Electronic Notes in Theoretical Computer Science 82 No. 6 (2003)
URL: http://www.elsevier.nl/locate/entcs/volume82.html 11 pages

Model Driven Performance Analysis of Enterprise Information Systems [1]

James Skene [2] and Wolfgang Emmerich [3]

Department of Computer Science
University College London
London, United Kingdom

Abstract

This paper describes the particular motivation for performance analysis in the domain of Enterprise Information Systems (EISs) and argues that the Model Driven Architecture (MDA) is a suitable framework for integrating formal analysis techniques with engineering methods appropriate to the domain. The MDA permits natural and economical modelling of design and analysis domains and the relationships between them, supporting both manual and automatic analysis. It incorporates the Unified Modelling Language (UML), which is extensively used to capture system designs. We present our general modelling approach and outline its use in relating models of Enterprise Java Bean (EJB) applications, annotated using standard profiles, to analysable formal models.

1 Introduction

Distributed component technologies support the development of distributed applications by combining a component model with a middleware technology. Components are reusable and composable units of software. They export functionality to their environment and may also import functionality from their environment using well defined interfaces. Distributed systems are typically required to possess properties of openness, scalability and heterogeneity. Middleware supports distribution by providing transparency for numerous implementation concerns [7]: Most importantly, local and remote communications are treated uniformly by encapsulating communications using interfaces. Distributed systems are more complicated to develop than non-distributed systems, and middleware often provides additional support in

[1] This research is funded by the EU through grant IST-2001-34069 (Trusted and QoS-Aware Provision of Application Services).
[2] Email: `j.skene@cs.ucl.ac.uk`
[3] Email: `w.emmerich@cs.ucl.ac.uk`

the form of standard services, for example transaction and persistence services. These are also accessed via standard interfaces, and in some cases the middleware requires the business logic to reside in a specialised environment, often called a transaction monitor. When the business logic is encapsulated in the form of components this is a distributed component architecture.

Standardised distributed component architectures are necessary to permit the reuse of components in different contexts, and interoperability between components developed by different organisations. The standards specify the form that the components must take and the services that they can expect from their environment. Several standards have been developed, the most prominent being Enterprise Java Beans (EJBs) [26], the CORBA Component Model (CCM) [19] and .NET [14]. These technologies are widely used in the deployment of enterprise information systems. Markets have been created both for components that can be redeployed within a purchasing organisation and for remote services that can be integrated into software using the location transparency properties of the middleware. This latter business model is called Application Service Provision (ASP) and is suitable in situations where the service provided is complex, centralised or benefits from economies of scale. For example, existing application services provide credit ratings, payment systems and fulfilment.

The Model Driven Architecture (MDA) was conceived to address the problem of architectural tie-in in software development, particularly when developing software that relies on middleware. At the core of the MDA is the Unified Modelling Language (UML), a graphical language widely employed in software engineering. The MDA advocates the separation of Platform Independent Models (PIMs), representing the business logic of an application, from Platform Specific Models (PSMs) representing the deployment of that business logic using a particular technology. Software development proceeds as a process of refinement from PIMs to PSMs, each specified using the UML, appropriately extended with domain concepts. UML is already used to design enterprise information systems and increasingly organisations will use the MDA approach to insulate themselves from change in the competitive distributed component architecture marketplace.

Enterprise information systems render the issue of performance analysis both important and difficult. The performance of a system may be characterised by the response time and throughput of its use-cases, conditioned by the workload that the system is experiencing. A system is said to be scalable if its performance characteristics vary gracefully under increasing workloads. If application service marketplaces are to be viable on a large scale, providers and their clients will need to enter into Service Level Agreements (SLAs), whose terms will explicitly describe their mutual responsibilities with respect to performance and workload. Both parties will need to reason about the implications of entering into such an agreement with respect to the overall performance of a composite application and the capacity of the provider to accommodate multiple clients. Such reasoning can only be provided in competition with the engineering benefits provided by distributed component technologies. In order to accurately predict performance we must know how components are deployed, the manner in which data is persisted, threading and component activation policies, and other concerns relegated to the middleware that normally need not be represented in the design of a distributed component application.

In this paper we propose that the MDA is a suitable framework for the integration of a variety of analysis techniques into the process of developing distributed applications. The meta-modelling facilities of the UML permit us to express the design and analysis domains naturally, using the concepts inherent to these domains. Crucially, this allows us to incorporate domain knowledge relating to the behaviour of distributed component architectures without imposing drastically on the normal modelling requirements of a software development project. The use of meta-model mappings between design and analysis domains allows us to describe the construction of valid analytic models in the context of a design. The mapping rules depend on the source and target domains, but are expressed seperately, permitting a flexible association between design domains and analysis methods. UML is supported by tools, allowing us to assist or automate the process of analysis. In this paper we elaborate our approach to incorporating analysis techniques into the MDA and defining mappings from design domains. We show in outline how this can be applied to response-time prediction for EJB applications, by defining design domain concepts, a domain of queuing network models and a mapping between them. This latter effort is a work in progress: We are not yet in possession of all of the rules that apply when mapping from an EJB design to a valid performance model. It is hard to determine valid analytic models for architectural components such as application containers and databases due to the opacity of their implementation and their lack of instrumentation. Section 5 discusses future work we mean to undertake in this area.

The remainder of this paper provides: In section 2, a survey of related work; in section 3, a description of our modelling approach; in section 4, an example of our approach applied to response time prediction of EJB applications; and in section 5 a summary of our contribution and future direction.

2 Related Work

This paper seeks in part to address the issues identified by Pooley in [25]. It is clearly related to other efforts to derive performance models from UML diagrams, of which there are many [24,23,8,5,11,4]. In our case-study use of the Profile for Schedulability, Performance and Time Specification we resemble [22]. Additional surveys of the area are provided in [3,2]. We differ from these approaches in our insistence that the analysis models should also be specified in UML, using profiles provided by the lightweight extension mechanism.

[8] suggests that UML could represent performance models, but relegates its duties to communications between software and performance engineers. We believe that our approach is complementary to most cited here as it can represent and integrate the many useful derivations hitherto proposed, and indeed other analysis methods taking UML diagrams as their source [10,6].

In [12] an analytical model of an EJB server is presented, but not identified with a specification method. [15] proposes a framework approach to modelling and monitoring EJB performance but does not elaborate a modelling method.

Our approach to meta-modelling using the UML lightweight extension mechanism is consistent with the official MDA white paper [17], but also resembles the approach of the precise UML group in their UML 2.0 infrastructure proposal [1]. Future versions of the MDA specifications may advocate stronger meta-modelling approaches (heavyweight extension mechanisms) whereby new semantic domains, such those used by our analysis models, are represented by seperate languages defined at the same meta-level as the UML using the Meta Object Facility (MOF) [20]. The MOF model is a near subset of the UML, and uses OCL constraints in the same way, so our approach would require little adaption to accomodate such a change, and may in fact benefit from a clearer seperation of semantic and notational domains.

3 Mapping to Analysis Models

The UML is capable of representing all aspects of a software system and is widely used for development, making it a logical starting point for the analysis of software designs. It may be extended to refine its fundamental semantics allowing it to model complex domains naturally, using the concepts inherent to those domains. A particular semantic extension of the UML is called a 'profile'. Profiles can incorporate logical constraints governing the form of models in a particular domain. These constraints can represent relationships between models from different domains, which we call 'mappings': for example from a PIM to a PSM (a deployment relationship). Checking this mapping would ensure that the implementation has the properties demanded by the design. We propose that analytical modelling can be supported by defining a mapping from a design domain (PIM/PSM) to a domain representing an analysis formalism. Each domain and the mapping between them are represented by a UML profile. This is unconventional as it introduces analysus models that are not clearly either PIMs or PSMs. Our approach is as follows:

(i) Identify the design domain and the profiles employed to describe it.

(ii) Identify those qualities that are of interest but require formal analysis to determine. Choose an appropriate analysis technique, for example, queuing networks for performance analysis and define a profile to represent the entities within the analysis domain (e.g. queues and workloads).

(iii) Define a mapping between the design domain and the analysis domain that correctly represents the semantics of each. Express it using logical constraints captured in a mapping profile.

(iv) Automate analysis of the analysis models.

(v) Automate the mapping, to the extent possible. Mappings are not generally adequate to permit automation. PSMs, for example, are expected to contain more information (relating to the platform) than can be derived from a PIM from which it maps. Mappings merely constrain the form of a model with respect to another.

It would also be possible to describe the domain of analysis results and a mapping back to design models. This would permit the results of analysis to be reintegrated into the design models and displayed in context.

To permit a consistent approach to modelling across organisations the OMG, standardises UML profiles for particular domains. The Profile for Schedulability, Performance and Time Specification [21] (henceforth the 'performance profile') permits the description of performance requirements and demands in the context of an abstract resource model. The specification suggests that analysis tools directly inspect UML diagrams annotated according to this profile in order to derive performance models and consequently predictions (figure 1a). We differ in our belief that by considering the mapping to an analysis model separately from the solution of the analysis model (figure 1b) and by expressing the mapping and the domain of the analysis model using the same modelling techniques employed in the design model, the following benefits can be realised:

(i) Analysis techniques can be flexibly applied to design domains by defining new mappings. This is important because new design domains will often emerge (for example as new implementation platforms are developed) and they will be expressed using new profiles. This is both natural and necessary. It is not feasible to represent all of the details of an application container, for example, using only the domain concepts available in the performance profile.

(ii) The semantic validity of analytic models can be checked against the design using existing model checking tools in the same way that PSMs may be checked against PIMs. Automatic derivation of analytical models may be possible in some cases.

(iii) Automatic derivation of analytical models is generally difficult. Human intuition may be required to construct meaningful models. Models are often infeasible, for example, because the resulting model has too large a state space to permit analysis. If the analysis model were exposed in the design tool then the user could intervene when problems occur, increasing the feasibility of performing correct analyses.

Our approach does not deny or reduce the benefits of standardisation of domain profiles such as the performance profile. It does acknowledge that not

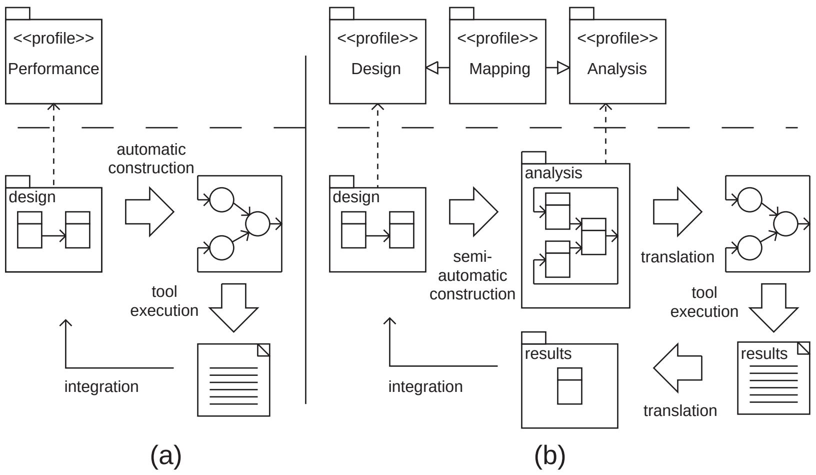

Fig. 1. (a) Performance profile approach to integrating analysis. (b) Our approach

all information pertinent to analysis can be naturally modelled using standard profiles as it is impossible to completely anticipate what design or analysis domains will be required. Uniformity of expression should nevertheless be preserved by reusing, combining and specialising standard profiles wherever possible.

The next section provides an example of this approach applied to response-time prediction of EJB applications.

4 Performance Analysis of Distributed Component Systems

We now provide an example approach to predicting the response-time of remote method invocations against EJBs, structured according to the approach described in the last section. Response time is defined to be the period between synchronous method invocation and the signalling of method completion to the client.

The design domain is modelled using the UML Profile for EJB [9] (henceforth 'EJB profile') to specify the architecture of the application, and the performance profile to specify resource demands for method invocations, and deployment relationships. The modelling prescription is broadly as recommended in the performance profile:

(i) A deployment diagram shows the relationships between application components, archetecture components (such as containers and database), infrastruture (CPUs, network links) and the application clients.

(ii) A use-case diagram describes the workload. The links between client entities and the use-cases are annotated with arrival rates (an open workload [13]).

(iii) For each use-case, an exemplary instance level sequence diagram shows the client view of the EJB application.

(iv) For each session bean business method, an exemplary instance-level sequence diagram shows the actions taken by the method, annotated with their expected delays and resource utilisations. The joint profile constrains these to be reasonable. For example, create methods invoked on the home interface of an entity bean are required to have a database utilisation demand.

(v) A component diagram indicates the packaging of classes into an application component.

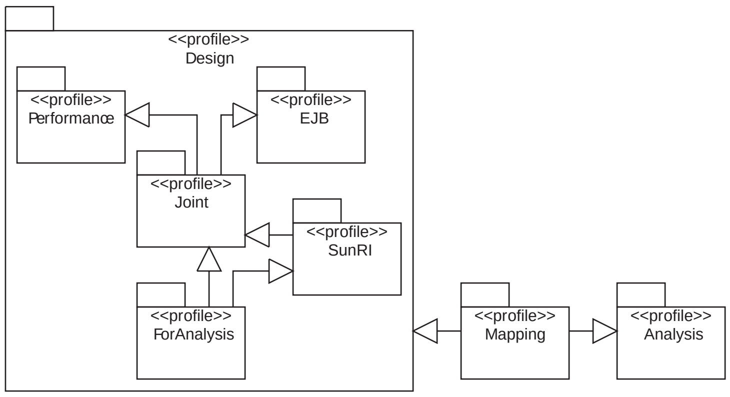

Fig. 2. Profiles used to model EJB applications

Figure 2 shows the profile packages used, including five design domain packages. The performance profile and the EJB profile provide the basic stereotypes. We combine these profiles in a joint profile containing constraints that ensure that performance models contain sufficient information to permit analysis. This profile also contains elements necessary for analysis but not present in either of the standards, such as stereotypes used to identify the container, naming service and database components in a deployment diagram. The EJB profile only supports the design of applications, not particular deployments, so these elements are omitted. Moreover, defining these elements would over-prescribe the implementation of the EJB standard. In the context of a performance analysis these elements must be represented, so we took a pragmatic approach and included them in our extended profile without suggesting that they have been unreasonably omitted from the standard.

The remaining domain profile packages were separated from the joint profile for reasons of convenience. The reference implementation package contains an inherited stereotype allowing us to identify the application server as being the Java 2 reference implementation, and therefore specify application specific properties relevant to performance, such as the memory allocated to the session bean pool, which potentially effects the latency of session bean creation. The simple analysis package contains constraints on the design domain only pertinent to our particular analysis method, such as the constraint that all workload classes specify an arrival rate. Such a constraint would not be reasonable if closed population queuing network models were being used for analysis.

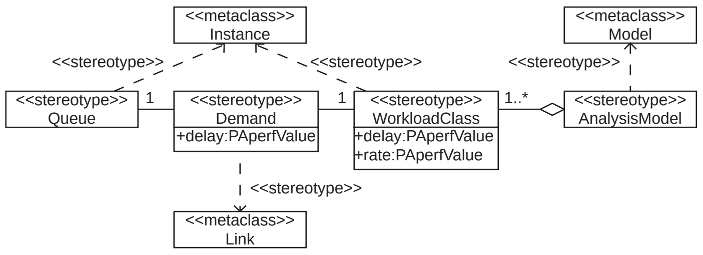

Fig. 3. A simple analysis domain profile

Figure 3 shows a view of the simple analysis domain, and figure 4 shows a fragment of the mapping domain, indicating that the number of workload classes in an analysis model should match the number of use-cases in a performance context. A comprehensive treatment of the profiles is beyond the scope of this paper. Moreover, the mapping profile is as yet inadequately specified, as discussed in the introduction and conclusion sections. In future work we will resolve this and address the automation of the mapping and analysis stages.

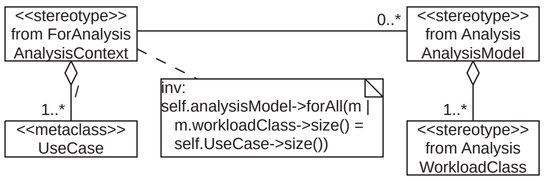
<<stereotype>> from ForAnalysis AnalysisContext
0..*
<<stereotype>> from Analysis AnalysisModel
1..*
<<metaclass>> UseCase
inv:
self.analysisModel->forAll(m |
m.workloadClass->size() =
self.UseCase->size())
1..*
<<stereotype>> from Analysis WorkloadClass

Fig. 4. A constraint from the mapping profile

In the development of our design profiles we encountered the following issues with the standard profiles:

(i) The EJB profile is out of date, covering only EJB 1.1. To represent our application we were forced to extend the profile to differentiate between local and remote interfaces.

(ii) The EJB profile uses UML version 1.3 [16]. The redefinition of components in version 1.4 [18] requires the representation of EJB packaging to be modified to use artifacts rather than components.

(iii) Both profiles specify their constraints using natural language. Using OCL in the manner of the UML specification would have been preferable.

(iv) The performance profile does not impose any constraints on the deployment models that it specifies. Several reasonable constraints could be included without limiting the applicability of the profile. For example, it is not reasonable for an instance to deploy itself, and all action execution with resource demands should correspond to entities deployed in a context where those resources are available.

(v) As noted in [22] the performance profile does not readily permit the expression of workloads containing multiple scenarios (use-cases).

5 Conclusions and Future Work

In this paper we observe that the UML exists within the wider context of the MDA and that the meta-modelling facilities provided by the MDA can be employed in at least two useful ways. Firstly, they provide a standard means to describe formal models and their mappings from development models. This captures the semantics of the analysis technique and could be used to validate analysis models or automate their production. If used in the context of a CASE tool, such mappings could be customised and the analysis models inspected and altered if necessary. This mitigates the extreme difficulty involved in automatically producing feasible and valid analysis models.

Secondly, the use of meta-modelling techniques would allow a CASE tool to incorporate new analysis techniques with mappings from design domains with non-standard semantics. This would accommodate the wide variety of performance analysis techniques and real world situations to which they may be applied. In this paper we have provided the example of enterprise information systems and discussed the economic benefits of applying performance analysis techniques. Using only techniques standard to the MDA we described the design domain of EJB applications, including its extended semantics, and described a mapping to a family of analysis models.

We have presented an overview of our modelling approach applied to the performance analysis of EJB applications, and presented our experience in adapting standard profiles to describe the design domain. We have illustrated the use of UML to represent an analysis formalism by presenting a profile for queuing networks. Immediate future challenges include determining precisely what constitutes a valid analysis model for our EJB designs, and incorporating this information into our mapping profile. If we persist in using queuing network models to analyse our designs then this will entail discovering and characterising resources and demands hidden in architectural components such as the application container and database. We also wish to enhance the level of automation for mappings. We intend to evaluate the practicality of our approach using real-life case studies, including a large e-science project and a distributed auction system, both of realistic complexity and under development within our department. [4]

[4] Thanks to Davide Lamanna, Genaína Nunes Rodrigues and Joanne Hacking for their assistance with this document.

References

[1] A. Clark, S. K., A. Evans, "Unambiguous UML (2U) Revised Submission to UML Infrastructure RFP," Object Management Group (2002), `http://cgi.omg.org/cgi-bin/doc?ad/02-06-14.pdf`.

[2] Balsamo, S. and M. Simeoni, *Deriving performance models from software architecture specifications*, in: *Proceedings of the European Simulation Multiconference 2001* (2001).

[3] Balsamo, S. and M. Simeoni, *On transforming UML models into performance models*, in: *Workshop on Transformations in UML*, 2001.

[4] Bernardi, S., S. Donatelli and J. Merseguer, *From UML sequence diagrams and statecharts to analysable petri net models*, in: *Proceedings of the Third International Workshop on Software and Performance* (2002), pp. 35–45.

[5] Cortellessa, V. and R. Mirandola, *Deriving a queueing network based performance model from UML diagrams*, in: *Proceedings of the Second International Workshop on Software and Performance* (2000), pp. 58–70.

[6] Cortellessa, V., H. Singh, B. Cukic, E. Gunel and V. Bharadwaj, *Early reliability assessment of UML based software models*, in: *Proceedings of the Third International Workshop on Software and Performance* (2002), pp. 91–92.

[7] Emmerich, W., "Engineering Distributed Objects," John Wiley and Sons, Ltd., 2000.

[8] Hoeben, F., *Using UML models for performance calculation*, in: *Proceedings of the Second International Workshop on Software and Performance* (2000), pp. 77–82.

[9] Java Community Process, "UML/EJB Mapping," (2001), `http://www.jcp.org/jsr/detail/26.jsp`.

[10] Kaveh, N. and W. Emmerich, *Deadlock detection in distributed object systems*, in: *Joint 8th European Software Engineering Conference (ESEC) and 9th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE-9)* (2001), pp. 44–51.

[11] Lindemann, C., A. Thümmler, A. Klemm, M. Lohmann and O. Waldhorst, *Performance analysis of time-enhanced UML diagrams based on stochastic processes*, in: *Proceedings of the Third International Workshop on Software and Performance* (2002), pp. 77–82.

[12] Lladó, C. and P. Harrison, *Performance evaluation of an enterprise javabean server implementation*, in: *Proceedings of the Second International Workshop on Software and Performance* (2000), pp. 180–188.

[13] Menascé, D. and V. Almeida, "Capacity Planning for Web Services: Metrics, Models and Methods," Prentice Hall PTR, 2002.

[14] Microsoft Corporation, ".NET Framework," http://www.microsoft.com/net/.

[15] Mos, A. and J. Murphy, *A framework for performance monitoring, modelling and prediction of component oriented distributed systems*, in: *Proceedings of the Third International Workshop on Software and Performance* (2002), pp. 235–236.

[16] Object Management Group, "Unified Modelling Language (UML), version 1.3," (2000), http://www.omg.org/cgi-bin/doc?formal/00-03-01.pdf.

[17] Object Management Group, "Model Driven Architecture (MDA)," (2001), http://cgi.omg.org/docs/ormsc/01-07-01.pdf.

[18] Object Management Group, "Unified Modelling Language (UML), version 1.4," (2001), http://www.omg.org/cgi-bin/doc?formal/01-09-67.pdf.

[19] Object Management Group, "CORBA Component Model, v3.0," (2002), http://cgi.omg.org/docs/formal/02-06-65.pdf.

[20] Object Management Group, "Meta Object Facility (MOF), version 1.4," (2002), http://www.omg.org/cgi-bin/doc?formal/02-04-03.pdf.

[21] Object Management Group, "UML Profile for Schedulability, Performance, and Time Specification," (2002), http://www.omg.org/cgi-bin/doc?ptc/02-03-02.pdf.

[22] Petriu, D. and H. Shen, *Applying the UML performance profile: Graph grammar-based derivation of LQN models from UML specifications*, in: *Performance TOOLS 2002*, Lecture Notes in Computer Science **2324** (2002), pp. 159–179.

[23] Pooley, R., *Using UML to derive stochastic petri net models*, in: *Proceedings of the fifteenth UK Performance Engineering Workshop (UKPEW)*, 1999, pp. 45–56.

[24] Pooley, R., *Using UML to derive stochastic process algebra models*, in: *Proceedings of the fifteenth UK Performance Engineering Workshop (UKPEW)*, 1999, pp. 23–34.

[25] Pooley, R., *Software engineering and performance: A road-map*, in: *Future of Software Engineering* (2000), pp. 189–200.

[26] Sun Microsystems, "Enterprise JavaBeans 2.0 Specification Final Release," (2001), http://java.sun.com/j2ee/docs.html.