Open Acquisition: Combining Open Source Software Development with System Acquisition

Final Report

Walt Scacchi
Institute of Software Research
University of California, Irvine
Irvine, CA 92697-3425 USA
949-824-4130, 949-824-1715 (fax)
Wscacchi@ics.uci.edu

June 2002

This report can be found on the Web at:
http://www.ics.uci.edu/~wscacchi/Papers/DAU/OpenAcquisition.pdf

The research described in this report was supported by the Defense Acquisition University through contract N487650-27803. No endorsement implied. Mark Ackerman at the University of Michigan, as well as Mark Bergman and Margaret Elliott at the UCI Institute for Software Research, are collaborators on the research described in this paper.

Overview to the Final Report

This report describes the results obtained during the first year of a research project that investigates open software acquisition processes and architectures. These results are organized and presented as a separable research paper in a form intended for external publication. It constitutes an original, previously unpublished research paper, though it also serves to document research that is still in progress. The intent is that the author will then submit research articles derived from this report for journal publication.

The report contain all required sections—including data, analysis, results and conclusions as appropriate, though this is not the exact structure of sections use to organize the presentation of the research results. The report begins with in introductory section to introduce the topic and motivate the problem of open software acquisition. The second section provides background technical information and/or literature review as appropriate for the topic and field of study. The third section is more dependent on the research strategy and other information appropriate for an academic article in the author's field of inquiry. The fourth section include data, analysis, results, findings, discussion or other new knowledge that represents a contribution in the field of study. The fifth section discusses and summarizes key research objectives and findings, along with conclusions that follow, and present an agenda for continued research along the lines of the investigation. A set of references follows adhering to the style and format appropriate for publication in the author's field. Figures and tables are incorporated into the document so the finished product is camera ready.

As such, the research paper that constitutes the Final Report from the first year of this research project follows.

Open Acquisition: Combining Open Source Software Development with System Acquisition

Walt Scacchi
Institute of Software Research
University of California, Irvine
Irvine, CA 92697-3425 USA
949-824-4130, 949-824-1715 (fax)
Wscacchi@ics.uci.edu

Abstract

This study explores and develops concepts leading to the combination of best practices from open source software development (OSSD) projects with emerging capabilities for virtual system acquisition. Virtual system acquisition is an evolving approach to demonstrate significant improvements in reducing the cost and cycle time for acquiring software-intensive systems, while improving their quality. It employs techniques and advanced information technology (IT) for electronic government applications. Open source software development is a relatively new approach to the development, deployment, and ongoing evolution of complex software system applications. The open source approach rethinks what are the resources, products, processes, and production environments necessary to develop large-scale, easy to use, and highly reliable software system applications. Open acquisition is a new concept that combines the best practices from advanced electronic government techniques with those from open source software development. The study described in this paper thus develops, demonstrates initial capabilities, and outlines further steps needed to make open acquisition techniques part of the evolving framework for realizing virtual system acquisition.

Introduction

Acquisition of complex, software-intensive systems is a subject of growing importance and substantial government expenditure both for civilian and military systems. The acquisition of research and system development services gives rise to the new technologies that get incorporated into complex systems. The current U.S. Federal investment in acquiring R&D now exceeds $100B for FY02. Within the DoD community, R&D in the Concept and Technology Development stage is a critical part of the overall acquisition life cycle for major program acquisitions like for the DD(X) [2002] class Naval surface warfighting ships, or the Joint Strike Fighter aircraft for use in all branches of the military in the U.S., and in the NATO ally countries. These are significant acquisitions representing hundreds of billions of dollars in investments and expenditures over periods of 10-20 years. However, the acquisition of complex software-intensive systems has been problematic for decades [Glaseman 1982]. Furthermore, the U.S. Department of Defense currently anticipates shrinkage in its acquisition workforce by up to 50% over the 1995-2005 period due to staff attrition and retirement. Thus, DoD in particular, and the U.S. Government overall, must find new ways to acquire complex systems, as well as the research and development services that get incorporate into them.

One approach to address and investigate problems in acquiring complex software-intensive systems employs the concept of iteratively modeling, simulating, and evolving the specification and architectural design of such systems. This approach is called *virtual systems acquisition* [Scacchi and Boehm 1998]. Virtual system acquisition is a long-term approach to researching and developing radically new approaches for acquiring systems [Nissen, Snider, and Lamm 1998] with lower cost, shorter development cycles, agile and adaptive techniques that can realize high quality systems. Work to date on this approach has resulted in the formulation and demonstration of the ability to model, simulate, and redesign system acquisition processes that can scale to large, distributed process architectures [Choi and Scacchi 2001, Scacchi 2000]. This research work has also produced Web-based environments that enable the prototyping and enactment of digital/electronic government (E-Government) processes and knowledge management capabilities [Scacchi 2001].

The U.S. Federal Government is investigating E-Government strategies for procurement and acquisition [Scacchi and Boehm 1998, Scacchi 2001], data storage and data entry (e.g., electronic filings of tax forms by individuals, and SEC forms by businesses), E-catalog based retail product sales (U.S. Mint), and smart cards [DG.O 2002, DGRC 2002, Steyaert 2001]. The procurement of materials, goods, and contracted services through acquisition activities represents one of the most frequently cited areas where E-Government investments are advocated or being considered. Thus, procurement and acquisition (hereafter acquisition) is compelling point of departure for this study.

How might open source development concepts be combined with emerging E-Government approaches to acquisition? One new way is through open acquisition. *Open acquisition* seeks to open for public sharing, discussion, review, ongoing development and refinement, and unrestricted reproduction (replication and redistribution) the "source code" of the products and processes of the business of government acquisition. Open acquisition represents a concept that seeks more than just the adoption and use of open systems [Meyers and Obendorf 2001], or open source software systems by government agencies in the acquisition community, though such actions are compatible and merit independent consideration.

The open acquisition concept introduced in this study seeks to *explore the opportunities that emerge when one views its purpose as including how to empower and engage an interested acquisition workforce in understanding how acquisition processes and practices can be made better, cheaper, and faster through the development of open source processes, practices, and communities of practice for government acquisition and related operations*. This paper describes these concepts and provides examples of such an approach. It also describes how both the existing and new generation of national information infrastructure may be employed to support interactive participatory development, refinement, redesign, continuous improvement, and community-based evolution.

Background

This section identifies some of the central issues, practices, and opportunities that give rise to the potential of open acquisition. These are found by examining the acquisition of software-intensive systems and open source software development practices. In particular, we are interesting in identifying shortcomings, challenges, and opportunities at hand, in order to establish a foundation for how to proceed towards open acquisition.

The Acquisition of Software-Intensive Systems

We begin by establishing the context for the acquisition of software-intensive systems. Acquisition of systems is governed within the DoD community by the DoD 5000 directives, and the acquisition life cycle process framework that they prescribe. This framework is most commonly conveyed using the graphic in Figure 1.

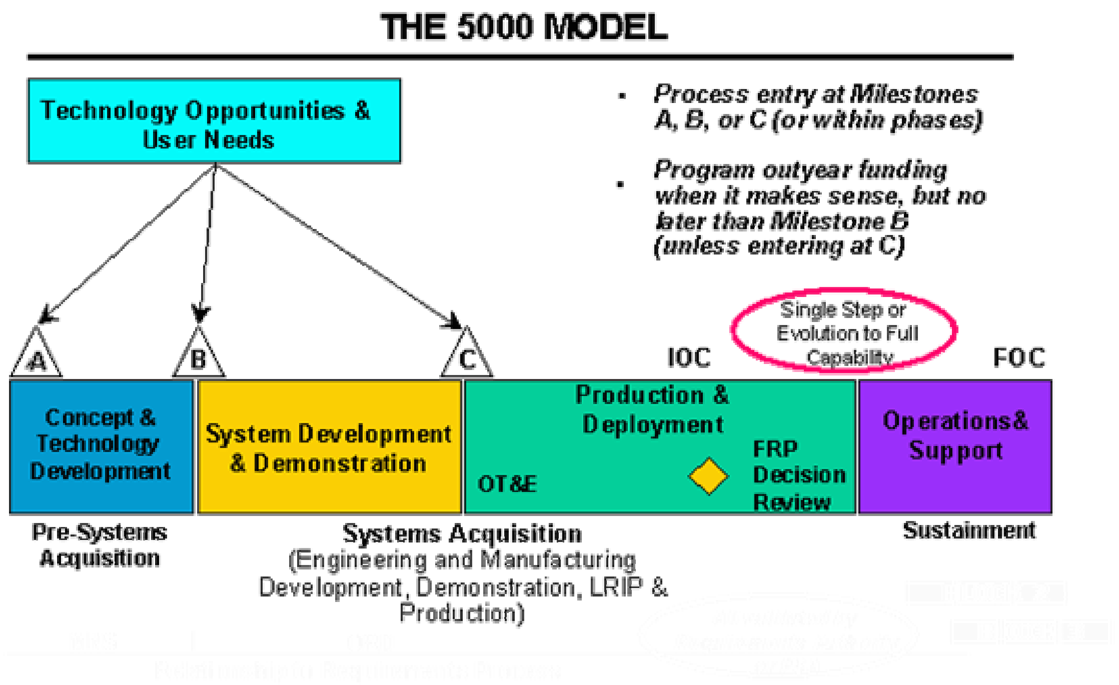
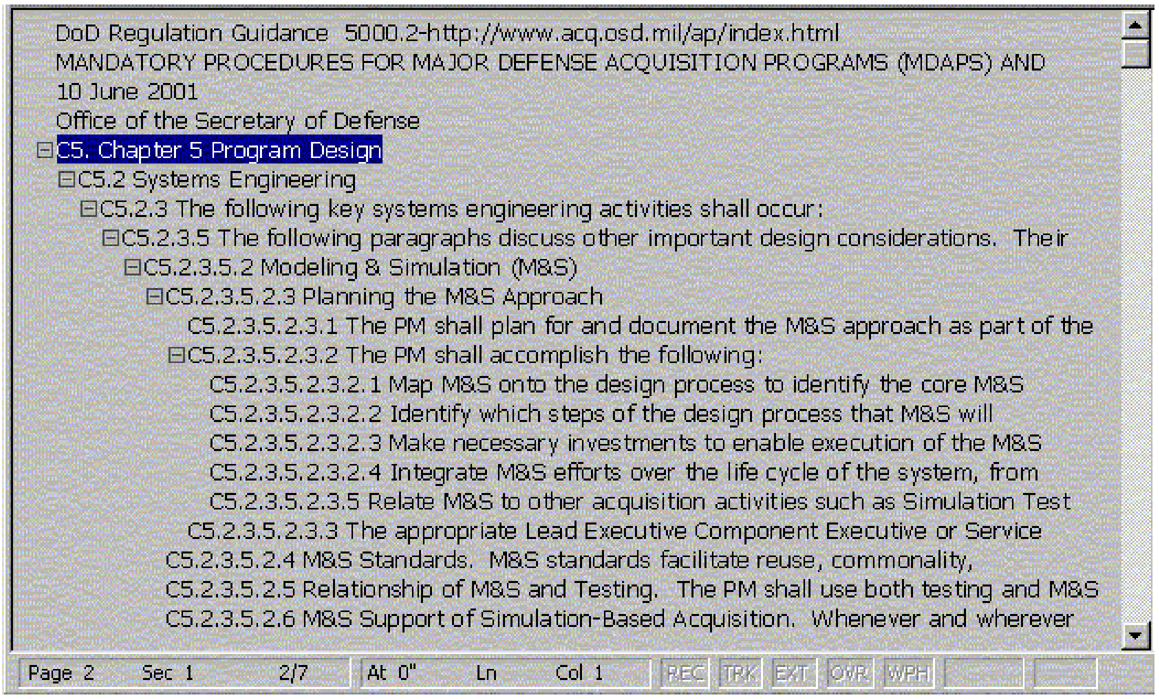

THE 5000 MODEL
Technology Opportunities & User Needs
Process entry at Milestones A, B, or C (or within phases)
Program outyear funding when it makes sense, but no later than Milestone B (unless entering at C)
A
B
C
IOC
Single Step or Evolution to Full Capability
FOC
Concept & Technology Development
System Development & Demonstration
Production & Deployment
OT&E
FRP Decision Review
Operations & Support
Pre-Systems Acquisition
Systems Acquisition (Engineering and Manufacturing Development, Demonstration, LRIP & Production)
Sustainment

Figure 1. The DoD 5000 Acquisition Model [2001]

Here we see an overall view of the acquisition life cycle model as a linear sequence that spans (a) Concept & Technology Deployment, (b) System Development and Demonstration, (c) Production and Deployment, and (d) Operations & Support. Unfortunately, such a model is monolithic, very abstract, and somewhat misleading. It is monolithic in that for each of the four principal acquisition processes, their decomposition, inputs, outputs, and supporting tools/capabilities is opaque rather than transparent. This opacity resists revealing how things are supposed to work, where or how important decisions are made, or how acquisition might be improved or redesigned. This model is also abstract in that the four major sub-processes are themselves very complex processes, since they might be highly iterative, incremental, and ongoing, with many consequential decisions being made throughout. It is therefore misleading to expect that all acquisitions follow the model in a strictly linear manner. Problems like these are well known in the software process research and practice community [Scacchi 2001b]. Furthermore, as we look more closely as the DoD 5000 directives we find little or no reference to how to most effectively operationalize or put the model into practice. We also do not have access to any reference process model that might be amenable to automated enactment support, redesign, or continuous improvement [Scacchi 1999, 2000, Scacchi and Mi 1997, Scacchi and Noll 1997].

In an effort to begin to address these shortcomings, the DoD 5000 directives have been further refined and articulated in a recent set of acquisition guidelines, like the DoD 5000.2-R [2001]. These guidelines stipulate mandatory procedures (or processes) that must be performed in conjunction with the major Defense acquisition programs, which these days are almost always software-intensive systems. However, looking closely at these procedures we know find the following dilemma, as characterized in Figure 2.



```
DoD Regulation Guidance  5000.2-http://www.acq.osd.mil/ap/index.html
MANDATORY PROCEDURES FOR MAJOR DEFENSE ACQUISITION PROGRAMS (MDAPS) AND
10 June 2001
Office of the Secretary of Defense
⊟C5. Chapter 5 Program Design
  ⊟C5.2 Systems Engineering
    ⊟C5.2.3 The following key systems engineering activities shall occur:
      ⊟C5.2.3.5 The following paragraphs discuss other important design considerations.  Their
        ⊟C5.2.3.5.2 Modeling & Simulation (M&S)
          ⊟C5.2.3.5.2.3 Planning the M&S Approach
              C5.2.3.5.2.3.1 The PM shall plan for and document the M&S approach as part of the
            ⊟C5.2.3.5.2.3.2 The PM shall accomplish the following:
                C5.2.3.5.2.3.2.1 Map M&S onto the design process to identify the core M&S
                C5.2.3.5.2.3.2.2 Identify which steps of the design process that M&S will
                C5.2.3.5.2.3.2.3 Make necessary investments to enable execution of the M&S
                C5.2.3.5.2.3.2.4 Integrate M&S efforts over the life cycle of the system, from
                C5.2.3.5.2.3.5 Relate M&S to other acquisition activities such as Simulation Test
              C5.2.3.5.2.3.3 The appropriate Lead Executive Component Executive or Service
            C5.2.3.5.2.4 M&S Standards.  M&S standards facilitate reuse, commonality,
            C5.2.3.5.2.5 Relationship of M&S and Testing.  The PM shall use both testing and M&S
            C5.2.3.5.2.6 M&S Support of Simulation-Based Acquisition.  Whenever and wherever
```

Page 2   Sec 1   2/7   At 0"   Ln   Col 1   REC TRK EXT OVR WPH

Figure 2. A partial view of the nested structure extracted from the DoD 5000.2-R mandatory procedures for major system acquisition [2001]

This figure provides a partial view of the overall hierarchical (or "nested") structure of the DoD 5000.2-R procedures, as shown for Chapter C5. What we see in this outline view is that this chapter, like many others in the procedure set, has at least *eight* levels of nested section sub-structure. Such complexity overwhelms most readers of the nearly 200 page procedures specification, making both performance and compliance with such procedures very difficult to manage, track, or certify. Furthermore, like most software development or systems engineering standards, the guidelines do not exist in a readily usable form, other than as narrative text within an electronic document [cf. Freericks 2001]. What is likely to help here is a transformation of these procedures from an informal narrative, into an open source set of process templates or electronic forms that can be organized and managed as a database [Freericks 2001]. Such a transformation would then establish a foundation for providing computer-based or Web-based deployment, and other forms of browsing, navigation, and (semi-)automated process enactment support. Computer-based models of the processes of the DoD 5000.2-R procedures can follow from the transformation.

Recent research in system acquisition and acquisition reform [cf. Boehm and Hansen 2001, Nissen, Snider, and Lamm 1998, Schooff, Haimes, Chittister 1997] has given rise to a new conceptualization of how dramatic improvements in acquisition cost, speed, and resulting system quality might best be achieved [Anderson 2000, Meyers and Oberndorf 2001, SA-CMM 2000]. These advances, in general, build on best practices from industry or industry-oriented research, or go beyond current "adversarial" views of acquisition in commercial settings [Verville and Halingten 2001]. For example, modeling and simulation of systems across the acquisition life cycle has been identified as a significant opportunity for cost reduction and improvement in resulting system quality [Brown, Grant, *et al*., 2000]. Similarly, virtual system acquisition has demonstrated how large, complex process models for software system acquisition can be modeled, analyzed, simulated, and redesigned using Web-based tools and techniques [Choi and Scacchi 2001, Noll and Scacchi 2001, Scacchi 2001a,b, Scacchi and Boehm 1998]. These research efforts are complementary to emerging practices for E-Government [DG.O 2002, DGRC 2002] that have realized greatest success in applications addressing procurement and acquisition [Nissen 1997, Scacchi 2001b, Steyaert 2001]. Thus the potential exists for modeling, analyzing, simulating, redesigning, and ultimately continuously improving the iterative, incremental, and ongoing practice of major system acquisition processes, as well as other forms or venues for E-Government processes and services.

Elsewhere, we have learned that radical change in acquisition practices will not occur simply by making available new technological alternatives [Nissen, Snider, and Lamm 1998]. Instead, we have found that transformation and sustained improved on acquisition processes and practices will most likely occur when members of the acquisition workforce community are active participants in the transformation effort, and when they are empowered and motivated out of personal and professional interests [cf. Kim 2000, Scacchi 2001]. Such a situation may need to be both bottom-up (action initiated by acquisition specialists) and top-down (management commitment of resources and empowerment of acquisition specialists to make changes) [cf. Scacchi 2001]. Thus, the lesson here is to pursue a course of action that combines or enables the capabilities of the acquisition workforce community to take advantage of the new technologies for acquisition support, as they participate in redesign and transform acquisition processes and practices.

Open Source Software Development Practices

Open source is not the same concept as "open systems". Open source is a broader, more encompassing technique for exposing access to the underlying functionality, operation, or interoperation of a software system. *Open systems* traditionally refer to a technology scheme that provides customers, external developers, or end-users to access the internal functions of a complex system via "public interfaces." These interfaces take the form of open, accessible connectors or plug sockets. The structure of these interfaces denote the points of contact through which pre-specified types of program data, control signals, and error messages flow in or out.

In open source software (OSS), the source code, as well as its surrounding documents and artifacts, all serve as the public interface to the system. Access to system functionality is not limited to functions calls through "application programming interfaces" (APIs). Access to functionality, as well as the ability to enhance, restructure, tune, debug, or re-host system functionality is realized through access to open source code, documents, and artifacts. An open system may consist of hardware, software, and network system components. Subsequently, the potential exists for making all of the components functionality accessible through public interfaces that consist of the components "source code", documents, and artifacts. Figure 3 provides a view of the layers of software, hardware, and network components that can be found in an open source, open system. As Figure 3 suggests, an open system can itself be composed out of open source components.

With this in mind, we now turn to examine the products, processes, and support environments for OSS.

Open Source Software Products

OSS program code is the typical focus of most open source development activities. These computer programs are written in a programming language like C, C++, Perl, Python, or others. Documents that specify or describe how these programs function or interoperate are also products of open source software development. These documents may include specification or design diagrams, end-user manuals, program installation scripts, threaded email discussion forums, Web-based source code repositories, and other Web site contents [Scacchi 2002b]. OSS development (OSSD) projects rely on diversity of software *informalisms* [Scacchi 2002b] as information resources, documents, artifacts, or products that can be browsed, cross-linked, and updated on demand. These informalisms are socially lightweight information structures for managing, communicating, and coordinating globally dispersed knowledge about who did what, why, and how. These informalisms are easy to learn and use as semi-structured representations that capture software requirements, system design, and design rationale. As OSS developers are themselves end-users of their systems, then software system requirements and design take less time to articulate and negotiate, compared to system acquisition projects that must elicit requirements and validate system design with end-users who are generally not software system specialists. Thus, a lesson learned from these observations is that practitioners of open acquisition should be both users *and* developers of system acquisition practices, and these practitioners should be provided with the ability to easily

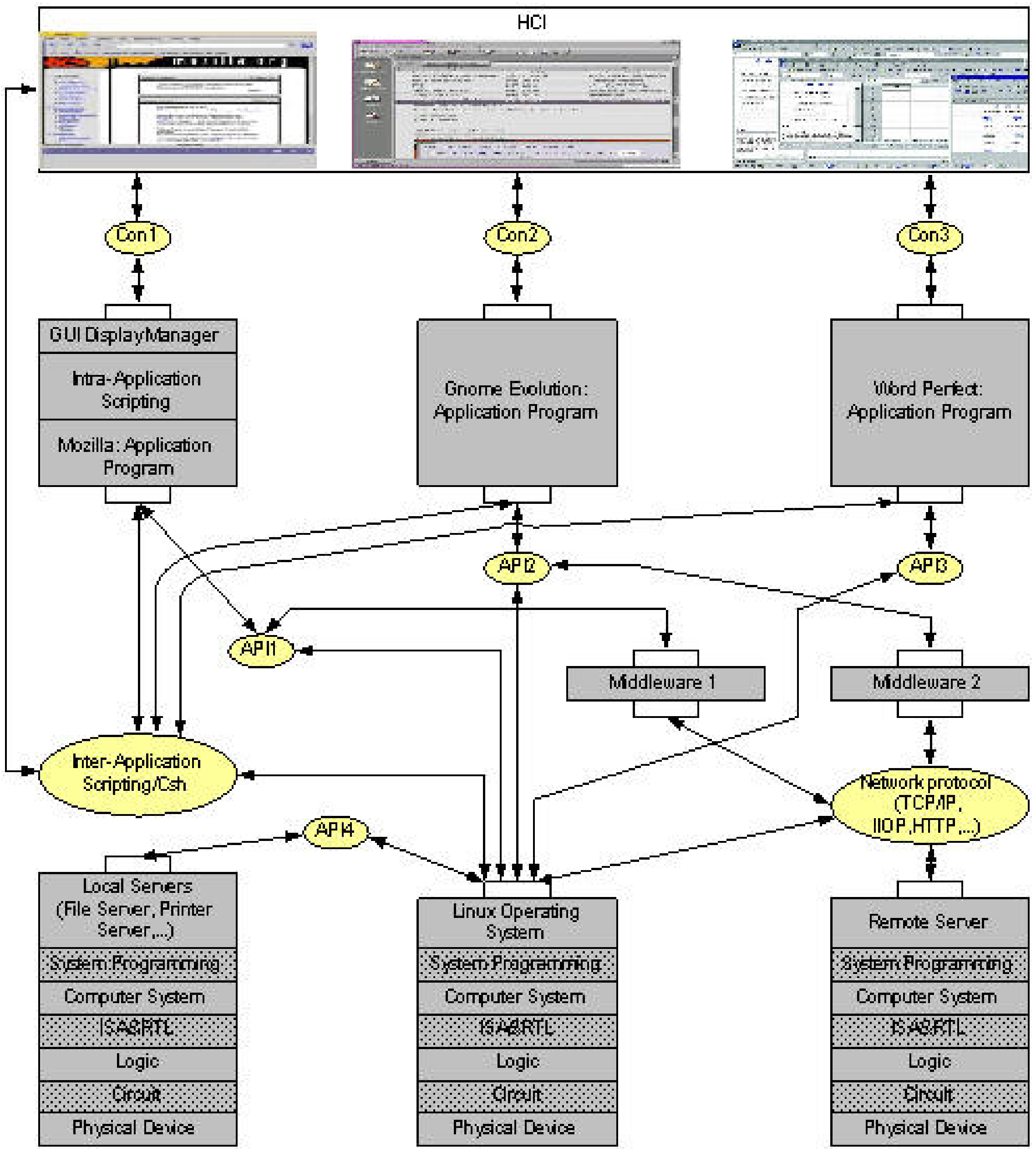


Figure 3. An example view of an open system of software and hardware components, that may also be found in open source formats.

create, modify, share, and discuss informal descriptions of open acquisition products, practices and outcomes. We recognize the opportunity for the templates or electronic forms needed to better support the DoD 5000.2-R procedures might themselves be treated and supported as informalisms for open acquisition.

OSS concepts can apply to any product that can be produced with or through the use of computing systems that can be networked together. As seen next, these products can also include the source code to processes for technical work or business operations involved in system acquisition, if these processes can be specified in a form that can be compiled or interpreted for automated or interactive execution in a networked computing environment.

Open Source Processes

OSS programs emerge as the result of technical activities that are arranged and ordered in a manner that can be described as a *process*—an OSSD process. This process may be ad hoc, difficult to describe and repeat, or it could be more structured and follow a pre-articulated scheme or formal process. In any case, it is reasonable to describe how the OSS products are developed, used, and evolved as processes. Furthermore, these processes may be descriptive, proscriptive, or prescriptive. Descriptive processes describe what occurred, by whom, when, where, etc. Proscriptive processes describe what could be done at certain points or in response to some event or condition. Last, prescriptive describe how software development activities are suppose to be done, perhaps supported by some means or mechanism that checks for compliance with the process prescription. In any of these forms, processes can be codified into computational models that can be analyzed, simulated, and iteratively refined by process users in many different ways [Noll and Scacchi 2001, Scacchi 2000, 2002a, Scacchi and Mi 1997, Scacchi and Noll 1997]. Examples will follow in a later section.

OSSD projects enact "Internet time" development practices, much like Microsoft, Netscape, and others [Cusumano 1999, MacCormack 2001] follow when market conditions demand rapid response to substantial competitive threats [cf. Meyers 1993]. Internet time efforts emphasize minimizing time to market and delivery of incremental improvements in functionality, instead of complete well-engineered functionality that gives rise to much less frequent product releases. Internet time development also focuses on collecting feedback from early users as a way to determine which incremental functionality, and which perceived errors in available functionality matters most, as a way to determine emerging or shifting system requirements [Truex, Baskerville, and Klein 1999].

OSSD projects are iteratively developed, incrementally released, reviewed and refined by OSS developers working as peers in an ongoing agile manner [cf. Cockburn 2001]. These methods ensure acceptable levels of quality, coherence, and security of system-wide software via continuous distributed peer review, testing and profiling. OSSD efforts are hosted within decentralized communities of peers [Scacchi 2001, 2002, Sharman 2002] that are interconnected via Web sites and OSS repositories. Community oriented OSSD also gives rise to new kinds of requirements for community building, community software, and community information sharing systems (Web site and interlinked communication channels for email, forums, and chat). In contrast, most system engineering projects associated with major system acquisition efforts are targeted for a centralized corporate setting, where access and visibility may be restricted to local participants. OSSD standards [Freericks 2001] reinforce best practices are apparently easier to access and follow due to their Web-based deployment, and a long history of community oriented participation in developing implementation-oriented standards in an open source manner. These compare favorably to the institutionally oriented processes used to develop software engineering and acquisition standards that are much more cumbersome and often less effective at ensuring system quality.

Open Source Support Environments

OSS emerges from the efforts of software developers who are typically distributed across space and time. They do not work in a single or central workplace, and often there is no formal management hierarchy in place to schedule, plan, and coordinate who does what, with what resources, etc. Instead open source developers contribute their effort to projects that they find interesting, significant, or otherwise professionally compelling. Open source developers generally have regular paid jobs, though they may or may not be paid to work on an open source project. Thus, traditional organizational models for how to motivate employees or how to organize and manage technical staff may not apply. Nonetheless, open source development projects thrive, as it now appears that tens of thousands of OSSD projects are underway.[1]

OSSD projects are "organized" as *a loosely knit community of interested developers and end-users* who work and interact online via Web-based computing environments [Scacchi 2002b]. These environments provide access to a global information infrastructure that includes routine support for email and discussion forums (electronic bulletin board), Web sites for posting or sharing open source artifacts, centrally administered multi-version source code directories, software extension schemes and mechanisms (e.g., multi-application scripting languages, like Perl and Python, for creating interoperating systems), and more [Scacchi 2002]. Developing trust, "geek fame", and being recognized by peers for making technical contributions [Pavlicek 2000] are part of the "glue" that binds open source developers together with their global information infrastructure to create the productive units or virtual organizations [Noll and Scacchi 1999] that populate the world of OSSD. These virtual organizations are thus part of what must be reproduced and enacted in the world of open government and open acquisition.

OSSD tools are inexpensive/free, comparatively easy to use and learn. These tools are freely available at little/no cost. They are both given and received as public goods or gifts [Bergquist 2001]. The most widely used OSS tools support concurrent version control and repository management, Web servers and browsers, communication applications

[1] For example, the Web portal site, www.sourceforge.net, identifies more than 40,000 registered open source development projects and more than 400,000 open source developers. 15% of these projects are identified by their developers as "stable" systems suitable for production application, or "mature" systems being sustained and incrementally evolved to improve their usability, system performance, and to expand the diversity of platforms on which they operate.

(threaded email discussion forums, instant messaging), bug/issue reporting and resolution tracking, and various code development tools (text editors, integrated development environments, etc.). Access to and availability of OSS tools is generally not a problem or barrier to participation in an OSSD project.

Faster and better OSSD conditions in tend to drive down the cost of developing software, at least in terms of schedule and budget resources. Most OSSD projects are voluntarily staffed who want to work on the project, who will potentially commit their own time and effort, and who find personal and professional benefit from the OSSD development efforts [Scacchi 2002]. Minimal management or governance forms [Sharman 2002] are used to direct OSSD efforts, compared to the more rigidly hierarchical, managed, planned, staffed, controlled, and budgeted project activities typical for traditional system engineering efforts associated with major acquisition programs.

Research Strategy

In order to explore, develop, and demonstrate the concept of open acquisition, we need a scheme that articulates and integrates relevant concepts, techniques, and tools that span the background issues already described. This means we seek to identify results from research into system acquisition, virtual system acquisition, E-Government, and studies of OSSD to lay the foundation for how open acquisition might be realized and demonstrated. The results of most interest are those that identify problems, opportunities, or challenges in acquisition and OSS products, processes and support environments. With this we can then develop and demonstrate/prototype an approach to open acquisition, as well as lay out a research agenda for further study.

Among the problems we found in looking at system acquisition in the context of the DoD 5000 model and mandatory procedures, most outstanding was the lack of standard electronic forms or templates that could be used to capture information that represents progress through the acquisition life cycle process. Similarly, we found the acquisition process was in a complex, highly nested narrative form that is not amenable to automated support. Subsequently, we also found there are essentially no automated support tools or environments which enable acquisition products to be disseminated in standardized electronic forms, instantiated, tracked and managed across a community of users who might be connected by the Web.

OSSD informalisms appear to be suitable candidates for the products of acquisition. Web-based process modeling notations may also serve as semi-structured informalisms that can be designed to be enactable via a Web-based process modeling and enactment environment. Web-based tools and environments have been previously demonstrated to be suitable to support the participative redesign of procurement and acquisition processes appropriate for E-Government applications. Tools from virtual system acquisition research also enable the distributed modeling and simulation [Kuhl, Weatherly, and Dahlmann 1999] of complex acquisition processes into process architectures that can be prototyped for enactment and review over the Web [Choi and Scacchi 2001]. Web-based community portals and other freely available OSS applications and tools are also essential elements that provide an opportunity for entry into the dispersed community of (acquisition workforce) users as developers and maintainers of their own (acquisition) OSS products, processes, and support environments.

Participative redesign of acquisition processes requires the processes be described in accessible open source notations, that can be modeled, analyzed, simulated, prototyped and redesigned in a distributed, iterative, and peer-reviewed manner [cf. Scacchi and Mi 1997, Scacchi and Noll 1997]. This is similar in kind with how OSS is developed, deployed, reviewed, and redesigned. Thus, we need an open source process modeling notation that is capable of supporting acquisition products, as well as simulating and enacting acquisition processes, in a manner that can be supported by process-directed, Web-based tools or environments.

With these concepts, techniques, and tools in mind, we can now provide the results of a small exploratory study that seeks to put this research strategy into a demonstrable prototype form.

Prototyping Study and Results

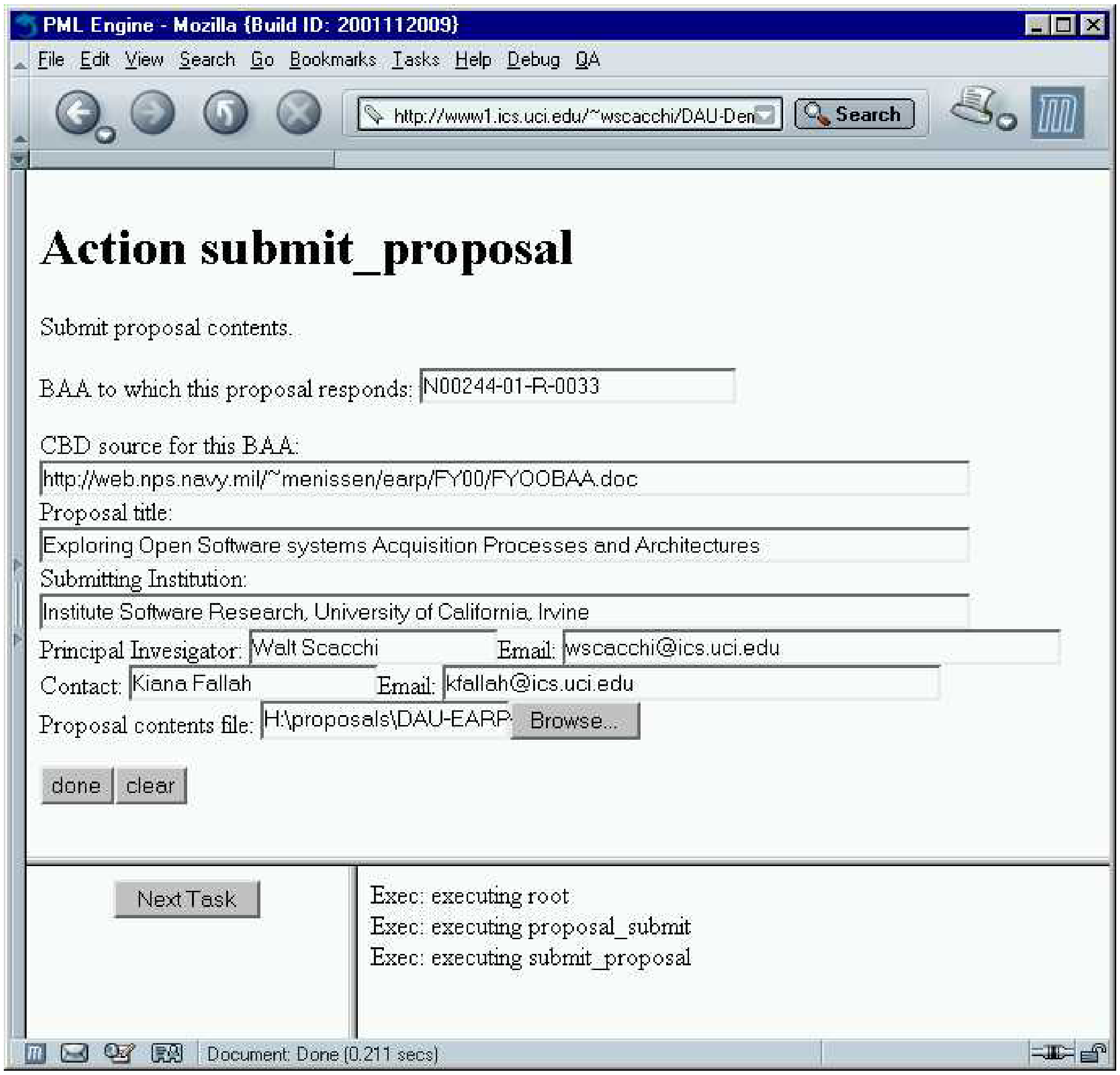
Our goal at this point is to present the results of an exploratory study that seeks to demonstrate the concepts, techniques, and tools for open acquisition. The study combines research results from system acquisition, virtual system acquisition, E-Government, and open source software development, following the research strategy above.

In this study, we propose a candidate process modeling language, PML [Noll and Scacchi 2001] to serve as an OS process modeling notation for specifying open acquisition processes. Exhibit 1 provides an example of a PML specification for a process for submitting proposals, proposal budgets, and certifications in response to a request for proposals [Nissen 1997], in this case, which appear within a Broad Agency Announcement (BAA). The exhibit presents an excerpt from the process, since the complete specification of this process is a few pages in length. What is presented reveals key features of a process model specification. These include the naming of the process, its component (process step) actions that denote where user-interaction occurs. It also identifies the required (input) and provided (output) resources for each process component, and the user (agent) roles for each process component. Then it includes a script that specifies input forms and outputs (products) that denote informalisms associated with each process step. PML thus offers a notational scheme that can specify acquisition processes. However, it may be clear that PML is not a natural language, thus it does require some technical skill for developing a process description in a form suitable for process modeling. Thus, it seems that perhaps a more visual scheme for graphically modeling of processes that automatically constructs the PML notation from a process flow-chart built visual process editing tool with may be needed to facilitate a more diverse set of users [cf. Scacchi and Mi 1997].

```
process Proposal_Submit {
    action submit_proposal {
        agent { PrincipalInvestigator }
        requires { proposal }
        provides { proposal.contents == file }
        script {"<p>Submit proposal contents.\
        <p>BAA to which this proposal responds: \
        <input name='baa' type='string' size=16>\
        <p>CBD source for this BAA: \
        <input name='cbd' type='string' size=50>\
        <br>Proposal title: <input name='title' type='string' size=50>\
        <br>Submitting Institution: <input name='institution' type='string' size=25>\
        <br>Principal Investigator: <input name='PI' type='string' size=20>\
        Email: <input name='PIemail' type='string' size=20>\
        <br>Contact: <input name='contact' type='string' size=20>\
        Email: <input name='contactEmail' type='string' size=12>\
        <br>Proposal contents file: <INPUT NAME='file' TYPE='file'>"
         }
      }
    action submit_budget {
        agent { PrincipalInvestigator }
        requires { proposal }
        provides { proposal.budget == file }
        script {"<p>Submit budget.\
        <br>Proposal title: <input name='title' type='string' size=50>\
        <br>Budget file: <INPUT NAME='file' TYPE='file'>\
        <br>Email address of contact: <input name='user_id' type='string'>"
         }
      }
    action submit_certs {
        agent { PrincipalInvestigator }
        requires { proposal }
        provides { proposal.certs == file && proposal.certifier == user_id }
        script {"<p>Submit electronically signed certifications.\
        <br>File containing signed certifications: <INPUT NAME='file' TYPE='file'>\
        <p>User ID of signature: <input name='user_id' type='string'>"
         }
      }
}
```

Exhibit 1. An excerpt from an acquisition process specified in a candidate open source process modeling language (Noll and Scacchi 2001).

Figure 4 provides a display of an electronic form that is used to capture information about a proposal being submitting a part of the modeled acquisition process.


PML Engine - Mozilla {Build ID: 2001112009}
File Edit View Search Go Bookmarks Tasks Help Debug QA
http://www1.ics.uci.edu/~wscacchi/DAU-Den
Search

Action submit_proposal

Submit proposal contents.

BAA to which this proposal responds: N00244-01-R-0033

CBD source for this BAA:
http://web.nps.navy.mil/~menissen/earp/FY00/FYOOBAA.doc

Proposal title:
Exploring Open Software systems Acquisition Processes and Architectures

Submitting Institution:
Institute Software Research, University of California, Irvine

Principal Invesigator: Walt Scacchi Email: wscacchi@ics.uci.edu

Contact: Kiana Fallah Email: kfallah@ics.uci.edu

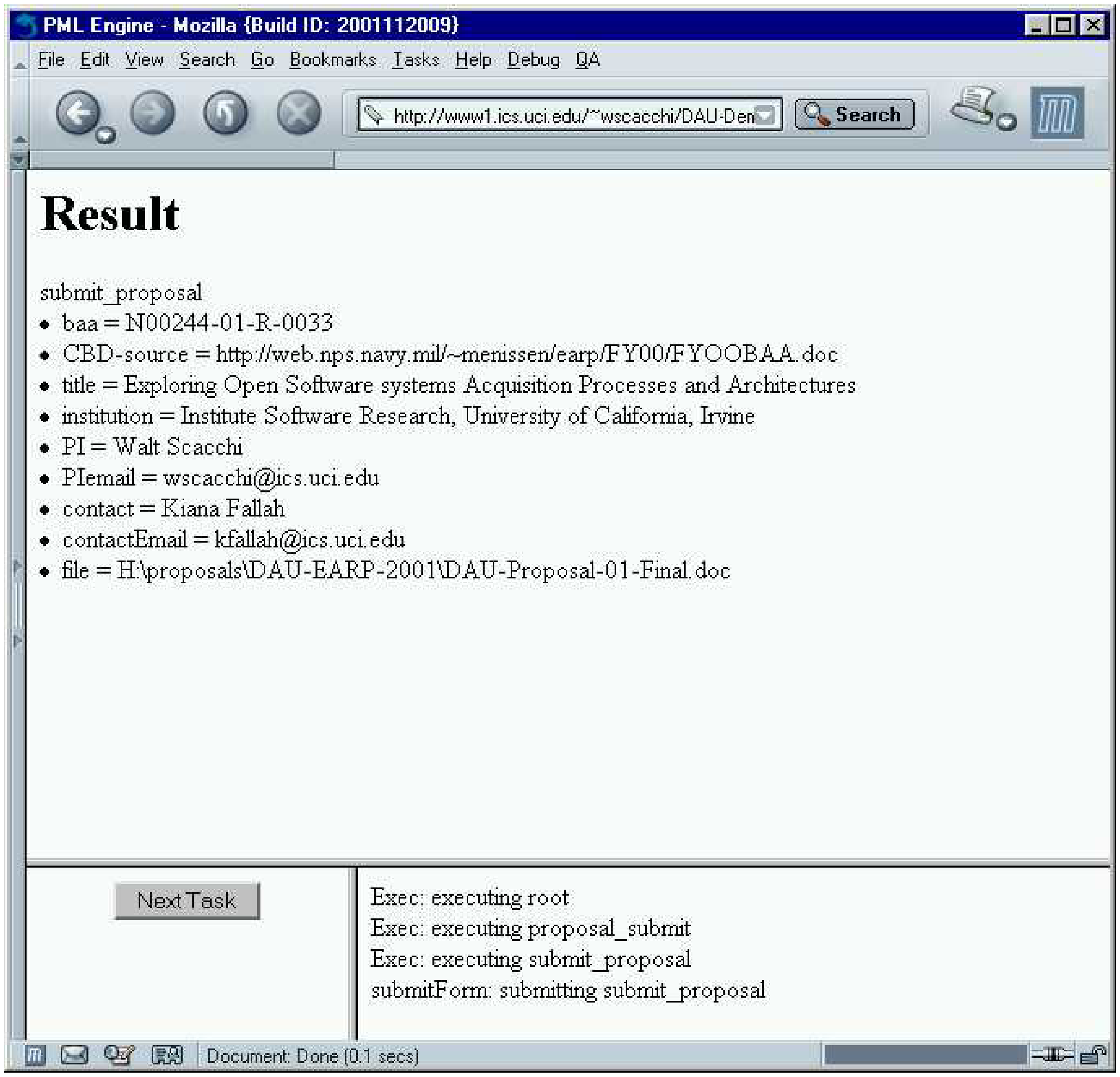
Proposal contents file: H:\proposals\DAU-EARP

```
Exec: executing root
Exec: executing proposal_submit
Exec: executing submit_proposal
```

Figure 4. An electronic form that captures information required for an proposal submission process as part of a mandatory acquisition procedure

This electronic form captures data such as the BAA solicitation number, the BAA source document, and other information provided by the proposal submitter. Information in the lower part of the screen provide navigation control buttons for the end-user (e.g., a proposal author or submitter) to signal progress in moving to complete the specified process step. In addition, in the lower right, a record of activities being tracked by the process support system is also reported to the user. This information is optional, but provides a scheme for documenting what activities were performed, and it is comparatively easy to add timestamps that would make for a more complete audit trail.

Figure 5 displays the output that result for the processing of the form from Figure 4. This "product" represents information that is uploaded into a target database management system that collects and stores the proposal submission data. This display is optional in that it displays what information is uploaded, rather than how it might be reformatted for end-user display or reporting purposes.


PML Engine - Mozilla {Build ID: 2001112009}
File Edit View Search Go Bookmarks Tasks Help Debug QA
http://www1.ics.uci.edu/~wscacchi/DAU-Den
Search
Next Task
Document: Done (0.1 secs)

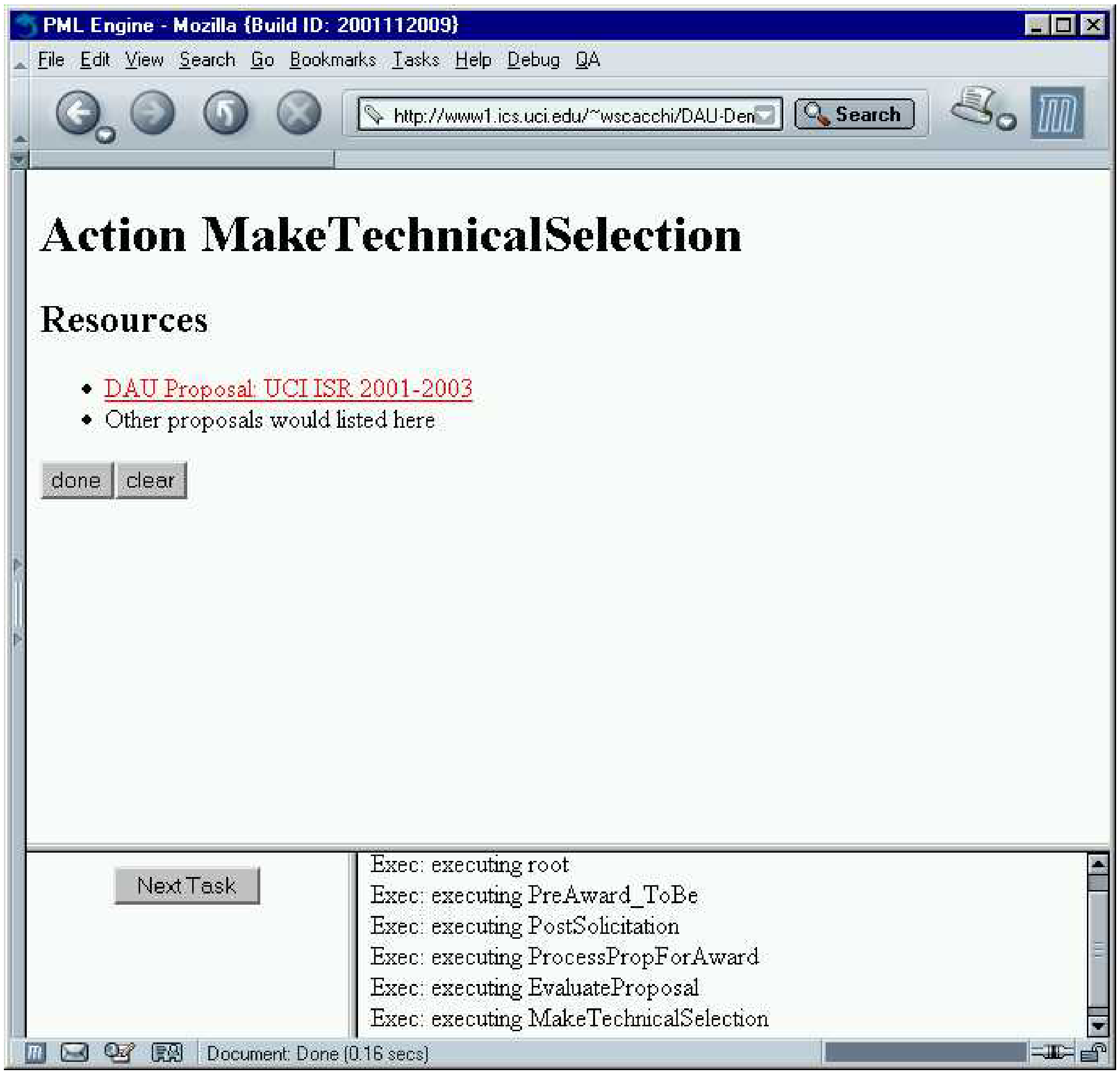
Result

submit_proposal
- baa = N00244-01-R-0033
- CBD-source = http://web.nps.navy.mil/~menissen/earp/FY00/FYOOBAA.doc
- title = Exploring Open Software systems Acquisition Processes and Architectures
- institution = Institute Software Research, University of California, Irvine
- PI = Walt Scacchi
- PIemail = wscacchi@ics.uci.edu
- contact = Kiana Fallah
- contactEmail = kfallah@ics.uci.edu
- file = H:\proposals\DAU-EARP-2001\DAU-Proposal-01-Final.doc

```
Exec: executing root
Exec: executing proposal_submit
Exec: executing submit_proposal
submitForm: submitting submit_proposal
```

Figure 5. A display of outputs collected from the electronic form in Figure 4 for transfer into a remote database management system supporting acquisition processes.

Other process actions depicted in Exhibit 1 follow a similar pattern of presenting electronic forms for capturing mandatory acquisition information about the submission of budget and certification documents. These actions are not shown. Figure 6 jumps to a later point in the acquisition process where a Program Manager has the opportunity to select a proposal for a funding award, if the proposal is acceptable and can be subsequently justified and approved [cf. Scacchi and Noll 1997]. So a Program Manager would select proposals for review from among those available (though in this example, only one proposal is available for review or selection). The Program Manager would then "click" on the proposal (hyperlink) chosen for review.


PML Engine - Mozilla {Build ID: 2001112009}
File Edit View Search Go Bookmarks Tasks Help Debug QA
http://www1.ics.uci.edu/~wscacchi/DAU-Den
Search

Action MakeTechnicalSelection

Resources

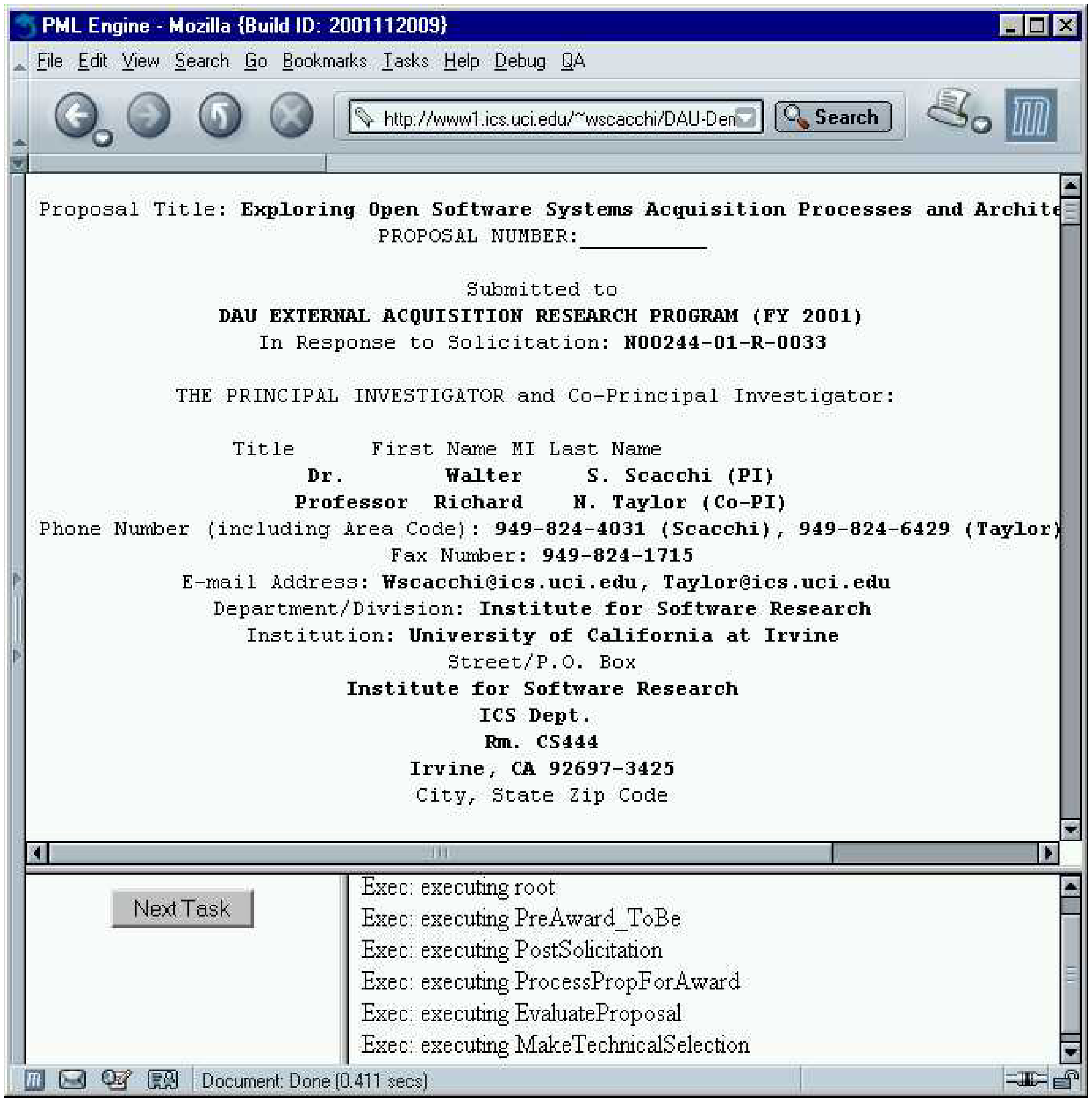
- DAU Proposal: UCI ISR 2001-2003
- Other proposals would listed here

```
Exec: executing root
Exec: executing PreAward_ToBe
Exec: executing PostSolicitation
Exec: executing ProcessPropForAward
Exec: executing EvaluateProposal
Exec: executing MakeTechnicalSelection
```

Document: Done (0.16 secs)

Figure 6. A form for selecting from a list of pending proposals awaiting review and a decision to select for funding by an acquisition Program Manager.

Figure 7 displays the result of clicking on the proposal the Program Manager has chosen to review from those on the preceding list. The PM can now browse and read the text of the proposal prior to making a selection decision. In this manner, it may be possible or convenient for a PM to be able to browse multiple proposals concurrently by viewing each one in a separate window. Such a choice would be at the discretion of the PM user.


PML Engine - Mozilla {Build ID: 2001112009}
File Edit View Search Go Bookmarks Tasks Help Debug QA
http://www1.ics.uci.edu/~wscacchi/DAU-Den
Search
Next Task
Document: Done (0.411 secs)

```
Proposal Title: Exploring Open Software Systems Acquisition Processes and Archite
                           PROPOSAL NUMBER:__________

                                  Submitted to
                 DAU EXTERNAL ACQUISITION RESEARCH PROGRAM (FY 2001)
                    In Response to Solicitation: N00244-01-R-0033

             THE PRINCIPAL INVESTIGATOR and Co-Principal Investigator:

                  Title       First Name MI Last Name
                        Dr.        Walter     S. Scacchi (PI)
                       Professor  Richard    N. Taylor (Co-PI)
Phone Number (including Area Code): 949-824-4031 (Scacchi), 949-824-6429 (Taylor)
                            Fax Number: 949-824-1715
            E-mail Address: Wscacchi@ics.uci.edu, Taylor@ics.uci.edu
              Department/Division: Institute for Software Research
                 Institution: University of California at Irvine
                                Street/P.O. Box
                         Institute for Software Research
                                   ICS Dept.
                                   Rm. CS444
                              Irvine, CA 92697-3425
                              City, State Zip Code
```

```
Exec: executing root
Exec: executing PreAward_ToBe
Exec: executing PostSolicitation
Exec: executing ProcessPropForAward
Exec: executing EvaluateProposal
Exec: executing MakeTechnicalSelection
```

Figure 7. A display that provides a Program Manager user with the source text to a proposal submitted for review and selection decision by the PM.

With these examples in mind, we can summarize the capabilities that have been demonstrated and displayed in Exhibit 1 and Figures 4 through 7.

First, the "source code" in Exhibit 1 seeks to convey that acquisition processes can be specified using a process modeling notation. Such a notation, in this case PML, may be a suitable candidate for use as an open source notation for specifying open acquisition processes.

Second, Figure 4 demonstrates the capability to specify electronic forms, such as those that are mandatory for acquisition. Though such an example may not seem too compelling, the forms or templates required by the mandatory DoD 5000.2-R [2001] procedures could be described in a common notation that could serve as open source approach for implementing acquisition guideline standards [cf. Freericks 2001]. Thus, PML supports the specification of electronic forms, artifacts (e.g., reports, screen displays, hypertext links, etc.), and other informalisms that might be products of open acquisition processes.

Third, the electronic forms and the sequences in which they appear are part of the overall workflow that is implied in the DoD 5000 model and directives. The order in which these forms are presented to the appropriate role-specific end-user, is subject to change over time, much like the content and diversity of forms used to collect the mandatory reporting data. The forms and the presentation sequence, when specified in an open source process modeling notation, can therefore be subject to distributed review and redesign within the acquisition workforce community.

Fourth, Figures 4 through 7 point to the ability to develop and deploy process-directed environments that can model, prototype, or enact open acquisition processes. In this regard, PML becomes the source code language for "programming" open acquisition processes that can be coded, browsed, reviewed, and executed by members of the acquisition workforce community. This of course is a technical task that not all members of the acquisition workforce may choose to master, but for those who do, they become the community members who are empowered to develop, code, review, or redesign open acquisition processes. As noted above, the availability of other tools, like a visual process model editor, may make this task simpler or easier to learn and master.

Fifth, Figures 4 through 7 also convey the capability that the process-directed environment for modeling, prototyping, and enacting open acquisition processes is Web-based. This is demonstrated by the use of a common Web browser as the user interface to the process environment. This of course implies that any information that might be accessible on the World Wide Web, such as the *Defense Acquisition Deskbook* [2001], can also be accessed or integrated into an open acquisition process Web environment. Furthermore, in these examples, we use the Mozilla Web browser, which is itself a large open source program that is available for use on a variety of computing platforms, as well as subject to continuous improvement from a very large community of Mozilla users distributed around the world. Therefore, the process-directed environment for supporting and enacting open acquisition products and processes can be accessed essentially anywhere acquisition workforce community members may be.

Sixth, as found in related research in virtual system acquisition, a PML-like language for specifying acquisition process models, could also be targeted for performance analysis through distributed simulation [Choi and Scacchi 2001, Kuhl, Weatherly, and Dahlmann 1999]. Such a capability for distributing the simulation of open acquisition processes, is to enable a broader range of acquisition workforce community members to participate in running and reviewing the results of given simulation run. This serves to insure that sophisticated acquisition process redesign tools and techniques are available on a global basis, if needed.

Seventh, a more primitive version of PML and its Web-based process environment has been used to support the redesign of contract management processes used within the Acquisition Directorate at the Office of Naval Research [Scacchi 2001]. In that setting, staff members from acquisition workforce at ONR Headquarters were empowered by senior management to redesign their acquisition sub-processes by interacting with developers capable of using the PML and its Web-based environment. ONR staff from other ONR field offices across the U.S. could at their convenience access and provide distributed peer reviews of the processes designed or redesigned by Headquarters staff, so as to collectively improve the process forms and sequencing so that it met their needs. This ability to participate in an open acquisition process redesign, distributed peer review and refinement was judged to be a critical factor by the ONR acquisition workforce staff in achieving the overall success that was realized. In their setting, the success was measured by a drop in "procurement action lead time", the acquisition performance metric used at ONR to measure process cycle time [cf. Meyers 1993], from weeks to days and then to hours [Scacchi 2001].

Last, given the capabilities and accomplishments demonstrated in this exploratory study, it should be clear more needs to be done before these results can be applied and deployed for widespread use across the acquisition workforce community. In particular, in order to succeed, we believe the acquisition workforce community needs to (a) be seeded with start-up resources, capabilities, and personally and professionally motivated developers, in order to (b) take over community ownership of open acquisition products, processes, and support environments. This opinion is derived from some additional, yet preliminary observations about what enables OSSD projects to succeed. The central observation motivating this opinion comes from an ad hoc review of "stable" and "mature" OSSD projects accessible from the www.sourceforge.com community portal Web site (cf. Footnote 1). Here it seems that most sustained OSSD projects do not start from a blank sheet, but instead start from an investment that provides a working system that its community of developers and users find attractive and full of potential to be improved and grow into something great. The exploratory study described here is merely a demonstration of the approach and capability, rather than the effort needed to provide a viable "seed" that the acquisition workforce community can take ownership of and grow into something evermore useful, powerful, and capable. Nonetheless, the research results presented here do being to show the direction for how to realize such potential.

Discussion, Conclusions and Agenda

Prior studies of computing and bureaucratic reform in government operations indicate modest success in improving operational efficiency, while more frequently reinforcing the dominant political order as the overall outcome. More substantial transformation of government operations and business processes like acquisition occurs through participation, empowerment, engagement, and accountability of the people who get to design or redesign their work processes and practices in a manner that can be supported by their information systems. Complex systems need to be designed and evolved in a manner that continually engages its developers as users, and enables end-users to act as developers, builds and sustains community values via sharing and caring for the system. Systems that are opaque or inaccessible to the capabilities and interests of their developers, users, or maintainers, won't encourage or facilitate the emergence of a community of interest that wants to support, sustain, and evolve the system. A community of peers, or even a meritocracy, cannot emerge or be formed by administrative order, command, or directive. Instead the participants must be able to access and change things about a system's operation or function in a manner they find socially empowering or personally satisfying.

Providing computer processable, open source representations of acquisition products (artifacts), processes, and support environments enables participants to see and discuss their existing as-is work practices, as well as identify problematic aspects and opportunities for improvement. A process representation notation that accommodates the open source specification of acquisition processes [Scacchi and Noll 1997, Scacchi 2001] needs to be developed, shared, peer reviewed, and evolved by the acquisition community. Providing an open source model of the to-be work processes that incorporates improvements that participants empowers them to select among additional suggested process redesigns that can mitigate or remove the problematic aspects they perceive. This kind of process redesign capability was successfully demonstrated in a case study performed within the acquisition directorate of the Office of Naval Research [Scacchi 2001].

Strategic assurances among the process owners, users, and system developers to work together in different roles but as peers, encourages accountability and shared responsibility, rather than suspicion, doubt, or bureaucratic obsfucation. Web-based prototypes provided the opportunity for remote participants to review, comment on, and identify further improvement with what the core developers achieved prior to their input.

Overall the transformation that occurred at ONR resulted from an open source, process redesign effort that was supported with Web-based information system technology for E-Government that could interpret and act on the process source representations [Scacchi 2001]. These results serve as the motivation and foundation for exploring how concepts and practices for E-Government and open source system development can be explicitly combined to create an evolving and continuously improving approach to open acquisition.

The Research Agenda for Open Acquisition

This concept of open acquisition raises a number of important new questions for research and further investigation. For example, would open acquisition allow for the establishment of community Web portals or other open test-beds where alternative system acquisition processes or practices might be (re)designed, prototyped, and evaluated via collaborative experimentation and engagement [Scacchi and Boehm 1998, Scacchi 2001]? Would open acquisition products (artifacts), processes, and IT support environments enable more complete assessment of the financial and infrastructural costs/benefits of new legislation that is created and imposed, but otherwise be unfunded? Thus, the overall purpose of this paper was to introduce, motivate, describe, and provide examples of the concept and enabling infrastructure for open acquisition of complex software-intensive systems. This purpose in turns serves as a point of departure for identifying some of the central topics that merit further research in order to advance the deployment, and practice of open acquisition.

The research described in this article builds on and further refines a promising new approach to the acquisition of large and complex software-intensive systems. This approach is called virtual system acquisition. However, the technology roadmap and five-year plan for research into virtual system acquisition [Scacchi and Boehm 1998], and the spiral model on which it is based [Boehm and Hansen 2001], did not anticipate the potential of open source system development, nor open acquisition processes and open process support environments. Nonetheless, the roadmap as conceived and published in 1998 remains viable and timely. Thus progress along the R&D paths outlined in the roadmap remains on course, and the research agenda for virtual system acquisition remains in tact, though revised to accommodate open acquisition concepts and techniques.

Open acquisition represents a combination of recent advances in virtual systems acquisition research, E-Government and open source software development concepts to create a new capability for open acquisition operations and processes. However, virtual systems acquisition and open acquisition are not incompatible; quite the contrary. Virtual systems acquisition and open acquisition are more a hand-in-glove relationship. Virtual systems acquisition is based on modeling and simulating the emerging architectures of software-intensive systems, and of the acquisition processes through which they are acquired, engineered, and deployed.

Open acquisition points to the need and benefit for making the models and simulations for virtual systems acquisition open source. Open source processes can enable the continuous sharing, review, and improvement of both system (product) and (acquisition) process by the community of practitioners, developers, end-users, and former participants involved in major system acquisition efforts. These models of the target systems and processes arising from open acquisition can be openly developed, shared, and improved using web-based process support environments. These Web-based environments then provide convenient access, while enabling rapid communication and timely update of system and process models and simulations.

Thus, the remaining agenda for research into virtual systems acquisition points to the need for exploratory development, prototyping, and experimental integration of an open, Web-based virtual systems acquisition environment with commercial-off-the-shelf software development and project management environments. These steps are in line with, and represent the final stages of, the five-year plan for research originally outlined by Scacchi and Boehm [1998]. That plan accounts for the contributions of government, industrial, and academic experts about how to best improve the acquisition of complex, software-intensive systems, as perceived in the mid-late 1990's. Beyond this, open acquisition also enables new research efforts to be focused on exploration of open government, as an extension and generalization of open acquisition, and on the prototyping of a new Web-based test-bed[2] that enables large-scale modeling, simulation, and related experimentation with complex systems and complex process architectures. As such, open acquisition represents a promising contribution to a new vision for how the acquisition of software-intensive systems might be dramatically improved.

References

BGen. (Ret.) F. Anderson, Smart Business 20/20: Preparing for the Future -- DAU Business Plan for 2000-2001, November 2000.

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

M. Bergquist and J. Ljungberg, The power of gifts: organizing social relationships in open source communities, *Info. Systems J.*, 11(4), 305-320, October 2001.

C.D. Brown, C. Grant, D. Kotchman, R. Reyenga, and T. Szanto, Building a Business Case for Modeling and Simulation, *Acquisition Review Quarterly*, 311-328, Fall 2000.

B. Boehm and W. Hansen, The Spiral Model as a Tool for Evolutionary Acquisition, *CrossTalk*, 2-11, May 2001.

M. Cusumano and D.B. Yoffie, Software Development on Internet Time, *Computer*, 32(10), 60-69, October 1999.

J.S. C. Choi and W. Scacchi, Modeling and Simulating Software Acquisition Process Architectures, *Journal Systems and Software*, 59(3), 343-354, 15 December 2001.

P. Clements. Software Product Lines: A New Paradigm for the New Century. *Crosstalk: The Journal of Defense Software Engineering*, 12(2), 20-22, February 1999.

[2] For example, a national computing grid of networked open acquisition service providers and community Web portals. This kind of Web-based information infrastructure collectively enables geographically and temporally dispersed communities of acquisition experts and practitioners to continuously improve system acquisition processes, practices, and work environments, all of which are in line with the vision for a new generation of acquisition workers [Anderson 2000].

A. Cockburn, *Agile Software Development*, Addison-Wesley Inc., Boston, MA, 2002.

DD(X) Information System, http://sc21.crane.navy.mil, 1997-2002.

(DG.O) Digital Government.Org, http://www.diggov.org, 2002.

(DGRC) Digitial Government Research Center, http://www.isi.edu/dgrc/, 2002.

DoD 5000 Acquisition Model, *Defense Acquisition Deskbook*, http://web2.deskbook.osd.mil/, 31 January 2001.

DoD 5000.2-R, *Mandatory Procedures for Major Defense Acquisition Programs (MDAPS) and Major Automated Information System Acquisition Programs*, Office of the Secretary of Defense, 10 June 2001.

C. Freericks, Open Source Standards on Software Process: A Practical Approach, *IEEE Communications Magazine*, 116-123, April 2001.

S. Glaseman, *Comparative Studies in Software Acquisition*, Lexington Books, D.C. Heath and Co., Boston, MA 1982.

A.J. Kim, *Community Building on the Web: Secret Strategies for Successful Online Communities*, Peachpit Press, Berkeley, CA, 2000.

F. Kuhl, R. Weatherly, J. Dahmann, *Creating Computer Simulation Systems: An Introduction to the High Level Architecture*, Prentice Hall, 1999.

A. MacCormack, R. Verganti, and M. Iansiti, Developing Products on Internet Time: The Anatomy of a Flexible Development Process, *Management Science*, 47(1), January 2001.

C. Meyer. *Fast Cycle Time: How to Align Purpose, Strategy, and Structure for Speed*, Free Press, New York, 1993.

B.C. Meyers and P. Oberndorf. *Managing Software Acquisition: Open Systems and COTS Products*, Addison-Wesley, Boston, MA, 2001.

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.

P. Mi and W. Scacchi. A Meta-Model for Formulating Knowledge-Based Models of Software Development. *Decision Support Systems*, 17(3), 313-330. 1996.

M.E. Nissen. Reengineering the RFP Process through Knowledge-Based Systems. *Acquisition Review Quarterly*, 4(1), 87-100, Winter 1997 .

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 1999.

J. Noll and W. Scacchi. Process-Oriented Hypertext for Organizational Computing*, J. Network and Computer Applications*, 24(1), 39-61, 2001.

R.C. Pavlicek, *Embracing Insanity: Open Source Software Development*, SAMS Press, Indianapolis, IN, 2000.

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*, 5(2/3), 183-195, 2000.

W. Scacchi. Redesigning Service Procurement for Internet-based Electronic Commerce: A Case Study, *J. Information Technology and Management,* 2(3), 313-334, 2001.

W. Scacchi, Software Development Practices in Open Software Development Communities, position paper presented at the *1st. Workshop on Open Source Software Engineering,* Toronto, Ontario, May 2001a.

W. Scacchi. Process Models in Software Engineering, in J. Marciniak (ed.), *Encyclopedia of Software Engineering*, 2nd. Edition, 993-1005, Wiley, 2001b.

W. Scacchi. Understanding the Social, Technological, and Policy Implications of Open Source Software Development, paper presented at the *NSF Workshop on Open Source Software*, Arlington, VA, January 2002a.

W. Scacchi. Understanding the Requirements for Developing Open Source Software Systems, *IEE Proceedings on Software*, 149(1), 24-39, 2002b.

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

W. Scacchi and P. Mi. Process Life Cycle Engineering: A Knowledge-Based Approach and Environment. *Intern. J. Intelligent Systems in Accounting, Finance, and Management*, 6(1), 83-107, 1997.

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

R.M. Schooff, Y.Y. Haimes, and C.G. Chittister. A Holistic Management Framework for Software Acquisition, *Acquisition Review Quarterly*, Winter 1997.

S.Sharman, V. Sugurmaran, and B. Rajagopalan, A Framework for Creating Hybrid-Open Source Software Communities, *Info. Systems J.,* 12(1), 7-25, 2002.

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

J. Steyaert, (Deputy associate administrator, Office of Information Technology, GSA), View From the Top: What Government is Doing, *Solution Series Conf. On E-Government*, 24 April 2001. Webcast version at http://www.gcn.com/webcast/070201gcn.html

R.G. Struth, Systems Engineering and the Joint Strike Fighter: The Flagship Program for Acquisition Reform, *Acquisition Review Quarterly*, 221-232, Summer 2000.

D. Truex, R. Baskerville, and H. Klein, Growing Systems in an Emergent Organization, *Communications ACM*, **42**(8), 117-123, 1999.

J. Verville and A. Halingten. *Acquiring Enterprise Software: Beating Vendors at Their Own Game,* Prentice-Hall, Upper Saddle River, NJ, 2001.