Process Life Cycle Engineering: A Knowledge-Based Approach and Environment

Walt Scacchi* and Peiwei Mi
University of Southern California, USA

ABSTRACT **We describe our approach and mechanisms to support the engineering of organizational processes throughout their life cycle, and our current understanding of what activities are included in the process life cycle. We then go on to discuss 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, we present examples drawn from a recent study that uses the approach and the mechanisms of our knowledge-based process engineering environment to support the (re)engineering of corporate financial operations in a mid-size consumer products organization.** © **1997 by John Wiley & Sons, Ltd.**

Intell. Sys. Acc. Fin. Mgmt. **6**: 83–107, 1997.

No. of Figures: 12. No. of Tables: 0. No. of References: 50.

Keywords: business process reengineering; knowledge-based process engineering; process life cycle; knowledge-based process engineering environment

INTRODUCTION

Workflow modeling, business process redesign, enterprise integration, teamwork support, and management of knowledge assets 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 collection 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 paper, we 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, we describe our approach to the engineering and redesign of complex organizational processes using a knowledge-based computing environment, as well as some

*Correspondence to Walt Scacchi, Information and Operations Management Department, University of Southern California, Los Angeles, CA 90089-1421, USA. Email: scacchi@gilligan.usc.edu

CCC 1055-615X/97/020083-25$17.50

of the associated technologies we have developed and deployed in large-scale business and government organizations. In particular, we draw upon examples resulting from the application of our approach and supporting knowledge-based environment to business processes found in corporate financial operations in a mid-size consumer products company (annual revenue more than $250m/year).

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 models 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 the following set of activities:

- *Meta-modeling*: constructing and refining a process concept vocabulary and logic 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, then converting them 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.
- *Redesign*: reorganizing and transforming the structure of relationships within a process to compress completion time, as well as reduce or eliminate the number of steps, handoffs, or participants.
- *Visualization*: providing users with graphic views of process models and instances that can be viewed, navigationally traversed, interactively editted, and animated to convey an intuitive understanding of process statics and dynamics.
- *Prototyping, walkthrough, and performance 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 can 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 system functions to end-users for process enactment.
- *Instantiation and enactment*: performing the modeled process through the environment using 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*: recording the enactment history and 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 or improvement opportunities.
- *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 opportunistic benefits associated with new tools and techniques.
- *Process asset management*: organizing and managing the collection of meta-models, models, and instances of processes, products, tools, documents, and organizational structures/roles for engineering, redesign, and reuse activities.

While such a list of activities might suggst 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 we have employed these activities and associated process mechanisms (e.g. at AT&T Bell Laboratories (Votta, 1993), Northrop-Grumman Corporation, Naval Air Warfare Center (China Lake, CA), McKesson Water Products Company, and elsewhere (Scacchi and Mi, 1993), 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 validation and 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 (Mi and Scacchi, 1990, 1992, 1993, 1996; Noll and Scacchi, 1991; Scacchi and Mi, 1993).

Each of these activities is necessary to address a clear and distinct problem that emerges when using a process engineering environment to support a redesign effort. For example, with an early version of our process engineering environment (Mi and Scacchi, 1990), we sought to understand complex processes via modeling, analysis, and simulation. This enabled us to construct knowledge-based models of multi-agent business processes, which we could then analyze through queries and simulation. However, trying to convey what was modeled, or to explain the simulation results, we found that others not involved in directly using the environment could often not readily grasped how we achieved our computational results. This in turn then gave rise for a need to improve the communicability of results through the development and use of tools to support model visualization and visual simulation animations. Similarly, as the computational results of process simulation studies became more accessible and more easily understood, we encountered requests to be able to let other users engage, try out, or 'fly' process simulations as a way to more effectively provide feedback about what process redesign alternatives made most sense to them. This in turn gave rise to the development of computational mechanisms for process prototyping and process enactment activities (Mi and Scacchi, 1992), and later still for automatically generating process enactable application environments (Garg *et al.*, 1994). Accordingly, each of the process life cycle engineering activities that we have investigated could generally be traced back to feedback from corporate users or others in our audience. Thus, while our approach and experience have led to a complex set of process life cycle engineering activities and supporting environment capabilities, each was found to be useful and necessary to address some particular process engineering need. Nonetheless, our experience as we will describe also suggests that it may still be the unusual circumstance where all process life cycle engineering activities are pursued with comparable effort. But

this may also just reflect the newness of the innovative capabilities we have at hand.

To date, our most substantial results of near-term value to businesses has been in supporting 'upstream' process engineering activities (meta-modeling through redesign), while much of our recent research attention has been directed at activities from redesign through evolution and asset management. As such, we now turn to briefly describe our approach to some of these activities, with particular emphasis directed to the upstream engineering of business processes common to many corporate financial operations. We follow with description of selected downstream process engineering activities, and then compare our approach to other related research efforts.

UPSTREAM PROCESS ENGINEERING: META-MODELING, MODELING, ANALYSIS, SIMULATION AND REDESIGN

We have developed a knowledge-based computing environment for engineering complex organizational processes (Mi and Scacchi, 1990). We call this environment the Articulator. It first became operational in 1988, and we have continued to use and evolve it since. The Articulator utilizes a rule-based object-oriented knowledge representation scheme for modeling interrelated classes of organizational resources (Mi and Scacchi, 1990, 1996). In this sense, the Articulator's knowledge representation ontology represents a resource-based theory of organizational processes, which in turn is in line with one of the principal basis for strategic planning and business management (cf. Grant, 1991, Boar, 1993). 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.

Figure 1 provides a graphic overview of common corporate financial operations that are situated between internal customers and external vendors. Note that many sub-processes, such as those for order fulfillment and accounts payable, are not shown. However, for our ontology to be useful in such a domain, it must provide the concepts and representational constructs that allow all resources indicated or implied. This includes multiple decomposable tasks that potentially involve multiple actors or agents working within or across organizational units on different types of documents (purchase orders, invoices, etc.) with information systems. Furthermore, we would like such an ontology to be domain-independent, and thus be useful with little or no modification is supporting diverse process domains such as large-scale software system development, insurance claims processing, military procurement, and others (Scacchi, 1989).

META-MODELING AND MODELING

The resource taxonomy we have constructed, explained in detail elsewhere (Garg and Scacchi, 1989; Mi and Scacchi, 1990, 1996), serves as a **process meta-model** which provides an ontological framework and vocabulary for constructing **organizational process models** (OPMs). Such an ontology is organized as a semantic network of object class schemata that define the name, attributes, relations and rule-based computational methods associated with each class of resource. For example, Figure 2 displays a schema definition for the TASK FORCE object class, which is a resource sub-class used to define the structure of a business process, as well as to characterize its possible set of attributes and common relations.

In this schema, a task-force (process) among other things has relations that define whether it is scheduled; controlled by some agent; part of some embedding process (a sub-process); preceded or followed by other processes; managed by some organizational authority; assigned to some agent within some organizational collective; and involve the use or manipulation of tools, outputs, experiences and skills. Furthermore, as a schedulable object, then it also has properties that indicate constraints and defaults that can guide scheduling mechanisms. Finally, values that can fill these relations or attributes may be other resource classes, or class instances.

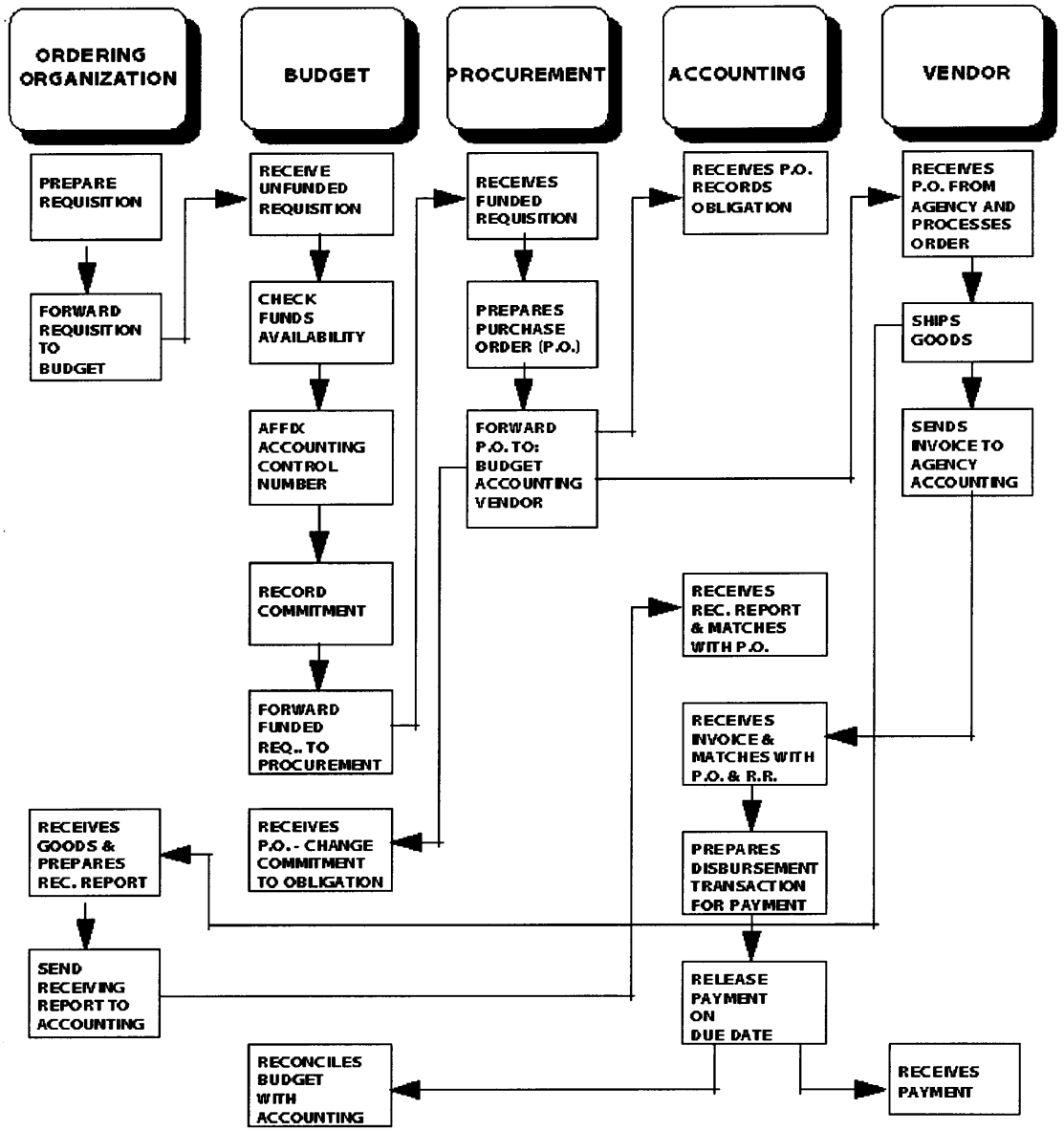

Figure 1 A view of corporate financial operations (from Federal Electronic Commerce Acquisition Management Program Office, 1994).

In simplest terms, the process meta-model states that organizational processes can be modeled in terms of (subclasses of) agents that perform processes using tools which consume or produce resources. Further, agents, tools, and tasks are resources, which means they can also be consumed or produced by other agents and tasks. Using this framework, we can then construct classes of OPMs for different business processes or process domains. For example, an OPM for a generic accounts payable (AP) process may model the following kinds of relations: the AP department manager may produce staff through staffing and allocation tasks

```
{{ 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
}}
```

Figure 2 A class schema for the Task-Force resource used in defining processes.

(sub-processes) that use funds required from departmental budget. In turn, these staff may then be assigned to other creative or routine production tasks (handling invoices) using the provided resources (e.g. computer workstations, corporate financial information systems, spreadsheet and desktop publishing packages, schedules, and salary) to construct the desired products or services (e.g., reports and documents). **OPM Instances** can then be created by binding values that denote real-world entities to the classes of corresponding entities employed in the OPM. For instance, Mary may be the AP department manager who is responsible for producing a weekly report on unresolved invoices (payable accounts) as part of a briefing to the Head of Accounting and others in senior management, possibly including the Chief Financial Officer. Mary's administrative authority enables her to assign 2–3 individuals in her department to use their desktop PCs that run Windows95 to invoke AP functions on the corporate financial system, Lotus 1-2-3 for spreadsheet calculations, and Powerpoint software in order to get the reports and presentation materials produced by the end of the week.

In OPMs and their instances, 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 interrlated 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, information system/tool aggregations, and others (Mi and Scacchi, 1990, 1996). Figure 3 provides a partial view of the functional decomposition of an AP subsystem components within a financial system from the vendor, JDEdwards.

Accordingly, when 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 workers using an ensemble of tools that consume resources and produce composed products/artifacts (Kling and Scacchi, 1982). In addition, it allows us to treate these models as **a reusable information resource** or **knowledge asset**, which can be archived, shared, generalized, or specialized for use in other organizations (Leymann and Altenhuber, 1994, Stein and Zwass, 1995).

Nonetheless, given the richness of the process meta-model and modeling representations, we must then face the challenge of iteratively eliciting, codifying, and revising actual OPMs and instance values from people who are experts in their business process domains. This is the knowledge-acquisition bottleneck we must endure. We have found that it is usually necessary to conduct two to four rounds of interviews with people who are knowledgable about their processes. Furthermore, we choose to elicit knowledge about both existing 'as-is-' processes, as well as possible 'to-be' process alternatives to help us better understand and represent the organizational knowledge at hand. Our experience has been that as-is business processes are ill-defined and not well understood, while most process experts or informants

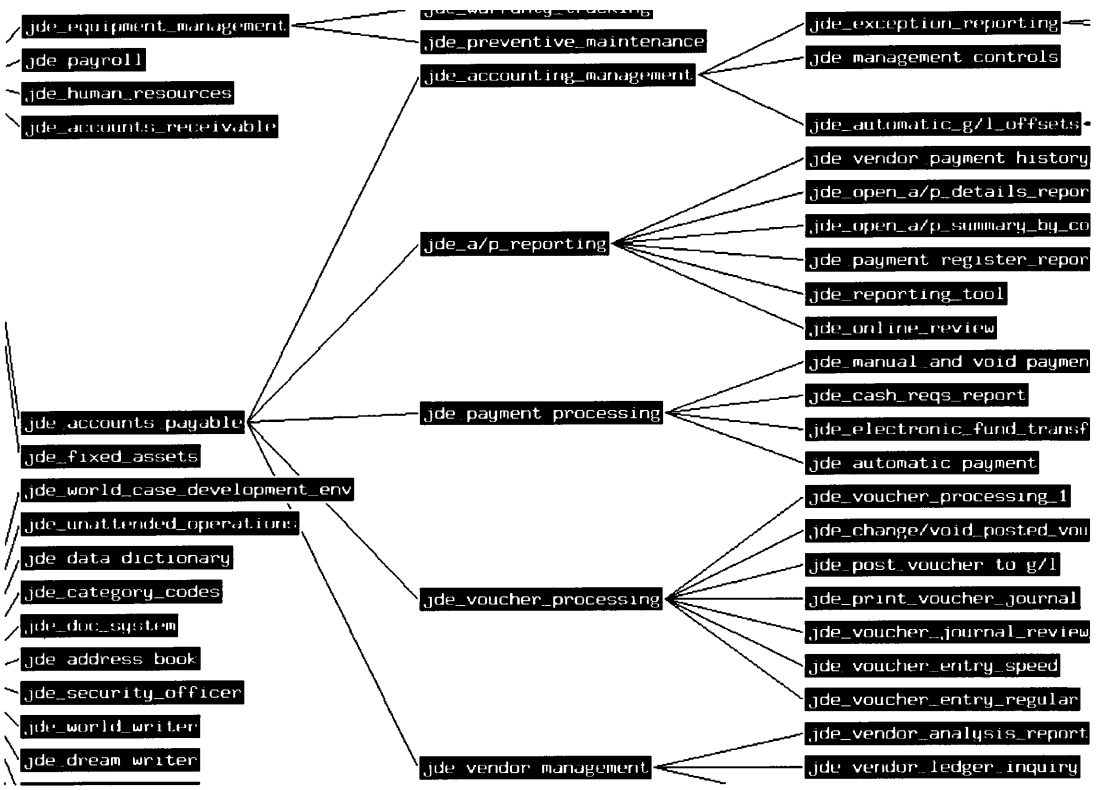

Figure 3 Hierarchical decomposition of AP financial system functional components.

want to focus on to-be alternatives without baselining the current as-is conditions. While this experience may be common to developers of expert systems, the lack of as-is baseline can undercut the effort to systematically analyze and identify process redesign alternatives or transformation sequences, as well as the quantification of potential savings.

ANALYSIS

As the process meta-model provides the semantics for OPMs, we can construct computational functions that systematically analyse the consistency, completeness, traceability and internal correctness of OPMs (Mi and Scacchi, 1990; Choi and Scacchi, 1996). Such functions help in **verifying** logical and semantic properties of OPMs we capture, as well as revealing the presence of gaps or bugs in the emerging OPMs. These functions represent batched or interactive queries through the Articulator to the knowledge base through its representational schemata. We have defined a few dozen parameterized 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 (Mi and Scacchi, 1990). For example, in Figure 4, we show part of the results from a query function applied to an OPM that calculates some descriptive statistics and reports a tally of the number and types of incomplete resource specification that were detected.

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. Similarly, reports can be automatically generated as desktop presentation materials or documents formatted for

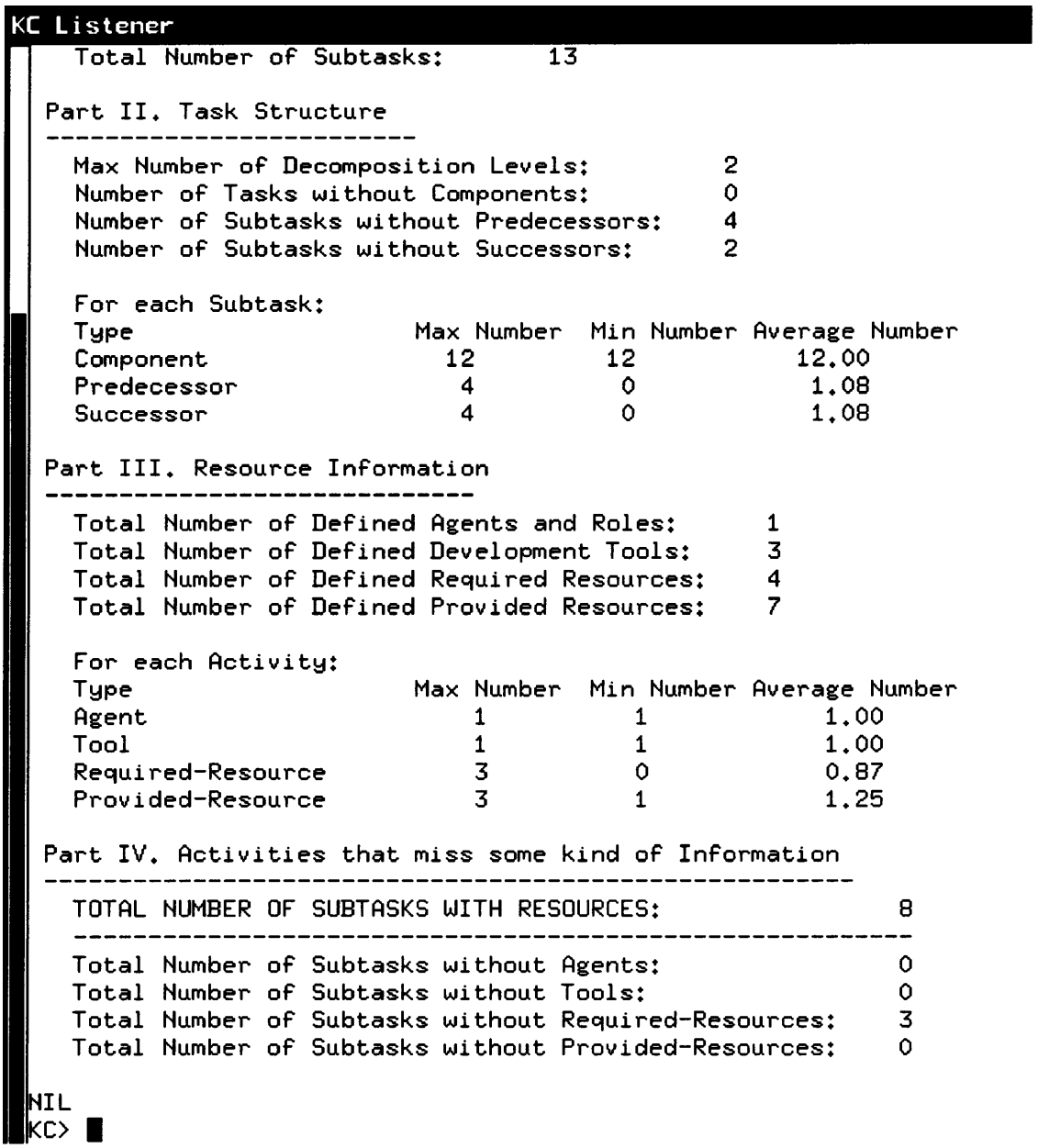

Figure 4 Example output from an OPM process analysis query

publication. The paraphrasing function employs classic natural language generation methods that traverse and unparse a semantic network following the approach originally due to Simmons and others from the 1970s (Simmons and Slocum, 1972). It also now includes routines supporting the generation of materials that instantiate report or presentation templates coded in HTML for dissemination using a corporate intranet, as we will show later.

We also must address **validating** the OPMs that we capture (O'Leary, 1987; O'Leary *et al.*,

1990). Here we rely upon an iterative and incremental method for modeling, verifying, refining, and validating OPM knowledge that is acquired from different people at different times. Typically, we have found that three iterations across these activities is required to achieve an external validation sign-off from the process participants. Further, when possible to-be process alternatives are identified, we must also assess their feasibility given resources available or likely to become available for different business processes. Thus, we rely on our experts to review, informally modify, and sign-off on various descriptions and visualizations of the OPM that are generated by the Articulator and related utilities.

Overall, our experience has been that automated verification and participatory validation of OPMs is a great source of short-term, high-value results and insight which can be provided to a business organization through a process engineering effort.

SIMULATION

Since process models in our scheme are computational descriptions, we can simulate or symbolicaly execute them using knowledge-based simulation techniques supported by the Articulator (Mi and Scacchi, 1990). 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 earlier example, this means that in simulating an instance of the AP OPM, Mary's agent would 'execute' her management tasks according to the task precedence structure specified in the OPM instance, consuming simulated time, budgeted funds, and other resources 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 preconditions 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).

Our use of knowledge-based simulation mechanisms also support features that are uncommon in popular commercial simulation packages. For example, using symbolic execution functions, the Articulator can determine the path and flow of intermediate process state transitions in ways that can be made persistent, queried, dynamically analyzed, and reconfigured into multiple alternative scenarios. Persistent storage of simulated events enables the ability to run simulation forward and backward to any specific event or update to the knowledge base. Queries provide one such mechanism to retrieve or deduce where an event or update of interest occurs. Dynamic analysis can monitor usage or consumption of resources to help identify possible bottlenecks in OPM instances. Then, using any of these functions, it is possible to run a simulation to some point, backup to some previous context, create new OPM instance values (e.g. add more time to a schedule, more staff to a overloaded workflow, or remove unspent money from a budget), branch off a new simulation trajectory that can subsequently be made persistent, and so forth. Furthermore, we can also employ the paraphrasing and report generation functions noted above to produce narrative-like descriptions or summaries of what transpired during a given simulation run. Thus, knowledge-based simulation enables the creation and incremental evolution of a network of event trajectories which may be useful in evaluating or systemically forecasting the yield attributal to to-be process alternatives.

Simulations also allow us to dynamically analyze different samples of parameter values in OPM instances. 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 (heuristic or statistical) arrival rates and service intervals. When used this way, as a classic discrete-event simulation, process experts find it easy to observe or discover process bottlenecks and optimization alternatives. Similarly, when repetitive, high-frequency processes such as AP are being stud-

ied, and when data on events and process step completion times/costs can be empirically measured or captured, then this provides a basis for assessing and validating the **replicability** of the simulated process to actual experience. As this was the situation for us during this study, we found we could achieve simulation results on as-is AP processes that were consistent with observed measurements within 85–98% for the instance value samples investigated.

Since commercially available discrete-event simulation now support animated visual displays, we employ them so that process experts can further validate as-is and to-be process simulations under different scenarios as animated displays ('business process movies'). These animated OPM simulations in turn can be modified, re-executed, and viewed like other simulations. 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 5, we have modeled an eight-person activity for performing an instance of the AP process. The figure depicts the structure of the workflow, which agents currently perform what tasks, and work units-in-progress quantities (e.g. the backlog of invoices cleared, problematic invoices, and checks released). Following this, Figure 6 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-

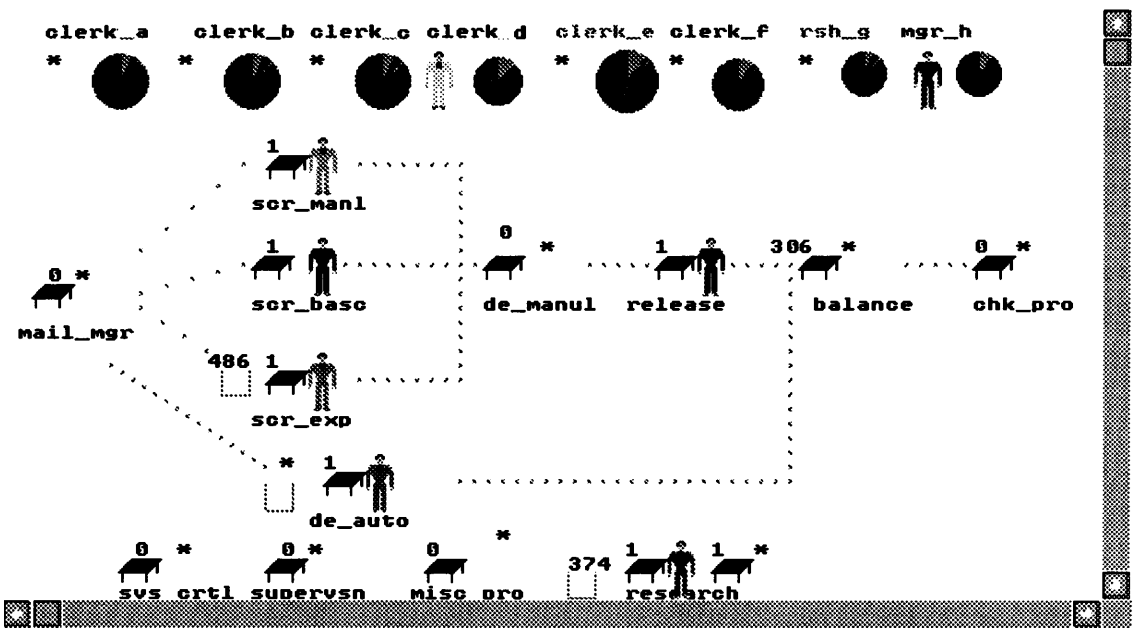

Figure 5 Visual display from an animated multi-agent simulation

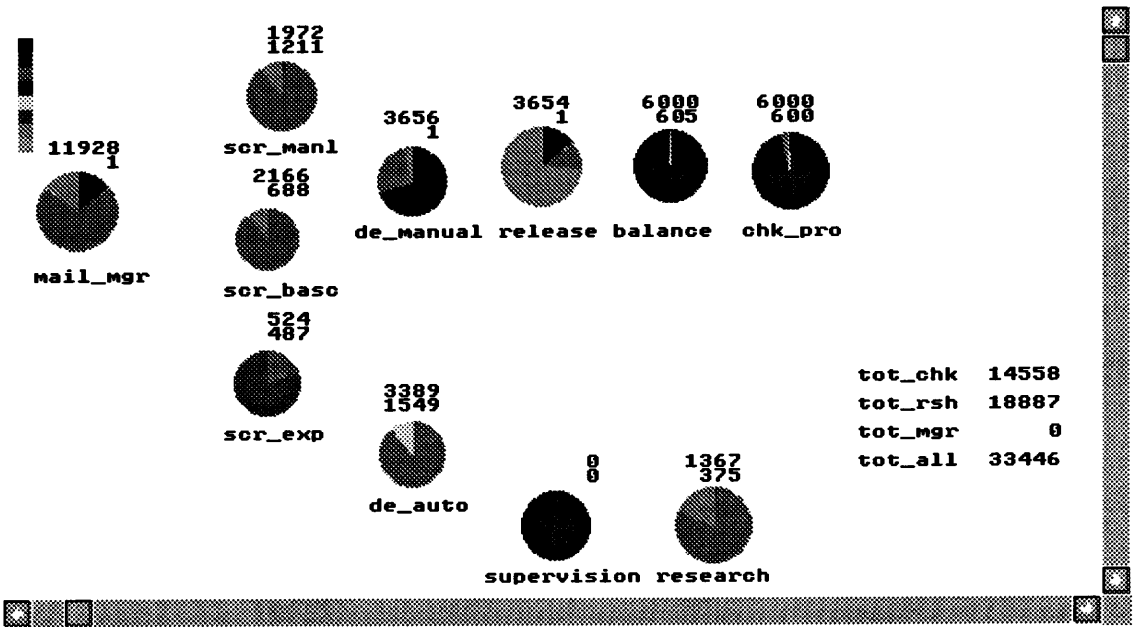

Figure 6 Visual display of dynamic pie chart depicting current workload, division of labor, and aggregate costs.

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

Overall, our experience with these simulation capabilities can be summarized according to the kind of mechanisms employed. We found that knowledge-based simulation was of greatest value in supporting exploratory analysis of OPMs where attention was focused on understanding fine-grained or deep causal relationships that could arise during a symbolic execution. This helps facilitate **qualitative** insights into the operation of OPMs. In contrast, discrete-event simulation was of greatest value in validating and assessing 'shallow' OPM instances using coarse-grain and statistically representative data samples. This helps facilitate **quantitative** insights into the operation of alternative OPMs. Thus, together we find that both knowledge-based and conventional simulation functions are most helpful when used to complement the strengths of one another.

REDESIGN

Process redesign is concerned with structural transformation of workflow or other relational properties associated with a process or sequence of process steps. At this point, most of the knowledge for how to transform a process, or what transformations are available or applicable remains informal. Most of the popular treatments on business process redesign are motivational rather than systematic and analytically reproducible.

Recent progress is beginning to suggest that when process descriptions can be represented as an attributed directed graph or semantic network, then knowledge-based techniques and computational mechanisms may be applied to identify or eliminate possible process redesigns.

These efforts involve the use of rule-based representations to recognize and transforms patterns in the formal representation of a process model (Ku *et al.*, 1996). In this regard, the condition part (the left-hand side) of the rule recognizes a process pattern, while the action part (the right-hand side) invokes a method that replicates some form of a case-based, 'best-practice', flow optimization, or other process redesign heuristic.

Within our research group, we have been exploring process redesign in the following manner. We specify a set of measurements on graph connectivity, interrelations, and node/link complexity that may be used as indices into a taxonomy of process transformations (Nissen, 1994, 1996). Such metrics distill domain-independent, application domain-specific, and setting-specific instance patterns in the formal representation of a modeled process. A taxonomic classification of business process transformations can then be employed when populated with domain-independent transformations (e.g., parallelize a sequence of process steps if mutually independent); application domain-specific transformations (reduce the handling of problematic invoices in AP by pre-filtering invoices received and recycling back those with problems); and setting-specific transformations (if Mary's workload is reduced, she can perform Patrick's tasks as well, thereby freeing up Patrick to perform other tasks).

Formalizing the condition metrics patterns, as well as the action transformations, appears most attractive and most widely reusable for domain-independent process redesigns. Alternatively, formalizing application-specific metrics and transformations should yield a more powerful approach leading to more dramatically streamlined process redesigns. The price paid to acquire such knowledge is great, but common business process such as AP are almost universally found in every business. This suggests the possibility of amortizing the investment in knowledge acquisition for the chosen application domain across many possible reapplications. Finally, setting-specific metrics and transformations are probably most likely not to be codified or automated, since the cost of capturing and codifying such idiosyncratic knowledge is likely to be outweighed by the potentially short duration of its utility. However, such intimate knowledge of the setting (and participants) where process redesign is being considered is likely to be among the most influential variables that determine the success or failure of a business process redesign effort. Thus, our approach is focusing on codifying the most reusable knowledge across common business process application domains, while relying on the active engagement and participation of the people who perform the process, as well as have a stake in its redesigned outcome, to select among process redesign alternatives which can be identified and further engineered.

DOWNSTREAM PROCESS ENGINEERING: VISUALIZATION THROUGH EVOLUTION

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 another computational facility that can automatically translate OPM instances into executable process programs. These process programs are then downloaded into a program interpreter that serves as a **process execution mechanism**. In turn, the process execution mechanism 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.

VISUALIZATION

PBI provides graphic visualizations of task precedence structure on a role-specific basis for each user (i.e. agent instance) (Mi and Scacchi, 1992). Since process tasks can be modeled and

hierarchically decomposed into subtasks or arbitrary depths, then PBI provides users with a subtask window and an associated workspace. Figure 7 shows an example of this presentation for the top-level view of the AP process, which highlights the process's logical workflow, from left to right. 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 process task or step (nodes in the graph). For ease of understanding, each of these status values is represented in the PBI display in a distinct color (not shown here), so that the current **state** of a process task can be observed as a color pattern in the directed 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 department managers in business 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 Mi and Scacchi (1992) and http://www.usc.edu/dept/ATRIUM/Process_Life_Cycle.html to see a number of examples.

PROTOTYPING AND PERFORMANCE SUPPORT

The process execution mechanism that backs PBI can also accept an OPM as its input. Since OPMs need not include instance details until process enactment, 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 how the emerging process-driven environment would appear. That is, the OPM serves to provide role-specific views of process task precedence structure, which in turn guides users in their

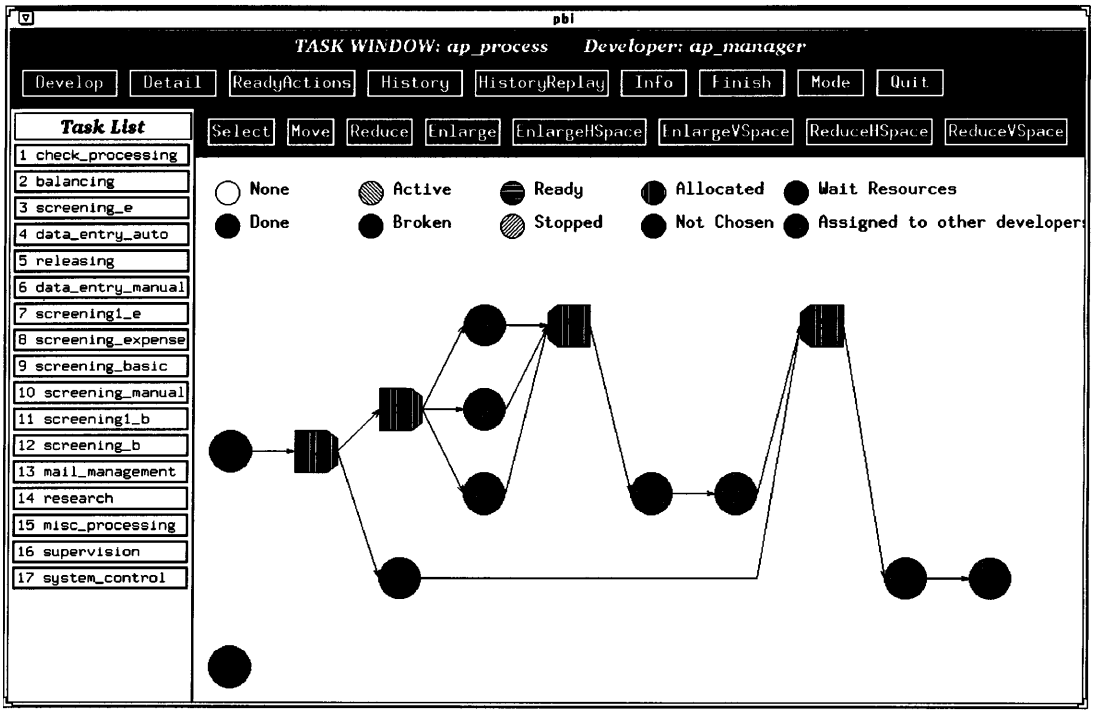

Figure 7 A top-level view of the Accounts Payable workflow

use of IT tools, systems, and data resources. This provides process users the support mechanisms and opportunity to try out or 'rehearse' new ways of doing their work through process redesign alternatives. Thus, since the Articulator accommodates partially decomposed OPMs, then these OPMs can also be downloaded into the process execution mechanism to visually display and interactively walk through role-specific usage scenarios.

We find this ability to walk through, tour, or rehearse process workflow prior to committing to its implementation extremely useful in supporting an OPM construction effort. In this way, we can support the iterative, incremental specification and refinement of OPMs in an improvement-oriented evolutionary sense. Accordingly, we have found that process prototyping is an effective enabler for eliciting user feedback. This helps to facilitate user-level understanding of their processes when supported by a process-driven computing environment. Process prototyping also provides a basis for user empowerment in controlling the design, refinement, and improvement of local processes. This helps to facilitate the adoption of redesigned business processes, since the people who perform the process can tailor them to better support and accommodate their process expertise.

We have also found that process prototypes can be sufficiently enriched to support process training and task performance. In particular, given the knowledge-based representation and processing mechanisms for upstream process engineering, we find that we can automatically generate multi-perspective views, documentary content, and tool invocations that support on-demand process training and navigation as an aid that supports process performance.

Gery (1991) defines an **electronic performance support system** (EPSS) as the use of computer-based systems to provide on-demand access to integrated information, guidance, advice, assistance, training, and tools to enable high-level job performance with a minimum of support from other people. The goal of an EPSS is to provide whatever is necessary to ensure performance and learning at the moment of need. As such, we have developed a facility for generating a process-centered EPSS from knowledge-based OPMs.

We have been experimenting with the use of new paraphraser that generates views and descriptive content from OPMs represented within the Articulator's knowledge base. Views are generated that center on the focal process flow, its description, participating organizational roles and task responsibilities, inputs and outputs, and supporting IT tools and systems. Further, we have adopted the use of descriptive templates—generic descriptions shells—that are coded in the hypertext markup language (HTML) for publication and wide-area distribution over a corporate intranet or the Internet. In this regard, this paraphraser is directed to generate certain kinds of content that populate the format coded into the description templates. Using this approach, we can generate OPM-based EPSS capabilities whose content can be delivered globally, but updated from a single point (the OPM specification). Figure 8 shows an example of the beginning of the first page of content generated for an Accounts Payable process that includes a generic prologue, and a process flow diagram produced by a process simulation tool noted earlier. Similarly, other pages from a generated web of performance support materials can be produced from a paraphrase of an OPM. Figure 9 shows another page that follows from links (not shown) on the page in Figure 8.

Finally, it is worth noting that the ability to accommodate end-users or process performers to update the performance support content corresponds to their ability to modify the OPM specification. This in turn represents an ability to provide support for user-directed **dynamic** refinement and on-demand generation of performance materials (Laffey, 1995), which can further facilitate the learning, codification, and ownership of corporate process knowledge (Stein and Zwass, 1995).

INTEGRATION, ENACTMENT, AND HISTORY CAPTURE

The process execution mechanism and PBI provides IT tools, systems and associated data

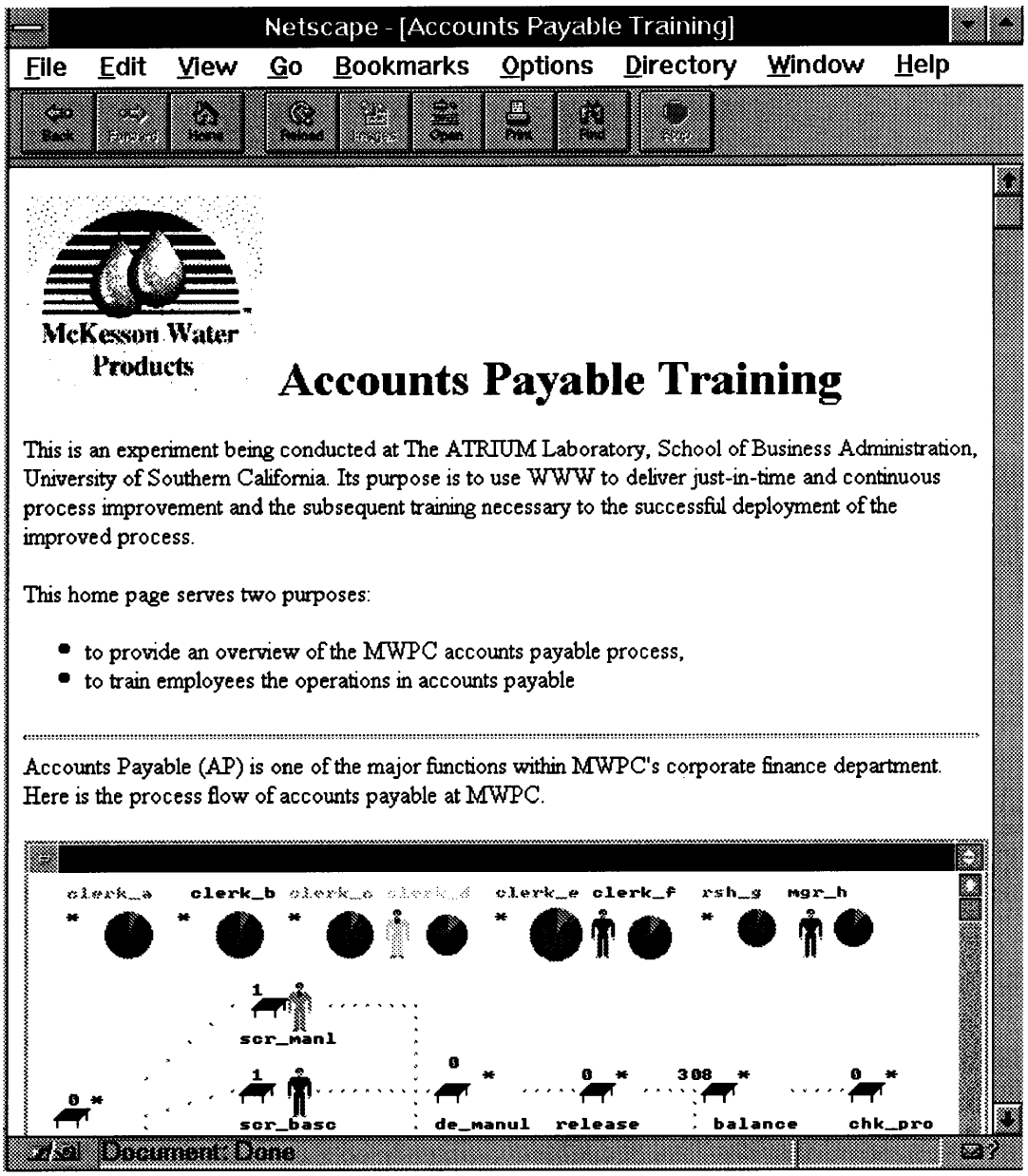

Figure 8 Intranet-based EPSS content generated from an Accounts Payable Process Model

resources (e.g. objects, files, databases, spreadsheets) which are served to users 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 using the IT tools, systems, and data resources delivered to them at their displays and fingertips when needed. Figure 10 shows an example of a process enactment view, which provides the IT applications, tools, data objects and workspace appropriate for the user assigned to this order-fulfillment process action.

Process enactment is also a computational

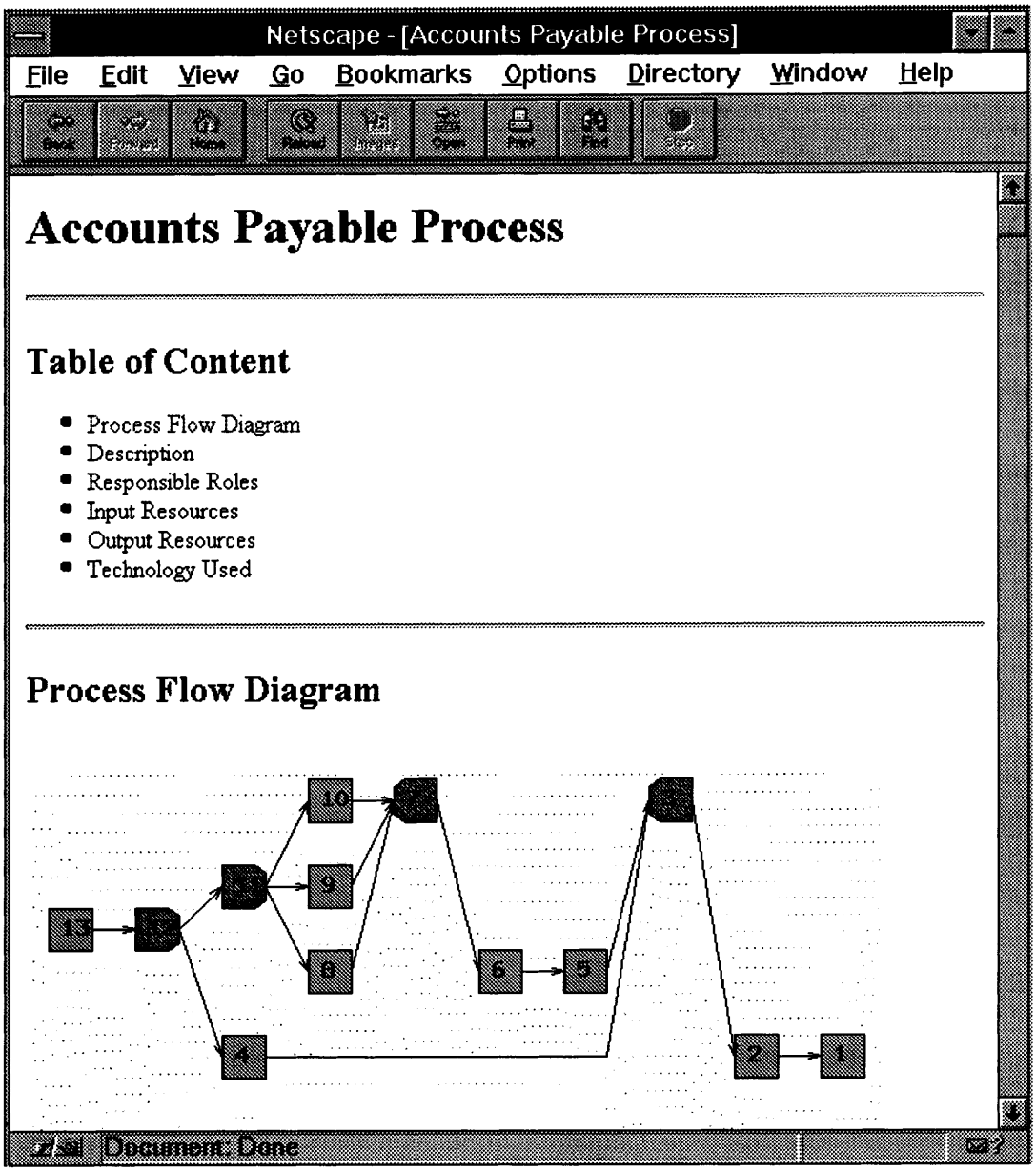

Figure 9 A later page in a web of performance support content generated from an Accounts Payable Process Model.

activity performed by the process execution engine. It interprets an OPM instance as its input. Thus, the OPM 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** (Osterweil, 1987). In this sense, our process programs are automatically derived from the process model specification by way of a special-purpose application generator (Mi and Scacchi, 1992; Karrer and

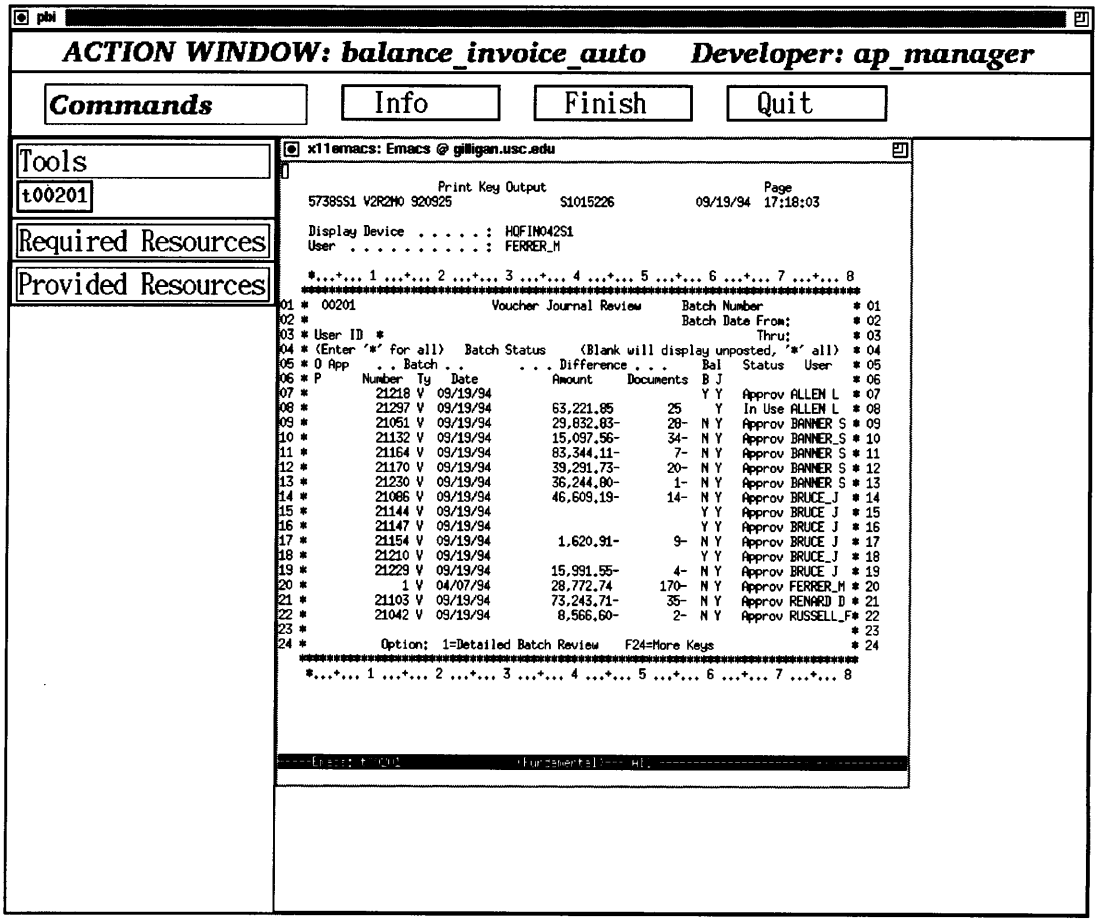

Figure 10 A lowest-level action workspace within the Accounts Payable Process Model.

Scacchi, 1993). Accordingly, the process enactment specification can incorporate any operating system command, system invocation script, or network access protocols to heterogeneous information repositories (Noll and Scacchi, 1991). 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 (Mi and Scacchi, 1992).

Process-guided work can seem onerous if the process forces people to do their work in an awkward, rigid, or unnatural manner. People can differ in the skill, knowledge, and prior experience in performing a process. Thus, we should not unnecessarily encumber an expert in a process by requiring her to follow the same sequence of process steps needed to guide and familiarize a newly assigned process novice. As a result, we have implemented a process guidance 'mode' variable that enables different levels of process guidance or enforcement to be followed. In this way, we have provided a mechanism for people in an organization, rather than us, to determine the policy for who needs to follow or conform to process guidance. We have found that this form of flexibility can serve as another mechanism to improve the acceptability and satisfaction with process guided support systems. Alternatively, it is also a way of expressing a lesson we learned for

how to make process enactment more accommodating to concerned process participants.

Finally, as process enactment can provide computer-supported guidance of work tasks, we have also incorporated a facility for keeping a history of what process steps were taken, in what order, and with what results. Such a facility can serve as a record or audit trail for processes when conformance to standards (e.g. adhering to financial controls) is needed. Figure 11 displays such a trace of process events for a 'sales' person performing part of an order fulfillment process. Furthermore, as this history record details who did what step at what time with what result (status update), we provided a PBI-based utility that allows the history to be replayed as a graphic animation of process flowgraph, such as that displayed in Figure 7. Similarly, the history record can serve as input to the simulation mechanisms described earlier. In both cases, this historical record serves to help gain insight for understanding how a modeled process instance got performed, which in turn can serve to support continuous process improvement efforts. As such, we have prototyped and demonstrated a number of process-driven environments in different business and government application domains that incorporate commercial off-the-shelf systems, internally developed systems, and prototype research software technologies that can operate over local-area and wide-area networks (Noll and Scacchi, 1991).

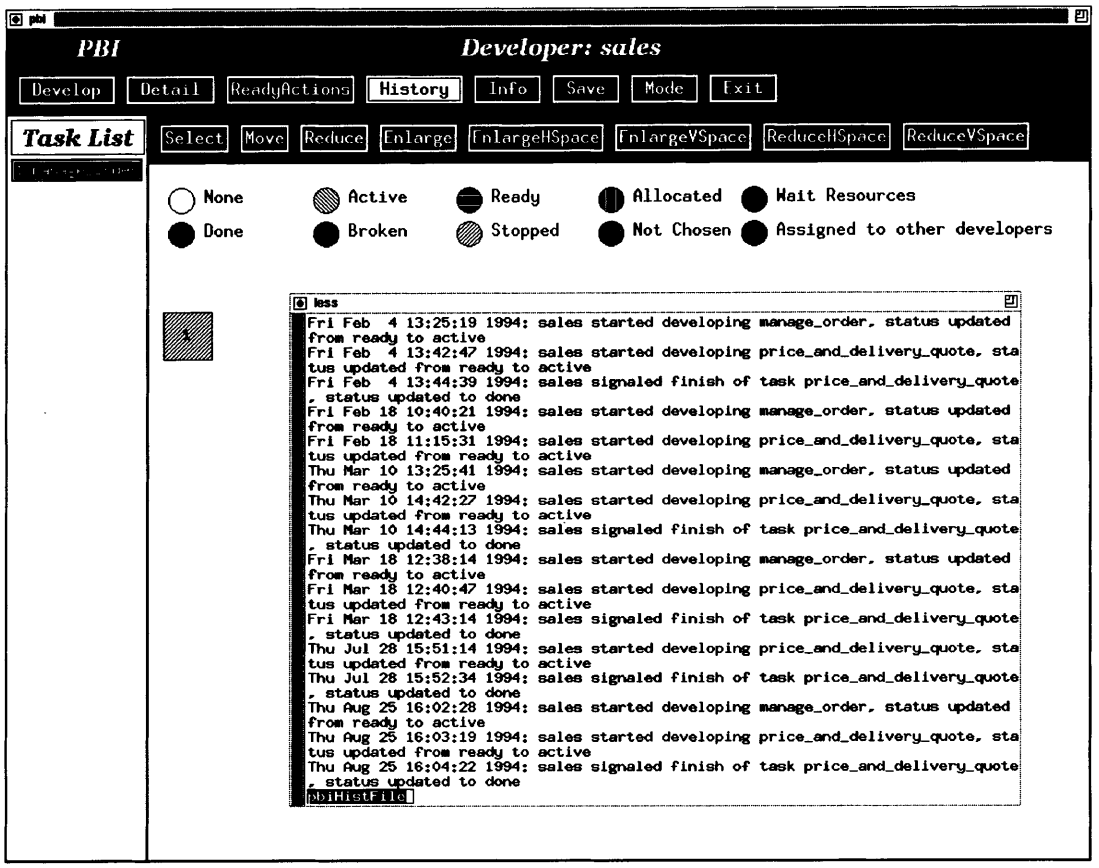

Figure 11 An historical record of process enactment events

ARTICULATION: REPAIRING PROCESSES THAT UNEXPECTEDLY BREAK DOWN

If one wonders why we chose to call our process engineering environment the Articulator it is due to our deep seated interest in articulation work. **Articulation work** refers to the activities people at work engage in when routine processes do not conform to the routine (Bendifallah and Scacchi, 1987; Suchman, 1987; Mi and Scacchi, 1991). This is when business processes activities break down or fail to apply. As a result of such dilemmas, people make circumstantial accommodations or negotiate with others for assistance and adjustments. Articulation work then entails the effort to figure out what's gone awry, and how to identify and implement ways to fix up, work around, or hand-off the dilemma at hand.

In open-ended real-world work settings, articulation work is common and widespread. Even a highly repetitive business process, such as the processing of invoices and issuing checks with an AP process, is never an activity that is repeated in an identical manner each time it is enacted and performed. Idiosyncratic circumstances, contingencies, and the movement of people and resources through time and space may account for this. However, it is a mistake to believe that business processes that involve people can be structured and constrained to be repetitive without variation. Thus, handling exceptions, special cases, and other contingencies, whether anticipated (i.e. as low probability events) or unanticipated, is in some sense a normal part of business processes and work. Logically robust but rigid computational procedures or other mechanistic means that disregard or ignore articulation work are doomed to fail, or to make adaptive work efforts troublesome. Accordingly, we have sought to develop an alternative to rigid automated process enactment mechanisms that recognizes and seeks to support articulation work.

We have developed a KB approach to supporting process articulation work (Mi and Scacchi, 1991, 1992, 1993). Although our efforts initially focused on activities in the domain of software engineering, we sought to develop representation scheme and processing mechanism that treated process articulation as a domain-independent phenomenon. In this regard, our approach to support process articulation using the Articulator focuses on three-phase computation involving diagnostic classification, replanning, and rescheduling (Mi and Scacchi, 1993). In our approach, we assume that business processes can be formally modeled then simulated or enacted using an Articulator-based environment. As the process model or model instance accounts for the logical representation of the process, and not all the circumstances and resource configurations at hand in the workplace at any moment, then contingencies can arise that are not part of the planned process instance. Thus, the enactment of the process instance will in some way not be effective. As all entitities represented in the Articulator's KB are a sub-class of the root class of 'resource', then any problem that the Articulator can possibly detect or reason about must be bound to some type of resource. Subsequently, in the Articulator's resource-based ontology, different classes of resource failures or breakdowns are the root cause of articulation work. Therefore, when a process enactment gets into trouble, it is due to a resource-based problem and causality. If the resources are those directly involved in enacting the process (people, IT tools and systems, task inputs and outputs, etc.), then it may be possible to detect their breakdown, to seek courses of action leading to a repair, and to resume the process instance if successfully repaired.

When a person (or simulated agent) encounters a process resource breakdown, the Articulator can be provided with the **context** of the problem at hand. This includes the model of the process being performed, the history of what steps have been completed so far, the resources (classes and instance values) involved in the process's enactment, and the schedule (if any) for completing the remaining process steps. The Articulator then performs a diagnostic classification of the resource breakdown, seeking to match with one or more of the known classes of process breakdowns that we have categorized through various empirical studies and abstraction (Mi and Scacchi, 1991). A view of this taxonomy is shown in Figure 12.

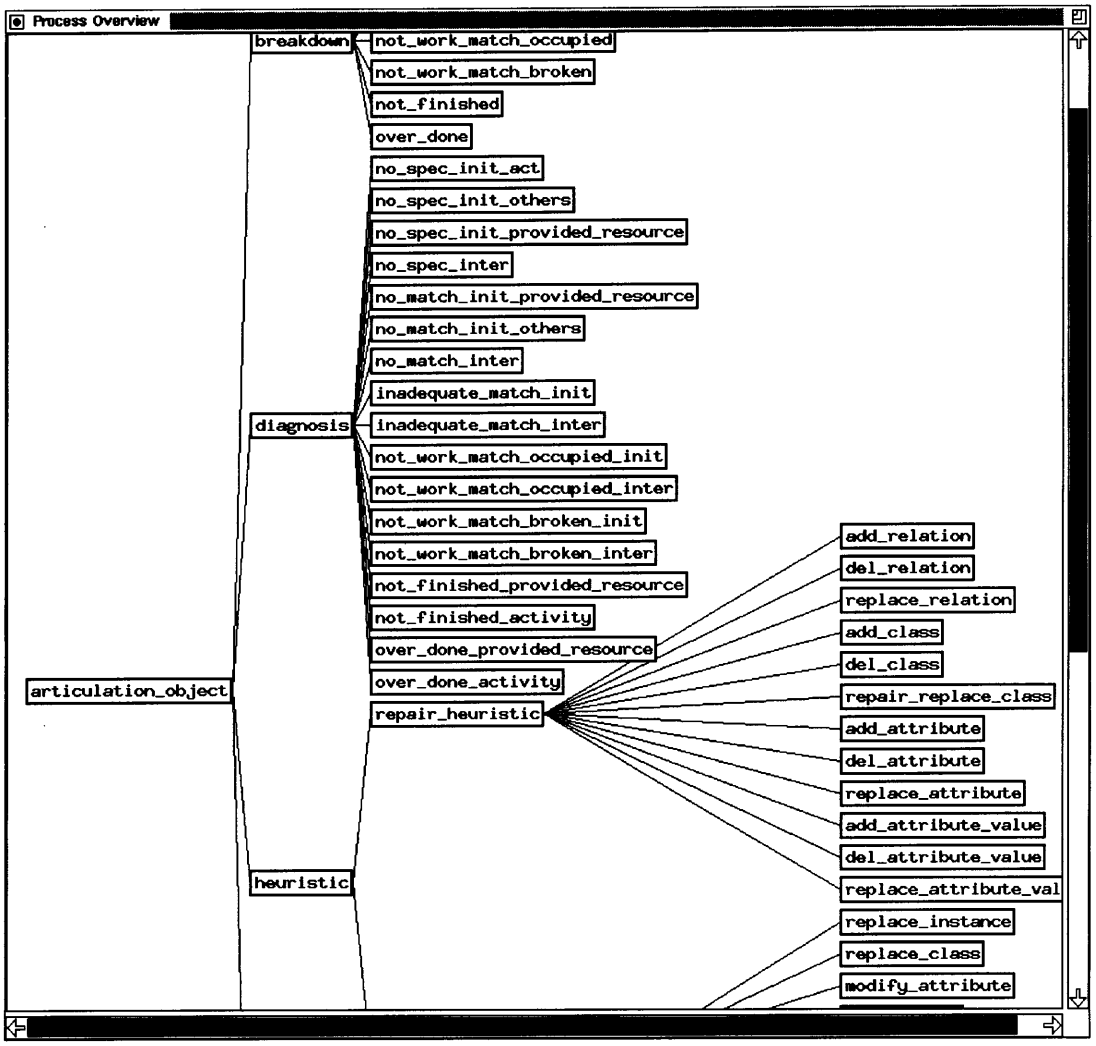

Figure 12 A partial view of a classification taxonomy for articulation breakdowns, diagnoses, and repair heuristics.

As a given breakdown or failure may not have a single cause, then the Articulator's breakdown classifier may find one or more possible classes of failure. In turn, each classified breakdown returns an index value which then guides processing in the second phase, the search for a root cause (or causes) of the breakdown. Again one or more causes may be found for each instance of a breakdown. As causes are surfaced, then another search begins to find possible ways to repair the process instance so that work can in some way continue. Repair heuristics, also derived from empirical study, are searched, and the results that match are returned in a partial order. These repairs look for information represented in the process enactment context to see what resources (e.g. other people, tools, alternative inputs, etc.) are nearby which may be available for use to repair the breakdown. If a repair is found and integrated, process enactment can resume, or be modified to accommodate the circumstances. Repairs can entail modifying the process instance by reconfiguring task flow by

gaining access to other resources, or by handing off the problem to someone else and defer completion of this process enactment instance.

Next, the choice for which repairs to make involves selecting from the alternatives that have been found. In the Articulator, this form of replanning involves evaluation of both local (context-specific) and global constraints that guide the selection of repairs. Such evaluation may in turn evaluate constraints imposed by a schedule or completion time target for the process enactment. In this way, process instance repair involves the interaction of replanning and rescheduling rule-bases, each of which includes between 150–200 rules for this purpose. After this computation cycle completes, a modified version of the remaining process enactment instance is produced.

Finally, if a suitable repair can be found, replanned and rescheduled, we face a final dilemma—that of determining whether the root cause of the breakdown or failure was in fact circumstantial (specific to this enactment instance) or systemic (specific to the process model). Our experience has been that this is still an empirical issue, requiring observations of recurrence or lack thereof. Accordingly, we use these articulation support mechanisms to suggest and repair process enactment instances under user control, and record their occurrence in a process repository for later evaluation. This in turn also serves to provide a basis for improvements to the process model. Nonetheless, the study and development of computational mechanisms needed to understand and support articulation work, as well as process enactment repair, remains a rich area for further research.

OTHER ADVANCED KBPEE TECHNOLOGIES

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

- Formalization of the knowledge representation meta-model (Mi and Scacchi, 1996) and product model (Choi and Scacchi, 1996)
- Process scheduling and administration (Mi, 1992)
- Knowledge-based repository for process model assets (Mi *et al.*, 1992)
- Process-driven information systems engineering and re-engineering environment (Choi and Scacchi, 1991; Mi and Scacchi, 1992)

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 business 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.

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 complex organizational processes and architectures for process-centered application support environments. Our earliest efforts focused on the processes associated with the engineering of software systems (cf. Selfridge and Terveen, 1996). One of the insights we gained from these experiences was that we are likely to achieve the most dramatic results and payback from a process engineering effort when focused on processes that (1) have a high rate of instantiation (are performed frequently), and (2) have relatively short process cycle times. Some software engineering processes fall into this category (e.g. software reviews, inspections, configuration management, and software testing), while others do not (software requirements analysis and system design). This insight perhaps foreshadows where significant gains and favorable return on investments in process reengineering efforts are most likely to be real-

ized when using a process life cycle engineering approach. Thus, when selecting opportunities for process improvement or reengineering in an application domain such as corporate financial operations, high-frequency and short-duration processes, such as those found in the handling of accounts payable and account receivable, are prime candidates, whereas processes associated with quarterly or annual book closings would not.

Over the past five or so years, we have broadened our focus to business process domains such as corporate finance, military procurement, electronic commerce, and supply chain logistics. Much of the early research into process engineering can be attributed to efforts focused on processes for software engineering (Curtis *et al.*, 1992; Scacchi and Mi, 1993; Heineman *et al.*, 1994). However, the focus is broadening to include other business processes and workflow technologies (Sheth *et al.*, 1996). Nonetheless, we will focus our attention to those efforts in process engineering that employ a knowledge-based approach, without restriction to application domain.

Work by Jarke, Mylopoulos, Yu and colleagues (Jarke *et al.*, 1990; Yu and Mylopoulos, 1994, 1996; Yu *et al.*, 1996) have developed knowledge-based representations for modeling and reasoning about complex business processes. Their approaches are similar in spirit to our approach to process meta-modeling, modeling, and analysis. However, process simulation or execution activities have not yet been addressed. The PSS project in England (Bruynooghe, *et al.*, 1991) developed an approach to process modeling that also supports an enactment language based on an object-oriented knowledge representation notation, as we have done as well. The Grapple system (Huff and Lesser, 1988), on the other hand, relies on a set of goal operators and a planning mechanism to represent process enactment operations. These are used to demonstrate goal-directed reasoning about software engineering processes during modeling and enactment. While these efforts lack support for upstream process engineering, the Articulator lacks support for generative process planning. However, the Articulator does provide a replanning mechanism for repairing process instance enactments that breakdown.

The AP5 environment (Balzer and Narayanaswamy, 1993), developed at the USC Information Sciences Institute, and the Marvel environment (Kaiser and Feiler, 1987) and successors developed at Columbia, use pattern-directed inference rules to model and trigger software engineering process actions during process enactment. AP5 has an implementation of the Articulator process meta-model, which was used to experiment with the integration of heterogeneous process enactment environments. The SMART environment developed at Hewlett-Packard Laboratories and USC also successfully undertook a similar experiment (Garg *et al.*, 1994). Marvel supports the creation of process models through rule-chaining which trigger automated events or procedures. However, its strength lies in its ability to support rule-based reasoning and generation of process integration parameter values that then provide reactive guidance for process enactment (Heineman *et al.*, 1992). In contrast, the Articulator and SMART only provides a mechanism for generating and integrating proactive process-driven application development or execution environments.

Beyond these efforts, work by Selfridge and Terveen (1996) and Stein and Zwass (1995) draw attention to the need to provide supporting mechanisms for the capture, management, and update of the **knowledge and learning** associated with business process performance and expertise. The acquisition and employment of this knowledge can be important when redesigning process structure or flow. People who perform mundane, arcane, or creative business processes as part of their work often possess deep knowledge about the intracacies, weaknesses, and failings of their processes (Suchman, 1987). Cavalier disregard of this knowledge and expertise in process redesign efforts is a likely contributor to the recurring failure of such efforts. Similarly, getting process participants to understand and learn how to perform their work in a redesigned process requires more than simply showing them process visualizations or providing them with nominal training seminars. As learning scien-

tists such as Roger Schank (1994) have found, people learn in different kinds of ways requiring different kinds of 'learning architectures' to support their learning. These architectures, when realized as computer-based support environments, should provide modalities for learning through modeling, analysis, simulation, rehearsal, performance support, enactment, and articulation. Thus, we have also sought to support knowledge management and learning through our approach to process life cycle engineering.

In sum, no other process engineering environment today supports the full process life cycle. However, we have investigated and demonstrated supporting mechanisms for each of the process life cycle activities described earlier. Similarly, while 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 (Davenport, 1993).

CONCLUSION

This paper 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. Thus, we believe our approach allows us to construct and demonstrate a knowledge-based process engineering environment for modeling, enacting, and integrating team-oriented process-drive IT-based work environments for redesigned business and government organizations. Furthermore, we believe this approach can be used by others to produce similar results, especially when effort is directed toward the goal of capturing, managing, and creating value out of the knowledge that process participants bring to their work when performing business processes.

We also sought to convey where we believe the process engineering effort should focus their efforts in order to realize the greatest return for the least amount of effort. We have consistently and repeatedly received positive feedback from our corporate sponsors on the subjective value and eye-opening insights they have experienced as a result of the application of our approach to the engineering of their selected processes. When the costs and benefits have been quantified and systematically measured (for example, as depicted in Figure 6), we could justify or compare the value of alternative process redesigns. Similarly, when the expected gains are rolled up to annual numbers, our experience has been that some of our corporate sponsors attribute five- to seven-figure annual returns to processes or operations that we helped to engineer using the tools, techniques, and concepts described here (cf. Bartholomew, 1994).

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

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 Service Excellence (CSE) 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, Mark Nissen and John Noll. We also want to thank Buzz Adams, Chief Financial Officer at McKesson Water Products Company at the time of this study, for his encouragement and support of this study. All these contributions are greatly appreciated.

References

Bartholomew, D., Keeping water flowing in L.A.: how McKesson met customer demand after the quake', *Information Week*, No. 463, 14 February 1994, pp. 41–43.

Balzer, R. and Narayanaswamy, K., 'Mechanisms for generic process support', in *Proc. First ACM SIGSOFT Symp. Foundations Software Engineering*, pp. 9–20. ACM, *Software Engineering Notes*, **18**(5), December 1993.

Bendifallah, S. and Scacchi, W. 'Understanding software maintenance work', *IEEE Trans. Software Engineering*, **13**(3), 1987, pp. 311–323.

Boar, B.H., *The Art of Strategic Planning for Information Technology: Crafting Strategy for the 90s*, John Wiley, New York, 1993.

Bruynooghe, R.F., Parker, J.M. and others, 'PSS: A system for process enactment', in *Proc. of the 1st International Conference on the Software Process*, pp. 128–141, Redondo Beach, CA, October 1991.

Curtis, B., Kellner, M. and Over, J., 'Process modeling', *Communications ACM*, **35**(9), 1992, pp. 75–90.

Choi, S.C. and Scacchi, W., 'SOFTMAN: an environment for forward and reverse CASE', *Information and Software Technology*, **33**(9), November 1991.

Choi, S.C. and Scacchi, W., 'Assuring the structural correctness of software life cycle descriptions', submitted for publication 1996.

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

Dhar, V. and Jarke, M., 'On modeling processes', *Decision Support Systems*, **9**(1), 1993, pp. 39–49.

Federal Electronic Commerce Acquisition Management Program Office, 'Streamlining procurement through electronic commerce', National Institute of Standards and Technology, Washington, DC, October 1994. http://snad.ncsl.nist.gov/dartg/edi/arch.html

Garg, P.K. and Scacchi, W., 'ISHYS: designing intelligent software hypertext systems', *IEEE Expert*, **4**(3), 1989, pp. 52–63.

Garg, P.K., Mi, P., Pham, T., Scacchi, W. and Thunquest, G., 'The SMART approach to software process engineering', *Proc. 16th Intern. Conf. Software Engineering*, Sorrento, Italy, 1994, pp. 341–350. Also appears in Garg, P.K. and Jazayeri, M. (eds) *Process-Centered Software Engineering Environments*, IEEE Computer Society, New York, pp. 131–140, 1996.

Gery, G., *Electronic Performance Support Systems*, Ziff Institute, Cambridge, MA 1991.

Grant, R.M., 'The resource-based theory of competitive advantage: implications for strategy formulation', *California Management Review*, **33**(3), 1991, pp. 114–135.

Heineman, G., Botsford, J.E., Caldiera, G., Kaiser, G.E., Kellner, M.I. and Madhavji, N.H., 'Emerging technologies that support a software process life cycle', *IBM Systems J.*, **32**(3), 1994, pp. 501–529.

Heineman, G., Kaiser, G.E., Barghouti, N., and Ben-Shaul, I.Z., 'Rule chaining in Marvel: dynamic binding of parameters', *IEEE Expert*, **7**(6), December 1992, pp. 26–34.

Huff, K.E., and Lesser, V.R., 'A plan-based intelligent assistant that supports the process of programming', *ACM SIGSOFT Software Engineering Notes*, **13**, November 1988, pp. 97–106.

Jarke, M., Juesfeld, M., and Rose, T., 'A software process data model for knowledge engineering in information systems', *Information Systems*, **15**(1), 1990, pp. 86–115.

Karrer, A., and Scacchi, W., 'Meta-environments for software production', *International Journal of Software Engineering and Knowledge Engineering*, **3**(1), 1993, pp. 139–102. Reprinted in Hurley, D. (ed.), *Advances in Software Engineering and Knowledge Engineering*, **4**, 1995, pp. 37–70.

Kaiser, G.E., and Feiler, P., 'An architecture for intelligent assistance in software development', in *Proc. of the 9th International Conference on Software Engineering*, pp. 180–187, Monterey, CA, April 1987.

Kling, R., and Scacchi, W., 'The web of computing: computer technology as social organization', in Yovits, M., (ed.), *Advances in Computers, Volume 21*, Academic Press, New York, 1982, pp. 3–90.

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

Laffey, J., 'Dynamism in electronic performance support systems', *Performance Improvement Quarterly*, **8**(1), 1995, pp. 31–46.

Leymann, F., and Altenhuber, W., 'Managing business processes as an information resource', *IBM Systems J.*, **33**(2), 1994, pp. 326–348.

Mi, P., *Modeling and Analyzing the Software Process and Process Breakdowns*, PhD dissertation, Computer Science Dept, University of Southern California, Los Angeles, CA, September 1992.

Mi, P., Lee, M. and Scacchi, W., 'A knowledge-based software process library for process-driven software development', *in Proc. 7th Knowlege-Based Software Engineering Conference*, McLean, VA, September 1992.

Mi, P. and Scacchi, W., 'A knowledge-based environment for modeling and simulating software engineering processes', *IEEE Trans. on Knowledge and Data Engineering*, **2**(3), 283–294, September 1990, pp. 283–294. Also appears in *Nikkei Artificial Intelligence* (in Japanese), **24**(1), January 1991, pp. 176–191.

Mi, P., and Scacchi, W., 'Modeling articulation work in software engineering processes', *Proc. of the 1st International Conference on the Software Process*, pp. 188–201, October 1991.

Mi, P., and Scacchi, W., 'Process integration in CASE environments', *IEEE Software*, **9**(2), March 1992, pp. 45–53. Also appears in Chikofski, E. (ed.), *Computer-Aided Software Engineering* (2nd edition), IEEE Computer Society, New York, 1993.

Mi, P., and Schacchi, W., 'Articulation: an integrated approach to diagnosis, re-planning, and re-scheduling', in *Proc. 8th Knowledge-Based Software Engineering Conf.*, pp. 77–85, Chicago, IL, 1993.

Mi, P., and Scacchi, W., 'A meta-model for formulating knowledge-based models of software development', *Decision Support Systems*, **17**(3), 1996, pp. 313–330.

Nissen, M., 'Valuing IT through virtual process measurement', *Proc. 15th. Intern. Conf. Information Systems*, Vancouver, Canada, pp. 309–323. December 1994.

Nissen, M., *Knowledge-Based Organizational Process Redesign: Using Process Flow Measures to Transform Procurement*, unpublished PhD dissertation, IOM Dept, USC School of Business Administration, Los Angeles, CA, January 1996.

Noll, J., and Scacchi, W., 'Integrating diverse information repositories: a distributed hypertext approach', *Computer*, **24**(12), December 1991, pp. 38–45.

O'Leary, D., 'Validation of expert systems—with applications to auditing and accounting expert systems', *Decision Sciences*, **18**, Summer 1987, pp. 468–486.

O'Leary, T., Goul, M., Moffit, K.E., and Radwan, A.E., 'Validating expert systems', *IEEE Expert*, **5**(3), June 1990, pp. 51–59.

Osterweil, L 'Software Processes are software too,' *Proc. 9th Intern. Conf. Software Engineering*, Monterey, CA, IEEE Computer Society, April 1987, pp. 2–13.

Scacchi, W., 'The power of domain-specific hypertext environments', *Jour. Amer. Soc. Information Science*, **40**(3), 1989, pp. 45–53.

Scacchi, W., and Mi, P., '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.

Schank, R., 'Active learning through multimedia', *IEEE Multimedia* **1**(1), Spring 1994, pp. 69–78.

Selfridge, P.G. and Terveen, L.G., 'Knowledge management tools for business process support and reengineering', *Intelligent Systems in Accounting, Finance, and Management*, **5**(1), 1996, pp. 15–24.

Sheth, A., Georgakopoulos, D., Joosten, S., Rusinkiewicz, M., Scacchi, W., Wileden, J., and Wolf, A., Report from the NSF Workshop on Workflow and Process Automation in Information Systems. Technical Report UGA-CS-TR-96-003, Dept of Computer Science, University of Georgia, October 1996. http://lsdis.cs.uga.edu/activities/NSF-workflow/

Simmons, R.F., and Slocum, J., 'Generating English discourse from semantic networks', *Communications ACM*, **15**, 1972, pp. 891–905.

Stein, E.W., and Zwass, V., 'Actualizing organizational memory with information systems', *Information Systems Research*, **6**(2), 1995, pp. 85–117.

Suchman, L.A., *Plans and Situated Actions: The problem of human–machine communication*, Cambridge University Press, New York, 1987.

Votta, L., '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.

Yu, E.S.K., and Mylopoulos, J., 'Understanding "why" in software process modeling, analysis, and design', *Proc. 16th Intern. Conf. Software Engineering*, Sorrento, Italy, pp. 159–168, 1994.

Yu, E.S.K., and Mylopoulos, J., 'Using goals, rules and methods to support reasoning in business process reengineering', *Intelligent Systems in Accounting, Finance and Management* **5**(1), 1996, pp. 1–13.

Yu, E.S.K., Mylopoulos, J., and Lesperance, Y., 'AI models for business process reengineering', *IEEE Expert*, **11**(4), August 1996, pp. 16–23.