

E-GOVERNMENT THROUGH PROCESS MODELING: A REQUIREMENTS FIELD STUDY

Norman K. Sondheimer, Leon P. Osterweil, Matthew P. Billmers, Joel T. Sieh, Bruce B. Southard
Electronic Enterprise Institute, The University of Massachusetts Amherst
Amherst, MA USA
sondheimer@som.umass.edu

ABSTRACT

e-Government offers its constituency the hope of engaging in government interaction at any time from any place. This has been slow to materialize. We believe this is due in large part to the complexity of government processes. This paper reports experimental field study evidence that a rigorously defined process modeling language can accurately map this complexity. It introduces Little-JIL, a rigorously defined process modeling language. It then reports experience using this language to capture government processes for an e-Government system to allow online license renewals for the government of Massachusetts. These processes were previously described using Use Case methodology. The errors and shortcomings identified provide opportunities for a more correct and efficient implementation of these processes. The paper concludes with a proposal for improved e-Government development methods.

KEYWORDS

e-Government, Process Modeling, Field Study, Use Cases

1. INTRODUCTION

e-Government offers its constituency the hope of engaging in government interaction at any time from any place. This has been slow to materialize (Fountain 2001). We believe this is due in large part to the complexity of government processes. This paper reports experimental evidence that a rigorously defined process modeling language can accurately map this complexity.

Government processes involve systems of people, organizations, computational systems and legal constraints. Even the most routine processes can be very complex. The driver's license renewal processing may appear simple when the applicant encounters little difficulty, but in actuality the process must allow for dozens of special considerations such as clerical errors, withdrawal and resubmission of the application, and handling issues relating to payment. Hidden from the applicant are the large numbers of activities that take place behind the scenes to ensure accurate processing. Additionally, most existing processes are defined in natural human language, if they are defined at all. This leaves open room for misinterpretation that can lead to process inefficiencies and errors. Because of this, a considerable amount of research has been focused on the development of languages and formalisms to define processes (Katayama 1989; Kaiser et al. 1993). This work has served to emphasize both the difficulty and the importance of research in this area.

We proposed in earlier work that processes are a form of software that is amenable to programming using appropriate languages (Osterweil 1987; Osterweil 1997). We suggested that processes can and must have well-understood requirements, which can then be used as the basis for the design and coding of processes meeting those requirements. Our success within these lines of research has led us to believe that these approaches can be of substantial value in capturing government processes. This work leads us to a hypothesis:

Process Modeling Hypothesis: e-Government processes can be represented clearly, completely, and precisely through rigorously defined modeling formalisms.

The evidence presented in this paper shows the benefits that can be achieved through modeling with an appropriate formalism. We first introduce a rigorously defined process modeling language. We then report our experience using this language in capturing government processes for an e-Government system. The errors and shortcomings identified provide opportunities for a more correct and efficient implementation of these processes. We end the paper with a proposal for improved e-Government development methods.

2. THE PROCESS PROGRAMMING LANGUAGE

To test our hypothesis, we employed a process modeling language designed to capture the characteristics and properties needed for the effective representation of complex man-machine processes. The importance of explicit process representation has been recognized in such diverse domains as manufacturing, where product quality is commonly attributed to process quality; software development, where there exists a strong emphasis on software development process studies; management, where there has been a strong emphasis on business process reengineering; and, recently, data mining, where there is an effort to define a cross-industry standard process for analyzing large data sets. We contend that process representation is no less central in e-Government.

Many languages and diagrammatic notations (ICSP1 1991; ICSP4 1996) have been evaluated as vehicles for defining complex processes. Osterweil (Osterweil 1986) suggests that processes be defined using a procedural language (Kadia 1992; Sutton Jr. et al. 1995) and be used to specify how tools are to be integrated to support software development. More recently, workflow (Paul et al. 1997; WACC-99 1999) and electronic commerce (Grosz et al. 1999) communities carried out similar research. This work has shown that various types of diagrams aid human understanding of processes, and various types of languages can provide the semantic rigor needed to support the verification and executability of processes to varying degrees of certainty. However, no proposals are sufficient to meet all of the challenges of clear, precise, detailed process definition. The principal failings of these earlier approaches include inadequate specification of exception handling, lack of resource management, limited concurrency control and inadequate support for artifacts.

This work has indicated that the central questions of process representation concern how to build languages that are precise and detailed, while still remaining human-accessible. Central to this is the identification of both high-level, domain-specific abstractions and visual depictions that are clear and intuitive. Osterweil has studied these issues for over 15 years, having produced a sequence of process definition languages (Osterweil 1986; Sutton Jr. et al. 1995; Sutton Jr. and Osterweil 1997). This work led to the Little-JIL process programming language (Cass et al. 2000). We believe that we have in Little-JIL an example of a language that incorporates a promising set of such abstractions and appropriate depictions. Thus, we propose to use Little-JIL as a basis for our study of the properties and characteristics necessary for a language to be an effective vehicle for the definition and analysis of government processes.

Little-JIL was developed to coordinate software development processes (Wise 1998). It has additionally been used to define complex processes in robotics, data mining, e-commerce and complex data analysis processes (Jensen et al. 1999). These process definitions have generally been clear, detailed, and precise. Thus, we will use Little-JIL as the starting point in understanding what is required to effectively specify e-Government processes. The current version of Little-JIL incorporates multiple features that make it well suited for representing such processes. Space limitations prevent presentation of full details about Little-JIL (the details can be found in (Wise 1998)). We instead suggest key language features through presentation of an example.

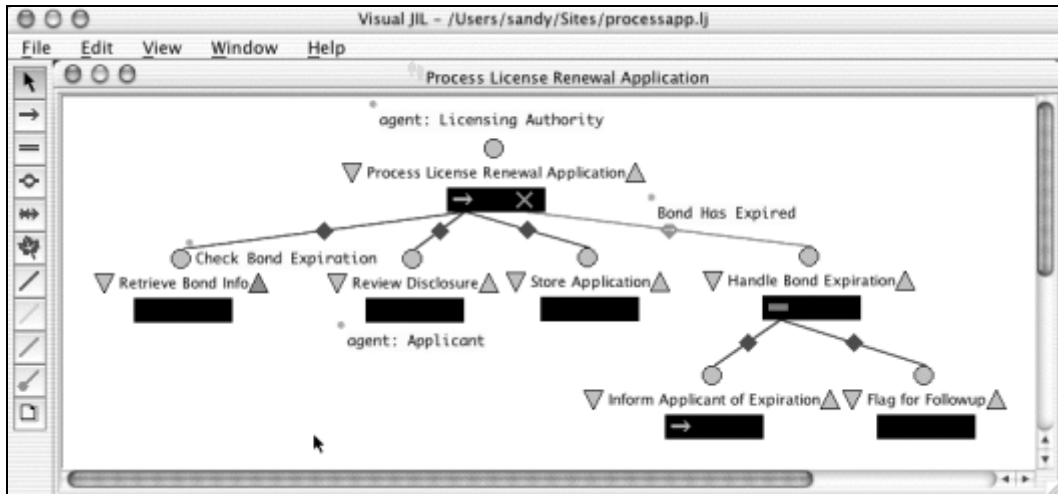


Figure 1: Example of Little-JIL process definition

The process shown in Figure 1 is based on one of the license renewal processes for the Commonwealth of Massachusetts (Sondheimer 2002). Process steps in Little-JIL are represented visually by a step name surrounded by several graphical badges that represent aspects of step semantics. (Note, most data declarations have been omitted from the figure for brevity.) The left-most element in the bar below the step name indicates how substeps are to be executed. In Figure 1, for example, the top-level step, Process License Renewal Application, uses an arrow as a sequential badge, indicating that substeps are executed in order, left to right. In contrast, the Handle Bond Expiration step shows a parallel badge (the two parallel lines), indicating that the substeps may be performed simultaneously.

Each Little-JIL step has an execution agent. By specifying the agent for each step in a process, Little-JIL explicitly identifies the participants in the process, and their responsibilities. This is a key vehicle for specifying the precise ways in which participants are to coordinate their activities. In the process in Figure 1, there are two agents: the licensing authority (specified at the root step and as the default for all its descendants) and the applicant (explicitly stated). Thus, the only part of the renewal process that the applicant must perform is Review Disclosure.

A Little-JIL step may also have a prerequisite and/or post requisite, represented by triangles on the left and right of the step name, respectively. The body of the requisite is a separately specified step (not shown in our figure) possibly containing multiple substeps. This feature supports the ability to program processes that perform internal runtime checks for the validity of evolving results. When exceptions are thrown, exception handlers may handle them elsewhere in the process. Exception handlers are attached to the right side of a step bar (e.g. Handle Bond Expiration is an exception handler attached to the Process License Renewal Application step.) Exception handlers may be simple or complex subprocesses, represented by additional substeps, and may integrate with the nominal control flow in multiple ways. This affords latitude in responding to unexpected results.

Experience to date suggests that Little-JIL is effective in supporting the definition of executable, verifiable, repeatable, and adaptable man-machine processes. Thus we are comfortable using this language as the initial basis for our exploration of the semantics and features most effective in a language for defining e-Government processes. Documentation, as well as Little-JIL and associated tools, is available for distribution on the Internet at <http://laser.cs.umass.edu/>.

3. THE FIELD STUDY

The United States Commonwealth of Massachusetts Office of Consumer Affairs and Business Regulation (OCA) is responsible for the administration of hundreds of various types of licenses, permits and registrations (henceforth called “licenses”). OCA is chartered to develop an online license renewal (OLR) system.

Hundreds of types of licenses are projected to be regularly renewed in Massachusetts in areas from banking to healthcare. Given the hundreds of licenses expected to be renewed, coupled with the approximately 500,000 current licensees, the potential exists to provide service to a significant number of Commonwealth citizens and businesses through the implementation of an OLR function on the Mass.gov portal. The hundreds of licenses to be implemented also means some substantial challenges in achieving this vision.

The authors acted in a consulting capacity to OCA in reviewing and assessing agency business processes as part of the Commonwealth of Massachusetts' OLR project requirements gathering phase. OCA chose the UML Use Case formalism as its vehicle for requirements specification. Use Cases use natural language, e.g., English, to list the steps of processes. They attempt to capture the main flow of a process, as well as exceptional behavior and other requirements concerns. They are very flexible and can be easily applied at different levels of specificity (Kulak et al. 2000). OCA's effort focused on producing Use Cases for the OLR application. They interviewed subject-matter experts and management and went through numerous revisions of material with agency teams and OLR Steering Committee members. These Use Cases documented the many agency business processes. The OCA began forwarding these processes to UMass as they were completed.

We began our part of the project by surveying the Use Cases we received. We then selected a representative set of those Use Cases and translated them into our process modeling language, Little-JIL. This allowed us to utilize Little-JIL's expressive power and its rigorous semantics to reveal any points of concern in the Use Case specifications. It also forced us to examine specific aspects of the Use Case definitions, highlighting possible inconsistencies in the content. We focused on a total of eight Use Cases that described seven licensing processes belonging to two agencies, the Alcoholic Beverages Control Commission (ABCC) and the Board of Registration in Medicine (BORIM). These processes covered a broad range of activities related to license renewal, including committee interaction, management oversight, interaction with outside agencies, and manipulation of real-world artifacts. They represented approximately 20% of the complete set of processes for the two agencies, required 45 pages in Use-Case form, and were translated into a total of 52 Little-JIL diagrams. To achieve a broader sample, we proceeded to identify the general approach we had applied to translate the Use Cases into Little-JIL and applied that approach to the rest of the ABCC and BORIM Use Cases without performing the entire translation. This allowed us to identify and further highlight possible inconsistencies. We then reviewed 19 more of the 43 Use Cases provided and classified the irregularities we found into three groups. The classifications are as follows:

Actor Structure: We found five actor inconsistencies during our experimentation. The omitted actors tended to play a non-central role in the use cases in which they appeared, and their tasks tended to be hidden in the center of the process. The ABCC use case "Process Ship License Renewal" illustrates this problem. It describes the renewal process for a license permitting sale of alcoholic beverages onboard. In the latter parts of the process specification, "interested parties", defined as, "all persons holding a beneficial interest in the applicant's business, including but not limited to: owners, partners, proprietors, officers, directors and stockholders," need to sign off on a document before the process can proceed. These "interested parties" were omitted from the actor definition, which lists the ship owner/operator (i.e., the applicant) as the only actor.

Problems like the one above arise because actor requirements for a particular Use Case are specified in natural language at the beginning of the Use Case. Little-JIL instead binds an actor (called an agent in Little-JIL notation) to each step of the process. In either case, for the process to be useable, an actor must be assigned to do the work of each step. During our experimentation we found that the looser control of actor specification in Use Cases allowed the requirements engineers to omit actors unintentionally. Little-JIL corrected this problem; during the creation of the process, and whenever it was changed (to fix an error or add features), the Little-JIL formalism required an actor binding for each step in the process that was changed or added. This prevented the accidental omission of these actors, no matter how small their role in the process.

As previously noted, we used our experience with Little-JIL when manually analyzing the BORIM and ABCC Use Cases. Methodically investigating each step in a Use Case while considering actor specifications and other required information as Little-JIL demands helped us locate and correct similar situations in which actor definitions were ambiguous or incomplete.

Analyzing the Use Cases in the Little-JIL mindset also revealed an interesting issue in the completeness of the actor specifications across the BORIM Use Cases. The Use Cases each dealt with one actor, and

limited the steps they described to those dealing with just that actor. Because of this, they appeared to model the user interface instead of the entire license renewal process. This is a misuse of the Use Case methodology. An example of this problem is as follows: The process requires the submittal of all documents regarding a criminal indictment. The Use Case models a single step with the description, “The User is presented with the message, ‘You must arrange for your lawyer or the court officer to submit copies of the indictment, complaint and judgment or other disposition in any criminal proceedings in which you were a defendant.’” A proper Use Case specification would include the court officer as an actor and provide steps for that actor to follow. Actor specification is easier when these extra steps are left out of the process, but one of the goals of the requirements specification is completeness. This is a misuse that is less likely to occur in Little-JIL because unlike Use Cases, Little-JIL contains data flow mechanisms that model the passing of artifacts such as the court documents being submitted by the court officer. This extra expressive power prompts the requirements engineer to recognize that the artifact needs to flow in from an actor other than the applicant. Adding the court officer to the process and factoring in the related work follows naturally.

Parameter Flow: We found seven errors in Use Cases caused primarily by lack of parameter flow information. These problems resulted because UML Use Cases provide no way to specify the flow of data parameters among their steps. However, the actual Use Cases provided by OCA extended the standard methodology by including a table of data artifacts required by each Use Case. We believe this helped mitigate some of the inadequacies of the methodology, but also highlighted the lack of consistency in their documents.

Little-JIL includes a type-safe parameter flow system. This system requires process programmers to clearly define what data are required for the completion of each step, what data each step produces, and how the data flow among the steps.

In our experimentation, we selected a number of the Use Cases and attempted to fill in all the missing parameter flow information. In the process, we discovered that for each Use Case, some step descriptions implicitly or explicitly referred to data that was not contained in the associated data table. We also observed other, slightly more complex problems in the Use Case that involved collections of data that were inconsistently and at times incorrectly passed or accessed. The rigor that Little-JIL provides helped us to detect data flow errors in the processes we analyzed which may have gone unnoticed otherwise.

Lack of Process Component Reuse: We found three significant redundancy issues in the Use Cases we examined. Since Use Cases are basically lists of steps written in natural language, finding commonalities among them and abstracting the commonalities to a higher level is time consuming and difficult. Similar groups of steps can be easily missed when searching for commonalities because they lie scattered throughout, or placed in different orders in multiple Use Cases. Steps with the same functionality may also have different wordings. Missing these commonalities can lead to errors as the requirements document undergoes changes and result in a confusing document filled with unnecessary redundancies.

Little-JIL solves this problem because it is a hierarchically structured process language. It encourages process programmers to identify logical clusters of related process steps and then group those steps together under one higher-level step. This increases readability and allows for reuse of steps or groups of steps from any level of the process. When the same unit of work is required be performed in another context, the requirements engineer can simply insert a reference to the related step in that context. This reference indicates that the procedure for performing that step in the process has already been detailed elsewhere. Instances in which we added higher-level steps, indicating that several steps from a Use Case could comprise a more general unit of work, are abundant in our Little-JIL requirements documentation.

After a time, performing this identification becomes second nature to a requirements engineer familiar with Little-JIL. It is with this experience that we examined the larger subset of OCA’s Use Cases outside of those we translated.

An example of one of these inconsistencies occurred across many of the BORIM Use Cases we reviewed. Each medical license renewal requires the applicant to answer a select set of questions, depending on the type of license being renewed and the applicant’s history. For some licenses, the applicant can be expected to answer more than 30 questions, some questions requiring additional information when answered in a certain way. Predictably, there was much redundancy in these questions across Use Cases, with slight ordering or wording differences. In the Use Cases we translated, grouping these questions according to their subject matter (personal information, insurance information, malpractice questions, etc.) was straightforward with Little-JIL. Identifying the redundant questions in the other Use Cases we examined was no worse.

The problems identified and corrected in our experimentation, as shown above, demonstrate that using Little-JIL to specify requirements prevents common errors that the Use Case methodology does not prevent. These results lead us to conclude that Little-JIL, a formalism with richer semantics than the UML Use Case, is useful for crafting clear, precise, and complete e-Government requirements documents. Our experimentation suggests that Little-JIL documents are larger than Use Case documents and take longer to specify. We believe this is because of the increased rigor required, and consider the benefits in clarity and completeness to be adequate compensation.

4. CONCLUSIONS AND NEXT STEPS

Our research supports our Process Modeling Hypothesis. It indicates that complex government processes can be effectively modeled by a rigorous process modeling language. Field studies addressing other types of government processes such as taxation and revenue are currently being planned. The process modeling language will be applied to the later stages of the OCA Systems Development Life Cycle and in other pertinent studies. This represents the technology side of realizing e-Government.

However, the computational issues in modeling government processes are shaped by the organizational dynamics that must be incorporated into e-Government. An example is the conflict between the US Customs Office and the interagency board charged with implementing the International Trade Data system (Fountain 2001). The board could not obtain the cooperation of the Customs Office. This lack of cooperation is common (Wilson 2000). We believe the crucial issue in any e-Government effort is building this cooperation.

Cooperation comes from trust which the Oxford English Dictionary (OED) defines as: “confidence in some quality or attribute of a person or thing”. If the stakeholders can trust that their input is an essential factor in the design of an e-Government system, they are more likely to cooperate. To develop and maintain that trust, methods must be found to obtain stakeholder input and to verify that the input is considered on an ongoing basis as the e-Government system evolves. We refer to these methods as a system for “Trust Resource Management (TRM).” We propose to use explicit process models as the basis for TRM and its use in e-Government.

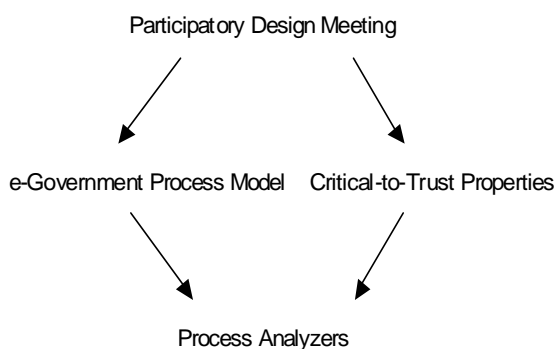


Figure 2. Trust Resource Management Flow

Figure 2 shows our proposed system. Participatory design meetings bring key stakeholders together. They define the e-Government process models and define the properties these processes must meet. Analyzers then determine if the models satisfy the properties. If not, modification of either the e-Government process or the requirements on it must occur. As the e-Government service develops, the model and properties will evolve and the analyzers will be reapplied.

Participatory Design Meetings: Our premise is that much government work can be characterized as the creation, execution, and

modification of processes. In the course of automating government, clarifying these processes and their properties to all stakeholders through the use of appropriately derived formal meetings will increase stakeholder trust, thereby making e-Government more effective.

Sophisticated computational and behavioral work is required to structure these interactions. In the US, a recognized approach to addressing the technical needs of system stakeholders is Joint Application Design (JAD) (Wood and Silver 1995; Yatco 1999). In order to address e-Government organizational design development, our method, Participatory Design Meetings, will extend JAD sessions to focus on modeling and adjusting the behavior of the systems development process to allow for trust-enhancing decision making processes. The heart of our system will be motivated by decision-making rules using a process developed by

Victor Vroom (Vroom 1973; Vroom et al. 1998). The key will be to decide how to engage stakeholders in making design decisions.

Analyzers: Participatory Design Meetings will establish proposed e-Government processes and elicit the Critical-to-Trust properties they must meet. Stakeholders must be convinced the processes realize the properties. The clear representation of e-Government processes through rigorously defined modeling formalisms must be combined with analysis tools to achieve this. Our previous research indicates that the definitiveness of analyses must ultimately depend upon the rigor with which the processes are defined. One key reason for the requirement in our proposal to define government processes in a language with formal semantics, which Little-JIL has, is to provide a stronger basis for automated analysis and verification of these processes. In particular we propose to apply a range of analysis and testing techniques to the government processes that we will define in order to determine how effectively we are able to demonstrate that these processes satisfy the Critical-to-Trust properties specified by stakeholders.

We propose to apply two complementary approaches: dynamic testing and static analysis. In earlier work (Osterweil 1996) we described the specific ways in which these two approaches are complementary. Dynamic testing monitors process execution and is effective in detecting deviations from desired or mandated behaviors. Static analysis is aimed at demonstrating the absence of defects and undesirable behaviors, but only for limited classes of defects. In the next stage of our work, we will study and evaluate both dynamic testing and static analysis approaches to understand their relative effectiveness in enhancing trust.

Our experience has shown that rigorous process modeling isolates errors that inhibit the effectiveness of e-Government. Without process modeling, e-Government struggles to offer to its constituency engagement in government interaction at any time from any place. With it, the foundation exists for a system that can establish and maintain trust among all e-Government stakeholders.

ACKNOWLEDGEMENT

This material is based upon work supported by funds from the Commonwealth of Massachusetts' Information Technology Division and the National Science Foundation under Grant No. EIA-223599. We would like to thank Val Asbedian, Tim Healy and Tom Price for supporting and helping supervise the work from the Government side; Tony Butterfield, Bob Marx and David Su from the University of Massachusetts for many fruitful discussions on Participatory Design Methods; and Annie Kelly for her stalwart editorial support. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the Commonwealth of Massachusetts or the National Science Foundation.

REFERENCES

- Cass, A. G. et al., 2000. Little-JIL/Juliette: A Process Definition Language and Interpreter. *International Conference on Software Engineering*, Limerick, Ireland, pp. 754-758.
- Fountain, J., 2001. *Building the Virtual State: Information Technology and Institutional Change*. The Brookings Institution, Washington, DC.
- Grosf, B. et al., 1999. A Declarative Approach to Business Rules in Contracts: Courteous Logic Programs in XML. *ACM Conference on Electronic Commerce (EC 99)*, Denver, CO, pp. 68-77.
- ICSP1, 1991. First International Conference on the Software Process, Redondo Beach, CA.
- ICSP4, 1996. Fourth International Conference on the Software Process, Brighton, UK.
- Jensen, D. et al., 1999. Coordinating Agent Activities in Knowledge Discovery Processes. *International Joint Conference on Work Activities Coordination and Collaboration*, San Francisco, CA, pp. 137-146.
- Kadia, R., 1992. Issues Encountered in Building a Flexible Software Development Environment: Lessons from the Arcadia Project, *Fifth ACM SIGSOFT Symposium on Software Development Environment*, Tyson's Corner, VA, pp. 169-180.

- Kaiser, G. E. et al., 1993. A Bi-Level Language for Software Process Modeling. *15th International Conference on Software Engineering*, Baltimore, MY, pp. 132-143.
- Katayama, T., 1989. A Hierarchical and Functional Software Process Description and its Enaction. *11th International Conference on Software Engineering*, Pittsburgh, PA, pp. 343-353.
- Kulak, D. et al, 2000. *Use Cases: Requirements in Context..* Addison Wesley Professional, New York, NY.
- Osterweil, L. J., 1986. *Software Process Interpretation and Software Environments*. Department of Computer Science, University of Colorado, Boulder, CO.
- Osterweil, L. J., 1987. Software Processes are Software, Too. *Ninth International Conference on Software Engineering*, Monterey, CA, pp. 2-13.
- Osterweil, L. J., 1996. Perpetually Testing Software. *Ninth International Software Quality Week*, San Francisco, CA.
- Osterweil, L. J., 1997. Software Processes Are Software, Too, Revisited. *19th International Conference on Software Engineering*, Boston, MA, pp. 540-558.
- Paul, S. et al., 1997. RainMan: A Workflow System for the Internet. *Usenix Symposium on Internet Technologies and Systems*. Monterey, CA, pp. 159-170.
- Sondheimer, N. K., 2002. *Final Report: Application of Process Modeling to Online License Renewal*. Electronic Enterprise Institute, University of Massachusetts, Amherst, MA.
- Sutton Jr., S. M. et al., 1995. APPL/A: A Language for Software-Process Programming. *ACM Transactions on Software Engineering and Methodology* 4(3): pp. 221-286.
- Sutton Jr., S. M. and Osterweil, L. J., 1997. The Design of a Next Generation Process Language. Joint Sixth European Software Engineering Conference and the Fifth ACM SIGSOFT Symposium on the Foundations of Software Engineering, Zurich, Switzerland, pp. 142-158.
- Vroom, V.H., et al., Participative Leadership, in *Leadership: The Multiple-Level Approaches: Classical and New Wave*, F. Dansereau and F.J. Yammarino, Editors. 1998, JAI Press: Stamford, CT. p. 145-189.
- Vroom, B.H., A New Look at Managerial Decision Making. *Organizational Dynamics*, 1973(2): p. 70-71.
- WACC-99, 1999. *Work Activities Coordination and Conference*, WAC, San Francisco, CA.
- Wilson, J., 2000. *Bureaucracy: What Government Agencies Do and Why They Do It*. Basic Books, New York, NY.
- Wise, A., 1998. *Little-JIL 1.0 Language Report*. Department of Computer Science, Univesity of Massachusetts, Amherst, MA.
- Wood, J. and Silver, D., 1995. *Joint Application Development*. Jossey-Bass, San Francisco, CA.
- Yatco, M. C., 1999. *Joint Application Design/Development*. University of Missouri - St. Louis, St. Louis, MO.