

# Turning Development Standards Into Repositories of Experiences

**Scott Henninger<sup>1</sup>**

Department of Computer Science & Engineering  
University of Nebraska-Lincoln  
Lincoln, NE 68588-0115  
scotth@cse.unl.edu  
Phone: 1-402-472-8394  
FAX: 1-402-472-7767

## **Abstract**

*There has been a resurgence of interest in developing organization-wide Standard Development Methodologies (SDMs) for software development. It is well-known that creating an SDM is only the first step in the process of improving software engineering practices in an organization. SDMs must become living documents that evolve with changing software development needs. The research presented in this paper presents a tool and associated methodology that helps organizations tailor SDMs to the individual needs of projects and capture project experiences that are used to refine and modify the standard. In addition to creating a flexible software process with necessary degrees of formal procedures to ensure high-quality products, a secondary goal of this research is to turn SDMs into a resource for software managers and developers, something that truly supports the development process as it is actually practiced. This approach features a methodology based on organizational learning principles, a rule-based system to tailor the SDM to meet the characteristics of individual projects, and a case-based architecture to capture and provide relevant development knowledge throughout the development lifecycle.*

*Keywords: Software Process, Software Engineering, Experience Factory, Organizational Learning, Knowledge Management*

---

<sup>1</sup> This research was funded by the National Science Foundation (CCR-9502461, CCR-9988540, and ITR/SEL-0085788).

# 1 Support for Software Development Methodologies

Software development processes have received a great deal of attention in software engineering research and practice. Approaches to this problem are as varied as lifecycle modes, software process programming [32], and software process frameworks, such as the Software Capability Maturity Model (CMM) and variants [33], Bootstrap [24], SPICE [9, 23] and ISO 9000 [19], and others. Certification of organizations against these standards are becoming increasingly important in contract awards and in the marketplace, causing software development organizations to pay greater attention to development practices.

A key element of these standards and approaches is the creation of a defined process, normally in the form of a Standard Development Methodology (SDM) that outlines the activities that should be taken when developing software systems. While the industry is experiencing a resurgence in the importance of SDM through compliance standards such as CMM and ISO, the creation of SDMs is familiar territory for many organizations, which have spent valuable resources creating comprehensive SDMs only to watch them collect dust for various social, political, and technical reasons. One of the most cited reasons for not following the organization standard is that it “does not meet our project’s needs,” usually because the technology has exceeded the reach of the standard, or so the argument goes. While the process frameworks normally mandate the use and maintenance of SDMs, it is critically important that tools and methodologies be created that support the development, use, evolution, and tailoring of SDMs to meet the dynamic needs of modern software development organizations.

There are a number of problems inherent to these kinds of standardization efforts. First, to accommodate all types of development efforts in an organization, the standard will inevitably settle at a level of abstraction that meets the needs of all projects, but fails to provide specific guidance to individual project activities. The Capability Maturity Model and its variants are a good example of this. Described most correctly as a framework for software process, this model explicitly describes high-level activities in the form of key practices such as “Estimates for the size of software products are derived according to a documented procedure,” but leaves the details on how the activity is conducted for organizations to define. A good deal of effort is needed to make this useful for a specific organization, and even more tailoring is necessary at the project level. Second, in the absence of constant monitoring and review, the standard will quickly become obsolete in today’s fast-paced world of technology. No longer a static document, SDMs must become a dynamic, living, document. CMM and ISO standards address monitoring the standard as part of their processes, but support is needed to coordinate and manage these dynamic documents and processes.

Creating tools and technologies to manage SDM documentation has received little attention in the software engineering research community. Current practice is to put process standards on-line through the Web or intranet facilities, a negligible improvement over the monolithic documents that characterize traditional SDM practices. Too often, these standards are buried

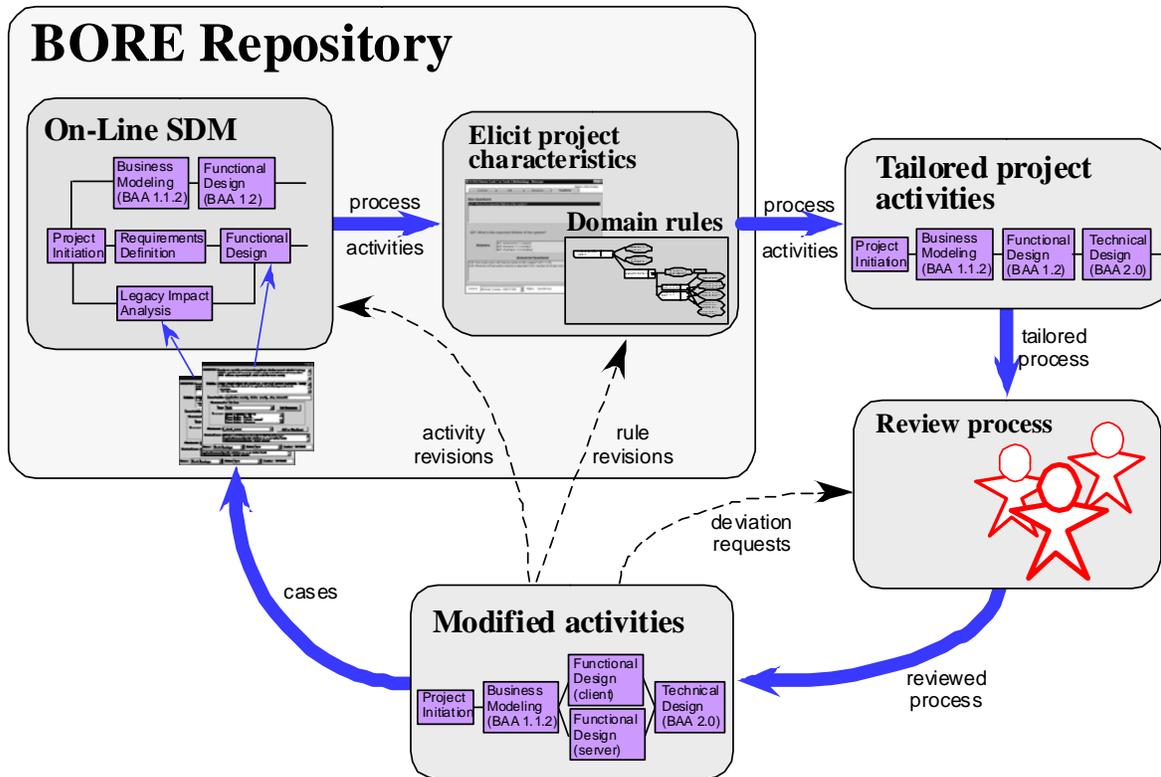
in Web pages that become severely underused in the daily activities of software development. The Web page approach alone fails to provide the means to update the standard as needed and does little in the way of making SDMs an integral part of the development process. Even if SDM curators are appointed, and reviews of the methodology instituted, it is difficult for a few people to keep pace with changes in large development organizations.

In order to become more than just a certification tool, a modern SDM must become more than a standard whose use is mandated. It must become a valuable resource that software developers and project managers can depend on for critical support and information. In many ways, the SDM embodies knowledge of development practices. But this knowledge is often idealized and abstracted away from the realities of everyday work practices. Process frameworks have done little to resolve issues of how the knowledge embedded in software processes can be turned into a repository of proven practices that can be brought to bear on software development efforts. Further research is needed into the development of tools and techniques that capture specific guidelines, examples, deviations and other information and place them within the framework of the defined process to become an integral and valuable tool for developers in the organization.

This paper presents an approach and associated tools that help software development organizations define SDM processes and use feedback from projects using the SDM to refine and improve their procedures. Tools are presented that define an SDM in terms of a set of activities and rules for when the activities should be applied to a project. Individual projects create an instance of the SDM and choose options, stated in the form of project requirements, which invoke rules to assign activities to the project. When assigned to projects, activities become cases, containers of context-specific problem solving knowledge of how the activity is adapted and used in the project. A meta-process is used to review project activities and potentially add new paths based on project experiences. The objective of this approach is to turn SDMs into a resource, something that truly supports the development process as it is actually practiced, while adding necessary degrees of formal procedures to ensure high-quality products.

## **2 A Methodology for SDM Evolution**

Turning static SDM definitions into living documents that provide valuable resources for developers and managers requires a combination of methodology and tool support. We have been investigating these issues through an exploratory prototype named BORE (Building an Organizational Repository of Experiences), a Web-based tool designed to support diverse software process models and capture and disseminate project experiences. Early research explored the case-based [14] and organizational learning approaches [18] for developing software systems. In this paper, we explore how BORE can be used to turn an organization's SDM into a living document that evolves and improves with use.



**Figure 1:** Methodology for tailoring SDMs to projects and refining the SDM.

Defining the meaning of process is difficult and open to many interpretations [27]. For our purposes, a process is defined as a set of tasks. Tasks can further be broken down into constituent tasks or activities. Activities are the leaf nodes in this definitional hierarchy that define specific work activities, such as “Develop a System, Software and Operations Concept” or “use the standard Web browser login package.” Our methodology refers to the meta-process that controls the definition, evolution, and maintenance of SDMs and their associated software development processes.

Our methodology begins with a defined, on-line, SDM, as depicted in the upper-left of Figure 1. Multiple paths through the SDM are allowed to accommodate the different kinds of projects typically encountered in the organization. The figure shows an SDM that defines three different paths in the standard process, which could, for example, define the differences between enterprise-wide, department-specific, and prototyping software development projects. More complex structures are possible, and, unlike monolithic SDM documents, not everyone is expected to understand the SDM in its entirety. Instead, a decision support system is used to reduce the complexity by presenting options for different paths through the SDM. Options are chosen that best accommodate the specific characteristics of the project. For example, a payroll system will probably need to choose options that lead to enterprise-level activities while an exploratory prototype for a specific group will need a more

lightweight process. Project characteristics are matched against rules that match process activities in the overall process against project characteristics to produce a process tailored to the specific need of the project. Figure 1 shows an example where a specific track of the on-line SDM has been selected to create the tailored project activities.

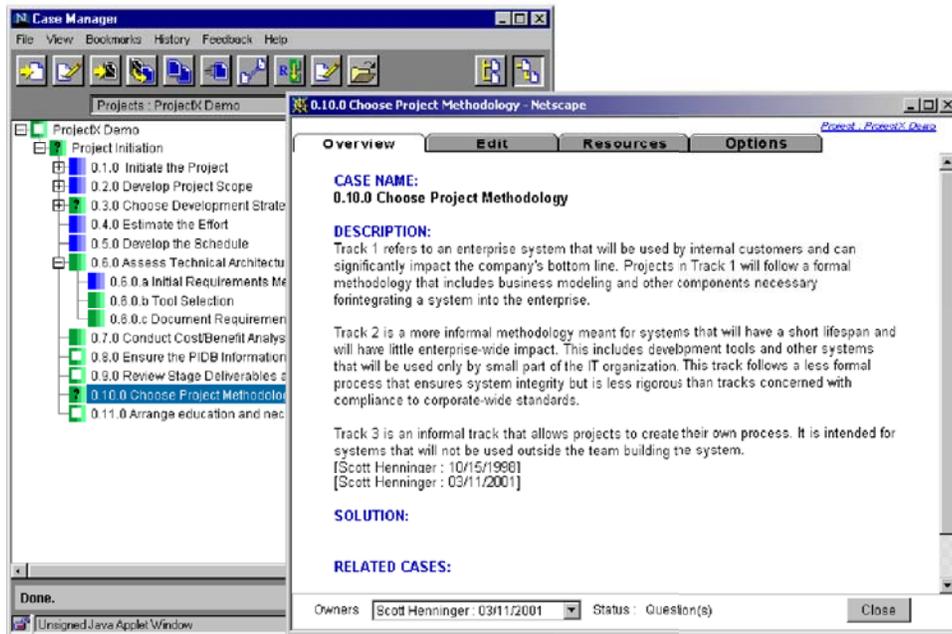
Once the tool has been used to create a tailored process for the project, the development team reviews the resulting activities and begins to perform them. The activities will undergo periodic review as the project progresses and new decisions are made. During these reviews, the options chosen in BORE are reviewed and any modifications made to the resulting process are carefully reviewed for accuracy and adequacy. The outcome of the review can be one of three actions: 1) If it is determined that the chosen options were incorrect or new events have necessitated a change to the option, then the options are changed and the process activities are chosen by BORE rules that match the new options. 2) Where the defined options and the activities available in the SDM are deemed inadequate, a deviation request is generated to define new options and activities. 3) The tailored process is approved.

The result is a tailored process meeting the characteristics of the project, as depicted in the “Modified activities” bubble in Figure 1. When executing the activity, the development team will document progress in “cases” that contain project-specific information. These cases are automatically associated with the activity in the standard (see later sections on the BORE tool) and become resources that other projects can draw on. Deviations from the existing standard are treated as opportunities for improvement and can be reviewed and fed back into the SDM as new paths that subsequent projects can follow. This is accomplished by creating new activities in the standard and using the rationale for the deviation request to create rules to dictate the circumstances under which the activities should be assigned to a project. In other words, projects with similar characteristics are allowed to follow the newly created path.

With this methodology, the SDM is allowed to grow and evolve to meet the dynamic needs of a software development organization. The initial SDM is seen as a “seed” that evolves as it is used [12]. The SDM is therefore conceptualized as not just a process to follow, but the focal point for an organizational learning process [17]. As the project cycles through the milestones and reviews, the process is continuously updated, allowing feedback to shape the SDM to meet their dynamic and fast-paced needs of modern software projects [34]. Implementing this methodology requires a flexible mechanism to turn the SDM into a *living document* designed to evolve and improve through use, drawing on the collective knowledge of the organization. The following sections describe such a tool.

### **3 Tools for Supporting SDM Tailoring and Modification**

The BORE prototype is a Web-enabled application using a three tiered architecture consisting of HTML, Javascript and Java for rendering the interface, Java for application logic and JDBC access, and relational database back-end. It can be accessed through a Web browser at



**Figure 2:** Case Manager and Case Window.

<http://cse-ferg41.unl.edu/bore.html>.<sup>1</sup> The BORE architecture consists of a server implemented as a Java application using JDBC to communicate with a relational SQL database. The server is primarily a database server, receiving queries from the client, passing them on to the database, getting results and packing the result set into a string that is sent back to the client. The client is largely rendered as a Java applet, although cases (explained later) are rendered by HTML to allow the easy definition of multimedia documents.

BORE has two main interfaces, as shown in Figure 2. The Case Manager, shown to the left in Figure 2, displays a hierarchical arrangement of cases that define the activities in a project's development process. In the figure, a project named "ProjectX Demo" has been chosen from the list of resources in the drop-down menu that displays projects. Each of the nodes in the project hierarchy, which are cases corresponding to project activities, can be opened to view information about the activity. In the window to the right of Figure 2, the activity named "Choose Project Methodology" has been opened.

Cases are used in a case-based decision support approach [22], and describe situation-specific solutions to software development activities. Cases consist of a description to a problem, a solution statement, and some ancillary information, such as related cases, and etc. For

---

<sup>1</sup> The system functions best when used by Netscape Communicator 4.x or later versions, or Internet Explorer 5.0 or later versions. Java and Javascript must be turned on. Guest users may log in as 'guest' or request an account.

example, in Figure 2, the description field describes the SDM standard for a specific activity,<sup>2</sup> choosing the methodology to be followed by a project, in this instance. The solution field is used to document project-specific information accumulated while executing the activity. The Edit tab is used to edit the activity fields and the Resources tab is used to attach documents to the activity. The Options tab, which will be described in detail later, is used to choose SDM options available to the activity. Activity status is kept for each activity and is displayed as a color-coded icon in the Case Manager (active, resolved, etc.), thus providing a way to get a visual overview of a project's status.

These core features represent the case-based organizational memory paradigm of BORE [14]. Each case represents a project-specific activity that is used to help coordinate and disseminate project experiences. The project hierarchy can be used as a dynamic checklist of project activities that capture information about projects in an organization-wide repository. This creates a repository of experiences that software development efforts can utilize to find potential solutions to development problems.

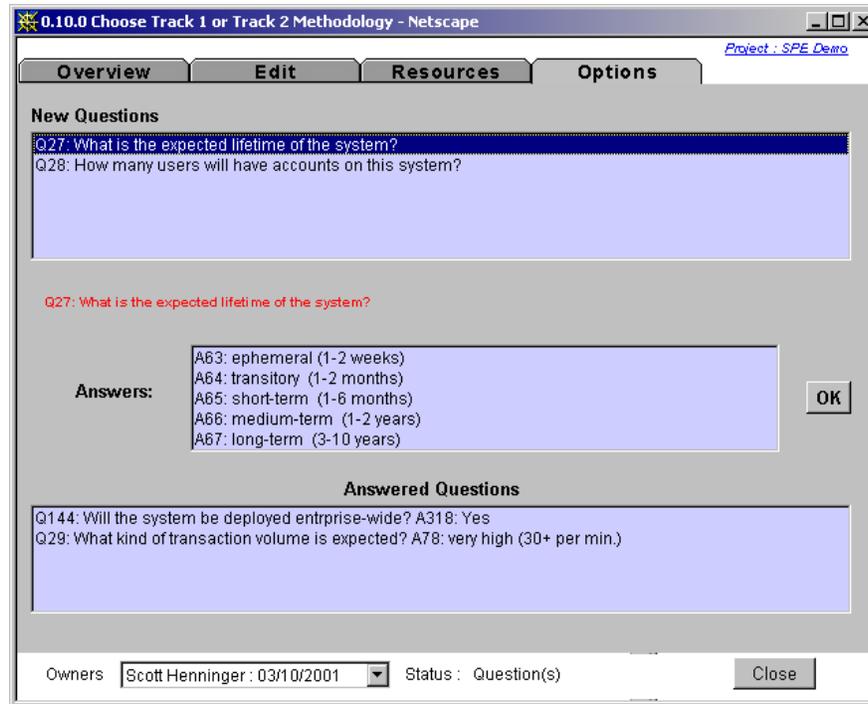
### 3.1 BORE Knowledge Domains

SDMs are defined in BORE through *domains*, which are independent SDMs defined by a set of domain *activities* and domain *rules* that define when an activity should be used. Projects belong to a single domain, which is chosen when a project is created in BORE. All subsequent project activities will use the cases and rules defined for that domain. Allowing multiple domains supports scalability and allows organizations to partition development activities into domains that mirror diverse environments or product lines.

Domain activities define the space of possible activities for projects within a given domain. They can be as high-level or detailed as desired. The Case Manager in Figure 2 shows a set of activities for project initiation, taken from the SDM of a medium-sized IT division for a large corporation. Activities are defined in a task/subtask relationship represented in a hierarchical work breakdown. Although task ordering and/or dependencies are often defined in process models [20], organizations we have worked with are more interested getting the required activities in place, a sizeable task in itself, than trying to impose a workflow of activities. At this point, task ordering is therefore left to human intervention through project management software or other techniques.

---

<sup>2</sup> When applying BORE to the domain of software development processes, cases are used to implement project activities. Therefore, the terms “activity” and “case” are used interchangeably in this text, depending on whether the context is software projects or the underlying case-based architecture.

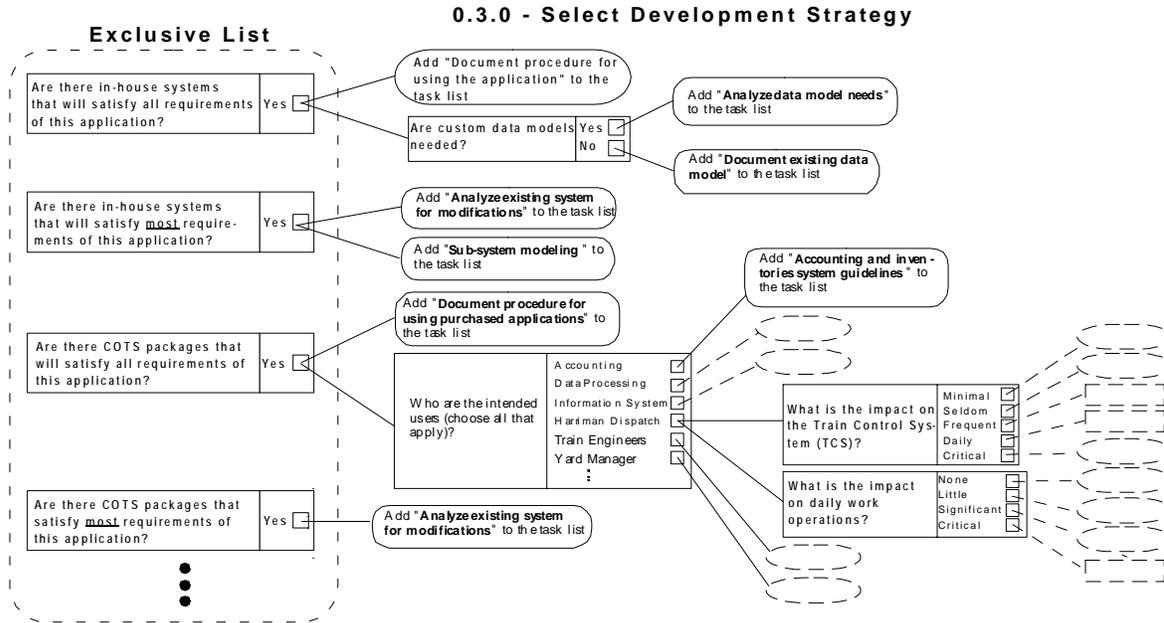


**Figure 3:** The Options tab.

The defined SDM can be tailored to meet project-specific needs through domain rules, which define a set of conditions under which a project is required to execute a specific activity. In other words, the domain rules choose which domain activities a project must follow to conform to the SDM standard. Therefore, the standard may define more activities than any one project will be held accountable for (such as the three tracks depicted in Figure 1), and the domain rules determine which of those activities are applicable.

### 3.2 Tailoring the SDM to Specific Projects

When a project is created in BORE, an instance of the SDM is created. This involves creating a set of project initiation activities that are specified in the domain's "rule 0" and copying the associated domain activities into the project hierarchy. Project personnel can then perform the work specified in the activities. The initial activities represent a first pass to define project activities that can be further broken down through options that can be attached to any of the domain activities. Activities that have options associated with them are denoted by the "?" in the icon next to the activity label in the Case Manager (see the project hierarchy in Figure 2). The options are found by opening the Options tab (Figure 3) and are used to tailor the process to the project's unique characteristics, usually by breaking down the activity into constituent sets of subtasks.



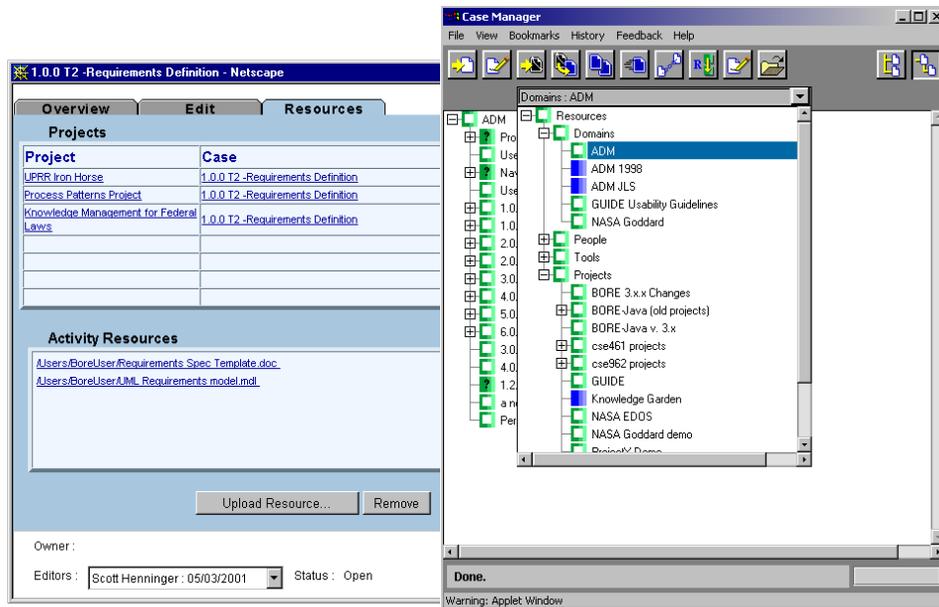
**Figure 4:** A partial decision tree for choosing development strategies.

For example, the development process in the example shown in Figure 1 is divided into three separate tracks depending on whether the project is a mission critical enterprise-wide application, a local application, or a research prototype. As part of the initiation process in this organization’s SDM, the project must choose one of these tracks, which will define the majority of the rest of the project. This is documented through the case named “0.10.0 Choose Project Methodology”, shown in Figure 3. At this point, two questions have been answered and two more remain in the New Questions field. When a question is chosen, the possible answers are displayed in the Answers field in the middle of the window. Selecting the answer and OK chooses the answer, and the question/answer pair is moved to the Answered Questions field.

A key feature of this tailoring process is the flexibility offered by allowing options to be associated with any domain activity. This allows SDMs to be defined that iteratively expand each part of the project’s process as they gain the knowledge necessary to address process issues [32]. Initial projects can define activities with options that add new activities that can in turn have options to further refine the process and provide resources for the project. In this way, the project tailors the overall SDM to its needs when it is able to address the issues involved. In addition, varying levels of detail can be supported by defining options as deeply in the tree of domain activities as desired.

### 3.2.1 The Rule-Based System

When an option is chosen by answering a question, the rule engine is invoked to find satisfied rules. Each rule has a set of preconditions consisting of one or more question/answer pairs.



**Figure 5:** Editing domain activities.

When all preconditions for a rule evaluate to true, the rule “fires,” causing a set of actions to be executed. Actions in BORE can take one of three forms: 1) A question can be removed from the question stack (displayed in the New Questions field of the Options tab of a case), 2) a question can be added to the question stack, and 3) a case or set thereof is added to the project in the proper places in the hierarchy as defined by the SDM domain. Other actions, such as adding information to a database, automatically alerting people of events through e-mail, and other types of actions as required by corporate standards are under development.

The rule engine provides the flexibility to implement complex decision trees. While AND relationships are inherent in the precondition/action structure of the rules, OR relationships are easily represented by having two rules with different preconditions and the same action. Combinations can be created by mixing these strategies as needed. Figure 4 shows a partial decision tree that has been implemented in BORE for choosing development strategies. The “Exclusive List” is implemented by displaying a list of questions to choose from. No ordering is imposed on questions displayed in the New Questions field of the Options tab (Figure 3), so the user is free to choose the most appropriate questions for their situation.

Answers can easily be changed or un-answered, allowing users to easily engage in a what-if dialogue with BORE. This is accomplished by choosing a question/answer pair in the Answered Questions field. All answers plus a “un-answer” answer will appear in the Answers field. Choosing a different answer causes bore to backtrack the actions previously executed and executing the actions required by any rules fired by the new question/answer pair.

Rules are stored in a relational database to facilitate permanence. Users may exit the Options tab at any point in the session. The state of the system will be stored in the database and restored when the user opens the Options tab for that case. Bore maintains a separate rule space for each project so that multiple projects can choose individual options.

### *3.2.2 Case-Based Resources for Questions*

The intention of BORE project cases is to provide decision support throughout the development cycle that is based on past experiences of the organization. Of crucial importance is how questions are answered. BORE must provide information that can help developers and managers make informed choices. Double-clicking on any of the questions provides hyper-links to documents, standards, and other on-line information, if available for the given question, to help users decide which answer is right for the project.

Another resource for answering questions is rooted in the case-based paradigm of BORE. Because every case in a project is copied from the SDM domain activities, users can easily see a listing of all projects assigned that specific domain activity. For example, all projects assigned the “0.10.0 Choose Development Strategy” activity can click on the Resource tab (see the Projects field in Figure 5) for a list of all instances of that domain activity and their projects. Users can click on this tab and browse for similar projects, view their answers to questions, and peruse any case information about that project’s experiences. Access to this experience-based information can lead to insight and information that can help make good choices.

## **4 Modifying the SDM**

Project tailoring of the SDM is accomplished through choosing options in activities. Modifying the content of the SDM to meet emerging development needs in the organization consists of either or both adding tasks to the SDM and editing the rules for adding tasks to a project (see Figure 1). For example, suppose a project is amongst the first to create Java applets, and designs a method to use JDBC to access Oracle tables. To do this means deviating from the SDM because certain issues of creating a Web-based database server haven’t been addressed before in this organization. The project goes through the review process (see Figure 1), is approved, and engages in documentation required to pave a path for projects with similar characteristics.

Individual projects can create new activities, but escalating these activities to the SDM should be performed by a person or team with organization-wide responsibility for the SDM. This person or group would review approved deviations and make appropriate changes. In our example, new domain activities, such as “Create Oracle configuration files” or “Evaluate JDBC software,” would be created and added to the domain in a task/subtask breakdown. The deviation rationale, in this example stating that the SDM needs to have activities for projects using JDBC and Oracle with a Java applet, is translated into rules. New questions are created and associated with the proper domain activities. For example, a domain activity

on the software architecture may add an entry for Java applets and a question on database back-ends. Choosing these options will assign the newly created activities on using JDBC to the project. In this manner, the project's experiences are used to pave a new path that other projects can follow as part of the refined SDM.

#### **4.1 Editing Domain Activities**

Domain activities are created and edited in the Case Manager using the same interface and tools available for editing project hierarchies. The window to the right in Figure 5 shows the resource menu with five different domains<sup>3</sup> defined, each of which has a separate set of domain activities and rules. To add new activities to the domain, personnel in charge of the SDM create new cases, place them in the domain's task/subtask hierarchy (shown behind the drop-down Resource menu in Figure 5), and populate them with information about the standard. Templates of any document type can be attached to domain cases as shown in the Activity Resources field of the Resources tab.

When an activity is assigned to a project, creating a project case, all information in the domain activity is copied to the case in the project. Project personnel edit the case fields and download the templates to complete them with project information. The edited documents can then be uploaded to the server so others can view them (version management of these documents is slated for a future version of BORE).

#### **4.2 Editing Domain Rules**

The Rules Manager provides tools for creating and editing domain rules and associated questions and answers. The initial window displays preconditions and actions for a given rule, as shown to the left of Figure 6. A rule is defined through a set of preconditions, question/answer pairs, and actions, that are fired when all preconditions evaluate to true. For example, when the question on expected lifetime of the system is answered "medium-term" (see the Preconditions field in the left-hand window of Figure 6) and the transaction volume is answered "high", then the actions specified in the Actions field are executed. In this instance, six cases are added to the project, a question is added to the New Questions stack (see Figure 3), and a question is removed from the New Questions stack (see the Actions field in the left-hand window of Figure 6).

Preconditions are edited in a separate window (not shown) that choose the question/answer pairs for the rule. Actions are edited through the Edit Actions window shown to the right in Figure 6. Actions involving adding or removing activities to/from a project are created by choosing the action, type, and activity to be added. Other options for the activity, such as the initial status and activity owner, can also be chosen. Questions can also be added or removed from the New Questions list by a rule. This allows decision trees such as the one in Figure 4

---

<sup>3</sup> "ADM" stands for "Application Development Methodology," a SDM created by the IT department of a large transportation corporation.

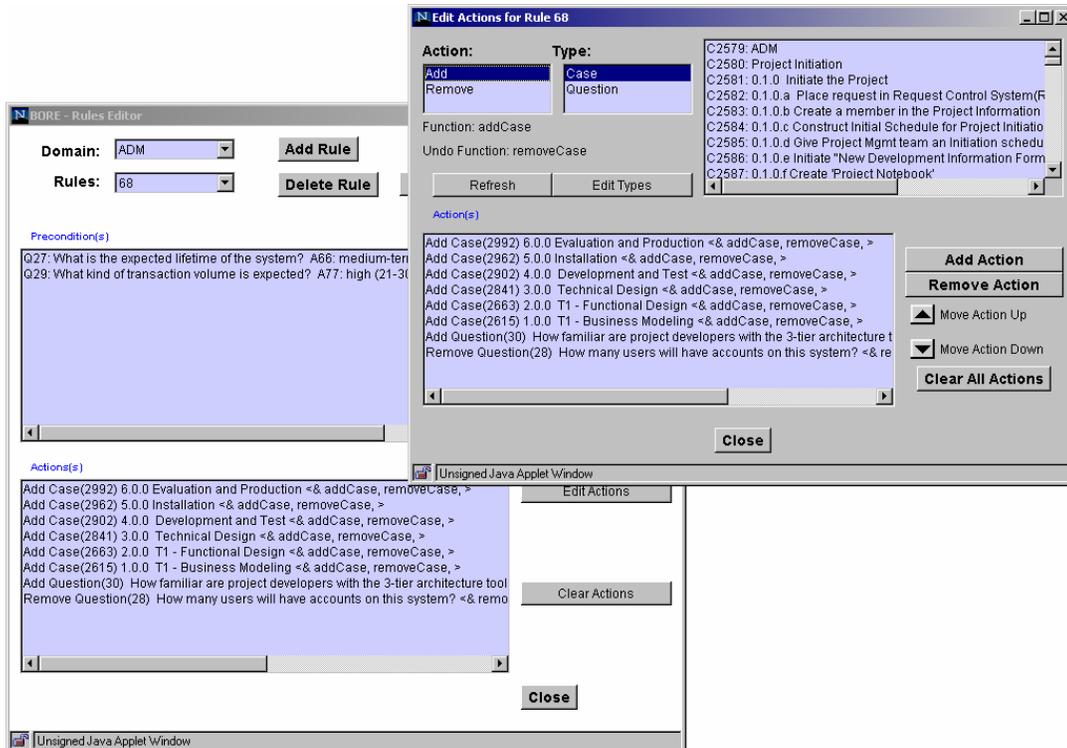


Figure 6: Editing Rules.

to be implemented. The list of actions is executed in the order shown in the Actions field, and can be moved by using the arrow buttons.

## 5 Analysis and Related Research

The general goals of this approach are twofold: 1) create a medium for flexible software process definitions that split the gap between overly-restrictive development methodologies and ad-hoc software development practices. 2) Provide the tools to create a flexible software process that evolves with use to provide valuable resources to software developers and managers. Currently, the industry standard is to define a SDM and then ignore it in its entirety or follow it enough to satisfy periodic audits. We wish to turn SDMs into a resource, something that truly supports the development process as it is actually practiced, while adding necessary degrees of formal procedures to ensure high-quality products. This involves not only defining a process, but also using feedback from projects using the SDM to refine and improve its procedures.

These concepts have their predecessors, particularly in the form of software factories [2-4], process programming [7, 32], and some research on interfaces and CASE tools to support these concepts [5]. This work attempts to fill this gap with the organizational learning perspective, case-based decision support, and the BORE prototype. Some of the important

concepts related to our approach are examined below to assess the contributions made by this research.

## **5.1 Software Process Research**

Most research on software processes has thus far focused on defining the process. Approaches include high-level strategies such as the waterfall, spiral, and prototyping models, methods for combining process elements [32], and universal process frameworks such as the CMM [33], SPICE [23], or ISO 9000. A significant contribution of this work is to define not only the process, but how the process evolves with the changing needs of the development organization. BORE does not define a specific development methodology, but provides tools to create and refine organization-specific methodologies. The SDM is seen as a “seed” that evolves as it is used [12]. In addition, the process can be defined at many levels of detail, allowing organizations and projects to define and adopt processes appropriate to diverse development needs.

This approach fills four important gaps in fulfilling the Level 3 CMM requirement for a defined process [33]. The first is that while the CMM calls for tailoring the defined process to individual projects, no guidance or procedures are provided on how this should be accomplished. Our approach formalizes the tailoring process through BORE question/answer sessions and a review process to consider deviations that can further refine the SDM. The second gap is the need for refining, evolving, and improving the development process. No “Level 5” process is assumed. Rather, feedback from actual projects is used to define the best known methods within the development context and application requirements of a specific development organization. Third, instead of being confined to defining process at the level of “Configuration Management”, and etc., this approach allows more detailed information to be specified where necessary, such as when full version control is necessary or different CM procedures for different kinds of source, such as HTML vs. programming source. In addition, the method provides the means to associate information with each process activity, supplied not only through documentation of the standard, but also links to cases detailing how specific projects have executed the activity. Fourth, conformance with the standard is an inherent feature of BORE. Each project creates an instance of the SDM and documents the project from the resulting project activities. A useful side-effect of this is that project experiences can easily become available for project personnel to draw on and improve.

## **5.2 Design Rationale**

Our approach also has some roots in the design rationale field [6, 10, 25, 28, 29]. Similar to the organizational learning approach, the motive for capturing design rationale is to avoid repeating common pitfalls or re-discussion of design decisions by describing the alternatives debated and decisions made for a given effort. Many schemes, from Procedural Hierarchy of Issue structures [6, 10] to more complex structures designed to capture logical relationships for computation [25] have been devised. All have the same basic structure of a set of

hierarchically arranged questions posed to flesh out issues. Alternatives and rationale for the alternatives can be attached to the questions to document discussion paths.

While a valuable brainstorming tool that organizes alternative solutions, design rationale techniques have not focused on how past rationale can be used to drive a reuse-based software development process. Furthermore, there is no mechanism to ensure the rationale is used to improve future efforts, and therefore often falls into the category of write-only memory. Our approach is more principled, putting explicit procedures in place to compare past and present design contexts as an impetus for continuous improvement. The approach described here also differs from the current trend in design rationale research to motivate developers to use the approach by lowering the cost of collecting information. This has led to the development of a plethora of retrieval or filter-based systems that work on existing documents, such as e-mail messages or discussion databases [11], that tend to suffer from either or both information overload and knowledge paucity. Our motivator for using the system is part conformance and part self-motivating. The approach embodies corporate procedures that need to be followed, but the extent to which the approach can be tailored to real project needs and provide developers and managers with valuable information will determine its fate as a development strategy.

### **5.3 Organizational Memory and Learning**

The organizational learning approach also incorporates elements of organizational memory and a formalized approach to supporting a learning organization with information technology. The Designer Assistant project at AT&T created an organizational memory system whose use became part of the design review process as a way of ensuring conformance to the usage of a specific piece of complex functionality in a large switching system [35]. In this setting, the design process was modified to include a trace of the Designer Assistant session as part of a design document. The appropriateness of the designer's choices and adequacy of the advice given by Designer Assistant are discussed during software design reviews. If the advice is found to be lacking, designers begin a formal change process to update the knowledge. Utilizing a combination of existing and new organizational processes to place use of Designer Assistant into development practices ensures that the knowledge will evolve with the organization.

The BORE methodology draws heavily from Designer Assistant's review process. The observation that "technology and organizational processes are mutual, complementary resources" [35] has served as a guiding principle for this work. The BORE tool and methodology further refines the review process and provides tools to edit the repository and SDM. The scope of BORE's repository is also broader. While the Designer Assistant represents programming constructs, BORE can represent other kinds of artifacts and manage the knowledge through the activity breakdown that defines the process.

The rule engine for selecting process elements is similar to branching networks of multiple-choice questions [1] and decision trees [35]. Checklists have also been used to provide

information about steps that need to be followed to achieve a certain task [26]. The Case Manager (Figure 2) can be seen as a kind of checklist that enumerates the activities that must be accomplished at each step in the development process. The color-coded representation of activity status can also provide a visual overview of project progress.

#### **5.4 Open Issues and Future Directions**

Early prototyping and work with software development organizations have demonstrated that the BORE approach is well suited to capturing existing SDMs, and provides possibilities that organizations have not previously envisioned [16, 17]. Many organizations have a need to create an SDM that reflects the true complexity and diversity that lies at the intersection of business needs, development tools, and developer skill sets that defines modern software development. But there is an inherent struggle between understanding that complexity and the desire to create a standard that is easily understood by managers and developers alike. The BORE tool provides some relief for this tradeoff. Project personnel do not need to understand the SDM in its entirety, just those elements the project is responsible for. But the burden of understanding a more complex process remains for personnel with responsibilities for maintaining the SDM. Better tools are needed to supply higher-level views of activity structures and relationships between rules.

The BORE system currently relies on rules to find relevant activities for projects. To the extent that the activities can be designed properly, this will work. But there may be instances where a search for related activities may be useful. BORE has a simple search tool, but we are looking into more sophisticated case-based retrieval. Hierarchical case-based planners [31], which generate plans based on case-based search criteria and options, seem particularly well-suited to the kinds of work we are pursuing. We hope to draw on some work that integrates rule-based and case-based systems [13, 30] to find work breakdowns that are relevant to the characteristics of a project.

Maintaining the BORE repository requires some personnel specializing in software process and maintaining rule bases. First is the need for process experts. Many organizations, fueled by the need to conform to the CMM or other standards, are creating Software Engineering Process Groups (SEPGs) or otherwise dedicating some effort to maintaining process standards. BORE can serve as a tool to assist these groups in the creation and maintenance of the organization's process. Rule-based expertise is also needed to maintain BORE's rule base. Complex rule structures can be as difficult to maintain as software systems, and require similar techniques, such as version control and test plans. Organizations often find it difficult to justify the overhead of such positions and continued effort is needed to minimize documentation overhead while further understanding the benefits gained by investing in an organizational learning strategy such as the one presented here.

Some of the research questions we will continue to explore with BORE include designing work practices centered around case-based organizational learning processes. Design tools must reward users using the tool with high-quality knowledge, as well as leaving information

for subsequent development and maintenance efforts. We believe these are compatible goals. Long-term projects need to track their progress and decision making process, but the challenge is to capture this knowledge in a form that can be used in other development efforts. Some major research questions include: designing methodologies based on standards set by previous projects yet flexible enough to accommodate changes in business needs and technology; creating a development process that begins by using the resources supplied by the repository and updates the repository as the project progresses; finding the “right” level of documentation that doesn’t overwhelm developers yet leaves a trace for subsequent efforts; and organizational changes needed to make such techniques work in the organization (i.e. the technology transfer and tool adoption problems).

Most important is the extremely difficult task of getting organizations to take part in efforts to evaluate emerging research in realistic settings. The BORE system is beginning to move out of the prototyping phase and we hope to deploy it in a few development organizations, including NASA Goddard [15], some local small organizations, and the Software Design Studio at the University of Nebraska-Lincoln [8].

## **6 Conclusions**

Centering SDM and project information around a single repository through an ongoing process of capturing project experiences can prevent the duplication of efforts, avoid repeating common mistakes, and help streamline the development process. Similar projects have shown that such a system will be used by development personnel, provided it contains relevant, useful and up-to-date information [35]. This mandates a strong tie between technology and process in which using the technology must become part of routine work activities.

To achieve these goals, we have coupled process and technology to turn SDM documents into dynamic “living” resources that can be extended and improved as new project needs and application requirements emerge. As the repository accumulates knowledge through principled evolution of the SDM, its contents improve and it is able to handle a wider range of circumstances [21], while evolving toward answers to problems that fit the organization’s technical and business context. The real question is not whether the repository is “correct” in some objective sense, but rather whether fewer mistakes are repeated and better solutions adopted when using the repository.

Early BORE evaluations indicate progress has been made on our overall goals of creating a medium for flexible software process definitions and turning SDM documents into a resource that supports the development process as it is actually practiced [16, 17]. Future work will continue to evaluate the system in realistic software development settings and further investigate tools and methodologies to capture experiences and feed them back into the process to facilitate continuous of the process and resulting products.

**Acknowledgments.** I gratefully acknowledge the efforts of a number of graduate students that have helped develop BORE, including Kurt Baumgarten, Michelle Conway, Shawn Clymer, and Roger Van Andel. This research was funded by the National Science Foundation (CCR-9502461, CCR-9988540, and ITR/SEL-0085788). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

## References

1. M. S. Ackerman and T. W. Malone, "Answer Garden: A tool for growing organizational memory," *Proceedings of the Conference on Office Information Systems*, pp. 31-39, 1990.
2. V. R. Basili, G. Caldiera, and G. Cantone, "A Reference Architecture for the Component Factory," *ACM Transactions on Software Engineering and Methodology*, vol. 1, pp. 53-80, 1992.
3. V. R. Basili and H. D. Rombach, "Support for Comprehensive Reuse," *Software Engineering Journal*, pp. 303-316, 1991.
4. V. R. Basili and H. D. Rombach, "The TAME Project: Towards Improvement-Oriented Software Environments," *IEEE Transactions on Software Engineering*, vol. 14, pp. 758-773, 1988.
5. G. A. Bolcer and R. N. Taylor, "Endeavors: A Process System Integration Infrastructure," *Proceedings of the Fourth International Conference on the Software Process*, Brighton, UK, pp. 76-85, 1996.
6. E. J. Conklin and K. Yakemovic, "A Process-Oriented Approach to Design Rationale," *Human-Computer Interaction*, vol. 6, pp. 357-391, 1991.
7. B. Curtis, M. I. Kellner, and J. Over, "Process Modeling," *Communications of the ACM*, vol. 35, pp. 75-90, 1992.
8. S. Dunbar, S. Goddard, S. Henninger, and S. Elbaum, "Bootstrapping the Software Design Studio," *Fifth Annual National Collegiate Inventors and Innovators Alliance National Conference*, Washington, DC, pp. 180-188, 2001.
9. K. E. Emam, J. N. Drouin, and W. Menlo, *SPICE: The Theory and Practice of Software Process Improvement and Capability Determination*. Los Alamitos, CA: IEEE Computer Society, 1998.
10. G. Fischer, J. Grudin, A. Lemke, R. McCall, J. Ostwald, B. Reeves, and F. Shipman, "Supporting Indirect Collaborative Design With Integrated Knowledge-Based Design Environments," *Human-Computer Interaction*, vol. 7, pp. 281-314, 1992.
11. G. Fischer, S. Lindstaedt, J. Ostwald, M. Stolze, T. Sumner, and T. Zimmermann, "From Domain Modeling to Collaborative Domain Construction," *Proc. Designing Interactive Systems (DIS '95)*, Ann Arbor, MI, pp. 75-85, 1995.

12. G. Fischer, R. McCall, J. Ostwald, B. Reeves, and F. Shipman, "Seeding, Evolutionary Growth and Reseeding: Supporting the Incremental Development of Design Environments," *Proc. Human Factors in Computing Systems (CHI '94)*, Boston, MA, pp. 292-298, 1994.
13. A. Golding and P. S. Rosenbloom, "Improving Rule-Based Systems Through Case-Based Reasoning," *Proc. 9th National Conference on Artificial Intelligence*, Anaheim, CA, pp. 22-27, 1991.
14. S. Henninger, "Case-Based Knowledge Management Tools for Software Development," *Journal of Automated Software Engineering*, vol. 4, pp. 319-340, 1997.
15. S. Henninger, "Software Process as a Means to Support Learning Software Organizations," *Twenty-fifth Annual NASA Software Engineering Workshop*, Greenbelt, MD, 2000.
16. S. Henninger, "Tools Supporting the Creation and Evolution of Software Development Knowledge," *Proceedings of the Automated Software Engineering Conference*, Lake Tahoe, NV, pp. 46-53, 1997.
17. S. Henninger, "Using Software Process to Support Learning Software Organizations," *1st International Workshop on Learning Software Organizations (LSO 1999)*, Kaiserslautern, FRG, 1999.
18. S. Henninger, K. Lappala, and A. Raghavendran, "An Organizational Learning Approach to Domain Analysis," *17th International Conference on Software Engineering*, Seattle, WA, pp. 95-104, 1995.
19. R. Kehoe and A. Jarvis, *ISO 9000-3 : A Tool for Software Product and Process Improvement*. New York: Springer, 1996.
20. M. I. Kellner, P. H. Feiler, A. Finkelstein, T. Katayama, L. J. Osterweil, M. H. Penedo, and H. D. Rombach, "ISPW-6 Software Process Example," *First International Conference on Software Process*, pp. 176 -18, 1991.
21. J. L. Kolodner, *Case-Based Reasoning: Morgan-Kaufman*, San Mateo, CA, 1993.
22. J. L. Kolodner, "Improving Human Decision Making through Case-Based Decision Aiding," *AI Magazine*, vol. 12, pp. 52-68, 1991.
23. M. D. Konrad and M. C. Paulk, "An Overview of SPICE's Model for Process Management," *Proc. 5th International Conference on Software Quality*, Austin, Texas, 1995.
24. P. Kuvaja and A. Bicego, "BOOTSTRAP - A European Assessment Methodology," *Software Quality Journal*, vol. 3, pp. 117-127, 1994.
25. J. Lee, "Design Rationale Capture and Use," *AI Magazine*, vol. 14, pp. 24-26, 1993.
26. A. C. Lemke, "Design Environments for High-Functionality Computer Systems," in *Department of Computer Science*. Boulder, CO: University of Colorado, 1989.

27. M. Lindvall and I. Rus, "Process Diversity in Software Development," *IEEE Software*, vol. 17, pp. 14-18, 2000.
28. A. Maclean, V. Bellotti, R. Young, and T. Moran, "Questions, Options, and Criteria: Elements of Design Space Analysis," *Human-Computer Interaction*, vol. 6, pp. 201-251, 1991.
29. T. Moran and J. Carroll (Eds.), *Design Rationale: Concepts, Techniques, and Use*. Hillsdale, NJ: Lawrence Erlbaum Associates, 1996.
30. H. Muñoz-Avila, D. W. Aha, L. A. Breslow, D. S. Nau, and R. Weber, "Integrating Conversational Case Retrieval with Generative Planning," *Proc. 5th European Workshop on Case Based Reasoning*, Trento, Italy, pp. 322-334, 2000.
31. D. Nau, Y. Cao, A. Lotem, and H. Muñoz-Avila, "SHOP: Simple Hierarchical Ordered Planner," *Proc. 16th International Conference on Case-Based Reasoning*, Stockholm, pp. 968-973, 1999.
32. L. Osterweil, "Software Processes are Software Too," *Ninth International Conference on Software Engineering*, Monterey, CA, pp. 2-13, 1987.
33. M. C. Paulk, B. Curtis, M. Chrissis, and C. V. Weber, "Capability Maturity Model, Version 1.1," *IEEE Software*, vol. 10, pp. 18-27, 1993.
34. S. Sutton, "The Role of Process in a Software Start-Up," *IEEE Software*, vol. 17, pp. 33-39, 2000.
35. L. G. Terveen, P. G. Selfridge, and M. D. Long, "Living Design Memory' - Framework, Implementation, Lessons Learned," *Human-Computer Interaction*, vol. 10, pp. 1-37, 1995.