code from requirements models [14] automatically, instead of writing code manually, and thus avoid common sources of errors and consequently improve the resulting application quality.

Figure 1 shows on the left the structure of the MDA approach: The separation of system functionality and implementation specifications is modeled in the Platform Independent Model (PIM), while separation of that functionality on a specific technology platform is described in the Platform Specific Model (PSM). Using a Computation Independent Model (CIM) the MDA framework can construct the models in a formal way, like UML [7]. The requirements of the future system are described in the CIM, which is refined into the PIM, normally by hand [14], [15]. The main point of the PIM is to specify the structure and the behavior of a system independently of the platform it may be deployed to [13]. The PSM is the result of the PIM transformation. The process refines the PIM based on the specification described in the Platform Module (PM) explaining how to use a specific platform [17].

The main advantages [7] of the MDA framework are (1) results are automatically generated which is expected to improve productivity, development duration, and cost; (2) the developer is likely to pay more attention to CIM and PIM and to develop conceptual models rather than deep logical and technical details; (3) PIM is portable to different target platforms; (4) once a transformation has been developed, it can be reused whenever needed; (5) changes have to be done in the PIM only if the target platform has changed; and (6) new requirements in the CIM are passed to PIM and PSM immediately and changes are reflected automatically [14], [15], [16].

The column “MDA inspired approach” in Figure 1 shows the structure of the approach used in this paper. Main differences to the generic MDA are: the results of the transformations are systems configuration models rather than code. From the structural similarity we expect similar properties of the approach as for the generic MDA.

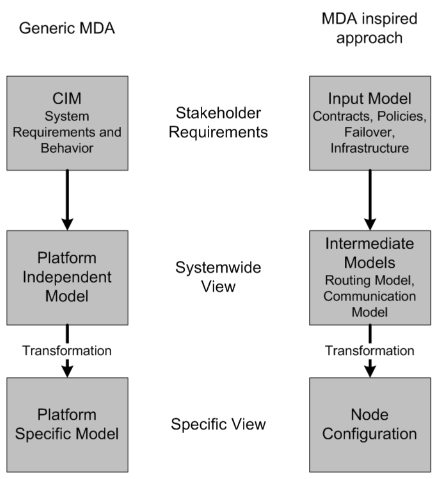

Figure 1. A MDA inspired approach.

3. Research Issues

New software development approaches, such as MDA, are expected to bring benefits to software development like faster or more efficient development. However, from a software quality point of view the question remains whether the means for quality assurance (QA) are comparable to or better than with a traditional approach; e.g., the complexity introduced by the MDA architecture may make QA actually harder.

From the goal to measure and ensure stakeholder-oriented quality of the product and the development process, we derive the following research issues:

1. *Explicit modeling of stakeholder requirements*: To what extent can domain-specific stakeholder value elements be explicitly modeled as input to MDA and QA?

2. *Tool support and QA for requirements transformation:* To what extent can the MDA approach transform the explicit quality requirements models into a running system without significant sources of defects like manual interaction; better quality measurement and feedback on intermediate models during systems development?

3. *Stakeholder-level quality measurement:* How can the required quality levels be measured and assured in the MDA life cycle; i.e., auditing capabilities of the systems development process?

4. Research Application

The case study project, *information sharing network* (ISN), is an industrial prototype that explores new approaches for providing dependable data connections in heterogeneous networks. Stakeholders are decision makers in safety-critical domains, their data providers, and information system developers. The ISN delivers the middleware to connect business applications that provide and/or consume data with defined quality levels. The software engineering process that comes with the ISN has to allow the measurement and feedback of relevant quality aspects at every step during development, test, and operation.

In this section we describe the MDA of the system, the development process, and how software quality can be measured and improved.

4.1. Scenario "Harbor Traffic Control"

The prototype case study, "Harbor Traffic Control", shown in Figure 2 presents a set of business applications and the *ISN:cloud* network consisting of nodes and connected by edges.

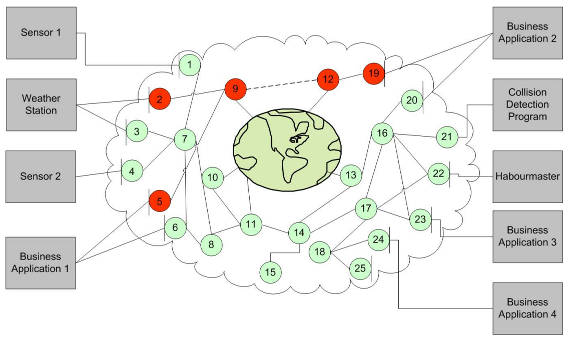

Figure 2. "Harbor Traffic Control" scenario.

There are two types of nodes: red nodes handle highly secure connections only, while green nodes do not provide specific security mechanisms. An edge refers to a network connection with specific characteristics, e.g. bandwidth. Business applications are listed on the left and on the right hand sides that they are loosely coupled, i.e., they do not know anything of each other apart from data providing and consuming contracts. Each application is connected to at least one network node. In our simplified example a business application is a sink service that requires a specific type of data to work properly; or a source service that produces data needed by sink services. The *ISN:cloud* network is a generic base for legacy applications rather than an specifically designed for the scenario.

The real-world scenario describes source services, like *Sensor 1*, *Sensor 2* and *Weather Station;* and sink services like the *Habormaster* or the *Collision Detection Program (CDP)*. The *CDP* needs every 2 seconds data updates provided by 3 reliable alarm sensors that react to conditions out of predefined limits. The *Habormaster* requires reliable sensor data on demand that must not be older than 0.5 seconds, to update the traffic situation overview and to coordinate the ships navigating around the port and harbor facilities, like locks. Based on the stakeholder data requirements inputs a new version of the *ISN:cloud* is generated, tested, and deployed whenever needed; changes are expected on a bi-weekly basis. The range of application data demands is very wide, but all applications have requirements on reliability, timeliness, safety, service quality, failover, performance, auditability, maintainability, and flexibility.

Development of the *ISN:cloud* focuses on explicit stakeholder quality requirements models to define the business application requirements (see Figure 3, "explicit models"):

- **Communication contracts** define the communication offers and needs of business application systems in the *ISN:cloud*. A contract has to identify the business application in the network, provide syntax of attributes for each message type of the business system, and specify the way how the system handles incoming and/or outgoing messages. The most basic form of contract contains information about the exchange trigger (on demand, event, or periodical) and the attributes of the contract (accuracy, confidentiality, urgency, priority).
- **Global policies** reflect interests of the organizations contributing to ISN and ISN-based applications; these policies specify the guidelines the MDA transformation process has to follow when building the system configuration plan. Such guidelines involve parameters concerning global settings (e.g., route secure message over specific secure nodes and edges only, the number of backup routes for failover), optimization criteria (e.g. favor low cost over speed), and operator restrictions (e.g. sensor data has to be routed via a certain node).
- **Failover** parameters define the failover mechanism behavior (automatic or user active) of the *ISN:cloud* to adjust to failures (e.g., node or edge failure) as well-defined graceful performance degradation to the business applications and stakeholders.
- **Infrastructure capabilities** describe the topology of the network, connection capabilities like capacity and type of connections, and useful information on lower-level middleware such as protocols used of the systems that contribute to *ISN:cloud.*

4.2. MDA software development life cycle

The tool support for development and transformation in the MDA approach is the "*ISN:Model Transformation Algorithm" (ISN:MTA*).

Figure 3 presents the system development life cycle, which follows the MDA structure in Figure 1, and the steps to transform and validate stakeholder requirements. Life cycle details can be found in [17] as this paper focuses on the quality assurance aspects of the MDA approach.

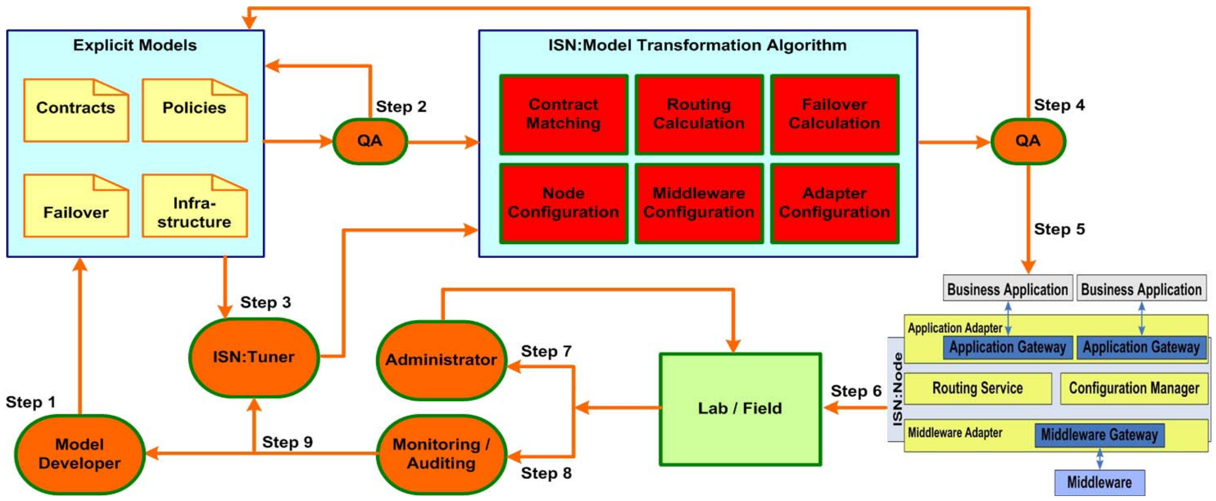

Figure 3. Generic MDA software development life cycle with QA steps.

4.3. MDA Quality Assurance

A major issue in safety-critical systems is quality measurement, quality assurance, and auditing. We describe the key steps in the *ISN* life cycle that deal with stakeholder-relevant QA.

Step 2: Before the models are used as input for the *ISN:MTA* they have to pass the first QA checkpoint where type checks and semantic validation tests ensure formal consistency and validity of the models. If the QA finds contradictions, the errors are reported and the result of the process is fed back to the model descriptions.

Step 4: The configuration plan created by the *ISN:MTA* is checked by another QA checkpoint. Here the intermediate models have to pass a static validity test, i.e., checking whether there is at least one contract matching source for each sink or whether there is a route between two services. Errors are reported, which can be used for correction in the models.

Step 6: The set of nodes that builds the *ISN:cloud* can be tested in several scenarios in a lab environment before rolled out into field operation. Status and performance values are sent to the administrator, who may change configuration settings of *ISN:nodes* directly.

Step 9: The values gained from monitoring the *ISN:cloud* are valuable contributions to the work done by the *ISN:Tuner* and *Model Developer*. Both roles may perform changes based on the results from previous calculations in the hope to improve the quality of the entire *ISN:cloud* and starting from step 1 again.

Although *ISN:MTA* represents another source of error, we think that using the approach does pay off. First of all the *ISN:MTA* life cycle has been designed in a way that allows the installation of several quality assurance checkpoints, the usage of monitoring components and making improvements in the model descriptions by using quality feedback methods offered by the *ISN:cloud*. The *ISN:MTA* allows concentrating the complexity of the system at a central part that can be maintained by few experts. The advantage is in the number of systems using the *ISN:cloud*. The result of corrected problems appearing in one system configuration affects not just the quality of the *ISN:cloud* itself but also any system based on *ISN*.

In the traditional development process, administrators have just a partial view of the entire system and may try to optimize their applications locally. This may result in an overall system behavior that maybe was not intended. However, the proposed MDA approach always has the view over the entire system trying to optimize it in a way by which each of the participating systems can benefit.

Second, the models themselves may contain errors, but similar to specifications in traditional engineering these errors could be found with appropriate verification. An advantage is the fact that those models support role-oriented abstraction; thus the system models are easier to understand and errors are easier to detect.

Although the life cycle contains several QA checkpoints, the ISN life cycle itself may contain errors as well. As these get detected and eliminated, the corrected life cycle will have a positive impact on all those projects using the ISN life cycle.

5. Discussion and Further Work

In an initial prototype case study similar to the "Harbor Traffic Control" scenario sketched in Figure 2, the concepts of the ISN have worked satisfactory according to the life cycle plan described in Figure 3.

MDA Explicit Requirements Models. The explicit modeling of stakeholder requirements was found useful. Compared to traditional development the input models of the MDA gathered all relevant system capability parameters needed to understand whether the stakeholder requirements could be fulfilled. This allowed to provide feedback to the stakeholders on system performance bottlenecks and a better-informed negotiation of stakeholder contracts.

In the rather small example the use of semantic web technology (ontologies) for the description of stakeholder requirements (see Section 4.1) and infrastructure capabilities seemed to introduce considerable overhead compared to the limited added benefit of the semantic reasoning capabilities provided. However, this technology approach can be expected to scale up well to describe several hundreds of contracts and infrastructure elements, which would be hard to handle in the traditional way of reconciling information scattered over many places. Empirical studies are needed to provide evidence on the performance in larger and more complex contexts.

MDA Model Transformation. The MDA transformation of the explicit stakeholder requirements into expressive intermediate models on aspects of the overall systems configuration such as contract matching and routing was found promising but needs more experience with design trade-offs and system tuning. While the simple approaches used for routing in a simple context are comparable to manual routing by experienced administrators, there is room for improvement to use the infrastructure more efficiently and at the same time provide more robust connections. Also, manual systems tuning took considerable effort of the ISN:tuners; considerable effort could be saved by putting some of the tuning parameters into the input models for the MDA, so these parameters can be used by the ISN:MTA transformation algorithm and are part of the systems performance feedback loop (see Figure 3).

A major MDA benefit for design quality comes from the option to cheaply generate system versions that can be analyzed to better understand the trade-offs of different transformation strategies in the case study context, e.g., the valuation of different contract matching options on the traffic volume in the system.

Measurement of quality in MDA. A major improvement of quality definition and measurement have been the role-specific explicit models of stakeholder requirements, the infrastructure capabilities, and the intermediate models coming from the ISN:MTA. These models effectively and efficiently allow the involved stakeholder roles to check the consistency of their models using tool support. Further, the results from running test scenarios documented in lab and field tests in a way that allows efficient quality analysis and comparison of the results with the original assumptions on infrastructure and traffic volume (from contracts and policies) as well as the model assumptions used when creating the intermediate models. Lessons learned from this data analysis in a case study context can be used to update the MDA input models in an auditable way, e.g., on actually measured infrastructure capabilities for cost settlement with network providers, or for design optimization and the diagnosis of problems in the ISN:MTA transformation.

This seems particularly important as developing the ISN:MTA takes learning iterations that depend on accurate feedback that can be well relate to changes in the input data and the transformation algorithm.

Note that the system described in this paper is particularly well suited for a MDA approach as the ISN is a middleware that a) interacts primarily with other systems rather than human users; b) follows well-defined behavior; and c) has stakeholder who mostly can provide consistent value propositions for rating a set of system variants.

Future work. Next steps after developing the core functionality of the MDA approach are systematic empirical studies to ensure the correctness and sufficient performance of the ISN:MTA and the resulting system configurations. An important aspect is early modeling for reliability design to consistently carry dependability concerns from the early to the late stages of software engineering.

For organizations that use a traditional systems development approach a major question is when it is worthwhile to introduce a new development approach, such as MDA, which are expected to bring benefits to software development like faster or more efficient development. Again, empirical studies are needed to get evidence on the actual benefits and risks in comparable settings.

References

[1] S. Biffl, A. Aurum, B. Boehm, H. Erdogmus, and P. Grünbacher (eds.) (2005) *Value-Based Software Engineering*, Springer Verlag, 2005.

[2] B. Boehm, L. Huang, A. Jain, and R. Madachy, "Quality as Stakeholder Value", *in Proceedings of the Second Workshop on Software Quality*, 2004, pp 1-3.

[3] S. Kan; *Metrics and Models in Software Quality Engineering*, 2nd Edition; Addison Wesley, 2002.

[4] N. Nagappan, L. Williams, M. Vouk, and J. Osborne; "Early Estimation of Software Quality Using In-Process Testing Metrics: A Controlled Case Study"; in *Proceedings of the third workshop on Software Quality*, 2005, pp. 1-7

[5] S. Chulani , B. Ray , P. Santhanam , and R. Leszkowicz, "Metrics for Managing Customer View of Software Quality", in *Proceedings of the 9th International Symposium on Software Metrics*, 2003, p.189

[6] P. Grünbacher, S. Köszegi, and S. Biffl "Stakeholder Value Propostion Elicitation and Reconciliation", in: S. Biffl, A. Aurum, B. Boehm, H. Erdogmus, and P. Grünbacher (eds.) (2005) *Value-Based Software Engineering*, Springer Verlag, 2005, p. 133-154.

[7] J.-N. Mazón, J. Trujillo, M. Serrano, and M. Piattini, "Applying MDA to the Development of Data Warehouses" in *Proceedings of the 8th ACM international workshop on Data warehousing and OLAP*, 2005, pp. 57-66

[8] A. D'Ambrogio, "A Model Transformation Framework for the Automated Building of Performance Models from UML Models" in *Proceedings of the 5th international workshop on Software and performance,* 2005, pp. 75-86.

[9] A. Balogh and A. Pataricza, "Quality-of-Service Modeling and Analysis of Dependable Application Models", Jan. 2007, http://www.cs.colostate.edu/csduml2006/CSDUML06-FinalPapers /nr1_side1-12.pdf

[10] H. Kitapci, B. Boehm, P. Grünbacher, M. Halling, and S. Biffl, "Formalizing Informal Stakeholder Requirements Inputs", in *Proceedings of the 13th international INCOSE Symposium*, 2003

[11] M. A. Babar and S. Biffl, "Eliciting Better Quality Architecture Evaluation Scenarios: A Controlled Experiment On Top-Down vs. Bottom-Up", in Proceedings of the 2006 ACM/IEEE international symposium on empirical software engineering, 2006, pp. 207-316

[12] I. Rus, M. Halling, and S. Biffl "Supporting Decision-Making in Software Engineering with Process Simulation and Empirical Studies", *Int. Journal of Software Engineering and Knowledge Engineering*, vol. 13, no. 5, pp. 531-545, October 2003.

[13] J. Miller and J. Mukerji, "Model Driven Architecture (MDA)", January 2007, http://www.omg.org/docs/ormsc/01-07-01.pdf

[14] S. Mellor, K. Scott, A. Uhl, and D. Weise, *MDA distilled: principles of Model Driven Architecture.* Addison-Wesley, 2004.

[15] A. Kleppe, J. Warmer, and W. Bast, *MDA Explained. The Practice and Promise of the Model Driven Architecture.* Addison-Wesley, 2003.

[16] D.S. Frankel, Model Driven Architecture. *Applying MDA to Enterprise Computing.* Indianapolis, Indiana. Wiley. 2003.

[17] [Online document] May 2007, Available at http://www.complang.tuwien.ac.at/richard/SW