Domain Analysis of Enterprise Service-Oriented Distributed System Computational Attributes using System Execution Modeling Tools

James H. Hill
Indiana U. / Purdue U. at Indianapolis
Indianapolis, IN, USA
hillj@cs.iupui.edu

Douglas C. Schmidt
Vanderbilt University
Nashville, TN, USA
schmidt@dre.vanderbilt.edu

I. Introduction

Challenges of enterprise distributed system development. Enterprise distributed systems (such as air traffic management systems, cloud computing centers, and shipboard computing environments) are steadily increasing in size (*e.g.*, lines of source code and number of hosts in the target environment) and complexity (*e.g.*, application scenarios). To address challenges associated with developing next-generation enterprise distributed systems, service-oriented middleware [Pezzini:07] is becoming the *de facto* standard for developing such systems. Service-oriented middleware increases the level of abstraction for software development so distributed system developers focus on the system's "business-logic" instead of wrestling with low-level implementation details, such as development and configuration, resource management, fault tolerance. Moreover, service-oriented middleware promotes reuse of the system's "business-logic" across different application domains, which helps reduce (re)invention of core intellectual property.

Although service-oriented middleware is improving the software lifecycle of enterprise distributed systems, domain analysis of computational attributes (*e.g.*, threading and lock synchronization) that affect system quality-of-service (QoS) properties (*e.g.*, latency, throughput, and scalability) are not evaluated until late in the software lifecycle, *i.e.*, during system integration time. This is due in part to the *serialized-phasing development* [Rittel:73] problem where the infrastructure- and application-level system entities, such as components that encapsulate common services, are developed during different phases of the software lifecycle. As illustrated in Figure 1, distributed system developers do not realize their system does not meet its QoS requirements until its too late, *i.e.*, during complete system integration time.

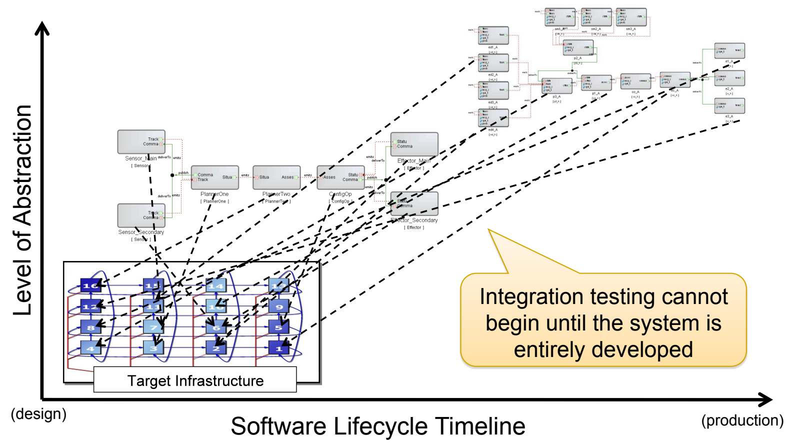
Level of Abstraction
Integration testing cannot begin until the system is entirely developed
Target Infrastructure
(design)
Software Lifecycle Timeline
(production)

Figure 1. Overview of Serialized-phasing Development in Enterprise Distributed Systems

To overcome the effects of serialized-phasing development, *software performance engineering* (SPE) [Smith:01] is as well-known domain analysis technique for estimating system performance during early phases of the software lifecycle. Enterprise distributed system developers construct high-quality models, such as UML diagrams augmented with performance characteristics, based on domain analysis of the system's computational needs, and then leverage traditional analytical techniques, such as queuing networks [Menasce:04], to evaluate constructed models and predict system performance (*e.g.*, end-to-end response time). This methodology enables enterprise distributed system developers to identify potential performance bottlenecks before they become too costly to locate and rectify. The amount of expertise required to construct such models for enterprise service-oriented distributed systems, however, lies outside domain knowledge of most enterprise distributed system developers, who therefore spend much time and effort constructing low-level performance models, such as queuing network models, rather than focusing more on modeling the necessary computational attributes for performance evaluation during early phases of the software lifecycle.

Solution approach → System execution modeling tools. To address problems associated with constructing high-quality models of service-oriented enterprise distributed systems for domain of performance analysis, there is a need for model-driven engineering methodologies that simplify the following:

1. **Modeling system behavior and workload** in terms of computational workloads, resource utilizations and requirements, and network communication. This step can be accomplished quickly and precisely using domain-specific modeling languages (DSMLs) [Ledeczi:99] that use high-levels of abstraction and enable distributed system developers to remain their knowledge domain.
2. **Evaluation of behavior and workload models** using existing SPE techniques, such as queuing network models. This step is accomplished using model transformations that convert high-level behavior and workload models into low-level detailed models, (*e.g.*, UML diagrams augmented with performance characteristics) which can then be evaluated using traditional techniques.
3. **Validation of high-quality performance models** using emulation techniques. This step is accomplished using high-level DSMLs (from Step 1) and model interpreters that transform constructed models into source code for the target architectures. Enterprise distributed system developers then compile and execute the auto-generated source code on their target architecture to validate performance predictions.

Figure 2 shows the relation between the steps above for using system execution modeling (SEM) tools to evaluate and validate high-quality models for early performance prediction.

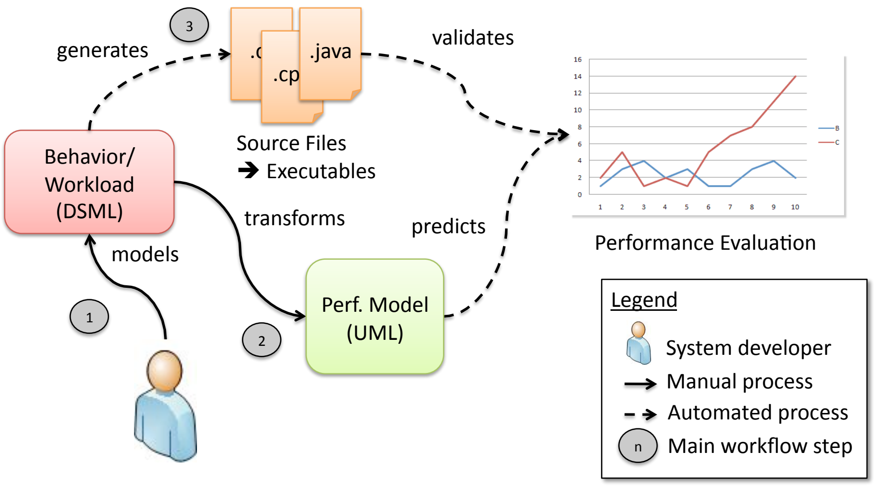

Figure 2. Overview of the Performance Prediction and Validation Process Workflow

The ability to accurately predict performance during early phases of the software lifecycle helps reduce cost and time rectifying performance bottlenecks during later phases, *i.e.,* system integration time. Likewise, validating performance predictions helps increase confidence levels that models used for domain analysis of performance accurately model the system under development. The ability to accurately predict performance, however, depends on the accuracy of the original model. As enterprise distributed systems grow larger and more complex, the ability to accurately construct such models is becoming harder for developers, especially during early phases of the software lifecycle when the system's design is constantly changing. SEM tools therefore will become increasingly important in reducing the complexity if domain analysis of computational attributes of enterprise service-oriented distributed systems for early predication and validation of performance.

II. Outline of the Remainder of the Chapter

The remainder of this chapter will be organized as follows:

Section 2 describes existing techniques for domain analysis of computation attributes for SPE, focusing on a survey of related work, including:

- **SPE techniques for distributed systems.** Liu et al. [Liu:05a] demonstrated that it is possible to construct traditional queuing network models for service-oriented middleware. Their approach focuses on predicting the performance of systems that communicate via their exposed services, which is analogous to a remote method invocation. Their approach predicts the performance of the system within 11% error of the actual performance. Liu et al. [Liu:05b] extended their previous efforts to constructing queuing network models for service-oriented middleware that communicate asynchronously using events. Using their technique for constructing a queuing network model, Liu et al. are able to predict performance within 10% error with a maximum error of 15%.

- **Modeling languages for evaluating performance.** KLAPER [Grassi:05] is a modeling language that specifies system behavior for service-oriented distributed systems. KLAPER allows developers to specify computational attributes, such as workload and resource utilization to enable performance and reliability analysis. RT-UML [Bertolino:04] is a UML profile for augmenting UML diagrams with computational attributes to enable performance evaluation of service-oriented distributed systems.

Section 3 describes a case study of a service-oriented enterprise distributed system that motivates the need for system execution modeling (SEM) tools for domain analysis of computational attributes and their effects on performance. This case study focuses on the *multistage workflow application* from the *QoS-Enable Dissemination (QED)* project [Loyall:09], which is a multi-team collaborative effort aimed at developing QoS-enabled middleware for the *Global Information Grid (GIG)* infrastructure and applications. Figure 3 shows QED in the context of the GIG infrastructure and its applications.

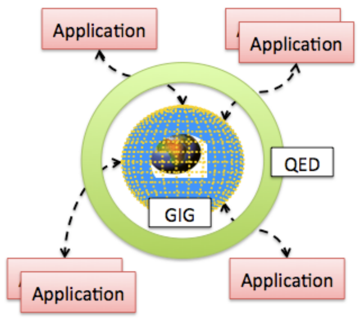

Figure 3. QED in the Context of the GIG Infrastructure and Its Applications

The aim of the to provide globally interconnected, end-to-end set of information capabilities for collecting, processing, storing, disseminating, and managing information on demand. The middleware for the GIG must therefore provide dependable and timely communication to applications and end-user scenarios that operate within dynamically changing conditions and environments, such as wireless ad-hoc networks and/or ban width and resource-constrained situations. To meet this need the QED middleware enhancements for the GIG provide timely delivery of information requested by users in mobile scenarios, tailoring and prioritizing information based on mission needs and importance, and operating in a manner that is robust to failures and intermittent communications.

Section 4 describes how we are applying system execution modeling tools (shown in Figure 2) in the QED project to enable domain analysis of computational attributes and their effects on performance using a tool we developed called CUTS. CUTS allows distributed system developers model behavior and workload (*i.e.,* its computational attributes) of distributed systems using high-level DSMLs. The high-level behavior and workload models are then converted to low-level performance models (such as queuing network models) for predicting performance. Likewise, the same models are transformed into source code for the target architecture to validate performance predictions.

This section shows how we used CUTS to model the multistage workflow application from the case study in Section 3. In addition, this section describes how CUTS yields several advantages compared with conventional domain analysis and SPE techniques. For example, CUTS enables distributed system developers to remain within their knowledge domain (i.e., at the application-level) when constructing high-quality models for domain analysis of the effects of computational attributes on performance.

Section 5 describes the design and results of experiments that use CUTS to evaluate the multistage workflow application and GIG's performance. This experiment is based upon work conducted in the QED project described in Section 3. The QED project provided a representative case study for evaluating CUTS ability to perform domain analysis on computational attributes because QED is aimed at improving QoS concerns of the GIG, and different computational attributes, such as those exhibited in the multistage workflow application, will have different effects on the GIG's performance. The experiments will therefore help QED developers identify aspects of the GIG where QED will improve overall quality of the GIG.

The experiments in this section also evaluate the following aspects of CUTS ability to performance domain analysis on computational attributes:

- Use high-level DSMLs to model different computational attributes of service-oriented distributed systems.
- Use model transformations to convert high-level models to low-level high-quality models for performance evaluation using traditional SPE techniques.
- Validate performance predictions of traditional SPE techniques by emulating the system on its target architecture.

Section 6 presents concluding remarks and lessons learned. Our experience applying CUTS to the QED project showed how SEM tools improve domain analysis of computational attributes and their effect on system performance. The following summarizes the benefits of applying CUTS to the QED project:

- CUTS allowed us to remain within our knowledge domain and capture computational attributes of complex service-oriented distributed systems for performance analysis.
- CUTS model transformations allowed use to leverage existing high-quality modeling and SPE technique for evaluating performance.
- CUTS emulation techniques allowed us to validate high-quality performance models and increase confidence levels in our domain analysis of computational attributes and their effects on performance.

CUTS is currently being transitioned to a production project for the Australian Defense Science and Technology Organization to perform domain analysis on computational attributes in heterogeneous architectures. An open-source version of the CUTS SEM tool described in this paper can be downloaded from www.dre.vanderbilt.edu/CUTS.

III. References

[Bertolino:04] Bertolino, A. and Mirandola, R., "Software Performance Engineering of Component-based Systems," *Proceedings of the 4th International Workshop on Software and Performance*, Jan 2004.

[Grassi:05] Grassi, V., Mirandola, R., and Sabetta, A., "From Design to Analysis Models: A Kernel Language for Performance and Reliability Analysis of Component-based Systems," *Fifth International Workshop on Software and Performance,* Palma de Mallorca, Spain, Jul 2005

[Ledeczi:99] Ledeczi, A., Maroti, M., Karsai G., and Nordstrom G., "Metaprogrammable Toolkit for Model-Integrated Computing*", Proceedings of the IEEE International Conference on the Engineering of Computer-Based Systems Conference (ECBS '99)*, 1999.

[Liu:05a] Yan Liu, Alan Fekete, and Ian Gorton. Design-Level Performance Prediction of Component-Based Applications. IEEE Transactions on Software Engineering, 31(11):928–941, 2005.

[Liu:05b] Yan Liu and Ian Gorton. Performance Prediction of J2EE Applications using Messaging Protocols. In Proceedings of the International SIGSOFT Symposium Component-Based Software Engineering (CBSE), May 2005.

[Loyall:09] J. Loyall, M. Carvalho, D. Schmidt, M. Gillen, A. M. III, L. Bunch, J. Edmondson, and D. Corman. QoS Enabled Dissemination of Managed Information Objects in a Publish-Subscribe-Query Information Broker. In Defense Transformation and Net-Centric Systems, April 2009.

[Menasce:04] Daniel A. Menasce, Lawrence W. Dowdy, and Virgilio A. F. Almeida. Performance by Design: Computer Capacity Planning By Example. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.

[Pezzini:07] M. Pezzini and Y. V. Natis. Trends in Platform Middleware: Disruption Is in Sight. www.gartner.com/DisplayDocument?doc_cd=152076, September 2007.

[Rittel:73] Rittel, H. and Webber, M. Dilemmas in a General Theory of Planning. Policy Sciences, pages 155–169, 1973.

[Smith:01] Connie Smith and Lloyd Williams. Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software. Addison-Wesley Professional, Boston, MA, USA, September 2001.

Author Biographies

Dr. James H. Hill is a Research Scientist at the Institute for Software Integrated Systems (ISIS), Vanderbilt University in Nashville, TN, and will be appointed Assistant Professor of Computer Science at Indiana University/Purdue University at Indianapolis (IUPUI) in Fall 2009. Dr. Hill's research focuses on algorithms, analytics, patterns, and modeling techniques, which have been realized in CUTS, to facilitate quality-of-service evaluation continuously throughout the software lifecycle. Dr. Hill has over six years experience in developing and evaluating large-scale distributed system performance and is the project leader for the CUTS system execution modeling tool.

Dr. Douglas C. Schmidt is a Professor of Computer Science and Associate Chair of the Computer Science and Engineering program at Vanderbilt University. He has published 9 books and over 400 papers that cover a range of topics, including patterns, optimization techniques, and empirical performance analyses of software frameworks and domain-specific modeling environments that facilitate the development of distributed real-time and embedded (DRE) middleware and applications. Dr. Schmidt has over fifteen years of experience leading the development of ACE, TAO, CIAO, and CoSMIC, which are open-source middleware frameworks and model-driven tools that implement patterns and product-line architectures for high-performance DRE systems.