Modelling and Enterprises
The Past, the Present and the Future

Vinay Kulkarni[1], Suman Roychoudhury[1], Sagar Sunkle[1], Tony Clark[2] and Balbir Barn[2]
[1]*Tata Consultancy Services, Pune, India*
[2]*Middlesex University, London, U.K.*

Keywords: Modelling, Meta Modelling, Model-Driven Development, Enterprise Systems, Adaptation, Analysis, Simulation.

Abstract: Industry has been practicing model-driven development in various flavours. In general it can be said that modelling and use of models have delivered on the promises of platform independence, enhanced productivity, and delivery certainty as regards development of software-intensive systems. Globalization market forces, increased regulatory compliance, ever-increasing penetration of internet, and rapid advance of technology are some of the key drivers leading to increased business dynamics. Increased number of factors impacting the decision and interdependency amongst the key drivers is leading to increased complexity in making business decisions. Also, enterprise software systems need to commensurately change to quickly support the business decisions. The paper presents synthesis of our experience over a decade and half in developing model-driven development technology and using it to deliver several business-critical software systems worldwide.

1 INTRODUCTION

Business applications typically conform to a layered architecture wherein each layer encapsulates a set of concerns and interfaces with adjoining architectural layers using a well-defined protocol. Typically, the architectural layers are wired together by middleware infrastructure that supports message passing in a variety of architectures such as synchronous, asynchronous, publish-subscribe etc. As a result, developing a distributed application demands wide-ranging expertise in distributed architectures and technology platforms which is typically in short supply. Large size of application further exacerbates the problem. Moreover, documenting critical design decisions is always sacrificed at the altar of delivery deadlines. Therefore maintenance of such systems becomes a nightmare especially when some key members have to leave the project or have to revisit a part of the system that have not received attention for a long time (Naur, 1985).

To address some of the challenges mentioned above, we have been applying MDE techniques for developing database-intensive enterprise systems using high-level models (Kulkarni and Reddy, 2008). These models capture some of the critical design decisions along multiple dimensions namely functionality, technology and architecture. A set of code generators transform these high-level models into low-level implementation encoding the various design decisions suitably. Thus, models help to shift the focus of application development from code to a higher level of abstraction promising enhanced productivity and quality.

In the remaining part of the paper, we begin by taking a look at the extent to which modelling is practiced in enterprises today and various uses these models are put to. We then discuss what sorts of models will be required to meet the needs of future enterprises and what uses can they be put to. Finally we conclude by presenting an analysis of key investigations necessary for realizing a model-driven enterprise.

2 THE PAST

Models-as-pictures has probably been the most common and widespread use of modeling techniques in enterprises. Here, models provide a common language for bridging business domain and software development worlds (Hailpern and Tarr, 2006). *Models-as-high-level-specifications* has recently witnessed increased following among practitioners (Hailpern and Tarr, 2006); (Hutchinson et al., 2011). Multiple variants of this usage are noticed, for instance, Models are automatically transformed to derive partial implementation to be taken to completion using code-centric development processes (known as code completion) with models forgotten hereafter or maintained so that changes introduced during code-completion can be taken back automatically to models (known as round-trip-engineering) (Medvidovic et al., 1999); and complete implementation is derived from models through model-transformation with models remaining primary SDLC artefacts (Kulkarni and Reddy, 2003). *Models-as-executable-artefacts* is the least common of all usages and that too in niche domains of life-critical systems (Rumpe, 2004).

Enterprises use IT systems principally to obtain mechanical advantage through automation of repetitive processes/tasks. As enterprises have traditionally valued stability, IT systems have been designed/architected so as to result in low maintenance costs. The underlying assumptions being: requirements of the IT system are fully known a priori and they are unlikely to change significantly during the lifetime of the application (complete-knowledge-hypothesis). Change requests are assumed to be few and far between, and each change is assumed to have small ripple effect. Therefore, high analysis/design cost for IT systems is justifiable and acceptable as long as the maintenance cost remains a tiny fraction of the former. Under complete-knowledge-hypothesis it is possible to know about foreseeable enough future and encode this knowledge into the implementation of IT systems using techniques such as parameterization, decision look-up tables, lazy instantiation, delayed binding etc. Thus, it shouldn't come as a surprise that *Models-as-high-level-specifications* approach remains the most widely adopted MDE approach by industry practice. Here, the focus had been on coming up with modelling languages (metamodels/DSLs etc) that are necessary and sufficient for automatic derivation of IT system implementation there from (France and Rumpe, 2007).

Model-based code generators compile the model specifications into a desired implementation using model-to-model (QVT, 2011) and model-to-text (MOFM2T, 2008) transformations. The proven idea of retargetable code generation helps deliver the same model into multiple technology platforms as long as care is taken to keep the model agnostic of platform concerns. Moreover, model-to-model and model-to-text transformation specification languages enable declarative specification of a model-based-code-generator which can either be interpreted for code generation or execution (Kulkarni and Reddy, 2008).

3 THE PRESENT

Globalization forces and increased connectedness have led to increased business dynamics and shortened time-to-market windows for business opportunities. Thus, IT systems designed for operation in an inherently stable environment are becoming a misfit (Truex et al., 1999). Moreover, we discovered that no two applications, even for the same business intent such as straight-through-processing of trade orders, back-office automation of a bank, automation of insurance policies administration, etc., are identical. Though there exists a significant overlap across functional requirements for a given business intent, the variations are manifold too.

Software Product Line Engineering (SPLE) attempts to address these needs by shifting the focus of application development from ground-up coding to assembly of pre-defined components (Kang et al., 1990). The idea is to identify *what* changes *where* and *when* in system functionality – the *what* leads to the variations, the *where* leads to the variation points, and the *when* leads to internally consistent set of *what-to-where* bindings. However, IT systems tend to vary along multiple dimensions - functionality, business process, extra-functional characteristics, and implementation platform to name only a few (Kulkarni and Reddy, 2003). Therefore, the notion of '*what* changes *where* and *when*' needs to be addressed along every dimension and then across them all at the application level. In theory, all it means is to define Meta Object Facility describable metamodels for each dimension but, as of now, there is no evidence of this issue being addressed at industry scale. In fact, modeling of/for extra-functional characteristics is pretty much in infancy and variability management as well as composition concerns are yet to be properly addressed for business processes though some work is reported (Kulkarni and Barat, 2010) (Barat and Kulkarni, 2011). Though *feature model* has become a popular notation for describing variability (Kang et al., 1990), there is no handle on tracing features to application specification and/or implementation artefacts. Ideally, feature should be a first class concept in realizing product lines so that all software development life cycle (SDLC) phases can be feature-centric (and hence time- and effort-optimal) and it should be possible to compose application specification/implementation from feature specification hierarchically ad infinitum (Sunkle, 2011). Enterprise IT systems constitute an ill-defined or hard-to-be-fully-defined space. As a result, complete-knowledge-hypothesis, the cornerstone for SPLE, does not hold. Therefore, there is a need to support product-line-by-evolution as opposed to product-line-by-design (Kulkarni, 2010); (Kulkarni et al., 2012).

4 THE FUTURE

4.1 Modelling Language Engineering Platform

From our past experience in delivering enterprise systems we have found that no two enterprises are exactly alike; it was not possible to meet their functional demands - even for identical business intent such as an order processing system for a financial services organization, policy administration system for an insurance organization, retail banking system for a bank, etc., - with one software system. In traditional code-centric approaches, it would mean introducing suitable changes in a copy of the implementation. In a model-driven approach, it means introducing changes in the various models, metamodels and the model-based code generators. Thus, the problem of evolutionary maintenance of application code gets transformed into evolutionary maintenance of models, modeling languages and model-processing infrastructure, and hence the need for a *modeling language engineering platform*. For want of space, we direct readers to (Kulkarni et al., 2012) for details of the platform.

Much of the core technology to implement such a platform is already available. For instance, Eclipse can provide the backbone plug-in architecture for the platform. Eclipse's eCore is a good starting point for the reflexive meta metamodel. Text-based (meta) model editors can be realized with little modification, if at all, to the various model editors available. OMG QVT (QVT, 2011) and OMG MOFM2T (MOFM2T, 2008) should suffice as specification languages for model-to-model and model-to-text transformation respectively. Both have many implementations available - licensed as well as freeware variety. In OCL (OCL, 2012), there exists a sophisticated declarative mechanism to specify model constraints. However, it is possible to imagine a situation where a new constraint specification language seems appropriate. Therefore, the platform should have the capability to define another constraint specification and execution mechanism.

The proposed modelling language engineering platform will provide the minimal tooling infrastructure for improving productivity of current MDE practitioners. Also, its existence is likely to make MDE enthusiasts to 'take the plunge' so to say. The high level of standardization should help develop MDE community for and around the proposed platform. We believe development (and continuous maintenance) of the proposed platform is best supported through open source community model.

4.2 Towards Formal and Precise Enterprise Architectural Modelling

Economic and geo-political uncertainties are putting increasingly greater stress on frugality and agility of enterprises. Large size and increasing connectedness of enterprises is fast leading them to a system of systems which is characterized by high dynamics and absence of a know-all-oracle. Multiple change drivers are resulting in increasingly dynamic operational environment for enterprise IT systems, for instance, along Business dimensions the change drivers are dynamic supply chains, mergers and acquisitions, globalization pressures etc., along Regulatory compliance dimension the change drivers are Sarbanes Oxley, HiPAA, Carbon footprint etc., and along Technology dimension the change drivers are Cloud, smartphones, Internet of things etc. At the same time, windows of opportunity for introducing a new service/product/offering and/or for adapting to a change are continuously shrinking. Furthermore, business-critical nature of IT systems means the cost of incorrect decision is becoming prohibitively high and there is very little room for later course-correction. Therefore it is important that we look beyond the traditional model-based generative/SPLE based techniques that we have been using in the past and put more emphasis on understanding of the target organizational environment including its business, IT systems, and stakeholder perspectives. In other words, *model the whole enterprise*. Formal and precise enterprise architecture modelling is an important step towards realizing this goal.

To translate business vision and strategy into effective enterprise change by creating and communicating the models centered on business and IT, a set of techniques are used, referred to as Enterprise Architecture (EA) techniques (IEEE 1471, 2000). Irrespective of the architectural methodology followed by an EA technique, there exist a few shortcomings in current EA techniques. Architectural artefacts in current EA techniques are only documents used as reference material by enterprise architects to communicate with various stakeholders for achieving goal such as Business-IT alignment. These models are not machine-manipulable. An enterprise architect is supposed to use these artefacts and his knowledge and experience in achieving enterprise-specific goals.

None of the available EA techniques provides a mechanism to evaluate the technique itself as it is applied to an enterprise. Some EA frameworks provide an assessment framework, but its use is again dependent on the knowledge and experience of the enterprise architect. This means that there is really no guarantee that these techniques will lead to correct EA.

These and other observations make clear that applying these EA techniques to an enterprise is a highly person dependent activity with complete reliance on the enterprise architect's knowledge and experience. Furthermore, validation of goals, such as business-IT alignment, is carried out in a blue-print way in current EA techniques (Wagter et al., 2012). It means that if the enterprise architect feels, based on his knowledge and experience, that an enterprise has been architected according to principles laid out by these EA techniques; then goals such as business-IT alignment have been accomplished by definition. An enterprise may also strive for other goals such as adaptability or cost optimality, for which no mechanism is provided by current EA techniques to prove that a property is satisfied across the enterprise.

Also, the as-is state of an enterprise captured in current EA techniques is not machine-manipulable. The various means of architectural description rely on the expertise of the enterprise architect to provide a path to the desired to-be state of the enterprise (Rolland et al., 1999). Essentially, the problem with regards to enterprise modeling boils down to - what help can be provided so that relatively less experienced person will be able to function at the level of an experienced and knowledgeable enterprise architect in applying EA techniques to enterprises?

4.3 Enterprise Adaptation

With enterprises having to become increasingly dynamic, their supporting IT systems are becoming increasingly complex. Ever-shortening window of opportunity means supporting IT systems need to adapt quickly. Business-critical nature of IT systems means there is no room for an error in what should the adaptation be and how should it be effected. Software engineering community has been focusing on mechanisms to support the latter, but, as of now, the former is still the preserve of gurus. Given the size and complexity of typical enterprises, even experts find it difficult to determine which adaptation would be the best response, as per the chosen criterion, for a given set of changes. Therefore, we strongly believe that modeling community should focus on providing help so as to make this problem more scientific and hence tractable.

Ideally, the more automatically a system can adapt, the better, but, given the nature of enterprise IT systems, it seems hard, at least as of now, to imagine all adaptations being automatic. Adaptation under human supervision seems a more pragmatic solution. Investigations on the role of software engineering for self-adaptive systems (Cheng et al., 2009); (Lemos et al., 2011) have emerged in the recent past. These investigations reveal two broad lines of attack: one applying control-theoretic ideas of model reference / mode identification adaptive control (Brun et al., 2009) and the other applying adaptation techniques from biology (Brun, 2008). Both have key dependence on the ability to sense changes in the environment. To summarize, some of the key questions that should be investigated to model enterprise adaptation are: What are the dimensions of adaptation with respect to functional or non-functional requirements? What are the adaptation architectures for business applications, business processes and the context (e.g., Goal/Decision/Component based) (Sykes et al., 2008)? How to design MAPE-K (Jacob, 2004) feedback loop for Business, IT and Infrastructure planes? How to determine the ideal adaptive controller (i.e., control theoretic, biological or hybrid) that is best suited for a typical business need? Can the required *Sensors* interface be fully realized using underlying middleware and operating system level sensors augmented with instrumentation of IT systems?

4.4 Open Issues and Possible Solution Approaches

With regards to enterprise modelling, a key open issue is to come up with a set of models for the enterprise that are amenable to rigorous analysis and simulation. Assuming, a Graph (or network) adequately models the structural aspect of the enterprise, the behavioural aspect can be modelled using an event paradigm wherein nodes, as producer and/or consumers of events, participate in publisher-subscriber protocol. Exchange of information within the nodes can be modelled as side-effects on a 'global' context and a variety of data can be obtained through instrumentation of enterprise system model. This leads to several interesting questions: Can first-cut model be automatically derived from this data? Can the desired analysis be expressed as a set of properties, structural or behavioural or both, of the graph? Can impact of graph perturbations on a given property be computed? Can the list of graph perturbations necessary to bring a property within acceptable value range be identified / computed?

Several results for networks with known topology seem useful: A property for the whole network can be computed / optimized (Nagurney, 2011); effect of perturbations such as deletion of a node and/or link on global properties can be computed (Nagurney, 2012); analysis support for 'network of networks' is claimed (Nagurney, 2012). However, this work needs to be built further along multiple dimensions leading to a set of questions: How to obtain the network topology in a largely automated manner? Can techniques pertaining to random graph model (Erdős and Rényi, 1959); (Barabási and Albert, 1999) suffice in arriving at first-cut topology that can be further refined by subject matter experts? Do agent-based ideas (Maes, 1990) help in devising an action plan so as to bring the network back to the desired range of a given global property after perturbation? How to simulate 'what-if' and 'if-what' business scenarios? Can belief propagation (Kim et al., 1983) help? How to translate inferences from analysis and simulation into an action plan for the enterprise IT systems?

With regards to Enterprise Adaptation, introducing MAPE-K architecture (Jacob, 2004) across the IT systems plane seems to be a good starting point. Presuming suitable sensors are in place, it boils down to coming up with a way to specify adaptation rules and mechanisms to effect application adaptation. Event-Condition-Action paradigm seems adequate for specifying adaptation rules, but a key challenge is - how to ensure adaptations are semantically correct i.e. intent-preserving. Moreover, adaptation mechanism should have component nature so that it is possible to decompose application into components and connectors both of which can be adapted independently or in concert. Making the abstraction first-class will help adaptation at any desired level of granularity. There exists reasonable handle on structural aspects of component and connector, but, more work is required for addressing the behavioral aspects. Plug-n-play architecture to enable open extensibility is another topic of investigation. As software processes are also software, application adaptation techniques are applicable to business processes as well (Osterweil, 1987). Therefore, the ability to support adaptations at application as well as business process levels seems critical for developing dynamic business platforms (SOA, 2008).

5 CONCLUSIONS

In the past, we have embarked upon a model-driven approach and the necessary tooling infrastructure for development of database-centric business applications. Our MDE endeavour has led to several benefits such as higher productivity, uniformly high code quality (i.e., best practices without developer dependence) and easy retargeting to multiple technology platforms. At present with increased globalization and variable business dynamics, SPLE helped us to create custom solutions for enterprises using the notion of 'variability' – i.e., what changes *where* and *when* in system functionality. However, with highly uncertain and demanding economic conditions in the future, enterprises would be encouraged to investigate the concept behind *modelling an enterprise* with a goal to analyse, predict, simulate and adapt an enterprise on demand. This paper summarized the role of modelling with respect to enterprises looking back at our experiences in the past to the immediate challenges and needs of the future.

REFERENCES

Barabási, A-L., Albert, R., 1999. Emergence of scaling in random networks. *Science, American Association for the Advancement of Science, 286*, pp: 509-512.

Barat, S., Kulkarni, V., 2011. A component abstraction for business processes. *Business Process Management*

Workshops 2011, pp: 301-313.

Beyond SOA: A new type of framework for dynamic business applications - Part II, 2008 http://www.infoq.com/articles/beyond-soa-dba-part-2

Brun, Y., 2008. Building biologically-inspired self-adapting systems - extended abstract. *Software Engineering for Self-Adaptive Systems*, Springer-Verlag, 2008.

Brun, Y., et al., 2009. Engineering self-adaptive systems through feedback loops. *Software Engineering for Self-Adaptive Systems*, Springer-Verlag, pp: 48-70.

Cheng, B., et al., 2009. Software engineering for self-adaptive systems: a research roadmap. *Software Engineering for Self-Adaptive Systems*, Springer-Verlag, pp: 1-26.

Erdős, P., Rényi, A., 1959. On random graphs. *Publicationes Mathematicae, Volume 6*, pp: 290-297.

France, R., Rumpe, B., 2007. Model-driven development of complex software: A Research Roadmap. *FOSE'07*, pp: 37-54.

Hailpern, B., Tarr, P., 2006. Model-driven development: the good, the bad, and the ugly. *IBM Systems Journal*, Volume 45 Issue 3, July 2006, pp: 451-461.

Hutchinson, J., Whittle, J., Rouncefield, M., Kristoffersen, S., 2011. Empirical assessment of MDE in industry. *ICSE 2011*, pp: 471-480.

IEEE Standard 1471-2000: IEEE Recommended practice for architectural description of software-intensive systems.

Jacob, B., 2004. A Practical guide to the IBM autonomic computing toolkit. www.redbooks.ibm.com/redbooks/pdfs/sg246635.pdf

Kang, K., Kohen, S., Hess, J., Novak, W., Peterson, A. 1990. Feature-orientation domain analysis feasibility study. *Technical Report*, CMU/SEI-90TR-21, November 1990.

Kim, J. H., Pearl, J. Bundy, A. *(Ed.),* 1983**.** A computational model for combined causal and diagnostic reasoning in inference systems. *IJCAI, pp:* 190-193.

Kulkarni, V., Reddy, S., 2003. Separation of concerns in model-driven development. *IEEE Software 20(5)*, pp: 64-69.

Kulkarni, V., Reddy, S., 2008. A model-driven approach for developing business applications: experience, lessons learnt and a way forward, *ISEC,* pp: 21-28.

Kulkarni, V., Reddy, S., 2008. An abstraction for reusable MDD components: Model-based generation of model-based code generators, *GPCE*, pp: 181-184.

Kulkarni, V., Barat, S., 2010. Business process families using model-driven techniques, *Business Process Management Workshops* 2010, pp: 314-325.

Kulkarni, V., 2010. Raising family is a good practice. *FOSD 2010*, pp: 72-79.

Kulkarni, V., Barat, S., Roychoudhury, S., Sunkle, S., 2012. Model driven development – where to from here, *ISEC 2012* workshops

Kulkarni, V., Barat, S., Roychoudhury, S., 2012. Towards Business Application Product Lines, *MoDELS 2012* pp: 285-301

Lemos, R., et al., 2011. Software engineering for self-adaptive systems: a 2nd research roadmap. *Dagstuhl Seminar*, http://drops.dagstuhl.de/opus/volltexte/2011/3156

Maes, P., 1990. Situated agents can have goals *Robotics and autonomous systems*. 6, pp: 49 – 70.

Medvidovic, N., Egyed, A., Rosenblum, D., 1999. Round-Trip Software Engineering Using UML:From Architecture to Design and Back, *2nd Workshop on object-oriented reengineering*, Sep'99, pp: 1-8.

Nagurney, A., 2011. Supernetworks: The science of complexity. *Journal of University of Shanghai for Science and Technology 33: (2011)*, pp: 205-228.

Nagurney, A., 2012. Supply chains and transportation networks. *Prepared for the Handbook of Regional Science,* 2012.

Naur, P., 1985. Programming as theory building. Microprocessing and Microprogramming, 15(5), pp: 253 – 261.

OCL Object Constraint Language, 2012. http://www.omg.org/spec/OCL/2.3.1/PDF

MOFM2T MOF Model to Text Transformation, 2008. http://www.omg.org/spec/MOFM2T/1.0/PDF

QVT Query/View/Transformation, 2011. http://www.omg.org/spec/QVT/1.1/PDF/

Osterweil, L., 1987. Software processes are software too. *ICSE '87*, pp: 2-13

Rolland, C., Loucopoulos, P., Kavakli, V., Nurcan, S., 1999. Intention based modelling of organizational change: an experience report, EMMSAD'99.

Rumpe, B., 2004. Executable modeling with UML - a vision or a nightmare?, www.se-rwth.de/~rumpe/publications/ps/IRMA.UML.pdf

Sykes, D., Heaven, W., Magee, J., Kramer, J., 2008. From goals to components: a combined approach to self-management. *SEAMS 2008,* pp: 1-8.

Sunkle, S., 2011. First-class features. *PhD thesis*. Otto von Guericke University Magdeburg.

Truex, D., Baskerville, R., Klein, H., 1999. Growing systems in emergent organizations. *Communications of the ACM*, Volume 42 Issue 8, Aug. 1999, pp: 117-123.

Wagter, R., Proper E., Witte, D. A practice-based framework for enterprise coherence. *PRET*, 2012, pp: 77-95.