Hypertext to Knowledge to Workflow

Roy Rada*, Antonios Michailidis*, Christian Frosch**, Ming Lei***

*Department of Information Systems
University of Maryland, Baltimore County
Baltimore, Maryland

**Institute of Toxicology
University of Mainz
Mainz, Germany

***Department of Electrical Engineering and Computer Science
Washington State University
Pullman, Washington

Abstract

The engineering hypothesis is that a hypertext-like, collaborative authoring system can provide an appropriate infrastructure for a knowledge and workflow management system. A hierarchical, hypertext infrastructure with typed, multi-attributed nodes provides the platform. People perform their scheduled activities by creating nodes in the system and they comment on one another's work. Such a system has been designed and built, as documented here. The engineering result suggests new issues for the design of the next generation of the system. The experimental hypothesis is that people will use such a system to manage knowledge and work. Knowledge management has been successfully supported in a software engineering team. Workflow management was only partly supported for these software engineers in part because they often relied on informal working methods that did not match well the scheduling capabilities of the system.

Table of Contents

1 Introduction

Today's business environment is characterized by increased complexity and change. Complexity may be attributed to increases in knowledge and specialization of the organizational resources (e.g. people, information, products, services). Business events are emerging faster, their duration is shorter, and are more likely to affect the organization structure and resources due to increased interdependencies (Malone and Crowston, 1990). Business needs that arise in response to increased complexity and change include:

- improved cooperation and communication between managers and the people they are managing,
- reduced time to make decisions and improved quality of decisions, and
- rapid re-engineering and redesign of organizational processes.

Organizations need the ability to re-engineer and optimize their information and business processes (Daft and Lengel, 1986; Malone et al, 1999). To achieve this goal they need to manage (create, access, communicate, evaluate, apply, and distribute) knowledge and use that knowledge to make informed decisions on how to re-engineer business processes. That is, knowledge and information have little value unless they lead to action towards achieving organizational goals (Papows, 1998).

Knowledge and workflow management share a common motivation: to ensure that organizational outcomes satisfy the organization's objectives. However, process re-engineering and technology interventions alone are not sufficient to achieve the organization's goals. It has been argued that changes in organizational structure, administrative policies, management style, information technology, organizational learning practices and workflow design, stimulate cognitive changes on how individuals perceive their work environment, which in turn lead to individual behavioral changes on how they perform their work. Individual behavior and performance is considered an instrumental mediating factor in determining the effect of organizational change on organizational output (performance, quality, productivity, production cost, and efficiency). Empirical evidence also suggests that enhanced individual performance would result in improved organizational outcome and performance (Leonard et al, 1997). Given these observations, organizations should augment knowledge and workflow management with quality control mechanisms in order to measure and monitor individual performance. Quality control requires that an organization's activities be documented and in line with its objectives (Rada, 1997). How do organizations exploit knowledge and workflow management to help address these problems? A seamless information infrastructure is proposed for integrating hypertext, knowledge, and workflow management.

The term 'knowledge management' was coined by Karl Wiig in a 1986 Swiss conference sponsored by the United Nations (Beckman, 1999). Wiig (1997) has said that knowledge management is the systematic, explicit, deliberate building, renewal, and application of knowledge to maximize an enterprise's knowledge-related effectiveness and returns from its knowledge assets.

Knowledge, within an organization, may exist in two interrelated forms: tacit and explicit. Tacit knowledge is stored in people's minds and is closely tied to user experiences. Explicit knowledge is externally represented, as in reports, videos, email, and business process descriptions. Knowledge, both explicit and tacit, is not static but rather it flows within an organization where it is shared, evaluated, communicated, augmented, and modified (Borghoff and Pareschi, 1997). Figure 1 shows the four processes of knowledge conversion and interaction: socialization, externalization, internalization, and combination. New tacit knowledge can be generated through internalization of external knowledge representations, such as by reading and revising documents or participating in lectures. New explicit knowledge can be created by externalizing tacit knowledge, such as through annotations on existing documents or describing and documenting effective work practices, policies, and work progress. Based on the above framework Broghoff and Pareschi (1997) define knowledge management as "… the management of the environment that makes knowledge flow through all the different phases of its life-cycle".

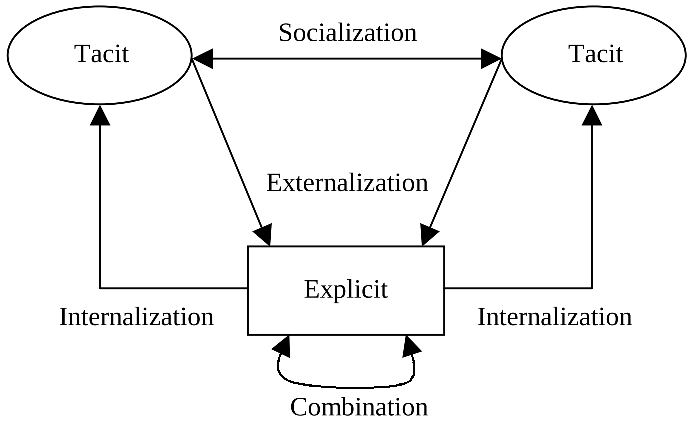


Figure 1. Knowledge conversion life-cycle.

Hypertext is an extended form of text that in turn is a predominant vehicle for knowledge (Rada, 1991). A hypertext system supports some of the requirements of a knowledge management system. However, knowledge management goes further in the direction of improving an enterprise's performance through decision-making. Macintosh et al (1999) say, "Knowledge management involves the identification and analysis of available and required knowledge assets and knowledge asset-related processes, and the subsequent planning and control of actions to develop both the assets and the processes so as to fulfill organizational objectives." This definition of knowledge management is remarkably similar to the definition of workflow management.

A workflow management system supports the flow of both data and control. Data flow is concerned with the routing of information and documents passed from one participant (human or computer system) to another. Control flow is concerned with the rules and constraints that determine the sequencing of activity execution (Hollingsworth, 1995). People, software systems, or a combination of these can execute workflow tasks. Typically a workflow management system provides the following functionality (Georgakopoulos and Hornick, 1995):

- process modeling and workflow specification
- workflow implementation and automation - scheduling, execution, automation, tools, correctness verification and validation.

The Workflow Management Consortium (WfMC) has defined an architecture for workflow management systems that emphasizes the interoperability of components (Lawrence, 1996). The key components are the workflow engine that in turn communicates with a monitoring module, a client module, a process definition module, and an application module.

Workflow management systems typically use two types of process modeling methodologies: conversation-based methodologies and activity-based methodologies (Sheth, et al., 1996). Conversation-based modeling represents workflow processes as a network of commitment (language acts rather than activities or tasks that agents perform) loops between a customer and performer. Performer reward is implicit in the agreement between customer and performer. However, this model does not by itself imply a workflow specification language for defining workflow processes. In the general Workflow Management System architecture described in (Medina-Mora, et al, 1992), the analyst defines the workflow activity. Activity-based models (Smith et al, 1989) represent the control flow (sequencing, parallelism) of activities as a network of activities and/or nested sub-activities.

Papows (1998) argues that the goal of knowledge management systems is to integrate structured and unstructured forms of information that exist and are produced within a company with the work activities of people. He identifies one type of structured information (data) and two types of unstructured information (information and knowledge). Data is simple facts. Information is data that has a meaning (different from raw data) within a given context. Knowledge is information that is accompanied with relevant "know-how" and "know-why". Finally, work occurs when someone is using a combination of data, information, and knowledge to act. The output of work may be new data, information or knowledge.

The distinction between workflow management and knowledge management concerns the emphases. Workflow management emphasizes people doing their tasks. Knowledge management emphasizes the recording of knowledge and the storing of it in such a way that someone requiring that knowledge can readily locate and use it. Through a hypertext base how would can one create first a knowledge management and then a workflow management system?

This work reported here began in 1985 to support teams operating across the Internet with a hypertext collaborative authoring system. The system has been called the Many Using and Creating Hypertext (MUCH) system. The goal of the work reported here was to make MUCH more suitable for dealing with roles and schedules in a new system called MUCH2000. The emphasis has moved from one of document authoring to one of knowledge management and then workflow management. Two claims are made:

1. **Engineering Claim:** The engineering hypothesis is that a hypertext system augmented with roles and schedules can provide an appropriate infrastructure for a knowledge and workflow management system. The design and implementation will be based on a hierarchy of attributed nodes. A Superbook-like (Egan et al, 1989) interface will provide access to the core functions.
2. **Usability Claim:** People will use the system to create, store, and retrieve their documents. Also the system will support the scheduling of activities for roles, people will assume the appropriate roles, and scheduled work will be performed.

The test for the first claim comes in building the system and showing its functionality. The second claim concerns the way that people will use the system and is tested in case studies.

2 Engineering

The requirements, then the design, and finally the implementation of this system are presented next.

2.1 Requirements

The goal is to create a system that supports workflow and knowledge management. Activities in the system are performed by roles rather than by individual end users. End users become agents in roles. An end user applies for a role, and if accepted into the role, then the end user becomes an agent of that role and may perform the role's activities. In theory, agents can be either humans or software robots, though in the current requirements only human agents are supported.

Agents create, schedule, and perform activities. The organization thrives and shares its resources with its agents in a way that encourages continual improvement in the organization. Activities that an agent performs include:

- Agent in role R1 defines subordinate role R2.
- R1 reviews agent applications to assume role R2 and either accepts or rejects.
- R1 creates objectives and schedules for R2.
- R2 performs scheduled work.
- R1 either comments on work of R2 or assigns others to do so.

Agents assume roles in a role hierarchy. Agents through their roles are able to create and manipulate subordinate roles. All opportunities to perform activities are determined by objectives and schedules that a role creates and assigns to a subordinate role in the role hierarchy. A schedule node specifies an objective node to which the role must go, and the schedule node specifies a type of descendant node that the role should create underneath the objective node.

Access control is role-based. A role is scheduled to achieve an objective. The agent assuming that role can access the objective and any nodes subordinate to the objective. In addition to access, the schedule for performing the objective will specify what type of node the agent may create beneath the objective node (Jaeger, et al, 1999).

Activities may be decomposed into nested sub-activities. For instance, a software project leader creates several group leader roles and an objective for each group to develop a software component and a schedule to do that by the end of the quarter. End users apply for the group leader position. An agent in the role of group leader might create subordinate roles of designer, programmer, or any other roles appropriate for working as a software group. Roles may be assigned to create objectives and schedules. Agents create further nodes of various types based on the schedules of the roles they assume. Quality control is enforced by assessment that one agent can make on the work of another.

2.2 Information Infrastructure

All work done in MUCH2000 is stored in nodes that are organized in a hypertext structure. Hypertext is nodes of documents connected by links, and a hypertext system supports traversal of these links. Nodes are organized in a directed, ordered tree based on child-parent and sibling relationship (Wang and Rada, 1998). Every node except the root node has exactly one parent. Every child has exactly one preceding sibling except the first child.

Functions in the system create options for users to create new nodes based on the values in existing nodes. The node types include:

- schedule nodes that specify role(s) that need to achieve objectives in certain time frames (specified by start date and end date),
- role nodes that indicate a position in the role hierarchy,
- people nodes that describe a person,
- agent nodes that indicate a person filling a role, and
- comment nodes that are generated as a result of an assignment to comment on another node.

Attributes of a comment node include narrative as well as numerical evaluations. Roles are dynamically defined in terms of schedules or tasks the role is assigned to perform. This role definition not only denotes a group of participants but also encapsulates the tasks that the role is assigned to perform (see Figure 2). A role R when viewing a node N has the option to generate a schedule node that takes N as the objective. R may assign N to any role subordinate to R (see Figure 2). R will specify type of response, start date, and due date.

In more detail, nodes have the following attributes and constraints. Note that all nodes inherit the attributes of the Generic node.

Generic

- *attributes*: id, type, name, date created or last modified, author name, author role, parent id, sibling id
- *constraints*: id is unique key; allowable types include schedule, role, people, agent, blob, and comment.

Schedule

- *attributes*: target id, start date, due date, and type of response (includes adding a child node of any specific type).
- *constraints*: parent id must be a node of type 'role'

Role

- *attributes*: same as GenericObject
- *constraints*: parent id must be a node of type 'role'

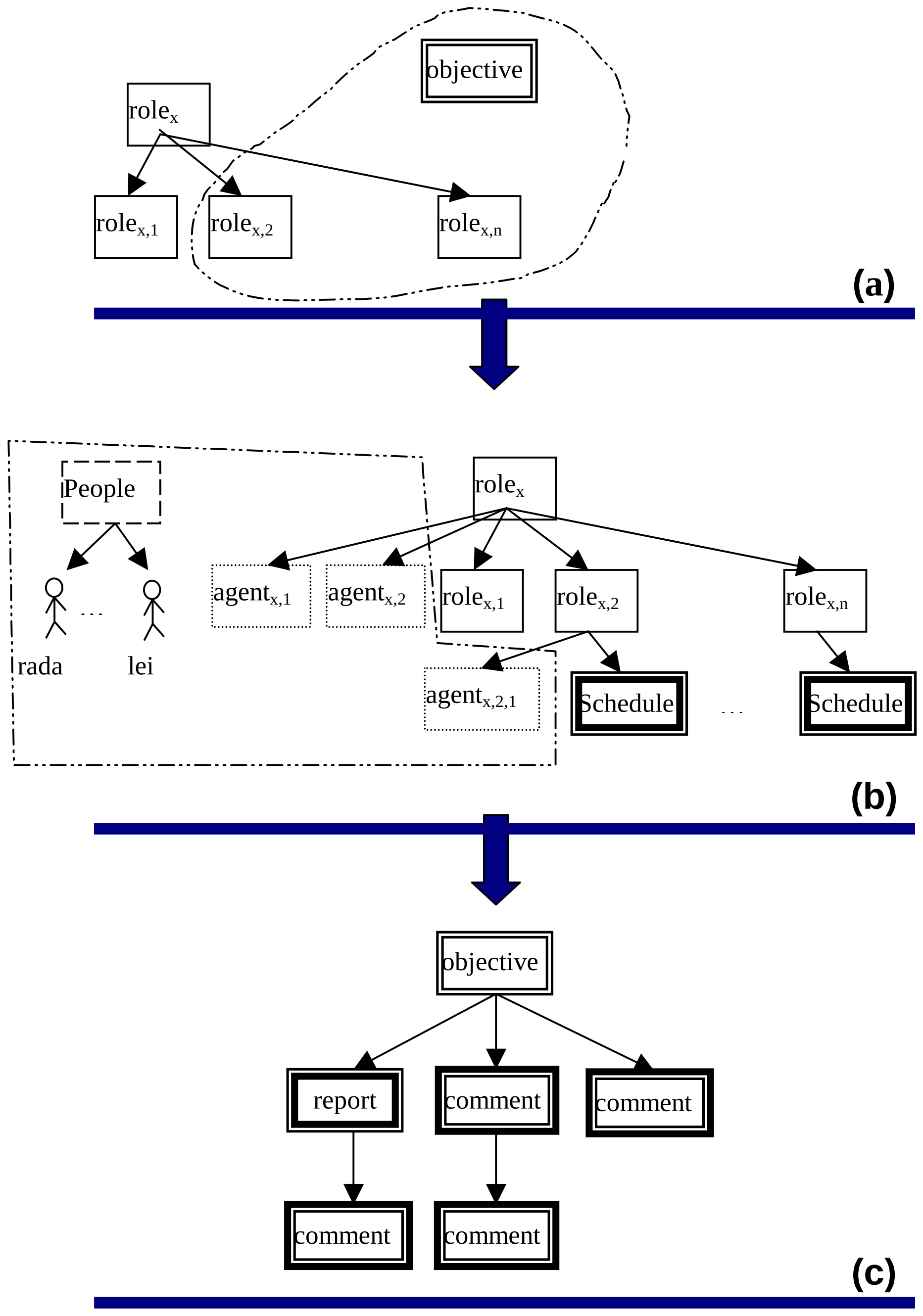
People

- *attributes*: first name, last name, userid, password, phone number, email address
- *constraints*: userid must be unique

Agent

- *attributes:* people id, status of accepted, applied, withdrew, or rejected
- *constraints*: parent node must be of type 'role';

Blob (represents a repository of node content)



- *attribute*: id, arbitrary multimedia object

Figure 2. (a) Role hierarchy and an objective node that role$_x$ has the opportunity to assign to subordinate role(s). (b) "Scheduling": Role$_x$ has created a schedule for role $_{x,2}$, …., role $_{x,n.}$ This involves first connecting an objective with some subordinate roles and associating a deadline with them. This is then stored as a schedule node for each role. (c) An agent in any role $_{x,2}$, …., role $_{x,n}$ may respond to the assigned objective by creating a progress report or comment. The supervisor role may also comment on that agent's response.

Comment
- *inherits*: all attributes of Blob
- *attribute*: score

Any non-guest user viewing any node has the option to generate schedule node(s) that take that node as the objective. The schedule nodes will be added beneath all subordinate roles that the assignee selects (see Figure 2b). The assignee will specify type of response, start date, and due date.

The system can be initialized to suit different organizations. The system developer initializes the information space with the following:

- *Roles* of Guest and Chief,
- *Blobs* with a Help Manual,
- *Schedule* nodes for Guest and for Chief to see Help Manual, and
- *Schedule* nodes for Chief to create other nodes.

The Chief then defines various roles, and users subsequently discharge the responsibilities of the roles.

2.3 Implementation and Functionality

MUCH2000 has been implemented with Active Server Pages on a Microsoft Internet Information Server (see Figure 3). The backend is a relational database management system, Microsoft SQL Server. The front end is a web browser that is Javascript enabled. The hierarchical outline may consist of any number of levels and contain any number of nodes. A fold/unfold outline has been implemented using Remote Scripting which allows the browser to call scripts on the server (in response to events such as unfold a node), receive data, and update parts of the page without having to reload the page.

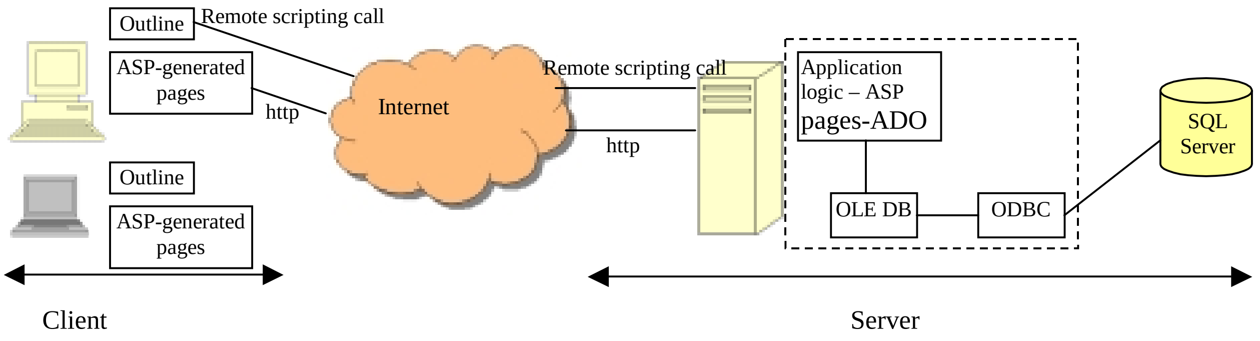


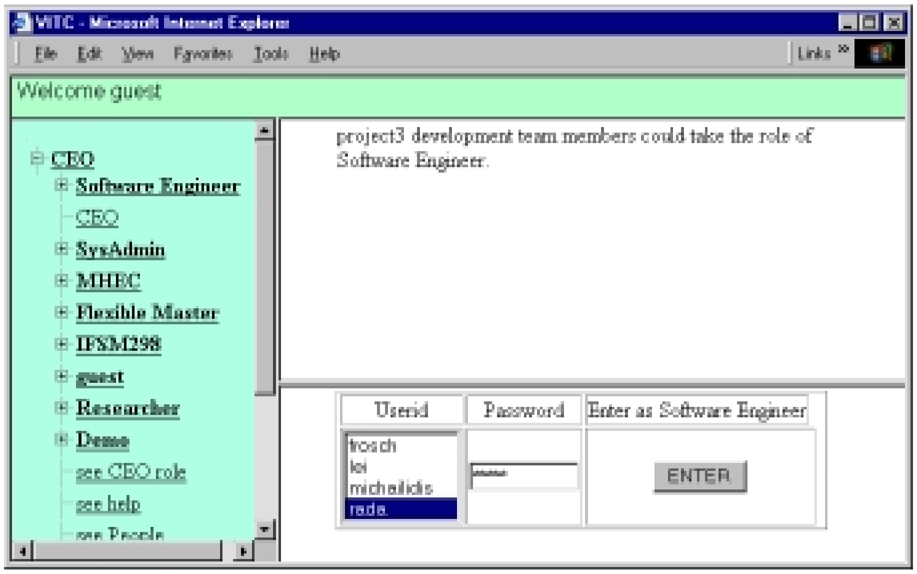
Figure 3. Client-Server Architecture: Any number of users interact directly with the Internet Information Server (IIS) via web browsers. The Active Server Pages on the IIS communicate with the SQL Server.

The interface is modeled after the SuperBook interface and shows a fold/unfold outline on one side and details on another side (Furnas, 1986; Egan et al, 1989). The interface has 4 frames: top, table_of_contents, details, and reply (see Figure 4). The top frame contains the name and role of the current user. The fold/unfold outline frame displays the node hierarchy. The attributes of a node appear in the details frame. The reply frame is for creating new nodes connected to the node whose details are being displayed.

The system supports workflow management by enabling users to create and schedule activities for roles and apply and assume appropriate roles. Knowledge management is supported by linking all relevant activities and the products of each activity (e.g. progress reports) to the relevant roles.

A registered person who has applied and been accepted into a role can subsequently enter the system in that role. The user browses the role hierarchy that begins with the node labeled 'CEO' and finds the user's relevant role and on selecting that role is given the option to enter it. See Figure 4 for an example of 'Rada' entering as 'Software Engineer'.

Having entered in a role, what a role player sees is different from what a guest sees. In Figure 5, the user sees a top frame that now includes options to see tasks, to customize the system, and to search the site. Also the detail frame now contains a list of scheduled activities for the 'Software Engineer'. In this case the options include that of reading various nodes. Unlike the guest role, this ‘software engineer’ role now sees in the Table of Contents a node called ‘Software Engineering’. Upon unfolding this ‘Software Engineering’ node, the user would see information germane to the activities of software engineering in this organization.


VITC - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Links
ENTER

Welcome guest

- CEO
  - Software Engineer
  - CEO
  - SysAdmin
  - MHEC
  - Flexible Master
  - IFSM298
  - guest
  - Researcher
  - Demo
  - see CEO role
  - see help
  - see People

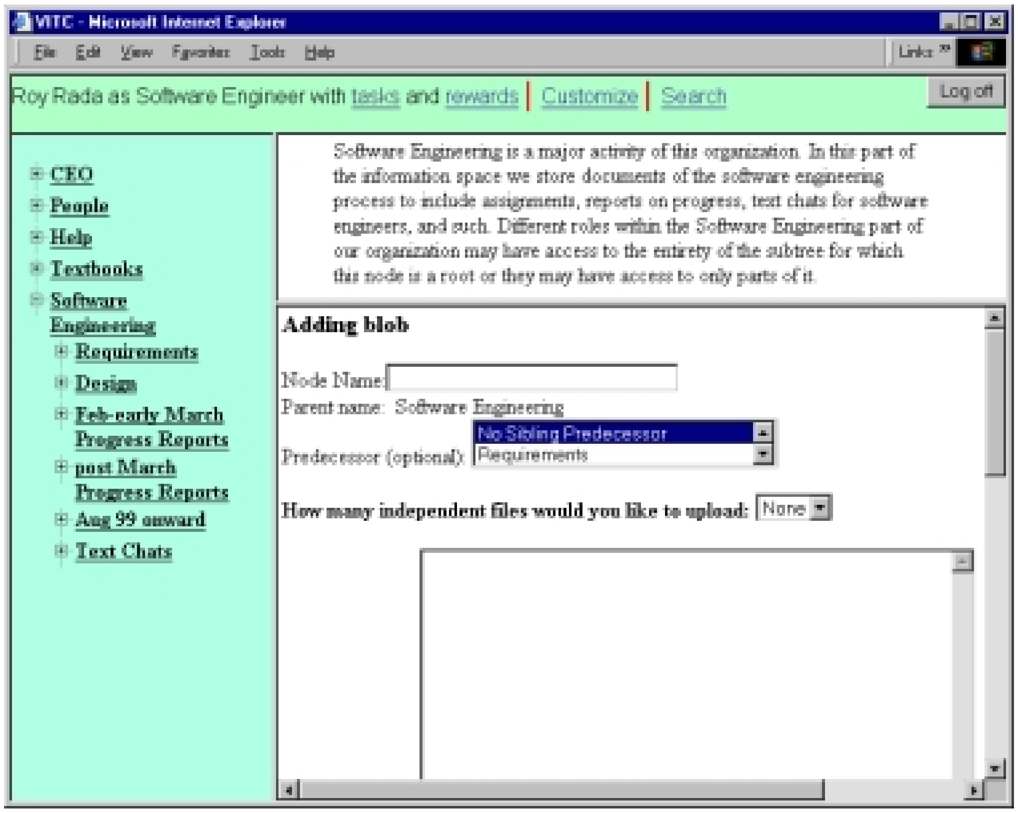
project3 development team members could take the role of Software Engineer.

Userid	Password	Enter as Software Engineer
trosch lei michailidis rada	*****	ENTER

Figure 4. “Enter Role”. The user has unfolded the role hierarchy on the left and selected the role of ‘Software Engineer’ on the left and is attempting now to enter in that role by selecting her/his userid and entering her/his password.

Only a user assigned to create a blob can create a blob. The relevant screen for the ‘Software Engineer’ is shown in Figure 5. In that figure the user is asked to create a node name, to specify the predecessor among the sibling nodes, and to enter text directly or to upload any pre-existing file in any format.

After entering the information required to completely specify a new node in the system, the author has the option to email this information to anyone in the organization. The email options specify by hierarchical position (supervisor, supervisee, peer, or other) the names of people that the author may choose as addressees of the email. The email form has options for the To, CC, and BCC fields. The message that is sent automatically includes the sender’s email address, the path from the root of the information space to this new node, and the node attributes.


VITC - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Links
Log off

Roy Rada as Software Engineer with tasks and rewards | Customize | Search

- CEO
- People
- Help
- Textbooks
- Software Engineering
  - Requirements
  - Design
  - Feb-early March Progress Reports
  - post March Progress Reports
  - Aug 99 onward
  - Text Chats

Software Engineering is a major activity of this organization. In this part of the information space we store documents of the software engineering process to include assignments, reports on progress, text chats for software engineers, and such. Different roles within the Software Engineering part of our organization may have access to the entirety of the subtree for which this node is a root or they may have access to only parts of it.

Adding blob

Node Name:

Parent name: Software Engineering

Predecessor (optional): No Sibling Predecessor / Requirements

How many independent files would you like to upload: None

Figure 5. *Software Engineer adds Blob*. The Software Engineer was scheduled to produce a node and is in the process of doing that. The first blanks of the form are shown here. At the bottom of this form the email option would appear.

The system supports a number of other functions. For instance, it supports synchronous communication in a chat room. This chat room is one instance of adding blobs under a parent node. When a short time frame is given for several people to enter short blobs, then a text chat session results.

3 Case Studies

Two case studies are presented that explore the utility of MUCH2000. The software engineering case involved a handful of people working in truly virtual mode for a long time. The classroom case for MUCH2000 was shorter lived because another tool proved more practical to use.

3.1 Software Engineering

A handful of people developed MUCH2000 over the past two years and used the system to support their work. How have they experienced MUCH2000 as a knowledge management tool?

Five people formed the core of the software engineering team. These five people worked in widely geographically distributed locations with one each in the West Coast USA, Mid-West USA, and Germany and two in the East Coast USA. The team never met face-to-face. Text chat sessions were held every Wednesday and Saturday at noon Eastern Standard Time on the MUCH system.

Microsoft Visual Internet Development Studio was used to develop the software that has resided on various Microsoft NT Servers. The MUCH system is not the authoring system for the code itself but is for all the other documents of the software life cycle.

After a person enters MUCH2000 as a software engineer, the person sees a subtree of nodes rooted under the node 'Software Engineering'. On unfolding that node in September 1999, the person sees the nodes (see Figure 5):

- Requirements
- Design
- Feb-early March Progress Reports
- post March Progress Reports
- Aug 99 onward
- Text Chats

The early nodes for progress reports prior to February were archived. When a person unfolds the post-March progress report, the person sees a list of subtrees for each role that contains:

- SE Access
- SE Admin
- SE Reward
- User Help
- SE Communicate
- SE (Chief)

This one-to-one mapping of progress report subtrees to roles facilitates management of the knowledge in the system. People in these software engineering roles were regularly reporting on progress for their activities.

To better understand the effort of the software engineers, the nodes created by the software engineering team between January 1, 1999 and July 27, 1999 are analyzed. Over 3,000 nodes were created (see Table 1).

The breakdown by type of node shows that chat nodes dominated the activity (see Table 1). Each contribution by a chat participant is a node -- just as any other contribution in the system is a node. Some of these chat nodes would be as simple as "Hello". 40 chat sessions were defined and scheduled, and 2831 contributions to chat occurred. An average of 71 nodes were created during each chat session. These chat sessions were valuable for the continuity of the team both work-wise and socially. If a person missed the scheduled time of a text chat, the person might visit the chat session later and review what had been said. A person can be accepted into more than one role, and more than one person can assume a role:

Node Type	Number
Blob	195
Schedule	290
Chat session	40
Contribution in chat session	2831
Objective	34
Report on progress towards objective	21
Role	6
Comment	23
Person	5
Agent	12
TOTAL	3457

Table 1: Nodes Created by Software Engineers. The node type is in the left column and the number of nodes of that type in the database is in the right column. The 195 'blob' nodes account for such entries as the 'Requirements' node. When blob or other nodes outside of text chat nodes were added to the system, they were typically also sent by email to the relevant people.

As a tool for collecting information, organizing it, and making decisions about how to achieve organizational objectives, MUCH2000 was extensively used by the software engineering team. It supported knowledge management.

Workflow management focuses on the scheduling of activities to achieve the organizational mission. What does the software engineering team data show with respect to workflow management?

The software engineers did regularly use the system to store information and to communicate. However, did they work to schedules in the system:

- People were able to create some nodes in a free-form way without necessarily earning credit for that. This would be the kind of effort reflective of a brainstorming session or informal discussion. 195 nodes were created in this mode.
- 24 schedule nodes gave 6 software engineering roles read access to 4 subtrees. The system did not track how often people read these nodes.
- 40 chat sessions were scheduled for 6 roles. 240 schedule nodes existed to account for each role being expected to visit each chat session.
- Only 26 schedule nodes were created that asked a software engineer to create a specific report in response to a specific objective within a few days and for non-zero credit. Only 20 nodes were responses to such assignments.

For multiple roles over multiple months this 20-report response was less than expected. The team did not have as regular a rhythm of weekly objectives and progress reports as originally expected. One might have expected for 6 roles for half a year to have 100 scheduled progress reports and 100 responses to this schedule.

Rather than using the “Report on progress towards objective” nodes, the team did in fact report progress in the chat meetings held twice a week. Those meetings consisted of the following activities:

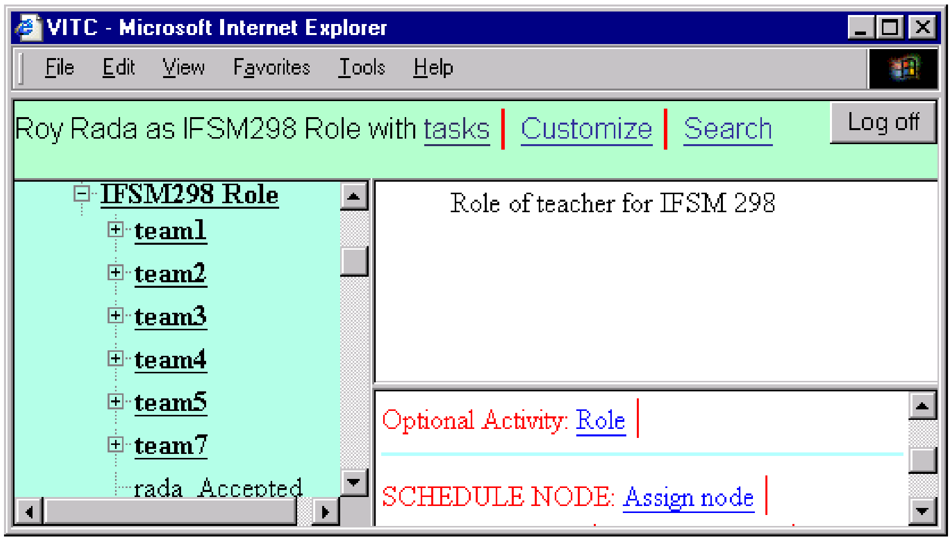
- brief progress reports,
- evaluation of progress reports by the group leader,
- decision-making,
- planning for future work, and
- informal discussions.

The text chat served as a substitute for the scheduled report mechanisms and as a shared workspace for posting progress updates and helped maintaining and coordinating the collaboration over a period of many months.

3.2 Classroom

The same MUCH2000 used for the software engineers was introduced into a class taught in the fall of 1999. The topic of the class was Visual Basic (the class code was ‘IFSM 298’), and the students worked in teams to build software. Seven teams were created and students were expected to self-organize into these teams and to create subroles for members of their team. One can unfold the role hierarchy from the CEO position and see the ‘IFSM 298’ role that is associated with the role of leader of ‘IFSM 298’, namely the teacher. Under this ‘IFSM 298’ role is the role for the leaders of the seven, student teams (see Figure 6).

Under the ‘Information’ node is another node for ‘IFSM 298’. Under this node are various other nodes, such as a ‘syllabus’ node. The syllabus node itself contains the initial syllabus of September 2nd as a Microsoft Word document, and the children underneath it are updates to the syllabus for the days after September 2nd (see Figure 7).


VITC - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Log off

Roy Rada as IFSM298 Role with tasks | Customize | Search

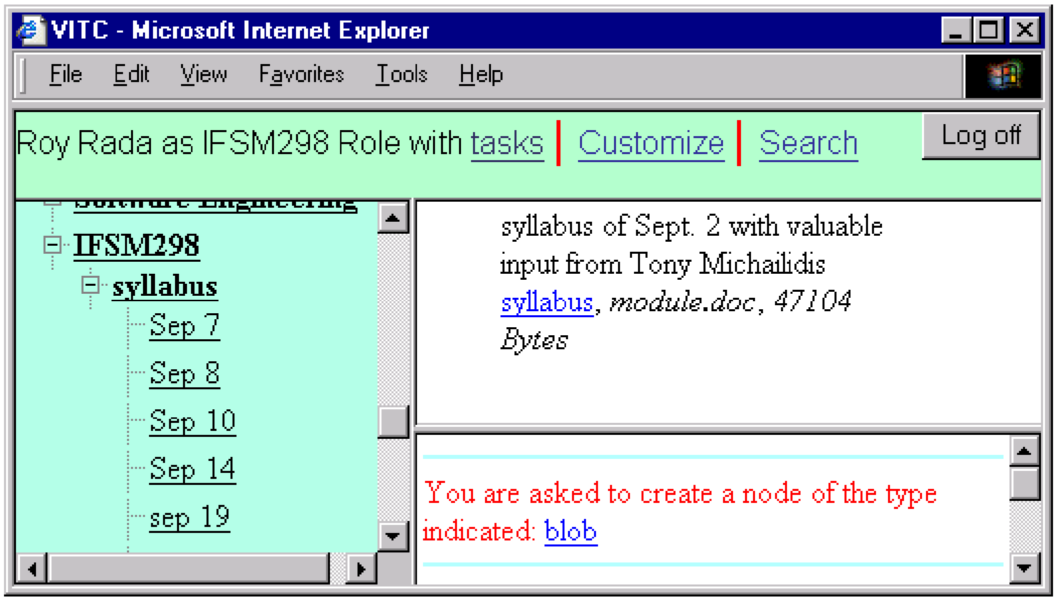
IFSM298 Role
team1
team2
team3
team4
team5
team7
rada Accepted

Role of teacher for IFSM 298

Optional Activity: Role

SCHEDULE NODE: Assign node

Figure 6. The teacher is in the IFSM 298 role. Seven teams were created to report to the teacher during the running of the course.


VITC - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Log off

Roy Rada as IFSM298 Role with tasks | Customize | Search

Software Engineering
IFSM298
syllabus
Sep 7
Sep 8
Sep 10
Sep 14
sep 19

syllabus of Sept. 2 with valuable input from Tony Michailidis
syllabus, *module.doc, 47104 Bytes*

You are asked to create a node of the type indicated: blob

Figure 7. The IFSM 298 syllabus was stored as a Microsoft Word document entitled module.doc. The nodes indented underneath 'syllabus' are the syllabus updates that were issued from day to day.

The students were not comfortable using MUCH2000 for several reasons. For one, they felt challenged to learn Visual Basic and did not want to learn how to use other software, such as MUCH. Also the students did not want the responsibility of creating roles in teams and managing the work of other students; instead they preferred a more traditional mode of operation. The teacher was not able to make a way of working clear to the students that they found comfortable and that at the same time would operate in MUCH2000.

In the spring of 2000, the teacher adopted a different tool, called CourseInfo, for supporting a new class. CourseInfo from Blackboard Corporation (www.blackboard.com) has relatively simple features compared to MUCH2000. Basically two roles are supported -- teach and student. As in MUCH2000, the interface relies on a split screen with the outline in the left and the details on the right. However the contents of the outline on the left are fixed and simple and include nodes such as 'Course Information' and 'Staff Information' that are clearly tailored to the classroom situation. As with MUCH2000, there can be assignments and students submit their assignments. For the purposes of teaching a class, the simplicity of CourseInfo seems superior to the complexity of MUCH2000.

4 Related Work

The vision of hypertext as an infrastructure for an organization to include all aspects of the functionality of the organization has been in some people's minds for a long time (Rada, 1991). The diffusion of information technology generally has made the realization of such visions increasingly practical. Selvin (1999) describes a hypertext system that supports team-based software design in a commercial organization. Verbyla and Watters (1999) also argue for the use of a hypertext infrastructure to support organizational activities in ways similar to those of the Workflow Management Consortium (WfMC). Whereas the WfMC focuses on workflow engines and process descriptions, Verbyla and Watters address the common information infrastructure that must underlie the organization. They advocate discrete components that communicate with one another in a fashion not unlike that of the WfMC.

A hypertext infrastructure has been used to support some aspects of workflow and knowledge management (Wang and Haake, 1998):

> Ordinary hypermedia links, nodes, pages, and media objects are application data, which are application specific and usually not managed by a workflow system. However, in this model they are represented in a unified hypertext model, and can therefore be managed by a single system. The distinction of task nodes and process links from ordinary hypermedia nodes and links is reflected in the computational semantics attached to them.

The semantics adopted by Wang and Haake reflect the Activity Modeling Environment semantics of Smith et al (1989). Namely, there is an emphasis on modeling agents, roles, activities, and a shared information space. The Wang and Haake model does not take into account credits earned or quality control more specifically. MUCH2000 intimately links monitoring into the architecture of processes and the output of those processes.

Building on the cooperative hypermedia infrastructure, Wang (1999) subsequently focused on role-based access control. Access permissions are assigned based on the types or the instances of hypermedia objects. Operations upon hypermedia objects are classified into four categories: Query, Update, Execute, and Assign. Noll and Scacchi (1999) present a hypertext-based environment for supporting virtual software teams. Their hypertext functionality supports collaborative data sharing, integrates existing tools and environments, and enacts software processes to coordinate development activities for teams across wide-area networks. The system is open to other tools when the other tools adopt the language specific to this hypertext-based environment. Access control is specified more at the traditional file control level than at the role-based access control level.

Prinz and Syri (1997) investigated workflow system support for knowledge management and in particular support for workflow process knowledge. They studied how a workflow system can be used to support transitions from tacit to explicit knowledge about how to perform a process (i.e. the sequence of steps and the flow of information from one step to another) in a ministerial environment. They showed how formal workflow process definition can be extended with shared workspace tools to support informal group work, ad hoc workflows, and communication among users. The combinations of the two tools can support a combination of transitions between tacit and explicit knowledge. MUCH2000 supports the specification of a workflow route using inheritance but it does not provide shared workspaces.

Simone and Divitini (1997) argued that both formalized workflow and shared workspaces constitute rather extreme approaches to managing work and knowledge that do not match divergent and evolving user abilities and interests. Instead they argue that more flexibility is needed. They proposed a mechanism for specifying and executing work process and for learning about those work processes. Users can then improve existing processes or design new processes based on how knowledgeable they become about the work they perform. MUCH2000 supports flexibility by encapsulating work schedules within role definitions. Roles correspond to job functions and are dynamically defined through their associations with the tasks they are assigned to perform. Users can learn about role responsibilities by browsing the list of assigned activities. This approach is similar to (Shum, 1997) who explored hypertext representations to link together activities and products of those activities to the relevant roles.

5 Conclusion

MUCH2000 is a hierarchical hypertext system with different node types. These types include people, roles, schedules, comments, and blobs. A hierarchical backbone supports both a command structure for roles and role-based or schedule-based access control. The interface supports overview first and detail second and logically displays node attributes.

The system has been successfully implemented with a web interface and relational database backend. The database is initialized to support a generic organization. Then individuals should use the system to instantiate specific roles and objectives of their team.

MUCH2000 served as a knowledge management tool for a small team of software engineers. The accumulation of nodes in the database shows a predominance of chat nodes. This reflected the importance of synchronous meetings to keep the team of geographically distributed people together. Freely created blobs were next in frequency, and a smaller fraction of scheduled weekly progress reports occurred. Without paper, face-to-face, or telephone modes of communication or information sharing, MUCH2000 provided the information and knowledge infrastructure of the team.

To move from knowledge management to workflow management requires a further emphasis on work performed to schedule. The software engineering team did not strictly adhere to workflow management, as it could be supported by MUCH2000. Diffusion theory emphasizes that a new information system should fit gracefully into people's way of working (Surry and Farquhar, 1997). This means more particularly that people should be able to try the new system in a small way and see clear gains. People should be comfortable to continue using their current tools for much of their work, while they get comfortable with the new tool.

To make MUCH2000 more valuable to its users the connectedness to their other tools and ways of working is necessary. This increased connectivity is being pursued in new technical and social features. Technically, XML is being explored as the markup for the importing and exporting of information. In this way, if people want another system of which they are fond to be able to share information with MUCH2000 the protocols for information exchange should be easier to specify. Additionally, connectivity to Microsoft Office 2000 is being specifically explored. For instance, the email module could invoke Microsoft Outlook 2000 and users could read and send email that both connected with their regular email activity and also stored and retrieved information from MUCH2000 as appropriate.

For social features important to success, the system is being extended so as to more clearly relate the hierarchy of objectives to one another in systematic ways and to represent monetary transactions. Agents will negotiate the performance of a role by agreeing the monetary reward to correspond with successful discharge of the role responsibilities. A high-level role that has certain obligations and monetary assets must be in accord with the low-level roles that report to the high-level role. Thus if a supervisor has n dollars to pay for work to be done and that money goes to immediately subordinate roles and those subordinate roles have no other source of money, then the subordinate role can not in turn agree with a role subordinate to it to give more than n dollars as reward for successfully completed work.

Ultimately, successful diffusion depends on the standardization of an organizational model and compliance by people to this standard. Once such standardization occurs, further automation of the activities of the organization become practical. MUCH2000 architecture and experiences with its usage suggest how a hypertext infrastructure can be augmented with workflow functionality so as to facilitate the standardization of organizational processes.

6 References

Beckman, Thomas (1999) "The Current State of Knowledge Management" in *Knowledge Management Handbook* edited by Jay Liebowitz, published by CRC Press LLC, Boca Raton, Florida, pp 1.1-1.21.

Borghoff, Uwe M. and Pareschi, Remo (1997), "Information Technology for Knowledge Management", *Journal of Universal Computer Science*, vol. 3, no. 8 (1997), 835-842.

Daft, R.L. and Lengel, R.H. (1986), "Organizational Information Requirements, Media Richness, and Structural Design", *Management Science*, Vol. 32, No. 2, pp 554-571.

Egan, Dennis E., Remde, Joel R., Gomez, Louis M., Landauer, Thomas K., Eberhardt, Jennifer, and Lochbum, Carol C. (1989) "Formative design-evaluation of SuperBook", *ACM Transactions on Information Systems*, Vol. 7, No. 1, pp. 30-57.

Furnas, George W., (1986) "Generalized fisheye views", *Proc. CHI86 Conference: Human Factors in Computing Systems,* ACM, New York, NY, pp. 16-23.

Georgakopoulos, Dimitrios and Hornick, Mark (1995), "An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure", *Distributed and Parallel Databases*, Vol. 3, pp. 119-153.

Hollingsworth, David (1995), *The Workflow Reference Model*, Workflow Management Coalition, Document Number TC00-1003, Document Status - Issue 1.1.

Jaeger, Trent, Michalidis, Antonios, and Rada, Roy (1999), "Access Control in a Virtual University", in *4th International Workshop on Enterprise Security of the IEEE Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, occurred in San Francisco, California in June 1999.

Lawrence, Peter editor (1996) *Workflow Handbook 1997*, John Wiley in association with Workflow Management Coalition (WfMC), pp 1-450.

Leonard, Nancy H., Beauvais, Laura L. and Scholl, Richard W. (1997) "Organizational Member Self Concept in the Organizational Adaptation and Change Process", *Annual Meeting of the Academy of Management*.

Malone, Thomas and Crowston, K (1990) "What is Coordination Theory and how can it help design cooperative work systems?" *Proceedings of the Conference on Computer Supported Cooperative Work* (CSCW'90) pp 357-370, ACM Press, New York.

Macintosh, A., Filby, I. and Kingston, John (1999) "Knowledge Management Techniques: Teaching & Dissemination Concepts" *Journal of Human Computer Studies*, September/October 1999.

Malone, T.W.; Crowston, K.; Lee, J.; Pentland, B.; Dellarocas, C.; Wyner, G.; Quimby, J.; Osborn, C.S.; Bernstein, A.; Herman, G.; Klein, M.; O'Donnell, E. (1999) "Tools for inventing organizations: toward a handbook of organizational processes" *Management Science*, Vol.45, No.3, pp. 425-443.

Medina-Mora, Raúl, Winograd, Terry, Flores, Rodrigo and Flores, Fernando (1992) "The Action Workflow Approach to Workflow Management Technology", *Proceeding of the ACM Conference on Computer-Supported Cooperative Work, CSCW'92*, pp. 281-288.

Noll, John and Scacchi, Walt (1999) "Supporting Software Development in Virtual Enterprises" *Journal of Digital Information*, Volume 1, issue 4, http://jodi.ecs.soton.ac.uk/Articles/v01/i04/Noll/.

Papows, Jeff (1998) enterprise.com: market leadership in the information age, Perseus Books, Reading, Massachusetts.

Prinz, Wolfgang and Syri, Anja (1997), "Two complementary tools for the cooperation in a ministerial environment", *Journal of Universal Computer Science*, vol. 3, no. 8 (1997), 843-864.

Rada, Roy (1991) *Hypertext: from Text to Expertext*, McGraw-Hill, London.

Rada, Roy (1997), *Virtual Education Manifesto,* Hypermedia Solutions, Ltd., Liverpool, England.

Selvin, Albert (1999) "Supporting Collaborative Analysis and Design with Hypertext Functionality" *Journal of Digital Information,* Vol. 1, No. 4, http://jodi.ecs.soton.ac.uk/Articles/v01/i04/Selvin/.

Sheth, Amit, Georgakopoulos, Dimitrios, Joosten, Stef M. M., Rusinkiewicz, Marek, Scacchi, Walt, Wilenden, Jack, and Wolf, Alexander (1996), *Report from the NSF Workshop on Workflow and Process Automation in Information Systems*, State Botanical Garden of Georgia, Athens, Georgia, May 8-10.

Simone, Carla and Divitini, Monica (1997), "Ariadne: Supporting Coordination through a Flexible Use of the Knowledge on Work Processes", *Journal of Universal Computer Science*, vol. 3, no. 8 (1997), 865-898.

Smith, H T, Hennessy, P A, Lunt, G. A. (1989) “The activity model environment: an object-oriented framework for describing organizational communication” *Proceedings of first European Computer-Supported Collaborative Work Conference*, pp. 160-172.

Shum, Simon (1997) "Negotiating the Construction and Reconstruction of Organisational Memories" *J.UCS, Vol. 3, No. 8,* pg 899-928.

Surry, Daniel and John Farquhar (1997) "Diffusion Theory and Instructional Technology" *Journal of Instructional Science and Technology*, Vol. 2, No. 1.

Verbyla, J and Watters, C (1999) “Cooperative Hypermedia Management Systems” *Journal of Digital Information*, Vol. 1, No. 4, http://jodi.ecs.soton.ac.uk/Articles/v01/i04/Verbyla/

Wang, Weigang and Rada, Roy (1998), “Structured Hypertext with Domain Semantics”, *ACM Transactions on Information Systems*, Vol. 16, No. 4, pp. 372-412.

Wang, Weigang and Haake, Jorge (1998) “Flexible Coordination with Collaborative Hypermedia” *ACM Proceedings of Hypertext 1998*, occurred in Pittsburgh, Pennsylvania, pp 245-255.

Wang, Weigang (1999) “Team-and-Role-Based Organizational Context and Access Control for Cooperative Hypermedia Environments” *ACM Proceedings of Hypertext 1999.*

Wiig, K. (1997) “Knowledge Management: Where Did it Come From and Where Will It Go?” *Expert Systems with Applications*, Vol. 14.