Mixed-Initiative Management of Dynamic Business Processes

Zachary B. Rubinstein
Department of Computer Science
University of New Hampshire
Durham, NH 03824-3591
zack@cs.unh.edu

Daniel D. Corkill
Department of Computer Science
University of Massachusetts
Amherst, MA 01003-4610
corkill@cs.umass.edu

Abstract— **Managing and participating in complex, dynamic business processes is difficult due to their inherent uncertainty, which undermines the predictability necessary for efficient planning and execution. Effective management of these processes hinges on the ability of the manager to recognize unanticipated difficulties in the process execution, determine the causes of the anomalies, and implement remedies. Current process-management approaches respond reactively to process dynamics, if they deal with them at all.**

In this paper, we present the ProME process-management environment, focusing on how human process managers and participants interact with a dynamic, on-line model of executing dynamic processes to proactively manage and operate in dynamic business processes. We show how having the best information available about a process *and its future* can provide managers with the time needed to detect and understand impending process anomalies and to develop and implement effective interventions. Furthermore, we show how enabling managers to update the model of executing processes and having the effects of those modifications be pushed to the relevant participants reduces the time it takes to implement remedies. ProME was used in a commercial product for managing design processes in the automotive and aerospace industries.

Keywords: dynamic-process management, decision-support systems, mixed-initiative systems, proactive resource allocation and scheduling, business-process representation and execution.

I. Introduction

Effective management and execution of business processes are playing an increasingly important role as economic pressures force businesses and government to work faster and with fewer resources. As operations are streamlined, difficulties in managing the details of important business processes become critical limits to increased performance and cost savings.

Managing complex business processes has always been difficult, but the increased interaction among processes (through shared resources and results) and increased concurrency of activities within processes (to complete processes sooner) now leave process managers with less time to make decisions and significantly expands the ramifications of those decisions. Sometimes, the process itself cannot be fully elaborated until it is underway. Today's managers must routinely investigate and evaluate process status, estimate progress in relation to deadlines, handle exceptions and resource problems, worry about potential downstream issues, and change the process structure and details accordingly. For example, key resources may need to be reassigned or may become unavailable, information and results may not arrive when expected, tasks may take more (or less) time than expected, results may be surprising and suggest other activities, and so on.

Due to process dynamics and uncertainty, *these events cannot be anticipated nor can contingencies be established for them beforehand*. Therefore, a significant contributor to the effectiveness of business-process execution is the ability of managers to make appropriate and timely process assessment, guidance, and adjustment decisions. These adjustments include adapting and modifying previously defined process definitions to fit the current situation and the creation of new process definitions on the fly. Information technologies have addressed pieces of the process-management problem [1]. For example, project-planning and process-modeling tools can be used to engineer business processes off-line, with workflow systems automating their execution. Project management and reporting tools document how execution has progressed. These technologies work well enough with routine, steady-state processes, but they fail to address the problems of managing complex, dynamic processes that are present in many real-world situations. What is needed are decision-support technologies that keep process-management activities ahead of the rate of change in the processes.

Recently, business activity monitoring (BAM) [2] technologies have been developed that provide near real-time access to critical process-status indicators. Although these technologies improve decisionmakers' understanding of the state of processes, they do not provide knowledgeable estimates of downstream process activities and potential problems that are required in order to make proactive management decisions. To break this time barrier, we have developed a *live-representation* process-management approach [3] that maintains a real-time view of past, present, and anticipated process activities and resourcing. Changes resulting from process dynamics are directly reflected in the live representation so that, at any point in time, the latest information about process status *and* downstream expectations is available. Managers can directly manipulate the representation to change process structure and execution behavior. These changes are immediately

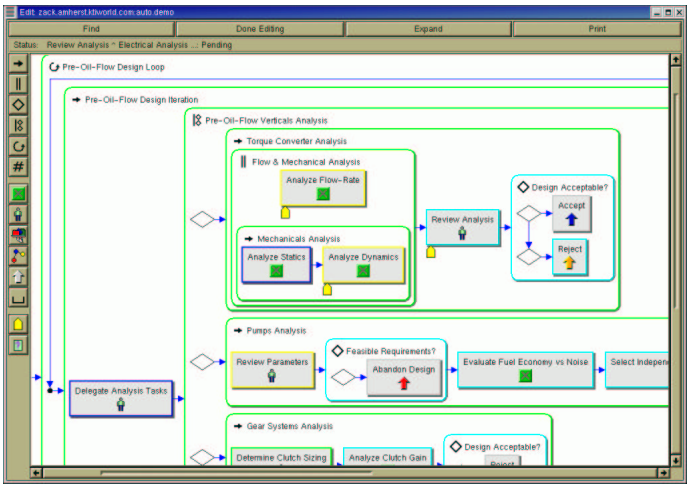

Fig. 1. An Executing Transmission-Design Process

propagated throughout the environment, keeping managers and process participants in sync with process changes and downstream expectations.

II. ProME: Process Management and Execution

ProME is an advanced process-management decision-support environment that implements the live-representation approach. It has been used commercially in a product for managing complex automotive and aerospace design processes [4]. Automotive and aerospace design involves dynamic processes that use limited and highly expensive physical and personnel resources and require the careful coordination of diverse organizational units.

ProME addresses the reality of dynamic process management by facilitating continual proactive process guidance and adjustment as well as on-the-fly process definition and elaboration. Consider the following simplified excerpt from an automotive transmission design process (Figure 1). This excerpt shows a portion of the initial design loop in which high-level design parameters are determined by a set of analysis tasks performed in parallel during each iteration. Some of the possible analyses are torque converter analysis, pumps analysis, gear systems analysis, electrical analysis, preliminary manufacturability analysis, and preliminary cost analysis. Depending on the state of the high-level design, only the relevant analyses are performed. Any of the executed analysis tasks can invalidate the current design, requiring that it be modified and that the new design be submitted to the relevant analyses for verification in the next iteration of the loop. This design, test, and modify loop is repeated until a feasible high-level design is produced.

When the process is first scheduled, ProME must estimate how many iterations of the loop are likely to be executed. Using knowledge specified in the process definition, assume that two iterations of the loop are needed and that, at this level of design, only the torque converter, gear systems, and the electrical analyses are likely to be performed. ProME uses these expectations to generate a process plan of the tasks that are expected to execute. In this case, the process plan includes the task for obtaining the new vehicle specifications, two instantiations of the tasks needed by each of the analyses expected to execute (one for each expected iteration), and the tasks needed to refine the design.

Once the process plan is constructed, an initial schedule is generated with resource reservations. As the process executes, the initial tasks for the analyses begin to execute. When the review analysis task for the torque converter analysis executes, it returns a failed analysis. At this point, the current design is not feasible and must be modified. In addition, the other analyses tasks for this iteration are no longer necessary, and ProME aborts the tasks that are executing and cancels those that are scheduled. The resources associated with the tasks are released and the remaining downstream tasks in the process are rescheduled. By integrating process representation, execution, and scheduling, ProME is able to capture the process designer's intent to abandon all other analyses if one fails, detect when that state arises, reconcile the process plan and schedule to the changes, and keep both the manager and the process participants apprised of the current and expected state of the executing process.

Continuing with the example execution of the process, the second iteration of the design-test-modify loop begins. In determining which analyses are needed, it turns out that a previously unexpected analysis, pumps analysis, is now required. Through the execution model, ProME automatically detects this new requirement and inserts the tasks that comprise the pumps analysis activity into the appropriate places in the process plan. In scheduling those tasks, a special test compressor is needed. However, it turns out that this compressor is being used by another process at the desired time, introducing delay into the schedule. The delay causes the schedule to extend beyond a specified deadline, and the manager is notified. Because the manager is informed of the deadline violation at the point of adding the new analysis, she is able to search for alternatives prior to the actual occurrence of the problem. In this example, she is able to borrow a similar compressor from another division. When this temporary availability of a new compressor resource is added to the system, its availability triggers the rescheduling of the delayed tasks that can use that resource. Once rescheduled, the tasks are no longer delayed, and the deadline is no longer violated. This type of proactive decision cannot be made without detailed downstream forecasting. Furthermore, an automated managerial intervention cannot be made unless the system supports on-the-fly modification of processes and resources.

The following five capabilities are pivotal in ProME:

1) **Complete process representation**—The approach begins with a knowledge-intensive definition of the dynamic process, with sufficient detail to allow automated execution and to make reasonable expectations of downstream activities.
2) **Direct execution**–The process definition is instantiated and executed for each process so that the representation

matches exactly what is happening as the process is executed. Direct execution is important for validation of the process representation and to ensure that on-the-fly modifications to the representation will be reflected in the executing process.

3) **Integrated downstream forecasting**—Dynamic scheduling of downstream activities and resources are needed to allow time for proactive intervention. The scheduling must be tightly integrated with the direct execution of process representations and must balance the need for process flexibility with the need to maintain process stability whenever possible. It must be able to make timing and resourcing decisions across multiple processes using criteria provided by process designers and execution managers.
4) **Presentation of execution status, history, and expectations**—The latest details of process state and downstream expectations must be presented, tailored for understandability, and relating directly to the process representations, to focus attention on problem areas.
5) **On-the-fly process modification**—Managers must be able to change the process structure and execution details of executing processes in response to unanticipated situations and problems.

In concert, these capabilities allow managers to quickly comprehend current process status and potential downstream problems, make proactive interventions, and have those interventions immediately reflected in the executing processes.

III. Process Representation

ProME's capabilities place a diverse set of requirements on its process representation. First, the representation must be fully executable. Everything that needs to be known for process execution must be represented: software, resources, data, interfaces, etc. Second, the representation must include the knowledge needed to perform downstream scheduling. This knowledge includes contextual expectations of each task's duration and resource requirements, likely conditional and looping choices, and so on. Third, the representation must be expressive and intuitive to process managers and participants. The representation should enable rapid understanding of what is happening and will happen and support both abstract and detailed presentation. Fourth, the representation must be amenable to in-process structural modification, including moving entire process subtrees within the process definition.

A variety of process representations have been developed [5, 6, 7, 8, 9, 10]. While each represents a variety of constraints and semantics of processes, none of them captures the control knowledge necessary for predictive, downstream scheduling. For example, scheduling processes with loops and conditionals requires estimating the number of iterations a loop may take and the specific branches to be taken. Although workflow management systems track the progress of process execution and facilitate the passing of data among tasks and the invocation of automated tasks [11], they do not schedule downstream activities and resource allocations [12]. The recent BPML specification for describing business processes within Web services [13] has been augmented with richer control semantics, but it is designed for execution and modeling, not dynamic scheduling. Lastly, these representations are not designed to quickly present process status and information to non-technical users—an important criteria in the live-representation approach.

ProME's process definition represents a family of potential process plans that are specified as a hierarchy of container and non-container tasks (Figure 1) [14]. Non-container tasks can be thought of as the detailed activities in a process, while container tasks form the control structure in which the activities are embedded. Container tasks have control semantics that are used when generating the process plan, but they are not explicitly part of the process plan. The container task types are: task structure, serial, parallel, branching, parallel-branching, looping, and quantified (a runtime replication of process substructure executed in parallel). Non-container tasks do not have child tasks and become part of the process plan. Examples of non-container task types are: executable (an arbitrary computation performed by the process manager), user assistant (a browser based interface to a human participant), task valet (a program launched by a fully automated desktop service), placeholder (reserves resources and time for a yet-unspecified detailed process-definition fragment), component (invokes another process definition as a subprocess), and return (provides non-local termination/aborting of a process subtree).

Each task in a process definition includes behavioral knowledge that is used in process scheduling and rescheduling. For container tasks, this behavioral knowledge controls the structure of the generated process plan. For example, when a process plan is generated for a parallel-branching task during initial scheduling, a set of scheduling-time functions corresponding to the task's children is evaluated to determine which children should be expanded and included in the process plan. Only data values that are available in advance of task execution can be used in these schedule-time predicates. When the parallel-branching task executes, a separate set of execution-time predicate functions corresponding to the task's children is evaluated to determine which children are actually executed. Non-container tasks include behavioral knowledge that determines the expected duration of the task and the possible sets of resources that can accomplish that task. The process definition is atemporal. It does not have separate objects for tasks that might be executed more than once, such as the tasks within a looping task. When the definition is first scheduled for execution, an explicit process plan with objects for all scheduled tasks is generated from the process definition.

IV. Resource Management, Scheduling and Execution

Fundamental to the live-representation approach is maintaining the best estimate of how the execution of that process will unfold. Major components of that estimate are a schedule of what activities are expected to be executed, the resources that are assigned for them, and their expected start and finish

times. The schedule for a dynamic process is a highly fluid entity that constantly changes in response to process execution, new information, and a changing environment. To generate and maintain this type of schedule, ProME uses PROTEUS, a unique distributed, constraint-based scheduler that is tightly integrated with process execution [14]. PROTEUS uses heuristics to evaluate and trigger appropriate responses to execution-time events, balancing the trade-offs among efficient resource allocation, schedule volatility, and rescheduling costs. It provides mechanisms for automatically manipulating the process plan in response to events, such as the execution of control constructs and structural edits performed by a manager. Shared resources for overlapping executions of processes require that these heuristics have an enterprise-wide perspective; i.e., they must assess the effect that individual process dynamics have on all the other executing processes that interact or share resources with that process.

For each resource (including human participants), ProME maintains a model of that resource and a schedule of commitments. These are managed by a distributed set of resource managers. Human participants are modeled as complex *multiplexing* resources that are able to choose when to perform activities within bounded assignment windows. PROTEUS ensures that there is sufficient time for participants to perform their activities, but only a macro-level schedule of due dates and estimated start times are provided to participants. Every resource can be assigned an arbitrary number of roles which it can perform. When a process activity specifies a particular set of resource-role specifications, resource managers are contacted by PROTEUS to satisfy the resource request.

The interplay between process execution and scheduling is a key aspect of the ProME architecture. Although there has been significant work done on rescheduling in response to execution dynamics—especially in the area of constraint-based schedulers, such as ISIS [15, 16], MICROBOSS [17], OPIS [18], and OZONE [19]—this work has not addressed *dynamics that involve changes to the structure* of the executing processes. There has also been scheduling work on selecting the best process plan given the current state of execution. Whether a Markov Decision Process is used, as in RTDP/ROUT [20], or the aggregation of statistical distributions throughout the process plan, as in design-to-criteria scheduling [21], this work assumes that the process definition includes all the possible paths that may be taken during execution. For the dynamic processes supported by ProME, however, the closed-world assumption cannot be made. Many decisions that affect the process structure cannot be made until execution of the process definition is underway. Furthermore, unexpected results or events may lead to changes to the process plan that were never anticipated. Since every execution may be radically different, each new execution is effectively a first-time execution that should be executed "first time best." Performing well on average is not good enough.

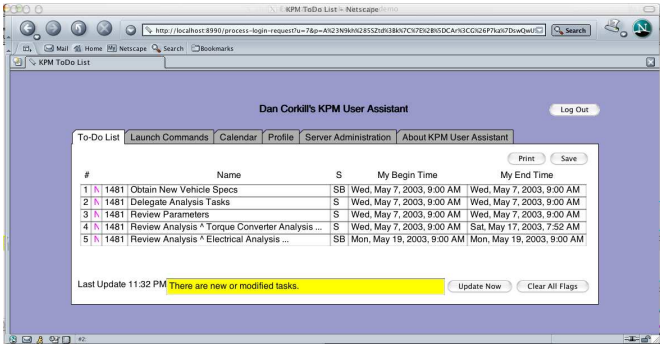

Fig. 2. To-Do List for Corkill

V. ProME At Work

To give a feel for how a process manager and participants might use ProME, we present the following simple example. Using the automotive-transmission design process shown in Figure 1, we show how ProME alerts a manager of a downstream deadline problem, how the manager uses ProME to modify the executing process, and how the participants in the process are notified of the changes. In presenting the example, we will refer to some of the major components in the ProME environment:

- **Server**—Provides process definition and execution services. Each Server maintains the on-line model of processes assigned to it and can cooperate with other Servers (within the same organization or with Servers in other entities, such as suppliers) in coordinating interactions and the use of common resources and information. Each Server also provides resource-management services for resources assigned to it.
- **Developer**—A graphical development client for creating, editing, and maintaining process definitions and libraries.
- **User Assistant**—A browser-based graphical desktop "agent" for people performing process tasks. The User Assistant notifies the user of task requests, maintains a to-do list of assigned tasks and tasks in process, allows users to launch desktop applications, and provides facilities for users to update and complete tasks. The User Assistant also includes a live calendaring facility that allows a participant to manage blocks of time that she is available as a resource for performing tasks assigned by ProME. These availability blocks can be changed on the fly and, as with other dynamic events, these changes can trigger rescheduling.
- **Execution Manager**—A graphical monitoring and control client for managing an executing process.
- **Task Valet**—A Java-based desktop proxy agent that allows a user to specify what fully-automated activities can be performed on her behalf, without user supervision.

The example begins with an executing process. As part of initiating the execution of the process, a schedule is generated for the tasks expected to execute. In this schedule, a process participant, Corkill, has been assigned several tasks. He is

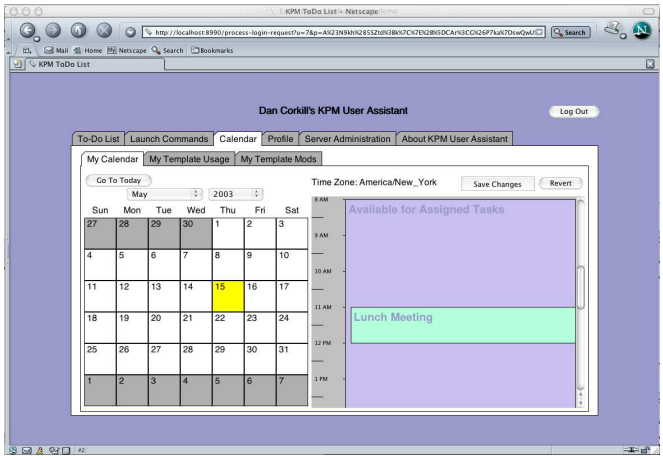

Fig. 3. Calendar for Corkill

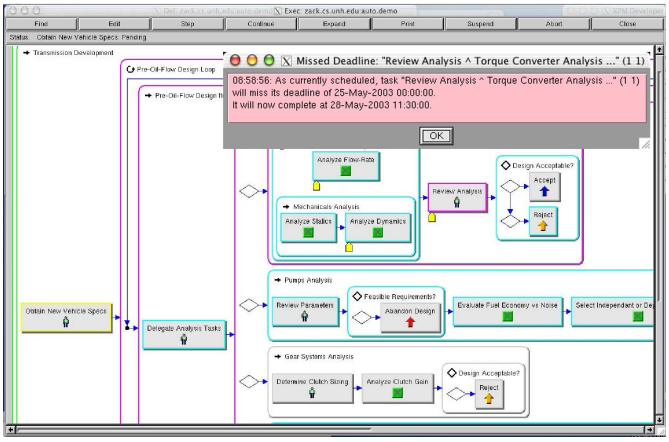

Fig. 4. Deadline Notification

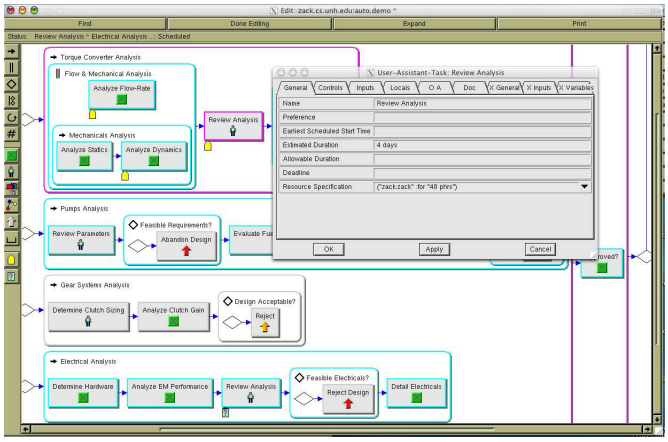

Fig. 5. Resource Change

notified of these tasks through the To-Do List on his User Assistant (Figure 2). Not only does Corkill's To-Do List contain assigned tasks that are currently active, but it also shows scheduled future tasks and the time windows within which they are expected to occur.

As a process participant, Corkill, has specified time blocks that he is available to work on ProME assigned tasks using the calendaring facility in the User Assistant. This facility keeps ProME apprised of changes in a participant's availability over time. In our example, after the process has begun to execute, Corkill decides he must take a personal day. He updates ProME by going to the appropriate day in the calendaring facility (Figure 3) and marks that day as a personal day. This informs the Server maintaining Corkill's availability of the change, and it checks all the existing reservations for Corkill during that day. The Server then notifies all Servers associated with those reservations, causing them to reschedule the affected tasks. In our example, this change causes a deadline constraint for the Review Analysis task to be violated in the revised schedule due to no other resources being available to complete the requisite tasks before the specified deadline. The manager is alerted to the scheduled violation by a pop-up notification in the Execution Manager (Figure 4).

In addition to the pop-up notification, the Execution Manager helps the manager identify the tasks that contribute to the delay due to resource contention by outlining them in purple, signifying their bottleneck status. In our example, the Review Analysis task itself is a bottleneck. The manager determines from this information that changing the resourcing on the Review Analysis task might remove the deadline violation and decides to add Zack (borrowed from another division) as an additional resource available for that task. The manager makes this modification to the executing process using the edit facility in the Execution Manager (Figure 5) and, indeed, Zack can perform the Review Analysis task in time be beat the deadline. Corkill is informed through his To-Do List that he no longer needs to participate in the Review task, and Zack is informed of the reassigned task through his To-Do List (Figure 6).

This simple example shows how a single availability change can affect a tightly scheduled dynamic process. The managerial intervention involved making an additional resource available to a bottleneck task. If resource sharing among divisions was routine, this possible remedy could be added to the process definition, in which case ProME would address the deadline violation without managerial intervention. In a more complex situation, our manager could have performed more substantial interventions, such as changing the structure of the executing process, should that have been appropriate. The Servers in ProME would handle such changes in a similar fashion, determining what scheduled tasks and resources are affected, rescheduling them, and notifiying process managers and participants as appropriate.

VI. Summary

In summary, ProME achieves the following goals:

- improves process coherence by keeping participants and managers informed of the current state and downstream expectations of processes
- enables proactive response to process and resource problems before they occur by providing a clear picture of what the processes will do in the future without

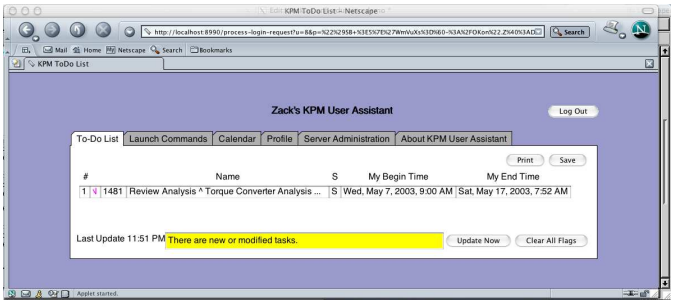

Fig. 6. To-Do List for Zack

intervention

- pushes appropriate information to participants when it is needed
- allows customization of individual activities based on process context
- supports resource allocations among processes
- supports cross-organizational collaboration processes via inter-organizational process linking and resource allocation strategies

By keeping process managers and participants informed of a process's past, present, and expected behavior, ProME enables them to make better decisions based on the most accurate information available. Furthermore, with on-the-fly modification, managers can directly and immediately implement the inevitable changes necessitated by those decisions into the model for the executing process, making the changes and the new responsibilities visible to all of the relevant participants. By providing for this critical cycle of informing participants of the current state of an executing process and of empowering them to make on-line changes, ProME reduces the time it takes to recognize and adapt to execution-time dynamics, leading to more effective process executions.

ACKNOWLEDGMENT

This work was supported, in part, by the National Science Foundation (DMI-0122173) and by internal research & development funding from Blackboard Technology Group and from Knowledge Technologies International. The views and conclusions presented are those of the authors and should not be interpreted as necessarily representing the official policies or endorsement of the supporting organizations.

We also thank Susan Lander, Kevin Gallagher, Ken Berthiume, and Suzanne Tromara for their contributions in developing and implementing the approach.

REFERENCES

[1] H. Kerzner, *Project Management: A Systems Approach to Planning, Scheduling, and Controlling*, 7th ed. John Wiley & Sons, 2001.

[2] J. Correia and N. Schroder, "BAM: A composite market view," Gartner, Inc., Tech. Rep. SOFT-WW-DP-0067, Apr. 2002.

[3] D. D. Corkill, Z. B. Rubinstein, S. E. Lander, and V. R. Lesser, "Live-representation process management," in *5th International Conference on Enterprise Information Systems*, vol. 8, Angers, France, Apr. 2003, pp. 202–208, (Also published as Technical Report 02-45, Department of Computer Science, University of Massachusetts, Amherst, Massachusetts 01003, November 2002.).

[4] S. Lander, D. Corkill, and Z. Rubinstein, "KPM: A tool for intelligent project management and execution," in *Workshop on Intelligent Workflow and Process Management: The New Frontier for AI in Business, Sixteenth International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, Aug. 1999.

[5] Workflow Management Coalition, *Workflow Management Coalition Terminology & Glossary*. http://www.aiim.org/wfmc/standards/docs/glossy3.pdf: Workflow Management Coalition, Feb. 1999, vol. WFMC-TC-1011, 3.0.

[6] R. J. Mayer, C. P. Menzel, M. K. Painter, P. S. deWitte, T. Blinn, and B. Perakath, "Information integration for concurrent engineering (IICE) IDEF3 process description capture method report," Knowledge Based Systems, Incorporated, College Station, Texas, Tech. Rep. AL-TR-1995-XXXX, 1995.

[7] J. Lee, M. Gruninger, Y. Jin, T. Malone, A. Tate, G. Yost, and other members of the PIF Working Group, "The PIF process interchange format and framework," MIT Center for Coordination Science (CCS), Tech. Rep. 194, 1996.

[8] K. Decker, "TÆMS: A framework for environment centered analysis & design of coordination mechanisms," in *Foundations of Distributed Artificial Intelligence*, G. O'Hare and N. Jennings, Eds. Wiley InterScience, 1996, ch. 16, pp. 429–448.

[9] A. Knutilla, "Process specification language: Analysis of existing representations," National Institute of Standards and technology, Gaithersburg, MD, Tech. Rep. NISTIR 6133, 1998.

[10] B. S. Lerner, L. J. Osterweil, J. Stanley M. Sutton, and A. Wise, "Programming process coordination in Little-JIL," in *Proceedings of the 6th European Workshop on Software Process Technology*, Weybridge, UK, September 1998, pp. 127–131.

[11] A. Cichocki, A. Helal, M. Rusinkiewicz, and D. Woelk, *Workflow and Process Automation: Concepts and Technology*. Kluwer, 1998.

[12] D. D. Corkill, "When workflow doesn't work: Issues in managing dynamic processes," in *Design Project Support using Process Models Workshop, Sixth International Conference on Artificial Intelligence in Design*, Worcester, Massachusetts, June 2000, pp. 1–13.

[13] A. Arkin, "Business Process Modeling Language, Version 1.0," Business Process Management Initiative (BPMI.org), Tech. Rep., June 2002.

[14] Z. B. Rubinstein, "Efficient scheduling of evolving, nondeterministic process plans in dynamic environments," Ph.D. dissertation, University of Massachusetts, Amherst, MA, May 2002.

[15] M. S. Fox, *Constraint-Directed Search: A Case Study of Job Shop Scheduling*, ser. Research Notes in Artificial Intelligence. London: Pitman Publishing, 1987.

[16] M. S. Fox and S. F. Smith, "ISIS—A knowledge-based system for factory scheduling," *Expert Systems*, vol. 1, no. 1, pp. 25–49, July 1984.

[17] N. Sadeh, "Look-ahead techniques for micro-opportunistic job shop scheduling," Ph.D. dissertation, Computer Science Department, Carnegie Mellon University, March 1991.

[18] S. F. Smith, "The OPIS framework for modeling manufacturing systems," The Robotics Institute, Carnegie Mellon University, Tech. Rep. CMU-RI-TR-89-30, Dec. 1989.

[19] S. Smith, O. Lassila, and M. Becker, "Configurable, mixed-initiative systems for planning and scheduling," in *Advanced Planning Technology*, A. Tate, Ed. Menlo Park CA: AAAI Press, 1996, pp. 235–241.

[20] J. Schneider, J. A. Boyan, and A. Moore, "Value function based production scheduling," in *Machine Learning: Proceedings of the Fifteenth International Conference (ICML '98)*, Mar. 1998.

[21] T. Wagner, A. Garvey, and V. Lesser, "Criteria-directed heuristic task scheduling," *International Journal of Approximate Reasoning,* Special Issue on Scheduling, vol. 19, no. 1, pp. 91–118, 1998.