Modeling, Simulating, and Enacting Complex Organizational Processes: A Life Cycle Approach[1]

Walt Scacchi
Information and Operations Management Department
University of Southern California
Los Angeles, CA 90089-1421
{scacchi}@gilligan.usc.edu
Voice: 213-740-4782, Fax: 213-740-8494

Abstract

I describe our approach and mechanisms to support the engineering of organizational processes throughout their life cycle. I describe our current understanding of what activities are included in the process life cycle. I then go on to describe our approach, computational mechanisms, and experiences in supporting many of these life cycle activities, as well as compare it to other related efforts. Along the way, I present examples drawn from a current study aimed at modeling, analyzing, and integrating an order fulfillment process in a product development organization.

[1]To appear in *Simulating Organizations: Computational Models of Institutions and Groups*, K. Carley, L. Gasser, and M. Prietula (eds.), AAAI Press/MIT Press, 1996.

1 Introduction

Workflow modeling, business process redesign, enterprise integration, and teamwork support are among the current generic goals for advanced information technology (IT) within organizations. Organizations are looking for ways to respond to competitive pressures and new performance levels by redesigning and continuously improving their production and operational processes. Organizations are also looking into IT as a strategy for establishing, sustaining and expanding presence in electronic markets for their goods and services. Such endeavors must therefore address complex organizational processes that entail tens, hundreds, or even thousands of organizational participants, as well as support the integration of a heterogeneous collections of both legacy and emerging ITs. Thus, we are faced with the problem of how to realize these goals in a coherent, scalable, and evolutionary manner.

In this chapter, I describe the approach and supporting mechanisms we have been investigating at the USC ATRIUM Laboratory in an effort to solve this problem and realize these goals. As such, I describe our approach to modeling, enacting, and integrating complex organizational processes using an advanced knowledge-based computing infrastructure, as well as some of the associated technologies we haved developed and deployed in large-scale business and government organizations.

2 The Process Engineering Life Cycle

In simplest terms, we see that support for organizational processes entails more than the modeling and creation of process descriptions or representations. Our view is that the goal should be to support the engineering of organizational processes across the *process life cycle*. Much like the way that the development of complex information systems entails more than programming, so does the development of complex organizational processes entail more than creating documents which describe them. As such, our work at USC has led to the initial formulation of an organizational process life cycle that is founded on the incremental development, iterative refinement, and ongoing evolution of organizational process descriptions. In this way, the organizational process life cycle spiral includes activities that address process:

- *meta-modeling*: constructing and refining a process concept vocabulary and logic (a resource ontology) for representing families of processes and process instances in terms of object classes, attributes, relations, constraints, control flow, rules, and computational methods.

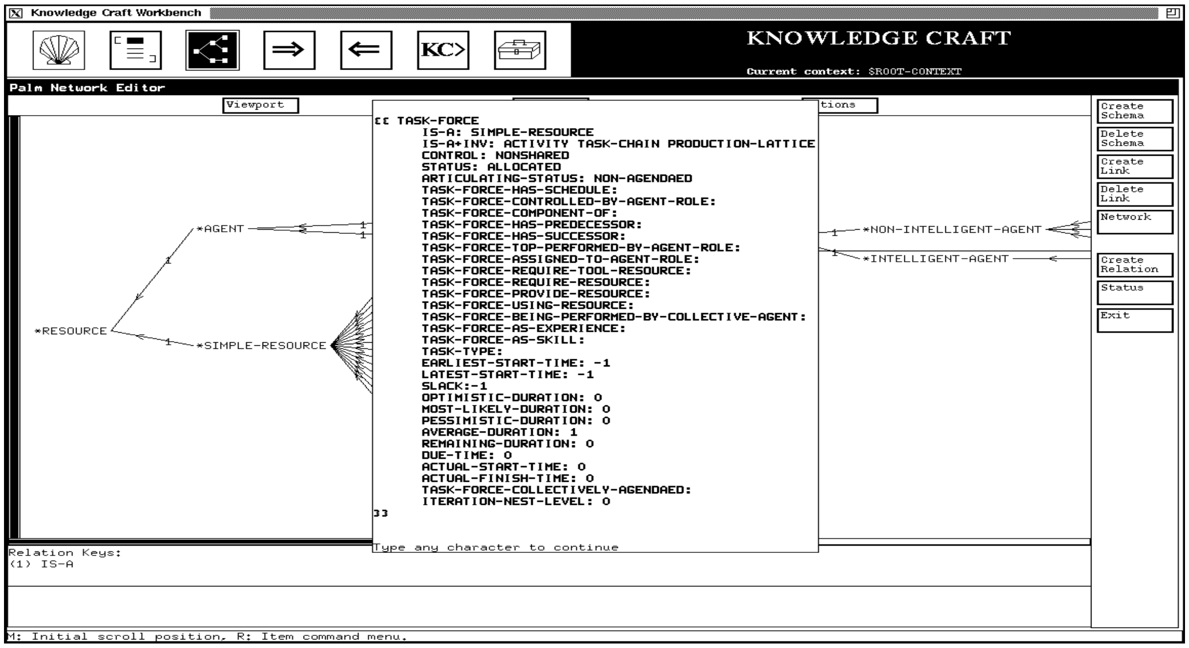
- *modeling*: eliciting and capturing of informal process descriptions, and their conversion into formal process models or process model instances.
- *analysis*: evaluating static and dynamic properties of a process model, including its consistency, completeness, internal correctness, traceability, as well as other semantic checks. Also addresses the feasibility assessment and optimization of alternative process models.
- *simulation*: symbolically enacting process models in order to determine the path and flow of intermediate state transitions in ways that can be made persistent, replayed, queried, dynamically analyzed, and reconfigured into multiple alternative scenarios.
- *visualization*: providing users with graphic views of process models and instances that can be viewed, navigationally traversed, interactively editted, and animated to convey process statics and dynamics.
- *prototyping, walkthrough, and training support*: incrementally enacting partially specified process model instances in order to evaluate process presentation scenarios through the involvement of end users, prior to performing tool and data integration.
- *administration*: assigning and scheduling specified users, tools, and development data objects to modeled user roles, product milestones, and development schedule.
- *integration*: encapsulating or wrapping selected information systems, repositories, and data objects that are to be invoked or manipulated when enacting a process instance. This provides a computational workspace that binds user, organizational role, task, tools, input and output resources into "semantic units of work".
- *environment generation*: automatically transforming a process model or instance into a process-based computing environment that selectively presents prototyped or integrated information systems to end-users for process enactment.
- *instantiation and enactment*: performing the modeled process using the environment by a process instance interpreter that guides or enforces specified users or user roles to enact the process as planned.

- *monitoring, recording, and auditing*: collecting and measuring process enactment data needed to improve subsequent process enactment iterations, as well as documenting what process steps actually occurred in what order.
- *history capture and replay*: graphically simulating the re-enactment of a process, in order to more readily observe process state transitions or to intuitively detect possible process enactment anomalies.
- *articulation*: diagnosing, repairing, and rescheduling actual or simulated process enactments that have unexpectedly broken down due to some unmet process resource requirement, contention, availability, or other resource failure.
- *evolution*: incrementally and iteratively enhancing, restructuring, tuning, migrating, or reengineering process models and process life cycle activities to more effectively meet emerging user requirements, and to capitalize on opportunitistic benefits associated with new tools and techniques.

While such a list might suggest that engineering a business process through its life cycle proceeds in a linear or *waterfall* manner, this is merely a consequence of its narrative presentation. In practical situations where these activities and associated process mechanisms have been initially tried out (e.g, at AT&T Bell Laboratories [Vot93], Northrop-Grumman Corporation, Naval Air Warfare Center (China Lake, CA) and elsewhere [SM93]), it quickly becomes clear that business process engineering is a dynamic team-based endeavor that can only lead to mature processes through rapid process prototyping, incremental development, iterative refinement, and the reengineering of ad hoc process task instances and models. To no surprise, many of our efforts addressing these life cycle activities and supporting prototype mechanisms have been described in greater detail elsewhere [MS90, MS91, NS91, MS92, MS93, MS94, SM93]. As such, I now turn to briefly describe our approach to some of these activities.

3 Modeling, Analysis and Simulation

We have developed a knowledge-based computing environment for modeling, anaylzing, and simulating complex organizational processes [MS90]. We call this environment the Articulator. It first became operational in 1988, and we have continued to use and evolve it since.



```
{{ TASK-FORCE
      IS-A: SIMPLE-RESOURCE
      IS-A+INV: ACTIVITY TASK-CHAIN PRODUCTION-LATTICE
      CONTROL: NONSHARED
      STATUS: ALLOCATED
      ARTICULATING-STATUS: NON-AGENDAED
      TASK-FORCE-HAS-SCHEDULE:
      TASK-FORCE-CONTROLLED-BY-AGENT-ROLE:
      TASK-FORCE-COMPONENT-OF:
      TASK-FORCE-HAS-PREDECESSOR:
      TASK-FORCE-HAS-SUCCESSOR:
      TASK-FORCE-TOP-PERFORMED-BY-AGENT-ROLE:
      TASK-FORCE-ASSIGNED-TO-AGENT-ROLE:
      TASK-FORCE-REQUIRE-TOOL-RESOURCE:
      TASK-FORCE-REQUIRE-RESOURCE:
      TASK-FORCE-PROVIDE-RESOURCE:
      TASK-FORCE-USING-RESOURCE:
      TASK-FORCE-BEING-PERFORMED-BY-COLLECTIVE-AGENT:
      TASK-FORCE-AS-EXPERIENCE:
      TASK-FORCE-AS-SKILL:
      TASK-TYPE:
      EARLIEST-START-TIME: -1
      LATEST-START-TIME: -1
      SLACK:-1
      OPTIMISTIC-DURATION: 0
      MOST-LIKELY-DURATION: 0
      PESSIMISTIC-DURATION: 0
      AVERAGE-DURATION: 1
      REMAINING-DURATION: 0
      DUE-TIME: 0
      ACTUAL-START-TIME: 0
      ACTUAL-FINISH-TIME: 0
      TASK-FORCE-COLLECTIVELY-AGENDAED:
      ITERATION-NEST-LEVEL: 0
}}

Type any character to continue
```

Figure 1: A Generic Business Process Model Class Hierarchy and Schema Example

The Articulator utilizes a rule-based object-oriented knowledge representation scheme for modeling interrelated classes of organizational resources. In this sense, the Articulator's knowledge representation ontology represents a resource-based theory of organizational processes (cf. [Gra91]). In this regard, its purpose and use is similar in spirit to that used in the TOVE system, as described in the chapter by Fox, Barbuceanu, and Gruninger. The Articulator's object classes characterize the attributes, relations, and computational methods associated with a taxonomy of organizational resources. Thus, using the Articulator, we can construct or prototype knowledge-based models of organizational processes. For example, Figure 1 displays a hierarchy of object classes that characterize common business processes that we built using the Articulator. In turn, associated with each of these classes is a schema of attributes, relations and rule-based computational methods that describe the named processes. Figure 2 then follows with a more detailed view of a model of a generic order-fulfillment process.

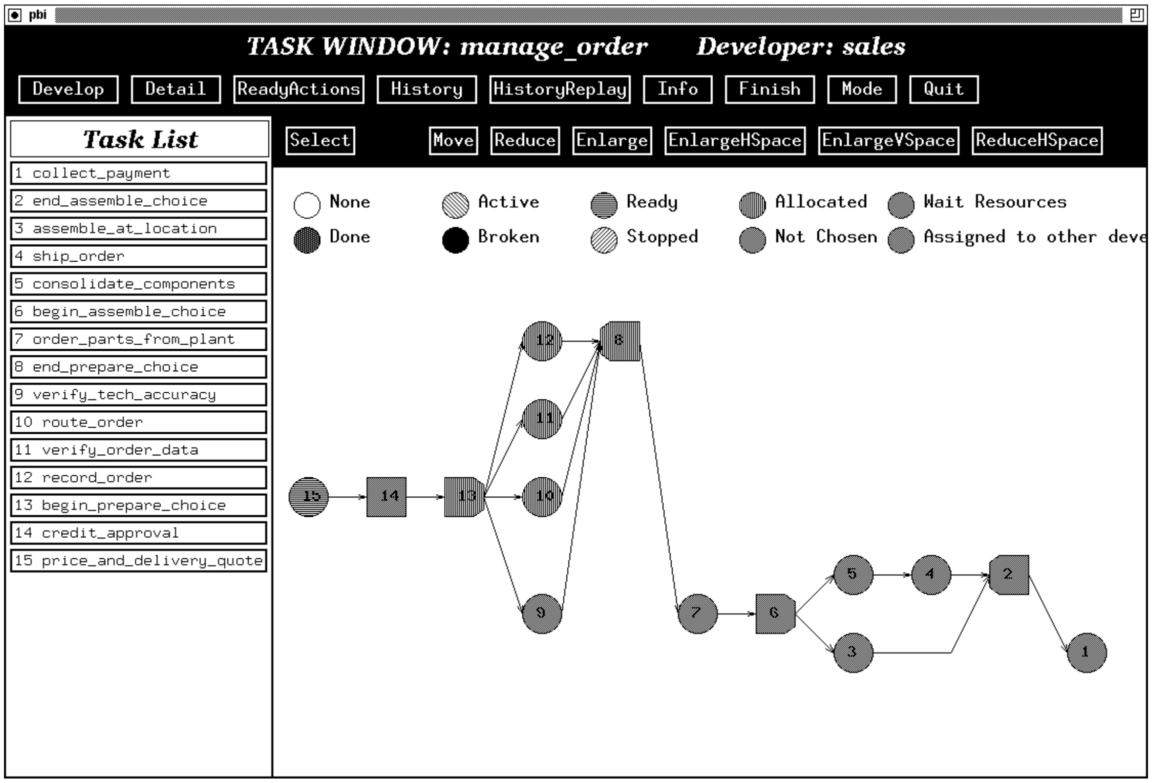


Figure 2: Top Level of an Order Fullfilment Process Model

3.1 Modeling

The resource taxonomy we have constructed, explained in detail elsewhere [GS89, MS90, MS94], serves as a *process meta-model* which provides an ontological framework and vocabulary for constructing *organizational process models* (OPMs). In simplest terms, the process meta-model states that organizational processes can be modeled in terms of (subclasses of) agents that perform tasks using tools and systems which consume or produce resources. Further, agents, ITs, and tasks are resources, which means they can also be consumed or produced by other agents and tasks. For example, a project manager may produce staff through staffing and allocation tasks that consume departmental budgets, while these staff may then assigned to other routine or creative production tasks using the provided resources (e.g., computer workstations, spreadsheet and desktop publishing packages, schedules, and salary) to contruct the desired products or services (e.g., reports and documents). *OPM Instances* can then be created by binding values of corresponding real-world entities to the classes of corresponding entities employed in the OPM. For instance, Mary may be the project manager who is responsible for getting a set of documents produced for an external client, and she is authorized to assign 2-3 individuals in her department to use their desktop PCs that run Windows95, Lotus 1-2-3, and Wordperfect software in order to get the reports produced by the end of the week.

The agents, tasks, product resources, tools, and systems are all hierarchically decomposed into subclasses that inherit the characteristics of their (multiple) parent classes for economy in representation. Further, these resource classes and subclasses are interrelated in order to express relationships such as precedence among tasks (which may be sequential, iterative, conditional, optional, or concurrent), task/product pre- and post-conditions, authority relationships among agent in different roles, product compositions, IT tool/system aggregations, and others [MS90, MS94]. Thus, in using these classes of process modeling entities, we are naturally led to model organizational processes as a web of multiple interacting tasks that are collectively performed by a team of developers using an ensemble of tools that consume resources and produce composed products/artifacts [KS82]. In addition, it allows us to treat these models as *a reusable information resource*, which can be archived, shared, or transferred to other organizations [LA94].

In addition, the meta-model enables us to model other complex phenomena associated with organizational processes, such as agents' resource sovereignties (i.e., the set of resources under the control of an agent), authority asymmetries (e.g., political relationships among agents), multiple belief systems, negotiation strategies, technology transfer strategies, etc. Accordingly, these relationships are defined in the meta-model, used and then instantiated in the OPMs. Then, we use the Articulator to query and simulate modeled processes as describe below.

3.2 Anaylsis

As the process meta-model provides the semantics for OPMs, we can construct computational functions that systematically analyze the consistency, completeness, traceability and internal correctness of OPMs [CS89, MS90]. These functions represent batched or interactive queries to the knowledge base through its representational schemata. At present, we have defined a few dozen paramaterized query functions that can retrieve information through navigational browsing, direct retrieval, or deductive inference, as well as what-if simulations of partial or complete OPMs [MS90]. Further, most of these analysis functions incorporate routines for generating different types of reports (e.g., raw, filtered, abstracted, or paraphrased into structured narrative) which can be viewed interactively or incorporated into desktop publication documents.

3.3 Simulation

Since process models in our scheme are computational descriptions, we can simulate–or, symbolically execute–them using knowledge-based simulation techniques supported by the Articulator [MS90]. In simple terms, this is equivalent to saying that simulation entails the symbolic performance of process tasks by their assigned agents using the tools, systems, and resources to produce the designated products. Using the previous example, this means that in the simulation, Mary's agent would "execute" her project management tasks according to the task precedence structure specified in the OPM instance, consuming simulated time and effort along the way. Since tasks and other resources can be modeled at arbitrary levels of precision and detail, then the simulation makes progress as long as task pre-conditions or post-conditions are satisfied at each step (e.g., for Mary to be able to assign staff to the report production task, such staff must be available at that moment, else the simulated process stops, reports the problem, then waits for new input or command from the simulation user).

Simulations also allow us to dynamically model different samples of parameter values. This in turn enables the simulated processes to function like transportation networks whose volumetric flow, traffic density, and congestion bottlenecks can be assessed according to alternative (hueristic or statistical) arrival rates and service intervals. When use this way, which follows classic discrete-event simulation techniques, users find it is often easy to observe or discover process bottlenecks and optimization alternatives. Further, since commercially available discrete-event simulation now provide animated visual displays, then users can watch process simulations under different scenarios as brief animated movies which can be modified, replayed, and viewed. Although we cannot conveniently show such animations in printed form, the following two snapshots captured from such an animated simulation may suggest what can be observed. In Figure 3, we have modeled an eight person/agent activity for performing an "accounts payable" process[2], which depicts the structure of the workflow, which agents currently perform what tasks, and pending workload quantities (e.g.,

[2]When an organization authorizes a purchase order to be fulfilled, payment for that purchase is provided by the accounts payable process.

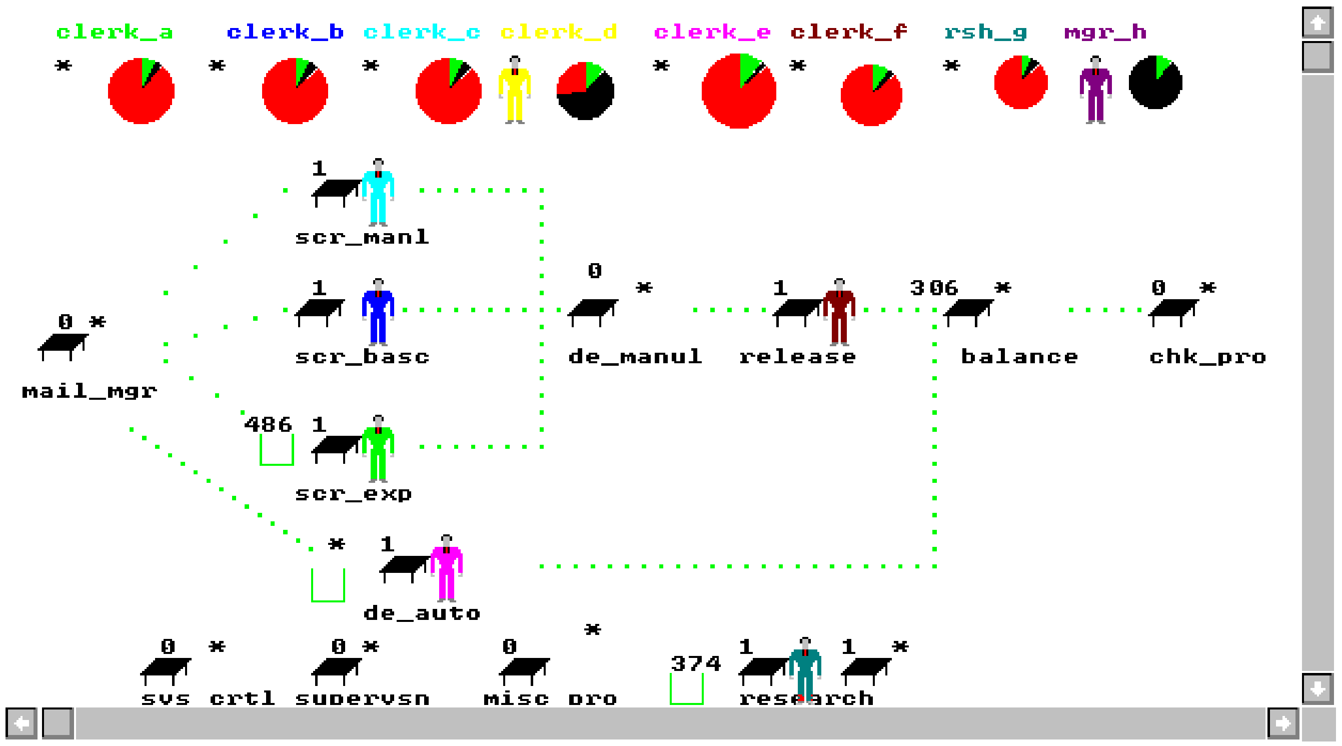
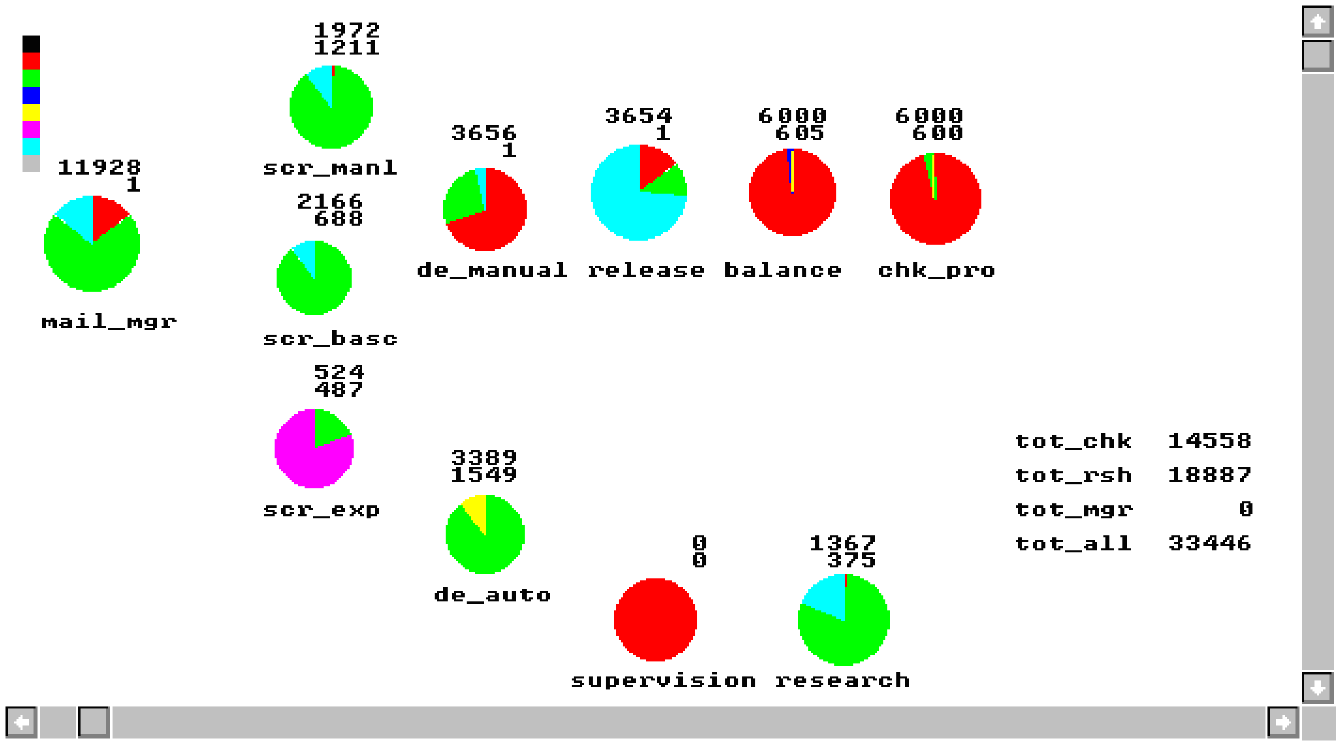


Figure 3: Visual display from an animated multi-agent simulation

the backlog of invoices, bills, and checks). Following this, Figure 4 displays a snapshot of an accompanying pie chart depicting current workload, division of labor, and activity-based cost figures (lower right) for a simulated workflow volume.

We have used the Articulator environment to model, analyze, and simulate a variety of organizational processes. In this regard, we have constructed OPMs and instances for organizations within businesses and government agencies, focused on activities involving team-based IT product design and review processes, as well as department and division-wide IT production and support processes that include tens to hundreds of participants. Such OPMs typically include dozens of classes of agents, tasks, resources, and products, but a small number of IT tools and systems, while the OPM instantiation may include 1-10+ instances of each class. Our experience to date suggests that modeling existing processes can take from 1-3 person-days to 2-3 person-months of effort, analysis routines can run in real-time or acceptable near-real-time, while simulations can take seconds to hours (even days!) depending on the complexity of the OPM, its instance space, and the amount of non-deterministic process activities being modeled. Note however that simulation perfor-



tot_chk	14558
tot_rsh	18887
tot_mgr	0
tot_all	33446

Figure 4: Visual display of dynamic pie-chart depicting current workload, division of labor, and aggregate costs

mance is limited to available processing power and processor memory, thus suggesting better performance can be achieved with (clusters of) high performance computing platforms.

4 Visualization, Prototyping, and Enactment

As we improve our ability to construct and redesign plausible models of different organizational processes, we have found that it is increasingly important to be able to quickly and conveniently understand the structure and dynamics of complex OPM instances. As such, we have developed a graphic user interface (GUI) for visualizing and animating OPM instances. This *process-based interface* (PBI) is coupled to a another computational facility to which OPM instances developed with the Articulator can be automatically translated into executable process programs that are then downloaded into a software program that serves as a *process driver*. In turn, the process driver and GUI enable OPM developers to prototype or enact *process-driven IT environments*. These capabilities can be used to reflect, guide, try-out, and support how users work with process-driven ITs. These capabilities are described next.

4.1 Visualization

PBI provides graphic visualizations of task precedence structure on a role-specific basis for each user (i.e., agent instance) [MS92]. For example, Figure 2 shows a visual rendering of an order-fulfillment process that reveals precedence, iteration, and concurrency relationships among tasks. Since process tasks can be modeled and hierarchically decomposed into subtasks of arbitrary depths, then PBI provides users with a subtask window and an associated (cached) workspace. Figure 5 shows an example of this presentation for a second-level decomposition of the Order Fulfillment Process Model, followed by Figure 6 which displays the third-level decomposition view. Since a subtask precedence structure appears as a directed graph, we associate a development *status* value (i.e., none, allocated, ready, active, broken, blocked, stopped, finished) with each subtask step. For ease of understanding, these status values are represented in the PBI as colors (not shown here), so that the current *state* of a process task can be observed as a color pattern in the direct graph. Further, as PBI also incorporates a facility for recording and replaying all changes in process task state, evolving process state histories can be maintained and visualized as an animation of changing task step status colors. Subsequently, we have found that project managers in industrial organizations can quickly browse such a PBI display to ascertain the current status of an arbitrarily complex production process to varying degress of detail. The interested reader should consult [MS92] to see a number of examples.

4.2 Prototyping

The process driver that backs PBI can also accept an OPM as its input. Since OPMs need not include instance details, then it is possible to use these OPMs to create prototype mock-ups of process-driven environments. These prototypes show the user the look-and-feel of what the emerging process-driven environment would look like. That is, the OPM serves to provide role-specific views of process task precedence structure, which in turn guides users in their use of IT tools, systems, and data resources. Thus, since the Articulator accomodates partially decomposed OPMs, then these OPMs can also be downloaded into the process driver

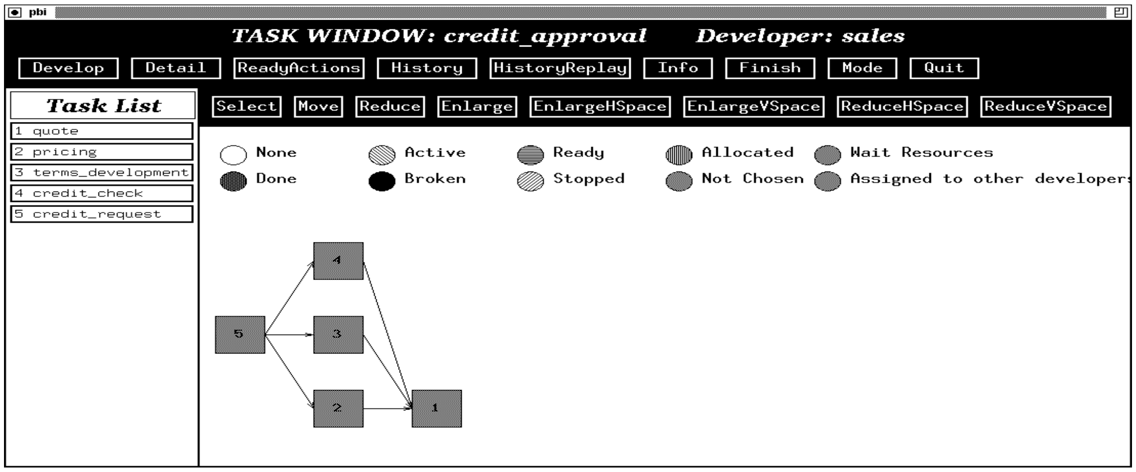


Figure 5: A Second Level Decomposition of the Order Fulfillment Process Model

to visually display and interactively walkthrough role-specific usage scenarios. We find this extremely useful in supporting an OPM construction effort that is iterative, incremental, and improvement-oriented in an evolutionary sense. Further, this prototyping capability can also be used to support training situations, which is especially important when introducing new users to the concepts and mechanisms that support process-driven IT environments.

4.3 Enactment and Integration

The process driver and PBI provides IT tools, systems and associated data resources (e.g., objects, files, databases, spreadsheets) which are served to users at the bottom level subtask actions so that they can perform their work. We refer to this capability as *process enactment*, meaning that users can perform or enact the modeled process tasks, subtasks, or actions assigned to them, and the IT tools, systems, and data resources are delivered to them at their displays and fingertips when needed. Figure 7 shows an example of a process enactment view, which provides the IT applications, tools, data objects and workspace appropriate for

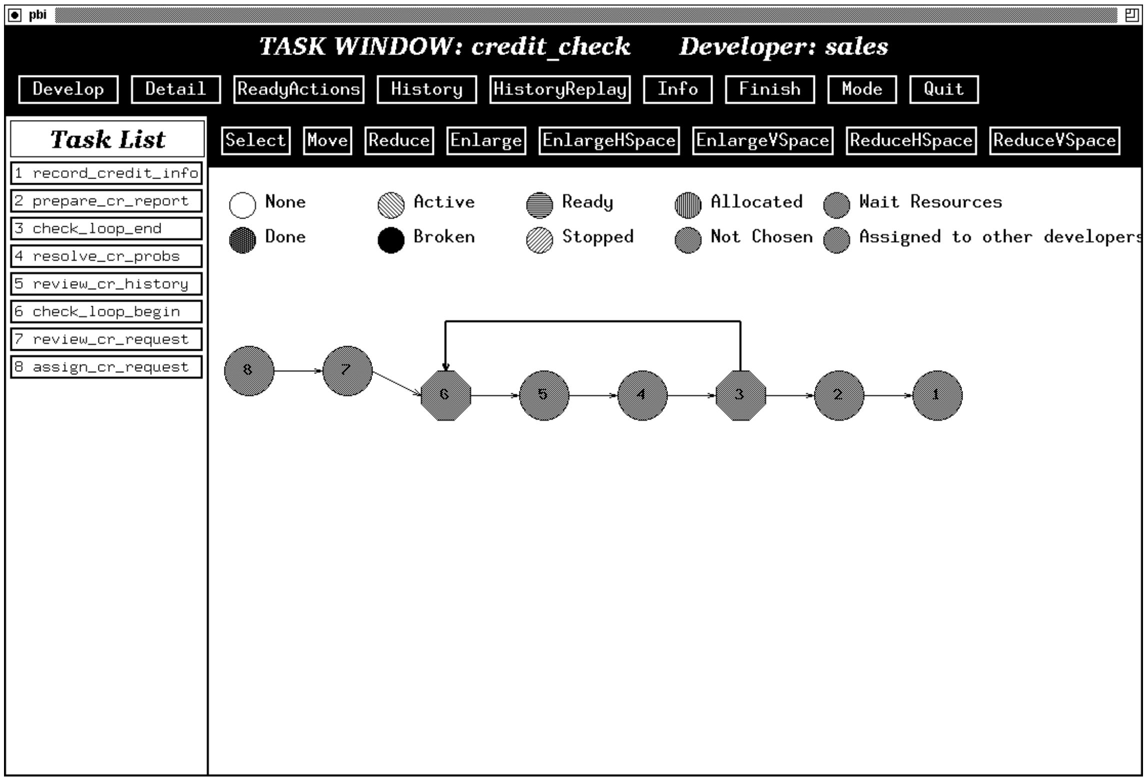
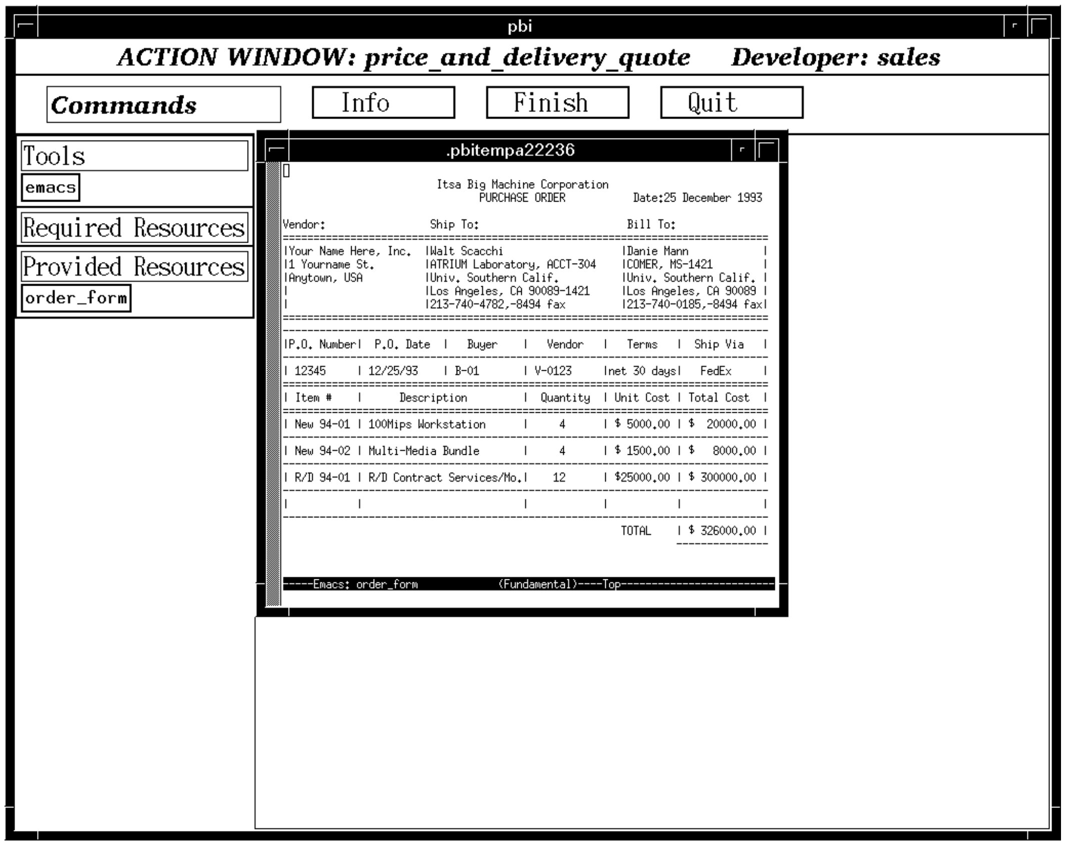


Figure 6: A Third Level Decomposition of the Order Fulfillment Process Model

the user assigned to this order fulfillment process action.

Process enactment is a computational activity. It interprets an OPM or OPM instance as an input. Thus, the OPM or instance output from the Articulator represents a process enactment specification that is coded in something similar to an object-oriented operating system scripting language, or what others have called a *process programming language* [Ost87]. In this sense, our process programs are automatically derived from the process model specification by way of a special-purpose application generator [MS92, KS93]. Accordingly, the process enactment specification can incorporate any operating system command, system invocation script, virtual mouse selections, or canned user input, as well as access protocols to heterogeneous information repositories [NS91]. This means it is possible for users to perform complex information processing tasks through a process-based interface that integrates access to local/networked data resources and IT tools/systems through a common GUI presentation [MS92]. As such, we are now working to prototype and demonstrate a number of process-driven environments in different business and government application domains that incorporate commercial off-the-shelf systems, internally developed systems,



```
                         Itsa Big Machine Corporation
                               PURCHASE ORDER          Date:25 December 1993

Vendor:                 Ship To:                         Bill To:
==========================================================================
|Your Name Here, Inc.  |Walt Scacchi                    |Danie Mann      |
|1 Yourname St.        |ATRIUM Laboratory, ACCT-304      |COMER, MS-1421  |
|Anytown, USA          |Univ. Southern Calif.            |Univ. Southern Calif. |
|                      |Los Angeles, CA 90089-1421       |Los Angeles, CA 90089 |
|                      |213-740-4782,-8494 fax           |213-740-0185,-8494 fax|
==========================================================================
--------------------------------------------------------------------------
|P.O. Number|  P.O. Date  |    Buyer    |    Vendor   |   Terms   |  Ship Via   |
--------------------------------------------------------------------------
| 12345     | 12/25/93    | B-01        | V-0123      |net 30 days|   FedEx     |
==========================================================================
| Item #    |       Description         |  Quantity   | Unit Cost | Total Cost  |
==========================================================================
| New 94-01 | 100Mips Workstation       |      4      | $ 5000.00 | $  20000.00 |
--------------------------------------------------------------------------
| New 94-02 | Multi-Media Bundle        |      4      | $ 1500.00 | $   8000.00 |
--------------------------------------------------------------------------
| R/D 94-01 | R/D Contract Services/Mo.|     12      | $25000.00 | $ 300000.00 |
--------------------------------------------------------------------------
|           |                           |             |           |             |
--------------------------------------------------------------------------
                                                TOTAL     | $ 326000.00 |
                                                           -------------

-----Emacs: order_form            (Fundamental)----Top-----------------------
```

Figure 7: A Lowest Level Action Workspace within the Order Fulfillment Process Model

and prototype research software technologies that can operate over local-area and wide-area networks [NS91].

5 Other Advanced SPE Technologies

In addition to the the computational mechanisms described so far, our approach utilizes mechanisms not described here. These include mechanisms for

- process scheduling and administration [Mi92],
- diagnosing, replanning, and rescheduling processes that unexpectedly breakdown or fail [MS91, Mi92, MS93],
- software re-engineering processes and environment [CS91], and
- knowledge-based process model repository [MLS92].

Thus, we believe our approach can allow us to construct and demonstrate a computational framework for modeling, enacting, and integrating team-oriented process-driven work environments for redesigned business organizations. As such, we are now working with our research sponsors to prototype and demonstrate a small number of process-driven environments in different industrial application domains that incorporate commercial off-the-shelf systems, internally developed systems, and prototype research mechanisms, all operating on Unix and PC workstations over local-area and wide-area networks.

6 Comparison with Other Ongoing Research

Much of the research in engineering process descriptions which influence our work at the USC Atrium Laboratory focuses on representations of the software engineering processes, and architectures for process-centered software engineering environments. However, the Virtual Design Team effort at Stanford described in the chapter by Levitt is also investigating similar issues. Likewise, the ACTION system at USC described by Gasser, Hulthage, and Majchrzak can be viewed as a complementary effort that could serve as a organizational design “front-end” for configuring the architecture of organizational processes that can be modeled, simulated, and enacted through the Articulator environment.

Elsewhere, software process modeling, analysis, and simulation has been a research topic in a number of efforts [CKO92]. At the Software Engineering Institute at Carnegie-Mellon University, Kellner [Kel91] has employed an explicitly enumerated state-space approach to process modeling, using the commercially available Statemate system. His approach to modeling and simulation has been successfully demonstrated on moderate sized project management processes. The PRISM project in Canada [MG90] is focussed on developing a process modeling and evolution methodology that is to be supported by future process-centered environments. However, at this time, neither of these two efforts addresses the support of process environment generation, enactment, replay or articulation.

A number of enactable representations of software engineering processes have been prototyped in recent years. For example, APPL/A [SSHO90] is a process programming language developed in the Arcadia project [TBCO89]. It expands the programming language Ada to include process constructors, relations, and other constructs to describe procedural aspects of a software process. Since APPL/A is targeted at process integration and enactment in an Ada-based environment, it is not at present well-suited for upstream process life cycle activities, such as incremental process modeling or simulation. In contrast, the PSS project in England [BPe91] has developed a process modeling and enactment language based on an object-oriented knowledge representation notation. Grapple [HL88], on the other hand, relies on a set of goal operators and a planning mechanism to represent software processes. These are used to demonstrate goal-directed reasoning about software processes during modeling and enactment. AP5 [BN93], developed at USC-ISI, and Marvel [KF87] developed at Columbia, use pattern-directed inference rules to model and trigger software process actions during process enactment. While Marvel has been extended to support the creation of process models [KBS90], its strength lies primarily in its ability to support rule-based process integration and enactment.

In the commercial arena, one recent process-centered environment is the Process Software Life Cycle Support Environment, ProSLCSE, from ISSI. We have found that the ProSLCSE representation of software processes lacks machine-readable data about the purpose and types of tasks and other activities. Next, Process WEAVER [Fer93], a commercial product from Cap Gemini Innovation based in France, supports process modeling using a notational scheme derived from Petri-nets. In contrast, the computer-aided concurrent engineering product, CACE/PM from Perceptronics [Mad90], also employs a notational scheme derived from Petri-nets. However, CACE/PM offers a more substantial representational capability (e.g., rules, frames, attributes, and timing information), as well as supporting process model analysis and simulation. Up to this time, CACE/PM has been targeted to modeling development processes in the area of Electronic Design Automation and Manufacturing. However, it has not been applied to modeling and analyzing other technical or business processes.

In sum, no other process engineering environment today supports the full process life cycle. However, we have been able to demonstrate supporting mechanisms for the process life cycle activities described in Section 2. Similarly, it should be noted that though our focus is targeted at engineering organizational processes, our approach can be applied to both complex technical domains (e.g, large-scale software engineering, electronic design automation, agile manufacturing) and to conventional business processes (new product development, corporate finance, business planning, etc.), albeit in a radically innovative way [Dav93].

7 Conclusion

This chapter provides a brief introduction to our approach and computational mechanisms to modeling, simulating, and integrating organizational processes that involve IT tools, systems, and data resources. These include a knowledge-based environment for re-engineering complex organization processes, and other facilities for realizing and executing these processes. We are using our results to help redesign existing organizational processes that employ large teams, and provide a coherent, scalable IT infrastructure for enacting and integrating IT-based organizational processes.

Finally, in addition to the the computational mechanisms described here, our approach utilizes mechanisms not described here. We are now investigating demonstrating mechanisms for (a) acquiring and re-engineering action invocation histories into reusable process model fragments, and (b) coordinating remote user processes via email-based process deployment and retrieval. Thus, we believe our approach can allow us to construct and demonstrate an advanced IT for modeling, enacting, and integrating team-oriented process-driven IT-based work environments for redesigned business and government organizations.

In closing, I recommend readers interested in an up-to-date view of ongoing research described in this chapter to examine on the World Wide Web, an interactive presentation found at http://www.usc.edu/dept/ATRIUM/Process_Life_Cycle.html for further details and examples.

8 Acknowledgements

This work is supported as part of the ATRIUM Laboratory at USC. Recent sponsors include Andersen Consulting's Center for Strategic Technology and Research (CSTaR), AT&T Bell Laboratories, EDS, Hewlett-Packard, IBM Canada Ltd., McKesson Water Products Company, Northrop-Grumman Corporation, Office of Naval Research (contract number N00014-94-1-0889), and the Center for Operations Management, Education and Research (COMER) in the USC School of Business Administration. However, no endorsements are implied. In addition, a number of people contributed to the ideas or system development work on which this report is based including Song C. Choi, Pankaj Garg, Anthony Karrer, Ming-Jun Lee, Jinhui Luo, Pei-Wei Mi, and John Noll. All of these contributions are greatly appreciated.

References

[BN93] R. Balzer and K. Narayanaswamy. Mechanisms for Generic Process Support. In *Proc. First ACM SIGSOFT Symp. Foundations Software Engineering*, pages 9–20. ACM, Software Engineering Notes, Vol. 18(5), December 1993.

[BPe91] R.F. Bruynooghe, J.M. Parker, and etc. PSS: A System for Process Enactment. In *Proc. of the 1st International Conference on the Software Process*, pages 128–141, Redondo Beach, CA, Oct 1991.

[CKO92] B. Curtis, M. Kellner, and J. Over. Process Modeling. *Communications ACM*, 35(9):75–90, 1992.

[CS89] S.C. Choi and W. Scacchi. Assuring the Correctness of Configured Software Descriptions. *ACM Software Engineering Notes*, 17(7):67–76, 1989.

[CS91] S.C. Choi and W. Scacchi. SOFTMAN: An Environment for Forward and Reverse CASE. *Information and Software Technology*, 33(9), Nov. 1991.

[Dav93] T. Davenport. *Process Innovation: Reengineering Business Processes through Information Technology.* Harvard Business School Press, Cambridge, MA, 1993.

[Fer93] C. Fernstrom. Process WEAVER: Adding Process Support to UNIX. In *Proc. 2nd International Conference on the Software Process*, Berlin, Germany, February 1993.

[Gra91] R. M. Grant. The Resource-Based Theory of Competitive Advantage: Implications for Strategy Formulation. *California Management Review*, 33(3):114–135, 1991.

[GS89] P.K. Garg and W. Scacchi. ISHYS: Designing Intelligent Software Hypertext Systems. *IEEE Expert*, 4(3):52–63, 1989.

[HL88] K.E. Huff and V.R. Lesser. A Plan-Based Intelligent Assistant That Supports the Process of Programming. *ACM SIGSOFT Software Engineering Notes*, 13:97–106, Nov 1988.

[KBS90] G.E. Kaiser, N.S. Barghouti, and M.H. Sokolsky. Preliminary Experience with Process Modeling in the Marvel Software Development Environment Kernel. In *Proc. of the 23rd Annual Hawaii International Conference on System Science*, pages 131–140, Jan 1990.

[Kel91] M. Kellner. Software Process Modeling Support for Management Planning and Control. In *Proc. 1st. Intern. Conf. Soft. Process*, pages 8–28. IEEE Computer Society, October 1991.

[KF87] G.E. Kaiser and P. Feiler. An architecture for intelligent assistance in software development. In *Proc. of the 9th International Conference on Software Engineering*, pages 180–187, Monterey, CA, Apr 1987.

[KS82] R. Kling and W. Scacchi. The Web of Computing: Computer technology as social organization. In M. Yovits, editor, *Advances in Computers, vol. 21*, pages 3–90. Academic Press, New York, 1982.

[KS93] A. Karrer and W. Scacchi. Meta-Environments for Software Production. *International Journal of Software Engineering and Knowledge Engineering*, 3(1):139–162, 1993.

[LA94] F. Leymann and W. Altenhuber. Managing Buiness Processes as an Information Resource. *IBM Systems J.*, 33(2):326–348, 1994.

[Mad90] A. Madni. A Conceptual Framework and Enabling Technologies for Computer-Aided Concurrent Engineering (CACE). In *Second Intern. Conf. Human Aspects of Advanced Manufacturing and Hybrid Automation.* Plenary Address and Invited Paper, August 1990.

[MG90] N. Madhavji and V. Gruhn. PRISM = Methodology + Process-oriented Environment. In *Proc. of the 12th International Conference on Software Engineering*, 1990.

[Mi92] P. Mi. *Modeling and Analyzing the Software Process and Process Breakdowns.* PhD thesis, Computer Science Dept. University of Southern California, 1992. September.

[MLS92] P. Mi, M. Lee, and W. Scacchi. A Knowledge-based Software Process Library for Process-driven Software Development. In *Proc. 7th Knowledge-Based Software Engineering Conference*, McLean, VA, September 1992.

[MS90] P. Mi and W. Scacchi. A Knowledge-based Environment for Modeling and Simulating Software Engineering Processes. *IEEE Trans. on Knowledge and Data Engineering*, 2(3):283–294, Sept 1990. Also appears in Nikkei Artificial Intelligence, Vol.24(1), Jan. 1991, (in Japanese).

[MS91] P. Mi and W. Scacchi. Modeling Articulation Work in Software Engineering Processes. *Proc. of the 1st International Conference on the Software Process*, pages 188–201, Oct 1991.

[MS92] P. Mi and W. Scacchi. Process Integration in CASE Environments. *IEEE Software*, 9(2):45–53, March 1992. Also appears in Computer-Aided Software Engineering, (2nd Edition), E. Chikofski (ed.), IEEE Computer Society (1993).

[MS93] P. Mi and W. Scacchi. Articulation: An Integrated Approach to Diagnosis, Re-planning, and Re-scheduling. In *Proc. 8th. Knowledge-Based Software Engineering Conf.*, pages 77–85, Chicago, IL, 1993.

[MS94] P. Mi and W. Scacchi. A Meta-Model for Formulating Knowledge-Based Models of Software Development. *IOM Department technical report 94-16*, (submitted for publication)(August), 1994.

[NS91] J. Noll and W. Scacchi. Integrating Diverse Information Repositories: A Distributed Hypertext Approach. *Computer*, 24(12):38–45, Dec. 1991.

[Ost87] L. Osterweil. Software Processes are Software Too. In *Proc. of the 9th International Conference on Software Engineering*, pages 2–13, Monterey, CA, Apr 1987.

[SM93] W. Scacchi and P. Mi. Modeling, Integrating, and Enacting Software Engineering Processes. In *Proc. 3rd. Irvine Software Symposium.* Irvine Research Unit in Software, University of California at Irvine, April 1993.

[SSHO90] Jr. S. Sutton, D. Heimbigner, and L.J. Osterweil. Language Constructs for Managing Change in Process-Centered Environments. In *Proc. of the 4th ACM SIGSOFT Symposium on Software Development Environments*, pages 206–217, Irvine, CA, Dec. 3-5 1990.

[TBCO89] R.N. Taylor, F.C. Belz, L.A. Clarke, and L. Osterweil. Foundations for the Arcadia Environment Architecture. *ACM SIGPLAN Notice*, pages 1–13, Feb 1989.

[Vot93] L. Votta. Comparing One Formal to One Informal Process Description. In *position paper circulated at the 8th. Intern. Soft. Process Work.* Dagstuhl, Germany, IEEE Computer Society, February 1993.