Advanced Workflow Management Technologies

Gregory Alan Bolcer Endeavors Technology, Inc.
gbolcer@endtech.com, http://www.endtech.com/gbolcer.html

Richard N. Taylor Information and Computer Science
University of California, Irvine
taylor@ics.uci.edu, http://www.ics.uci.edu/~taylor/

Effort sponsored by the Defense Advanced Research Projects Agency, and Air Force Research Laboratory, Air Force Material Command, USAF, under agreement number F30602-97-2-0021. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency, Air Force Research Laboratory or the U.S. Government.

Summary - Process, workflow, and groupware projects both in the commercial and research worlds have each approached the problem of human communication and coordination with different sets of tools and technologies implemented from different perspectives. Activity dependencies and the philosophical assumptions about how work is performed are managed differently in each discipline. While most of these tools represent promising approaches, they have yet to be widely adopted, particularly in the face of increased usage of online information through the WWW and computer networks. Electronic information coming to the individual and into the corporation is becoming increasingly unmanageable. People are able to create information regardless of their location or method of access due to increasing mobility and interoperability of the tools they use. Because of the common concerns, trends are moving towards overlap and convergence of requirements. However, these communities have yet to identify the union of cross-discipline requirements for a supporting technical infrastructure addressing the level of flexibility, scalability, and support for evolution over time required in these real-world settings. This paper surveys the current approaches to project communication and coordination, details the range of requirements across several related disciplines, and identifies various tradeoffs and trends which may effect the adoption, design, and evolution of an advanced technical workflow infrastructure.

Keywords - workflow, process, groupware, requirements, tradeoffs, trends

1.0 Introduction

Workflow Management Technologies allow people and organizations to automate, manage, and improve the processes that govern interpersonal coordination and collaboration. Processes are the series of day-to-day activities in which people participate and may involve such diverse tasks as planning a project, submitting a travel receipt for reimbursement, getting approval for a proposal, circulating a memo for feedback, track-

ing customers, or writing software. These tasks by themselves are complex enough when performed by an individual, but adding the overhead of communication between several individuals makes the management of these processes difficult. The information needed to accomplish any particular activity may be dispersed around an organization and difficult to collect and use. Likewise, a workflow participant may not know which other participants depend on the documents or information artifacts being produced or the next step in accomplishing the task at hand.

What tools do process stakeholders, such as end-users or managers, have to really manage these processes? The answer is three-fold. The academic and industry disciplines of (1) software process, (2) basic workflow, and (3) groupware-based computer supported cooperative work (CSCW) approach the problem with different sets of tools implemented from different perspectives. Each manages the dependencies between activities differently in addition to making different philosophical assumptions about how the system is used to accomplish work. While these systems represent a promising approach to coordination, they have yet to be widely adopted. Usage has shown that no single system brings together all the features needed to address the level of flexibility and evolution required in a real world work context. This paper surveys the current technological trends in each of these disciplines, discusses several of the leading systems in the area, and extracts the relevant features of representative systems in order to define the properties needed in an Advanced Workflow Management System and the criteria for evaluating its usefulness.

This first section provides a brief introduction and a roadmap delimiting the scope of the paper with respect to current disciplines addressing coordination, communication, and control. Section 2 defines current technological shortcomings, provides a brief historical accounting, and outlines key differentiators between an advanced workflow management system and other approaches involving process, workflow, groupware, management information systems, project management systems, and commonly used software productivity tools. Section 3 analyzes an advanced workflow management system's requirements including a detailed look at support for multiple stakeholders, incremental adoption, object-oriented workflow, external tool integration, customization and reuse, distribution, and dynamic change. Section 4 explores some of the tradeoffs between the requirements enumerated in the previous section, identifies some of the most prominent trends, and illustrates the impact in the design of several systems with respect to these criteria. Section 5 concludes.

2.0 What is Advanced Workflow?

Advanced workflow is a amalgam of disciplines combining technologies from the domains of traditional or basic workflow, software process, groupware and computer supported cooperative work, management information systems, business process management, project management, and a common usage of widely available commercial off-the-shelf (COTS) software tools including drawing and spreadsheet programs. Advanced Workflow technologies differ from traditional offerings in that evolution is embraced as a fundamental philosophy in the design, deployment, and implementation of workflows supporting communication, coordination, and collaboration between individuals, groups, or organizations.

Workflow and process technology has the potential to provide major benefits with regard to information access, understanding, and efficiency. Information flow between stakeholders is critical. Workflow and process infrastructure provides the means to support seamless and timely transfer of, access to, and manipulation of pertinent information. Placing the right information at the right place and time is crucial for uninterrupted work including cooperation and collaboration. Discovering, modeling, and visualizing processes is also a precursor for improving daily work by allowing complex, error-prone, or tedious tasks to be understood and automated. While traditional workflow management technologies are useful, they have yet to be widely adopted in areas where they are most needed. The current generation of technologies generally:

- Assume a fixed user model
- Require an all-or-nothing buy-in
- Lack the execution level semantics to do optimization and analysis
- Are point solutions, not only for a domain, but over time
- Provide only limited types for modeling of products
- Embed inflexible control policies which occlude reactive control, graceful exception handling, and decentralized coordination

2.1 Genealogy of Disciplines

In the early 1900's Frederick Taylor called public attention to the problem of 'national efficiency' by proposing to eliminate 'rule of thumb' management techniques and replacing them with the principle of *scientific management* [151]. Scientific management reasons about the motions and activities of workers and states that wasted work can be eliminated through careful planning and optimizations by managers. While Taylor was concerned about the inputs and outputs of manufacturing and material processes, computers and the information age brought about parallel concepts in the management and organization of information and its processes. Work efficiency could now be measured not by bricks or steel, but by their information flows. Information, in addition, has another dimension, namely quality. In the early days of information technology, it was seen as a tool for management to improve decision-making by evaluating alternatives **[138]**. As computers spread to the desktop, this view evolved to allow workers to make more informed decisions. The problem then became how to place the right information, in front of the right people, at the right time. While traditional workflow, software process, and CSCW share this goal, their ancestry differs significantly. Only in recent years have the morphological differences been put aside to try and find convergence between the tools, techniques, and disciplines.

Traditional workflow was first considered in image-based systems in order to help automate the flow of paperwork through an organization. This idea led to the concept of the paperless office. Image-based systems usually received documents which were then scanned into the system as images. These images could then be stored and handed off based on their number or other metrics to perform work balancing. The problem though was that more intelligent routing and scheduling was difficult because the contents of the images could only be deciphered by the human agents doing the work. Forms-based workflow supplanted most image-based systems in that forms could contain editable fields to allow multiple changes throughout the path of the document. Intelligent routing

is accomplished by examining values in the form and determining the next appropriate action and route based on processes and rules. Business Process Re-engineering (BPR) **[80][83]** uses business rules to determine actions and minimize mis-directed energies. Management Information Systems (MIS) route documents along fixed procedural routes and encourage adherence to the workflow process. Winograd and Flores **[66][167]** developed the theory of coordination and communication to model how people honor, request, and make commitments. By proposing the concept that work takes place as a 'cycle of coordination' between individuals, groups, and organizations, the perception of work was changed from a data-processing paradigm to computer support for human activities. Modern day workflow systems carry on this distinction and attempt to merge the modeling of human and computer executed activities, improve coordination **[24][119][125]**, expand on the current personal computer and network technology infrastructure to support evolution of activities.

CSCW, also known as groupware, has received a significant amount of interest over the past decade. CSCW which combines research in both the computer and social sciences, attempts to capture the inherent complexities of collaboration between people. As the adoption of workflow and office automation systems was limited due to overspecialization of work procedures **[50][51]** when they were first introduced, research turned to less constrictive models that encouraged collaboration. The development of such systems has been stilted by underestimating the dynamic, evolutionary nature of most collaborative activities. CSCW encompasses a broad listing of technologies. The most successful have been the original technologies of electronic mail (email) and messaging billboards (bboards), but inroads have been made in group decision making **[110]**. Modern day systems such as Lotus Notes **[115]**, carry on that tradition with their core technologies being organization and presentation of electronic, asynchronous messaging. Synchronous technologies such as video conferencing, shared whiteboards, internet telephony, while interesting, have been supplanted in favor of simpler technologies that work across a variety of platforms such as 'chat' (a one-to-many text-based communication technology) or 'talk' (one-to-one) **[76][77]**. Because capturing real work practices was the goal of most CSCW systems, temporal ordering of activities, task dependencies, or the dynamic manipulation and scheduling of these constructs and work models was excluded from many early systems. While these systems placed minimal structural requirements on the participants work, they required a fixed collaboration model. Systems like Object Lens **[112]** added a dynamic aspect to collaboration. wOrlds **[64][65]** augmented this by allowing multiple, dynamic collaboration models to be chosen by the participant during the planning and execution of activities being performed and by adding the dimension of temporal trajectories as the work activity progresses. Some approaches to adding structured work involve negotiation models **[8]** or graphical presentations of collaborative activities. The most successful CSCW technologies are ones where a balance is maintained between unobtrusive work models and lightweight, low-cost of adoption and usage versus structured, managed work and reconfigurable collaboration technologies. Current research is addressing dynamic evolution. Introspect**[156]** provides an architectural framework supporting reconfiguration, the Obligations prototype **[31][32]** provides a continuum of formal to informal work models and inter-person negotiations, and AnswerGarden **[6][7]** exhibits self-organizing evolution including reflection **[46]** of the content and context determination. Each supports dynamism across a wide range of collaborative technologies including calendaring, scheduling, notification, and communication technologies.

A major catalyst for the identification of software process as a research discipline was the 1987 proclamation that “Software Processes are Software Too” **[130]**. This “shot heard ‘round the software world” launched dozens of research projects to explore the interactions between budding technologies for managing the complexity of software with budding technologies for managing the complexities of human communication and coordination in building these information artifacts. A similar impact was the ‘Spiral Model’**[30]** which instituted a shift towards evolutionary software development to more accurately reflect the dynamic and incremental nature of software development processes as they are executed. The Arcadia **[98][99][153]** project was a research group studying process-based environments and identified several key areas for supporting them. Among them are process programming **[146]**, object management **[150]**, and user interfaces **[155]**. Interaction between these initiatives became apparent in second generation projects. Teamware **[164][166]** intermixed user interface and process programming research; ProcessWall **[88][89]** intermixed object management and process. Later, these areas were expanded to include hypermedia (including the WWW) and software architectures leading to 3rd generation environments offshooting from the project **[33][147]**. Concurrent to this project, but similar in scope was the Eureka Software Factory (ESF) **[55][57]**. ESF was a large research effort to study technologies for providing support to large software factories including their lifecycle processes and reference architecture. The consortium developed various process formalisms combining them with other technologies: specialized Petri net models **[79][134]**, software data modeling **[92]**, and document objects and rules **[52][97]**. Similarly, the Atlantis **[103]** project is exploring issues in process-centered environments with CSCW **[104][106]**.

Other independent initiatives introduced other key concepts. Early process modeling languages were borrowed from software development such as dataflow definitions using control and data flow techniques **[34][67]**, IDEF or SADT, entity-relationships **[35][78]**, and state charts **[85][107]**. Use of logic languages **[87]** were also explored. Flexible rule-chaining, and later flexible transaction support via WWW access and tool integration was also pursued **[18][100][102]**. The abundance of formalisms resulted in meta-models to assess the appropriateness of each **[39][141]** and detailed looks into execution requirements of a process **[11][54]**. Further research led to the proposal of a process execution reference framework similar to an operating system **[40]**, but further studies and experiments later highlighted the differences between human and non-human executed operations **[165]**. ProcessWeaver **[56]** allowed graphical depiction of processes with some relaxing of semantic formalisms and their representations leading to further projects in visual modeling and specification understandable by non-technical as well as technical users **[19][21][166]**. Integration of tools and software using multiple languages was explored in **[120]**. Slang and SPADE introduced the concept of reflection and virtual machine layers **[15][17]**. Later systems explored support for dynamic change and evolution in a process system **[16][68][69]**. Further explorations in dynamism and evolution included **[86][96][109]**. Endeavors **[33]** added in concerns of adoption, context adaptation, and transportable, lightweight mobile processes.

Many commonalities are starting to become apparent across disciplines **[45]** such as explicit, commutative work models **[142]** and a need to move beyond individual automation to group coordination and collaboration **[108][136]**. One thing they have all shared is lack of widespread acceptance for all but the simplest of systems, even in their home field of software. Adoption issues have been studied in each respective area, and while strategies and guidelines for successful adoption are available in all three disci-

plines (workflow and office automation **[29][49]**, process **[36][37][38]**, CSCW **[116]**), no single system has been flexible enough to address the broad spectrum of concerns.

2.2 Technology Limitations and Confounds

Traditional online work solutions come in various flavors with overlapping limitations which prevent their effortless adoption into a work environment. In addition, the applicability of these technologies is being demoted as work environments are becoming interconnected, allowing lightweight dissemination of information and artifacts using various networking technologies such as the WWW and the Internet. One such domain is virtual enterprise engineering (VEE **[60]**). Virtual enterprises require the need to evolve even to the point of swapping out organizational divisions, groups, and people, while at the same time guaranteeing precise, consistent interactions with high levels of assurance. Management and measurement infrastructure needs to be in place to guarantee that at any given point in time, the best interests of the virtual enterprise are served, usually with respect to trust, quality, time-to-market, profitability, efficiency, and other variables. Various traditional technologies do not live up to this task because of the following limitations and confounds.

2.2.1 Process Technology

Process technologies assume closed world data consistency. This is usually enforced by limiting the actions of the user or agent to those only pre-specified by the process designer. In a dynamic, evolving real-world environment, sometimes it is necessary to jump out of the process or out of the system to adhere to the specified process model. Process models that are over-specified are difficult to follow because their requirements for completion are too rigid. Process technologies that support evolution as a mechanism to address this problem require that changes are monotonic, i.e. each successive change to the workflow model is additive, incremental and always strongly dependent upon previous models for data integrity and consistency. Synchronization between a process model, its data, and offline activities is difficult in situations where a participant's activity diverges from the activity descriptions. Often this results from an exceptional case, escalated priority, or work redefinition. Consistency sometimes must be violated to meet both functional and non-functional requirements of the process being followed or the product being produced. Process models assume a uniform representation. Coordination between dispersed process participants is sometimes difficult because a uniform representation of activities, artifacts, and resources may differ among people, groups, and organizations. Adding the complication that various participants possess differing skills, different levels of understanding of their obligations, and may require domain and context specific views, misrepresentation and miscommunication is often the result.

2.2.2 Workflow Technology

Traditional workflow technologies have achieved relative success in the workplace through simplification. These systems have been applied to problems of limited scope and scale, usually automating a single point task as in an ad-hoc workflow, or a small series of never changing steps as in a production workflow. Workflow specifications are ambiguous or contain inflexible semantics. Non-graphical workflow specifications

often are hard-coded to include both the context and control policies. Understanding of the overall process is limited to those who can delve into the textual description. Graphical representations allow some abstraction but at the same time introduce ambiguity. Visual workflow programming languages often lack representations for timing and execution constraints as well as complex relationship specification and management between process objects and people. Lack of relationship management can precipitate poor exception handling.

Workflow processes that diverge from the intended series of planned activities or require some mid-stream changes result in exceptions. An exception can be something as simple as a missing resource or input; it can also be something more serious such as a design failure requiring significant amount of rework or alternate procedures. Workflow technologies typically lack the infrastructure to query status and change values of their own processes, correct mid-stream deviations, or locate and reserve needed resources and artifacts. All exceptions, whether requiring minor automated intervention or human participation to correct, often require the reset and restart of a workflow process. This rigidity in handling exceptional cases shows up in a typical workflow product's deployment model. Workflow processes once they are deployed into their execution context are rarely changed. Significant changes are difficult to accomplish in this environment as the workflow process model, the execution infrastructure, and the process presentation are tightly coupled.

2.2.3 Groupware and CSCW Technology

Groupware and CSCW are broad labels that describe a gamut of synchronous and asynchronous technologies including email, shared whiteboards, meeting schedulers, collaborative desktops, video conferencing, and other shared electronic media. While these technologies are useful for overcoming communication problems over time and collaboration problems over distance, they lack the guidance and automation mechanisms for performing structured tasks. While using these tools to accomplish unstructured tasks mimics some real world work environments, they don't guarantee consistency of results, maintain a standard of practice and procedure, nor lend themselves to optimization, improvement, and training. There is no explicit work model other than ad hoc utilization of the components at hand. There is no measurable status for determining completion, progress of a task, or even mechanisms for describing what work needs to be done other than capturing communication and collaboration relationships between participants. Adding a work model and management capabilities to a groupware or CSCW technology often leads to additional work on the part of the participant with no direct benefits such as guidance or automation. Synchronization between the proposed work and the actual work is often done by hand. This overhead of model maintenance may lead to the demotivation of the participants which is crucial to the success of an activity.

2.2.4 Management Information Systems

Management Information Systems (MIS) are domain specific software solutions for managing the dissemination and control of information across large organizations. These systems are custom built for specific information and automation needs, including domain and task specific structuring of data and documents as can be seen in health maintenance or corporate tax preparation. A large portion of MIS is simply translating or reformatting the data and documents into a format applicable to each specific task or context. Control flow between people and groups is usually embedded in the implemen-

tation of the system, and handoff of data artifacts is only discharged along pre-ordained communication channels. The ability of the MIS to adapt to changing work procedures is difficult because components aren't readily reusable and changes to a workflow task or data structure often have system-wide consequences making the cost of change prohibitive. Changes in execution behavior may require re-coding of data translators, user interfaces, and execution scheduling and semantics. Because of this fixed-execution model, evolution of the MIS over time is done through whole system upgrades. Divergence of a participant's work from the specified task or structured data is usually not captured and swept under the rug until the benefits outweigh the cost of changing the software by the information technology (IT) staff and not the participants themselves.

2.2.5 Project Management Systems

Project management systems model and track the progress of user tasks against dates, times, priorities, costs, and precedence dependencies. Tasks can be resource-driven or effort driven, and a critical path to completion can be specified using different cost and time constraints. While some project management systems have multiple views into the work model, most are applicable to a single stakeholder, namely the project manager. This mismatch reduces the burden of status collection for the manager, but increases the effort needed to keep the status reports and state changes reliable for the project participants. End-users that derive no direct benefit from using the system have no incentive to maintain the data it uses. This additional work may lend to the divergence of the work model and the actual status. Another problem with project management systems is that the formalisms they use overconstrain the work specifications with respect to time and resource dependencies. Accuracy of this information changes the further away project projections are made from the present. As the project progresses, the plan usually remains fixed, and evolution causes inaccurate data and high maintenance, and the 'best laid project plans' are cast aside.

2.2.6 Common Usage

Common usage describes a large class of desktop software that is used in the day-to-day informal planning and execution of workflow processes. These include drawing programs, spreadsheets, batch files and shell scripts. Most non-technical users can visually describe processes using a desktop diagramming tool, analogous to a casual drawing on the back of a napkin. The diagram is then used as a means to communicate the intended process as an informal flow chart, series of sketches, or graphical checklist. Execution mechanisms to support such workflow specifications are non-existent. Execution takes place by hand, and whatever automation that is incorporated into the workflow occurs by implementing the description using a scripting or macro language by a technical user. The problem with this is twofold. There is a complete separation between process model and process execution in that changes in one or the other cannot be reflected back in order to update the ongoing status. Secondly, workflow process descriptions lack formality and expressibility which causes miscommunication of tasks, confusion, an inability to perform analysis, an inability to determine status, and lack of agreement on common representations, task definitions, shared artifacts, or required resources.

3.0 Advanced Workflow Requirements

Because workflow involves the interaction of humans and computer processes to generate and share information, its requirements need to reflect both the technical as well as social and organizational needs. While some process, workflow, and groupware systems excel at particular subsets of these requirements, they have been limited in their level of flexibility, scalability, and evolution over time. This section draws upon the current research across various disciplines and assembles the key requirements assuming an information-rich and highly-connected world. Support for multiple stakeholders, incremental adoption, object-oriented workflows, integration of external tools, customization and reuse, distribution, and dynamic change are all explored in detail.

3.1 Support for Multiple Stakeholders

Support for multiple stakeholders is important in an advanced workflow system. **[5][110][165]** Stakeholders may have backgrounds in systems integration, business process modeling, process improvement and optimization, human factors, or even domain specific knowledge about the domain in which the workflow process is being applied. Stakeholders may be technical or non-technical. Interaction with the workflow system may be explicit, where the series of activities and dependencies are viewable by the end-user, or implicit, where the end-user is concerned with the local activities and may not be aware that they are part of a larger workflow process.

Each of these stakeholders may require a different presentation on the activities at hand. The interfaces presented to the end-user participating in a workflow process during execution is the work context. This may include interactions with tools independent of the workflow system. In the case the end-user only interacts with the tools with which they are familiar to accomplish their work, the user is participating in an implicit workflow. An implicit workflow usually has no formal and manipulable view available to the end-user concerning how their work fits into the larger context. Information about where to find the artifacts needed to accomplish their work is usually informally specified or left up to the individual. Designing an implicit workflow requires the process designer to anticipate the tools and data required to accomplish the activity. This information, as well as the technical details of integration and tool invocation are often specified in the workspace management component of the workflow system. The advantage of an implicit workflow is that in a smoothly running, repeatable workflow process, the end-user is unburdened with the communication overhead. Details of activities above and beyond the immediate task are abstracted away. To implement an implicit workflow, it is often necessary to present the information in the application domain. This requires customization of the views into the system either from the process designer’s or end-user’s perspectives.

While implicit workflows are useful for hiding details of a workflow process not appropriate to the end-user, making them explicit allows the participant to see how their activities fit into the greater context of the project. Greater coordination between process participants can be achieved by allowing end-users to view the data artifact dependencies and locations, tool availability for manipulating them, and explicit definition of which other participants are relying on their work. The expectation of data and document handoff or resource sharing has the self-enforcing effect of encouraging partici-

pants to meet their social contract obligations in a timely manner. Adding functionality to the end-user system to both understand and change the workflow processes in which they are participating makes the system flexible by allowing local optimizations with minimal or no impact on global process constraints. However, when imposed across different parts of an organization, these constraints may often in fact be conflicting. It is important that explicit workflow systems provide mechanisms to allow both the local optimization such as multiple solution paths, guidance, and multiple coordinated views, while at the same time not breaking or disrupting the flow of the global process by providing mechanisms for help and answer, data agents for resolution, enforceable rules and constraints, and a strong separation of concerns between the data artifacts and the process by which they are created.

3.1.1 Accessibility

System integrators, workflow process programmers, managers, and end-users all may need access to tools, interfaces, documents, status information, to-do and activity lists, and other resources.**[128]** The WWW has made it possible for non-technical end-users to quickly share information by publishing it on the Web. Intranets have made it possible to control that information in order to keep it proprietary and 'in the family'. While the availability and dissemination of information aids in the completion of tasks, that alone does not provide the framework within which it should be used. Similar to how non-technical end-users can both browse and author Web pages to disseminate information and reflect task progress, workflow processes should support the same capabilities. In addition to this, a wide audience of process stakeholders should be able to search, browse, initiate, view, edit, annotate, replace, and, at the very least, understand the workflow processes as appropriate.

3.1.2 Data Agents

Data agents are small, autonomous software modules that automatically perform a localized task. Intelligent agents, in addition, have the ability to reason about the input data and manipulate it in such a way that it can work in conjunction with other agents without violating any global constraints.**[84]** Examples of activities that data agents perform range from simple to complex. An agent's activities may include automatic notification via email of the availability of a report, sending a reminder of a scheduled meeting, advising a user of alternate meeting times as the result of a scheduling conflict, or even actively going out and doing research based on some specified criteria. Agent's activities, from the viewpoint of the workflow system, should allow the activity to be accomplished by either a human or non-human agent or both. By allowing the mix of both human and non-human activities, the availability of more intelligent agents allows for the incremental automation of certain activities. The people participating in the workflow should be allowed to hand off or delegate activities to an agent or take ownership of the activity back from one. Similarly, agents should be allowed to delegate tasks to sub-agents or request the help of human participants as needed. The management of multiple agents allows end-users to accomplish tedious or repetitive tasks with minimal effort and avoids the 'keep checking back' syndrome. Agents should allow customization by individuals with or without technical programming knowledge, notification based on existence and changes, constraints on values and appropriate strategies for resolution, and significant event filtering and topic interest registration. At some level of abstraction, an agent and an automated workflow process are interchangeable. Agents historically have been focussed on non-human involved, small-scale coordination activities.

3.1.3 Guidance

Guidance allows the workflow system to walk a number of process participants through a series of steps to accomplish a goal. Guidance can be used to teach or train as in a computer based learning system that teaches math or job specifics, or it can be used with trained and highly skilled participants to ensure quality of service, quality of outcome, or tracking adherence to standards.**[44][63]** An example of this is the health care industry where the workflows require an efficient flow of precise information and steps available to doctors, nurses, and administrators in order to ensure a certain standard of care is met.

Guidance should be flexible and configurable for the work context. Subsequent choices and their consequences should be made clear. Decisions on choosing the next activity are either made by the user after being advised by the system of their alternatives or by the workflow system based on data input and values. End-users should be presented with information detailing how far into the workflow they have gone and how far they have left. As an example, this is can be done with the number of questions in a training exercise, “you have finished 30 out of 40 questions”, or with time estimates, “sending off the results to the lab will take approximately 3 hours”. Guidance is related to enforcement. Systems that strictly limit the choices by the end-user enforce compliance with the process.

3.1.4 Separation of Concerns

Separation of concerns allows greater flexibility and dynamism in the workflow system. Decoupling components of the system allows them to be swapped out, replaced, or even added to the system without impacting other parts.**[19][21][75]** Strong separations between the workflow model and its presentation, the work context and the process, the process implementation and its design are all important.

In a system where the visualization and the process are tightly coupled, it is difficult to provide alternate views of the workflow. Design of a workflow process where the visual representation is the language with no underlying programmatically manipulatable model severely limits the interactions and automations that can be performed with the system, especially while the workflow process is executing. While graphical representations are useful for abstracting detail to allow model changes by non-technical users, this abstraction often causes ambiguity. This ambiguity is often resolved semantically at the model level which requires model manipulation independent of its presentation.

Workflow process models and the context in which they are executed should also be decoupled. The access and availability of tools, documents, and artifacts to the end-user should be configurable by both the workspace designer or the end-user themselves. By separating the semantics of the workflow from how the workflow is executed, processes can be easily reused in new and different domains or downloaded across a network and executed in the end-user’s work context without violating the constraints of their work culture. The separation of the workflow model from its execution context allows late-binding and dynamic loading of tools and resources as needed or guidance and enforcement policies as required. In a sense, the separation of the work context from the process model is analogous to cascading style sheets (CSS) on the WWW which separates Web content from the style guidelines. Similar to how the HTML renderer makes context dependent display decisions about its presentation, the workflow execution engine can use the workspace context to scope the execution.

Separating implementation from design allows changes to the process implementation such as language or tool integrations without changing the specification. Model changes can also be made by reusing implementations or supplying alternate ones. This separation allows late-binding of resources and flexible execution based on availability of these resources at the time they are needed. Also, because workflow process context differs across development, deployment, and execution, a separation of concerns minimizes changes needed to accommodate these stakeholders. In the workflow process world, testing a process is difficult without actually deploying or executing it. By minimizing the changes made to the process by being able to switch implementations and context, it is possible to use the same process for testing, simulation, feedback, and experimental design as well as actual deployment and in-place execution of the process. This allows smoother transition, adoption, and reuse models.

3.1.5 Solution Paths

The final outcomes and goals of a workflow process should allow for multiple solution paths to that goal. Completion of a process should not only be proactive as in a production workflow system with work being pushed along by the workflow engine, but reactive as well where new messages, events, and actions are triggered by some state change in the model.**[31][32]** Add to the equation exceptions, inpredictability of humans, and the common occurrence of work sometimes just going wrong such as miscommunications, missing or incomplete information, it becomes important to allow for multiple solution paths. Complex activities should be configurable to allow a range of options. The activity may be left unspecified relying on the ingenuity of the end-user to achieve the goal at hand. Otherwise several methods and processes for achieving the goal can be selected by the participant or automatically selected based on the availability of resources at the time. Process participants, whether human or automated agents, should be allowed to stray from the intended path if appropriate. This sometimes will violate global constraints such as communication and coordination with other process stakeholders. While it is important to provide alternate solution paths to encourage process improvement and optimization, the workflow system must be flexible enough to balance this with enforcing process obligations across organizations and levels of abstraction.

3.1.6 Cross Organizational

Organizations are large groups of people working together to deliver products or services. These groups usually focus on the sub-tasks of creating a larger product or service and unfortunately sometimes work at cross-purposes.**[65]** A quality assurance team may focus on ensuring a product is sufficiently tested while a sales team may want to release a product or service to stay ahead of the competition. Organizations are social constructs whose groups share terminology and closely related sets of tasks. Skills for an individual to succeed in one part of an organization don’t necessarily transfer easily into another. Likewise, tools, techniques, management styles, work ethics, and environments vary across an organization. This makes it difficult for various stakeholders to participate in the same process. Two individuals from different parts of an organization often do not even use the same terminology when speaking of the same project, much less share the same technical infrastructure.

A cross-organization workflow needs to be able to leverage and adapt to the existing infrastructure. Tools and software should be substituted as needed or else bundled in with the workflow process. Views and presentations should fit the user model and abstract away unnecessary detail or inversely annotate it with group specific informa-

tion. Mismatches of data or expectations which most often occur at organizational boundaries should be identifiable and allow for automated or human involved negotiation and resolution. Global rules and constraints should be enforced at the organization level, but allow for enough flexibility to allow groups to 'grease the cogs' of the process to reduce bottlenecks or handle exceptional cases. Finally because of the diverse social, political, and technical infrastructures, the implementation of the process itself should be cross-platform, lightweight, and mobile. This allows fragments of the process to be distributed and executed when and where they are appropriate or needed.

3.1.7 Rules and Constraints

Rules and constraints are the mechanisms used to maintain process and data consistency. Rules are formal or informal, local or global. Informal and global rules are often described as business directives such as 'all timesheets must be turned in on time or else the disbursement of funds will be delayed until the next pay cycle'. Other rules and constraints may alert a project manager when an activity misses a deadline or falls off the critical path. While natural language style rules are easy to specify, they are often ambiguous. The actual implementation of workflow rules should be separate from the specification and allow for as formal a description as is appropriate from the author.

Rules and constraints are separated into conditions and actions. Conditions should allow timing and synchronization, flexible pattern or value matching, and configurable generality and appropriateness.**[100][101]** Actions effect some notification or state change in the workflow system that may be visible or invisible to the process participants. The implementation of the action, similar to the implementation of the workflow process, should allow for flexible tool integration and support for multiple implementations which may be dynamically chosen. Rules can be mandatory or optional, and in the second case, end-users should be allow to turn on and off the firing of the rule. One example of this is email registration. An end-user can register interest in the posting of a report or document. The workflow system can then notify the appropriate parties of its availability.

Some workflow processes lend themselves well to being specified in a rule-based manner. The flow of the process is governed by the series of rules that it matches. These types of workflow engines are called reactive systems. The execution and firing of rules should be flexible also. Multiple or even conflicting rule matches should allow for prioritization. Missing or impartial information should allow for both forward and backward chaining, and the number of times a rule is allowed to fire should be configurable. Also, it is important to allow human interaction in both recognizing conditions and executing actions.

3.1.8 End-User Configurable Views

Workflow processes involve the coordination of activities, control of artifacts, and communication between stakeholders. Each of these may require presentation to an end-user in different ways or highlighting different aspects or importances. Some users may require domain specific information or layout while others may become confused or misled by extraneous or unneeded information. End-user configurable views are needed in addition to stakeholder views in that it allows the end-user to form a customizable context to which they can return.**[6]** Process participants may require functionality similar to how a group leaves their scattered papers in a meeting room when they break for lunch in order to preserve the context and make it easier to pick up the meeting when

they return. End-user configurable views, by allowing the participant to customize the workspace as appropriate and to their liking, increases the chances that the work will be done within the workflow context where it is easily captured and measured in order to share status with others dependent on the work. At the same time, it allows the end-user easy access to local state and information. This gives participants the ability to revise a document or test code until the point of handoff, as well as the ability to coordinate between multiple projects and processes.

3.1.9 Help and Answer

Help and answer is important not only when using a workflow system, but also when using and selecting processes. Help and answer should be context aware and configurable in the level of technical detail based on the role of the participant requesting help. Related concepts should make use of hyperlinks, and in the case of a question, the system should allow that the answer be a workflow process also if appropriate.**[7]** Help and answer implemented as a workflow helps determine the source of the problem, identify the appropriate resources to use or people to contact, and provide interactive guidance to converge on a relevant answer. In addition to end-user help and answer, workflow developers should have access to process wizards and templates.

Help and answer should include evolutionary capabilities as well. This may involve annotations on a process's usefulness, relevancy to certain domains, overviews, feedback, or even customization and reuse instructions. Information should be synchronized and co-evolve with the process. Techniques for accomplishing this include automatic generation of system APIs, tight integration with hyperlinks and hypermedia, and flexible searching and browsing including standard reference locations and libraries. Decoupling of the help and answer from the actual process allows for the latest information on the process or product to be updated over the WWW or other network protocols. Process and task specific tips and troubleshooting should also be available.

3.2 Incremental Adoption

Incremental adoption is key to the successful implementation, deployment, and execution of an evolving workflow system and its processes. Previous workflow and software process systems have taken a top-down, all-or-nothing approach to adoption. While this may work in some organizations, it inhibits these systems from being more widely accepted. Likewise, CSCW systems often require a plurality of users before the group benefits can be realized. Incremental adoption allows the benefits gained from using the system to match the amount of effort and resources available by controlling the scope and pace of adoption.**[29][36][37][38][49][76][77][116]**

Workflow systems often require a certain amount of computer expertise requiring training of personnel to both develop processes and participate in them. Often times this requires changing the work culture, and the technology is seen as getting in the way of accomplishing their work, significantly impacting the adoption and use of the technology. Participants who lack the ability to change or optimize the process either because of lack of skills and training or inflexibility of the system, tend to follow a parallel process duplicating work. Allowing end-users to evolve some or all of the workflow processes in which they participate amortizes the cost of adoption across all of the stakeholders but at the same time distributes the rewards.

Process stakeholders fall into the trap of duplicating work, both online and offline, because the tools they need to accomplish the task at hand are difficult to integrate with the workflow system. Also, process objects are difficult to reuse because they are usually over-specified and only relevant to a particular process, project, or individual work context. Few departments, groups, or individuals are willing to pay the overhead costs that benefit others down the line and are unable to incorporate global charges of time, money, or effort into current projects due to immediate resource constraints. Workflow objects that represent activities, artifacts, or resources tend toward special case behaviors because of context customization necessary for execution. The maintenance, configuration, and deployment of all these objects becomes difficult to manage and track. The cost of deployment of these workflows across diverse machines, protocols, languages, and installing minimum and consistent process tools and technology on every relevant desktop is very high. There are very few tools to distributedly manage all the necessary componentry, and most workflow systems are closed or difficult to customize because they have no open APIs.

3.2.1 Scalability

Workflow systems not only need to support individuals, groups, large organizations, or an entire enterprise, they must also gracefully scale with the number of participants and the size, complexity, scope, and purpose of the project over time. Performance is always an issue, but more importantly, end-users require access to information that is timely, appropriate, and easy to find. Abstraction is the key concept. End-users need to be isolated as much as possible from problems arising with the addition of significantly more users. Workflow systems can address this by hierarchically abstracting the workflow and its representations. In order to maintain consistent and coherent project data across large or increasing numbers of people, details that may embody large, multi-person workflows themselves can be encapsulated. Workflow systems should support scaling to meet the needs of the process developers and maintainers in addition to the end-users. This can be done with process support tools similar to how complexity in software is handled. Requirements and rationale tracking, debugging and analysis tools, optimization, reuse, standardization, version control, and banding groups of functionality at the same level of abstraction into a virtual machine are all techniques applicable to scaling workflow processes as well as software.

3.2.2 Entry Barrier

The entry barrier is measured by the amount of customization, administration, and installation costs required before some benefits can be derived from putting the workflow system into use. This involves domain customization where the objects, components, and views of the system are tailored to the specific target domain, such as healthcare, electronic commerce, finance, manufacturing, software development, or others. Each domain should share the core infrastructure of the workflow system but include one or more application layers built on top. By evolving the workflow infrastructure to accommodate shared capabilities across domains or even concerns within a domain, the amount of customization required for changes and adoption becomes less. New uses of the technology can be targeted by adding a thin application layer and reusing activity, artifact, and resource definitions as well as workflow processes if applicable. Reuse can be facilitated by providing extendable and customizable user interface and workflow components in an online marketplace setting.

Administration and installation costs also raise the entry barrier. Deployment costs of installing minimum and consistent process tools and technologies on every relevant desktop is problematic. Add to this the fact that stakeholders may require diverse machines, protocols, and implementation languages all of which may lack cross-platform coordination and standard use capabilities, the cost of deployment begins to outweigh the perceived benefits. In addition to the workflow execution infrastructure, the workflows themselves need to minimize the entry barrier to their use. Both of these can be addressed by a lowest common denominator approach. The WWW follows this approach because of its ubiquity, and end-users leverage their training and experience with hypermedia and hyperlinks. This also makes it possible for end-users to participate in a process with no explicit installation. Adding a mobile applet or scripting execution language (similar to Java, Python, Perl, or Tcl) and supporting text-based representation with an explicit, well-defined workflow language allows workflows to be deployed, invoked, and monitored using standard network protocols such as HTTP (HyperText Transfer Protocol **[27][58]**) or SMTP (Simple Mail Transfer Protocol).

3.2.2.1 Inform/Automate

Workflow systems provide information to the participants needed to accomplish a task but may also automate these tasks based on available information. Workflow processes intended for execution are most easily deployed initially as inform only. This allows coordination and communication between end-users but leaves the execution of the activities to the discretion of the individuals. Automation capabilities can then be incrementally added by addressing the highest payoff areas first. Inform tasks may include flexible access, searching, sorting, prioritizing, gathering, editing, and so on. Automation tasks may include collation, routing, synthesis, format conversion, notification, and update among other things. Intermixing the information and automation aspects of a workflow system is crucial to its success. The automation tools must be able to recognize the results of the participant's work such as activity and completion information. Likewise, participants need to be able to easily utilize the automation capabilities of the workflow system.

3.2.3 Work/Reward Matching

Workflow may not benefit those individuals who directly use it. It is the group they are part of that benefits through improved coordination and communication. The degree of which end-users benefit from using the system in balance to the amount of work required to maintain consistent data and state for a process is called work/reward matching. Care should be taken to ensure that mismatches are not encouraged in the system. Individuals who have the ability to formulate processes to automate their own tasks match their work with the benefits gained from it. Groups require matching also. Because group benefits are shared among the members, it is important not to 'saddle' individuals with the majority of the work. Similarly, shared work that benefits only a single or small number of individuals within a group may inhibit adoption as end-users may be unwilling to put the time or effort into using the tool if the benefits are perceived to be only accomplishing the work and goals of the few. The best approach to work/reward matching is to distribute the work and the benefits evenly among the members of a workgroup. This means that the workflow system may require development of processes by non-technical as well as technical end-users. Further, the cost to deploy and execute a process should be commensurate with the eventual benefits from that process. A simple, commonly used workflow should require minimal skills and effort to customize, deploy, and execute.

3.2.4 Maintenance

Processes as well as the means to execute them change and evolve over time. The ability to incrementally change a process regardless of whether it is executing or deployed is a maintenance task. Maintenance involves versioning and configuration management for both the workflow process and the state it keeps. The difficulty encountered to evolve, distribute, and deploy the process and state over time while making these changes should be minimized. Tools to aid in configuration and customization of the workflow system are the easiest way to reduce maintenance tasks. Also, workflow processes should be built with the assumption that they will change over time. By building in a maintenance process into the workflow system, the effort to checkpoint, upgrade, evolve, and even recall a process is reduced.

3.2.5 Adoptability

Adoptability targets how well a workflow system integrates in with existing work culture and technical infrastructure. Work cultures that have had very little exposure to automated technologies may lack the sophistication to immediately adopt a workflow system. Training is an inhibitor, but also trust. Workers first of all need to recognize the utility of the system and then trust it enough to delegate or take ownership of tasks usually performed by hand. As the reliability and usage of a workflow system increases in an organization, the sphere of its applicability expands. Workflow systems should support a continuum of tasks from the simple to the complex in addition to seamlessly integrate activities performed by both humans and computers.

Technically, the workflow system is more easily adopted by using the tools and infrastructure that is in place. For example, systems that require real-time distribution of data and immediate feedback of results may be difficult to adopt in an organization where significant numbers of systems are intermittently used, disconnected from the main network services, or require special tools or software. Eventually as the technology becomes integral to the organization, technology infrastructure can be evolved to include new requirements, but again, the lowest common denominator approach to infrastructure ensures adoptability.

3.2.6 Granularity

Granularity of both the workflow system and the workflow process effects adoption. Large-grained system components may encompass related groups of shared functionality such as persistence or distribution into a server thus removing the need for duplicate infrastructure in the clients. However, packaging functions in this way may reduce the applicability of the system by adding in unneeded services, fixing the user model, or advocating execution, security, or distribution policies that may be difficult to reconcile with the work culture. Fine-grained and lightweight, interoperable components increase the flexibility of the system with respect to incremental adoption. Small, targeted sets of components allow evolution of the system to address new priorities and goals of the organization. Fine-grained components offer small, efficient modules of functionality, and are more easily exchanged or upgraded with less impact on other parts of the system. Maintenance of the workflow system is also easier for both the end-users and workflow developers through a series of small, incremental changes as it reduces the complexity. Often evolution of a system by introducing new components will result in inconsistency of either the data or tools requiring some amount of rework. By using fine-grained components, it becomes easier to isolate and address the impact of these changes. Workflow primitives should be fine-grained also. While these primitives can

be encapsulated or abstracted into larger grained processes, the activities, artifacts, and resources should be interchangeable allowing for process reuse.

3.3 Object-Oriented Workflows

Workflow objects should be defined in an object-oriented manner. The basic unit of the workflow should be object definitions ideally abstracted from real-world structures. This provides a clean, conceptual one-to-one mapping between the real-world objects and the model objects maintained within the system. Object-orientation, by using information hiding and abstraction, makes the workflow easier to understand by end-users and maintain by developers. This approach allows for a consistent work model across human executed and computer executed activities even as the definition and understanding of these tasks evolve over time.

Object-oriented workflows in addition to encouraging consistency, facilitate the clean definition of interfaces enhancing the ability of the activities, artifacts, and resources to be reused. Designing and implementing a workflow in this manner focuses on the definition of the data and the interfaces needed to manipulate it. This data is combined with the behavior specifying how to create or transform particular structures and values of the data to form a workflow object. Independent manipulations of both the data and behaviors facilitate reusing object definitions for various ends using object-oriented mechanism such as inheritance, polymorphism, concurrency, and encapsulation. Object-oriented workflow design makes it is easier integrate tools to manipulate the objects while hiding their implementation and details. This allows the workflow to be built in incremental stages focussing first on the critical parts of the workflow's activities and encourages alternate or evolving implementations. Clean definition of interfaces also allows a control point for access. Different roles in an organization may restrict access to sets of artifacts such as documents or data. In addition, the activities to create, define, and revise the workflow artifacts can be limited to those with the appropriate permissions or security.

3.3.1 Process and Development Tools

Workflow systems should have available to various stakeholders a multitude of tools to help manage the design, development, deployment, and execution of a workflow process. These tools should allow the stakeholder to manipulate the conceptual model of the workflow at all stages. Workflow development tools should mimic the capabilities of software development with the exception that the target language is a process modeling language. Implementation of the workflow process also should easily leverage software development and integration technologies. Testing and simulation tools should be available to the process architect for pre-deployment execution or simulation tools to verify the process with respect to its anticipated target environment and end-user's requirements. Execution tools and technologies should aid in managing changes while the process is running or to provide feedback to help guide in its execution and evolution. Finally, the workflow system should fit seamlessly into the end-user's environment, being invisible when required. Support for the tools should be flexible and configurable by the end-user, but also provide basic services for the most common tasks across all stakeholder's day-to-day work.

Various technologies recently adopted in the software field are easily re-targeted to the workflow domain. Visual programming is a promising technology in the development

and design of workflow processes because it allows non-programmers to manipulate a program model at a high conceptual level. Distributed execution and resource levelling of program execution also map well to the workflow domain because of the inherent attributes of individuals to work independently and concurrently. Scripting and interpreters allow incremental execution and interactive changes. Parsing and generation tools are also helpful for managing a process's implementation allowing round trip changes from implementation code to design or vice-versa. Both workflows and their implementations may easily benefit from component reuse libraries, process templates, and rapid prototyping and design tools and environments.

Workflow processes can not always be tested before being deployed or executed because of the complexity of the process, the necessity of human interaction, participation of human executed activities in differing roles, and the prohibitive cost and effort to build and maintain separate execution contexts for testing and simulation. To aid the process developer in delivering a more effective process, tools for process testing and analysis such as syntax checking, workflow path coverage, exception handling, and even correctness of the workflow are necessary. These tools ensure that the design of the workflow is sound, and they should complement the traditional software testing and analysis of the workflow's implementation. Bottlenecks in the workflow should be flagged as well as issuing warnings for incomplete or missing items. Control, data, and resource flow values should be verified for existence, definition, and use. Also, standard project management analysis tools should be used to check the feasibility of the process with respect to time and budget constraints

Once a workflow is deployed, execution tools should be available to the process and system programmers to make changes to the running process. Execution tools may include remote or peer interpreters in order to inspect or change the state of objects, agents for capturing evolving state and process interactions, version control of processes and workspaces, printing, searching, browsing, and on-the-fly tool management and integration. These services should be used together for evaluation of the process to measure the performance and provide for process optimization. Also important is the ability to debug a process before, during, and after execution. When a workflow exception occurs, the system and process programmers need to have access to the execution context at the time it was triggered. Rollback of process state and playback of sequences of events aid in diagnosing what went wrong in the process.

In addition, end-users should have their own set of tools and services. End-user workspace customization allows conceptual rearranging of tasks to fit the work context. Also, traditional tools shared across workflow domains such as email, planning, scheduling, calendaring, event notification, and interest registration should all be made available to the end-user. and presented in a way that is easy to apply to all aspects of their work.

3.3.2 Semantics

Workflow descriptions should be based on a semantically rich modeling language. This language should be precise, consistent, expressive, and extensible.**[132]** The language should be expressible enough to represent steps, loops, branches, conditionals, routing, assignment, timing, scheduling, and constraints. The semantics of the system should also allow definition of control, data, and resource flow as well as definition of policies on how to handle interruptions to them. Overall, the modeling language should be able to completely and accurately describe non-trivial workflow processes and support refin-

ing those definitions to be as precise as the work requires over time. While some workflow systems allow processes to be designed using a visual modeling language, it is important that there is a strong semantic description underneath the graphical representation. This is necessary because while graphical views of the process are easier to understand, they do not always convey the precision and details of a complex process which may cause ambiguity. Confusion between process stakeholders that may result from the differing interpretations of the graphical representation creates discrepancy between what is intended and what is actually done. The underlying semantic model should be sufficiently rich in its workflow description to allow queries, either programmatically or human-queried, to resolve these type of interpretation problems.

3.3.3 Meta-Process

Meta-processes are the means and mechanisms which govern how a workflow process can change. This change can result from several directions including insertion of new technology into an organization, feedback from process participants, or competitive market pressures to name a few. Meta-processes are processes themselves and may describe the steps and constraints of how to develop, evolve, upgrade, install, uninstall, distribute, deploy, integrate tools into, or customize a process.**[13][16][54][156]** The end product of a meta-process is an executable workflow process model for some specific domain or work context. Meta-processes are subject to the same requirements as the processes that are modeled with the workflow system, i.e. the meta-processes should also be decoupled from the workflow system. This allows meta-processes to evolve over time similar to domain specific workflow processes.

3.3.4 Ad Hoc

Ad hoc workflow systems stem out of the CSCW field. In this type of system, users are allowed to perform their work in an ‘ad hoc’ way where solutions and goals are accomplished with whatever tools and data are immediately available, usually from an end-user’s browser or desktop. The primary purpose of ad hoc systems is to inform users of the current problems, issues, and data needed to accomplish their particular tasks. This approach is appealing because it mimics the fast, furious, and unstructured pace of some work activities. While ad hoc workflows provide minimal guidance and automation support they are useful because the technology doesn’t inhibit real work from being done. A workflow system should allow tasks to progress at their natural pace while maintaining a model of the work being performed. It is also important to carefully keep track of the status of ad hoc workflows because data and control dependencies as well as assignments are often not immediately viewable within an ad hoc system. The system should allow for dissemination of information to the widest possible boundaries of the organization to encourage ad hoc tasks, but at the same time take care to ensure that proprietary information and products are kept proprietary. Clear and concise interfaces to workflow tasks eases the access management problem.

3.3.5 Simulation

Similar to how code walkthroughs are used to review pre-execution behavior of software systems to ensure quality, the same benefits are gained from performing process walkthroughs. This may involve mock-ups, hand execution of the workflow process, or in cases where the specification is too large or complex, simulation of the functioning system. Simulation allows the process stakeholders to perform ‘what if’ scenarios to ensure chances that undesirable side-effects after the workflow is deployed are kept to a minimum.**[80]** Simulation capabilities of the workflow system should allow seamless

integration with the actual execution environment merely by switching execution contexts. Simulation should support multiple executions across a data input set in addition to playback, capture, and rewind. 'What if' or 'Is this possible with these constraints' are questions that should be answerable by simulation results. Finally, any simulation should include extensive tracking, audit, analysis, and evaluation capabilities to help determine if one set of outcomes or changes is more desirable than another.

3.3.6 Roles

Roles define the expected behavior of a process participant as determined by their responsibilities. Roles at a minimum should control access to data through a user-model that includes login and password permissions. Ability to read, write, change, or update data or even processes should be role-specific. In addition, execution of tools and processes should be reserved for clearly defined roles. Roles should be flexible enough to have a many-to-many mapping to process participants. This allows role sharing, delegation, intersection, subjugation, and re-assignment even during the execution of a workflow process. At their core, roles should be people-centric and present the particular participant with the data and tools necessary to accomplish their work from the perspective of their role. Definition of roles along multiple dimensions e.g. technical versus non-technical, allows role definitions to share organizational aspects. Questions of 'who owns/designs/implements/executes the process?' and 'who are the people involved?' should be easy to answer and define.

3.3.7 Multi-Tiered Architecture

Most current workflow systems assume that its software architecture is static in the sense that it does not evolve during execution. In addition, many systems also impose the restriction that the architecture of a workflow system does not change after the workflow is deployed into its execution environment and any changes require re-deployment of the updated workflow and infrastructure. Implementation of the workflow system in a hierarchical manner aids in making the system flexible and extensible by minimizing the amount of software infrastructure that needs to change.**[33]** A good approach is to abstract at the core workflow functions most closely tied to network and operating system services and then add a series of thin architectural layers, each incrementally closer in capability to the workflow domain to which it will be deployed. Each architectural layer should have limited visibility, i.e. subsequent layers of the architecture should act as virtual machines enforcing a clear set of APIs to its services at a common level of abstraction. This multi-tiered approach allows minor changes in the configuration of the workflow system architecture to support major policy changes with respect to distribution, ownership, or presentation of processes and data. In addition, impact caused by changes to the workflow system even while running can be minimized with a layered architecture approach.

3.3.8 Security and Access Control

The problem of security and access control is receiving a lot of attention with the popularization of networked software including the Internet and the WWW. The purpose of these mechanisms is to provide assurances to participants of the confidentiality, integrity, and authentication of the users, services, and data involved in a workflow process. Most security problems aren't inherent to the workflow protocols or infrastructure itself, but with the policies surrounding its use. Security and access control mechanisms in a workflow system must accommodate variations in site policies including those due to external restrictions. This may include tradeoffs between performance, usability, and

security. Workflow system designers should find a balance between these requirements based on the sensitivity of the data and participants. Also, the level of security should be easily configurable in the face of changing security and access control requirements. Major policy changes such as relaxing restrictions to previously sensitive data or tightening restrictions based on inappropriate access as well the minor change of an employee switching job responsibilities should all be easily configurable with the workflow system or on a process-by-process basis, even after deployment.

System programmers have multiple mechanisms for addressing security and access control. Tunneling involves encrypting and decrypting the communication stream between two secure points which is useful for process participants at remote sites participating in a workflow process through the open internet. Digital signatures can be used to ensure that the data or messages being sent aren't intentionally or accidentally corrupted guaranteeing the integrity at both ends of communication. Digital watermarks can guarantee uniqueness in order to better control artifacts. Signed applets are small pieces of executable software that can be configured to have certain permissions for changing or updating local data and configurations. Certificates may ensure the integrity of the software preventing unwanted access to data and can be controlled by the group or organization internally or externally to verify participating users or agents. A sandbox approach limits its access to certain hosts, domains, times to data and services, but within these limits users and agents are allowed a wide latitude for access to information. Quarantine procedures allow participants to use new software or perform new actions after a period of time and trial should be based on an explicit level of trust and should be incorporated into the workflow evolution meta-process. Similar to how the software and architecture of the workflow system can be configurable with respect to security and access control issues, users and their actions can also be managed in this way using these authentication schemes. This information can be used simply for auditing and tracking or to actually protect the integrity of the information. Firewalls and proxies can filter actions by users or prevent the dissemination of restricted or sensitive information. System programmers should be able to incrementally add or remove security mechanisms as the context dictates.

3.3.9 Fault Tolerance

A workflow system is considered fault tolerant if minimal interruptions to work occur in the face of inconsistent or corrupted data and missing tools or resources. The strategy for ensuring the appropriate level of fault tolerance in a workflow system is a function of the consistency and coherence of the data. Workflow process execution takes a bipolar approach to managing data. One technique tightly constrains the actions that can be performed by a process stakeholder or software agent but guarantees the internal data consistency of the workflow model and execution data. The other allows stakeholders to perform arbitrary actions to accomplishing their activities but places minimal constraints on the structure of the intermediate and final data artifacts. Fault tolerance requirements are very different for each system. In the first, data consistency is guaranteed because the actions that a process stakeholder can perform are limited to only those preserving consistent data transformations. In these types of systems, transactions are supported and allow rollback of workflow processes to previous states. This happens most when the workflow model becomes out of sync with the real activities taking place, oftentimes the result of an unanticipated or unpredictable event. In the second approach, stakeholders may perform arbitrary actions toward accomplishing their activities, but minimal constraints are enforced on the structure of the intermediate and final

data artifacts. The ad hoc nature of this approach may cause the data and artifacts being created and shared to be inconsistent. This inconsistency can be managed automatically by a constraint system, and failing that, a process participant. All automatic resolutions should be made explicit so that the process participants don't get side-swiped during midstream execution by the system dumping an irreconcilable set of data constraints. The key being to allow the workflow system and its participants to manage inconsistent as well as consistent working data to minimize the amount of effort that is needed to get the workflow back on track due to unpredictable occurrences.

3.3.10 Configuration Management

Configuration management should be pervasive, but decoupled, in a workflow system.**[54]** Not only do the workflow processes themselves need to be versioned and kept under configuration management, the intermediate and final data artifacts such as documents, data, process execution state, and the system architecture should be. Any component of the system that is relevant to the advancement of the workflow that will change over time should support checkpointing and versioning. Collections of artifacts, including the workflow system's architectural components, workspace views, and execution context information should be managed similarly to individual components. Depending upon how volatile the workflow process is, checkpoints can be made at timed intervals or at the point of significant events. For instance, a document may change versions on write events or on the handoff of ownership. Likewise, in a software development process, the source code could be checkpointed at the end of each working day to allow an automated nightly build. Recovery capabilities should be complemented with auditing and access control so that progress can be traced or context can be recreated, if necessary. This information is also useful for simulation and analysis leading to process improvement. An important aspect of a complex, distributed, multi-person workflow process is figuring out what happened as well as keeping track of what is currently happening. Inconsistent configurations and versions should be detectable by a constraint system. The ability to store and access previous activities and artifacts, retrieve their history, or annotate revisions with comments about the changes are all important features. In addition, storing this information using a full-featured configuration management system allows comparisons to be made between versions. This is especially useful when tracking and comparing workflow changes over time. Because process execution and evolution are seldom monotonic, participants may require certain versions of activities to be performed along with the relevant tool configurations and architectural components. As with other policies, the workflow system should be decoupled from and not imbed an implicit configuration management and versioning policy in order to allow for flexible changes.

3.3.11 Transactions

Transactions allow the workflow system to manage change during the execution of a process following an enforced set of policies governing that change. A transaction in a workflow system should be able to lock a collection of resources in order to atomically complete an activity.**[9][91]** Complementary to configuration management, transactions should provide a flexible set of change policies that can be dynamically enforced during process execution. Transactions may include timing constraints such as an action being taken as the result of a timeout of service or data constraints such as 'insufficient information' or 'result not produced yet' warning message. In either case, transactions ensure the step-by-step integrity of the workflow execution state by completing actions as required or rolling back the workflow system to a meaningful state if the action can't

be accomplished without violating data constraint or consistency principles. Transactions may include privileges such as *grant* and *revoke* as well as the procedural or declarative knowledge of how and when to commit a valid transaction. Transactions may also describe *triggers* which precipitate some additional action on the successful completion of another transaction.

3.3.12 Automation

Workflow automation may include automated handoff and forwarding of documents, report generation, software installation, notification of significant events, tool invocation, configuration of an end-user's workspace, and most importantly the execution and completion of tasks independent of the workflow participants. The extent to which these activities can be accomplished without human intervention is the level of automation that the system supports. A workflow system's automation infrastructure should include configurable support for both human as well as non-human involved participation such as intelligent agents. Scripting, including capture and playback of user interface manipulation, aids in the transition from human executed tool interaction to machine controlled. Scheduling can be used to initiate and carry out daily, weekly, or recurring tasks. The more a workflow process is executed, the more familiar it becomes. Processes that have become fixed, tedious, common, or procedural are excellent candidates for automation. The workflow system should allow an incremental path from human-executed activities to computer-executed. Automation may further the execution of a process proactively or reactively by executing process fragments or sending out information to fillip both work and participants and respond to changes, completions, or postings by them.

3.3.13 Concurrency

Concurrency should occur at both the architectural level as well as the process description and presentation level. Concurrency at the architectural level allows components to execute or change state independent of the other parts of the system.**[18]** This may allow a greater degree of overall system efficiency, increased scalability and evolution, and a greater degree of dynamic change. A high degree of concurrency in a process system's architecture should be matched with a lightweight, multi-threaded component model. This allows small, incremental changes to take place with minimal change impact on the rest of the system, even during runtime. In addition, concurrent components allow the overall system to be more easily customized by instituting different control and execution policies governing the scheduling, behavior, and constraints of the components.

Many of the same benefits from software and hardware concurrency and parallelization are applicable to scheduling human-executed activities where concurrent events are commonplace. Day to day work often contains concepts such as lookahead, simultaneity, data parallelism, resource sharing, overlapping, multiplicity, replication, time sharing, resource sharing, and distributed execution at multiple levels. Because of these concepts, online workflow system have the potential to realize efficiency benefits that aren't easily effected in an offline setting. When modeling a workflow process concurrency should be explicitly presented in the majority of cases. When confronted with a seemingly serial list of activities, process executors may simply assume that no concurrency is possible which may reduce the efficiency of the process. In a large and complex process, potential concurrences and serializations might not be pursued without explicit guidance. There are many factors effecting the combination of human and computer executed activities in a workflow system. Multitasking or multithreading capabilities of both human and software agents should be exploited in a process execution environment

as long as it doesn't increase the complexity of the task or the overhead of communication and coordination. The independence and interdependence of the activities at hand is a good measure for determining whether activities should take place in parallel. Without an explicit presentation of concurrency, it is difficult to determine the efficiency of the process.

3.3.14 Abstraction/Escalation

The ability to employ simplifying assumptions and abstractions in the modeling, scheduling, and execution of workflow processes is necessary to allow it to scale to a large, complex series of activities involving the coordination of many people over a long period of time. Not all of the same issues will apply to the modeling and execution of a process across all levels of abstraction because of organizational and multiple-stakeholder issues. In a workplace setting, abstraction centers around common process goals, and escalation follows a corporate chain of command. Because of these issues, it may be necessary to compartmentalize aspects of a process through information hiding and abstraction. As an activity or process progresses, the prioritization of the tasks evolves. This evolution may shift the encapsulation boundaries depending on how the changes are handled. Anatomization of specific tasks should support delegation, rerouting, or reassignment in addition to prismatic separation of tasks across social boundaries. Conversely, combining several tasks into a larger workflow with possibly competing goals at different levels of abstraction should be done with the strictest of encapsulation boundaries to aid in the success of its execution. Technologies such as hyperlinks and mobile process or agent technologies can aid in the scoping and context of a workflow that is specified across multiple levels of abstraction.

3.4 External Tool Integration

Tool integration is crucial to the execution of a workflow process. Workflow systems should specialize in delivering the right information to the right person at the right time. Once a person has the data artifacts at hand, the tools to produce, consume, manipulate, translate, annotate, or augment should be easily within reach as well. While the goal of a workflow process is to appear seamlessly integrated with the end-user's desktop and tools, the implementation to do so may not be easily accomplished. Tool and workflow integration may require multi-directional communication where events or messages can be initiated by the user, tool, or workflow system. Multi-synchronous communication may be required in external tool integration by allowing notification and synchronization to be additive and independent of the tools, agents, and participants involved. Not all tools were designed to be integrated in this way because of constraints of the implementation language or platform. Because of this, tool communication should be supported by multiple protocols, across platforms, and in multiple languages if possible.**[12][20][57][92][98][111][157]**

Because workflow systems inherently support communication and collaboration, it follows that integration with communication and collaboration tools such as online conferencing, email, shared workspaces, synchronized browsing, collaborative authoring, calendaring and other computer supported cooperative work approaches should be easily accessible from within a workflow process. Going the other way, capturing and measuring how participants use desktop tools to accomplish work may form a basis for improving the efficiency of workflow processes or discovering new ones. Instrumenting on- and offline work with metrics and measurement tools to capture interesting events

helps accomplish this. Also, maintaining a strong distinction between the processes that participants follow and the data they create allows the greatest flexibility in the evolution of them over time.

3.4.1 Multi-Directional

Inter-tool communication needs to be multi-directional. In order to keep its internal workflow model consistent, a workflow system needs to be notified of work initiated or completed by external tools. Also, collections of tools may require notification and forwarding of appropriate artifacts at the correct points in the workflow process in order to control their execution and use by participants. This notification can occur serially or in parallel. Serial communication usually occurs in a point-to-point manner. Messages are sent from a sender to receiver, sometimes in a fixed interval. Serial connections should have the ability to be initiated by either the external tool or the workflow system. While serial communication with external tools helps guarantee the data or workflow model consistency by enforcing a partial ordering on the communication, complex workflows involving integration of multiple tools and coordination of many people across several levels of abstraction or organizational boundaries may depend on parallel communication. A component, tool, or person may be acting as both sender and receiver at several points in time during a workflow execution. In addition, multicasting of data and artifacts and multi-tuning based on interest along different data distribution channels whether by the workflow system or the external tools allows for a more efficient distribution of information. Because it allows greater scalability and evolution, this is an elegant way to model communication of data and artifacts among a set of cooperating people and agents, but it requires a higher degree of complexity in the infrastructure to handle contention and fairness issues.

3.4.2 Multi-Synchronous

Execution by external tools and execution of a process by the workflow system may occur independently of each other. The workflow engine may suspend execution pending completion of work or handoff of artifacts from external tools in a synchronous fashion, or it can continue downstream effecting the handoff and control of the process as work is completed asynchronously. Process fragments that have inter-dependencies such as artifact handoff or resource contention should be specifiable with synchronization and serialization primitives in the workflow modeling language. Otherwise, independent fragments should be able to be scheduled and executed independently and in parallel when possible. Asynchronous execution should not be overconstrained. This allows work to continue without unnecessary dependencies. Also, asynchronous execution between the workflow system, external tools, or even process fragments allows greater flexibility in reusing and scheduling processes because different models of execution can be used in completing the processes. The workflow system should be able to support both synchronous and asynchronous coordination of workflow processes.

3.4.3 Multiple Languages

Workflow process support environments must integrate and interoperate with diverse tools, platforms, and environments. Flexible mechanisms such as integration with multiple languages provide the ability to evolve the workflow system in concert with its deployment environment and changing work context.**[120]** The ideal workflow system affords workers transparent integration with the appropriate external systems as needed, desired, or dictated by the work culture. Workflow activities may need to integrate with external systems either through application programmatic interfaces (APIs) or through

intermediate forms, depending upon the “openness” of the system. Because of the diversity of work performed and the stakeholders that may be involved, system integrations need to occur in the language best suited to accomplish the integration. To handle this, multiple scripting and programming languages can be used to coordinate between the workflow system and external tools as long as calls to the workflow system and the external objects, tools, and services can both be accommodated. In addition, intermediate forms that embody the process or integration description are useful for integrating tools across implementation languages. A helpful strategy for this is to allow easily embedable components or applets for parsing the exchange formats such as plain text, XML, or other interchange formats.

3.4.4 Cross Platform

Work contexts and environments not only involve diverse stakeholders and tools, they include multiple computing platforms. A flexible workflow system in order to successfully deploy and execute a process may need process fragments and software architectural components to run across multiple operating systems and computing platforms due to the availability of resources and tools. Cooperation, control, and coordination in a workflow process do not always happen within the sphere of a standardized or shared technology infrastructure. Processes that lack the ability to run across platforms risk inhibiting the effectiveness of putting the right information at the right place in a timely manner. Processes that go through a technical disconnect because of cross-platform issues risk derailing the efficient passing of artifacts by placing an extra burden of format translation or tool iconsistency resolution on participants which is tangental to the flow of work. Further, control and coordination policies may be limited by technology issues rather than the workflow needs of the stakeholders.

3.4.5 Metrics and Measurement

Metrics and measurement provide a foundation for the workflow infrastructure for better management and feedback. Both runtime and static metrics are useful for determining the efficiency and effectiveness of a workflow process. Collection tools can be used to discover new processes or provide qualitative measurements for process comparison and optimization. Depending on the degree of automation, collecting and analyzing metrics may allow the basis for self-correction or self-optimization dependent upon the amount of dynamic change and reflexivity the workflow system supports. Metrics and measurement tools can be used to accomplish automated tracking and monitoring to ensure the smooth execution of large and complex processes which would otherwise be difficult for a human to monitor. Metrics also provide a basis for model-checking to determine consistency and completeness as well as evaluating the throughput or change impact either through actual deployment or simulation. Finally, metrics and measurement infrastructure aids in accountability by maintaining information on audit trails or access logs as well as versioning and recovery information.

3.4.6 Process Data and Programs

An explicit manipulatable process model is crucial to providing an accurate representation of work, determining status, and enforcing control policies. Separation of the workflow process model from the data such as artifact and resource models allows evolutionary changes to the process independent of the products being worked on. Likewise, the data can be changed with minimal impact on the process’s control, resource, and data flow specifications. By maintaining a clear abstraction between process data and programs, reuse and customization are facilitated.**[88][89]** Depending on the needs

of the process and the sophistication of the workflow infrastructure, separation of the process model's behaviors from its state can allow dynamic customization, configuration, and contextualization to more easily fit the work environment in which the process is being deployed.

3.4.7 Communication and Collaboration Tools

Execution of a workflow process involving many people inherently requires managing the association of individuals with varying degrees of contact and interactions in these groups. Activities performed automatically or by the process participants occur over both time and space as well as synchronously and asynchronously. Completion of these activities often require both formal and informal communication channels beyond the scope of the workflow process specification. These channels of dialog can be encouraged via the integration of communication and collaboration tools. Integration with calendaring, scheduling, shared bulletin boards, and email provide a strong basis for maintaining the workflow history and managing its completion. Computer supported cooperative work tools such as chat tools, distributed authoring and versioning, shared editing, event notification, or messaging and registration lower the barrier for participants to reach agreement on issues or to quickly communicate rationale and reasoning. Also, allowing participants the ability to address outstanding issues with communication and collaboration tools aids in averting and resolving potentially work stopping issues. The integration of these tools into a workflow environment works both ways. As an example, being able to embed an executable workflow process into an email message, a WWW page, or a bulletin board thread provides a powerful mechanism for actively guiding a participant to an activity's completion, possibly avoiding unnecessary work. The addition of a structured work description that a workflow system can provide allows a clearer specification of the work to be completed and individual responsibilities. This removes some of the risk of overlapping or working towards cross-purposes.

3.4.8 Events

Events involve asynchronous notification of possibly distributed components at some instant of time. Events can be broadcast to interested parties based on filtering and registration. Workflow systems built on an event-based model are flexible and reconfigurable because they allow exchange of components and re-routing of events to alternate destinations possibly over the course of the workflow execution. Workflow systems based on an event broadcast model assume a certain style of integration and interoperation. Work environment tools and process fragments can be loosely coupled allowing coordination between heterogeneous components using this approach.**[22][23][71][154]** Also, event-based integration encourages cooperation between stakeholders as events are lightweight and notification takes place asynchronously allowing work to coincide more frequently. However, the workflow system and process execution should be designed such that process participants are not overwhelmed or bothered by unnecessary information. Software and process components as well as people should be able to register their level of interest or latency of response to filter events appropriately. Event notification can be implemented using push technologies that send events as they are generated or a pull model where a participant is required to actively seek the information. Both the push and pull models can also be scheduled to send or receive events at regular intervals. Specification for handling publish/subscribe and local/global events should also be definable within the workflow system. It is important for a workflow system to support dynamic change of notification and broadcast policies, especially with respect to interest and filtering. Finally, because workflow process execution may occur across many

platforms using multiple technologies, multi-protocol support aids in its adoption and usage. Support for email, WWW, and various Internet and messaging protocols allows events to be broadcast over existing network infrastructures. Integration of paging, phone, radio, wireless, and other communication messaging tools to send events is helpful for capturing offline work. Multi-protocol support for events increases the ability to track, trace, observe, and capture significant occurrences in a workflow process.

3.5 Customization and Reuse

Customization and reuse amortizes the cost of development and deployment of a workflow process over time and changing work contexts. One of the canons for reuse in software is that if a software application is useful, it will be used in other work contexts. The same principle can be applied to user interface components, object libraries, and even workflow processes. Workflow processes and their execution infrastructure should be flexible enough to be re-targeted to new work environments and easily reused with different hardware, software, and social constraints. Because workflow processes and their execution differ so greatly between organizations and groups because of technical and social factors, adoption into a new context may require some ‘glue code’. Programmability of the system is required to allow better matching between the ‘off the shelf’ process and the expectations of the stakeholders. Also, because a priori knowledge of a workflow’s evolution requires prescience beyond our current predictability and understanding of most processes, the workflow system and the process which includes the views and the execution model should both be extensible.**[14][17][68][118][143][147][161][164]**

Because control and coordination policies vastly differ among groups, support versus dictate issues should be configurable and consistent with the work culture. Work policies should be customizable to allow the workflow system to adapt to the group or organization rather than the other way around. User interactions with the system, tool integration, and environment should also be customizable. Management of data, artifacts, resources, and processes should include sorting and searching to allow users to find the best solution match for the problem at hand. This may include partial solutions where stakeholders only use the fragment of the workflow that is relevant to their task.

3.5.1 Flexibility

Flexibility of a workflow system is evident in the way workflow processes are presented, used, executed in the face of changing real world activities. A workflow system should support dynamic creation, deletion, and update of views into the process including different views for different stakeholders based on their interest and expertise. The workflow modeling and execution infrastructure should be flexible enough that graphical representations can be exchanged or even removed without interrupting the underlying process model. The system will need to adapt to the target environment rather than the other way around. Flexibility in specifying the execution model as well as the initiation or completion policies should be easily changeable within the system. Tool availabilities and integrations should also be configurable to allow on-the-fly or on-demand access in addition to pre-specified tool sets. Resource notification should also be flexible allowing email, scheduling including time constraints, feedback, announcement, and agenda management for people and other process resources. Artifact routing and synchronization should support a wide range of policies, even during the course of a workflow execution to allow for alternate processes and workarounds to occur. Support for

concurrent access to data as well as concurrent execution of workflow process fragments allows greater flexibility in sharing artifacts and encouraging cooperation between steps of a process. As with any best laid plans, exceptions will always occur. The workflow system should allow for multiple paths to completion and exceptions such that unforeseen obstacles and limitations can be easily hurdled without invalidating the workflow process model or its planned execution.

3.5.2 Programmability

Workflow processes should be programmable in multiple ways. A workflow system should allow workflows to be programmed both visually and textually depending on the abstractions and level of expertise of the user. Visual workflow programming languages allow the process programmer or end-user to more easily view complex interactions and arrive at an acceptable workflow specification because object interrelationships or transformations are sketched out and concrete. Despite the advantages of visual programmability of workflow processes, they sometimes lack the power and expressibility to succinctly describe a complex series of actions and require textual codification. Open APIs for each architectural component of the system allow for greater flexibility, customization, and reuse by supporting access to only the data and functionality that is relevant to the work context at hand. These APIs should be supported at multiple levels of abstraction to allow both fine-grained customizations as well as high level changes. Support for an open API across multiple languages also improves coordination of diverse tools and components. Support for all of these approaches aids in the reuse of work when adopting, specifying and executing off-the-shelf processes.

3.5.3 Extensibility

Successful workflow processes, like successful software, will need to be extensible to fit into a new usage context or environment. This new context may be a use of the workflow process that the original process designer hadn't anticipated, such as a new domain. This may include embracing new domain models, enabling domain specific presentations, subclassing and overriding data fields and methods possibly even during the execution of the process. The process model should allow functionality to be incrementally added in such as new process execution primitives, alternate user interaction models, and the set of desktop tools and components used in the completion of process tasks. A strong separation of concerns, possibly through event-based integration, allows minimal impact on pre-existing components when extending the system. Clear architectural layers with open APIs at each layer aids in reducing process rework from extending existing process objects and fragments. The workflow infrastructure should be extensible also. Providing support in the infrastructure to dynamically change and integrate new components even after deployment helps ensure the usefulness of the system in the face of changing work requirements.

3.5.4 Support/Dictate

A workflow system must address the tradeoff between supporting a user activity versus dictating the activity to them. The question of how closely a process must be followed and in what order the steps must be taken is largely a secondary effect of the work culture in which the process is being executed. The less leeway a worker has to improvise because of quality assurance, work standards, or artifact dependencies the more the workflow system dictates the worker's activities. Processes that require strict adherence should support dictating the activities, however, over-enforcement of a process may result in an unsolvable path to a solution. At these times the process participant will go

outside of the system to overcome the limitations of the workflow infrastructure which may have been caused by insufficient customization to the problem at hand or an exception which the system wasn't able to recognize or handle. Each work environment has its own requirements for accomplishing workflow processes, and the workflow system should be flexible enough to allow it to adapt to the changing needs of the organization. Historically, workflow and process systems have suffered from over-constraining the work being described, so care should be taken to err on the side of support when possible.

3.5.5 Customization

Customization of the workflow system is key to its usefulness when adopting processes in a new work environment by reducing startup costs and increasing the applicability. Customization may need to take place across all levels of the system including execution and notification policies, object locations, and even persistency and update policies. Customizations should be separated into different levels by abstracting away or hiding the technical details for stakeholders that don't need them in order to complete their tasks. It follows common sense that a non-technical end-user will not want to customize the execution model but may want to change the graphical presentation to allow the interaction to be more consistent with other tools in their workspace. Further, supporting individual customizations such as allowing end-users to decide whether to be notified by email or by updating their agenda, increases the flexibility and chances for success in the deployment and execution of the workflow process. Customizations should be allowed using property sheets, templates and style sheets, programmatic calls to the system's APIs, and integration with rapid application development and architecture manipulation tools.

3.5.6 Searching/Sorting

With the advent of the WWW, searching and sorting has become indispensable to the user's desktop. The ability to quickly find information that is relevant to completing an activity is important to a smooth running workflow process by making sure the right information is accessible from the right desktop. Users may not only need information, but may require guidance too. The answer to 'How do I do this?' should be derivable by searching for solutions which may be workflows themselves and sorting through their relevance. Searching and sorting can be both proactive and reactive. Proactive searches allow the process participant to go out and seek tools and artifacts they feel is necessary to completing their task and provide a summary of relevant results rather than what is given to them by the process designer in their workspace or on their desktop. Reactive searching is usually initiated by some constraint violation or change in data. It is usually accomplished by an automated agent which combs the information space and returns summaries that might be relevant at a particular step in a workflow process. Sorting can be made using various keys such as date, priority, or amount to determine the relevance and quickly generate an overview of complex and distributed data. It is a difficult problem to manage the visibility of artifacts over time or distributed over a network. Searching and sorting capabilities integrated with a workflow infrastructure allow a more decentralized management model of distributed work by locating and using artifacts and tools on demand and not depending on centralized tracking of all the workflow components.

3.5.7 Process Fragments

Processes are made up of relationships between many components such as people, activities, tools, documents, etc. Not all components are useful or relevant in the execution environment of the workflow process. Process fragments are small sets of relationships between a small number of process objects in a specification. Process fragments should be lightweight, easily manipulatable, self-contained, and usable independent of other parts of the process. What makes up a fragment is usually determined by the process programmer or end-user, so the workflow system should allow grouping and embedding of these processes as well as the traditional actions of cutting, copying, and pasting. Process fragments are useful in that they are usually a small series of steps that have proven useful in the application and execution of another workflow, but reused in a new context. Fragments also encourage the cleaner breakdown of work specification. A small, self-contained process fragment might more easily be embedded in an email or WWW page than the textual description of the process and with better interactive guidance capabilities. Process fragments not only encourage abstraction and reuse, they encourage evolution as well. As fragments are reused over time, the most useful relationships and process patterns may begin to emerge.

3.6 Distribution

Tools and networks have co-evolved advancing the sophistication of the computer technology and resource availability to the average user. While this technology has greatly advanced, basic coordination and collaboration support hasn't kept pace. Distributed computing and network technology have yet to exploit the full potential productivity benefits of such widespread cooperation over communication networks. Collaboration over these networks to date has largely been limited to sharing of data in common formats and across common infrastructures and protocols.**[102][103][104][125]** The policies and interdependencies for sharing, guiding the use of, creating, destroying, manipulating, and handing off of this data have largely been implicit or unspecified. This often leads to miscommunication especially with workflow process participants who interact infrequently such as team members at distributed sites. Because the participants may not be within the same geographic location, and may have working schedules that never overlap, people participating in a distributed workflow process such as a virtual workgroup lack the ability to observe and anticipate the factors that affect the interdependencies between tasks. A virtual workgroup typically lacks the opportunities for informal conversations, such as those encountered in a traditional organization during coffee breaks or hallway exchanges, that would provide the 'overall picture' needed to anticipate coordination problems. Likewise, the time cost in asking what everyone is working on, in order to avoid duplication of effort, may exceed the time taken to simply perform the task at hand.

The Internet, including both private and semi-private networks such as intranets and extranets, aids in scaling down the issues of remote access and availability of both data and resources. Leveraging these types of network infrastructures allows decentralized cooperative work to take place at distributed sites. Various parts of the workflow process can be executed at the location that is most appropriate using the resources available. This overcomes the problem in which some resources are potentially non-distributable because the cost would be prohibitive, but the work utilizing the resources can be completed elsewhere. Each remote process component or fragment may progress independently of other processes with which it coordinates. Local data and execution state can

be protected or handed off to other remote sites as appropriate. Handoff of artifacts and process control can easily piggyback off of the naming and routing idiom of the Internet and WWW using Domain Name Services (DNS) and Uniform Resource Locators (URLs **[28]**) which even the most non-technical user has come to understand. Various parts of the workflow process such as tools, artifacts, specifications, user interfaces, and even the infrastructure itself including the means to change or execute the process should have built-in network support. This encourages design of highly distributed processes, even across low-bandwidth and non-PC-based clients.

3.6.1 Intranet/Internet/Extranet

The Internet is a publicly shareable set of communication networks that cooperate through standardized protocols and allow global access to an abundance of information. Networked communication is critical to distributed workflow within an organization that is geographically dispersed, however, issues of proprietary data and access control limit the types of workflow that can be accomplished using the Internet at large. An Intranet is a private Internet that limits access and data to the appropriate stakeholders. While Intranets allow the sharing of data such as product information, client databases, standards of practice, and work guidelines, the information transmitted usually does not contain workflow guidance or automation capabilities beyond structured text and applet execution for displaying the data. Even applets that may perform a series of complex activities don't involve coordination with other automated agents, activities, workflow processes, or people across time and space. A workflow system should allow downloading of a process specification as well as the means to execute the specification over the protocols in place in addition to managing the coordination and cooperation aspects of each partial process. The mechanism to do this should be the same for executing a workflow process on both an Intranet or across the Internet with the exception that extra care should be taken to ensure data and task privacy.

Extranets, external intranets, allow remote process participants to share information behind firewalls, even between two different company's internal intranets, while maintaining security of the information. Access is usually through a Web site using secure channels to allow a process participant access to resources and data previously shared only at the host company's actual work site. Workflow processes should allow communication across extranet sites. A good approach is to serialize all intermediate messages using only the protocols available for transporting secure information across a public network and those that are allowed to bring data packets into the protected area and through firewalls, like HTTP and secure HTTP. Support for digitally signed processes, tools, and artifacts, permits occlusion of other group's or organization's proprietary processes through abstraction and data protection. The ability to privately tunnel across public communication networks, a form of secure point-to-point data transfer over public lines, provides data, tools, and processes to remote users setting up a component of the infrastructure needed for execution of network-enabled workflow processes.

3.6.2 Distributed Processes

Distributed work requires complementary and cooperating distributed processes. Activities are heavily influenced by local affordances such as local work context and culture, tool and resource availability, and mechanisms for handing off artifacts and documents. Different activities or process fragments can be coordinated through messages using various transport mechanisms and network protocols. The threshold for collaboration by remote process participants is reduced accordingly with the number of communication

mechanisms provided and the amount of overhead the user ‘pays’ in order to handle communication above and beyond the work that is required of them. Allowing participants to choose which mechanism is most appropriate for both their local work context and the shared goals with other users such as WWW browsers, network-aware applications, modem updates, email, or other forms of telecommunication device interactions, helps avoid potential miscommunications. Supporting processes in the workflow infrastructure that can be easily downloaded over these communication channels in addition to the means to execute them strongly encourages distributed cooperation.

Distributed execution of workflow processes are more prone to site failure. Failing one mechanism of communication should precipice action to be taken in the form of an alternate means of contact and handoff. It is important that the workflow system also incorporates the concept of a critical path. Work that ‘falls in the cracks’ or ‘gets lost in the system’ usually occurs between organizational and group boundaries that often involve social expanses as well as physical ones. Policies about distribution should be flexible and allow for distribution of solely the data, the process, or both depending on the process requirements. Distributing a process generally leads to better utilization of distributed resources, but extra care must be taken to ensure the consistency and integrity when distributing the data.

3.6.3 Local/Global

A workflow process can involve thousands of process fragments ranging from simple automation tasks to inter-process coordination across different levels of abstraction. Workflow should fit in at the individual, group, and enterprise levels in addition to supporting the ability to assemble smaller processes into larger ones. Participants in these processes can view their work as being either local or global.**[105]** In a cooperative work setting, all participants are expected to emphasize the work of the community over the individual. However, if a workflow is viewed as only benefiting the global work community and not the individual, the adoption and execution of the process will not be successful. It is important that the workflow is seen for its value in accomplishing local tasks in the work context in addition to not violating global constraints. Individuals, while accomplishing their work, may customize an activity or artifact to best perform their task at hand. While there may exist numerous local optima, some changes violate the global constraints of the workflow. Other activities that relied on the specific format, tool, or process model may break because of the local customizations causing the output of one activity no longer meshes with the input of another. Local data and intermediate artifacts may be used, created, and discarded during the course of a participant’s work if they will not be passed along or used by others outside of the context. Limiting the participants access to customization methods ensures that global constraints are adhered to but at the cost of reducing the flexibility of the system. Integrating constraint management into the workflow infrastructure helps to maintain consistent processes, however, the enforcement of a constraint should only occur at the level of abstraction it is relevant. The impact of changing, reusing, integrating, or embedding a local process into a larger one is minimized when a workflow can be viewed as both local and global depending on the participant. Escalation of workflow processes to encompass global concerns or delegating them to local work contexts should be supported through resolution and abstraction mechanisms in the workflow system.

3.6.4 Routing and Naming

Large organizations may divide workflow processes strategically across groups, participants, sites, systems, and desktops to take full advantage of the capabilities dispersed throughout the company or even across companies. The problem then is twofold: How does a technical or non-technical workflow participant identify an activity, artifact, and resource and how can they correctly find and handoff information artifacts across network protocols? The WWW offers an excellent example of global and inter-networked object routing and namespace management **[25][160]** through its use of universal resource services including Identifiers (URI), Locators (URL), Names (URN), Citations (URC), and Agents (URA **[43]**). Leveraging the Web's routing and naming mechanisms for symbolic representations, bindings to their object in a specific context, and the explicit set of reference instructions provides a comprehensive infrastructure for network-based information dissemination. Using the routing and naming mechanisms of the WWW allows participants to leverage their familiarity with protocols commonly in use across most desktops. The WWW provides the flexibility to allocate objects across a department or enterprise-wide network to best fit a workgroup's needs and intergroup coordination strategies. Adhering to the routing and naming conventions of the Web also has the added benefit that other technologies such as searching, sorting, indexing, and validation engines, sometimes called spidering, can be used to maintain the distributed constraints. URLs allow a non-technical user to specify where a particular query should be routed and which object is desired. A URN represents an institutional commitment to a persistent resource and by releasing an artifact as a named reference to a resource, distributed groups can access the latest object version with the social contract of availability. This allows workflow participants to more easily incorporate remote services into their local workflows in order to build up their automated infrastructure. URCs are usually name-value pairs that represent some type of meta-knowledge about the activity, artifact, or resource. This allows workflow process rationale and assumptions to be embedded in the resources themselves. Because the routing and naming mechanisms of the Web have become so commonplace and the potential for collaboration across these protocols so great, a workflow infrastructure would increase its applicability to a wider range of workflow problems through support of its routing and naming conventions.

3.6.5 Built-in Networking

Built-in networking allows flexible and timely updating of information between remote workflow participants. The meshing of the workflow process infrastructure, the user's process execution workspace, and various networking protocols allows for individual customization and automation of their local tasks while providing dynamic and invisible mechanisms for communication, forwarding, and handoff of messages and artifacts. Providing network connectivity in the workflow infrastructure and even the tasks themselves encourages participants to explore alternatives and build upon others work through shared resources without the overhead of explicitly sending out or posting update announcements.**[2]** For example, a workflow system might use Internet email to send data files between applications that automatically read email, extract data, perform tasks, and generate new email as a response. Traditionally the "workflow" part doesn't include the transport of messages which is done by Internet email, nor the processing of them which is done by the individual applications. By merging the functionality to provide built-in networking and permitting the specification of the delivery transport and schedule, workflow processes can be made to cover a wider domain than simply whether an item is available to be handed off or to whom it is to be delivered.

3.6.6 Low Bandwidth and Disconnected

A major portion of the work that takes place in a workflow process happens offline. Support for low bandwidth and disconnected clients helps in lowering the cost of data maintenance and integrity as well as a process's adoption into a complex and dispersed work setting.**[133][139]** Workflow process tools should allow for both high and low bandwidth collaboration and messaging. High bandwidth tools typically involve video-conferencing or real-time remote data visualization and require unhampered and continuous network connections. While these tools are useful in some contexts, the overall cooperation in a workflow process usually isn't dependent upon these sophisticated utilities and support. Network traffic limitations often make critical path dependence of these technologies in a workflow process impractical. A workflow infrastructure that is not only designed to get around network connection limitations, but embraces a low bandwidth and disconnected model of work will be able to map more coherently to the actual tasks because of its flexibility. Participants should have the capability to detach the process fragments that are necessary for the particular tasks without unnecessarily disrupting the work of others in addition to skillfully re-integrating them when connection is resumed to the overall work process. This synchronization should be supported over standard network access techniques such as modems, infra-red ports, cable, satellite, or normal plugin network connections over common protocols. Assignment, synchronization, and completion policies for handing off processes, data, and documents as well as other artifacts and resources should be flexible to allow alternate models of structural and supervisorial control and coordination. Aspects of a process including confidentiality of data, tracking and status of remote process fragments, timeout, fallback, handoff, and process re-synthesis should be customizable per work context. This allows the individual participants the flexibility and supporting information and infrastructure they need while addressing the global concerns of the process and resource management.

3.7 Dynamic Change

Dynamic change in a workflow system is characterized by continuous change, activity, or progress. Support for dynamic change in a workflow infrastructure may enhance the speed and efficiency of execution in the target context, the convenience of data and artifact tracking and handoff, or the number of choices a participant may make to best accomplish tasks including value-added changes made at the local level.**[1][42]** Also, dynamic change allows re-assignment of execution on-the-fly which provides for greater resource utilization and replacement. The ability to dynamically change a process's definition, the set of views into the process, and the execution model or behaviors at the time it is in progress allows the workflow to better adapt to changing requirements. In a complex, multi-person, ongoing workflow, not all task requirements nor work interruptions can be completely anticipated. Changes to the workflow or changes in the execution context may cause an exception. It is important for a workflow infrastructure to gracefully recover from such occurrences. This may involve dynamic substitution of alternate execution behaviors, transference of objects to the appropriate context, or manipulation of the workflow model either by a participant or by the activity and exception handlers themselves. By allowing the workflow model to query and dynamically change the activities, artifacts, and resources in the face of unexpected conditions, the infrastructure has the ability to organize, optimize, and more closely adapt to the needs and resolve the problems of the organization.

3.7.1 Contingencies and Handoff

Even the best planned workflow will likely encounter exceptions requiring contingencies during execution. Contingencies usually involve planning for the possibility that something may go wrong in addition to the expectation that work will proceed correctly and in synchronization with the workflow specification. In a complex workflow it is impossible to plan for all contingencies, so a good recovery mechanism should be in place that permits a disrupted workflow execution to resume, start from a midpoint, or rollback to a previous point in the process using computer or human consummation of reset, restart, undo, complete, abort, and recover.**[62][135]** Handoff and recovery of artifacts, re-assignment of activities, and delivery of both should be supported at the local level if permissible. Flexible exception handling which includes scoping is key to minimizing the detrimental effects of such unplanned occurrences. Local exceptions should not halt all of the work but rather be propagated through a series of handlers until the problem can be resolved or a workaround such as an alternate path to a solution can be specified. Conversely, a show-stopping exception should be broadcast to the appropriate participants and agents to minimize misdirected work or reflect the newest work priorities in the face of the new constraints.

To facilitate workarounds, handoff of process objects and workflow specifications across networks to dispersed participants should be easily accomplished including context handoff. Clear communication channels, unambiguous work and activity model representations, and common understanding of what is being handed off, and which expectations are being placed upon the recipient are all requirements for successfully transferring work assignments. Handoff of these process objects usually happens during regular workflow execution, but when an exception occurs, roles and assignments can become indeterminate. Mismatch of expectations may occur, and important tasks run the risk of getting lost in the reshuffling of work assignments. Sharing of process objects across sites may not be enough to guarantee the successful handoff and resumption of a process due to differences in local work contexts. Handshaking may need to be done during handoff. Online, the confirmation of items received can be accomplished using an electronic docket or invoice to ensure the existence of items and a digital signature to ensure the quality and accuracy. Offline, a handshake's still a handshake between people, but some online record should be kept to reflect the acceptance, agreement, and social obligations between all participants involved. Because handoffs may require changing execution context, it's important for workflow processes and objects to be able to adapt to the new environment or handle any contingencies gracefully.

3.7.2 Evolution and Optimization

Evolution of process objects and workflows occurs over time as a result of changing tasks, priorities, responsibilities, and even people. Optimization occurs when an improvement to a previous work model results in a better way of doing things either by adding, removing, or redefining process activities and their constraints.**[5][96]** As a workflow process is repeatedly executed small changes are made each time through. Eventually a process will converge on a common practice and become institutionalized. Unfortunately, a common practice and a best practice may not be the same thing. In order to determine this, metrics must be kept to evaluate one execution from another. This evaluation is subjective because the criteria changes the same way the workflow does. Successful workflows, like successful software, will be applied in new situations that may have been unanticipated by the original creator, and unsuccessful workflows will be abandoned or changed. It is important in the workflow infrastructure to allow the

change to occur both before and after deployment of the workflow, by both technical and non-technical participants. Some optimizations may even be performed by the system itself through agents, validation tools, or optimizers.

To better support process evolution and optimization, reuse of process fragments which include small sets of specifications and objects should be easily reusable, divisible, understandable, and capable of being evaluated against some measurable criteria with respect to expected execution or anticipated behavior. Expectation agents and event-monitoring infrastructure are useful for gauging the effectiveness of a workflow process. Also, sometimes when a new process is introduced to a work environment and culture, there is some pushback to its adoption. It is important for the process to be able to adapt to the environment. In addition, process discovery tools are useful for comparing and validating the workflow model with the actual work being accomplished. Wide divergences can indicate technology mismatches, inapplicability, or the need for evolution and optimization of the process. Reuse of process fragments is key to evolutionary development and optimization of workflow processes. A fragment that is successful in one context has a greater likelihood of being successful when tried in another. For instance, in a testing process the participant may require that the same outcome is reached over repeated executions. Different outcomes imply different qualities of the software. Similarly, a process can be used to develop the skill of a particular participant as in a training domain where the end-user's path through the process is dependent upon their skill. The inculcation results in the goal of visiting every activity or series of activities through repeated executions and increased skills. The process may change based on feedback, usage, level of interaction, and number of times executed. As with all changing components, change management techniques such as version control and transactions should be integrated with the system.

3.7.3 Dynamic Behaviors

The ability to dynamically change a process definition, the set of views into the process, or the execution model at the time it is in progress to better fit changing requirements, availability of resources, and the applicability to the current work context is crucial for workflow process execution and evolution over time.**[86]** While introduction of dynamic change to a process and its representation may require additional infrastructure to recognize and enforce consistency, limit access to change mechanisms as appropriate, or correct problems that impact other parts of the process, the ability to dynamically evolve in conjunction with the work environment, culture, and context is important to keeping the online description of work consistent with the actual work being performed. Long-running, distributed processes involving multiple stakeholders will encounter a multitude of situations demanding change, escalation, and reorganization. Dynamic change allows these issues to be addressed with minimal impact to the ongoing, executing workflow process. Late-binding of resources in a workflow allows completion of activities using the resources at hand at the specific point in time the work is actually done. Planning and scheduling components should complement late-binding to ensure that the required amount of resources are available at the appropriate times to complete the tasks at hand.

Handoff of control and data from one person or agent to the next regardless of whether it was planned or unplanned will result in context switching. To aid in the resolution of context awareness and adaptation, it may be necessary to dynamically change object fields, methods, and behaviors at runtime. In the best case, parameterization can accom-

plish this if the changes, handoff, and trans-computation of the workflow are predictable. Most likely, however, the process activities, artifacts, and resources will require some acclimation to the target environment. This may involve downloading of new and existing coordinated views of the work or data models, multiple process interpreters for automation or coordination, and bi-directional exchange of dynamic and changing process object updates in way of work refinement and task clarification. A good mechanism for supporting dynamism in process object management is to separate object data from its behaviors. Object behaviors such as messages can be dynamically loaded and resolved similar to how a model-view-controller implementation style of graphical user interfaces allows dynamic addition and removal of presentation views.

3.7.4 Reflexive

A workflow process is reflexive if during execution it has the ability to remodel itself either automatically by an agent or by enlisting the aid of a human stakeholder in response to status and feedback information about how the process is perceived to be progressing. Reflexivity is particularly useful for generative processes where during the execution of the process, either itself or another process can be built up for delivery and execution. A workflow process or infrastructure component should have the ability at any time to construct, query, and manipulate its own process fragments based on the running state and the target environment. Context awareness, i.e. how the component relates to itself, the outside environment, and other components, may allow self-integration or adaptation to new situations, self-optimization of scheduling and constraints, and self-organization of relationships and dependencies.**[3][15][46]** This level of introspection can be combined with learning rules or business strategies to form an evolution path over time. A reflexive workflow can experiment with resources at its disposal over multiple process iterations to determine the best way to accomplish individual or small group tasks or make explicit inter-group dependencies. Workflow fragments and objects should be symmetric. This means that a workflow process should be able to manipulate its own objects and relationships. In addition the objects themselves should be able to manipulate, query, and dynamically construct existing or new workflows including dynamic and on-demand composition, augmentation, annotation, embedding, and actuation. Supporting reflexivity in a workflow infrastructure allows knowledge about a workflow's applicability to the context and the effectiveness of its deployment to be evaluated over time. Keeping track of a change history in addition to being able to derive change trees and programmatically form queries about them provides a mechanism for re-visiting evolutionary branches. Catalysts for change, whether for better or for worse, are easier to isolate and recognize. Continuous evaluation and adaptation through reflexive understanding of the workflow lends itself well to continuous, automated process optimization through self-modification and knowledge building activities such as self-organization and decentralizing organizational knowledge throughout the participants and agents.

3.7.5 Partial Execution

Partial execution means that the execution of a workflow process is dynamically composable in addition to its components and specification. The effect of this is that the whole process doesn't have to be specified before it starts executing. This is also known as 'just in time' execution, where the definition of the workflow isn't created until it is needed. This is helpful for projects where the downstream details are intentionally left undefined because they are dependent upon upstream results. While just-in-time techniques for process execution may create ambiguity and lack the ability to do global opti-

mizations across all activities before execution, work specifications are generated on-the-fly and on-demand allowing many local optimizations for discriminants other than execution time. Partial execution is analogous to a cat's cradle, a child's game in which an intricately looped string is transferred from the hands of one player to the next, resulting in a succession of different loop patterns. A process's execution should be able to be picked up and executed from any point specified including the dynamic reorganization of local relationships and constraints to fit the new work context. Partial execution supports multiple iterations of a process fragment or multiple alternate iterations of the same process fragment changing order, priority, focus, or completion criteria of the fragment. Support for resolving and integrating processes via pipe-and-filtering, re-stringing, rework, and amalgamation of diverse processes which include possibly competing priorities should be included into the workflow execution infrastructure. This may include temporal and spatial context awareness in addition to the possible execution space and control, coordination, and collaboration policies. For individuals or groups, this may include several partial executions in order to repeat until it's right, i.e. the artifact is good enough to post or handoff to the next participant.

4.0 Advanced Workflow Issues

The principles of an advanced workflow system as enumerated in the previous section aren't applicable across all domains, nor are they mutually consistent. The question is, how are we able to build a workflow design, execution, and deployment infrastructure that adheres to this collection of non-independent and sometimes contradictory principles? The answer lies somewhere in a system's flexibility such that it can evolve over time to address new issues and be applied to new areas. For the most part, there are very few qualitative measurements of what is a good workflow process or how to compare one to another in order to choose the better of the two. Much less, there is very little consensus across disciplines on what the supporting environment for cooperative and collaborative work should look like. The convergence of these disciplines has unshadowed areas where there is no clearly defined best approach. This section looks at some of the tradeoffs, charts the prevailing trends, and explores some of the design impacts.

4.1 Tradeoffs

There are different classes of users, disciplines, and problems. Determining what the idealized environment which would be all things to everyone is a difficult, if not an impossible task. How people work varies so much that to accommodate such diversity, tradeoffs need to be addressed and prioritized. The difficulty of defining such an environment does not make that goal unachievable and some principles can be borrowed from the disciplines studying automated and intelligent agents **[84][94]**. Various approaches have been taken to identify broadly applicable constructs and services that can be used as a basis for specialized problem domains or applications through factoring workflow modeling languages **[132]**, supporting componentization and contextualization in a mobile workflow execution infrastructure **[4][33][75][121]**, or controlling change and change mechanisms for data in a database through transaction management **[91][163]**. All these approaches share a high degree of flexibility with respect to tradeoffs among various uses of the technologies. Solutions require consideration of these issues **[148]**, their approach being far from settled:

- **Proactive vs. Reactive.** Systems that effect execution internally in anticipation or expectation of external changes either by humans or other systems are proactive. Proactive systems perform checks and evaluations, often continuously executing, and assume that determination of state is an inexpensive operation. Proactive systems usually have higher assurances that the process's state is internally consistent and its model more accurate with respect to the world outside the system. Reactive systems generally only perform execution when explicitly requested.
- **Efficiency vs. Effectiveness.** Efficiency is the relation of output to input; effectiveness is the total output. In information theory, a process is both elegant and efficient if no smaller or less costly process can produce the same output in the same amount of time. While in some cases it may be theoretically impossible to determine this optimal process, high resource utilization and broad artifact sharing contribute to its pursuit. An effective process may sometimes maintain resources not directly related to outputs and is only concerned with increasing the final quantity or quality of results possibly independent of inputs, usage, or sharing concerns.
- **Implicit vs. Explicit.** Concurrency of tasks, distribution of objects, and visibility of workflows can all be made implicit or explicit depending on the level of abstraction. Explicit execution of a workflow requires knowledge about the structure and the participant's awareness of where in the process the activity is taking place. Implicit execution implies a focus on the task at hand independent of the progression of the overall process. Explicit representations allow understanding of the details and supports increased precision and unambiguous control. Implicit structures hide details but encourage modularity and are generally less sensitive to changes. Implicit process execution requires more attention as assumptions are made about the context in which it exists.
- **Internalized vs. Externalized.** Using a process to guide a participant through a series of complex steps in order to train them how to accomplish a task is an example of internalization. Symmetrically, writing down or modeling job knowledge such that it can be used by another is externalization. In addition to humans, processes can be internalized or externalized in software systems also. Internalized processes assume that the world is it's own best model and are characterized as a passive process where some knowledge is acquired through unconscious (in humans) or automatic (in agents) mechanisms. Externalized processes maintain a separate model in the real world and measures are taken to ensure it's applicability and consistency.
- **Declarative vs. Procedural.** A declarative statement is an assertion about what is or should be true or valid; a procedural statement is a description of how to achieve it. Ideally, if the control information for a workflow was general, flexible, and adaptable enough, the workflow could be specified completely with declarative, goal-based imperatives. Because human-executed activities are highly dynamic and require flexible exception handling **[135]**, it is difficult and sometimes impossible to completely specify a process in this way. Procedural details for any reasonably complex workflow process may be required for efficiency, policy, quality, or accountability concerns. Also in terms of representability, some workflows are more easily described in procedural rather than logical statements. While the two aren't irreconcilable **[41]**, they represent different paradigms.
- **Guidance vs. Enforcement.** Lack of adequate guiding principles to perform a task will result in wasted effort or mis-directed goals. When a workflow is meant to inform or support, participants seeking guidance willingly give up control to the sys-

tem by allowing their freedom of choices to be limited to the prescripted sequence. Systems that dictate or automate, however, tend to clash with expert or knowledgable participants leading to the failure of a workflow's adoption in that context. Enforcement limits the actions that a participant may choose in order to maintain strict coherence for reasons of quality, accountability, or liability. Over-enforcement of workflow steps can sometimes impede work, but will aid in maintaining standards.

- **Exploration vs. Instruction.** Instruction is a series of activities designed to impart knowledge. This knowledge is often dependent upon information previously learned, incrementally building up to more sophisticated understanding of complex concepts. Instruction usually has set learning goals which have dependencies creating a full or partial ordering on their presentation. Training is a form of instruction where knowledge is built up through multiple iterations over time. Exploration assumes that there are many paths as well as many goals to building up knowledge. Exploration doesn't try to constrain the ordering of nor the relationships between information. It assumes that if an exploration path is beyond understanding, a participant will backtrack to more solid ground. Also, an exploration approach assumes that the goal may be unknown and the level of understanding evolves over time.
- **Precision of Model vs. Understandability.** A planning paradox exists **[143]** in that the more precise a workflow model is the less likely it will be followed. Models generally abstract away detail to focus on specific and relevant concerns, but sometimes lose the information that was important. Many graphical formalisms are able to convey an intuitive impression of a work model, but these formalisms often lack the semantics to support precise, reliable reasoning about them **[131]**. This graphical pen-and-napkin approach of diagramming boxes and arrows to convey a workflow is adequate as long as access to clarification is immediate, but the participant will usually find himself pondering ambiguous markings once back in the office. Formalisms can be sorted by their 'maturity' **[13]** to choose a formalism adequate for the level of precision required, or alternate formalisms and their coordinated representations may aid in the understanding for participants with different skill sets.
- **Structure vs. Consistency.** The structure of workflow representations and data can determine the maintenance costs of keeping the workflow process and data usable, relevant, and consistent. Highly structured representations encourage automation, making it easier to maintain consistency. Tolerating inconsistency increases flexibility in the evolution of the system because multiple competing or irreconcilable representations can be accommodated. This duality is most apparent in computer-executed versus human-executed activities. Tightly constraining human activities may reduce the ability to accomplish tasks in a timely and appropriate manner. Loosening computer automated activities may lead to ambiguity causing exceptions. The cost of data maintenance and correction should be weighed in the face of the requirements and characteristics of the problem being solved by the workflow technology.
- **Testing vs. Evolution.** Concerning workflow deployment, testing of a process before putting it into use improves the quality of the process, similar to how software testing is performed. The similarity, however, breaks down because unlike software, workflow processes need to be contextualized because it is very seldom that they can be used off-the-shelf. Testing requires foreknowledge of the desired workflow possibly using discovery tools or methods and often embodies a desire to standardize human-executed activities to adhere to the tested process. Evolutionary processes are

put into place with the understanding that work changes over time. Evolution allows in-place testing and changes to occur, sometimes even automatically. Testing allows better control over change mechanisms and pruning unproductive evolution branches early before spending resources. Evolution allows human or automated optimization of workflows stemming from greater freedom to change processes to become more applicable to day-to-day work.

4.2 Trends

One trend that is sure to continue, is the utilization of the WWW to perform communication, collaboration, and coordination activities **[123]**. The Web has been successful in helping accomplish tasks in many domains, bringing non-technical users to the network, supporting an enterprise scale computing infrastructure, and advancing cross-platform issues (with the Java VM **[73][74],** legacy access **[82]**, and component technologies **[81]**). The impact of executing workflow processes allowing exchange of ideas and encouragement with colleagues located thousands of miles apart using the Web is a precursor for non-traditional group projects and organization. In order for these 'virtual workgroups' to enter into formal commitments to the completion of a project, even with the absence of interrelationships between participants, several issues remain to be addressed **[59][60]**. Distributing authoring, distributed annotation, linking artifacts and processes (people and tasks), visibility and management of artifacts over time, user involvement, group voting and selection issues, and distributed coordination and project management are all areas where the current Web infrastructures of remote Java execution **[144][145]**, HTTP extensions and optimizations **[61]**, additional protocol specifications and their supporting tools and environments **[105][140][160]**, scripting and markup **[26]**, and net-based notification services fall short. Some successes however, such as the Apache, WebSoft, or (potentially) Openscape (see further references) projects, validate the incremental benefits and open, low-cost adoption to this approach, even without this full set of services such as versioning and transactions **[9][101]**. Augmenting work on the Web with 'diplomacy' between sites or 'socially driven protocols' **[162]** between people and agents to determine agreed upon formats and activities such as in a Summit/Treaty model **[23]** allows negotiation to determine best-fit interaction policies without a top-down enforcement model.

The virtual workgroup approach is not too far from being reality. While organizational processes **[93]** and production workflows have dominated the commercial field the past decade, the counter trend of adopting groupware and ad hoc approaches has broken ground in new directions based on converging technologies and garnered interest at-large **[1][47][48][117]**. Research and concern are being applied to more dynamic and semi-formal work models. With more domains migrating to computers and better techniques for tracking offline activities and artifacts, online work models are becoming more supportive of the unstructured nature of human-to-human and human-computer interactions. Convergence between human and computer performed activities and blending the boundaries between these will continue. Researchers already are extending the continuum. The work to date applying more formal models loosely based on computer execution to describe interactions has come up against the boundaries of human factors. Already the opposite approach of applying human interaction models to agent and system interactions to better coordinate between human executed and computer executed activities is an apparent trend **[90][128]**. The approaches of managing inconsistency, informal, and intelligence (in the form of automated agents) are being pursued.

Convergence of asynchronous workflow with real-time groupware can be seen in joint problem solving systems and support for rapid reorganization based on communication needs. Research experiments grounded in these approaches have appeared in recent literature **[5][14]**. Increased semantic integration between conversation tools and workflow environments instead of blind handoff of control has appeared as work in data integration **[113]**, event-based integration **[129][154]**, heterogeneous systems **[111]**, and enveloping behaviors **[157]**. Some other technological trends that are sure to continue:

- **Lightweight and Componentized**. User interface components, workflow execution infrastructure, and even tools and their integrations can be swapped to provide incremental scaling of the solution and incremental adoption of the technology. Lightweight components encourage customization and reuse at a fine level of granularity fostering the simplest solutions and just-in-time software component assembly and problem solving.
- **Heterogeneous and Open.** Support for open hyperweb protocols and multiple desktop environments are critical success factors for an elaborative process **[12]**. Open systems that provide coherent interfaces to their internal components allow integration with tools in ways the original designers hadn't envisioned. Multiple programming **[120][149]** and process languages allow alternate means of specification of the work model and its behavior increasing a system's flexibility.
- **Ubiquitous and Mobile.** Future computer environments **[2][72][158][159]** are intended to allow access to information when the user demands, across time and space. Computing technology innovation is providing solutions that fit the way people work rather than the other way around. As PDAs, pagers, kiosks, information terminals, organizers, set-, lap-, and desktop information and communication devices become commonplace, more and more information will be delivered through these channels and mechanisms. This underlies the need for increased synchronization and better control and coordination of the information as provided by workflow and event structures for remote communication, wide-scale notification **[10][22][71]**, and architectures amenable to distribution across a variety of platforms **[133][137]**.
- **Dynamic and Adaptable.** In addition to the obstacles of information delivery, transference of information between contexts will require adapting and resolving conflicts and appropriateness. Rapid and automatic reorganization, possibly as the result of self- and context awareness may be necessary even dynamically during the course of execution. The same infrastructure used to adapt to an execution context may be able to perform discovery and evolution of the process in response to environmental feedback thus automating adaptation through task reuse and providing context specific desktops and views **[3][44][114][126]**.
- **Reflexive and Introspective.** Workflow adaptation can't be accomplished without being able to view the state, purpose, and context of a component. An object or component is introspective is it has the ability to assess its own health or integrity and is reflexive if it knows how to relate and change in response to other components in its environment. These properties when combined with various strategies and policies can provide the basis for simulation, discovery, self-correction, self-organization, and self-optimization. Components which can determine this are able to reason about their applicability and perform rich, semantic interoperations with other components **[20]**. Systems such as in **[16][33]** have defined interfaces to support reflexive queries or support introspection as in **[81]**, but policies guiding evolution are not yet readily derivable.

- **Innovative and Invisible Interfaces.** Task specific interfaces or end-user domain-oriented presentations **[63]** are important for supporting multiple stakeholders with varying degrees of knowledge and expertise. Workflow systems should be fundamentally integrated with hypermedia protocols and the research trend should continue to explore the interaction between these two areas for specifying formal and informal relationships **[127]**. The methods of presentation should hide behind familiar paradigms and abstractions rather than forcing end-users into learning new ones, and researchers should work to make process more invisible **[152]**. That is not to say new innovative interfaces using technologies such as VRML and 3D-based systems **[70]**, shared walkthroughs, electronic blackboards, shared whiteboards, and joint problem solving shouldn't be considered. Process should not be seen as a goal in itself, but a means for providing the steps to a solution to support the end-user goals of the right information at the right time.

4.3 Design Impacts

The WWW is expected to play a major part in communication, collaboration, and coordination of remote participants. While the Web itself is a widely universal infrastructure, its data-centric approach and stateless nature of its protocols limit the task dependencies and sychronization, scheduling, and sharing of work. Systems to date that perform workflow process execution in conjunction with the WWW take a bipolar approach. One approach tightly constrains the actions that can be performed by a process stakeholder (such as a participant, end-user, or designer) or software agent, but guarantees the internal data consistency of the workflow model and execution data. The other allows stakeholders to perform arbitrary actions to accomplish their activities but places minimal constraints on the structure of the intermediate and final data artifacts, providing few guarantees of consistency. This section gauges several approaches exhibited by systems along this spectrum.

4.3.1 The Database Model

Traditional workflow systems are built on top of databases. Activities are modeled as schema which include values for who (or what role) is responsible for the activity, what tools are needed to accomplish it, the relative priority of the activity at hand, and data dependencies, including precedence and document handoff. This approach limits the types of actions that an end-user can perform on each activity, as well as the changes that a workflow designer can make to the activities. Because the interactions are limited to pre-specified tools at pre-specified steps in the workflow, data consistency is guaranteed. Database systems also allow transaction support and rollback of processes to a previous state of a workflow progression. This most often happens when the workflow model becomes out of sync with the real activities taking place. Each of the workflow process steps are data-driven. Specification of the control flow is minimal and often hard-coded into the database schema. Fixing the workflow steps and schema in this way requires the process to be stringently adhered to, thus impacting the dynamic evolution capabilities of the system, and also strongly constraining the work practices of the performing organization. Also, by restricting the tools and actions allowed to guarantee data consistency, database systems tend to limit their flexibility. End-users cannot use arbitrary tools to accomplish their tasks. It is difficult to integrate new tools into the process or maintain local process state external to the database. Database systems typically have a high cost of adoption and require a critical mass of users to participate in using the system before many of the workflow benefits are seen. There is no incremental

adoption path for easily mixing human performed activities not stored in the database and those of the process model.

4.3.2 Ad Hoc Systems

An opposite approach to the database model is the ad hoc workflow model. This stems out of the CSCW field and is typified by such commercial systems as Netscape Collabra **[124]** and Microsoft Exchange **[122]**. Users are allowed to perform their work in an "ad hoc" way where solutions and goals are accomplished with whatever tools and data are immediately available. These systems are easily adopted and accessible through the user's desktop or browser. Their primary purpose is to inform users of the current problems, issues, and data needed to accomplish their particular tasks. While this approach tends to be appealing in that it mimics more closely the fast, furious, and unstructured pace of some projects, it provides minimal automation and guidance support. Workflow models lack a coherent, semantically rich, and precise description of the work being performed. Data and control dependencies, as well as assignments, are not immediately viewable, and measurable status of the project is difficult to ascertain. The ad hoc nature of these systems may cause the data and artifacts being created and shared to be inconsistent. However, because the changing data and rationale is recorded over time and is both viewable and shared by all relevant participants, any data inconsistencies are managed by the participants and not the system. This distinction allows for a broad degree of flexibility in accomplishing work, but increases the difficulty for managing it.

4.3.3 Hybrid Approaches

Networks, not databases, should be the focus of attention for workflow systems evolution. Underlying characteristics of the database and storage mechanisms have largely determined the structure of the workflow or process system. A hybrid approach allows distributed work to take place independently with 'just enough' synchronization to maintain workable data consistency, i. e. enforcing local constraints or ignoring global ones when concerns are more immediate. When data consistency becomes stretched, hybrid systems employ some mechanism for resolving inconsistencies, either through replication, negotiation, or mobility. Tolerating temporary deviations and inconsistency of data in workflow process execution is acceptable and can be managed using various techniques **[42][53][62]**.

4.3.3.1 Lotus Domino

Lotus Domino (the WWW extension to Lotus Notes **[115]**) takes a hybrid approach. Domino mixes an underlying messaging and database system with mechanisms to allow stakeholders to view and change data and artifacts through a WWW browser. Data is stored in a Lotus Notes server, and distributed sites use a replication strategy. Domino includes development tools for workflow development and tool integration, and workflows are actually deployed as applications. Domino's support for Java and Java components provides for some post workflow deployment evolution by allowing the presentation or view of the workflow to be changed. However, the underlying workflow model typically does not evolve once the workflow application has been deployed. Domino provides some WWW interfaces for manipulating and extracting information out of the underlying relational database, but typically this is controlled by the workflow application developer and not the end-user. Domino represents a slightly more open approach than the traditional database workflow systems in that custom views into the workflow can be created via the WWW through a standard set of APIs. These views, however, are usually fixed before deployment to maintain data consistency. The tools

integrated into the workflow process, as well as the control flow and data dependencies, rarely (if ever) change once the workflow process begins execution. While this may be necessary to guarantee data consistency, it represents an inhibitor to the usefulness and accuracy of the system.

4.3.3.2 OzWeb

OzWeb **[103]** allies itself much more closely with the WWW-based workflow system**s.** While it too has an underlying database, it provides a much more semantically rich process representation. The OzWeb system is highly componentized and has specific components for dealing with process descriptions, transactions, rules and constraints, tool integration and management, and inter-site coordination. Data or events can trigger rules and constraints to both proactively and reactively drive the workflow process. OzWeb supports evolution in that control flow and data constraint rules can be dynamically added, evaluated, and triggered over HTTP. In order to maintain distributed data consistency, OzWeb employs a summit and treaty negotiation model. When two distributed sites encounter a condition where the data is inconsistent, the system initiates a resolution mode to reconcile, either automatically or with the involvement of human participants, the inconsistent data. In this way, OzWeb allows the distributed components of a workflow process to proceed with local execution while providing coordination and control support. Because OzWeb processes are at its center rules, modification or evolution of a running process often requires a high degree of technical expertise. Those participating in a running process are not always the same stakeholders that have the ability to change the process to fit their work context or habits. While rule-based systems provide excellent support for automation, they create a mismatch between those developing or defining the workflows and those who use or benefit from it. This limits the types of end-user customizations that can be made to an executing workflow system.

4.3.3.3 Endeavors Workflow

The Endeavors **[33]** philosophy is derived directly from the WWW-based model. In a similar manner to an end-user managing the exception when encountering a “page not found” error on the WWW, Endeavors places the burden of resolution initially with the end-user. The end-user then has the option to employ automated or manual workarounds to the exception or inconsistency. Endeavors, however, differs from ad hoc workflow systems in that it includes a semantically rich process description language. This allows highly structured process modeling and execution capabilities to be integrated into the workflow while not being so constraining that ad hoc work is inhibited. Endeavors processes allow multiple, customizable views. The default process view allows both non-technical and technical users to visually edit the workflow. Endeavors is also a highly componentized system. In addition to being able to reference and embed various Endeavors components into WWW pages and e-mail, various process fragments including their data and tools can be sent and executed using HTTP. Endeavors is flexible in its approach. Depending on how the system is customized, the level of data consistency enforcement allows Endeavors to be used to either primarily inform the users of their data and activities as in an ad hoc system or automate certain activities based on their inputs as in a standard workflow system. Similarly, Endeavors’ policies can be customized to enforce the workflow process execution, by limiting the user’s interactions thus keeping the data consistent, or as a support infrastructure to enhance the coordination capabilities between users by loosening these constraints.

5.0 Conclusions

Ubiquity of WWW and Java lends itself well to allowing participation by remote people and teams in a workflow. While these technologies alone are insufficient for addressing all the requirements of an advanced workflow system, they provide a grounding point for incrementally adding in cooperation and collaboration policies in order to demonstrate the feasibility of some of the approaches mentioned for providing distributed workflow functionality. Web protocols and Java allow the infrastructure to be open and extendible, thus providing a mechanism for integration with current and evolving internet technologies including security, data versioning, document modeling and management, broadcast, notification and other distribution or collaboration protocols.

Flexibility will continue to be an emphasis in the development of workflow systems. Networks of servers, clients, and workflow processes are dynamic structures that must tolerate discontinuity, change, and inconsistency. Workflow participants will likely find it necessary to disconnect from the network for a period of time, to travel with a laptop computer for example, and continue their individual activities even using low-bandwidth or disconnected clients **[139]**. A workflow infrastructure should provide the mechanisms to assign, synchronize, and reconcile a workflow participant's state with that of other stakeholders across a wide variety of traditional and non-traditional computing platforms. Changes by the participants doing the work even while the work is being performed will be supported and encouraged as the technologies mature to encourage low-cost adoption, extension, and adaptation to new problems. Workflow design will emphasize support for virtual workgroups through rapid development, customization, and reuse. Non-technical participants will specify and evolve their own workflows through clever, graphical cutting-and-pasting of workflow representations and their behaviors similar to techniques used now in informal editing of directed graphs **[95]**. Complex processes will be easily instantiated and deployed from a process handbook **[118]** or purchased in an open marketplace **[161]**. As advanced workflow system infrastructures begins to embrace the ingenuity, creativity, and dynamism of how human workers accomplish tasks, these systems will help groups become more efficient and effective in their daily activities through better communication and coordination.

Acknowledgments. The authors would like to acknowledge the members of the UCI EDCS projects, particularly Roy Fielding, Jim Whitehead, Ken Anderson, Peyman Oreizy, and Rohit Khare in addition to the Endeavors project team members Clay Cover, Art Hitomi, Ed Kraemer, and Adam Hampson, through whose discussions and interactions have provided an extensive, idea rich environment. Special thanks is reserved for Mark Bergman and Peter Kammer for helping shape the issues, outline, and arguments, and of course, David Rosenblum, David Redmiles, and Michael Leon for their review and comments.

References

1. K. Abbott and S. Sarin, "Experiences with Workflow Management: Issues for the Next Generation", Proceedings of the Conference on CSCW, Chapel Hill, NC, pp. 113-120, 1994.

2. G. Abowd, “Ubiquitous Computing: Research Themes and Open Issues from an Applications Perspective”, Technical Report GIT-GVU 96-24, GVU Center, Georgia Institute of Technology, October 1996.
3. G. Abowd and et al., “CyberGuide: A Mobile Context-Aware Tour Guide”, ACM Wireless Networks, March, 1997.
4. G. Abowd and et al., “Applying Dynamic Integration as a Software Infrastructure for Context-Aware Computing”, GVU Technical Report GIT-GVU-98-01, February, 1998.
5. G. Abowd and B. Schilit, “Ubiquitous Computing: The Impact on Future Interaction Paradigms and HCI research”, CHI’97 Workshop, March 1997.
6. M. Ackerman and D. McDonald, “Answer Garden2: Merging Organizational Memory with Collaborative Help”, CSCW 96 Proceedings, ACM, 1996.
7. M. Ackerman, “A Field Study of Answer Garden”, CSCW 94 Proceedings, ACM, pp.243-252, 1994.
8. Action Technologies, Enterprise Workflow Tools, http://actiontech.com/, 1998.
9. G. Alonso, “Processes + Transactions = Distributed Applications”, Proceedings of the High Performance Transaction Processing (HPTS) Workshop at Asilomar, California, September 14-17th, 1997.
10. G. Alonso, C. Hagen, H.-J. Schek and M. Tresch, “Towards a Platform for Distributed Application Development”, 1997 NATO Advance Studies Institute (ASI). Istanbul, Turkey, August, 1997.
11. V. Ambriola and et. al., “Software Process Enactment on Oikos”, Proceedings of the Fourth ACM SIGSOFT Symposium on Software Development Environments, pp183-192, Irvine, CA 1990.
12. K. Anderson and et al., “Chimera: Hypertext for Heterogeneous Software Environments”, European Conference on Hypermedia Technology (ECHT’94), Edinburgh, Scotland, September 1994.
13. R. Balzer, “Process Definition Formalism Maturity Levels”, Proceedings of the 9th International Software Process Workshop, pp.132-133, October, 1994.
14. S. Bandinelli, “Report on the Workshop on Software Process Architectures”, Milan, March 20-23, 1995.
15. S. Bandinelli and et al., “Process Enactment in SPADE”, European Workshop on Software Process Technology, 1992, pp. 66-82
16. S. Bandinelli, A. Fuggetta, and C.Ghezzi, “Software Process Model Evolution in the SPADE Environment”, IEEE Transactions on Software Engineering, vol. 19 No 12, December 1993.
17. S. Bandinelli, A. Fuggetta, and S. Grigolli. “Process Modeling In-the-large with SLANG”, In Proc. of the Second International. Conference on the Software Process, pages 75-83, 1993.
18. N. Barghouti and et al, “Modeling Concurrency in Rule-Based Development Environments”, IEEE Expert, December, 1990.
19. N. Barghouti and S. Arbaoui, “Process Support and Workflow Management with Improvise”, NSF Workshop on Workflow and Process Automation in Information Systems: State-of-the-Art and Future Directions, May 1996, Athens, GA.

20. N. Barghouti and J. Estublier, “Semantic Interoperability of Process-Sensitive Systems”, AT&T Laboratories and Adele/LSR, France, Research Direction in Process Technology Workshop, Nancy, France, July, 1997.

21. N. Barghouti, E. Koutsofios and E. Cohen, “Improvise: Interactive Multimedia Process Visualization Environment,” Fifth European Software Engineering Conference (ESEC’95), Published as Lecture Notes in Computer Science, W. Schäfer (Ed.), Springer-Verlag, September 1995.

22. D. Barrett, L. Clarke, P. Tarr, and A. Wise, “An Event-Based Software Integration Framework”, Technical Report 95-048 Computer Science Department, University of Massachusetts at Amherst, Revised: January 31, 1996.

23. I. Ben-Shaul and S. Ifergan, “WebRule: An Event-Based Framework for Active Collaboration Among Web Servers”, In Proceedings of The Sixth International World Wide Web Conference, pages 521-531, Santa Clara, Calif., USA, April 1997.

24. M. Bergman, “Criteria for Workflow Systems”, UCI Technical Survey, December, 1996.

25. T. Berners-Lee, “Universal Resource Identifiers in WWW: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web” Internet Engineering Task Force (IETF), draft document, RFC 1630, CERN, June 1994.

26. T. Berners-Lee and D. Connolly, “HyperText Markup Language Specification -- 2.0”, Internet Engineering Task Force, draft document, Work in Progress, MIT/W3C, June 1995.

27. T. Berners-Lee, R. Fielding, and H. Frystyk, “Hypertext transfer protocol -- HTTP/1.0. Internet Informational RFC 1945”, MIT/LCS, UC Irvine, May 1996. http://www.ics.uci.edu/pub/ietf/http/rfc1945.txt

28. T. Berners-Lee, L. Masinter, and M. McCahill. Uniform resource locators (URL). RFC 1738, CERN, Xerox, Univ. of Minnesota, December 1994. ftp://ds.internic.net/rfc/rfc1738.txt

29. J. Blair, (editor), “Office Automation Systems: Why Some Work and Others Fail”, Stanford University Conference Proceedings, Stanford University, Center for Information Technology, 1981.

30. B. Boehm, “A Spiral Model of Software Development and Enhancement”, IEEE Computer, 21(2), pp.61-72, May, 1988.

31. D. Bogia, “Supporting Flexible, Extensible Task Descriptions In and Among Tasks”, Ph.D.Thesis. University of Illinois at Urbana-Champaign. Tech. Report #UIUCDCS-R-95-1925. 299 pages

32. D. Bogia and S. Kaplan, “Flexibility and Control for Dynamic Workflows in the wOrlds Environment”, ACM Conference on Organizational Computing Systems, pages 148-159, Milpitas, CA, 1995.

33. G. Bolcer and R. Taylor, “Endeavors: A Process System Integration Infrastructure”, In 4th International Software Process Workshop, pages 76-85, Brighton, U.K., IEEE Computer Society Press, December 1996.

34. M. Broy, “Control Flow and Data Flow: Concepts of Distributed Programming”, Springer Verlag, 1985.

35. P. Chen, “Entity-Relationship Approach to Information Modeling and Analysis”, Amsterdam: North Holland, 1983.

36. A. Christie, “A Practical Guide to the Technology and Adoption of Software Process Automation”, Software Engineering Institute, CMU/SEI-94-TR-007, March, 1994

37. A. Christie., “Software Process Automation - the Technology and its Adoption”, Springer Verlag, 1995

38. A. Christie and et al. “Software Process Automation: Interviews, Survey, and Workshop Results”; CMU/SEI Technical report SEI-97-TR-008, ESC-TR-97-008, October, 1997.

39. A. Clough, “Choosing an Appropriate Process Modeling Technology”, CASE Outlook, 7(1), pp9-27, 1993.

40. R. Conradi and et al., “Towards a Reference Framework for Process Concepts”, in Software Process Technology, pp 3-17, Springer Verlag 1992.

41. I. Craig, “Converting Declarative into Procedural (and Vice-Versa)”, Technical Report, Department of Computer Science, Reflective Architectures in the VIM Project, University of Warwick Coventry, 1997.

42. G. Cugola, E. Di Nitto, A. Fuggetta, and C. Ghezzi, “A Framework for Formalizing Inconsistencies and Deviations in Human-Centered Systems”, ACM Transactions on Software Engineering and Methodology (TOSEM), 5(3), pp.191-230, 1996.

43. L. Daigle and et al., “Uniform Resource Agents (URAs)”, Internet Engineering Task Force (IETF), draft document, November, 1995.

44. A. Dey, G. Abowd, A. Wood, “CyberDesk: A Framework for Providing Self-Integrating Context-Aware Services. Proceedings of Intelligent User Interfaces ‘98 (IUI ‘98). Jan, 1998.

45. G. Dimitrios, M. Hornick, and A. Sheth, “An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure”, Distributed and Parallel Databases, 3(2), pp.117-153, April, 1995.

46. P. Dourish, “Developing a Reflective Model of Collaborative Systems” ACM Transactions on Computer-Human Interactions, 2(1), pp40-63, March, 1995.

47. E. Dyson, “Why Groupware is Gaining Ground,” Datamation, pp. 52-56, March, 1990.

48. E. Dyson, “Workflow”, Technical report, EDventure Holdings, September, 1992.

49. S. Ehrlich, “Strategies for encouraging successful adoption of office communication systems.” ACM Transactions on Office Information Systems, 340-357, 1987

50. C. Ellis, “Information Control Nets: A Mathematical Model of Office Information Flow” Proceedings of the 1979 ACM Conference on Simulation, Measurement, and Modeling of Computer Systems, 1979.

51. C. Ellis and G. Nutt, “Workflow: The Process Spectrum,” NSF Workshop on Workflow and Process Automation in Information Systems: State-of-the-Art and Future Directions, Athens, Georgia, pp. 140-145, May, 1996.

52. W. Emmerich, “MERLIN: Knowledge-based Process Modeling”, in First European Workshop on Software Process Modeling, published by Associazione Italiana per I’Informatica ed il Calcolo Automatico, Milan, Italy 1991.

53. W. Emmerich and et. al., “Standards Compliant Software Development”, Proceedings of the International Conference of Software Engineering Workshop on Living with Inconsistency, IEEE, 1997.

54. J. Estublier, “APEL: Abstract Process Engine Language”, University of Grenoble, France, 2nd Process Technology Workshop, February 20-23 at U.C. Irvine, 1996

55. C. Fernstrom, “The Eureka Software Factory: Concepts and Accomplishments” vol. 550, Lecture Notes on Computer Science, pp.23-36, October, 1991.

56. C. Fernstrom, “Process Weaver: Adding Process Support to UNIX”, In Proc. of the Second International Conference on the Software Process, 1993.

57. C. Fernstrom and L. Ohlsson, “Integration Needs in Process Centered Environments”, First International Conference on the Software Process, IEEE Press, Redondo Beach, CA, October 1991.

58. R. Fielding and et al., “Hypertext Transfer Protocol -- http/1.1”, Internet Engineering Task Force (IETF), draft document, April 23, 1996.

59. R. Fielding and et. al., “Software Engineering and the Cobbler’s Barefoot Children, Revisited”, Technical Report, EDCS Software Group, University of California, Irvine, November, 1996.

60. R. Fielding and et. al., “Web-Based Development of Complex Information Products”, Communications of the ACM, vo.41, no.8, August, 1998.

61. R. Fielding, J. Gettys, J. Mogul, H. Nielsen, and T. Berners-Lee, “Hypertext transfer protocol -- HTTP/1.1. RFC 2068”, UC Irvine, DEC, MIT/LCS, January 1997. http://www.ics.uci.edu/pub/ietf/http/rfc2068.txt

62. A. Finkelstein and et. al., “Inconsistency Handling in Multi-Perspective Specifications”, Fourth European Software Engineering Conference (ESEC’93), Garmish, Germany, September, 1993.

63. G. Fischer, A. Girgensohn, K. Nakakoji, and D. Redmiles, “Supporting Software Designers with Integrated Domain-Oriented Design Environments” IEEE Transactions on Software Engineering, 18(60), pp.511-522, June, 1992.

64. G. Fitzpatrick, S. Kaplan, and T. Mansfield, “Physical Spaces, Virtual Places and Social Worlds: A Study of Work in the Virtual. CSCW’96, pp.334-343, Boston, MA, 1996.

65. G. Fitzpatrick, W. Tolone, and S. Kaplan. Work, “Locales and Distributed Social wOrlds”, In Proceedings of the 1995 European Conference on Computer Supported Cooperative Work (ECSCW ‘95) pages 1-16, Stockholm, September 1995

66. F. Flores, “Management and Communication in the Office of the Future”, Ph.D. Thesis, Department of Philosophy, University of California at Berkeley, 1979.

67. L. Fosdick, L. Osterweil, “Data Flow Analysis in Software Reliability”, Computing Surveys; vol.8, no.3; pp 305-330, 1976.

68. A. Fuggetta, “Architectural and Linguistic Issues in Supporting Cooperation and Interaction”, Politecnico di Milano, 2nd Process Technology Workshop, February 20-23 at U.C. Irvine, 1996.

69. A. Fuggetta and C. Ghezzi, “State of the Art and Open Issues in Process-Centered Software Engineering Environments”, Journal of Systems and Software, vol.26, pp.53-60, 1994.

70. T. Funkhouser, “RING: A Client-Server System for Multi-User Virtual Environments”, Symposium on Interactive 3D Graphics, pp85-92, 1995.

71. A. Geppert, M. Kradolfer, D. Tombros, “EvE, an Event Engine for Workflow Enactment in Distributed Environments”, Technical Report 96.05, Department of Computer Science, University of Zurich, May 1996.

72. G. Gilder, “The Coming Software Shift”, Telecosm, Forbes ASAP Magazine, August, 1995.

73. J. Gosling, B. Joy, and G. Steele, “The Java Language Specification”, Addison-Wesley. August 1996. http://java.sun.com/docs/books/jls/html/index.html

74. J. Gosling and H. McGilton, “The Java(tm) Language Environment: A White Paper”, Sun Microsystems Inc. October 1995.

75. A. Griff and G. Nutt, “Tailorable Location Policies for Distributed Object Services,” technical report, December, 1997.

76. J. Grudin, “Why CSCW Applications Fail: Problems in Design and Evaluation of Organizational Interfaces”, CSCW 88 Proceedings, ACM, pp85-93, 1988.

77. J. Grudin and L. Palen, “Why Groupware Succeeds: Discretion or Mandate?”, In Proceedings of the 4th European Conference on Computer-Supported Cooperative Work, pages 263-278, Kluwer Academic Publishers.

78. V. Gruhn, “Business Process Modeling in the Workflow Management Environment LEU”, Proceedings of the Entity-Relationship Conference, Manchester, UK, December 1994.

79. V. Gruhn and R. Jegelka, “An Evaluation of FUNSOFT Nets”, in Software Process Technology, pp 3-17, Springer Verlag 1992.

80. V. Gruhn and S. Wolf, “Software Process Improvement by Business Process Orientation”, Software Process--Improvement and Practice, Pilot Issue, 49-56, 1995.

81. G. Hamilton, Ed. “JavaBeans, revision 1.01”, Sun Microsystems, July 1997. http://java.sun.com/beans/spec.html

82. G. Hamilton and R. Cattell, “JDBC(TM): A Java SQL API”, JavaSoft White Paper, April 4, 1996.

83. M. Hammer and J. Champy, “Reengineering the Corporation: A Manifesto for Business Revolution”, Harper, New York, 1993.

84. S. Hanks, M. Pollack and P. Cohen, “Benchmarks, Test Beds, Controlled Experimentation and the Design of Agent Architectures”, AI Magazine, Winter, 1993.

85. D. Harel, “Statecharts: A Visual Formalism for Complex Systems”, Science of Computer Programming, North-Hollad, vol.8, pp.231-274, 1987.

86. P. Heimann and et al., “DYNAMITE: Dynamic Task Nets for Software Process Management”, Proceedings of the 18th International Conference on Software Engineering, pp.331-341, March, 1996.

87. D. Heimbigner, “P4: a Logic Language for Process Programming”, Proceedings of the 5th International Software Process Workshop, pp67-70, Kennebunkport, Maine, 1989.

88. D. Heimbigner. “The ProcessWall: A Process State Server Approach to Process Programming”, In Proc. Fifth ACM SIGSOFT/SIGPLAN Symposium on Software Development Environments, Washington, D.C., 9-11 December 1992.

89. D. Heimbigner, “The ProcessWall: A Process State Server Approach to Process Programming (Revised)”, University of Colorado, Boulder, 2nd Process Technology Workshop, February 20-23 at U.C. Irvine, 1996.

90. D. Heimbigner and A. Wolf, “Micro-Processes”, Software Engineering Research Laboratory, University of Colorado, Boulder, Research Directions in Process Technology Workshop, Nancy, France, July, 1997.

91. G. Heineman and G. Kaiser, “The CORD approach to extensible concurrency control”, Columbia University Department of Computer Science, Technical Report CUCS-024095, February 1996.

92. L. Hubert, “Eureka Software Factory - OPIUM - an Environment for Software Process Modeling Integrated with Project Management and Product Management Facilities”, In First European Workshop on Software Process Modeling, May, 1991.

93. W. Humphrey, “Managing the Software Process”, Addison-Wesley, Reading, MA, 1989.

94. M. Hyun, E. Miller, J. Phillips and R. Smith, “Cognitive Architecture Project”, Agent Properties, on the WWW http://krusty.eecs.umich.edu/cogarch2/index.html

95. B. Ibrahim, “Optimizing Cut-and-Paste Operations in Directed-Graph Editing”, HCI International ‘97, San Francisco, California, August, 1997.

96. M. Jaccheri and R. Conradi, “Techniques for Process Model Evolution in EPOS”, IEEE Transactions of Software Engineering, 12:19, December 1993, Special issue on Software Process Evolution, pp 1145-1156.

97. G. Junkermann and W. Schaefer, “Merlin: Supporting Cooperation in Software Development through a Knowledge-based Environment”, Advances in Software Process Technology. Publisher John Wiley, 1994.

98. R. Kadia, “Issues Encountered in Building a Flexible Software Development Environment: Lessons from the Arcadia project”, In Proceedings of ACM SIGSOFT Fifth Symposium on Software Development Environments, pages 169-180, Tyson’s Corner, VA, December 1992.

99. R. Kadia, “The Collected Arcadia Papers on Software Engineering Environment Infrastructure”, vol 1 & 2, Second Edition, University of California, Irvine, 1995.

100. G. Kaiser, “Rule-based modeling of the Software Development Process”, Proceedings of the 4th International Software Process Workshop, October 1988. Published in ACM SIGSOFT Software Engineering Notes, June, 1989.

101. G. Kaiser, “Talks on OzWeb, DkWeb, Amber, Pern, CORD, Summits and Treaties” Columbia University Department of Computer Science, 2nd Process Technology Workshop, February 20-23 at U.C. Irvine, 1996.

102. G. Kaiser and et al., “WWW-based Collaboration Environments with Distributed Tool Services”, World Wide Web Journal, Baltzer Science Publishers, January, 1998.

103. G. Kaiser, S. Dossick, W. Jiang, and J. Yang. “An Architecture for WWW-Based Hypercode Environments”. In Proceedings of the 19th International Conference on Software Engineering, pages 3-13, Boston, Mass. USA,
May 1997.

104. G. Kaiser, S. Dossick, W. Jiang, and J. Yang. “WWW-based Collaboration Environments with Distributed Tool Services” World Wide Web Journal, vol. 1, pp. 3-25, January, 1998.

105. G. Kaiser and E. Whitehead, “Collaborative Work: Distributed Authoring and Versioning”, Column IEEE Internet Computing, pp.76-77, March/April, 1997.

106.S. Kaplan, G. Fitzpatrick, T. Mansfield, and W. Tolone, “MUDdling Through”, In proceedings Thirtieth Annual Hawaii International Conference on Systems Sciences (HICSS’97), vo.II, IEEE Computer Society Press, pp.539-548, Jan. 1997.

107.M. Kellner, “Software Process Modeling Support Management Planning and Control”, In Proc. of the First International Conference on the Software Process, 1991.

108.R. Kling, “Cooperation, Coordination, and Control in Computer Supported Work”, Communications of the ACM, 34(12), pp.83-88, 1991.

109.H.-U. Kobialka and A. Gnedina, “Customizing Dynamics in Configuration Management Systems”, 2nd International Conference on Concurrent Engineering, pp.557-564, August, 1995.

110.K. Kraemer and J. King, “Computer-Based Systems for Cooperative Work and Group Decision Making”, ACM Computing Surveys, 20(2), pp.115-146, June, 1988.

111.N. Krishnakumar and A. Sheth, “Managing heterogeneous Multi-System Tasks to Support Enterprise-Wide Operations”, Distributed and Parallel Databases, 3(2), ppp.155-186, April, 1995.

112.K.-Y. Lai, T. Malone, and K.-C. Yu, “Object Lens: A ‘Spreadsheet’ for Cooperative Work”, ACM Transactions of Office Information Systems, vol.6, pp.332-353, 1988.

113.J. Lee, Y. Gregg, and PIF Working group, “The PIF Process Interchange Format and Framework”, Technical report, University of Hawaii Information and Computer Science Department, February, 1996.

114.M. Lehman, “Laws of Software Evolution Revisited”, Proceedings of 5th European Workshop of Software Process Technology, EWSPT’96, pp.108-124, Nancy, France, October, 1996.

115. Lotus Development Corp., “Domino.Doc Whitepaper”, Lotus Development Corp, April 1997. http://www2.lotus.com/domino/doc.nsf/

116.M. Markus and T. Connolly, “Why CSCW Applications Fail: Problems in the Adoption of Interdependent Work Tools”, CSCW 90 Proceedings, ACM, pp371-380, 1990.

117.T. Malone and K. Crowston. “The Interdisciplinary Study of Coordination”, ACM Computing Surveys, 26(1), pp87-119, March, 1994.

118.T. Malone, K. Crowston, J. Lee, and B. Pentland, “Tools for Inventing Organizations: Toward a Handbook of Organizational Processes”, Workflow 95 Conference Proceedings, pp.57-78, 1995.

119.T. Malone and K. Crowston, “What is Coordination Theory and How Can It Help Design Cooperative Work Systems?” Proceedings of the Conference in Computer-Supported Cooperative Work, Los Angeles, CA, pp.357-370, October, 1990.

120.M. Maybee, D. Heimbigner, and L. Osterweil, “Multilanguage Interoperability in Distributed Systems: EXPERIENCE REPORT” Proceedings of the 18th International Conference on Software Engineering (ICSE-18), pp 451-463, March, 1996.

121.D. McCrickard, “Information Awareness on the Desktop: A Case Study”, Technical Report GIT-GVU-97-03, GVU Center, Georgia Institute of Technology, March, 1997.

122. Microsoft Corporation, Exchange Server, http://www.microsoft.com/products/prodref/599_ov.htm

123.J. Miller and et al. “The Future of Web-based Workflows”, LDIS, Department of Computer Science, University of Georgia, Athens, Research Direction in Process Technology Workshop, Nancy, France, July, 1997.

124. Netscape Communications, “Netscape collabra server 3.0, Open Discussion Server for Enterprise Collaboration, Datasheet”, Netscape Communications Inc., 1997. http://home.netscape.com/comprod/server_central/product/news/collabra3_data.html

125.G. Nutt, “The Evolution Toward Flexible Workflow Systems,” Distributed Systems Engineering, 3, 4 (December 1996), pp. 276-294.

126.G. Nutt and et al., “Dynamically Negotiated Resource Management for Virtual Environment Applications,” to appear IEEE Transactions on Knowledge and Data Engineering, 1999.

127.G. Nutt, T. Berk., S. Brandt, M. Humphrey, and S. Siewert, “Resource Management for a Virtual Planning Room,” 1997 International Workshop on Multimedia Information Systems, pp. 129-134, Como, Italy, September, 1997.

128.B. Nuseibeh, “Software Processes Are Human Too”, Department of Computing, Imperial College, London, Research Direction in Process Technology Workshop, Nancy, France, July, 1997.

129. Object Management Group, “The Common Object Request Broker: Architecture and Specification, Revision 2.0”, July 1996. http://www.omg.org/corba/corbiiop.htm

130.L. Osterweil., “Software Processes are Software Too”, In Proceedings of the Ninth International Conference of Software Engineering, Monterey CA, March 1987.

131.L. Osterweil, “Software Processes Are Software Too, Revisited”, In Proceedings of the International Conference on Software Engineering ‘98, Boston, MA, pp. 540-548, May 1997.

132.L. Osterweil and S. Sutton, “Factored Language Architecture (of Julia)”, University of Massachusetts, Amherst, 2nd Process Technology Workshop, February 20-23 at U.C. Irvine, 1996.

133.D. Perry, “Direction in Process Technology - An Architectural Perspective”, Bell Laboratories, Research Direction in Process Technology Workshop, Nancy, France, July, 1997.

134.J. Peterson, “Petri Net Theory and the Modeling of Systems”, Prentice-Hall, Inc. 1981.

135.H. Saastamoinen, M. Markkanen, and V. Savolainen, “Survey on Exceptions in Office Information Systems”, Technical Report CU-CS-712-95, Department of Computer Science, University of Colorado, Boulder, 1994.

136.P. Sachs, “Transforming Work: Collaboration, Learning, and Design”, Communications of the ACM, 38(9), pp36-44, September, 1995.

137.J. Salasin, DARPA briefing, “Overview: Military Applications of Future Software Technologies”, 1997.

138.H. Simon, “The Sciences of the Artificial”, 2nd edition, MIT Press, Cambridge, MA, 1981.

139.P. Skopp, “Low Bandwidth Operation in a Multi-User Software Development Environment”, Columbia University Department of Computer Science, Master’s Thesis.

140.J. Slein, F. Vitali, E. Whitehead Jr., and D. Durand, “Requirements for Distributed Authoring and Versioning on the World Wide Web”, Internet-draft, work-in-progress. July 1997. ftp://ds.internic.net/internet-drafts/draft-ietf-webdav-requirements-01.txt

141.X. Song and L. Osterweil, “Engineering Software Design Processes to Guide Process Execution” Proceedings of the 3rd International Conference on the Software Process. Reston, VA, October 1994, pp 135-152.

142.L. Suchman, “Office Procedure as Practical Action: Models of Work and System Design”, ACM Transactions on Office Information Systems, 1(4), pp.320-328, October, 1983.

143.L. Suchman, “Plans and Situated Actions: The Problems of Human-Machine Communication”, Addison-Wesley, Reading, MA, 1987.

144. Sun Microsystems, “Java Remote Method Invocation Specification”, Sun MicroSystems, Inc., 1997. http://www.javasoft.com/products/jdk/1.1/docs/guide/rmi/spec/rmiTOC.doc.html

145. Sun Microsystems, “The Java servlet API. Whitepaper”, Sun MicroSystems, Inc.,1997. http://jserv.javasoft.com/products/java-server/webserver/fcs/doc/servlets/api.html

146.S. Sutton, Jr., D. Heimbigner, and L. Osterweil. “APPL/A: A Language for Software Process Programming” ACM Transactions on Software Engineering and Methodology, July 1995.

147.S. Sutton Jr. and L. Osterweil “The Design of a Next-Generation Process Language”, Proceedings of the Fifth ACM SIGSOFT Symposium on the Foundations of Software Engineering, Zurich, Switzerland, September 1997. Lecture Notes in Computer Science #1301, pp. 142-158.

148.K. Swenson and K. Irwin, “Workflow Technology: Tradeoffs for Business Process Re-engineering”, Conference on Organizational Computing Systems, pp22-29, 1995.

149.T. Taft, “Programming the Internet in Ada95”, draft submitted to Ada Europe ‘96. March 15, 1996.

150.P. Tarr and L. Clarke, “PLEIADES: An Object Management System for Software Engineering Environments”, In Proc. of the First ACM SIGSOFT Symp. on the Foundations of Software Engineering, pp. 56-70, December 1993.

151.F. W. Taylor, “Principles of Scientific Management”, Harper & Row, New York, NY, 1911.

152.R. Taylor, “Dynamic, Invisible, and on the Web”, University of California Irvine, Research Direction in Process Technology Workshop, Nancy, France, July, 1997.

153.R. Taylor, F. Belz, L. Clarke, L. Osterweil, R. Selby, J. Wilden, A. Wolf, and M. Young. “Foundations for the Arcadia Environment Architecture”. In *Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments*, pages 1–13. ACM Press, November 1988.

154.R. Taylor and et al, “A Component- and Message-Based Architectural Style for GUI Software”, IEEE Transactions on Software Engineering, June 1966.

155.R. Taylor, and et al., “Chiron-1: A Software Architecture for User Interface Development, Maintenance, and Run-Time Support”, ACM Transactions on Computer-Human Interaction, Volume 2 Number 2, June 1995.

156.W. Tolone, “Introspect: a Meta-level Specification Framework for Dynamic, Evolvable Collaboration Support”, Ph.D. Thesis. University of Illinois at Urbana-Champaign.

157.G. Valetto and G. Kaiser, “Enveloping Sophisticated Tools into Process-Centered Environments”, Automated Software Engineering, Kluwer Academic Publishers, 1995

158.M. Weiser, “The Computer of the 21st Century”, Scientific American, 265(3), pp.66-75, September, 1991.

159.M. Weiser, “Some Computer Science Issues in Ubiquitous Computing”, Communications of the ACM, 36(7), pp.75-84, July, 1993.

160.E. Whitehead, “World Wide Web Distributed Authoring and Versioning (WebDAV): An Introduction”, ACM Standard View, 5(1): 3-8, March 1997.

161.E. Whitehead and et al., “Software Architecture: Foundation of a Software Component MarketPlace”, Proceedings of the First International Workshop on Architectures for Software Systems in cooperation with ICSE-17, Seattle, Washington, April 24-25, 1995

162.T. Winograd, “Categories, Disciplines, and Social Coordination,” Computer Supported Cooperative Work, vol.2, pp.191-197, 1994.

163.J. Yang and G. Kaiser, “WebPern: An Extensible Transaction Server for the World Wide Web”, Tech Report CUCS-004-98, Columbia University Department of Computer Science, January, 1998.

164.P. Young, “Customizable Process Specification and Enactment for Technical and Non-Technical Users”, Ph.D. Dissertation, University of California Irvine, 1994.

165.P. Young and R. Taylor, “Human-Executed Operations in the Teamware Process Programming System”, Proceedings of the Ninth International Software Process Workshop.

166.P. Young and R. Taylor, “Process Programming Languages: Issues and Approaches”, 17th International Conference on Software Engineering, Workshop on Software Engineering and Programming Languages, Seattle, Washington, April 23-30, 1995.

167.T. Winograd and F. Flores, “Understanding Computers and Cognition”, Addison-Wesley, 1987.

Web Citations and Further References

[ActionTech] - http://www.actiontech.com/ - Action Technologies

[Advanced Workflow] - http://gbolcer.ics.uci.edu/papers/AdvancedWorkflow.pdf - this document

[Apache] - http://www.apache.org/ - Apache Web Server Project

[C2] - http://www.ics.uci.edu/pub/c2/ - Chiron-2 Software Architecture

[Chimera] - http://www.ics.uci.edu/pub/chimera/ - Chimera Open Hypertext

[DARPA-EDCS] - http://www.darpa.mil/ito/research/edcs/ - DARPA's Evolutionary Design of Complex Software - http://www.darpa.mil/ito/ - Information Technology

[Endeavors] - http://www.ics.uci.edu/pub/endeavors/ - Advanced Workflow

[Java] - http://www.javasoft.com/ - Javasoft

[LSDIS] - http://lsdis.cs.uga.edu/ - Large Scale Distributed Information Systems, University of Georgia

[Openscape] - http://www.openscape.org/ - Openscape WWW Browser Project

[PSL] - http://www.psl.cs.columbia.edu/ - Programming Systems Lab, Columbia University

[SEI] - http://www.sei.cmu.edu/ - Software Engineering Institute

[W3C] - http://www.w3.org/ - The World Wide Web Consortium

[WebDAV] - http://www.ics.uci.edu/~ejw/authoring/ - Distributed Authoring and Versioning on the WWW, IETF Working Group

[WebSoft] - http://www.ics.uci.edu/pub/websoft/ - WWW Software and Protocols

[WfMC] - http://www.wfmc.org/ - Workflow Management Coalition

Definitions and Usage

Advanced Workflow: a body of technologies and approaches that makes a philosophical assumption about the evolutionary nature of cooperative work in that it is most elegantly accomplished when it can be described, shared, visualized, distributed, dynamically changed, and interconnected.

Collaboration: a joint intellectual effort in which sharing of information is required.

Communication: exchange of thoughts, messages, or information; or the mechanisms by which they are transmitted.

Cooperation: the act of association for furthering a goal that is of mutual benefit.

Coordination: the act of managing interdependencies between activities performed to achieve a goal.

Elegance: the information theoretic principle that no other construct can be proven to achieve the same goal with the same constraints; usually size, time, or cost.

Evolution: a process that involves change over time usually in the face of a more efficient or elegant solution.

Market: a public gathering for exchange of merchandise or ideas.

Monotonic: incremental change that does not invalidate previous changes and maintains their consistency.

Procedure: a style of performing or effecting a goal.

Process: a series of actions, changes, or functions toward accomplishing a goal.

Stakeholder: one who has a stake, interest, reward or penalty in the outcome of an endeavor.

Workflow: the interaction of humans and computer processes to generate and share information.