A Process Centred Environment for Cooperative Software Engineering

Alf Inge Wang*

ABSTRACT

The cooperative aspects of software engineering have often been either eliminated or ignored, because it has been to hard to model cooperative activities in existing systems, or there has not been an interest for doing so. This paper describes the CAGIS process centred environment (PCE) that initially has been designed for modelling and providing support for cooperative software processes. The CAGIS PCE consists of three main parts: A simple activity-based workflow tool, a software agent system and a process middleware that glues the workflow tool and the software agent system. The paper describes the overall architecture and each part of the system.

Keywords: *Cooperative Software Engineering, Software Process Technology, Computer-Supported Cooperative Work, Multi-Agent Systems*

1. INTRODUCTION

Traditionally, modelling and enactment of software processes have been focusing on ”forcing” and guiding people to work according to a specified model, where interaction between people has been coordinated through a strictly defined control/data flow. Cooperative aspects of the software development process have often been either eliminated or ignored, because it has been hard to model cooperative activities in existing systems, or there has not being an interest for doing so. Also, software development processes are human-centred processes. In [2], Cugola and Ghezzi state that ”Human-centred processes are characterised by two crucial aspects that were largely ignored by most software process research: They must support cooperation among people, and they must be highly flexible”. This paper addresses these two challenges, and proposes a highly flexible framework for supporting cooperative software engineering (CSE) processes.

In the workflow and Computer-Supported Cooperative Work (CSCW) community, some work has resulted in the development of cooperative workflow systems. Most of these systems are role-based systems, where the roles and the cooperative interactions between these roles are modelled. Lately, software agents have been used to model and enact cooperative activities. The software agents represent users in cooperative efforts and act according to the users' requirements to reach a specified goal. By using software agents, we can benefit from the agents' ability to learn and adapt to a changing environment. Activity-based workflow and process systems on the other hand, are efficient to model pre-planned activities that e.g., can be derived from a project-planning tool. Activity-based workflow is not suitable for modelling cooperation, because interaction between roles are hard to represent in activity networks. Many workflow systems have also a problem to represent and support dynamic processes. This paper presents a framework to combine software agents with activity-based workflow, to gain flexibility and to be able to model most aspects of a process.

*Dept. of Computer and Information Science, Norwegian University of Science and Technology (NTNU), N-7035 Trondheim, Norway. Phone: +47 73594485, Fax: + 47 73594466, Email alfw@idi.ntnu.no

SEKE'02 July 15-19, Ischia, Italy.

2. THE CAGIS PROCESS CENTRED ENVIRONMENT

This section gives a detailed description of the CAGIS PCE [5] and its main parts.

2.1 A Motivating Scenario

The scenario describes a part of a process of creating a new 3D graphics game in a computer game company named CoolGames. This scenario focuses two departments that cooperate using the Internet because they are geographically distributed. Figure 1 illustrates a part of the development process for a new game for the two departments.

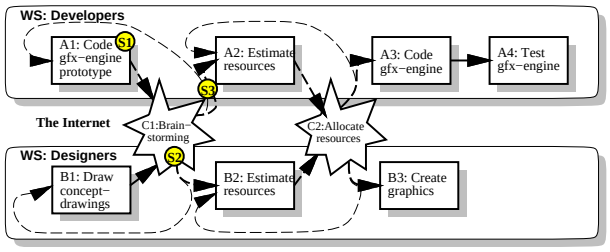

Figure 1: A scenario used to illustrate the CAGIS PCE architecture

The developers start to implement a 3D graphics-engine prototype (A1), while the designers are drawing some concept-drawings (B1). They then decide to have some brainstorming (C1) where they share ideas, drawings, and testing results from experiments with the 3D

graphics-engine. Based on the outcome of the brainstorm, the process can either go back to A1 and B1, or to continue to A2 and B2. In the activities A2 and B2, each department estimates how much human resources they will need (own and others). A resource negotiation is then initiated (C2) between the two departments. If the allocation process goes into a deadlock, the two departments must change their estimates (go back to A2 and B2). After a successful allocation, the process can proceed to the activities A3 and B3. The activity A4 will be started as soon as the activity A3 is finished. In figure 1 we can divide the process into three parts:

S1 **Individual activities** are activities that can be performed by individuals (A1-A4 and B1-B3).

S2 **Cooperative activities** are activities that only can be performed when two ore more persons are involved (C1-C2).

S3 **Cooperative rules** are relations between cooperative and individual activities (e.g. between C1 and A2) drawn in the figure as dotted lines.

2.2 The CAGIS PCE Architecture

We have chosen an architecture consisting of these three main components also shown in figure 2 providing process support according to the classification described in the last part of section 2.1:

S1 The **CAGIS SimpleProcess** workflow tool provides local process support for *individual activities* (A1-A4 and B1-B3 in figure 1).

S2 The **CAGIS Distributed Intelligent Agent System (DIAS)** provides support for *cooperative activities* (C1 and C2).

S3 The **CAGIS GlueServer** makes it possible for the CAGIS SimpleProcess workflow tool to interact with the software agents by specifying the *cooperative rules* between individual workflow and cooperative workflow in a GlueModel (represented as dotted lines in figure 1).

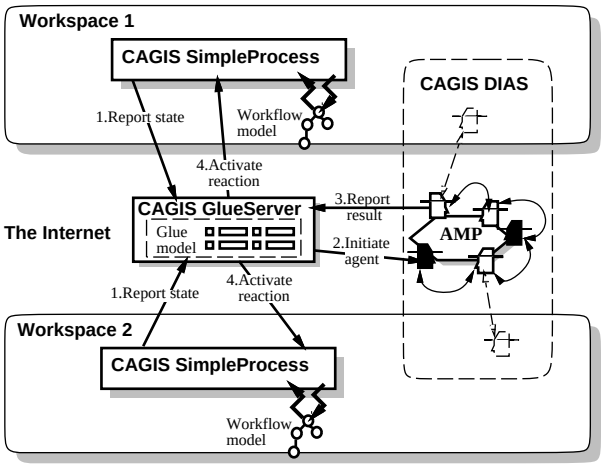

Figure 2: Architecture for the CAGIS PCE

A typical interaction between the different components in the CAGIS PCE will work according to the four steps as shown in figure 2:

1. The CAGIS SimpleProcess workflow tool will *report its state* to the CAGIS GlueServer, e.g. that it is finished with executing the activity A1 and B1 in the motivating scenario.

2. The CAGIS GlueServer will look through a GlueModel to see if anything is specified for the activities A1 and B1, and it will *initiate brainstorming agents* in the CAGIS DIAS (the cooperative activity C1).

3. A brainstorming agent can typically return two results: successful or unsuccessful. The *result is reported* to the GlueServer when C1 as finished.

4. The GlueModel can specify different reactions depending on the result reported to the GlueServer (successful or unsuccessful). Depending on the result, the GlueServer will *activate a reaction* in the CAGIS SimpleProcess (e.g. execute an activity), the CAGIS DIAS (e.g. initiate new agents) or the CAGIS GlueServer (e.g. change a definition of a cooperative rule).

The CAGIS SimpleProcess and the CAGIS GlueServer communicate through CGI, and the CAGIS DIAS and the CAGIS GlueServer communicate through CORBA using the MASIF [1] standard. The CAGIS PCE architecture is flexible since the CAGIS GlueServer can be used to interact with other agent systems through the mobile agent interface, and the CAGIS GlueServer can also be used to communicate with other workflow tools through the interoperability workflow-XML binding framework. In this way, the CAGIS PCE can federate systems offering a variety of process support.

3. CAGIS SIMPLEPROCESS

Flexibility has been the main motivation when designing the CAGIS SimpleProcess,by allowing the process model to be re-arranged and changed during enactment. This flexibility allows us to gradually build a process model from existing process fragments, to allow parts of the process to be unspecified, and to re-arrange and change the sequence of activities of the process model run-time.

3.1 The Concept and the Process Modelling Language

Process models in CAGIS SimpleProcess are specified in XML and the document type declaration (DTD) of the CAGIS SimpleProcess PML is as shown in figure 3. The figure shows that an activity has a name, is located in a workspace and is defined by the parts: Prelink(s), postlink(s), a state, a due time, a feedback option, a description, and a code part. The *code* part is used to specify HTML-code or an URL to a web page to be presented when an activity is activated. The HTML-code can be used to present some texts and pictures, list hyper-links to important documents and tools, present the user HTML-forms, or executing Java-applets. *Prelinks* specify the activities to be executed before current activity, and *postlinks* specify the activities to be executed after current activity. An activity will go from the state *Waiting* to *Ready*, if all prelinks (the prior activities) have the state *Finish*. The activity network works similar to hyper-linked web pages with states.

3.2 The Architecture

The CAGIS SimpleProcess architecture is based on a traditional web-architecture consisting of four CGI applications: A *process server* responsible for managing process changes and process states, a *process modeller* for modelling the process, an *agenda manager* presenting user agendas and activities, and a *monitor tool*. The CAGIS SimpleProcess has been implemented in Perl, and need an Apache web-server to run.

[1] MASIF is short for Mobile Agent System Interoperability Facility defined by the Object Management Group (OMG).

```
<?XML encoding=''UTF-9''?>
<!ELEMENT process (name,
                  (processfragment)+>
<!ELEMENT processfragment (name,
                           (workspace),
                           (activity)+)>
<!ELEMENT activity (name,
                    (workspace),
                    (prelink)*,
                    (postlink)*,
                    (state)?,
                    (due)?,
                    (feedback)?,
                    (description),
                    (code)*)>
<!ELEMENT name (#PCDATA)>
...
```

Figure 3: XML Document Type Declaration of the CAGIS SimpleProcess PML

4. THE CAGIS DISTRIBUTED INTELLIGENT AGENT SYSTEM

The CAGIS Distributed Intelligent Agent System (DIAS) is used in the CAGIS PCE to provide support for cooperative activities where more than one role participates. CAGIS DIAS consists of four main parts: (1) **Agents** are set up to achieve a modest goal, with the characteristics of autonomy, interaction, reactivity to environment, as well as pro-activeness. There are three main types of agents: System agents, Interaction agents and User agents. (2) **Agent Meeting Places (AMP)** are the agent places where agents exchange messages and services. (3) **Workspaces** are temporary containers for relevant working data in a suitable format, together with processing tools. (4) **Repositories** can be global, local or distributed, and are persistent storages.

The CAGIS DIAS offers an high-level agent-API, that can be used to implement cooperative activities such as trading of objects or services (marketplace), electronic brainstorming, electronic voting, resource negotiation, detection of file violation and synchronisation of files, etc.

4.1 The Application Interface

The agents are in the CAGIS DIAS able to communicate using KQML as a communication language using typically KQML performatives such as ask-if, insert etc. The CAGIS DIAS agent API provides methods for creating and killing agents, for message handling, for registering and un-registering agents, for moving agents, for negotiating between agents, for locating agents, and for information queries.

4.2 The Architecture

Our multi-agent architecture is an extension and specialisation of the more general **Agora** architecture proposed by Matskin et al. [4] suitable for modelling and supporting all kinds of cooperative work. The starting point for our multi-agent architecture was that it should be open to existing tools, systems, and operating systems, and it should be easy to configure, maintain and expand. We wanted the CAGIS DIAS to use free standard software components whenever available, and use mobile agents, because they provide efficient usage of network bandwidth, and less computation on the server is needed. Figure 4 shows the initial design for our multi-agent architecture. We have used a multi-tier architecture based on the agent, places, and things paradigm. The lower part of the figure (component infrastructure and agent infrastructure) defines the foundation, based on available standard implementations, which will provide functionality and services to the prototype of the multi-agent architecture. A central component both in workspaces and AMPs in our architecture is the facilitator that simplifies the implementation of agent communication, agent security, the mediation between agents, and the monitoring of agents. The reason for this is that the interaction between various entities can be controlled from one central point.

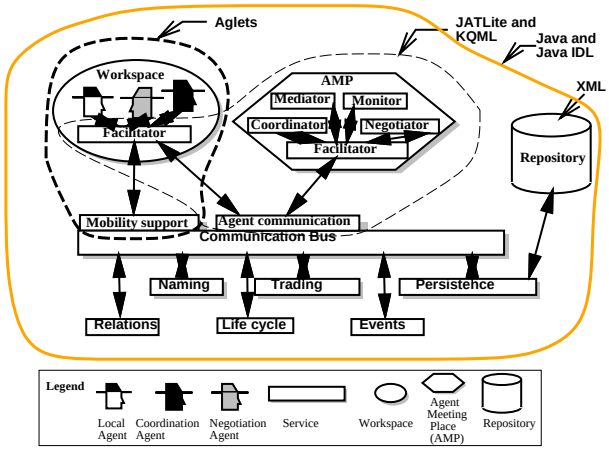

Figure 4: Recommended technologies for the CAGIS DIAS architecture

Further, figure 4 illustrates the technologies we have used to implement the various parts of the CAGIS DIAS architecture. *Java and Java IDL* have been used as the component infrastructure because Java provides code portability and is broadly accepted standard. *KQML and JATlite* were used to provide support for agent communication. Further to provide mobility support for agents, the *Aglets framework* from IBM [3] was selected. The Aglets framework was chosen because at the time we conducted the technology study, the Aglets implementation was closest to OMG's Mobile Agent Facility specification. In addition we suggested to use *XML* to represent information and work-productions in the architecture because a lot of XML tools are available in Java, and XML does not put any restrictions on the format of the information it shall represent.

5. THE CAGIS GLUESERVER

The *GlueServer* is a piece of middleware used to provide interaction between multi-agent systems (CAGIS DIAS) and the workflow systems (CAGIS SimpleProcess). A *GlueModel* specifies the relationship between process fragments and agents, making it possible for process fragments to delegate tasks to software agents, to use software agents to evaluate what to do next after completion of execution of a process fragment, or to monitor the environment for events to detect exceptions.

5.1 The GlueModel

A GlueModel is specified in XML using the *Glue Modelling Language*. Figure 5 shows the GlueModel for modelling the dependencies between the activities A1 and C1 in the scenario presented in figure 1. The GlueModel specifies that as soon as the workflow tool reports that the process fragment A1:Code gfx-engine is finished, a brainstorming agent should be initiated in the agent system representing the activity C1:Brainstorming. Based on the result returned by this brain-storming agent (successful or unsuccessful), the workflow tool should execute the process fragment A2 or A1 respectively.

Here is a more detailed explanation of the GlueModel. The *first part* (02-05) of the GlueModel specifies the **agent** involved in the

```
01 <fragment-agent-pair>
02   <agent agent-class="agents.brainstorming" amp-id="CoolGamesAMP">
03      <interaction-type>Periodic invocation</interaction-type>
04      <result>successful|unsuccessful</result>
05   </agent>
06   <fragment fragment-id="Developers/A1:Code gfx-engine prototype">
07     <reaction>
08       <result>successful</result>
09         <action fragment-id="Developers/A2:Estimate resources"
10                body="execute_process_fragment_PFNUMBER"></action>
11       <result>unsuccessful</result>
12         <action fragment-id="Developers/A1:Code gfx-engine prototype"
13                body="reexecute_process_fragment_PFNUMBER">
14         </action>
15       </reaction>
16   </fragment>
16 </fragment-agent-pair>
```

Figure 5: CAGIS GlueModel example

cooperative activity by an agent class and an amp-id to the agent place where the agent will interact with other agents (CoolGamesAMP). The **interaction type** specifies how the workflow system and the agent system should interact. When *Periodic invocation* is used, agents decide what to do next at the termination of a process fragment. There are two other interaction types: 1) *Predefined interface* meaning that the workflow tool delegates an activity to an agent. 2) *Dynamic monitoring* where monitoring agents are continuously probing the environments for certain events or states. Whenever an agent detects an abnormal situation, this abnormal situation is reported to the GlueServer that can initiate a reaction in the workflow tool.The **result** tag is used to specify what values the agent(s) can return.
The *second part* (06-16) of the GlueModel specifies the **process fragment** by a process fragment ID. The rest of the process fragment part is used to describe a **reaction** specified by **result - action** pairs. The result is the possible results returned from the agent, where as the action specifies what to do if there is a match. The user can specify the GlueModel directly in XML, or use a tool with a graphical user-interface for entering the information required.

5.2 The Architecture

Figure 6 shows the architecture of the CAGIS GlueServer consisting of three main components:

- **GlueEngine:** The main purpose of the GlueEngine is to parse the GlueModel and look for process fragment - agent pairs in the model matching with state information received from the workflow system or the agent system. If a process fragment - agent pair is found, the GlueEngine will initiate a reaction through the workflow interface (to a process fragment) or the agent interface (initiate agent).
- **Workflow Interface:** The workflow interface interacts with workflow systems through a XML-interface.
- **Agent Interface:** The agent interface interacts with agent systems through a MASIF interface.

A typical interaction between the three components in the CAGIS PCE can be as follows (the numbers are illustrated in figure 6):

1. The Workflow system reports its state to the GlueServer via the workflow interface.
2. The GlueServer finds a process fragment - agent match in the GlueModel using the GlueEngine.
3. The agent interface initiates an agent as specified in the GlueModel.

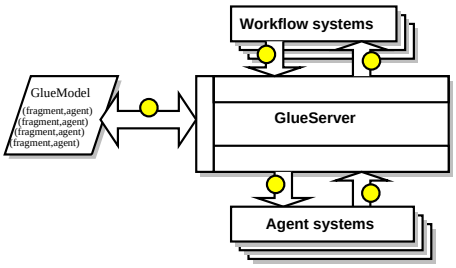

Figure 6: CAGIS GlueServer architecture

4. The agent system reports the result of an agent interaction back to the GlueServer through the agent interface.
5. The GlueServer will activate a reaction according to the GlueModel.

The GlueServer was also implemented in the Java programming language, and the GlueModels are stored in XML. The agent system interface was implemented with ORBIX CORBA according to the MASIF standard, using an interface agent in DIAS.

6. CONCLUSION

The goal when designing and implementing the CAGIS PCE was to make a PCE that provided cooperative support and was highly flexible. In [5], we present an evaluation of the CAGIS PCE in regards to this goal. In this evaluation, a conference organising process consisting of cooperative and individual activities was modelled in our CAGIS PCE, in Endeavors [1] and ProcessWeb [6]. The results showed that Endeavors had some problems in modelling the cooperative activities, while ProcessWeb was most efficient for modelling such activities. CAGIS PCE was overall most efficient in modelling the process (both individual and cooperative activities). We also measured the effort doing some specified process changes. Because of the flexibility provided by the GlueServer and the GlueModel, the CAGIS PCE was more efficient that the others in handling process changes. The main contribution of our work is therefore the GlueServer and GlueModel that allow agent(role)-based and activity network-based process tools to live side by side.

7. REFERENCES

[1] G. Bolcer and R. Taylor. Endeavors: a process system integration infrastructure. In *Proceedings of the 4th International Conference on the Software Process*, pages 76 – 89. IEEE Computer Society Press, 1996.

[2] G. Cugola and C. Ghezzi. Software Processes: a Retrospective and a Path to the Future. *SOFTWARE PROCESS – Improvement and Practice*, 4(2):101–123, 1998.

[3] D. Lange and M. Oshima. *Programming and deploying Java mobile agents with Aglets*. Addison-Wesley, 1998.

[4] M. Matskin, M. Divitini, and S. Petersen. An Architecture for Multi-Agent Support in a Distributed Information Technology Application. In *International Workshop on Intelligent Agents in Information and Process Management*, page 12, Bremen, Germany, 15-17 September 1998.

[5] A. I. Wang. *Using a Mobile, Agent-based Environment to support Cooperative Software Processes*. PhD thesis, Norwegian University of Science and Technology, Dept. of Computer and Information Science, NTNU, Trondheim, Norway, February 5th 2001.

[6] B. Yeomans. Enhancing the World Wide Web. Technical report, Computer Science Dept., University of Manchester, 1996. Supervisor: Prof. Brian Warboys.