

INTEGRATING WORKFLOW WITH INTERACTING AGENTS TO SUPPORT COOPERATIVE SOFTWARE ENGINEERING

ALF INGE WANG, REIDAR CONRADI*, and CHUNNIAN LIU †

Dept. of Computer and Information Science,
Norwegian University of Science and Technology (NTNU), N-7491 Trondheim, Norway.

ABSTRACT

Cooperative software engineering processes involve structured, repeatable processes as well as dynamic, cooperative processes. Existing workflow systems are suited to model and support the former type of processes, and multi-agent systems are suited to model and support the latter. We have designed and implemented a gluing-framework for integrating workflow processes with software agents. By using this framework, support for cooperative software engineering processes can be provided in a better and more flexible way. This paper focuses on how to integrate these two kinds of components into a functioning multi-agent based cooperative software engineering system.

Keywords: Agents, Cooperative software engineering, Workflow, Web.

1 INTRODUCTION

Cooperative Software Engineering (CSE) means that large-scale, software development and maintenance can be conducted in a distributed organisation or across organisations. CSE can be regarded as a special case of Computer Supported Cooperative Work (CSCW). If we categorise CSCW according to increasing complexity of process support [6], CSE falls into the most complex category of CSCW, characterised by distributed process fragments, partly shared workspaces, cooperation planning, and frequent interactions in intra/inter-workspaces. To support CSE processes, you have the problem of dealing with dynamic, unpredictable processes as well as stable, repeatable processes with totally different characteristics. Workflow and process systems like InConcert [14], Lotus Notes [11], Active Mail [4], and MAFIA [7] offer good support for stable, pre-planned processes, providing users activity agendas, invocation of tools, presentation of the state of the process etc. Multi-agent systems are better fit to model and support users involved cooperative processes [8, 22, 2]. By combining these two technologies, cooperative software engineering processes can be modelled and supported more completely.

In [19], we proposed a multi-agent system (MAS) architecture for CSE, which is an extension and specialisation of the more general architecture [9] for CSCW. The MAS is particularly useful in modelling and providing support to cooperative activities (communication, coordination, collaboration and negotiation). We have also developed two prototyped systems based on the proposed architecture. One is a multi-agent architecture [12, 5, 13, 17] supporting cooperative work through software agents. The other, is a workflow tool [16, 18] to model and support simple stable, repeatable processes. In this paper we describe how we combined these two systems, by using a GlueModel defining interaction between these systems, and a Glue-Server executing the GlueModel and communicating with the agent system and the workflow tool. By doing this, we integrate the workflow process with interacting agents into a MAS-based CSE system.

There are several advantages by combining workflow technology with multi-agent technology. Dynamic cooperative processes between people are hard to model in traditional process modelling languages, because these languages have difficulty representing interaction between autonomous participants. However, *software agents* provide a natural way of modelling dynamic cooperative processes by letting a software agent play a role to represent a person. In this way, the software agent can execute, e.g. a negotiation process on behalf of a person and his wishes. Software agents are also useful when there is an uncertainty in the process regarding resources, availability, the state of work etc. Software agents can be used to gather required information for making decisions on what to do next in the process. For more structured and more repeated processes, workflow tools can provide a better support. This is because they make the process modelling simpler by offering process modelling languages specialised for modelling such processes. Also many companies have workflow tools already installed, and want to keep them. By combining these technologies, we can benefit from both.

Our approach provides a framework for doing *component-based workflow*. This means that process fragments (parts of the whole process model) can be combined in a flexible way. The GlueModel defines the rules for how the workflow components can be combined and (re-)configured, and software agents are used to get information from the work-

*Tel: +47 73 594485, Email: alfw/conradi@idi.ntnu.no

†Beijing Polytechnic University (BPU), Beijing, P.R. China, Email: bpvliu@public.bta.net.cn. Chunnian Liu's work was supported in part by the Natural Science Foundation of China (NSFC), Beijing Municipal Natural Science Foundation (BMNSF), and the 863 High-Tech Program of China.

ing environment to make the correct decisions. The process fragments are highly distributed, so process owners can change and manage their process fragments themselves. The GlueServer framework can also be applied on other workflow systems, by using the Interoperability Workflow-XML Binding framework. Since GlueModel can be used by different workflow systems, an organisation can let sub-organisations use different types of workflow tools locally. The GlueModel will define the interaction between these sub-organisations, and interactive-agents will execute the according cooperative processes.

2 RELATED WORK

In [1], Chang and Scott present an agent-based workflow system. They claim, that unlike groupware products that simply provide a passive information space, agent-based workflow can enable active collaborative work among participants. The agent-based workflow system proposed in their paper support people working in different places, to work together at the same time in virtual rooms. The core of the system is based on WWW and a set of distributed agents to facilitate various parts of workflows (as activity agendas, document sharing and tool sharing) to improve efficiency among participants. The architecture is based on stationary agents, and the HTTP-protocol serves as an infrastructure for agent-communication. The coupling to workflow is provided through a workflow agent that interfaces with an existing workflow repository through a standard Workflow Management Coalition API. This means that various workflow tools can access the workflow repository, while the workflow agents act as facilitators, providing an interface between other agents and the workflow tool.

In two papers [15, 10], Shepherdson and Merz et al., present two approaches to use software agents for cross-organisational workflow. Both papers claim that an obstacle for cross-organisational workflow is that organisations have to give up their local autonomy to cooperating partners. One workflow system for the cooperating partners is therefore not an appropriate choice. In [10], Merz et al., propose to use mobile agents to overcome problems for cross-organisational workflow, typically lack of a common communication infrastructure, lack of central management, and high coordination costs of workflow management systems. Their multi-agent infrastructure serves as a communication mechanism that bridges organisational boundaries. In [15], a approach called *agent enhanced workflow* has been chosen. Agent enhanced workflow is achieved by combining a layer of agents with a commercial workflow system. The agent layer is given responsibility of business process management. The advantage of this approach is to provide automatic provisioning, interoperability, support for visualisation and verification services, while protecting the original investment in workflow technology.

The main difference in our approach compared to the ap-

proaches described above, is that we use agents only to model and support cooperative activities between actors, while a traditional workflow system is used to model and support isolated activities. We propose a model and a framework to glue these two different models and systems, to benefit from both.

3 REVIEW OF MAS-BASED ARCHITECTURE FOR CSE

In our MAS-based CSE architecture, there are four components [19]:

1. *Agents*: An agent is a piece of software created by and acting on behalf of the user or some other agent. It is set up to achieve a modest goal, with the characteristics of autonomy, interaction, reactivity to environment, as well as pro-activeness. We distinguish between the following different types of agents: *Work agents* to assist in normal software production activities, *Interacting agents* to help participants in their cooperative work, and *System agents* to give system support to other agents.
2. *WorkSpaces (WS)* A workspace is primarily a temporary container for relevant data in a suitable format, together with the processing tools. Workspaces can be private or shared, and the work in a workspace is often guided by a process model where data can be checked in and out from persistent repositories.
3. *AgentMeetingPlaces (AMPs)* AgentMeetingPlaces are where agents meet and interact. AMPs are built on the underlying communication mechanisms, but provide agents with more intelligent means to facilitate their interaction by using KQML [3] as a communication language and a defined ontology.
4. *Repositories* Our architecture allows persistent repositories to be global, local or distributed. Repositories can be used as a storage for products, but also as an Experience Base.

In the architecture, the four components are interconnected and interoperated as follows:

1. Agents are created by people to assist their work; or by other agents to perform delegated work; or by default to manage WSs or AMPs. In this paper we are concerned mainly with interacting agents triggered through the GlueServer by a workflow tool.
2. Agents are clustered mainly according to people grouping in workspaces.
3. Communication between agents is via AMPs.
4. Within a group of agents and their shared WS, any existing process models and their tools are allowed, and any traditional process architecture can be applied. Different

process fragments may be expressed in different Process Modelling Languages (PMLs) and require different process tools.

Figure 1 shows an example of our MAS-based architecture for CSE containing workspaces, one AMP, agents and repositories. The *negotiation agents* and *coordination agents* are used to provide cooperative support between the workspace WS1 and WS2.

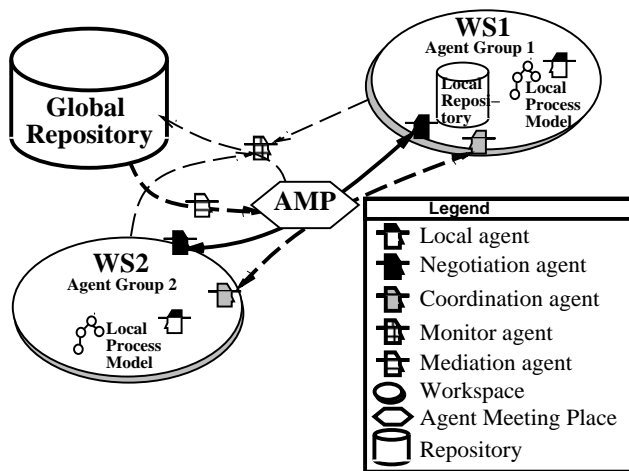


Figure 1. An example of the CAGIS MAS-based architecture for CSE

4 GLUING WORKFLOW WITH INTERACTIVE AGENTS

In a distributed setting, the overall process of a software development and maintenance project is distributed on various WSs, so the local software process running in a particular WS is a fragment of the overall process. Our main focus in this paper is the integration of process fragments with interacting agents. A process fragment can be seen as a sub-process of the overall process, and the Workflow Management Coalition defines a sub-process as the following: "A process that is enacted or called from another (initiating) process (or sub process), and which forms part of the overall (initiating) process. Multiple levels of sub process may be supported." [20]

The smallest building block in our workflow PML is an *activity*. We define a *process fragment* as a part of a process consisting of partial ordered activities, located in a workspace, and its *working context* (data, tools, human roles and agents). Our framework defines the integration between software process and interacting agents in terms of process fragments, rather than the whole process or individual activities. We are not losing any generality by doing this, because a fragment may consist of a single activity. A process fragment is identified by a name, workspace (URL), a list of activities and associated data. Since the

workspace is identified by a URL, the process fragment name and workspace name make an unique identifier.

4.1 THE GLUESERVER

The main functionality of the GlueServer is communication between workflow systems and the multi-agent system, and installation and execution of the GlueModel. The GlueServer is implemented in Java using OrbixWeb CORBA implementation, and it consists of four main parts (see figure 3):

1. **FragmentServer** The FragmentServer receives requests from the workflow system. A *GlueServer request* formatted in XML looks as shown in figure 2. It describes the identifier for the process fragment requesting the GlueServer, the other process fragments involved (interactors), and the name of the agent-class that should be initiated.

```
<GlueServerRequest>
  <From>Fragment-id</From>
  <Interactor>Fragment-id</Interactor>
  ...
  <AgentClass>Agent-Class</AgentClass>
</GlueServerRequest>
```

Figure 2. Workflow request to GlueServer

2. **GlueEngine** The GlueEngine parses the GlueModel, and finds matching fragment-agent pairs based on agent-class and fragment-ids (more on this in section 4.2).
3. **AgentClient** When fragment-agent matches are found in the GlueModel, the AgentClient connects to the multi-agent system and initiate agents as specified in the GlueServer request and the GlueModel.
4. **FragmentClient** Depending on the result returned from the initiated agents and what is specified in the GlueModel, the FragmentClient will send a response back (formatted in XML) to the workflow system on what actions to take.

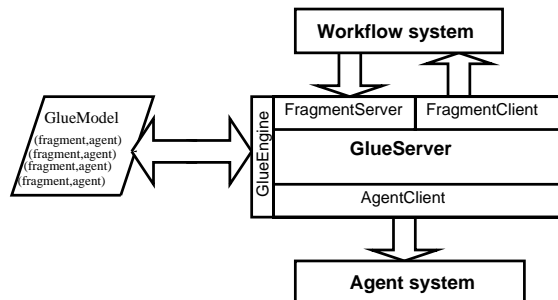


Figure 3. GlueServer architecture

Fragment-agent connection We perceive the following types interactions between a workflow system and interacting agents:

- *Predefined Interface*: Some process fragments have a predefined interface with interacting agents. For example, a fragment may have a negotiation activity that is delegated to an agent. When the execution arrives at that activity, a negotiation agent should be initiated, and the fragment may wait until the negotiation result is reported by the agent.
- *Periodic Invocation*: For example, by the termination of a fragment, usually a decision will be made about the next step of the process. The possible decisions include: Going on to the next fragment, iterating on the current fragment, changing the model of current fragment and re-executing it, moving the fragment to another WS, etc. The decision may depend on environmental information, reported by interacting agents.
- *Dynamic Monitoring*: Some monitoring agents may be designed to continuously probe the events and status of the environment. Whenever an abnormal situation is detected by an agent, the monitoring agent should report the information to relevant process fragments and the latter should be aware of the information in time and react accordingly.

4.2 THE GLUEMODEL

We use the GlueModel to specify how process fragments and interactive agents interact. The GlueModel is specified in XML, and process fragments and interactive agents are grouped into (*fragment, agent*) pairs. These pairs model all possible inter-operations of fragments and agents in a CSE system.

Result-reaction pairs The formalism specifies both the agent and the process fragment. The agent specification contains the class of the involved agent, the identifier of the AMP where the agent works, the type of interaction (communication, coordination, negotiation, etc.), and the result reported by the agents. The fragment specification contains the process fragment identifier, a list of (*result, reaction*)-pairs specifying the corresponding reaction by the process fragment for each possible result reported by the agent. Figure 4 shows the definition of the GlueModel given in XML document type definition. The set of possible reactions are shown in figure 4 as the attribute *body* in the XML-element *action*.

In the above formalism, a keyword marked by '+' or '*' means "one or more" or "none or more", respectively. The PFNUMBER, AGENTNUMBER, WSNUMBER and AMPNUMBER are used as place-holders for identifiers to process fragments, agents, workspaces, and agent meeting places respectively. We have implemented a graphical Java-based tool to make it easier to enter fragment-agent

```
<?xml version="1.0" encoding="UTF8"?>
<!ELEMENT GlueModel (fragment-agent-pair)+>
<!ELEMENT fragment-agent-pair (agent,fragment)>

<!ELEMENT agent (interaction-type,result)+>
<!ATTRLIST agent agent-class CDATA #REQUIRED
              amp-id CDATA #IMPLIED>
<!ELEMENT interaction-type (#PCDATA)>
<!ELEMENT result (#PCDATA|result)*>

<!ELEMENT fragment (reaction)*>
<!ATTRLIST fragment fragment-id CDATA #REQUIRED>
<!ELEMENT reaction (result,action)+>
<!ELEMENT action EMPTY>
<!ATTRLIST action fragment-id CDATA #IMPLIED
              agent-id CDATA #IMPLIED
              workspace-id CDATA #IMPLIED
              amp-id CDATA #IMPLIED
              body (execute_process_fragment_PFNUMBER
                  |move_process_fragment_PFNUMBER_to_workspace_WSNUMBER
                  |halt_process_fragment_PFNUMBER
                  |change_and_reexecute_process_fragment_PFNUMBER
                  |add_new_process_fragment_PFNUMBER
                  |remove_process_fragment_PFNUMBER
                  |start_new_interaction_by_AGENTNUMBER
                  |stop_interaction_by_AGENTNUMBER
                  |create_a_new_AgentMeetingPlace_AMPNUMBER
                  |remove_AgentMeetingPlace_AMPNUMBER
                  |change_fragment-agent-pair_PFNUMBER_AGENTNUMBER)
              #REQUIRED>
```

Figure 4. Fragment-agent-pair specification

pairs. A screenshot of the GlueModel Maker is shown in figure 5.



Figure 5. GUI tool for adding fragment-agent pairs to the GlueModel

4.3 INTERACTION WITH OTHER WORKFLOW TOOLS

Current implementation of the GlueServer is not capable of interaction with other workflow systems than our own prototype. We will however describe how other workflow systems can be integrated in the system using the Workflow Management Coalition Interoperability workflow-XML binding specification (Wf-XML) [21]. The intention of this specification is to allow workflow systems supporting simple chained and nested workflow interoperate both asynchronously and synchronously. The Wf-XML defines two types of messages that can be sent between workflow systems; requests (initiate an operation on a remote resource), and response (send the result of an operation to its requesting resource). In addition four opera-

```

<?xml version="1.0"?>
<WfMessage Version="1.0">
  <WfMessageHeader>
    <Request ResponseRequired="yes"/>
    <Key>{Interactor process fragment}</Key>
  </WfMessageHeader>
  <WfMessageBody>
    <ContextData>
      <AgentClass>{Agent class}</AgentClass>
    </ContextData>
  </WfMessageBody>
</WfMessage>

```

Figure 6. *GlueServer* request written as a Wf-XML request

tions are defined for Wf-XML messages; `CreateProcessInstance`, `GetProcessInstanceData`, `ChangeProcessInstanceState`, and `ProcessInstanceChanged`. These four operations could typically be used as reactions in the *GlueModel*. Note that the reactions described in section 4.2 are not supported by the operations defined in Wf-XML. Figure 6 shows how a *GlueServer* request from a Wf-XML compatible workflow system would look like.

5 APPLICATION SCENARIOS

In [19], we presented a scenario based on the software development and maintenance process from a Norwegian software company, in this paper called AcmeSoft. In this section, we will look at some cases of interconnection and interoperation between process fragments and interacting agents, and show how the *GlueServer* improves support for these situations.

First we recapitulate the relevant parts of the scenario. In company AcmeSoft, among other things, there is a Maintenance Planning Group (MPG), a Release Planning Group (RPG), and a Development Group (DG). The MPG major concern is to plan and estimate error corrections to existing software products based on error reports from customers. The MPG process consists of the following process fragments: *register_report*, *estimate_resources*, *allocate_resources*, *sends_out* (a Change Order to the DG) etc. On the other hand, the RPG plans what new functionality should be implemented in existing software products based on requirements from emerging technology and market demands. AcmeSoft is committed to quarterly update-releases and one major release per year per product. The RPG process also includes fragments like *estimate_resources*, *allocate_resources* and *sends_out* (a Work Order to the DF). Finally, all real development and correction work is carried out by the DG consisting of software engineers. The process fragments *estimate_resources* in MPG/RPG try to allocate a particular developer (or a group of developers) from DG for a particular maintenance/develop task. The identity of a fragment is determined by a fragment name and a WS name. For example, the process fragment *estimate_resources* in MPG will be denoted as MPG/*estimate_resources*. To give unique identifiers to workspaces, they should be named with an URL

(like <http://www.acmesoft.no/project/MPG>). In this paper we name the workspaces only with the local workspace name.

5.1 SCENARIO 1: NEGOTIATION

The first situation we consider, is when MPG and RPG compete for a particular developer. Remember that the same DG is responsible for both maintaining existing products, and for developing new updates/releases. Suppose that both MPG and RPG are trying to allocate the same developer because of his/her special skills. Here we have a conflict in resource allocation between the process fragments *estimate_resources* for MPG and RPG. To solve the conflict, negotiation between MPG and RPG is necessary. In this case, two negotiation agents representing the two groups (equipped with problem-specific negotiation strategies) carry out the inter-group interaction. The negotiation process will be carried out in Acme Agent Meeting Place (AMP), providing inter-agent communication facilities. Figure 7 shows the XML for the fragment-agent pairs involved in the negotiation process.

This situation shows how periodic invocation (see section 4.1) can be used to combine workflow with agents. When MPG and RPG are estimating resources, they both will access a shared repository denoting what resources they will need. As a result, the workflow tool will be able to know if a resource conflict has occurred. After the termination of for instance MPG's process fragment *estimate_resources*, the workflow tool will send a *FragmentServer* request for negotiation between MPG and RPG where the agent-class is specified. The *GlueServer* will then look in the *GlueModel* for matching fragment-agent pairs both for MPG and RPG, and initiate the agents involved in the negotiation process. The result of the negotiation process will be sent back to the *GlueServer*, and the *GlueServer* will forward the reactions, according to the *GlueModel*, back to the workflow tools in MPG and RPG.

5.2 SCENARIO 2: COORDINATION

Suppose that a particular developer in DG is assigned to a maintenance task by MPG, but quits or is given another job before the task can be completed. Assume that DG is not sure if they can assign this job to another of their own developers, or if they give the assignment to an external organisation. Then the DG (or rather, its process fragment *report*) would report this difficult problem to MPG. The *replanning* process fragment in MPG may decide to delegate the unfinished work to someone else in DG if possible, or an external organisation (company or another department). Coordination agents are used to decide whether the work should be re-assigned in DG or to an external organisation. The coordination agents will check updated information about the workload of DG. If the outcome of the coordination is to assign the task to an external organisation, it will move the remaining work together with its process fragment PF# to the external work space WS#. A *GlueModel* for scenario 2

```

<fragment-agent-pair>
  <agent agent-class="agents.Negotiation" amp-id="AcmeAMP">
    <interaction-type>negotiation</interaction-type>
    <result>yes|no</result>
  </agent>
  <fragment fragment-id="RPG/estimate_resources">
    <reaction>
      <result>yes</result>
      <action fragment-id="RPG/allocate_resources" body="execute_process_fragment_PFNUMBER"></action>
      <result>no</result>
      <action fragment-id="RPG/allocate_resources" body="change_and_reexecute_process_fragment_PFNUMBER">
        </action>
    </reaction>
  </fragment>
</fragment-agent-pair>
<fragment-agent-pair>
  <agent agent-class="agents.Negotiation" amp-id="AcmeAMP">
    <interaction-type>negotiation</interaction-type>
    <result>yes|no</result>
  </agent>
  <fragment fragment-id="MPG/estimate_resources">
    <reaction>
      <result>yes</result>
      <action fragment-id="MPG/allocate_resources" body="execute_process_fragment_PFNUMBER"></action>
      <result>no</result>
      <action fragment-id="MPG/allocate_resources" body="change_and_reexecute_process_fragment_PFNUMBER">
        </action>
    </reaction>
  </fragment>
</fragment-agent-pair>

```

Figure 7. GlueModel for scenario 1

is shown in figure 8.

In scenario 2, MPG will issue a request to the GlueServer to initiate a coordination process involving the process fragments MPG/replanning and DG/report. Coordination agents will find a solution to who is going to finish the job specified in the process fragment *implementation_changes*. Depending on the result of the coordination (local/external), *implementation_changes* will be executed by DG or an external organisation. For the fragment-agent pair (MPG/replanning, agent.Coordination), the result *external* will cause more than one action to be executed. Note also that the coordination process is executed in ExternalAMP. This is to make the coordination possible beyond local organisational borders.

5.3 DISCUSSION

As we have shown in the two scenarios above, our framework for combining process fragments with interactive agents enable us to model and support dynamic and ad-hoc cooperative processes with software agents, while traditional workflow process described in some process modelling language takes care of the routine processes. In scenario 1, the software agents are used to negotiate about human resources between two groups. The GlueModel is used to define how the workflow tools shall act according to the result of the negotiation between the software agents. This means that each group can model the local process concerning themselves on their own, while the GlueModel will model how to deal with interaction with other groups. Scenario 2 shows how the glue model can deal with exception handling, when a particular developer assigned for a specific task is resigning or leaves. In this scenario, we

could also use software agents to search for external human resources capable of doing this specific task. In addition, the software agents could form a market-place where they can look for expertise and negotiate about the price to do a specific job. The glue model will then specify what activities to do next based on the result returned by the agents.

The GlueModel is a part of the total process model, and defines how different cooperating groups shall interact. Agents are then used to do inter-group interaction as representatives for the groups. It is also possible to use the GlueModel to specify for what situations local process models have to be changed, based on work-environment informations collected by agents. This means that the GlueModel also models part of the meta-process. The GlueModel will notify the local workspace, when a process model must be changed, and the changes will be executed locally in the workspace. In addition, the GlueModel is reflective. This means that a reaction in a fragment-agent pair can cause a change in the same or other fragment-agent pairs.

When the GlueServer architecture is applied to a specific environment, one or more GlueServers can be used. For small GlueModels (less than 50 fragment-agent pairs), one GlueServer can be used. If larger GlueModels are demanded, several GlueServers should be used. By using several GlueServers, it will be easier to maintain the GlueModels. This is because each GlueModel will be smaller, and it is more likely that the maintenance of the GlueModel will be done by a person that has more local knowledge of how people cooperate for a given group. In this way, area of concern for maintaining the GlueModel can be distributed according to how people are organised. Using sev-

```

<fragment-agent-pair>
  <agent agent-class="agents.Coordination" amp-id="ExternalAMP">
    <interaction-type>coordination</interaction-type>
    <result>local|external</result>
  </agent>
  <fragment fragment-id="DG/report">
    <reaction>
      <result>local</result>
      <action fragment-id="DG/implement_changes" body="change_and_reexecute_process_fragment_PFNUMBER">
      </action>
      <result>external</result>
      <action fragment-id="DG/implement_changes" body="halt_process_fragment_PFNUMBER"></action>
    </reaction>
  </fragment>
</fragment-agent-pair>
<fragment-agent-pair>
  <agent agent-class="agents.Coordination" amp-id="ExternalAMP">
    <interaction-type>coordination</interaction-type>
    <result>local|external</result>
  </agent>
  <fragment fragment-id="MPG/replanning">
    <reaction>
      <result>local</result>
      <action fragment-id="MPG/estimate_resources" body="change_and_reexecute_process_fragment_PFNUMBER">
      </action>
      <result>external</result>
      <action fragment-id="MPG/estimate_resources" body="change_and_reexecute_process_fragment_PFNUMBER">
      </action>
      <action fragment-id="DG/implement_changes" workspace-id="EXTERNAL"
        body="move_process_fragment_PFNUMBER_to_workspace_WSNUMBER">
      </action>
    </reaction>
  </fragment>
</fragment-agent-pair>

```

Figure 8. GlueModel for scenario 2

eral GlueServers will also shorten the GlueServer response time. Since both AMPs and GlueServers provide cooperative support between workspaces, one GlueServer per AMP could typically be used. A GlueServer should be maintained by higher management, for instance a project manager should be responsible for a GlueServer serving a project team. However, all people affected by a GlueModel must participate in order to create a useful GlueModel. The GlueModel will evolve over time, and hopefully reflect the cooperation protocols between actors involved.

The GlueServer was relatively easy to implement, and it runs on Java Virtual Machine, with OrbixWeb and Sun's XML package installed. Agent-interaction is provided through OMG's MASIF standard, making it possible for the GlueEngine to interact with other agents systems. Results sent back from the agent system is translated from KQML to XML in the GlueServer. Multiple workflow interfaces can be added to the GlueServer through CORBA, CGI-calls or Java RMI.

6 CONCLUSION

We have presented a framework for giving flexible process support to cooperative software engineering processes, combining software agents with traditional workflow systems. The framework consists of a GlueModel specifying process-agent interaction and a GlueServer providing integration of agent systems with workflow systems. Some real-world CSE scenarios suggest that the mechanism is conceptually concise, and practically useful. We are cur-

rently working on a validation of the GlueServer framework, where other real-life CSE scenarios are modelled using other existing systems as well. From this work we would gain more experiences in the advantageous and disadvantageous our framework offers.

ACKNOWLEDGEMENT

We want to thank to Bjørn Haakenstad for implementing the GlueServer, and executing the scenarios. The CAGIS project is sponsored by the Norwegian Research Council's Distributed Information Systems (DITS) Programme.

REFERENCES

- [1] Jin W. Chang and Colin T. Scott. Agent - based Workflow: TRP Support Environment (TSE). In *Fifth International World Wide Web Conference*, Paris, France, May 1996.
- [2] J. E. Doran, S. Franklin, N. R. Jennings, and T.J. Norman. On cooperation in multi-agent systems. *The Knowledge Engineering Review*, 12(3):309–314, 1997.
- [3] Tim Finin, Richard Fritson, Don McKay, and Robin McEntire. KQML as an Agent Communication Language. In *Software Agents (ed. by J.M. Bradshaw etc.)*, MIT Press, 1997.
- [4] Yaron Goldberg, Marilyn Safran, William Silverman, and Ehud Shapiro. Active Mail: A Framework for

- Integrated Groupware Applications. In D. Coleman, editor, *Groupware'92*, pages 222–224. Morgan Kaufmann Publishers, 1992.
- [5] Anders Aas Hanssen and Bård Smidsrød Nymoén. DIAS II - Distributed Intelligent Agent System II, January 2000. EPOS TR 372 (diploma thesis), 324 p., Dept. of Computer and Information Science.
- [6] C. Liu and R. Conradi. Process View of CSCW. In Proc. of ISFST98, *Ocon Technology Application*, pages 46–51, Bremen, Germany, 15-17 September 1998. International Workshop on Intelligent Agents in Information and Process Management.
- [7] Ernst Lutz, Hans von Kleist-Retzow, and Karl Hornig. MAFIA - An Active Mail-Filter-Agent for an Intelligent Document Processing Support. In S. Gibbs and A.A. Verrijn-Stuart, editors, *IFIP*, North-Holland, 1990. Elsevier Science Publishers B.V.
- [8] Zakaria Maamar and Bernard Moulin. An agent-based approach for intelligent and cooperative systems. In *Knowledge and Data Engineering Exchange Workshop*, pages 19–26. IEEE Computer Society Press, 1997.
- [9] 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.
- [10] Michael Merz, Boris Liberman, and Winfried Lamersdorf. Using Mobile Agents to Support Interorganizational Workflow-Management. *International Journal on Applied Artificial Intelligence*, 11(6), September 1997.
- [11] W.J. Orlikowski. Learning from Notes: Organizational Issues in Groupware Implementation. In *Proceedings of the Conference on Computer-Supported Cooperative Work, CSCW'92*, pages 362–369, Toronto, Canada, 1992. The Association for Computer Machinery, ACM Press.
- [12] Geir Prestegård, Anders Aas Hanssen, Snorre Brandstadmoén, and Bård Smidsrød Nymoén. DIAS - Distributed Intelligent Agent System, April 1999. EPOS TR 359 (pre-diploma project thesis), 396 p. + CD, Dept. of Computer and Information Science, NTNU, Trondheim.
- [13] Terje Salvesen and Jan Waage. DIAS III - Distributed Intelligent Agent System Using JavaSpaces, April 2000. EPOS TR 390 (pre-diploma project thesis), 118 p. + 21 p. App.
- [14] Sunil K. Sarin. Object-Oriented Workflow Technology in InConcert. In *Forty-First IEEE Computer Society International Conference: Technologies for the Information Superhighway*, pages 446–450, Santa Clara, California, February 25-28 1996.
- [15] J.W. Shepherdson, S.G. Thompson, and B.R. Odgers. Cross Organisational Workflow Co-ordinated by Software Agents. In *Workshop on Cross-Organisational Workflow Management and Co-ordination*, San Francisco, USA, February 1999.
- [16] 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.
- [17] Alf Inge Wang. Experience paper: Implementing a Multi-Agent Architecture for Cooperative Software Engineering. In *Twelfth International Conference on Software Engineering and Knowledge Engineering (SEKE'2000)*, Chicago, USA, 6-8 July 2000.
- [18] 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.
- [19] Alf Inge Wang, Chunlian 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.
- [20] WfMC. Workflow Management Coalition - Terminology & Glossary. Technical report, The Workflow Management Coalition, February 1999. Document Number WFMC-TC-1011.
- [21] WfMC. Workflow Standard - Interoperability WfXML Binding. Technical report, The Workflow Management Coalition, May 1 2000. Document Number WFMC-TC-1023, see <http://www.wfmc.org>.
- [22] M. J. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.