Multimodel-Driven Software Engineering for Evolving Enterprise Systems (Position Paper)

Richard F. Paige[1], Radu Calinescu[1], Dimitrios S. Kolovos[1], Nicholas Matragkas[1] and Dave Cliff[2]

[1] Department of Computer Science, University of York, UK.
{richard.paige, radu.calinescu, dimitris.kolovos, nicholas.matragkas}@york.ac.uk
[2] Department of Computer Science, University of Bristol, UK.
dc@comp.bristol.ac.uk

Abstract. We advocate the use of multimodel-driven software engineering for the principled evolution of enterprise systems whose stakeholder concerns are captured using multiple interdependent models. *Enterprise systems* that evolve are increasingly common in healthcare, transportation, e-government and defense. These important systems must be regularly extended with new components satisfying interdependent functional, governance and quality-of-service (QoS) requirements that are modelled using different domain-specific languages. We describe key challenges associated with modelling, reasoning about QoS properties, and evolving such systems. The concepts of this engineering paradigm are presented in the context of a statistical reporting project carried out in collaboration with healthcare organisations.

1 Introduction

The recent advent of technologies ranging from cloud and mobile computing to smart sensor networks has led to the emergence of new types of data and applications at an extremely fast rate. This trend is amplified further by equally rapid changes in public information governance. The open data movement, in part spearheaded by the UK Government [1] and recently embraced by all G8 Governments [2], has opened up a wide range of public datasets for research and commercial use. Healthcare, transportation, education and local government are only a few of the areas in which new public datasets are released (e.g., `data.gov`, `data.gov.uk`) on a daily basis.

These developments have created business and research exploitation opportunities for both public and commercial organisations. To exploit these opportunities – and to comply with changes in information governance and other requirements – such organisations must *evolve* their enterprise systems on a regular basis. Information systems supporting new business processes must be engineered and integrated *on the fly* with existing enterprise systems, and must then be updated frequently in response to new stakeholder requirements and governance policies. Plausible examples of such evolving enterprise systems are

encountered in the health and social care domain. In the UK for instance, the recently enacted Health and Social Care Act 2012 [3] required an organisation to "*establish and operate a system for the collection or analysis of information of a description specified in the request*" of "*any person*", at any time.

The stringent demands of *evolving enterprise systems* cannot be achieved using today's software engineering approaches. Traditional enterprise system engineering approaches comprise manual processes that cannot respond to such demands in a timely manner, and are costly and error prone. Model-driven software engineering - which automates development processes by synthesising software artefacts from models - is challenging to apply, as the concerns of new information systems (e.g., functionality, governance and quality-of-service) cannot easily be captured by a single model.

We envisage that overcoming the limitations of existing approaches and achieving the goals of evolving enterprise systems requires *multimodel-driven software engineering* (MMSE). This software engineering paradigm will automate key processes of evolving enterprise systems whose concerns are described by multiple interdependent models, when these models are specified by different stakeholders, in different domain-specific languages. Significant research challenges must be addressed to achieve this vision. A key concern is integrating QoS models throughout the engineering lifecycle. In particular, the research community will need to address the open research questions of how to devise multimodel transformation techniques that consider inter-model dependencies (where some are QoS models), and how to co-evolve sets of interdependent metamodels. Another key challenge is the joint analysis of quality-of-service (QoS) models for dependability, performance and resource usage, to identify system configurations that deliver effective QoS trade-offs.

Our paper summarises these key challenges, and sets a research agenda for the delivery of the software engineering formalisms, techniques and tools needed to automate the development, analysis, adaptive configuration and evolution of information systems specified by sets of interdependent functional, governance and QoS models.

2 Background

Recent advances in model-driven software engineering (MDSE) address many challenges of developing traditional software systems. MDSE is a software development paradigm that aims to use (software) models as the main development artefact instead of code [4]. Achieving this aim within a problem domain involves the use of domain-specific languages (DSLs) to define models of the systems to develop [5]. Automated transformations can be applied to them to generate models of lower abstraction levels, which ultimately can be transformed into code. The advantages of MDSE over other approaches to software development include significant improvements in software quality and development efficiency, and increased reusability of software components [4, 5]. The research efforts to

exploit these advantages produced a broad range of effective MDSE modelling languages and tools (e.g., [6, 7]).

More recently, the MDSE paradigm was extended to also cover the post-development stages of the software lifecycle. In this extended MDSE approach, models continue to be used as the primary artifact in the maintenance and evolution of software systems. A key challenge of using MDSE in this context is that the models and metamodels used at different levels of abstraction may change asynchronously as the software system evolves, creating ripple effects on related artefacts such as model transformations and validation constraints. Different aspects of this challenge have been addressed by recent research on *model management* [8], *model-metamodel co-evolution* [9], and *incremental and bidirectional model synchronisation* [10].

In parallel with these advances, the *performance* (or *QoS*) *engineering* area of MDSE uses performance, reliability and cost models as key artifacts in all stages of the software lifecycle [11, 12]. Model-driven QoS engineering aims to ensure that software systems satisfy their QoS requirements "by construction" when initially delivered [13], and continue to do so as they *self-adapt* in response to changes in environment, requirements or internal state [14]. QoS models may be developed explicitly or may be synthesised from annotated variants of the structural and behaviour models used in the traditional MDSE process [15], and typically need to be updated continually at run time based on the observed system behaviour [16]. Such efforts can be linked to relevant modelling standards such as the MARTE and QoS profiles for UML [17, 18].

3 A motivating scenario

In the UK, there are organisations responsible for accumulating and managing data related to health and social care. They act as a trusted repository and broker for such information, and also provide statistical expertise and domain knowledge relevant to such data. In particular, they may produce and provide statistical reports on health and social care data to a variety of stakeholders. Stakeholders may include casual browsers of health and social care data – who may simply be interested in learning what information or reports are available – to sophisticated commercial users of data/reports, who may base important commercial decisions on what they acquire from this trusted broker. Additionally, the organisation may be required to provide information to government departments or ministers. As such, information in the form of customised reports and data may be requested by commercial, public or governmental stakeholders, at any time, and the organisation must be able to respond to such requests. In particular:

- QoS requirements may be applicable to requests coming from stakeholders, e.g., a certain data quality, a report available within a certain hard or software deadline.
- Information governance requirements and policies may also be applicable, requiring the health and social care organisation to determine whether prospec-

tive customers are permitted to access either raw data, or pseudonymised data, of a certain type or kind. Checking compliance with information governance policies is particularly time consuming; it would be beneficial to be able to determine if a customer was permitted to access particular data items before continuing with the rest of the procurement process, but sometimes this is not possible – research (see the next point) may need to be carried out in parallel with checking compliance.
- A customer may request an existing type of report, or a report that is similar to one that is currently available. But they may also request reports that are new and novel, for which a *research process* must be carried out. In such a process, the organisation may have to 'buy in' expertise it may not have, may need to investigate new statistical methods or practices, and may need to synchronise the research process with parallel business processes that are in place to ensure proper billing and compliance with governance policies and regulations.

Such an organisation would potentially benefit from the application of different models: for capturing business processes; for capturing QoS properties and requirements; for capturing data and interrelationships; and for capturing information governance rules. Such models could be used for understanding the complexity of the organisation, analysing the effectiveness or potential bottlenecks in a particular stakeholder interaction, and for analysing the effects of *evolution*, e.g., through new QoS requirements or stakeholder requirements. However, the stakeholders interested in the organisation's research, data and results can change at any time, leading to new report/data requirements. As such, the models relevant to the organisation should be considered highly volatile, and may be subject to evolution at any time.

4 Multimodel-driven software engineering

Fast and robust evolution represents a key requirement for a growing number of important enterprise systems. Achieving this requirement needs software engineering technology capable of developing "on demand" information systems that satisfy the overlapping concerns of different stakeholders, and of integrating them into evolving enterprise systems on the fly. We envisage that this role will be played by *multimodel-driven software engineering* (MMSE) – a novel software engineering approach that will combine:

- *multimodel-driven automated code generation*—to take advantage effectively of interdependent models associated with different but overlapping areas of concern, and specified by different classes of stakeholders in distinct and co-evolving domain-specific languages;
- *multimodel-driven QoS engineering*—to ensure that evolving enterprise systems achieve the performance, dependability and cost-related requirements specified across the interdependent models mentioned above.

Significant research is required to provide the theoretical foundation for the MMSE vision and open-standards MMSE tools that realise this theory. Extending the applicability of MDSE to large-scale, evolving software systems whose characteristics spread multiple domains is a hard open problem [4, 19], whose solution involves addressing several major challenges:

1. *Transformations of interdependent models.* Identifying the dependencies among multiple concern-specific models and devising model transformations that comply with these dependencies is notoriously difficult and error prone [4].
2. *Co-evolution of interdependent metamodels.* Managing the co-evolution of heterogeneous sets of interdependent metamodels and the synchronisation of their associated models is a complex problem that is not addressed by existing MDSE approaches [19].
3. *Cross-analysis of interrelated QoS models.* Analysing the relationships and tradeoffs between interacting performance, dependability and cost attributes specified across multiple models is a complex and non-scalable task that is deemed a major challenge for QoS engineering [11].

The research agenda in the next section suggests research objectives that need to be pursued by the software enginnering community in order to tackle these challenges and to realise the vision of multimodel-driven software engineering.

5 MMSE research agenda

Research objective 1. To develop a theoretical foundation comprising formalisms, algorithms, model transformations and techniques for multimodel-driven software development, and for the management and co-evolution of interdependent metamodel and model sets.

Addressing this objective requires the development of new techniques for multi-modelling, including the following multimodel-aware code generators and co-evolution approaches:

1. DSLs for modelling the architecture, business processes and governance rules associated with enterprise information systems. These DSLs must be defined using *generic modelling concepts* (generic in the sense that they can be used throughout the enterprise domain, and derived from existing modelling languages or frameworks such as ArchiMate and TOGAF), wherein metamodels are specified in terms of *required* arguments. This will allow substantial sharing between DSLs.
2. Formalism based on OCL and OCL extensions such as the Epsilon Validation Language [8], for defining *generic inter-dependencies* between the DSLs from (1). These formalisms must allow instantiation of the generic modelling concepts from (1) and support the specification of *strong* (i.e., must-hold) and *weak* (i.e., may-hold) consistency rules on models. These formalisms need to be elaborated to allow such rules to be defined for QoS models.

3. A theory of *generic code generation* for transforming multi-models into platform-specific enterprise information systems application code. The transformations must be capable of instantiating generic parameters and of producing *monitors* to be used to detect whenever QoS properties (not guaranteed to hold via construction) have been violated. The novelty here is that the code generation must take into account QoS models as well as other enterprise systems domain models.
4. A theory of co-evolution for generic multi-models, investigating patterns of generic metamodel change (including changes to parameters), as well as defining strategies for co-evolving generic models, metamodels and interdependencies (from (1) and (2)) and code generators (from (3)).

Research objective 2. To devise a suite of scalable QoS engineering techniques for: (i) co-analysing the relationships between QoS concerns specified through multiple mathematical models; and (ii) identifying effective tradeoffs between conflicting QoS concerns. The real challenge here is to be able to manage QoS models in a structurally identical way to other MDSE models.

Achieving this objective will require the use of a combination of stochastic, Markovian and queueing models to *co-analyse QoS properties* including: (i) dependability (e.g., availability and reliability); (ii) performability (e.g., response time and throughput); and (iii) resource usage (e.g., battery power, data storage capacity, bandwidth and cost). New research results are needed to enable the co-analysis of interdependent QoS attributes specified by heterogeneous mathematical models, in order to identify and exploit effective tradeoffs between the dependability, performability and resource usage requirements of complex software systems. To accomplish this, the research must develop the following new theoretical contributions for the co-analysis of interdependent QoS models:

1. Techniques for the automated co-extraction of stochastic, Markovian and queueing QoS models from enhanced structural models specified using QoS-related UML profiles [20, 17, 18]. These techniques could build on results from [21, 22, 12, 15, 23], extending them with support for extracting new types of QoS models, and with the ability to identify and encode the interdependencies among multiple QoS models.
2. A high-level formalism for specifying (i) the QoS constraints that an information system must comply with at all times; and (ii) the *utility* associated with the achievable trade-offs between the dependability, performability and resource usage of these systems.
3. Algorithms for translating the QoS constraints and utility levels that system developers and operators provide in the high-level formalism from (2) into verifiable low-level properties expressed in temporal logics augmented with probabilities, event rates, costs and rewards.
4. Quantitative analysis techniques for the co-analysis of the formally expressed sets of QoS properties from (3) against the mathematical QoS models from (1). These techniques will need to take into account the dependencies among multiple QoS models, in order to identify trade-offs between performabil-

ity, dependability and resource usage that satisfy the QoS constraints and achieve a high utility.

Research objective 3. To develop an open-standards MMSE platform.

This multimodel-driven software engineering platform needs to include:

1. An integrated toolset supporting multimodel-driven software development, QoS engineering and metamodel/model management. Some of this technology already exists, but extensions to existing model management platforms (such as ATL or Epsilon) need to be made to fully support QoS models, particularly for code generation.
2. A methodology comprising methods for the effective engineering of evolving enterprise systems.

The MMSE platform must integrate, implement and hide the complexity of the formalisms, algorithms and techniques devised by the research described under the research objectives 1 and 2, enabling developers of large-scale evolving software systems to exploit these theoretical results effectively without the need for expert training. Extending an established platform (e.g., the Eclipse platform) for achieving this would speed up the adoption of MMSE considerably.

Research objective 4. To employ MMSE in the development of exemplar evolving enterprise systems.

It is essential that emerging MMSE techniques and tools are evaluated in the development of real-world evolving enterprise systems, as part of joint projects between the research community and organisations whose business processes are supported by such systems. The wide dissemination of the results of these projects, ideally as open-sourced exemplars, is essential to the early adoption of the much needed MMSE technology.

6 MMSE technology delivery in health and social case

We recently embarked on a new project to devise MMSE technology components, and to use them for the development of a prototype evolving enterprise system, in collaboration with a health and social care partner. This will allow us to validate, refine and extend our preliminary version of several MMSE technology components, and to contribute significantly to the efforts to achieve the objectives stated in the UK Health and Social Care Act 2012 [3]. As part of this, our planned MMSE platform will be instantiated with specific types of models relevant to a health and social care industry partner. In particular, we envision building QoS models relevant to assessing the timeliness of treatment or care, information governance models capturing the policies, rules and procedures related to health and social care, etc. This approach is summarised in Fig. 1.

Our research will be carried out while being driven by scenarios and use cases from the health and social care domain. A plausible use case that we have available involves using multimodel-driven software engineering to support *reporting*

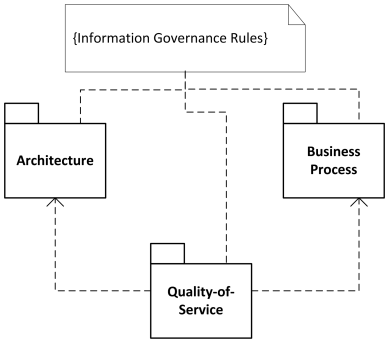

Fig. 1. Conceptual model of instantiation of the MMSE approach

scenarios. Such scenarios involve collecting different types of raw pseudonymised data (e.g., number of incidents of stroke in a particular region) as well as statistical data (e.g., percentage of residents of a certain age being supported by a care provider in a region). Such data may need to be collated and presented in a report for a particular stakeholder group - like a health trust (who may have responsibility for reporting care outcomes to government), or even a government minister.

The data and statistics either gathered or produced for these reports must be audited and validated, produced within strict time bounds, for specific costs (for example, some reports may be produced for a commercial organisation for a fee). The type, quantity and complexity of the data that is being managed, and the reports that are being generated, may change at any time – that is, new IT systems may be brought in to the report-generating organisation (e.g., from new health care providers) and integrated into the report generating process.

Such systems clearly require handling of multiple models and QoS concerns, as well as the need to handle evolution of models. What is particularly challenging about this scenario is that the types of models evolve in different ways and at different rates. Consider Fig. 1: the architecture of such an enterprise system at one of our health and social care partners does not change frequently, but the QoS requirements do (e.g., on a problem-by-problem basis: different customers require their results at different rates), and the information governance rules also change frequently (though not as frequently as QoS models) due to new legislation and Department of Health rules. As such, the multimodel evolution problem is extremely challenging in this context.

7 Conclusions

We have motivated the need for and the challenges of multimodel-driven software development (MMSE), particularly focusing on evolving enterprise systems. We have described a research agenda for developing the MMSE theory and tools required to work with such systems, as well as a motivating scenario in the health

and social care domain. Currently, we are developing the MMSE infrastructure for addressing the research objectives of this agenda in the health and social care domain, and are identifying suitable enterprise systems scenarios with two practicing health and social care partners.

References

1. Nigel Shadbolt, Kieron O'Hara, Tim Berners-Lee, Nicholas Gibbins, Hugh Glaser, Wendy Hall, and M. C. Schraefel. Linked open government data: Lessons from data.gov.uk. *IEEE Intelligent Systems*, 27(3):16–24, 2012.
2. UK Cabinet Office. G8 Open Data Charter and Technical Annex, G8 communiqué and documents, June 2013. `https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/207772/Open_Data_Charter.pdf`.
3. The Health and Social Care Information Centre, Part 9, Chapter 2 of the Health and Social Care Act 2012, 2012. Available at `http://www.legislation.gov.uk/ukpga/2012/7/enacted`.
4. Robert France and Bernhard Rumpe. Model-driven development of complex software: A research roadmap. In *ICSE Workshop Future of Softw. Eng.*, pages 37–54.
5. M. Fowler. *Domain-Specific Languages*. Addison-Wesley, 2010.
6. Dimitrios S. Kolovos, Richard F. Paige, and Fiona Polack. The Epsilon transformation language. In *ICMT 2008*, pages 46–60.
7. Dave Steinberg, Frank Budinsky, Marcelo Paternostro, and Ed Merks. *EMF: Eclipse Modeling Framework*. Addison-Wesley, 2008.
8. Dimitrios S. Kolovos, Richard F. Paige, Louis M. Rose, and James R. Williams. Integrated model management with Epsilon. In *ECMFA 2011*, pages 391–392.
9. Louis Rose, Dimitrios Kolovos, Richard Paige, and Fiona Polack. Model migration with Epsilon Flock. In *ICMT*, LNCS 6142. 2010.
10. H. Giese and R. Wagner. From model transformation to incremental bidirectional model synchronization. *Softw. & Syst. Modeling*, 8(1):21–43, 2009.
11. R. Calinescu, C. Ghezzi, M. Kwiatkowska, and R. Mirandola. Self-adaptive software needs quantitative verification at runtime. *Commun. ACM*, 55(9):69–77, 2012.
12. Stephen Gilmore, Laszlo Gonczy, Nora Koch, Philip Mayer, Mirco Tribastone, and Daniel Varro. Non-functional properties in the model-driven development of service-oriented systems. *Softw. & Syst. Modeling*, 10(3):287–311, 2011.
13. Steffen Becker, Heiko Koziolek, and Ralf Reussner. The Palladio component model for model-driven performance prediction. *J. of Systems & Softw.*, 82(1):3 – 22, 2009.
14. Radu Calinescu, Lars Grunske, Marta Kwiatkowska, Raffaela Mirandola, and Giordano Tamburrelli. Dynamic QoS management and optimisation in service-based systems. *IEEE Trans. Software Eng.*, 37(3):387–409, May 2011.
15. M. Tribastone and S. Gilmore. Automatic translation of UML sequence diagrams into PEPA models. In *QEST 2008*, pages 205 –214, 2008.
16. I. Epifani, C. Ghezzi, R. Mirandola, and G. Tamburrelli. Model evolution by runtime adaptation. In *ICSE 2009*, pages 111–121.
17. Object Management Group. UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms v1.1, 2008.
18. Object Management Group. UML Profile for Modelling and Analysis of Real-Time and Embedded Systems (MARTE) v1.1, 2011.

19. Ian Sommerville, Dave Cliff, Radu Calinescu, Justin Keen, Tim Kelly, Marta Kwiatkowska, John McDermid, and Richard Paige. Large-scale complex IT systems. *Commun. ACM*, 55(7):71–77, 2012.
20. Object Management Group. UML Profile for Schedulability, Performance and Time v1.1, 2005.
21. Radu Calinescu and Marta Kwiatkowska. CADS*: Computer-aided development of self-* systems. In *FASE 2009*, pages 421–424, 2009.
22. M. L. Drago, C. Ghezzi, and R. Mirandola. Towards quality driven exploration of model transformation spaces. In *MoDELS 2011*, pages 2–16.
23. Murray Woodside, Dorina C. Petriu, Dorin B. Petriu, Hui Shen, Toqeer Israr, and Jose Merseguer. Performance by unified model analysis (PUMA). In *5th Intl. Workshop on Software and Performance*, pages 1–12. ACM, 2005.