Supporting Distributed Cooperative Work in CAGIS

Heri Ramampiaro, Alf Inge Wang, and Terje Brasethvik

Dept. of Computer and Information Science
Norwegian University of Science and Technology (NTNU)
N-7491 Trondheim, Norway.
Phone: +47 73 594485, Fax: +47 73 594466, Email: {heri,alfw,brase}@idi.ntnu.no

Abstract. This paper describes how the CAGIS environment can be used to manage work-processes, cooperative processes, and how to share and control information in a distributed, heterogeneous environment. We have used a conference organising process as a scenario and applied our CAGIS environment on this process. The CAGIS environment consists of three main parts: a document management system, a process management system, and a transaction management system. The paper describes how these main parts may be configured and used together in order to support cooperative work in distributed environments.
Keywords: Document Modelling, Process Modelling, Transaction Modelling, and Cooperating Agents.

1 Introduction

After the introduction of the Internet, more and more projects are taking place in heterogeneous environments where both people, information and working processes are distributed. Work is often dynamic and cooperative and involves multiple actors with different kinds of needs. In these settings there is a need to help people coordinate their activities, share documents and information, and to manage access to shared resources. While the web makes it fairly easy to distribute information, the web itself does not contain explicit mechanisms to plan and coordinate activities and tasks, to organise, describe and classify information, or to control access to - and ensure consistency of - project documents.

In the absence of ”web-librarians” that can fulfil such tasks, the users themselves often have to figure out ad hoc solutions for doing this. To help the users, what is needed is a small set of powerful, easy-to-use and flexible tools that may be readily configured to support the task at hand. The CAGIS project - Cooperative Agents in the Global Information Space - aims to support such tasks by using a combination of software agents and small web-accessible tools. This paper describes how our CAGIS environment addresses the challenges given above.

2 The CAGIS environment

The CAGIS environment consists of three main components: A system for handling of distributed documents and document understanding, a system for supporting cooperative processes in a distributed environment, and a flexible transaction management system for shared, distributed resources. The following three sub-sections will describe our effort in these research areas in more details.

2.1 Document models and tools

Documents published on the web have to be organised, classified and described to facilitate later retrieval and use. One of the most challenging tasks is the semantic classification - the representation of document contents. This is usually done using a mixture of text-analysis methods, a carefully defined (or controlled) vocabulary or ontology, as well as a scheme for applying this vocabulary

when describing a document. The CAGIS document model toolset helps the users of a project group to do this semi-automatically, by way of a domain model expressed in a conceptual modelling language and by using text analysis tools as an interface to perform the actual classification and search. In other words,we use a conceptual model as a basis for creating meta-data descriptions (figure 1). These meta-data descriptions may then be accessed through our java-model viewer that enables search and browsing of documents through a standard web browser environment.

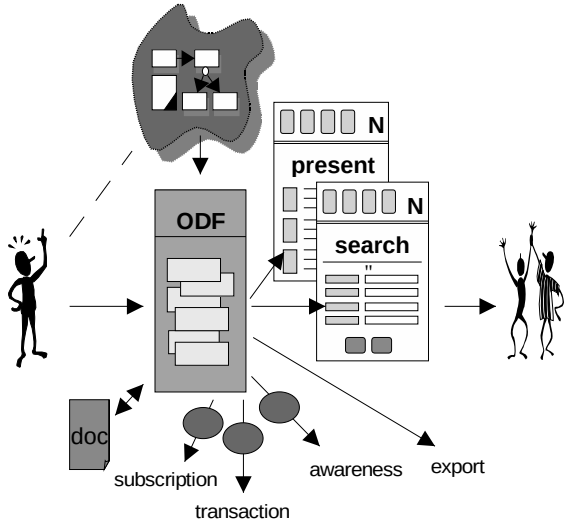

Fig. 1. Conceptual modelling for meta-data descriptions

Fundamental to our approach is the use of a conceptual modelling language to define and visualise the domain specific vocabulary to be used in the classification and retrieval process. Conceptual modelling languages contain the formal basis that is necessary to define a proper ontology, yet at the same time they offer a visual representation that allows users to take part in the modelling, and to read and explore documents by interacting directly with the models. The conceptual modelling language may thus be used throughout the entire process of classifying and retrieving documents on the web. In our approach, we use the Referent model language [2] an ER-like language with strong abstraction mechanisms and sound formal basis.

Our approach may be described as a three-step process: ***Domain Model Construction, Document Classification*** and ***Browsing & Retrieval*** - outlined below.

Domain Model Construction: Conceptual modelling is mainly a manual process. However, our domain models must be related to the text of the documents to be classified, hence we use a textual analysis tool as input for the modeling. A reference set of documents from the domain is run through a word frequency analysis tool, which produces a list of high frequency terms as input candidates for the actual conceptual modelling task. This is a manual and cooperative task performed by a selected set of users. Concepts are carefully selected, related to each other and given a textual definition. In order to prepare the finished domain model for later document classification, we then add lexical linguistic information to the model, i.e. the model is enhanced by adding a term-list for each of the concepts in the model. The term-list is a list of synonyms, instances and and conjugations for each concept that will be used later in the classification of a particular document.

Document Classification: Documents are classified by selecting domain model fragments that reflect the document content. This is performed semi-automatically by matching the document text against the term-lists for each of the concepts in the model. Concepts found in the document are then shown to the user as a selection in a graphical model viewer and the user may manually refine the classification by selecting and deselecting concepts and relations. When the user is satisfied, the selected model fragment is translated into XML and is stored as an ”Object Descriptor File”. The user also has to provide a selected set of properties for the document, such as its author, title etc. These attributes are also stored within the ODF.

Browsing and Retrieval: In order to retrieve documents, the users enter a natural language query phrase which is matched against the conceptual model in a similar way as in the classification process. The domain model concepts found in this search phrase (if any) are extracted and used to search the stored document descriptions. Users may then refine their search by interacting with the model. Found documents are presented as list in a Web-browser interface. We also have an enhanced ”document reader”, that is, when reading a document, all the terms in the document that matched a model concept is marked as a hyper-link pointing to the definition the model concept.

The layered architecture of our document tool is shown in figure 2. As mentioned, the main parts of the system are the web-enabled user interface and a set of servlets running on a standard web-server.

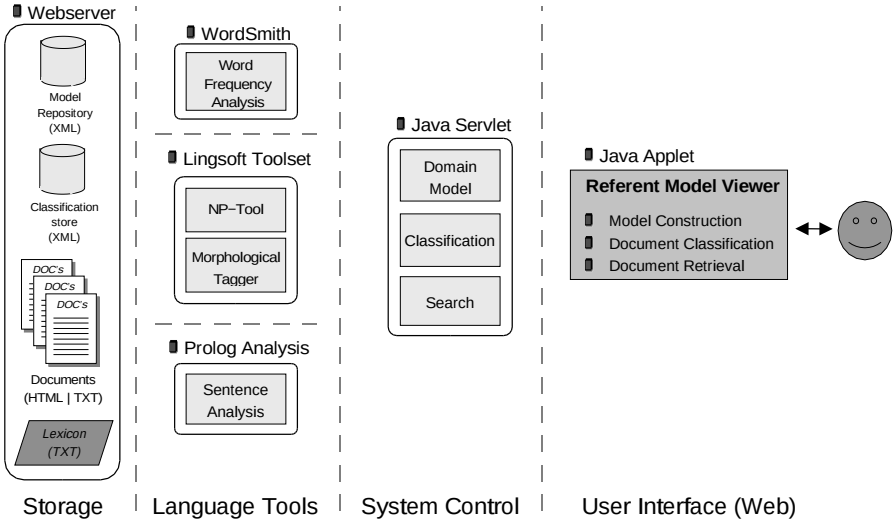

Fig. 2. Overview of system architecture

- The user interface is centred around a Java-based Referent-model viewer. As mentioned, users may interact with the model, explore concept definitions and relations, and then use the viewer directly in order to perform both classification and retrieval.
- The Java servlets define the overall functionality of the system. They are invoked from the model viewer and coordinate the linguistic tools incorporated into the system.
- At an ”intermediary” layer, between the servlets and the web-server, we use a number of linguistic tools analyse natural language phrases and give the necessary input to construct domain vocabularies and classify and retrieve documents. The Word frequency analyser from WordSmith is a commercially available application for counting word frequencies in documents and

producing various statistical analyses. A Finnish company, Lingsoft, has two tools for analysing nominal phrases and tagging sentences needed for the classification and retrieval of documents. A smaller Prolog application for analysing relations between concepts in a sentence is being developed internally at the university.
– Finally, the documents and their classifications are stored as files at the web server in HTML/TXT and XML format respectively. The domain model is also stored in XML and must be maintained separately. The linguistic tools rest on lexical information that is partly stored within the model XML file and partly integrated with the tools themselves.

A more detailed presentation of our approach and the system is given in [13, 14]

2.2 Process models and tools

A prototype of a process centred environment (PCE) has been developed to give process support to distributed, cooperative processes in CAGIS. The CAGIS PCE consists of three main components:

Workflow System supporting Distributed Mobile Processes This workflow system is used to model simple, repeatable workflow processes, and the system offers an agenda-browser for the end users. The workflow system allows an instantiated workflow model to be distributed as several process fragments on different workspaces. One benefit of this is the possibility to adapt the workflow to local environmental conditions. The workflow model instances are defined as XML-files distributed over several workspaces, and can be modified by the owner of the workspace. The ability to move and change workflow instances during enactment, can be used for reallocation of activities, dealing with exceptions (someone responsible for a particular activity acts sick), and delegation of work. The Workflow system is implemented in Perl, providing a CGI-interface through a web-server. For a more detailed description, see [15, 16].

The Process Modelling Language (PML) for the workflow system defines a process as set of activities that can have mutual pre-order relationships specified in XML syntax. That is, an *activity* can specify a set of *pre-links* identifying what activities to be executed before, and *post-links* identifying activities to be executed after the activity current activity. The pre- and post-links can be written as URLs, and therefore allow the process to be distributed over several workspaces. Every activity definition specifies a code part (a script). This code part is expressed in HTML, and can be used to simply present information, to specify a user input through a form, or to start a Java-applet. The term *process fragment* is used to name a group of activities in a workspace as part of the whole process. A process fragment is specified by a name, a workspace (location), and a list of references to activities.

Software Agents to support Dynamic, Cooperative Processes While the workflow system described above takes care of simple, repeatable process, we use software agents to support more cooperative and dynamic processes. Software agents typically takes care of inter-workspace (inter-group) activities such as negotiation activities (e.g., about of resource allocation), coordination of artifacts and workflow elements between workspaces, brain-storming, voting, marked support (e.g., agents as buyer and sellers of services), etc. Our multi-agent architecture consists of four main elements:

– **Agents** An agent is set up to achieve a modest goal, characterised by autonomy, interaction, reactivity to environment, as well as pro-activeness. We have identified three main types of agents: (1) *Work agents* to assist in local production activities, (2) *Interaction agents* to assist

with cooperative work between workspaces, and (3) *System agents* to give system support to other agents. Interaction agents are mobile, while system and work agents are stationary.

- **Agent Meeting Place (AMP)** AMPs are where agents meet and interact. AMPs support agents in doing efficient inter-agent communication. There can be different types of AMPs for different purposes. Each AMP will have a defined ontology (the framework described in section 2.1 can be used here), which the agents have to follow. We can perceive special AMPs for negotiation, coordination, information exchange, selling and buying services etc.
- **Workspaces** A workspace is a temporary container for relevant data (artifacts, models etc) in a suitable format to be accessed by tools, together with the processing (work) tools. It can be private, as well as shared. Files stored in a repository can be checked in and out to a workspace.
- **Repositories** Repositories can be global, local, or distributed, and are persistent storage of data. Experience Bases are one specific type of repository, that we can use in our multi-agent architecture to support community memory.

The multi-agent architecture is implemented in Java, using IBM Aglets framework to provide mobile agents, KQML is used for inter-agent communication, and ORBIX CORBA is used to offer communication to other applications and other agent systems. More detailed description of the multi-agent architecture can be found in [19, 11, 7].

Agent-Workflow GlueServer The Agent-Workflow GlueServer provides interaction between the workflow system and the multi-agent system. A **glue model** in XML defines the relationship between workflow elements and software agents. The **GlueServer** will offer services for a workflow activity to trigger an agent and vice versa. The GlueServer is implemented in Java, and ORBIX CORBA is used to facilitate communication with the agent system and workflow systems. More information about the GlueServer can be found in [18, 4].

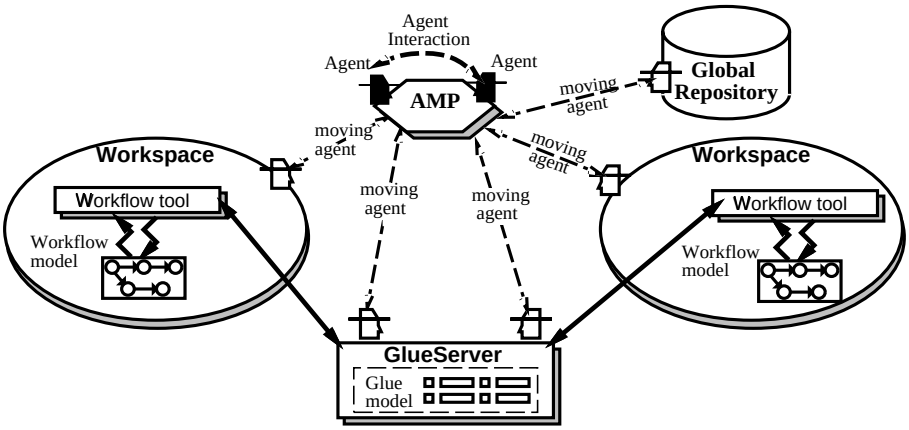

Fig. 3. The CAGIS Process Centred Environment

Figure 3 shows a simplified illustration of how the different components in the CAGIS PCE interact. In figure 3, there are two workspaces, each running a workflow tool (engine) with a local workflow model. In reality, this workflow tool can be shared, and the local workflow models in the two different workspaces can have relationships between them. The figure illustrates two different ways that software agents can interact with workspaces. In the first way, the agents can interact

directly with the user in the workspaces, using a graphical user interface to configure and interact with the agents. In the second way, all interaction with software agents goes through the GlueServer and the workflow tool. The workflow tool can activate an agent, or an agent can activate the workflow tool. The figure also shows that agents can be used to access repositories, but workspaces can also access files in the repository directly (not shown explicitly in the figure).

2.3 Transaction models and tools

A transaction is a basic work unit (or possibly a program segment) executed to perform some function or task by accessing and manipulating a shared database. Transaction modelling aims to capture the essential characteristics of transactions. In general, these characteristics include transaction behaviour and applied constraints.

Transaction modelling in CAGIS was motivated by the assumption that database management systems (DBMSes) will be used to manage resources. The main purpose of a transaction management system is then to control and manage access to shared resources, and to make sure that this access is done in accordance with prespecified consistency or correctness preservation constraints. This section briefly describes our effort in developing a transaction framework and a transaction management system for cooperating agents.

The transaction specification framework We have proposed a transaction framework to specify and execute customisable transaction models [10].

The main purpose of this framework is to provide configurable, application-specific transaction models. The ability to adjust the required degree of control to be provided through the transaction models is crucial in order to cover different situations. Some situations may, for instance, require strict control for data correctness, others may see control as just a burden, and so on.

The framework support has two parts: *Transaction characteristics specification* and *transaction execution specification*.

The former characteristics specification defines the main properties of the transactions to be executed: *ACID properties*, *relationship among the involved transactions* (e.i., transaction structures and transaction dependencies), *adopted correctness criteria* (i.e., user/application dependent criteria) and *applied policy* (i.e., rules for what mechanisms are to be used and how they are used). These characteristics are statically defined and must be done before the designated transactions are executed. The latter execution specification defines how the transaction execution is to be performed at run-time, in terms of composition of *management operations* (e.g., delegate, abort, commit etc.) and *regular access operations* (e.g., read, write etc.). The execution specification must conform the former, and has some fixed (though tailorable) initial operations, while the remaining operations can be adjusted at run-time.

Thus, CAGIS transaction framework distinguishes between static and runtime dynamic specifications. Compared with related frameworks, such as ACTA [5], ASSET [3], and TSME [6], the main difference is on the dynamic property. Such a property is important since it is not always possible to predict all aspects of application in advance. This becomes particularly relevant when taking software agents as well as cooperative work into consideration.

A detailed presentation of this transaction framework is given in [10].

The transaction management system A system for the transaction specification described above is depicted in Figure 4. This system is divided into two components; a specification environment and a runtime management system.

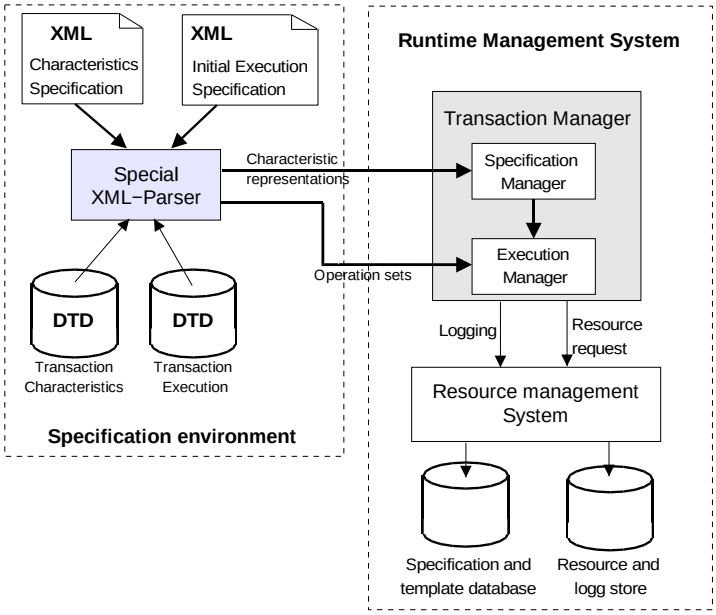

Fig. 4. Transaction management architecture

- **The specification environment** allows a transaction model designer to specify the characteristics of a transaction, and a set of operations to be run by a transaction. Both specifications (characteristics and execution) are done in XML. A special XML parser is used to verify specifications, against a corresponding prespecified Document Type Definition (DTD). This parser transforms (1) the specification of characteristics into an internal representation, and (2) the execution specification into a set of internal operations. They are both used by the transaction manager (see below) to control and monitor transaction executions. Further, both specifications are kept in an internal specification database. To avoid redundant specifications, the specification environment allows the designer to browse and check whether the given transaction characteristics are already in the specification database. This implies that future adjustments can be performed without needing to do the specification from scratch. Finally, if changes are made, the designer is asked whether the current specification shall be saved as a new version or to replace the old one.
- **The runtime management system** consists of a transaction manager and a resource management system. *The transaction manager* is responsible for managing the specification and execution of transactions, and to ensure that any transaction execution is done according to the specified characteristics. For example, if an ACID model is chosen, the transaction manager will enforce atomic and isolated execution. Correspondingly, if the atomicity property is relaxed, then the same manager will ensure that any failure would not necessarily cause global rollback. Instead, partial abort can be issued. As shown in Figure 4, the transaction manager again consists of a specification manager and an execution manager. *The specification manager* is responsible for controlling that all necessary representations from the specification environment are complete. In other words, it has to make sure that the specification of transaction characteristics can indeed be supported and that all necessary semantics are represented. If such a specification is not fully satisfiable, it will either notify the designer and ask him/her to adjust the specification, or it will choose a closest supported specification that can be found in the specification database. Otherwise, it makes the characteristics information available to the *execution manager*. The latter manager ensures that a transaction is executed consistently with

respect to its stated transaction characteristics and execution specifications. Based on these specifications, the execution manager issues the necessary and suitable transaction management operations. This means, that it issues `begin`, `abort` or `commit` operations, and other management operations that the user has specified.

The resource management system is responsible for managing and providing system resources for those actually executed transactions. It also maintains execution information of running transactions, and uses this to handle transaction aborts and system recovery. Finally, the resource management system is responsible of making sure that committed results are kept in a persistent store.

The transaction management system described above was implemented in a working prototype [12, 8] based on Java and the IBM Aglet-workbench. It has served as a test-bed for the transaction specification framework.

3 Conference Scenario

This section describes briefly a conference organising process which we shall use as a scenario. This based on the scenario presented in [9]. The seven main activities of the conference organising process are shown in bold-face in next paragraph.

First the Program Chair will initialise the conference management process by **Planning and announcing the conference**. People wanting to contribute to the conference will submit their papers. PC members will later **Record submitted papers** as well as information about the authors. Then, **Reviewers will be chosen** based on their expertise, and the **Paper reviewing** starts. The PC members will then **Collect reviewing results**, and a (electronic) review meeting will be held to **Determine acceptance of papers**. Accepted papers will then be **Grouped into sessions** and a final program, including a time-table for conference sessions, will be produced.

In this paper, we will focus on the last main activity, *Group Accepted Papers into Sessions*. Figure 5 shows the two main sub-activities of this activity.

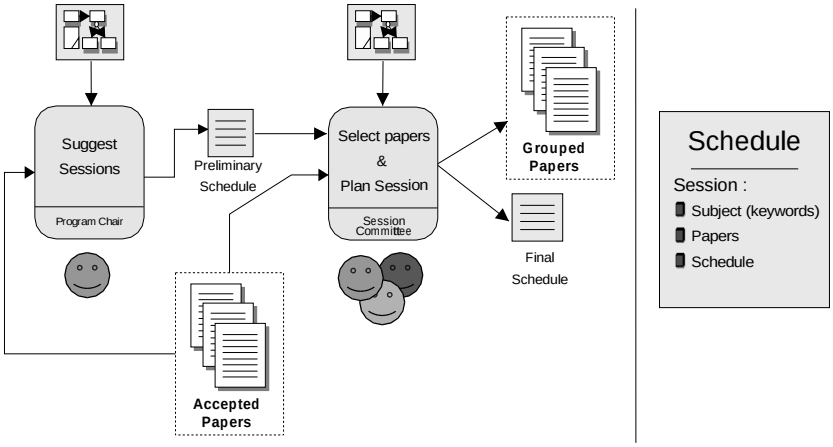

Fig. 5. Grouping of Accepted Papers into Sessions

Suggest Sessions The Program Chair is responsible for this activity, which can be decomposed into the following process steps:

1. Match all papers against a document model defining terms and expressions, and the relationships between them for the research domain.
2. Suggest a session division according to subjects.
3. Create a preliminary session schedule.
4. Set Up Session Committees (from Program committee members), one for each committee.

Select papers & Plan Sessions All members of Sessions Committees are responsible for this activity, and it can be decomposed into the following process steps:

1. Determine session subject & goals: An *initial* session description will contain session subject and goals.
2. Check papers for session: Sessions Committee members should mark papers that are relevant for a session to notify their interest. Papers will be marked *"possible"*.
3. Paper allocation: If papers are marked by more than one Session Committees, these committees must negotiate about which session is going to get the paper. Papers finally allocated to a session will be marked *"taken"*.
4. Check timeslot for session: Each session committee will mark the timeslot for the session.
5. Session allocation: Sessions that have the same timeslot will have to negotiate. When all sessions are allocated, the result will be added to the session description. The session description will now have the state *"final"*.
6. Publish session description: Each Session Committee will publish their session description to the other Session Committees and Program Chair.

In this paper we assume that the Program committee members will be distributed on different locations and the work with organising the conference will be done through computer interaction (without any physical meetings).

4 The CAGIS environment applied on the scenario

This section outlines how the CAGIS environment, consisting of tools and models to support documents, processes and transactions, can be applied on the scenario described in section 3. Our suggested architecture to support this scenario is shown in figure 6.

The two main activities we are focusing on, **Suggest Session** and **Select papers & Plan Session**, are executed by the Program Chair and the Sessions Committees respectively. In our solution we have therefor chosen to model the scenario using one workspace for Program Chair and one workspace for each Session Committees. Each workspace has a local process defined in a process model, and a workflow tool that enacts this process model. The process models are defined according to the process steps defined for **Suggest Sessions** and **Select papers & Plan Session** as given in section 3.

The Program Chair's first process step is to classify all papers according to their themes. This is done by using the **Document classification tool** to match all papers against the domain model. The domain model defines the vocabulary of keywords, extracted from the preliminary conference topics and from the submitted papers. The matching of papers against the domain model is visualised in the document model viewer, thus illustrating how the papers are thematically related to the concepts in the domain model. The Program Chair may interact with the model viewer to achieve the proper subject division of papers.

The ***Workflow tool*** will here notify the ***GlueServer***, that will initialise a *document agent* that may be used to access the documents through the document servlet. In the two next process steps,

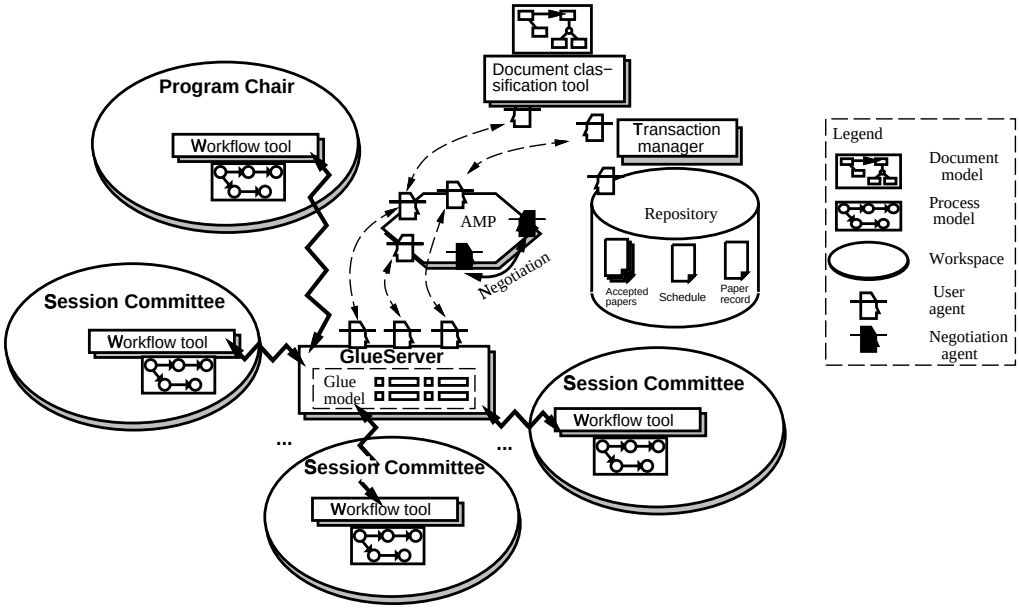

Fig. 6. The CAGIS framework applied on the scenario

the workflow tool will present the Program Chair with necessary documents and tools for creating a preliminary session schedule and setting up session committees. When session committees are selected among PC members, the session committee members will be notified through email describing what session committee to attend and what workspace to access.

The Session Committees will then start to work in their workspaces according to the process model enacted by the workflow tool (remember that all conference organising work will be done distributed on computers). First they have to determine session subject & goals, the workflow tool will notify the GlueServer that will initialise *brainstorming agents* for each session committee member. The result of this brainstorming process will end as an initial session description written by session chair. The next step of the process will be for the session committees to choose what papers to be in the session. Here the workflow tool notifies the GlueServer to initialise *paper select agents*. The paper select agents will retrieve information about available papers, and let the session committees mark interest of papers. The paper select agents will then mark papers in the *Paper record* in the repository (see figure 5). The result from marking papers will be returned to the GlueServer. If papers have been marked by several session committees, negotiation agents will be initiated to negotiate about which session is going to get the paper. If the negotiation process goes into a deadlock, the Program Chair will be notified, and he/she will make a final decision. The next step, is for the session committee to mark a timeslot for the session. This process works exactly like paper selection, but *session selection agents* will be used instead. When all session committees have selected their timeslots, the final session description is published to all participants, and the final conference program can be produced.

The **transaction manager** is responsible for managing the integrity of the document in the repository, and to ensure that agents always leave the system in a consistent state. Based on the description above, the Session Committees share both the schedule document and the paper record. Therefore, conflicts are likely to occur. There are several possibilities to resolve these conflicts. For example, one may provide an exclusive lock for each access, thus prohibiting other to see any changes until the related process is finished. This is usually unacceptable, since it might de-

lay the session arrangement process. Another possibility is to permit reading access. This allows other committees to see the intermediate changes, and therefore eases their decision process. A third solution is to permit simultaneous updates (i.e., write/write conflict). However, achieving consistency is possible only if the system supports multiple-version handling, with a sophisticated merging mechanism to capture all possible changes.

Next, tasks to support the session selection process are assigned to agents. As mentioned, this may involve document access too. To allow the transaction management system to ensure consistency, all agent operations involving repository access are managed as part of transaction execution. This also ensures that conflicting document access is managed properly. Moreover, suppose that a transaction consisting of several agent operations is initiated by the GlueServer. Then, assume that one of the involved agents fails, for example, while selecting papers from the paper record. Using traditional ACID transactions, this would cause a global rollback, that would discard all changes made so far, and kill all associated agents. However, if a lot of effort has already been invested, restarting all tasks from scratch may be expensive. To cope with this, we model each agent operation as a subtransaction of that executed by the GlueServer. Therefore, instead of aborting the transaction and killing all involved agents, the transaction manager allows the failing agent to just undo some of its changes. Other agents may proceed as normal.

5 Conclusion

This paper has presented the CAGIS toolset and its applications to a conference organisation scenario. The CAGIS toolset consists of a set of separate tools that may be used together to provide support for cooperative work across the web. The three major components of CAGIS are the Workflow tool, the Document classification tool and the Transaction manager. Each of these tools is implemented in true Web style, i.e. they are built around a standard Web server and use XML as a data storage and interchange format. These tools may all be configured according to the actual situation and use. The *Workflow tool* allows the individual workspaces to model and enact their activities in a local process model. The *document classification tool* uses a domain specific vocabulary, the domain model, to help users classify and search for documents. The *transaction tool* offers support for the specification and execution of customised and application-specific transaction models. The transaction manager thus offers the ability to design the required correctness criteria and to define and execute transactions that enforce these criterias on shared resources. Central to our system, and binding the individual tools together, is a GlueServer. The *GlueServer*, configures a set of software agents that can activate the different CAGIS tools. The *glue model* defines the relations between the individual workflow elements that may reside in different workspaces and the software agents that may be used to access the individual tools. This way, the various components of the CAGIS toolset may be used together in order to provide situation specific cooperative support.

Our CAGIS environment is not only applicable to conference organisation processes, but also for any process where people are working together, and where people and information are distributed. Examples of such processes can be cooperative software engineering processes, distributed educational processes, distributed organising processes, processes of selling and buying merchandises on the web etc. All these processes are characterised by distribution of people and information, and require people to interact and cooperate to reach the goal of the process.

Furthermore, when combining the tools in the CAGIS environment, we enhance the functionality of one specific CAGIS tool. In [1], an example of how the document models and tools can enhance the CAGIS multi-agent architecture is given. The document models and tools are here used to model the agent ontology, which define the language software agents can speak. In [17],

the transaction models and tools (in this paper called workspace manager) offer a way managing consistency of changing workflow models. This means that the CAGIS environment offers a selection of tools, which can be used in different combinations to give specific support. Future work will investigate more thoroughly what tools to pick for different scenarios.

Acknowledgements
We would like to thank Reidar Conradi for his constructive comments. The CAGIS project is sponsored by the Norwegian Research Council's Distributed Information Systems (DITS) Programme.

References

1. Alf Inge Wang and Anders Aas Hanssen and Bård Smidsrød Nymoen. Design Principles for a Mobile, Multi-Agent Architecture for Cooperative Software Engineering. Submitted to Software Engineering and Applications'2000 (SEA'2000).
2. Arne Sølvberg. Data and what they refere to. In P. Chen, editor, *Conceptual modeling: Historical perspectives and future trends*, Los Angeles, California, USA, 1998. 16th Int. Conf. on Conceptual modeling.
3. Alexandros Biliris, Shaul Dar, Narain H. Gehani, H. V. Jagadish, and Krith Ramamritham. Asset: A system for supporting extended transactions. In Richard T. Snodgrass and Marianne Winslett, editors, *ACM SIGMOD International Conference on Management of Data (SIGMOD 94)*. ACM Press, May 1994.
4. Bjørn Haakenstad. GlueServer, support for integrating workflow-systems with interactive agents. Technical report, Norwegian University of Science and Technology (NTNU), March 2000. Technical Report, Dept. of Computer and Information Science.
5. Panos K. Chrysanthis and Krithi Ramamritham. Synthesis of extended transaction models using ACTA. *ACM Transactions on Database Systems*, 19(3):450–491, Sept. 1994.
6. Dimitrios Georgakopoulos, Mark F. Hornick, and Frank Manola. Customizing transaction models and mechanisms in a programmable environment supporting reliable workflow automation. *IEEE Transactions on Knowledge and Data Engineering*, 8(4):630–649, August 1996.
7. Anders Aas Hanssen and Bård Smidsrød Nymoen. DIAS II - Distributed Intelligent Agent System II. Technical report, Norwegian University of Science and Technology (NTNU), January 2000. Technical Report, Dept. of Computer and Information Science.
8. Lars Killingdal and Mufrid Krilic. Programmerbare transaksjoner – prosjekoppgave. Technical report, Norwegian University of Science and Technology, April 2000. Student Project.
9. T. W. Olle, H.G. Sol, and A. A. Verrijn-Stuart. Information Systems Design Methodologies: A Comparative Review. North-Holland, 1982.
10. Heri Ramampiaro and Mads Nygård. Cagistrans: A transaction framework to support cooperating agents. Technical Report IDI-5/00, Norwegian University of Science and Technology, 2000.
11. Geir Prestegård, Anders Aas Hanssen, Snorre Brandstadmoen, and Bård Smidsrød Nymoen. DIAS - Distributed Intelligent Agent System. Technical report, Norwegian University of Science and Technology (NTNU), April 1999. Technical Report, Dept. of Computer and Information Science, 387 p.
12. Rune Selvåg. Web-transactions. Master's thesis, Norwegian University of Science and Technology, 2000.
13. Terje Brasethvik and John Atle Gulla. Semantically accessing documents using conceptual model descriptions. In P. Chen, D.W. Embley, and S.W. Little, editors, *Advances in conceptual modeling*, Paris, France, November 1999. WEBCM*99.
14. Terje Brasethvik and John Atle Gulla. Natural language analysis for semantic document modeling. In E. Metais, editor, *Proceedings on 5th International Conference on Application of Nature Language to Information Systems (NLDB'2000)*, Versailles, France, June 2000.
15. Alf Inge Wang. Experience paper: Using XML to implement a workflow tool. In *3rd Annual IASTED International Conference Software Engineering and Applications*, Scottsdale, Arizona, USA, 6-8 October 1999.
16. Alf Inge Wang. Support for Mobile Software Processes in CAGIS. In *Seventh European Workshop on Software Process Technology*, Kaprun near Salzburg, Austria, 22-25 February 2000.
17. Alf Inge Wang. Using Software Agents to Support Evolution of Distributed Workflow Models. In *In Proc. International ICSC Symposium on Interactive and Collaborative Computing (ICC'2000)*, page 7, Wollongong (near Sydney), Australia, December 12-15 2000.
18. Alf Inge Wang, Reidar Conradi, and Chunnian Liu. Integrating Workflow with Interacting Agents to support Cooperative Software Engineering. Submitted to Software Engineering and Applications'2000 (SEA'2000).
19. Alf Inge Wang, Chunnian Liu, and Reidar Conradi. A Multi-Agent Architecture for Cooperative Software Engineering. In *Proc. of The Eleventh International Conference on Software Engineering and Knowledge Engineering (SEKE'99)*, pages 1–22, Kaiserslautern, Germany, 17-19 June 1999.