An Environment for Research in Software Systems Acquisition

Walt Scacchi[1], A.Valente[2], J. Noll[3] and J.S. Choi[4]
[1]Institute for Software Research, University of California at Irvine,
[2]Fastv.com, Los Angeles, CA
[3]University of Colorado at Denver
[4]California State University at Fullerton

Abstract:

In this paper, we present initial results from basic research and exploratory studies in the area of software systems acquisition. This research sought to design a web-based, computer-supported work environment that facilitates research and development in the area of software systems acquisition. This environment supports the capture, representation, and operationalization of various forms of knowledge associated with a new vision for virtual system acquisition, called VISTA. The schemes for organizing and managing knowledge rooted in software feasibility heuristics and informal/formal models of software acquisition and systems engineering processes are called knowledge webs. Accordingly, the environment that provides the mechanisms for capturing, representing, interlinking and operationalizing access to these knowledge webs is called a knowledge web management system (KWMS). Thus, the environment that is the focus of this research effort is designed to provide a KWMS capability that provides access to an incrementally evolving knowledge web for software acquisition research and practice in line with the VISTA vision. This environment for software acquisition web management is called SAWMAN.

Background

The acquisition of major software-intensive systems is often problematic. Recent reports from the US General Accounting Office (GAO 1997) and others (e.g., Holland 1998, Nissen, Snider and Lamm 1998) describe a number of problems with the way complex systems are acquired. We in the acquisition research community are at a time when there is substantial opportunity to rethink how the acquisition of software-intensive systems should occur to address the recurring problems. At the same time, we should pursue new opportunities to re-engineer the systems acquisition process in order to realize operational savings, efficiencies, increased satisfaction, and continuous improvement. Similarly, we should provide a strategy for managing the transition to these re-engineered system acquisition processes, as they can represent a radical departure from current practices. Subsequently, we should engage in research that explores how these opportunities can be supported through use of advanced information processing tools, techniques and concepts.

The long-term objective of this line of research is to make the acquisition of software-intensive systems more agile and adaptive. Our effort constitutes an ambitious departure from the status quo, rather than an incremental extension of current best practices (ARO 1999). In pursuing this objective, we need to explore the development of new generation of information technologies and associated processes that can support a new vision of what software systems acquisition can become (cf. Boehm and Scacchi, 1996, SA-CMM 2000, Scacchi and Boehm 1998, Schoof, Haimes and Chittister 1997, SPMN 1999, STSC

1996). This emerging capability is called *virtual system acquisition,* or simply VISTA (Scacchi and Boehm 1998).

The VISTA strategy provides a road map for software acquisition R&D (Boehm and Scacchi 1996, Scacchi and Boehm 1998). This road map appears top to bottom, left to right, in Table 1. It lays out the research, technology, and usage needed to support the acquisition of large software-intensive systems.

			-- *Technology Maturity* --		
			Research	**Technology**	**Acquisition Usage**
Software/ System Life Cycle Stages	**Concept Definition**		**Software Feasibility Heuristics**	*VISTA-1:* **Top-Level Feasibility Advisor, Parametric Models**	**Concept Feasibility Determination**
	Software Architecture Definition		**Architecture representation and analysis M&S, Advanced cost/schedule/quality M&S.**	*VISTA-2:* **Models and Simulations of Subsystems and Elements**	**Architecture Feasibility Determination**
	Spiral Development		**Integration into commercial Software Development Environments**	*VISTA-3:* **Hybrid Measurement, Modeling and Simulation Environment**	**Virtual System Acquisition**

Table 1: VISTA Research, Technology, and Usage Context (Scacchi and Boehm 1998)

In this paper, we address the Concept Definition stage of the VISTA strategy. Supporting the VISTA vision for research in the acquisition of software systems requires the ability to collect, represent, and organize access to online case studies, heuristics and formal process models within a Web-based information systems environment (Boehm and Scacchi 1996, Scacchi and Boehm 1998). While some of this knowledge is specific to particular acquisition programs, there is a substantial body of software acquisition heuristics available from a variety of published sources (e.g., Recthin 1991, SA-CMM 2000, Schoff, Haimes and Chittister 1997, STSC 1996, SPMN 1999). Initial studies indicate the software acquisition heuristics and other "best practices" can be formalized as inference rules that can be applied to analyze software acquisition processes that are modeled in a suitable computational form (cf. Nissen 1997,1998, Noll and Scacchi 1999b, Valente and Scacchi 1999).

We found the kinds of heuristics noted above are not generally represented nor accessible in an information systems environment that would support their browsing, query, systematic classification, evolutionary update, wide-area distribution, application and evaluation during system acquisition. Furthermore, from a classification standpoint, there are different classes of software acquisition or feasibility heuristics that may apply to assessing the architecture and engineering of complex software systems, in contrast to those that apply to acquisition processes, systems engineering processes or project management techniques. As such, we found there are three challenges that had to be addressed in order to provide support for the analysis of software system acquisition using software feasibility heuristics.

The first challenge to be addressed in re-tooling system acquisition is the need for an information systems environment that can support the capture, representation, and operationalization of various forms of software feasibility heuristics (Boehm and Scacchi 1996, Scacchi and Boehm 1998). More specifically, a Web-based environment for capturing, representing, applying and evolving software feasibility heuristics is needed (cf. Fielding, *et al.* 1998, Noll and Scacchi 2000).

The second challenge is to design a conceptual framework that can capture, represent and operationalize informal and formal models of different kinds of as-is, to-be, or here-to-there processes for software systems acquisition (Scacchi 2000).

The third challenge problem addresses how different forms of acquisition knowledge and redesigned acquisition processes can be represented, interlinked, evaluated and managed over the Web (Valente and Scacchi 1999, Choi and Scacchi 2000, Noll and Scacchi 2000).

Addressing the three challenge problems required both basic and exploratory research studies. The basic research problem entails how to develop a common solution to the three challenges for how to support a new generation of research in software systems acquisition. Developing and exploring possible solutions to this basic research problem would benefit from an exploratory study of new concepts, techniques and tools for acquiring, representing and operationalizing knowledge webs for software systems acquisition. Subsequently, our research evaluated, extended, and refined the concept of *knowledge web* and *knowledge web management system* (KWMS) in ways that could be applied to the domain of software systems acquisition (cf. Valente and Scacchi 1999).

A knowledge web provides a Web-based knowledge management capability and repository for representing various forms of knowledge in a domain like software systems acquisition. A KWMS is a Web-based information systems environment for managing the capture, representation and operationalization of knowledge webs that are accessible over the Internet. A KWMS provides access to the various knowledge management mechanisms needed to operationalize, reason with, and incrementally update a knowledge web.

The research effort in this study focused on the design of a KWMS environment for modeling and analyzing feasibility heuristics for software acquisition processes and software system architectures. The resulting infrastructure consists of a process modeling and knowledge representation capability, called *software acquisition knowledge webs* (cf. Valente and Scacchi 1999), together with the design of an information systems environment, called the *Software Acquisition Web MANagement system,* or simply SAWMAN (Choi and Scacchi 2000). SAWMAN in turn serves as the platform that will be combined with the Software Architecture Definition component of the VISTA road map in Table 1. This component is identified with the label, "architecture representation and analysis M&S", which is a shorthand contraction for the representation and analysis of software architecture models and simulations that can support software system acquisition. This effort is now in progress. Nonetheless, at this point we turn to describe our approach to the modeling, analysis and simulation of software acquisition processes and other heuristic knowledge associated with software acquisition best practices.

4. Approach and Results

There is a growing body of studies and techniques that address the modeling, analysis and simulation of software development and complex business processes. Yet none of the extant studies address the subject of software systems acquisition as their primary focus. However, following the VISTA strategy, software system acquisition becomes a motivating factor in practical applications of acquisition process modeling, redesign, analysis and simulation. As such, how can modeling, analysis and simulation of software processes be employed to directly support software systems acquisition?

Modeling acquisition processes and related knowledge

Process redesign heuristics are often independent of application domain and therefore applicable to a large set of processes (Bashein, Markus and Riley 1994, Caron, Jarvenpaa and Stoddard 1994). Alternatively, redesign heuristics may be domain-specific, thus applicable to specific processes in particular settings. In examining how heuristics for software acquisition are to be applied, we can learn the circumstances in which different types of heuristics or practices are most effective or least effective (cf. Software Programs Managers Network 1999). Similarly, we can learn which process redesigns are considered most effective and desirable in the view of the participants working in the redesigned process, and what techniques should be employed to facilitate process transformation and change management [Kettinger and Grover 1995, Scacchi and Noll 1997, Valente and Scacchi 1999]. Therefore, both domain-independent and domain-specific heuristics are of interest, as are techniques for determining how to transform software acquisition processes and the organizational settings in which they are to be applied.

Knowledge about best practices or new strategies for software acquisition is often cast as heuristics derived from results of empirical or theoretical studies. These results may then be coded as production rules for use in a rule-based or pattern-directed inference system (Nissen 1997), or as tuples (i.e., records of relation attribute instance values) that can be stored in a relational database (Kuh, Suh, and Tecuci 1996). These mechanisms can then be integrated with other tools for software process engineering (Brownlie, *et al*. 1997, Scacchi and Mi 1997). Nonetheless, these alternative representation mechanisms do not focus on what needs to be modeled, which is the focus here.

From a modeling standpoint, there is need to potentially model many kinds and forms of software acquisition process knowledge. These include (a) the acquisition process (or processes) to be redesigned in its legacy "as-is" form before redesign, (b) the redesign heuristics (or transformations) to be applied, (c) the "to-be" process resulting from redesign, and (d) the empirical sources (e.g., narrative case studies) from which the heuristics were derived. Furthermore, we might also choose to model (e) the sequence of steps (or the "here-to-there" process) through which different redesign heuristics were applied to progressively transform the as-is process into its to-be outcome. Modeling the processes identified in (a), (c) and (e) is already within the realm of process modeling and simulation capabilities. However, (b) and (d) pose challenges not previously addressed by software process modeling technologies. Furthermore, (b) and (d) must be interrelated or interlinked to the process models of (a), (c) and (e) to be of greatest value for external validation, traceability, and incremental evolution purposes (Valente and Scacchi 1999, Zelkowitz and Wallace 1998). Finally, software process modeling will play a role in (f) facilitating the continuing evolution and refinement of the software acquisition knowledge web.

Modeling Approach and Results

In developing models of processes for SPR, we used two process modeling notations (one a subset of the other) and two modeling tools. The modeling notations and tools are described in turn.

First, we used a process modeling language called PML to develop basic models of acquisition processes (Noll and Scacchi 2000, Scacchi and Noll 1997). PML allows processes to be modeled in a hierarchical manner (i.e., processes consist sub-processes, sub-sub-processes, etc. and actions), and allows process workflow to be modeled as a sequence, conditional choice, iteration, or concurrent set of actions. PML also associates an agent who performs an action together with the tools they employ to transform required resources into intermediate or final products. In this regard, PML is a process modeling notation that conforms to the process meta-model for software engineering and business processes that we had previously developed (Mi and Scacchi 1996). A sample of PML code that describes one software acquisition (sub-)process for "proposal submission" follows in Figure 1. This process entails a sequence of three actions, submitting a proposal by a principal investigator, submitting a proposal budget, and submitting required certifications. The process script indicates that the tools used in this process are Web-based data entry forms that allow a specified file to be electronically submitted with each of these project actions. PML also allows for other application programs (e.g., helper applications, Java servlets) to be employed in process scripts.

```
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>\
        <br>Proposal title: <input name='title' type='string' size=50>\
        <br>Submitting Institution: <input name='institution' type='string' size=25>\
        <br>Principle 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'>"
        }
}
}
```

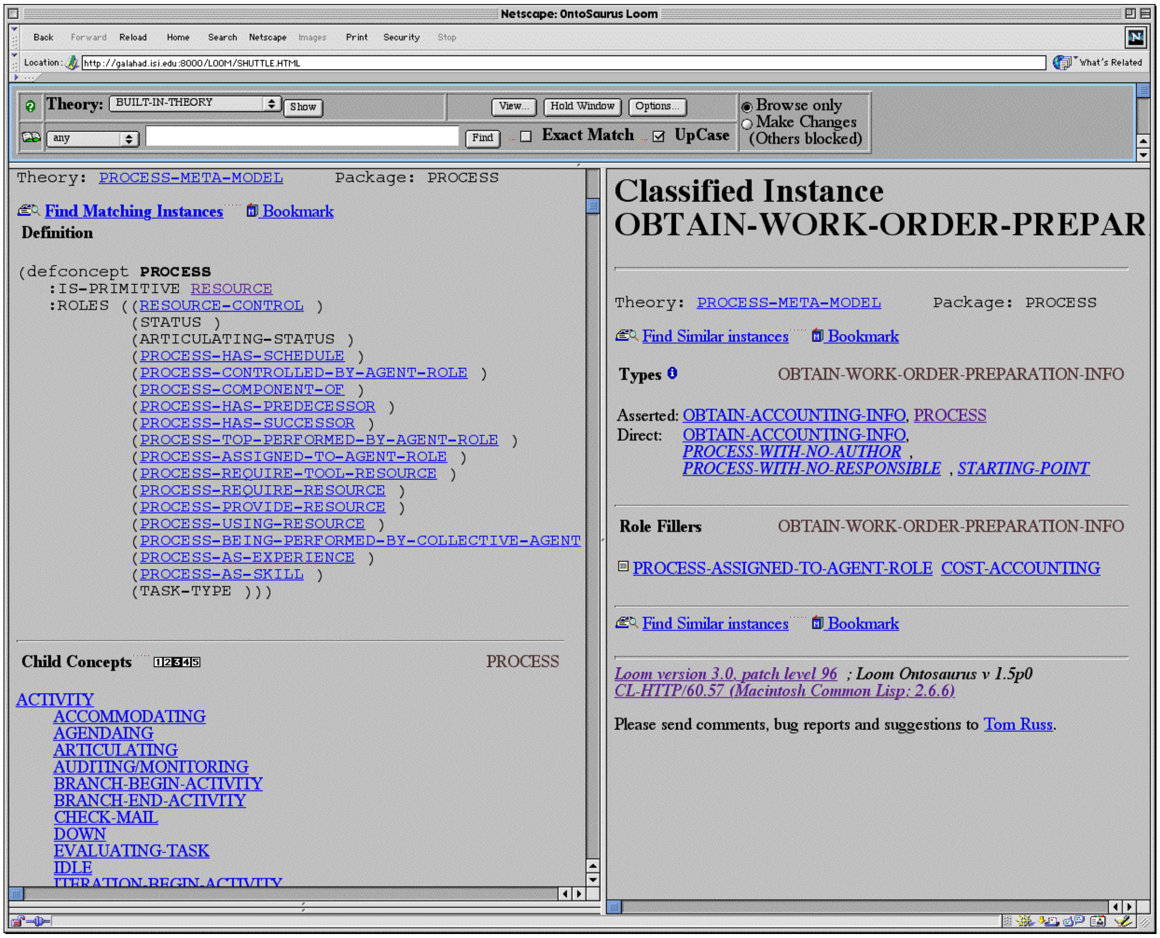
Figure 1: An excerpt from a proposal submission process modeled in PML (Noll and Scacchi 2000).

Thus, with PML descriptions of software acquisition processes we can rapidly create and interactively navigate/browse the "look and feel" of processes that can be accessed and evaluated over the Web (Noll and Scacchi 2000).

Next, in order to represent software acquisition process knowledge more formally and to reason with it, we choose to use the Loom knowledge representation system as our first process modeling tool (MacGregor and Bates 1995). Loom is a mature language and environment for constructing ontologies and intelligent systems that can be accessed over the Web (Valente, Russ, *et al.* 1999). By using Loom to re-implement the Articulator process meta-model ontology (Mi and Scacchi 1990, Mi and Scacchi 1996), formal models of software acquisition or related business processes, classification taxonomies and process redesign heuristics can be represented and manipulated. In turn, process knowledge can be analyzed, queried, and browsed, while relevant redesign alternatives for processes can be identified and linked to source materials on the Web. Nonetheless, Loom does impose a discipline for formally representing declarative knowledge structures in terms of concepts (object or pattern types), relations (link types that associate concept) and instances (concept, link, attribute values).

Loom's *deductive classifier* utilizes forward-chaining, semantic unification and object-oriented truth maintenance technologies. This capability enables Loom to compile the declarative knowledge into a semantic network representation designed to efficiently support on-line deductive query processing (MacGregor and Bates 1995, Valente *et al.* 1999). Further, Loom's classifier can be used to taxonomically classify and update a process redesign knowledge base as new process redesign cases, lessons learned, or best practices are entered and formally modeled. This in turn enables the software acquisition knowledge web to evolve with automated support (Valente and Scacchi 1999).

Finally, in order to support the visualization of the knowledge bases and process models that have been constructed, a Web browser interface to the Loom system is used (Valente *et al*. 1999). Ontosaurus (1999) is a client-side tool for accessing a Loom server loaded with one or more knowledge bases. It supports queries to Loom and produces Web pages describing several aspects of a knowledge base, including hypertext links to materials on the Web. It is also able to provide simple facilities for editing the contents of knowledge bases. Figure 2 shows a browser window accessing Ontosaurus. The display consists of three window panels; Toolbar (top), Reference (left side) and Content (right side). The Toolbar panel consists of buttons to perform various operations such as *select domain theory*, *display theory, save updates*, etc. The Reference and Content panels are designed to display contents of a selected ontology. Links in both panels display their contents in the Content window. This facilitates exploring various links associated with a word or concept in the Reference window without the need to continuously go back and forth. The bookmark window holds user-selected links for Web pages to Ontosaurus pages, and is managed by the buttons in the bottom of the bookmark window.



Location: http://galahad.isi.edu:8000/LOOM/SHUTTLE.HTML

Theory: BUILT-IN-THEORY

any

□ Exact Match ☑ UpCase

● Browse only
○ Make Changes
(Others blocked)

```
Theory: PROCESS-META-MODEL      Package: PROCESS
```

Find Matching Instances Bookmark

Definition

```
(defconcept PROCESS
   :IS-PRIMITIVE RESOURCE
   :ROLES ((RESOURCE-CONTROL )
           (STATUS )
           (ARTICULATING-STATUS )
           (PROCESS-HAS-SCHEDULE )
           (PROCESS-CONTROLLED-BY-AGENT-ROLE )
           (PROCESS-COMPONENT-OF )
           (PROCESS-HAS-PREDECESSOR )
           (PROCESS-HAS-SUCCESSOR )
           (PROCESS-TOP-PERFORMED-BY-AGENT-ROLE )
           (PROCESS-ASSIGNED-TO-AGENT-ROLE )
           (PROCESS-REQUIRE-TOOL-RESOURCE )
           (PROCESS-REQUIRE-RESOURCE )
           (PROCESS-PROVIDE-RESOURCE )
           (PROCESS-USING-RESOURCE )
           (PROCESS-BEING-PERFORMED-BY-COLLECTIVE-AGENT
           (PROCESS-AS-EXPERIENCE )
           (PROCESS-AS-SKILL )
           (TASK-TYPE )))
```

Child Concepts 12345 PROCESS

ACTIVITY
    ACCOMMODATING
    AGENDAING
    ARTICULATING
    AUDITING/MONITORING
    BRANCH-BEGIN-ACTIVITY
    BRANCH-END-ACTIVITY
    CHECK-MAIL
    DOWN
    EVALUATING-TASK
    IDLE
    ITERATION-BEGIN-ACTIVITY

Classified Instance OBTAIN-WORK-ORDER-PREPAR

```
Theory: PROCESS-META-MODEL      Package: PROCESS
```

Find Similar instances Bookmark

Types OBTAIN-WORK-ORDER-PREPARATION-INFO

Asserted: OBTAIN-ACCOUNTING-INFO, PROCESS
Direct: OBTAIN-ACCOUNTING-INFO, *PROCESS-WITH-NO-AUTHOR* , *PROCESS-WITH-NO-RESPONSIBLE* , *STARTING-POINT*

Role Fillers OBTAIN-WORK-ORDER-PREPARATION-INFO

PROCESS-ASSIGNED-TO-AGENT-ROLE COST-ACCOUNTING

Find Similar instances Bookmark

Loom version 3.0, patch level 96 ; Loom Ontosaurus v 1.5p0
CL-HTTP/60.57 (Macintosh Common Lisp: 2.6.6)

Please send comments, bug reports and suggestions to Tom Russ.

Figure 2: An Ontosaurus display of a formally modeled process

Loom and Ontosaurus were used to prototype a knowledge-based system that can represent and diagnostically evaluate software acquisition process models, redesign heuristics and taxonomies, as well as manage hyperlinks to materials accessible on the Web. The system employs the Articulator ontology of software and business processes (Mi and Scacchi 1990, 1996) that are expressed as concepts, attributes, relations and values in Loom. Loom provides a semantic network framework based on description logics. Nodes (objects) in a Loom representation define *concepts* that have roles or slots to specify their attributes. A key feature of description logic representations is that the semantics of the representation language are very precisely specified. This precise specification makes it possible for the classifier to determine whether one concept *subsumes* another based solely on the formal definitions of the two concepts. The classifier is an important tool for evolving ontologies because it can be used to automatically organize a set of Loom concepts into a classification hierarchy or taxonomy based solely on their definitions. This capability is particularly important as the ontology becomes large, since the classifier will find subsumption relations that people might overlook, as well as modeling errors that could make the knowledge base inconsistent.

Overall, 30 process redesign heuristics have been identified and classified. Six taxonomies were also identified for grouping and organizing access to the process redesign materials found on the Web. These taxonomies classify and index the cases according to:

- *Generic type of organization or application domain for process redesign:* financial, manufacturing, procurement and acquisition, research, software development, etc.
- *As-is "problems" with existing process:* off-line information processing, workflow delays, lack of information sharing, etc.
- *To-be "solutions" (goals) sought for redesigned process:* automate off-line information processing tasks, streamline workflow, use email and databases to share information, etc.
- *Use of intranet, extranet or Web-based process redesign solutions:* build intranet portal for project staff information, store version-controlled software development

objects on Web server, use HTML forms for data entry and validation process steps, etc.

- *How-to guidelines or lessons learned:* explicit techniques or steps for how to understand and model the as-is process, identify process redesign alternatives, involve process users in selecting redesign alternatives, etc.
- *Generic process redesign heuristics:* parallelize sequence of mutually exclusive tasks, unfold multi-stage review/approval loops, disintermediate or flatten project management structures, move process or data quality validation checks to the beginning, logically centralize information that can be shared rather than routed, etc.

In turn, each of these taxonomies could be represented as hierarchically nested indices of Web links to the corresponding cases. Navigation through nested indices that are organized and presented as a "portal" site is familiar to Web users. Typically, each taxonomy indexes 60-120 case studies or narrative reports out of the total of more than 200 that were found on the Web and studied (Valente and Scacchi 1999). This means that some cases could appear in one taxonomy but not another, while other cases might appear in more than one, and still others might not appear in any of these taxonomies if they were judged to not possess the minimal information needed for characterization and modeling.

Analyzing acquisition processes and related knowledge

Process models accounting for software systems development or engineering can be analyzed in a number of ways (Mi and Scacchi 1990, Scacchi and Mi 1997). These analyses are generally targeted to improving the quality of the process model, as well as to detect or prevent common errors and omissions that appear in large models (Scacchi 1999). Nonetheless, software acquisition poses additional challenges when analyzing process models.

First, it is necessary to analyze the consistency, completeness, traceability and correctness of multiple, interrelated process models (e.g., the as-is, here-to-there, and to-be models). This is somewhat analogous to what happens in a software development project when multiple notations (e.g., for system specification, architectural design, coding, and testing) are used, therefore requiring analysis across, as well as within, different software model notations (Choi and Scacchi 1998).

Second, it is necessary to account for software process resources throughout the acquisition redesign effort. For example, are resources that appear in an as-is process replicated, replaced, subsumed, or removed in the to-be process? Acquisition process redesign can change the flow of resources through a process, and thus we want to observe and measure these changes on process performance.

Last, one approach to determining when domain-independent process redesign heuristics can apply results from measuring structural attributes of the formal or internal representation (e.g., a semantic network or directed attributed graph) of a process as index for selecting process redesign heuristics (Nissen 1997, Nissen 1998, Scacchi and Noll 1997). Each of these challenges necessitates further description and refinement, as well as characterizing how they can interact in a simplifying or complicating manner.

Analysis Approach and Results

The first challenge in analyzing processes for redesign points to three types of problems that arise when processes (Choi and Scacchi 1998). First, *consistency problems* can appear. These denote conflicts in the specification of several properties of a given process. For example, a typical consistency problem is to have a process (e.g., for Proposal Submission) with the same name as one of its outputs (a proposal submission). This is something that occurs surprisingly often in practice, perhaps because the output is often the most visible characteristic of a process. Second, *completeness problems* cover incomplete specifications of the process. For instance, a typical completeness problem occurs when we specify a process with no inputs. Such a process can be considered a "miracle", since can produce outputs with no inputs. Similarly, a process that lacks outputs denotes a "black hole", where process inputs disappear without generating any output. Third, *traceability problems* are caused by incorrect specification of the origin of the model itself: its author, the agent(s) responsible for its authoring or update, and source materials from which it was derived. Subsequently, a process model that is consistent, complete and traceable can be said to be internally correct. Thus, solving these model-checking problems is required once process models are to be formalized.

One of the main reasons Loom is interesting as a formal process representation language is its capability to represent the abstract patterns of data that are the very definition of the problems discussed above. This capability is useful in producing simple and readable representations of model-checking analyses. For example, it is possible to define incomplete process model in plain English as "a process with no outputs", or as a

black-hole. This can be described in Loom as a process that provides exactly zero resources:

```
(defconcept black-hole
       :is (:and process
             (:exactly 0 process-provide-resource)))
```

Using the process modeling representations discussed above, the user describes a process model through Ontosaurus for processing by Loom. Then the system diagnoses the model provided. One of the advantages of using Loom is that once we define an instance, Loom automatically applies its classifier engine to find out what concepts match and apply to that instance. This offers a big advantage, since there is no need to specify an algorithm for the analysis process: instead, process models are analyzed automatically as a new model is specified. In addition, the classifier performs truth-maintenance. Therefore, if process model is updated to correct a problem found by the system, the classifier will immediately retract the assertion that the problem applies to that process. Thus, the classifier automates this activity for knowledge acquisition and update.

In order to provide a more direct interface to the diagnostic process analysis system, the Ontosaurus browser was extended to display two new types of pages. The first displays a description of process in a less Loom-specific way (e.g., for reporting purposes). The second displays a list of all problems found in the current process model we input. Figure 3 shows a screenshot of the Web page constructed by the server to describe the problems found in a model of a sample process.

The other two challenges for analyzing processes to support SPR can be addressed with a common capability that builds on the one just described. Since a formal representation of a software process model can be viewed as a semantic network or directed attributed graph, it is possible to measure the complexity attributes of the network/graph as a basis for graph transformation, simplification or optimization. This means that measures of a richly attributed "process flow chart" could reveal attributes such as the number of process steps, the length of sequential process segments, the degree of parallelism in process control flow, and others (Nissen 1997, Nissen 1998). Subsequently, redesign heuristics can be coded as patterns in the structure of a process representation. In turn, it then becomes possible to cast a process redesign heuristic as a pattern-directed inference rule or trigger whose antecedent stipulates a process complexity measure pattern, and whose consequent specifies the optimization transformation to be applied to the process representation (Nissen 1998). For example, when analyzing a software process model, if a sequence of process steps has linear flow and the inputs and outputs of the steps are mutual exclusive, then the process steps can be performed in parallel. Such a transformation reduces the time required to execute the redesigned process sequence.

Thus, process analysis for SPR can focus on measurement of attributes of a formal representation of a software process model that is internally correct.

Simulating acquisition processes and related knowledge navigation

Software process models can be simulated in a number of interesting and insightful ways using either knowledge-based, discrete-entity or system dynamics systems (ProSim 1999, Scacchi 1999). However, is there still need for another type of system to simulate processes performed by process users, and under their control?

When considering the role of simulation in supporting software process redesign a number of challenges arise. For example, how much of a performance improvement does an individual redesign heuristic realize? Will different process workload or throughput characterizations lead to corresponding variations in simulated performance in both as-is and to-be process models? How much of a performance improvement do multiple redesign heuristics realize, again when considered with different workloads or throughputs? Can simulation help reveal whether all transformations should be applied at once, or whether they should be realized through small incremental redesign improvements? As such, simulation in the context of software acquisition processes raises new and interesting problems requiring further investigation and experimentation.

As suggested earlier, there is need to simulate not only as-is and to-be processes but also the here-to-there transformation processes. Following from the results in the BPR research literature, transforming an as-is process into its to-be counterpart requires organizational change management considerations. The process users who should be enacting and controlling the transformation process can benefit from, and contribute to, the modeling and analysis of as-is processes (Scacchi and Mi 1997, Scacchi 1999). Similarly, users can recognize possible process pathologies when observing graphic animations of process simulations. However, the logic of the process simulation may not be transparent or easy to understand in terms that process users can readily comprehend.

Conventional approaches to process simulation may not be empowering to people who primarily enact software use processes (cf. Scacchi and Noll 1997). Instead, another option may be needed: one where process users can interactively traverse (i.e., simulate) a new to-be process, or the here-to-there process, via a computer-supported process walk-through or fly-through. In such a simulation, user roles are not simply modeled as objects or procedural functions; instead, users play their own roles in order to get a first-person view and feel for the new process. This is analogous to how "flight simulators" are used to help train aircraft pilots. In so doing, user participation may raise a shared awareness of which to-be alternatives make the most sense, and how the transformations needed to transition from the as-is to to-be process should be sequenced within the organizational setting. As such, simulation for SPR raises the need for new approaches and person-in-the-loop simulation environments.

Simulation Approach and Results

Questions pertaining to simulated process throughput performance or user workloads before/after process redesign can already be addressed by process simulation tools and techniques (ProSim 1999). No significant advances are required for this. Similarly, knowledge-based simulation capabilities can be employed to determine process performance improvements when multiple redesign heuristics are used to create alternative scenarios for software process enactment (cf. Caron, Jarvenpaa and Stoddard 1994, Scacchi 1999). Nonetheless, the challenge of how to support the transformation of as-is software processes into to-be redesigned alternatives is not addressed by existing process simulation approaches. Thus a new approach is required.

One key requirement for managing the organizational transformation to a redesigned software process is the engagement, motivation and empowerment of process users. The goal is to enable these users to participate and control process redesign efforts, as well as to select the process redesign alternatives for implementation and enactment. As the direct use of available simulation packages may present an obstacle to many process users, another means to support process management and change management is needed.

The approach we choose was to engage an acquisition process user community in a multi-site organizational setting and partner with them in redesigning their acquisition processes (Noll and Scacchi 2000, Scacchi and Noll 1997, Scacchi 2000). In particular, we developed, provided and demonstrated a prototype wide-area process walkthrough simulator that would enable the process redesign participants with a means to model, redesign and walkthrough processes that span multiple settings accessed over the Internet. With this environment, 10 process redesign heuristics were found applicable, while the process users chose 9 to implement (Scacchi and Noll 1997). In so doing, they eventually achieved a factor of 10X in cycle time reduction, and reductions in the number of process steps between 2-1 and 10-1 in the software use processes that were redesigned (Scacchi and Noll 1997). A process simulator played a central role in the redesign, demonstration and prototyping of these processes. How was this realized?

A Process Simulator Example

Process prototyping is a computer-supported technique for enabling software process models to be enacted without integrating the tools and artifacts required by the modeled process (Keller and Teufel 1998, Noll and Scacchi 2000, Scacchi and Mi 1997). It provides process users the ability to interactively observe and browse a process model, step by step, across all levels of process decomposition modeled, using a graphic user interface or Web browser. Creating a basic process execution run-time environment entails taking a prototyped process model and integrating the tools as helper applications that manipulate process task artifacts attached to manually or automatically generated Web/intranet hyperlink URLs (Noll and Scacchi 2000). Consider the following example of a sample sequence of acquisition process actions displayed in Figures 4-8.

This process can be modeled in terms of the process flow (precedence relations) and decomposition. It can also be attributed with user roles, tools and artifacts for each process step. Further, as suggested above, the directed attributed graph that constitutes the internal representation of the process can be viewed and browsed as a hyperlinked structure that can be navigated with a Web browser. The resulting capability enables process users to traverse or walkthrough the modeled process, task by task, according to the modeled process's control flow. This in turn can realize a Web-based or intranet-based process simulator system (Noll and Scacchi 1999, Scacchi and Noll 1997). In addition, the lower right frame in Figure 5 displays a record of the history of process task events that have transpired so far.

Using this process prototyping technology, we could work with process users to iteratively and incrementally model their as-is or to-be processes. Subsequently, modeled processes could then be interactively traversed using a Web browser interface to the resulting process simulator. Process users, independent of the time or location of their access to the process model, could then provide feedback, refinement or evaluation of what they saw in the Web-based process simulator.

Simulators are successful in helping process users to learn about the operational sequences of problem-solving tasks that constitute a software process (cf. Kettinger and Grover 1995, Scacchi and Mi 1997). Flight simulators have already demonstrated this same result many times over with flight operations process users (aircraft pilots). Process walkthrough simulators can identify potential patterns of software process user behavior, as well as potential performance or workflow bottlenecks in their use. This information in turn can help to identify parameter values for a discrete-event simulation of the same process. However, this has not yet been attempted.

Overall, discrete-event and knowledge-based simulation systems, together with process walkthrough simulators, constitute a learning, knowledge sharing, measurement and experimentation environment that can support and empower process users when redesigning their software processes (cf. Bashein, Markus and Riley 1994, Kettinger and Grover 1995). Therefore, these process simulation capabilities, together with other organizational change management techniques, should help minimize the risk of failure when redesigning software processes used in complex organizational settings.

Discussion

Given this introduction to the design of an environment for research in software systems acquisition, an explanation of how software process modeling, analysis and simulation fit it, and a demonstration of how it can operate through examples, there is still more work to be done. Thus, the purpose of this discussion is to identify some of the future needs that have become apparent from this investigation.

First, whether dealing with a legacy software process in a real-world setting, or when browsing a process description found on the Web, capturing, formalizing or otherwise modeling as-is processes is cumbersome. Part of the problem at hand is that many organizations lack explicit, well-defined or well-managed processes for the acquisition of software-intensive systems. Consequently, attention is often directed to focus only on creation of to-be alternatives, without establishing an as-is baseline. Without a baseline, acquisition process redesign efforts will increase their likelihood of failure (cf. Bashein, Markus and Riley 1994, Kettinger and Grover 1995). Thus, there is need for new tools and techniques for the rapid capture and codification of as-is software acquisition processes to facilitate their redesign.

Second, there is need for rapid generation of to-be and here-to-there processes and models. Acquisition process redesign heuristics, as well as the tools and techniques for acquiring and applying them appear to have significant face value. They can help to more rapidly produce to-be process alternatives. However, knowledge for how to construct or enact the here-to-there transformation process in a way that incorporates change management techniques and process management tools is an open problem. Further study is needed here.

Third, acquisition process redesign heuristics or transformation taxonomies may serve as a foundation for developing a theoretical framework for how to best represent such knowledge. Similarly, such a framework should stipulate what kinds of software process concepts, links and instances should be represented, modeled and analyzed to facilitate acquisition process redesign. Nonetheless, there is also a practical need to design and tailor process redesign taxonomies to specific software process domains and organizational settings. At this point, it is unclear whether heuristics for redesigning software use processes are equally applicable to software acquisition, development or evolution processes. The same can be said for any other combination of these types of software processes.

Fourth, in the preceding section, software tools that support the modeling, analysis and simulation of software processes for redesign were introduced. However, these tools were not developed from the start as a single integrated environment. Thus their capabilities can be demonstrated to help elucidate what is possible. But what is possible may not be practical for widespread deployment or production usage. Thus, there is a need for new environments that support the modeling, analysis and simulation of software processes that can be redesigned, life cycle engineered and continuously improved from knowledge automatically captured from the Web (cf. Brownlie et al. 1997, Maurer and Holz 1999, Scacchi and Mi 1997, Valente and Scacchi 1999).

Last, as highlighted in the results from research studies in business process redesign (e.g., Bashein, Markus and Riley 1994, Caron, Jarvenpaa and Stoddard 1994, Kettinger and

Grover 1995), and from first-hand experience (Scacchi and Noll 1997, Scacchi 1999), process users need to be involved in redesigning their own processes. Accordingly, the temptation to seek fully automated approaches to generating alternative to-be process designs from the analysis of an as-is process model must be mitigated. The concern here is to understand when or if fully automated software acquisition process redesign is desirable, and in what kinds of organizational settings. For example, there can be acquisition process situations where automated redesign may not be a suitable goal or outcome. This is in organizational settings where process users seek empowerment and involvement in redesigning and controlling their process structures and workflow. In settings such as these, the ability to access, search/query, select and evaluate possible process redesign alternatives through a through the system capabilities described above may be more desirable (cf. Scacchi and Noll 1997). Thus the ultimate purpose of support environment for acquisition process redesign may be in *supporting and empowering process users* to direct the redesign of their processes, rather than in automating acquisition processes.

Beyond this, one of the goals of SPR should be to minimize the risk of a failed SPR effort. Solutions that focus exclusively on technology-driven or technology-only approaches to SPR seemed doomed to fail. Thus, there remains a challenge for those that exclusively choose the technology path to SPR to effectively demonstrate how such an approach can succeed, in what kinds of organizational settings, and with what kinds of skilled process participants.

6. Conclusions

This paper addresses three research questions that identify and describe what software process redesign is, how software process modeling and simulation fit in, and what an approach to SPR might look like. SPR is proposed as a technique for achieving radical, order-of-magnitude improvements or reductions in software process attributes. SPR builds on empirical and theoretical results in the area of business process reengineering. However, it also builds on knowledge that can be gathered from the Web. Though the quality of such knowledge is more variable, the sources from which it is derived--experience reports, case studies, lessons learned, best practices and similar narratives--can be formally represented, hyperlinked and browsed during subsequent use or reuse. A central result from the knowledge collected so far is that SPR must combine its focus to both techniques for changing the organization where software processes are to be redesigned, as well as for identifying how software engineering and information technology-based process management tools and concepts can be applied.

Software process modeling, analysis, and simulation technology can be successfully employed to support SPR. In particular, knowledge-based tools, techniques and concepts appear to offer a promising avenue for exploration and application in this regard. However, new process modeling, analysis and simulation challenges have been also identified. These give rise to the need to investigate new tools and techniques for capturing, representing and utilizing new forms of process knowledge. Knowledge such as SPR heuristics can play a central role in rapidly identifying process redesign alternatives. Software process simulation techniques in particular may require computer-supported, person-driven process simulators, which enable process users to observe, walk-through or fly-through process redesign alternatives. Finally, software process modeling, analysis and simulation capabilities that support SPR activities may need to be deployed in ways that engage and facilitate the needs of users who share processes across multiple organizational settings, using mechanisms that can be deployed on the Web.

Last, we presented an approach to understanding, redesigning and evaluating software acquisition processes and related knowledge that utilizes Web-based tools for process modeling, analysis, and person-driven simulation. Initial experiences in using these tools, together with the business process reengineering and change management techniques they embody, indicates that the objective of order-of-magnitude reductions in acquisition process cycle time and process steps can be demonstrated and achieved in complex organizational settings (Scacchi 2000). Whether results such as these can be replicated in all classes of software processes--acquisition, development, usage, and evolution--remains the subject of further investigation. Similarly, other research problems have been identified for how or where advances in software process modeling and simulation can lead to further experimental studies and theoretical developments in the art and practice of software process redesign.

Supporting the VISTA vision for research in the acquisition of software systems requires the ability to collect, represent, and organize access to online case studies, heuristics and formal process models from a Web-based information systems environment (Boehm and Scacchi 1996, Scacchi and Boehm 1998). Similarly, other research studies in the area of acquisition, procurement, project management and organizational transformation will produce results and knowledge in the form of online studies, heuristics or formal process models (cf. Gebauer, Beam and Segev 1998, Hocevar and Oven 1998, Nissen 1997, Scacchi and Noll 1997, Scacchi 1998a,c). Thus, the exploratory aspect of our research was to evaluate, extend and refine the KWMS we prototyped and demonstrated by in a manner that provides a common solution to the three challenge problems identified above.

Conclusions

Overall, this research seeks to establish the conceptual foundation and baseline for a Web-based, computer-supported work environment that can support future research in the area of software systems acquisition as outlined in the VISTA reports (Boehm and Scacchi 1996, Scacchi and Boehm 1998). The proposed effort is a modest first step that leverages the results from prior research in the area of software acquisition, Web-based process modeling and engineering environments, software system engineering, business process redesign, procurement and Electronic Commerce. It therefore proposes to build on state of the art knowledge representation tools and techniques that have been developed by others for R&D studies in military and non-military domains (Valente, Russ, *et al*. 1999, Valente and Scacchi 1999).

Acknowledgements

The contributions of Dr. Andre Valente were made while he was a research scientist at the USC Information Sciences Institute in Marina Del Rey, CA. Early in 2000, he joined the firm Fastv.com, where he is currently director of knowledge operations. This research was supported by a grant N487650-27803 from the Defense Acquisition University. This research also drew on results from research funded by ONR through grant N00014-94-1-0889.

References

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

B.J. Bashein, M.L. Markus, and P. Riley 1994. Preconditions for BPR Success: And How to Prevent Failures. *Information Systems Management*, **11**(2):7-13.

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.

R.A. Brownlie, P.E. Brown, K. Culver-Lozo, and J.J. Striegel, J.J. Tools for Software Process Engineering, *Bell Labs Technical Journal*, **2**(1):130-143, 1997.

J. Caron, S.L. Jarvenpaa, and D.B. Stoddard. Business Reengineering at CIGNA Corporation: Experiences and Lessons Learned from the First Five Years. *MIS Quarterly*, **18**(3):233-250, 1994.

J.S. 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.

J.S. Choi and W. Scacchi. Modeling and Simulating Software Acquisition Process Architectures, *Workshop on Software Process Simulation and Modeling* (*ProSim 2000*), London, UK, July 2000.

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.

T.H. Davenport and L. Prusak. *Working Knowledge: How Organizations Manage What They Know*. Harvard Business School Press, Boston, MA, 1998.

DD21 Information System, http://sc21.crane.navy.mil, 1997-2000.

R. T. Fielding, E. J. Whitehead Jr., K. M. Anderson, G. A. Bolcer, P. Oriezy, R. N. Taylor. Support for the Virtual Enterprise: Web-based Development of Complex Information Products. *Communications ACM*, **41**(8): 84-92. August,1998.

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

Lt. Col. A.B. Garcia, Col. R.P. Gocke Jr., Col. N.P. Johnson Jr. *Virtual Prototyping: Concept to Production*, Defense Systems Management College Press, Fort Belvoir, March 1994.

J. Gebauer, C. Beam and A. Segev. Impact of the Internet on Procurement. *Acquisition Review Quarterly*, **5**(2):167-184, Spring 1998.

S.P Hocevar and W.E. Owen. Team-based Redesign as a Large-Scale Change. *Acquisition Review Quarterly*, **5**(2):147-166, Spring 1998.

L. Holland. The Weapons Acquisition Process: The Impediments to Radical Reform. *Acquisition Review Quarterly*, **5**(2):235-249, Spring 1998.

W.J. Kettinger, and V. Grover. Special Section: Toward a Theory of Business Process Change Management, *J. Management Information Systems*, **12**(1):9-30, 1995

S. Ku, Y.-H. Suh, and G. Tecuci. Building an Intelligent Business Process Reengineering System: A Case-Based Approach. *Intelligent Systems in Accounting, Finance, and Management*, **5**(1):25-39, 1996.

D.B. Leake (ed.). *Case-Based Reasoning: Experiences, Lesson and Future Directions,* AAAI/MIT Press, Menlo Park, CA, 1996.

MacGregor, R., and Bates, R. Inside the Loom description classifier. *SIGART Bulletin* **2**(3):88-92

P. Mi and W. Scacchi. A Knowledge-Based Environment for Modeling and Simulating Software Engineering Processes. *IEEE Trans. Knowledge and Data Engineering,* **2**(3):283-294, 1990.

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. Redesigning Reengineering through Measurement-Driven Inference. *MIS Quarterly*, **22**(4): 509-534, December, 1998.

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. Integrated Diverse Information Repositories: A Distributed Hypertext Approach, *Computer*, **24**(12):38-45, December 1991.

J. Noll and W. Scacchi. Specifying Process-Oriented Hypertext for Organizational Computing. *J. Networking and Computing Applications,* (to appear), 2000.

D.E. O'Leary. Enterprise Knowledge Management. *Computer*, **31**(3):54-61, 1998.

Ontosaurus Web Browser home page. http://www.isi.edu/isd/ontosaurus.html, 2000.

W. Rechtin. *System Architecting: Creating and Building Complex Systems*. Prentice-Hall, Englewood Cliffs, NJ. 1991.

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

W. Scacchi. Redesigning Service Procurement for Internet-based Electronic Commerce: A Case Study, *Journal of Information Technology and Management*, to appear, 2000.

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.

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

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

P.G. Selfridge and L.G. Terveen. Knowledge Management Tools for Business Process Support and Reengineering. *Intelligent Systems in Accounting, Finance and Management*, **5**:15-24, 1996.

(SPMN) Software Program Managers Network. *The Condensed Guide to Software Acquisition Best Practices*, 1999. 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. Dept. of the Air Force, June 1996. http://stsc.hill.af.mil/stscguid.asp

A. Valente, T. Russ, R. MacGregor, and W. Swartout. Building and (Re)Using an Ontology for Air Campaign Planning. *IEEE Intelligent Systems*, **14**(1):27-36, 1999.

A. Valente and W. Scacchi. Developing a Knowledge Web for Business Process Redesign. Presented at the 1999 Knowledge Acquisition Workshop, Banff, Canada, October 1999.