

Modeling, Analyzing and Simulating Software Acquisition Process Architectures

James Choi	Walt Scacchi
Computer Science Dept.	Institute for Software Research
California State University	University of California
Fullerton, CA USA	Irvine, CA USA
sjchoi@ecs.fullerton.edu	wscacchi@ics.uci.edu

Submitted to *ProSim 2000*, London, UK, July 2000.

Extended Abstract

In this paper, we describe our efforts in the development of an environment that supports the modeling, analysis and simulation of processes associated with software system acquisition activities. Software acquisition is generally a large, multi-organization endeavor concerned with the funding, management, engineering, system integration, deployment and long-term support of large software systems. These systems are primarily intended for use by government agencies in defense, transportation, and other areas of public administration. The acquisition of large software systems, especially those for use in the military and commercial aviation (e.g., for air traffic control) take years of effort, often at a cost in the range of hundreds of million dollars, or more. Unfortunately, software acquisition processes are often unsuccessful or problematic in realizing their objectives and goals, in a timely and cost effective manner [GAO 1997]. Rectifying these problems through modeling, simulation and redesign of these processes is thus a motivation for research [Boehm and Scacchi 1996, Nissen, Snider, Lamb 1998, Scacchi and Boehm 1998].

As already noted, acquisition includes system engineering activities, which in turn include the processes we typically associate with the software engineering life cycle. Thus, the processes involved with software acquisition include those for software engineering, but also those involved in the use of complex information systems to fund, manage, integrate, deploy and support software systems before, during, and after their software engineering life cycle. The need to address engineering, inter-organizational and management processes together is what establishes our baseline of interest in modeling and simulating software acquisition processes.

Next, we choose to examine software acquisition processes from an architectural perspective. In this regard, our position is that modeling and simulating software acquisition processes will require some kind of *factoring*. This is needed to realize both a *separation of concerns* through a factorable architecture of interconnected and interrelated processes, as well as facilitating, guiding or managing the configuration or creative *composition of component processes* that together constitute software acquisition. Subsequently, in order to be able to construct and simulate factorable models

of software acquisition processes, we require *architectures* that can separate, compose and configure a web of software acquisition processes. Therefore, we will refer to this structured web as a software acquisition process architecture.

The focus of our research effort to be developed in a full paper is to describe our approach to modeling, analyzing and simulating software acquisition process architectures. This will include an identification and characterization of new issues and challenges that arise when we move from the modeling and simulation of conventional software processes to those that span the software acquisition life cycle [Boehm and Scacchi 1996, Scacchi and Boehm 1998]. It will also include addressing the design and prototyping of a loosely coupled Web-based environment, called SAWMAN, which supports the modeling and simulation of acquisition process architectures, as well as a variety of analysis, process prototyping, and process enactment capabilities. Beyond this, SAWMAN also includes tools and techniques for facilitating the capture, organization, update and evolution of heterogeneous sources of knowledge regarding scenarios of use, best practices, reference materials and the like for use by practitioners in the software acquisition community [ARO 1999, SA-CMM 2000, SPMN 1999, STSC 1996].

Modeling Software Process Architectures

Noll and Scacchi [1999b] have developed and demonstrated the design of a process modeling language (PML) and Web-based run-time environment. PML has been used to model legacy as-is, redesigned to-be, and transition here-to-there acquisition processes at the U.S. Office of Naval Research [Noll and Scacchi 1999b]. PML acts as an extensible process markup notation that can be compiled into an executable form to support process prototyping and process enactment across the Web, as well as serving as a person-in-the-loop process simulator [Scacchi 2000]. These process modeling capabilities enable multi-user process redesign across a distributed virtual enterprise or network of cooperating enterprises [Noll and Scacchi 1999a, Scacchi and Noll 1997].

Other researchers at the UC Irvine Institute for Software Research have been investigating new languages, tools and environments that focus attention on software system architectures [Medvidovic, Rosenblum and Taylor 1999, Medvidovic and Taylor 2000, Taylor, *et al.* 2000]. In our work, we chose to adopt the architectural design language (ADL) from this related research, and combine with PML in order to support the modeling of composable software process architectures.

Overall, our approach is to combine these two language notations into a single one that constitutes a new Process Architecture Design Language (PADL), as the basis for specifying models of software process architectures. Process architectures in turn enable us to configuration manage the concurrent composition of multi-version processes for software development or use in a manner that scales well beyond the limits of large process models [cf. Scacchi 1999]. It also enables the construction of "software process lines" or "process families" that can be (re)used across multiple software acquisition or development projects [cf. Bergey, Fisher, Jones 1999]. As such, PADL represents an important new class of process modeling technology that can be employed to specify

models of software acquisition process architectures. This approach and examples will be included in the full paper for the ProSim 2000 Workshop.

Analysis of Software Process Architectures

Complex processes that must span and interlink people working in geographically and temporally distributed enterprises are challenging to develop, verify and validate. In previous work, we described how individual process models could be analyzed for a number of semantic properties, such as determining the presence of "black holes" in process model specifications [Scacchi 2000]. In a related line of work, we have also focused attention to the problem of how to verify the consistency, completeness, traceability and correctness of configured software descriptions [Choi and Scacchi 1998]. In that work, we developed a formalization scheme that defines and provides proofs for a generic set of verification conditions that establish the internal correctness of a well-formed software product architecture. Well-formed implies each product component has an interface who inbound and outbound resources are consistently and completely mapped to interconnected components in a traceable manner. In our current effort, we have taken this formalization framework and adapted it for use in defining a set of generic correctness properties that can be assessed for a configured software process architecture. Further, we extend and apply this notion of internal correctness to a software process architecture modeled in a process architecture design language (PADL). We then use the domain of a software acquisition process architecture to help illustrate these concepts.

Simulation of Software Acquisition Process Architectures

In previous work, we have demonstrated and comparatively examined different approaches to the simulation of software processes [Scacchi 1999,2000]. This includes the introduction of software process simulators that enable interactive exploration (e.g., browsing, process walkthrough and inspection, process prototyping and "process surfing") of software processes [Scacchi 2000]. Given the approach to modeling and analyzing software process architectures we introduce in our current effort, we need to explain and demonstrate how software process simulation fits into our overall scheme.

To no surprise, we continue to employ and extend the process simulator techniques noted above, but now we apply them to the domain of software acquisition process architectures. As our software process architectures are configured and interlinked (i.e., "hyperlinked"), then their internal/external representation can be navigated as a process-oriented hypertext [Noll and Scacchi 1999a,b]. This capability provides a basis for providing Web-based process prototyping, simulator and enactment services. The following figure provides a quick glance of a single screen captured from a process simulator that traverses a software acquisition process architecture.

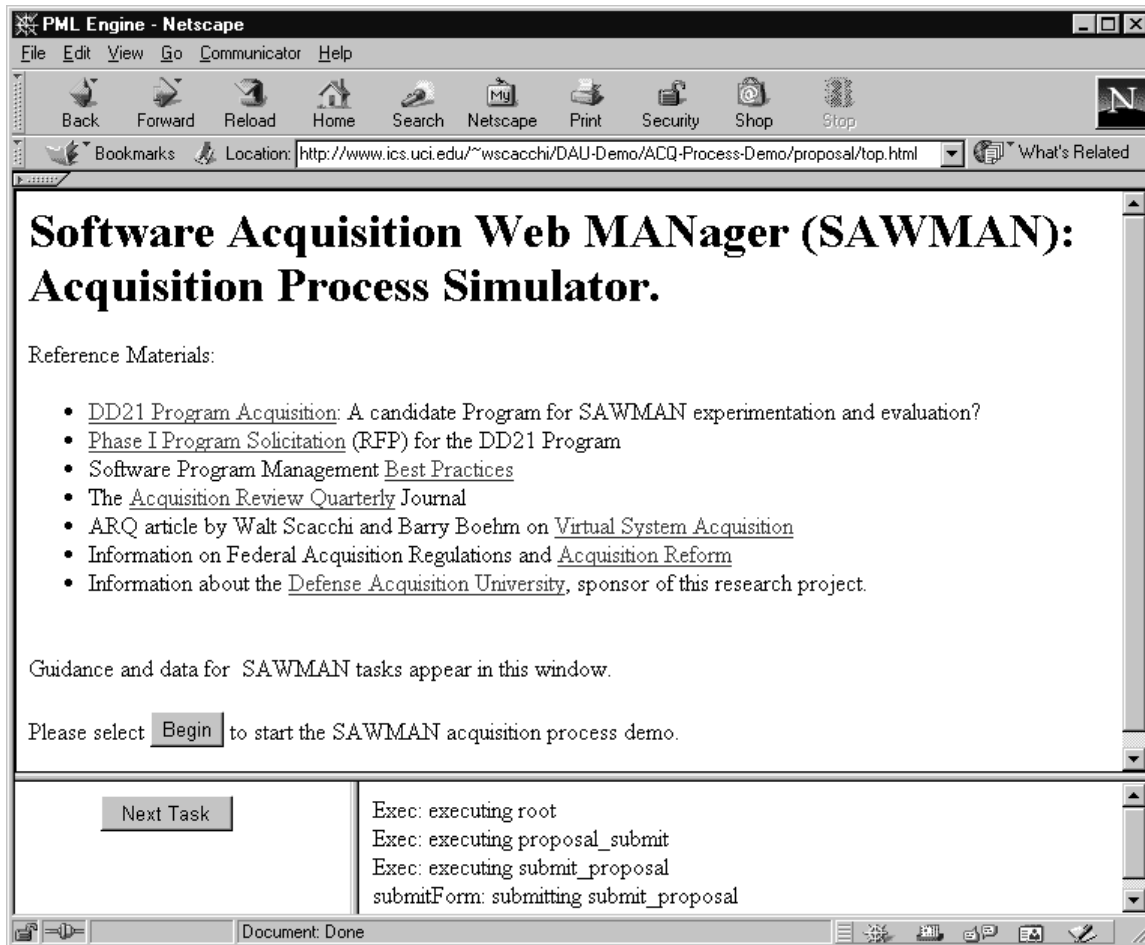


Figure 1. A Screen Display from an Acquisition Process Simulator following [cf. Scacchi 2000].

Beyond providing a process simulator, we also are investigating the use of architecture-level simulation techniques to assess the dynamic performance of alternative process enactment scenarios associated with different software acquisition processes or configured process architectures. Here we have been exploring how the proposed High Level Architecture (HLA) standard and its supporting Runtime Infrastructure (RTI) can be adapted to support the simulation (i.e., simulated enactments) of software process architectures [cf. Kuhl, Weatherly, and Dahmann 2000]. Current implementations of the RTI provide the ability to simulate, monitor, measure and display the performance of a distributed or federated software system architecture (e.g., see <http://www.pitch.se/pRTI>). However, our challenge is to determine the appropriateness and performance of the RTI as a simulation facility for software process architectures in general, and for software acquisition process architectures in particular. Accordingly, in the full paper for the ProSim 2000 Workshop, we intend to characterize our experiences with the RTI as a software process simulation technology.

Conclusions

We conclude by highlighting what's new in this research.

To begin, this work represents the first effort to investigate and provide results on how software process modeling and simulation tools, techniques and concepts can be applied to the domain of software system acquisition.

Next, this is the first effort that combines a process modeling language (PML) and an architectural design language (ADL) to support the modeling of software processes and process architectures.

We also introduce a framework for formalizing and analyzing the construction of internally correct software acquisition process architectures in a manner that can assure such an architecture's consistency, completeness and traceability as a configured software artifact.

Last, we demonstrated ongoing development of a prototype tool that serves as a user-driven (person-in-the-loop) simulator for complex, multi-role, multi-organization processes associated with software acquisition. We also introduced an effort to adapt and assess the viability of the HLA RTI as a platform for simulating the performance of enactable software acquisition process architectures.

Acknowledgements

This work is supported by a grant N487650-27803 from the Defense Acquisition University as part of their External Acquisition Research Program (EARP). Prof. John Noll of the Computer Science Dept. at the University of Colorado at Denver, and Dr. Andre Valente, previously at the USC Information Sciences Institute and now at fastv.com, contributed to the concepts, techniques and tools that we have incorporated in this study. All of these contributions are greatly appreciated.

References

ARO, *Implementing Acquisition Reform in Software Acquisition*, Navy Acquisition Reform Office, <http://www.acq-ref.navy.mil/turbo/refs/software.pdf>, 1999.

J.K. Bergey, M.J. Fisher, and L.G. Jones, *The DoD Acquisition Environment and Software Product Lines*, CMU/SEI-99-TN-004, Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1999.

B. Boehm and W. Scacchi. *Simulation and Modeling for Software Acquisition (SAMSA)*, Final Report, Center for Software Engineering, University of Southern California, Los Angeles, CA, <http://sunset.usc.edu/SAMSA/samcover.html>, March 1996.

J. Choi and W. Scacchi. Formalization and Tools Supporting the Structural Correctness of Software Life Cycle Descriptions, *Proc. IASTED Conf. on Software Engineering, International Association of Science and Technology for Development (IASTED)*, Las Vegas, NV, 27-34, October 1998.

DD21 Information System, <http://sc21.crane.navy.mil>, 2000.

GAO, General Accounting Office. *Air Traffic Control--Immature Software Acquisition Processes Increase FAA System Acquisition Risks*, Report GAO/AIMD-97-47, 1997.

F. Kuhl, R. Weatherly and J. Dahmann. *Creating Computer Simulation Systems: An Introduction to the High Level Architecture*. Prentice-Hall PTR, Upper saddle River, NJ, 2000.

N. Medvidovic, D. S. Rosenblum, and R. N. Taylor. A Language and Environment for Architecture-Based Software Development and Evolution. *Proc. 21st Intern. Conf. Software Engineering*, 44-53, Los Angeles, CA, May 1999.

N. Medvidovic and R. N. Taylor. A Classification and Comparison Framework for Software Architecture Description Languages. *IEEE Transactions on Software Engineering*, (to appear), 2000.

M.E. Nissen, K.F. Snider and D.V. Lamm. Managing Radical Change in Acquisition. *Acquisition Review Quarterly*, 5(2):89-106, Spring 1998.

J. Noll and W. Scacchi. Supporting Software Development in Virtual Enterprises. *J. Digital Information*, 1(4), February 1999a.

J. Noll and W. Scacchi. Process-Oriented Hypertext for Organizational Computing, (submitted for publication), November 1999b.

W. Scacchi. Experience with Software Process Simulation and Modeling, *J. Systems and Software*, 46:183-192, 1999.

W. Scacchi. Understanding Software Process Redesign using Modeling, Analysis and Simulation, *Software Process--Improvement and Practice*, (to appear), 2000b.

W. Scacchi and B.E. Boehm. Virtual System Acquisition: Approach and Transition. *Acquisition Review Quarterly*, 5(2):185-215, Spring 1998.

W. Scacchi and J. Noll. Process-Driven Intranets: Life-Cycle Support for Process Reengineering. *IEEE Internet Computing*, 1(5):42-49, September-October 1997.

SA-CMM, Software Acquisition Capability Maturity Model, Software Engineering Institute, Carnegie-Mellon University, Pittsburgh, PA. 2000.
<http://www.sei.cmu.edu/arm/SA-CMM.html>

SPMN, Software Program Managers Network. *The Condensed Guide to Software Acquisition Best Practices*, October 1997. Available from SPMN at <http://www.spmn.com/products.html>.

STSC, Software Technology Support Center. Guidelines for Successful Acquisition and Management of Software-Intensive Systems: Weapon Systems, Command and Control Systems, Management Information Systems. Volumes 1 & 2. (Version 2.0) Dept. of the Air Force, June 1996. <http://stsc.hill.af.mil/stscguid.asp>

R. Taylor, A. van der Hoek, P. Oreizy, D. Redmiles, D. Rosenblum, W. Scacchi and W. Tracz. Proteus: An Environment for Assessment and Adaptation through Dynamic Adaptation Technology, internal report, Institute for Software Research, UC Irvine, February 2000.