A METAMODEL FOR MODELING COLLABORATIVE SYSTEMS

I.T. HAWRYSZKIEWYCZ
University of Technology, Sydney
Sydney, NSW 2007

ABSTRACT

Computers are now being increasingly used to support collaborative knowledge intensive processes. These are collaborative processes where intense interaction takes place between team members working towards a common goal. The paper identifies technology as an enabler of the collaborative work practices in such processes. To do this technology must provide ways to set up systems to support collaboration and at the same time assist users to adapt systems to changing work practices. The paper defines a collaborative metamodel to define collaborative work practices and to provide the ontology to define software agents that help users adapt it to changing needs. It then describes a prototype implementation.

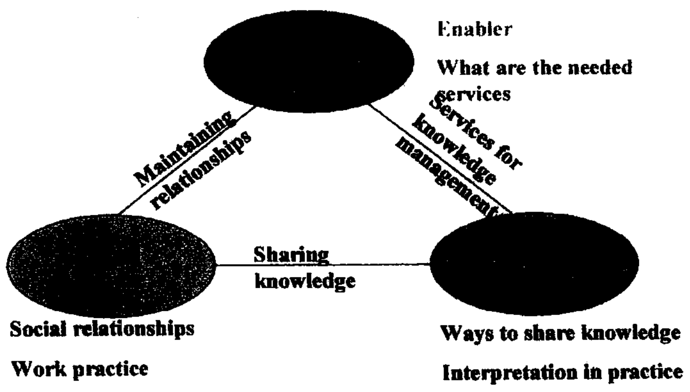
Keywords: Collaboration, Software agents, knowledge intensive processes.

INTRODUCTION

Collaboration is becoming more prominent in the business world with many distributed organizations requiring people to collaborate across distance. The Internet has been suggested and often used as a way to support collaboration. However, studies (9) have shown that collaboration using Internet services such as e-mail usually does not go beyond exchange of documents. Hence the complexity of relationships or knowledge processes that can be supported using Internet services is limited. Knowledge processes require intense exchange of information between team or alliance members to reach some desired outcome and must go beyond simple exchange of messages or documents but operate interactively in a shared context. One generic example of such processes is where tacit knowledge of experts is combined with refined explicit knowledge to create new products and services (11). Another is to characterize them as innovation (22) processes where people combine knowledge from a variety of fields to create new knowledge. Just to name a few applications, there are distributed software project teams (5), design teams, planning and evaluation teams, client support teams, as well as the need for meetings during various stages of the business process. For example, producing a new car model can require well over 200 designers of different components to coordinate their activities and share their knowledge.

The paper premises that successful collaboration requires the combination of the three important dimensions shown in Figure 1. First there is the social culture where people develop the relationships and work practices necessary to share and create knowledge in mutually acceptable ways. Then there are ways to manage organizational knowledge. Knowledge management includes support for interpreting information in its context using the tacit knowledge of participants and the distribution of such interpretations. Technology then becomes the enabler for both maintaining the personal relationships and providing the tools and services to share knowledge in meaningful ways.

FIGURE 1
Dimensions of Collaboration


Enabler
What are the needed services
Maintaining relationships
Services for knowledge management
Sharing knowledge
Social relationships
Work practice
Ways to share knowledge
Interpretation in practice

To be an effective enabler, technology must support the social relationships in a given application and provide people with services to construct knowledge. This paper proposes that technology facilitation requires two parts. One is to select technologies that best support social relationships and maintain such relationships. The other is to assist users to setup and change systems as their work practices change. Suggested ways to do the latter include using software agents (12). Furthermore, such assistance must be provided in general ways rather than specifically for each application. The paper proposes a metamodel of collaboration to provide a foundation of generalized support. The model is used:

- define collaborative processes for any application and implement them as electronic workspaces, and
- provide the ontology that provides a foundation to define agents, which assist users to adapt their system to changing work practices.

The paper then describes an implementation based on open system Java architecture that directly implements the concepts. The implementation allows workspaces to be created in terms of these concepts, and provides the ability to change workspace structures dynamically as the work situation changes. The open architecture provides a way to easily integrate the agents into the system. The paper then shows how the metamodel provides the framework to identify reusable agents and how they can be combined into flexible agent architectures. The reusable agents are modeled on social structures such as assessing situations, arranging plans or allocating responsibilities.

MODELS OF COLLABORATION

Collaboration has been characterized by a variety of social models and implementations that support these models. One way to categorize models is by distinguishing between specialized approaches to model one class of application, and generic approaches that can model any application. From a specialized approach we have early applications such as IBIS (8) that provide a shared resource for decision making. This is followed by more recent systems such as CoGenTex's EMMA, which have a more collaborative model to support decision making. Other examples of specialized models are joint document editing (27), or models of team support in global environments (28).

Generic models can also cover a large range of application areas. Corresponding technologies can be specialized or general. One is coordination models, in particular workflows. Coordination models have emphasized the sequencing of tasks to achieve some precise goal. One of the earliest models was DOMINO (21), which supported office flows over the Internet. Standards for such systems then evolved primarily through the workflow management systems coalition. Other early systems, in particular ConversationBuilder (19) and Oval (25) placed more on community semantics providing objects, views for presentation to users and supporting these users with conversational tools. LOTUS Notes was an early commercial system that supported similar semantics. Later research has concentrated on systems that provide more flexibility in dynamically defining coordination as for example, Ariadne (10), which concentrated on flexible coordination.

Another is community models, and there are also general social models. Community models have emphasized sharing of predominantly explicit knowledge through technologies such as newsgroups. Community models can be open or they can have specific goals. The most common support here is Internet systems such as newsgroups, which supported open communities. Recently more goal oriented support systems such as BCSW have been developed that provide communities with specific tools such as circulation folders, briefcases as well as workflow support.

Social models such as coordination based on speech acts (6) or activity theory (23) have also been used as a basis for developing collaborative systems. The best example of a support system is Action Workflow (33) that is based on speech acts.

Our work over a number of years has been to identify the semantic concepts in these models and combine them into one metamodel, which provides a general set of semantics that provide a platform to support any coordination model. We thus combine community and work process semantics to reflect the current work environment while providing the ability to dynamically change processes. The model also provides a framework for generating agent support systems.

SEMANTICS TO DESCRIBE COLLABORATION

Our goal has been to integrate the semantics of the generic models into one metamodel that can be used to model community, coordination and social models. Current modeling methods cannot be used to do this. Most current conceptual modeling methods concern relatively stable processes and emphasize functions rather then people. In stable processes, the flow of transactions can be predefined and the data structures are stable. Methods such as dataflow diagrams, ER models or object modeling can be used to model such systems. Existing modeling methods have a number of limitations in modeling dynamic systems. Objects in object modeling can be such agencies, as can data objects. However, with object modeling, organizational entities cannot easily adapt their agencies to changing requirements. The model proposed in this paper satisfies three characteristics. These are to:

- provide concepts that map directly to an implementation so that dynamic changes can be made at the interface level,
- design agent based systems that actively support users. The semantics can be used to construct system descriptions in methodologies such as Prometheus (29) that can subsequently lead to specifications of agent networks to support dynamic organizations, and
- provide a basis for managing knowledge by using the ideas of organization computational theory (4) by selecting agencies that result in a subdivision of knowledge into self-contained chunks, which can be readily recombined into emerging organizational structures.

Thus rather than identifying data objects and processes, we identify organizational entities and their agencies. This approach also supports the notions of Rehfeldt and Turowski (31) of organizations supported by interacting agencies. In such organizations people have their agency as can documents. Thus in any interaction, a document agency may possess the knowledge of how best to structure a document whereas a person can use their knowledge to add content following the structure. The new emphasis, however, needs to go beyond simply functional agencies but to include social and process agencies in the model. The agent architecture is defined using the same ontology to interpret collaborative actions and to signal suggested agent actions to the collaborative environment.

Elementary concepts

Figure 2 illustrates some simple concepts that concentrate on elementary work activities. Each concept is represented by an ellipse. Relationships between the concepts are represented by links between the ellipses. The elementary concepts allow users to define activities and workgroups and can support what have been defined earlier as community semantics. The workgroups participate in the activities. Workgroup participants are assigned to roles that can take actions, and view and change selected artifacts. Thus the semantics are that a workgroup is created and this workgroup can create any number of activities and assign responsibility to workgroup members in each activity. Semantic commands here center on creating objects that correspond to the metamodel concepts and the links between them. The commands include:

Create workgroup (workgroup-name=<text>),

Create activity for workgroup (workgroup-name, activity-name=<text>),

Create role in activity (activity-name, role-name=<text>),

Add artifact-to-activity (activity-name, artifact-name = <text>),

Add action-to-activity (action-name, artifact-name = <text>),

Assign role to action (role-name, action-name),

Assign artifact to action (artifact-name, action-name),

Add-participant-to-workgroup (workgroup-name, participant-name=<text>),

Assign-participant-to-role (participant-name = <text>

These commands are implemented in our workspace system, LiveNet, and can be used to create models defined in terms of the semantics. A model using the semantics is shown in Figure 3.

FIGURE 2
Elementary Concepts

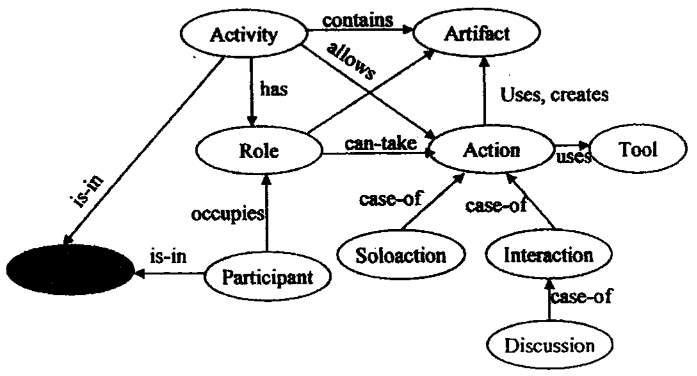


FIGURE 3
A Simple Model

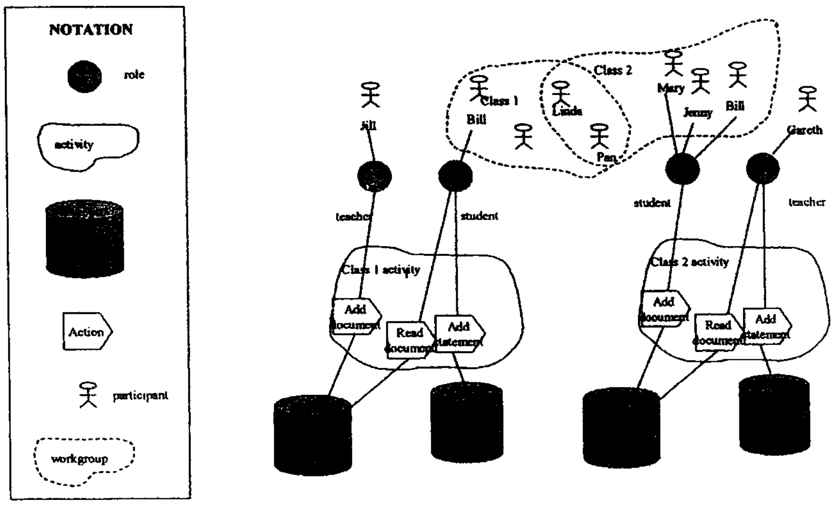


Figure 3 illustrates a model using the concepts shown in Figure 2. The model shows two classes each modeled as an activity, named class 1 activity and class 2 activity. Each class has two roles, teacher and student. Each role has its separate responsibilities. The teacher role can add documents to the activity. The student can read the documents and make comments about them. The participants in each role can be different or they can overlap. Thus there are two participants, Linda and Pan, who participate in each of the two classes. The others take part in one class only.

The system is dynamic in the sense that changes can be made at any time. Thus the semantics allow participants to be created and added to workgroups at any time. Similarly artifacts and actions can be added or deleted to activities as required. Existing roles can be assigned responsibility for these actions. Alternatively new roles can be created.

Extending to groups and work processes

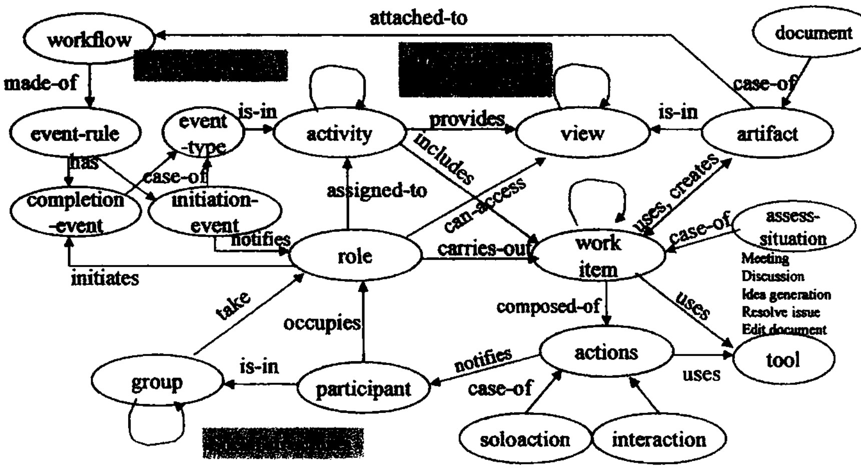
The type of semantics supported by the elementary concepts has a number of drawbacks and the final model includes a number of extensions.

- Create and manage groups independently of activities. This allows teams to be treated as whole and assigned to roles as a group rather than as individuals.
- Introduce work-items within activities to provide an intermediate level to describe organizational work, and
- Introduce events that can be combined into workflows, by distinguishing between completion and initiation events, and specifying workflows by defining the initiation events that are activated by a given completion event.

Figure 4 shows the metamodel that supports the enriched semantics. It now includes a more sophisticated concepts to model communities within organizations. The communities now overlap and can be assigned to roles as groups rather than as individuals. The model also includes events to model workflows thus combining community and coordination semantics. The metamodel shown in Figure 4 combines organizational structures such as activities and work-items, work processes including events and workflows, and social structures that enable groups to be formed and participants to be included in such groups. It provides ways to combine work-items into activities with members of groups assigned responsibilities through roles for those work-items. Furthermore work-items can be grouped in different ways to match the perspective of different users. It supports social interactions, through group formations, discussions or notifications, as well as more structured workflows by associating events with artifacts. The main concepts are:

Artifact - data objects such as documents, calendars. It can also be a record of discussions or other personal interactions.
View - a collection of artifacts. These can be documents, calendars, or multi-media records. They can also be other views.
Activity - produces a well defined artifact as its output (e.g. produce a planning document), and can include many work-items to do so. Provides views necessary to carry out the work-items.
Role - defines responsibilities in system in terms of work-items that it can carry out and the views that it can access.
Participant - a specific person that is in a group and can be assigned to a role.
Group - a collection of participants that can be assigned to a role.
rk-item - a set of actions and interactions needed to produce intermediate outcomes that eventually produce an activity output (e.g. review part of a planning document - which may include a number of actions, assess a situation). A work-item is composed of a number of actions and provides tools to carry out the actions.
Action - a specific unit of work carried out by a role (e.g. change an artifact, send an artifact). Can notify selected roles when completed. Can be a:
Soloaction - carried out by one participant, or
Interaction - the basic exchanges between people when they collaborate in the activities.
nt type - is in an activity and can either be an 'initiation-event' that notifies a role in the activity to carry out work-item, or is a 'completion-event' initiated by a role following completion of a work-item.
nt-rule - defines the next initiation-event or events to be activated following the completion-event.
rkflow - describes a sequence of event rules. Can be attached to an artifact.

FIGURE 4
Collaborative Metamodel



The additional semantics now are:
Assign group to role (workgroup-name, role-name),
Add work-item in activity (work-item-name, activity-name),
Add action to work-item (action-name, work-item-name),
Create initiation-event-in-activity (event-name=<text>, role-name-to-cause)
Create initiation-event-in-activity (event-name=,text., role-name-to-be-notified))
Create event rule (initiation-event, completion-event)

The effect of the introduction of groups is shown in Figure 5.

FIGURE 5
Another Example

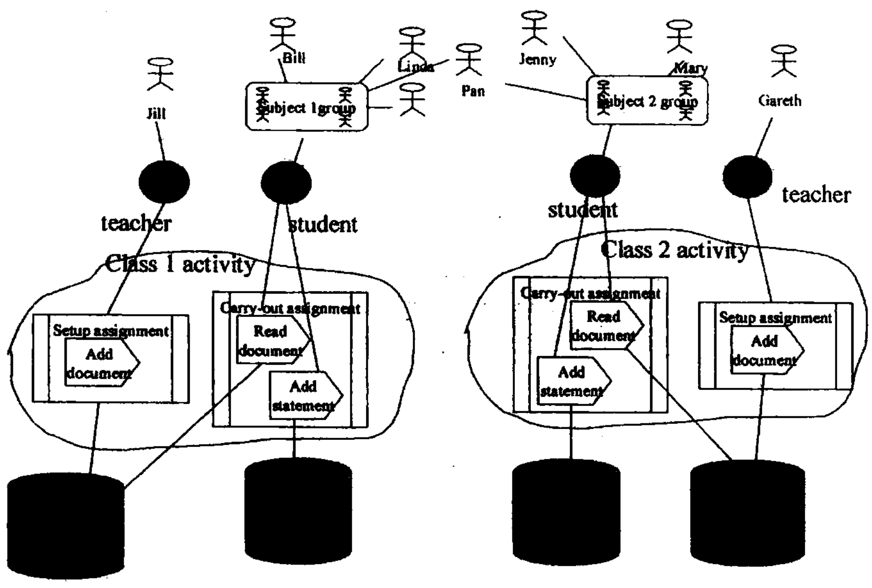
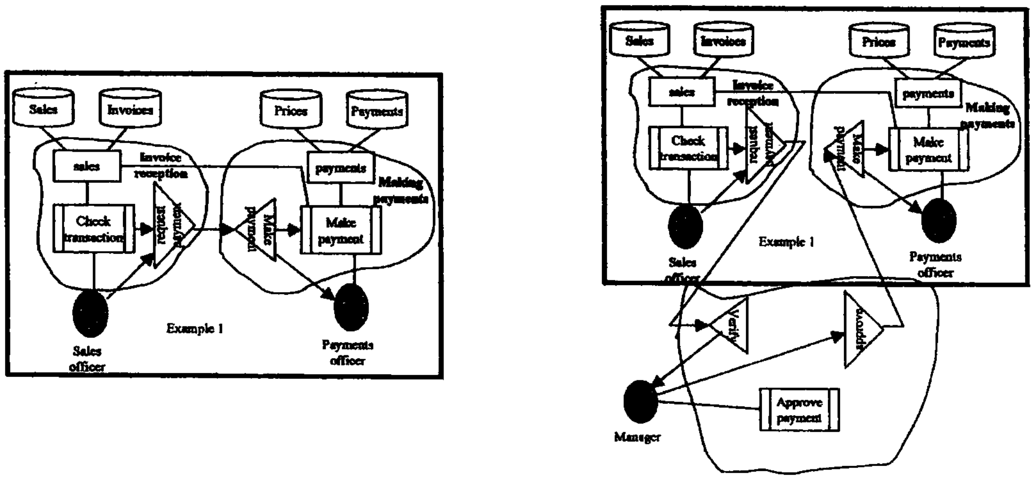


Figure 5 also illustrates the complete notation. Some symbols correspond to those in other methodologies such as Prometheus as the goal is to extend to agent systems. Others are specific to the metamodel. Thus we use a loose cloud notation for activities as we see activities not being fixed but evolving over time through creation of new work-items or roles as needed. The loose cloudy shape portrays the fact that new activities can be easily formed by grouping existing work-items. The event structure is also different to make a distinction between completion and initiation events and also to distinguish from agent events that are part of agent architectures.

Work process evolution

Work evolution can be managed by executing metamodel commands dynamically as work practices change. The left hand side of Figure 6 is a simple example. This shows the use of events to model workflows.

FIGURE 6
Modeling Emergent Workflows



The model in Figure 6 includes two activities.

- invoice reception that has one view, sales that includes sales and invoices. One role, 'sales officer', carries out the 'work transaction' work-item to check invoices against sales. Once checked event 'request payment' is initiated. This activates event 'make payment' in the make payments activity.
- make payments, where the role 'payments officer' is notified by event 'make payment' to make a payment.

The right hand side shows the ability to change the system using simple operations. This is to extend the system to include an approval activity and thus model dynamic changes to workflows. It requires elementary commands to:

- Create a new activity ('approval'),
- Create a work item ('approve-payment') in that activity,
- Create a new role of 'manager',
- Delete event rule,
- Create new initiation event,
- Create new completion event,
- Create new workflow rule.

EXAMPLES OF PRACTICAL APPLICATION

The metamodel semantics can be a tool used in analysis or can serve as a way of choosing agencies with methodologies such as Prometheus (29) or form a basis for choosing agents in AgentUML (3). A typical set of steps followed to create a model is:

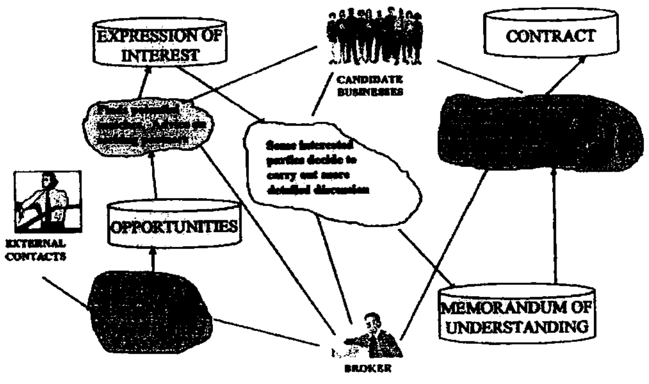
1. Define the high level activities in the system and the work-items in each activity.
2. Define groups or teams and add participants to the groups.
3. Add artifacts needed by the work-items and create views for the work-items to identify the artifacts needed by the work-item.
4. Define roles and assign responsibilities for work-items to roles and then assign participants to the roles.
5. Expand work-items in terms of their actions and identify views for these to 'use or create' artifacts.
6. Identify any predefined events and consequent rules and specify the 'initiation-events; to be activated following a 'completion-event'. Workflows are 'made of' that specify sequences of event rules.

This process can be followed to describe a system in terms of the concepts shown in Figure 4. The system model uses a notation that borrows from a number of such methodologies but makes a clear distinction between organizational entities and agents. The paper now describes the notation and illustrates the model using some examples. The metamodel described here has evolved through a variety of applications such as business networking (13), strategic planning (14) and administrative systems (15). The metamodel concepts can be used to develop high level conceptual models of applications. These models are then used to design technical implementations.

Example – Business Networks

A model of business network formation is illustrated in Figure 7. Here brokers are contracted to facilitate the formation of business networks.

FIGURE 7
Conceptual Model of Business Networks


EXPRESSION OF INTEREST
CANDIDATE BUSINESSES
CONTRACT
Some interested parties decide to carry out more detailed discussion
EXTERNAL CONTACTS
OPPORTUNITIES
MEMORANDUM OF UNDERSTANDING
BROKER

There are four activities in Figure 7, namely:

- Identify business opportunities where external contacts notify brokers of business opportunities,
- Find potential businesses where brokers identify some of their clients, who may be willing to form a network to respond to the opportunity,
- Prepare a memorandum understanding with these candidate businesses, and
- Form a contract.

Figure 7 was the first step in a study to design support systems for supporting business network formation. The strategy was to develop a contact service for brokers and a portal service to post opportunities, and identify brokers and business to respond to these opportunities.

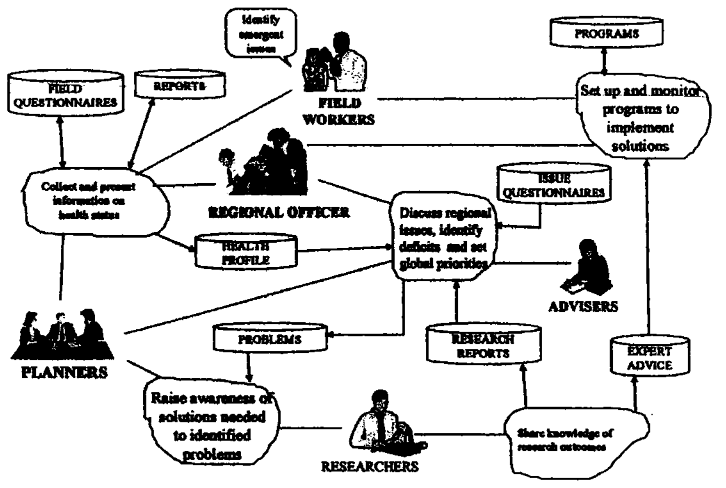
Example – Global Planning

Another example is shown in Figure 8. The field officers here collect information about the health status in the localities. This is summarized at the regional office and then made available for identifying problems and raising awareness of these problems with researchers. The activities in Figure 8 are:

- Collecting health information of the different regions to identify regional health deficits,
- Discuss regional issues to identify global issues and set priorities in terms of world health problems,
- Raise awareness of world problems through publications and public awareness programs,
- Initiate research programs to solve problems by seeking out grant bodies and interested researchers and research institutes,
- Set up and monitor projects to implement solutions.

In both of the above cases the model was used to set strategic directions and define projects to develop systems that support these directions. They can also be used at an applications level as described in the following.

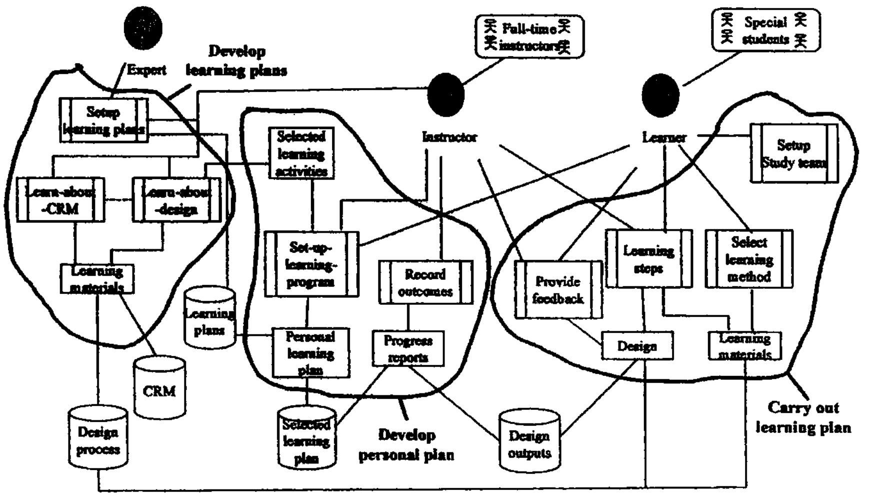
FIGURE 8
Conceptual Model of Global Planning


Identify emergent issues
FIELD QUESTIONNAIRES
REPORTS
FIELD WORKERS
PROGRAMS
Set up and monitor programs to implement solutions
Collect and present information on health status
REGIONAL OFFICER
ISSUE QUESTIONNAIRES
HEALTH PROFILE
Discuss regional issues, identify deficits and set global priorities
ADVISERS
PLANNERS
PROBLEMS
RESEARCH REPORTS
EXPERT ADVICE
Raise awareness of solutions needed to identified problems
RESEARCHERS
Share knowledge of research outcomes

Modeling Personalized Learning

Figure 9 shows a conceptual model that describes a personalized learning environment. The basic idea of the system is that a learner looks at the things that they need to do and identifies their learning goal.

FIGURE 9
Conceptual Model of a Learning Process



The main activities here are:

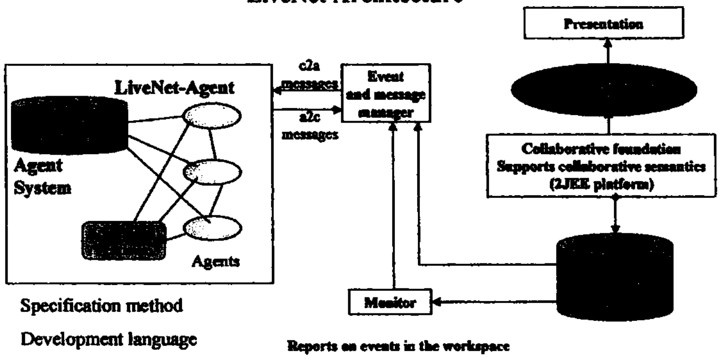
- **Develop learning plans**, which is an activity where instructors develop learning material and setup learning plans. They can invite experts or other people to assist and review their learning plans.
- **Develop personal plans** for learners. Here the instructor together with the learner set up a learning program that includes a personalized learning plan. It also includes ways to assess and record student learning outcomes.
- **Carry out learning plan**, which is an activity where the student follows the learning plan. It helps students to select the best learning method for their plan and carry out any learning tasks.

The work-items can be expanded in terms of their actions and the activities can themselves be redrawn to show different perspectives. The metamodel concepts can be directly implemented in our LiveNet system.

SYSTEM ARCHITECTURE

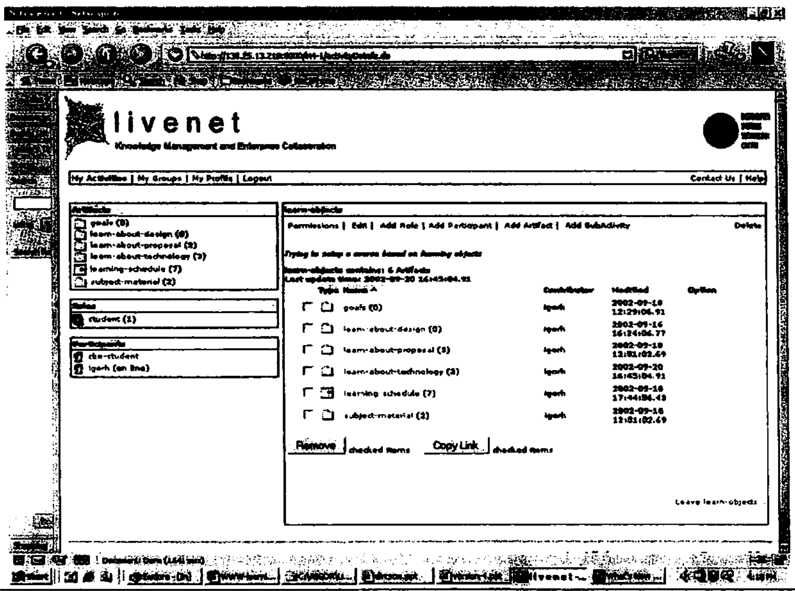
In the implementation, activities become workspaces while the other concepts become components of the workspace. The overall architecture to implement the metamodel is illustrated in Figure 10. The foundation stores a collaborative application in a relational database in terms of the metamodel concepts. Thus there us a table for each activity and other tables that describe the components of each activity. The basic operations become sequences of commands that transform the database. The foundation basically supports the collaborative semantics, which are provided through a 2JEE platform. Interfaces are generated through a JSP mapping. The database structure is not directly related to the interface and alternate interface mappings can be generated from the collaborative application.

FIGURE 10
LiveNet Architecture


Presentation
LiveNet-Agent
c2a messages
a2c messages
Event and message manager
Collaborative foundation Supports collaborative semantics (2JEE platform)
Agent System
Agents
Monitor
Specification method
Development language
Reports on events in the workspace

There are a number of options of presenting conceptual models in a technical implementations. The simplest is to present the concepts as a generalized interface to users. Each activity in that case becomes an electronic workspace. An example of such presentation is illustrated in Figure 11. This shows a "carry out learning plan" activity implemented as a LiveNet workspace showing the different learning steps, including "learn-about-design", "learn -about-proposals", and so on. In addition it includes supporting artifacts such as a learning schedule.

FIGURE 11
A Workspace for Learning


livenet
Knowledge Management and Enterprise Collaboration
My Activities | My Groups | My Profile | Logout
Contact Us | Help
Permissions | Edit | Add Role | Add Participant | Add Artifact | Add SubActivity
Delete
Remove
Copy Link
Leave learn-objects

Artifacts
goals (0)
learn-about-design (0)
learn-about-proposal (3)
learn-about-technology (3)
learning-schedule (7)
subject-material (2)

Roles
student (1)

Participants
cba-student
igorh (on line)

learn-objects

Trying to setup a course based on learning objects

learn-objects contains: 6 Artifacts
Last update time: 2002-09-20 16:45:04.91

Type	Name ^	Contributor	Modified	Option
	goals (0)	igorh	2002-09-18 12:29:06.91	
	learn-about-design (0)	igorh	2002-09-16 16:24:06.77	
	learn-about-proposal (3)	igorh	2002-09-18 12:51:02.69	
	learn-about-technology (3)	igorh	2002-09-20 16:45:04.91	
	learning schedule (7)	igorh	2002-09-18 17:44:56.43	
	subject-material (2)	igorh	2002-09-18 12:51:02.69	

checked items
checked items

An interface based on generalized semantics places the additional burden of users having to map their perception of the work onto workgroup semantics. User must first of all see their problem in terms of the generalized semantics and express it at the interface in terms of these semantics. Thus although the system is generalized in the functional sense and can be easily customized to a particular application, the customization still requires users to think in ways to map their problem semantics to generalized semantics when using the system, thus placing barriers in its use. Alternatively a specialized interface can be provided where each concept is mapped onto a concept natural to a user problem domain. In general this maps each metamodel to a specific user domain concept. Alternatively the metamodel can be enhanced by directly representing derived relationships at the user interface. The paper now illustrates two examples of such representations.

Often the process of setting up and changing such workspaces as collaboration evolves can become onerous and is abandoned. Another approach is the use of software agents. Hattori, (12), proposed agent support for such systems emphasizing workspace and personal agents. Their work suggested activity and personal agents and concentrated on loose groups exchanging information. However, again the coordination needed to work towards some goal is not provided in this model. We use our metamodel to define generic agents that can integrate community and coordination semantics.

USING THE METAMODEL TO DEFINE GENERIC AGENTS

Software agents obtain information about their environments expressed in terms of collaborative concepts as percepts from the collaborative system. These percepts are then used by the agent system to determine actions to be taken in the collaborative system. Such actions are again expressed in terms of the collaborative metamodel. Our goal is to design an architecture with reusable agents that possess knowledge that can be combined to construct a number of collaborative processes. The initial goal is to have the generic agent object classes "mirror" the collaborative object classes. These agent classes are:

Activity agent – is set a goal and defines a plan to attain the goal. The plan requires tasks to be set up and monitored to realize parts of the goal. It initiates work-items to carry out the tasks.

Work-item agent – Carries out a specific task and reports to the activity agent. Arranges work in terms of role responsibilities. Can select specific collaborations such as a discussion, document management or notification service.

Role agent – determines the skills needed by the roles and finds participants with specific skills to carry out the role. The role agent finds such participants by interacting with a broker agent.

Group Agent – has as its goal to establish a group with common characteristics and share knowledge between them. Proactively, it can suggest that people join particular groups.

Personal agent – knows about a person's interests, responsibilities and commitments. Can maintain a person's schedules and keep track of their activities.

Artifact agent – possesses knowledge of how to structure a document and the skills needed to construct each part. Defines a plan in terms of events to be followed to achieve the artifact goal and interacts with coordination agents to find the activities and people to contribute to document construction

Connect (Broker) agent – knows domain-specific workspaces that can be specialized to selected goals. Selects or creates workspace instances with those goals.

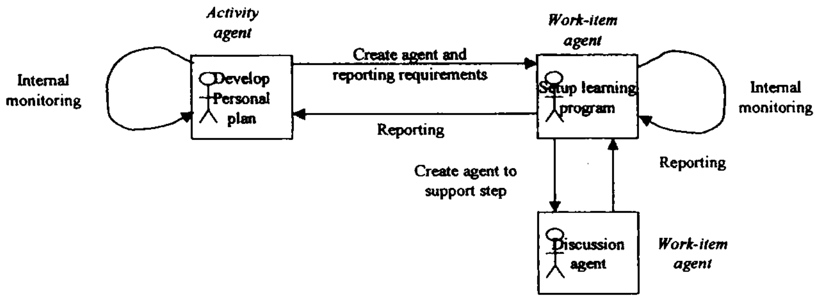
Coordination agent – Coordination (planning) agents create plans for other agents, as for example a plan to complete an artifact agent. Coordination agents monitor events and respond to these events following interaction with other agents.

At a more detailed level, we use a standard reasoning model for agents, as shown in Figure 12, and implement it using the three layer architecture (26) chosen from a number of alternative architectures (36). Agents are used to achieve goals using plans defined by agent users and details of these can be found in Hawryszkiewycz and Lin (17).

Integrating Agents with Conceptual Models

Generic agents can then be associated with the objects in the conceptual model. Thus for example an activity agent is associated with activity 'develop personal plan' in Figure 9. A work-item agent would then be associated with each work-item in 'develop personal plan'. As shown in Figure 12, the agents communicate with each other as a multi-agent system (17). The activity agent would create the work-items and work-item agents and monitor their progress. The activity agent perceives the state of each work-item through its work-item agent and suggests times to start the next. The activity agent then monitors the progress of each work-item. The goal is for the activity agent to decide how to expedite progress, either through calling for actions to be taken or possibly creating new work-items.

FIGURE 12
Agent Communication



SUMMARY

This paper described a collaborative metamodel that can be used to model collaborative applications and provide a framework for extending implementation to provide agencies that support collaboration. It described the model and illustrated its implementation and extension to flexible interfaces and software agencies.

ACKNOWLEDGMENTS

The work described here was supported by an ARC Discovery grant. The contribution of Brian Henderson-Sellers in refining the modeling concepts and relationships to object modeling is acknowledged.

REFERENCES

1. Applegate, L.M. "Technology Support for Cooperative Work: A Framework for Studying, Introduction and Assimilation in Organizations," **Journal of Organizational Computing,** 1, 1991, pp. 11-39.
2. BCSW - http://bscw.fit.fraunhofer.de/. Last accessed Sept. 24, 2004.
3. Bauer, B. "UML Class Diagrams: Revisited in the Context of Agent-based Systems," **Proceedings of Agent Oriented Software Engineering,** Montreal, 2001, pp. 1-8.
4. Carley, K.M. and L. Gasser. "Computational Organizational Theory." In Weiss, G. (Ed.). **Multiagent Systems.** MIT Press, 1999.
5. Carmel, E. **Global Software Teams.** Upper Saddle River, NJ: Prentice-Hall, 1999.
6. Chang, M.K. and C. Woo. "A Speech-Act-Based Negotiation Protocol: Design, Implementation, and Test Use," **ACM Transactions on Information Systems,** 12:4, 1984, pp. 360-382.
7. CoGenTex, Inc. EMMA Knowledge Representation. http://www.cogentex.com/research/emma/metamodel/index.html. Last accessed Sept. 27, 2004.
8. Conklin, J. and M. Begeman. "gIBIS: A Hypertext Too; For Exploratory Policy Making," **Transactions on Office Information Systems,** 6, October 1988, pp. 303-331.
9. Cummings, J.N., B. Butler, and R. Kraut. "The Quality of OnLine Social Relationships," **Communications of the ACM,** 45:1, July 2002, pp. 103-111.
10. Divitni, M. and C. Simone. "Supporting Different Dimensions of Adaptability in Workflow Modeling," **Computer Supported Cooperative Work,** 9, 2000, pp. 365-397.
11. Grant, R.M. "Prospering in Dynamically-competitive Environments: Organizational Capability as Knowledge Integration," **Organization Science,** 7:4, July 1996, pp. 375-387.
12. Hattori, F., T. Ohguro, M. Yokoo, S. Matsubara, and S. Yoshida. "Socialware: Multiagent Systems for Supporting Network Communities," **Communications of the ACM,** March 1999, pp. 55-59.
13. Hawryszkiewycz, I.T. "Support Services for Business Networking." In Altman, E. and N. Terashima (Eds.). **Proceedings IFIP96,** Canberra. London: Chapman and Hall, 1996.
14. Hawryszkiewycz, I.T. "A Framework for Strategic Planning for Communications Support," **Proceedings of the Inaugural Conference of Informatics in Multinational Enterprises,** Washington, October 1997, pp. 141-151.
15. Hawryszkiewycz, I.T. "Knowledge Networks in Administrative System," **Proceedings of the Working Conference on Advances in Electronic Government,** Zarazoga, Spain, February 2000, pp. 59-76.
16. Hawryszkiewycz, I.T. "Designing Learning Activities from Learning Objects," **Proceedings of the 19th Annual Conference of the Australasian Society for Computer in Learning in Tertiary Education (ASCILITE),** Auckland, December 2002, pp. 251-260.
17. Hawryszkiewycz, I.T. and A. Lin. "Process Knowledge Support for Emergent Processes," **Proceedings of the Second IASTED International Conference on Information and Knowledge Management,** Scottsdale, Arizona, November 2003, pp. 83-87.
18. Hiltz, R. and M. Turoff. "What Makes Learning Network Effective?" **Communications of the ACM,** 45:4, Apr 2002, pp. 56-59.
19. Kaplan, S.M., W.J. Tolone, D.P. Bogia, and C. Bignol "Flexible, Active Support for Collaborative Work wit ConversationBuilder," **Proceedings of the CSCW'9 Conference,** Toronto, November 1992, pp. 378-385.
20. Klockner, K., P. Mambrey, M. Sohlenkamp, W. Prinz, L Fuchs, S. Kolvenbach, U. Pankoke-Babatz, and A. Syri "POLITeam Bridging the Gap Between Bonn and Berli for and with the Users," **Proceedings of the Fourtl European Conference on Computer-supporte Cooperative Work,** ECSW'95, 1995, pp. 17-32.
21. Kreifelts, T., E. Hinrichs, K.-H. Klein, P. Seuffert, and G Woetzel. "Experiences with the DOMINO Offic Procedure System." In Bannon, L., M. Robinson, and K Schmidt (Eds.). **Proceedings of the European Conferenc on Computer Supported Collaborative Work, ECSW91** Kluwer Publications, Doedrecht, 1991.
22. Kuczmarski, T. **Innovation: Leadership Strategies fo the Competitive Edge.** Chicago: NTC Business Books 1997.
23. Kutti, K. and T. Arvonen. "Identifying Potential CSCE Applications by Means of Activity Theory Concepts: A Case Example," **Proceedings of the CSC@'92 Conference,** Toronto, November 1992, pp. 233-240.
24. LiveNet, http://livenet4.it.uts.edu.au.
25. Malone, T.W. and C. Fry. "Experiments with Oval: A Radically Tailroable Tool for Collaborative Work," **Proceedings of the CSC@'92 Conference,** Toronto, November 1992, pp. 289-297.
26. Müller, J.P. **The Design of Intelligent Agents.** Springer Verlag, 1996, pp. 7-44.
27. Neuwirth, C., D. Kaufer, R. Chanhok, and J.H. Morris. "Computer Support for Distributed Collaborative Writing: Defining Parameters of Interaction," **Proceedings of the 1994 Conference on Computer Supported Collaborative Work,** CSC@94, 1994, pp. 145-152.
28. O'Hara-Devereaux, M. and R. Johansen. **GlobalWork: Bridging Distance, Culture and Time.** San Francisco: Jossey-Bass, 1994.
29. Padgham, L. and M. Winikoff. "Prometheus: A Pragmatic Methodology for Engineering Intelligent Agents," **Proceedings of the OOPSLA 2002 Workshop on Agent-oriented Methodologies,** Seattle, WA, November 2002, pp. 97-108.
30. Parunak, H. and J. Odell. "Representing Social Structures," **Proceedings of Agent Oriented Software Engineering, Agents,** Montreal, 2001, pp. 17-31.
31. Rehfelldt, M. and K. Turowski. "Business Models for Coordinating Next Generation Enterprises," **Proceedings of the IEEE Academic/Industry Working Conference on Research Challenges,** April 27-29, 2000, pp. 163-168.
32. Walsh, K.R. and S.D. Pawlowski. "Collaboration and Visualization: Integrative Opportunities," **Journal of Computer Information Systems,** 44:2, 2003/2004, pp. 58-66.
33. Winograd, T. and F. Flores. "A Language/Action Perspective on the Design of Cooperative Work." **Human Computer Interaction,** 3, 1987, pp. 3-30.
34. Wooldridge, M. "Intelligent Agents." In Weiss, G. (Ed.). **Multiagent Systems.** MIT Press, 1999.
35. Workflow Management Coalition. http://www.wfmc.org. Accessed Sept. 30, 2004.