SOFTWARE PROCESS IMPROVEMENT AND PRACTICE
Softw. Process Improve. Pract. 2000; **5**: 183–195 (2000)

Understanding Software Process Redesign using Modeling, Analysis and Simulation

Research Section

Walt Scacchi*,†
Institute for Software Research and Information and Computer Science Department, University of California at Irvine, Irvine, CA 92697-3425 USA

Software process redesign (SPR) is concerned with the development and application of concepts, techniques and tools for dramatically improving or optimizing software processes. This paper introduces how software process modeling, analysis and simulation may be used to support software process redesign. This includes an approach to explicitly modeling process redesign knowledge, analyzing processes for redesign, and simulating processes before, during and after redesign. A discussion follows which identifies a number of topic areas that require further study in order to make SPR a subject of software process research and practice. Copyright © 2000 John Wiley & Sons Ltd

KEY WORDS: software process; process redesign; business process redesign; process modeling and simulation

1. OVERVIEW

Software process improvement (SPI) has traditionally been focused on addressing how to improve the capabilities of a software development organization through maturing and comparative benchmarking of its software processes. The Capability Maturity Model from the Software Engineering Institute is the most visible SPI initiative of its kind. However, the CMM is targeted to *incremental improvement* of existing software processes. The CMM top-most level, Optimization (level 5), characterizes those organizations whose software processes are incrementally improved and refined through monitoring, measurement and reflexive analysis of well-defined and well-managed processes. However, the CMM does not provide specific guidance for how to improve specific software development or software use processes, nor does it provide guidance for how to redesign legacy processes or how to design new processes. In addition, there is no maturity level that prescribes how to dramatically optimize software processes to achieve a 10× improvement in productivity or quality through *radical transformation* (Shelton 1999). Are radical transformations of software processes of the same kind as incremental evolutionary improvements? Probably not, though they appear to lie along a common dimension or metric that characterizes the scale or scope of process change that is sought. As such, *software process redesign* (SPR) merits investigation to determine whether and how it might lead to dramatic improvements in process efficiency or effectiveness.

*Correspondence to: Walt Scacchi, Institute for Software Research and Information and Computer Science Department, University of California at Irvine, Irvine, CA 92697-3425, USA
†E-mail: Wscacchi@ics.uci.edu
Contract/grant sponsor: Office of Naval Research; contract/grant number: N00014-94-1-0889
Contract/grant sponsor: Defense Acquisition University; contract/grant number: N487650-27803

The study presented in this paper describes how concepts, techniques and tools for software process modeling, analysis and simulation may be employed to support SPR studies. In particular, three research questions that explore and elaborate these topics can be identified as follows:

- What is software process redesign?
- How can modeling, analysis and simulation help in redesigning software processes?
- What approach is needed for acquiring, checking and applying the knowledge required to dramatically and continuously improve software processes?

Accordingly, in the sections that follow, each of these questions is addressed, elaborated and investigated in turn.

2. WHAT IS SOFTWARE PROCESS REDESIGN

SPR is concerned with identification, application and refinement of new ways to dramatically improve and transform software processes. Software processes of interest include not only those associated with software development, but also those for software system acquisition, use and evolution (Scacchi and Boehm 1998, Scacchi and Mi 1997, Scacchi and Noll 1997). Process redesign heuristics, and the source materials from which they are derived, serve as the main source of knowledge for SPR (see Scacchi and Noll 1997, Valente and Scacchi 1999). Process redesign heuristics can be independent of application domain and therefore applicable to a large set of processes (Bashein *et al*. 1994, Caron *et al*. 1994). Alternatively, redesign heuristics may be domain-specific, thus applicable to specific processes in particular settings. In examining how SPR heuristics are applied, we can learn the circumstances in which different types of heuristics or practices are most effective or least effective (see 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 SPR heuristics are of interest, as are techniques for determining how to transform the organizational settings in which they are to be applied.

Source materials represent on-line narrative reports from formal analyses, case studies, best practices, experience reports and lessons learned for how to dramatically improve the cycle time, defect prevention and cost effectiveness of various kinds of software/business processes. Increasingly, these materials can be found on the World-Wide Web as on-line publications or hypertext documents. For example, we can all use search engines on the Web to conduct a global search and information retrieval. Though we may find little matching a search for 'software process redesign', a search for 'process redesign' or 'process re-engineering' will return much more. Here we might find case studies, experience reports, best practices or lessons learned as narrative documents posted on academic, non-profit or commercial sites. Clearly, the quality of the 'knowledge' and results gleaned from sources on the Web may be more variable than those found in system research studies, but in searching for SPR heuristics, when potentially relevant source materials are found, hyperlinks can be used to designate the connection between the materials and the heuristics they instantiate. In turn, when a heuristic is a potential candidate for use in redesigning a software process, its source materials can be browsed and re-examined to help determine its similarity, relevancy, trajectory or outcome. Subsequently, as interest in SPR activities and outcomes grows, then more SPR case studies, experience reports, lessons learned, best practices, counter-examples and caveats may soon find their way onto the Web (Maurer and Holz 1999). Thus, the development of SPR heuristics or SPR knowledge repositories should be viewed as areas for further research, while their application should be integrated in continuous process improvement initiatives, rather than as topics that can be exhaustively analyzed with limited effort.

3. HOW CAN MODELING, ANALYSIS AND SIMULATION HELP SPR

There is a growing body of studies and techniques that address the modeling, analysis and simulation

of software processes (ProSim 1999), yet none of the extant studies addresses the subject of SPR as their primary focus. However, SPR is often implicit as a motivating factor in practical applications of software process modeling and simulation. As such, how can modeling, analysis and simulation of software processes be employed to directly support SPR?

3.1. Modeling Process Redesign Knowledge

As already noted, SPR knowledge 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 *et al*. 1996). These mechanisms can then be integrated with other tools for software process engineering (Brownlie *et al*. 1997, Scacchi and Mi 1997). However, 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 SPR knowledge. These include: (a) the process 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 SPR knowledge web.

3.2. Analyzing Processes for Redesign

Software process models 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). None the less, software process redesign 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.

Second, it is necessary to account for software process resources throughout the redesign effort. For example, are resources that appear in an as-is process replicated, replaced, subsumed or removed in the to-be process? SPR 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.

3.3. Simulating Processes Before, During and After Redesign

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 SPR 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 (see 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 walkthrough or flythrough. 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.

4. APPROACH AND RESULTS

Given the challenges identified in the previous section on how modeling, analysis and simulation can support SPR, this section presents the approach and initial results from an effort in each of these three areas.

4.1. Modeling Approach and Results

In developing models of processes for SPR, we used two tools. First, in order to represent SPR knowledge formally and reason with it, the Loom knowledge representation system was selected (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 *et al*. 1999). By using Loom to re-implement the Articulator process meta-model ontology (Mi and Scacchi 1990, 1996), formal models of software (or 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. None the less, Loom does impose a discipline for formally representing declarative knowledge structures in terms of concepts (object or pattern types), relations (link types that associate concepts) 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 the SPR knowledge base as new SPR cases are entered and formally modeled. This in turn enables the SPR knowledge web to evolve with automated support (Valente and Scacchi 1999).

Second, 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 1 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.

Loom and Ontosaurus were used to prototype a knowledge-based system that can represent and diagnose software 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

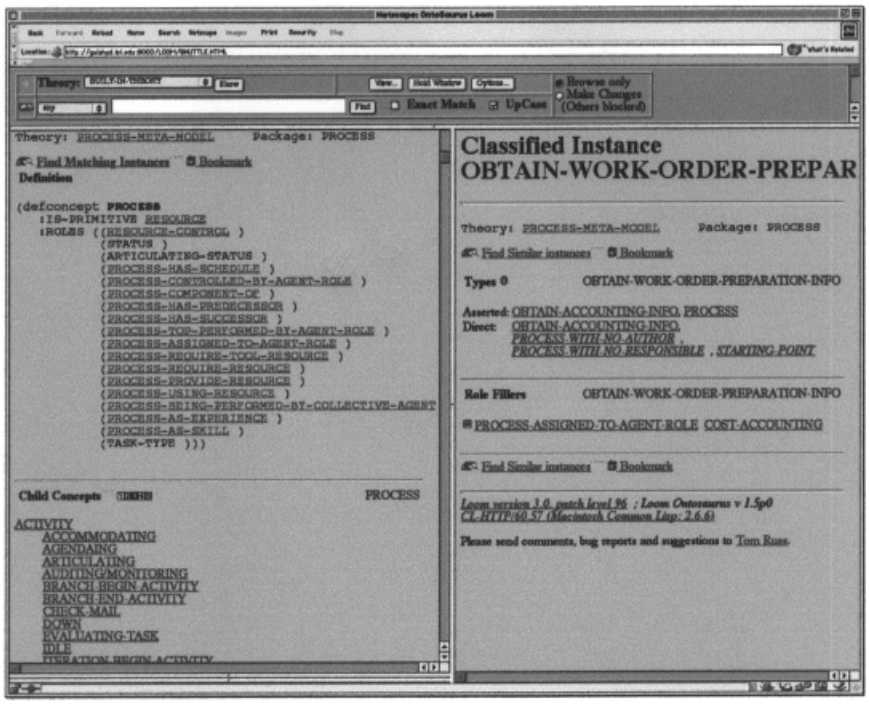

Figure 1. Ontosaurus display with Process concept definition loaded in the Reference window and a process redesign instance in the Contents window

errors that could make the knowledge base inconsistent.

Overall, 30 process redesign heuristics were 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, 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 e-mail 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.
- *SPR 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.
- *SPR 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.

4.2. Analysis Approach and Results

The first challenge in analyzing processes for redesign points to three types of problems that arise when processes. 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 Software Design) with the same name as one of its outputs (a software design). 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 it 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 a 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 2 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, 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 mutually 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. This is similar to how compilers perform code optimizations during compilation, after parsing and semantic analysis while prior to code generation (see Mak 1996).

4.3. 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 (see Caron *et al*. 1994, Scacchi 1999). None the less, 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 chose was to engage a process user community in a multi-site organizational setting and partner with them in redesigning their software use processes (Scacchi and Noll 1997). 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

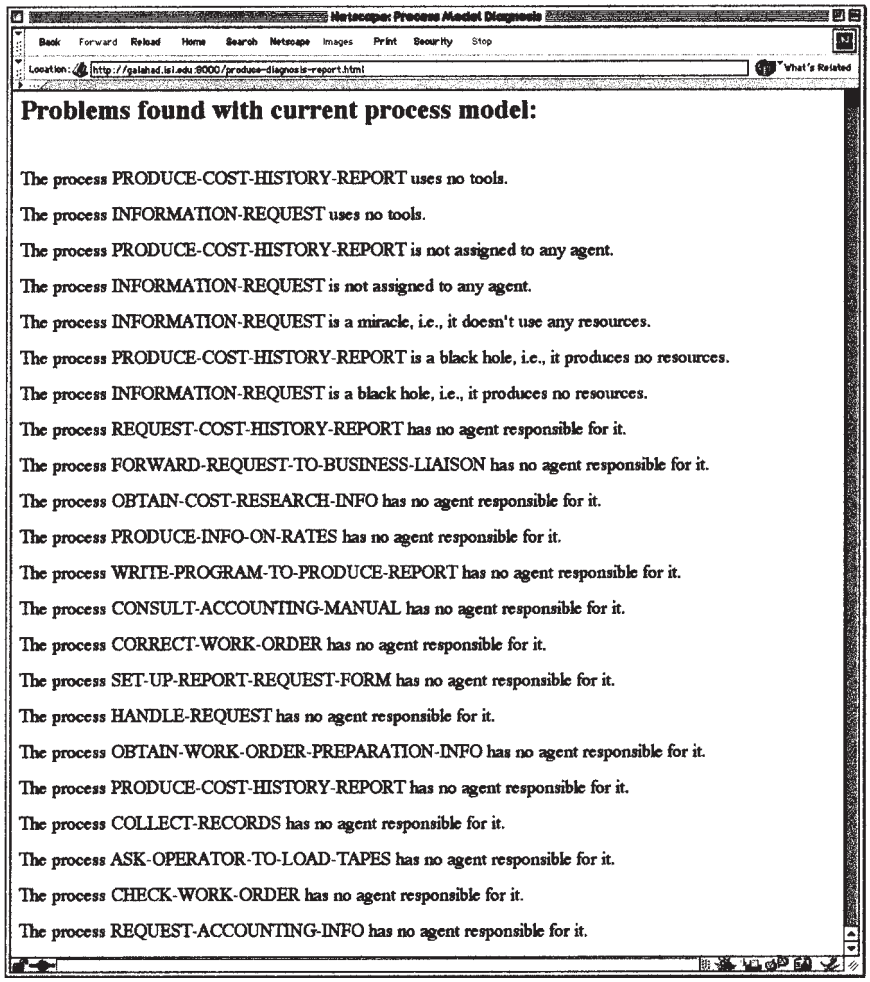

Figure 2. Generated report from Loom analysis of a process redesign case

(Scacchi and Noll 1997). In so doing, they eventually achieved a factor of 10× 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?

4.3.1. 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, 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 1999). Consider the following example of a simple software development process displayed in Figure 3 (Noll and Scacchi 1999).

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

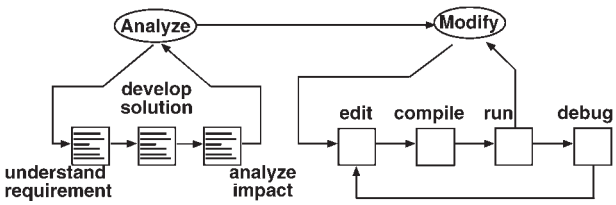

Figure 3. A simple software development process depicted as a directed graph

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). Figure 4 provides a view of a set of artifacts that might be associated with the process in Figure 3. Figure 5 provides a similar view of a selected task ('edit'), tool (the Emacs editor), and artifact (loaded in the Emacs edit buffer) associated with a user role as a 'developer' (not shown). 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 (see 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 (see Bashein *et al*. 1994, Kettinger and Grover 1995). Therefore, these process simulation

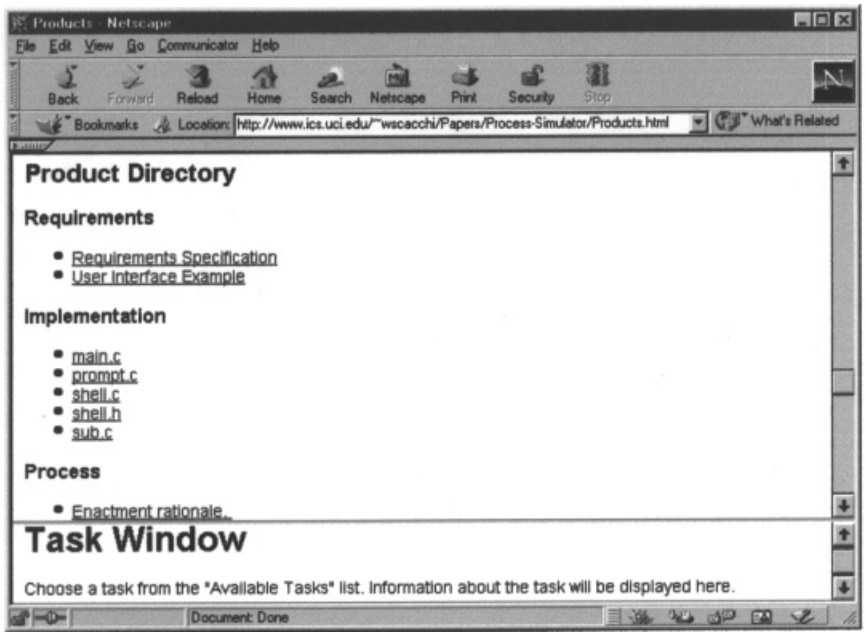

Figure 4. A set of artifacts associated with the software process in Figure 3

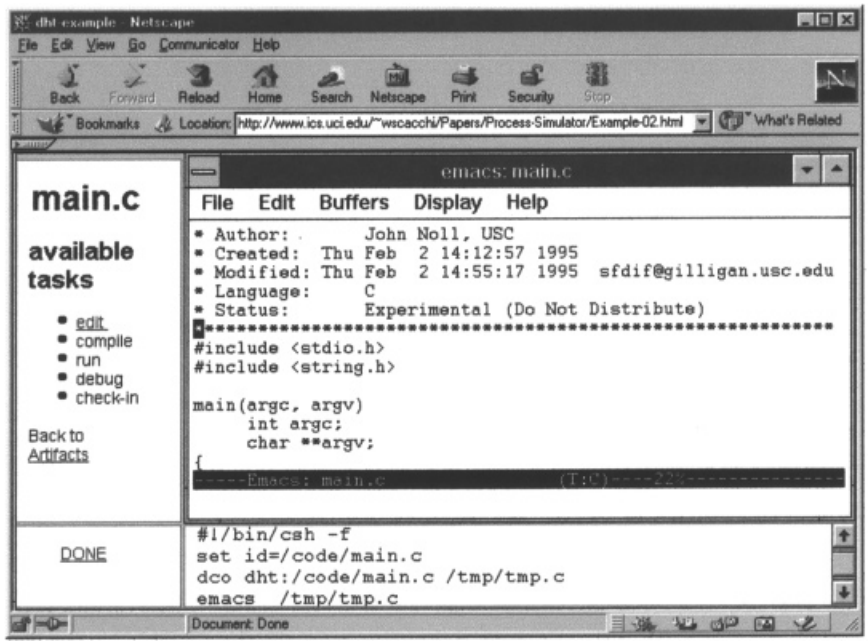

Figure 5. A software process enactment step presented in a process simulator

capabilities, together with other organizational change management techniques, should help minimize the risk of failure when redesigning software processes used in complex organizational settings.

5. DISCUSSION

Given the introduction to the subject of SPR, an explanation of what it is, 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 most organizations lack explicit, well-defined or well-managed processes as the starting point for an SPR effort. Consequently, attention is often directed to focus only on creation of to-be alternatives, without establishing an as-is baseline. Without a baseline, SPR efforts will increase their likelihood of failure (see Bashein *et al*. 1994, Kettinger and Grover 1995). Thus, there is need for new tools and techniques for the rapid capture and codification of as-is software processes to facilitate SPR.

Second, there is need for rapid generation of to-be and here-to-there processes and models. SPR 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, SPR heuristics or transformation taxonomies may serve as a foundation for developing a theoretical framework for how to best represent SPR knowledge. Similarly, such a framework should stipulate what kinds of software process concepts, links and instances should be represented, modeled and analyzed to facilitate SPR. None the less, there is also a practical need to design and tailor SPR 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, lifecycle engineered and continuously improved from knowledge automatically captured from the Web (see 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 (Bashein *et al*. 1994, Caron *et al*. 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 SPR is desirable, and in what kinds of organizational settings. For example, there can be SPR 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 the system capabilities described above may be more desirable (see Scacchi and Noll 1997). Thus the ultimate purpose of support environment for SPR may be in *supporting and empowering process users* to direct the redesign of their processes, rather than in automating SPR.

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 seem 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 re-engineering. 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 walkthrough or flythrough 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, an approach to SPR that utilizes Web-based tools for software process modeling, analysis, and person-driven simulation has been presented. Initial experiences in using these tools, together with the business process re-engineering and change management techniques they embody, indicates that the objective of order-of-magnitude reductions in software process cycle time and process steps can be demonstrated and achieved in complex organizational settings. 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.

ACKNOWLEDGEMENTS

The research described in this paper resulted from collaborations with the following people at the USC ATRIUM Laboratory. Dr Andre Valente, now at Fastv.com, developed the modeling and analysis system prototype with Loom and Ontosaurus displayed in Figures 1 and 2. Professor John Noll, now at the Computer Science Department, University of Colorado at Denver, developed the modeling and process simulation walkthrough shown in Figures 4 and 5. The process measurement technique and rule-based formulation used for automated process redesign analysis was first developed by Professor Mark Nissen, now at the Systems Management Department, Naval Postgraduate School, Monterey, CA. Finally, this research was supported by grants from the Office of Naval Research (N00014-94-1-0889) and the Defense Acquisition University (N487650-27803). All of these contributions are gratefully acknowledged.

REFERENCES

Bashein BJ, Markus ML, Riley P. 1994. Preconditions for BPR success: and how to prevent failures. *Information Systems Management* **11**(2): 7–13.

Brownlie RA, Brown PE, Culver-Lozo K, Striegel JJ. 1997. Tools for software process engineering. *Bell Labs Technical Journal* **2**(1): 130–143.

Caron JR, Jarvenpaa SL, Stoddard DB. 1994. Business Reengineering at CIGNA Corporation: experiences and lessons learned from the first five years. *MIS Quarterly* **18**(3): 233–250.

Keller G, Teufel T. 1998. *SAP R/3 Process-Oriented Implementation*. Addison-Wesley Longman Limited: Essex, England.

Kettinger WJ, Grover V. 1995. Special section: toward a theory of businesss process change management. *Journal of Management Information Systems* **12**(1): 9–30.

Ku S, Suh Y-H, Tecuci G. 1996. Building an intelligent business process reengineering system: a case-based approach. *Intelligent Systems in Accounting, Finance and Management* **5**(1): 25–39.

MacGregor R, Bates R. 1995. Inside the LOOM description classifier. *SIGART Bulletin* **2**(3): 88–92.

Mak R. 1996. *Writing Compilers and Interpreters*. Wiley: New York.

Maurer F, Holz H. 1999. Process-oriented knowledge management for learning software organizations. Twelfth Workshop on Knowledge Acquisition, Modeling and Management, Banff, Canada, http://sern/ucalgary.ca/KSI/KAW/KAW99/papers.html

Mi P, Scacchi W. 1990. A knowledge-based environment for modeling and simulating software engineering processes. *IEEE Transactions on Knowledge and Data Engineering* **2**(3): 283–294.

Mi P, Scacchi W. 1997. A meta-model for formulating knowledge-based models of software development. *Decision Support Systems* **17**(3): 313–330.

Nissen ME. 1997. Reengineering the RFP processs through knowledge-based systems. *Acquisition Review Quarterly* **4**(1): 87–100.

Nissen ME. 1998. Redesigning reengineering through measurement-driven inference. *MIS Quarterly* **22**(4): 509–534.

Noll J, Scacchi W. 1999. Supporting software development projects in virtual enterprises. *Journal of Digital Information* **1**(4).

Ontosaurus Web Browser. 1999. http://www.isi.edu/isd/ontosaurus.html.

ProSim. 1999. Selected Papers from the ProSim'99 Workshop, Special Issue on Software Process Simulation Modeling. *Journal of Systems and Software* **46**(2/3).

Scacchi W. 1999. Experiences with software process simulation and modeling. *Journal of Systems and Software* **46**(2): 183–192.

Scacchi W, Boehm BE. 1998. Virtual system acquisition: approach and transitions. *Acquisition Review Quarterly* **5**(2): 185–216.

Scacchi W, Mi P. 1997. Process life cycle engineering: a knowledge-based approach and environment. *Intelligent Systems in Accounting, Finance and Management* **6**: 83–107.

Scacchi W, Noll J. 1997. Proccess-driven intranets: life cycle support for process reengineering. *IEEE Internet Computing* **1**(5): 42–49.

Shelton R. 1999. The Long Game: Revolutionary Change through Radical Innovation. Prism, Third Quarter, Arthur D. Little: Cambridge, MA; 27–39. http://www.arthurdlittle.com/prism/prism.html.

Software Program Managers Networks. 1999. http://www.spmn.com.

Valente A, Russ T, MacGregor R, Swartout W. 1999. Building and (re)using an ontology for air campaign planning. *IEEE Intelligent Systems* **14**(1): 27–36.

Valente A, Scacchi W. 1999. Developing a knowledge web for business process redesign. IJCAI-99 Workshop on Workflow and Process Management, Stockholm, Sweden, August 1999.

Zelkowitz M, Wallace DR. 1998. Experimental models for validating technology. *Computer* **31**(5): 23–31.