Using Multi-Agent Systems for Change Management Processes in the Context of Distributed Software Development Processes

Kolja Markwardt, Daniel Moldt, Sven Offermann and Christine Reese

Department of Informatics, University of Hamburg, Germany

Abstract. Today software engineering is facing the problem of the development of distributed software systems. Due to distribution these systems inherit specific problems that need to be tackled during the development. Our approach to handle the problem is to provide an integrated development environment that is based on clear and powerful concepts and which allows to structure a system in a domain-oriented way. As the conceptual basis we apply agent-oriented Petri nets. For the practical part of the control of such applications we use agent-based workflow management system (WFMS) technology. In this paper we illustrate for Change Management, as an important part of the development process, how to apply our approach.

1 Introduction

Large-scale software projects today are concerned with the development of distributed software products. Additionally through globalisation and business process outsourcing within the software industry, development activities as well are more and more distributed between different organisations and even geographically.

Therefore, the development of software is in itself a process that can and should be supported by distributed software systems. Our intention is to provide the concepts and tools for distributed software development environments.

As the conceptual background we use object-based, high-level Petri nets called reference nets (see [5]). To impose structures and patterns on our models we use the agent paradigm (see [4]). A basic practical platform for such systems is a Multi-Agent System based workflow management system (WFMS) as described in [10]. Important contributions here are to build our approach on the integration of reference nets to ensure true concurrency and to impose a structure on multi-agent based systems by workflow concepts as the main structuring mechanism. To explain the background for the application of our approach we will shortly introduce this system and describe how it can be used as a basis for the development of new applications.

Ultimately the goal of this development is the aforementioned development environment. As a first case study we focus on one aspect of distributed software development, the field of Change Management (CM). This is a central field of interest in the software life-cycle, it involves different parties and should be handled in a well structured and reproducible way to minimise the danger of introducing new problems in the process.

For these reasons it is a good example for an inherently process-based, distributed application that can be coordinated by a WFMS.

Section 2 introduces the WFMS we built, the technical background and the way, applications can be built on top of a WFMS. Section 4 focuses on the actual Change Management application, defining the term Change Management as it is used here and showing the development of a process-based application based on a domain-specific problem formulation.

2 Description of the Distributed WFMS

First we will describe the technical foundations, on which the system is built, quickly introducing reference nets, our Multi-Agent System platform Capa and then describing the WFMS built on top of it.

Reference nets. Reference nets are a higher level Petri net formalism. Petri nets allow a formal specification of processes with a graphical representation which is especially well suited to express concurrency and independency. Coloured Petri nets allow complex tokens and an expressive inscription language, using objects and method calls in some other programming language. Reference nets add the concept of net instances, the concept of synchronous channels and the concept of nets within nets while providing a tight integration with Java (as an inscription language).

Net instances have a similar relation to net templates as objects to classes. Synchronous channels allow the bidirectional information exchange (like tokens) between two or more transitions in one or more net instances. The applied transitions are synchronised and fire atomically as one merged transition would fire. Nets within nets is a concept that allows to hold (a reference to) a net instance as a token on a place. This allows to model complex systems as parts that are contained in each other. Synchronisation and information exchange happens vertically through synchronous channels. Reference nets are used for many years as a modelling language and as a programming language in different settings.

The Capa Agent Platform. Using reference nets and Java we implemented a standards-compliant agent platform called Capa. Capa is modelled in layers: The agent platform provides basic services like sending and receiving messages using standard-compliant communication channels like HTTP, the agents on the second level are contained within the platform. An agent contains some nets on the third layer which make up the agent's behaviour, i.e. a knowledge base and protocol nets. The knowledge base provides processing intelligence held in a decision component besides knowledge in the form of keys and Java objects. Protocol nets are workflow-like specifications of sub-processes. Protocol nets are instantiated either in response to a received message or proactively.

Distributed WFMS. Using reference nets and Capa we developed a distributed workflow management system (WFMS) presented in [10]. A distributed system, as assumed here, is made up of several local systems which are combined to build a virtual system on top of them. Our system architecture is illustrated in Figure 1. It implies that an appli-

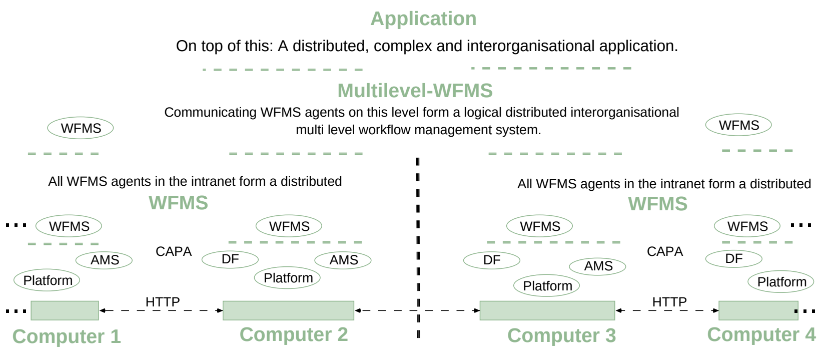

Application
On top of this: A distributed, complex and interorganisational application.
Multilevel-WFMS
Communicating WFMS agents on this level form a logical distributed interorganisational multi level workflow management system.
WFMS
WFMS
All WFMS agents in the intranet form a distributed
WFMS
All WFMS agents in the intranet form a distributed
WFMS
WFMS
WFMS
WFMS
WFMS
CAPA
CAPA
AMS
DF
AMS
DF
AMS
DF
Platform
Platform
Platform
Platform
HTTP
HTTP
Computer 1
Computer 2
Computer 3
Computer 4

Fig. 1. Infrastructure as provided by the distributed WFMS.

cation such as a Change Management system is built on top of this virtual WFMS. The local WFMS can be implemented with reference nets as we have shown in [3]. Other implementations just have to fulfil the APIs. In our own environment we use an agent-based implementation of this WFMS, which enhanced the old version by the features of multi-agent based systems (MAS). Each WFMS agent is composed of inner agents. The inner agents encapsulate the functionality and the outer one bundles their functionality. Different WFMS agents are combined and represented by another WFMS agent one level higher. On top of this infrastructure, a distributed application can operate.

This described local WFMS can be used in a distributed environment because it provides the agent interface and thus receives asynchronous messages via the Internet. It can also be distributed by migrating the participating agents to other workflow management system platform agents. In the following this is especially interesting for our concept of Task Agents which bridge between WFMS-internal agents and application-specific or User Agents. A further possibility for distribution is to integrate several such local systems into a distributed one using an agent for remote communication. The whole distributed system works then like one workflow management service. Figure 1) gives an overview of the described infrastructure: the communication channel at the bottom is provided by CAPA. For CAPA, the platform agents are shown (*directory facilitator* (DF), *agent management system* (AMS) and the platform agent itself). The WFMS agent is a platform agent itself, containing nested agents as described above what is not detailed in the figure.

3 Building Applications

On top of the WFMS it is possible to build process-based, distributed applications. These applications use the WFMS as an engine to execute a workflow process definition. The application consists of several different aspects defined according to the application domain. Standard means of capturing these aspects like UML or BPEL can be used to gather the requirements which are then translated into the corresponding aspects of the Multi-Agent System. In [10] the tool set which can be used in our actual running environment is described.

In whole, the definition of a new application consists of the roles and agents that execute the process, the domain objects used, the client applications invoked in the process, the workflow process and other, non process-related aspects.

Roles. An important step in designing an application is to decide which types of users, or roles, will be working with it. This information can be drawn from use case diagrams and makes up the roles in the process definition later on.

Domain Objects. The business objects needed in an application are usually modelled using methods like use cases and scenarios. For the use of these objects in a Multi-Agent System a common ontology needs to be defined so that all parties involved can share a common vocabulary of concepts.

For the definition of this ontology, the tool Protégé [9] can be used. Our custom plug-in for Protégé then provides the generation of Java classes that can be used in FIPA-compliant ACL-messages between agents because they include application specific agent message handling methods.

Client Applications. Each step in the process requires some agent to execute a certain step to handle a task. Some tasks can be executed without user interaction, these are called automatic tasks and can be provided by specialised agents within the Multi-Agent System or via a Webservice gateway (see [7]). Client interaction tasks require a human user to interact with the system and execute some kind of client application. These types of interaction can be standard tasks that are reusable among different processes or special tasks custom programmed as agents of their own.

Users access the system via a User Agent. It presents the worklist to the user and allows them to choose activities and execute them. When a user requests to execute an activity, he receives a description of the task at hand. This consists of the type of client application and usually associated data. A number of standard tasks are already implemented in a special User Agent concept, while more complex client applications can be deployed as specialised Tool Agents [6], such as the Task Agents mentioned above.

Standard Tasks For simple tasks that occur frequently in different processes, task handlers are already implemented in the User Agent and can be used immediately. The most important ones are:

– Simple Confirmation Task: No action in the system needs to be taken by the user. Instead with this task, the user signals the completion of an activity outside the scope of the application. No data needs to be exchanged.
– Choice Task: Similar to the Simple Confirmation, this task is used to get a single decision from a user that determines the the further control flow of the process.
– Form Task: Many types of user interaction can be broken down into filling out forms, so this client application allows the presentation of a form to the user to gather information in a formalised way. To describe the layout of the form and the data to be entered a special form ontology is used.

Task Agents If these standardised tasks are not sufficient to model the interaction needed, a specialised Task Agent can be deployed that enhances the capabilities of the User Agent. The Task Agent is created on the agent platform of the WFMS and migrated to the platform of the User Agent if necessary.[6]

Once the Task Agent is located in the same Java VM as the User Agent, Java object references can be used to plug into the User Agents interface. A Task Agent can give instant feedback to the user about data entered and check certain conditions before reconnecting to the server and trying to complete the activity. For more complex client interaction this is therefore the preferred way.

Workflow Process Definition. Since both, the WFMS as well as the underlying Multi-Agent System, are implemented using Petri Nets as an implementation language, it is an obvious choice to use a Petri Net formalism for the process definition as well.

The workflow engine used supports special task transitions that bear inscriptions denoting the type of task to execute, the rules for picking resources to execute the task and parameters that can be passed around as tokens within the net. Transitions begin to fire when the execution of a task starts. They can restore the removed tokens if the execution is not successful (rollback functionality), as described detailed in [3]. This section

has shown the way in which process-based applications can be developed based on our framework. It has of course not been able to demonstrate all aspects of developing a complete application. Even in an application that is heavily process-oriented, there will always be parts that are independent of the current process. For example users might want to check data which is not part of a process they are currently working on. These aspects must still be taken care of in extension to our approach. For further discussions of software development based on agent-oriented Petri nets see e.g. [11].

4 Case Study - Change Management

This section describes the development of an application for Change Management according to the principles laid out in Section 3. First the concept of Change Management is introduced and its position within the software life-cycle. Then the requirements for a real world sized CM application are outlined, as we see it in the bigger context of distributed software development.

Afterwards the design and implementation of a simpler application is shown to point out the procedure for the development. Finally some notes will be made on how to further develop the system according to real world requirements.

4.1 Change Management

In a productive environment the requirements an application needs to fulfil can change very frequently and make adaptations necessary. This applies to software, organisation and hardware changes, however, we will focus on software changes.

These adaptations unfortunately can introduce new problems into the software, therefore great care needs to be taken to devise the change process as reliable and reproducible as possible to minimise these risks. The IT Infrastructure Library (ITIL [8]) classifies Change Management as a part of service support activities. The main goal is to ensure effective and on-time implementation of changes. One way to achieve this in a transparent way is to use computer supported systems for Change Management. These systems need to be distributed to support the coordination of the different involved parties. CM is a central process for anyone dealing with software development (see [1]). The challenge here is to support CM in a distributed environment. Our solution which will be discussed here will be based on the use of a MAS-based system to support the requirements of adaptability and flexibility of such processes.

4.2 CM Application in Software Development

Change Management requires the coordination of the different stake holders in the application from the first notion of a deficiency in the software to the final implementation of a solution. This process in a way mirrors the original software development process and therefore can be as diverse as software development process models. So it needs to be customised to the individual needs. The process described in the following is based on the actual process used for a medium sized web application in the B2B environment.

CM Process. The process can be broken down into three phases: Assessment, realisation and implementation. During the assessment phase a proposed change is classified with the impact on the existing system, priority, benefit etc. by the department(s) using the software or the IT coordination department. If the change seems to be worthwhile, the developer is asked for an offer on the realisation of that change. If that offer is acceptable the change is authorised for realisation.

Realisation of the change is then given to the developer. It could be broken down into several tasks based on the organisation of the developer, on the software and the type of change. Once the realisation is done, the task of testing it is given to the client organisation. If the test is not successful, the developer has to remedy this until acceptance can be reached. Finally the changed software needs to be deployed to the production environment.

Roles. The roles as much as the process reflect the organisational conditions in which the system is used. Again, a sample set of roles is described here, that usually are involved in the CM process.

On the client side there is a number of people that can enter new Change Requests (CR) based on user feedback or their own experience with the system under consideration. A change manager categorises the CRs and supervises the CM process. To determine concepts usually is also a task of the change manager. The change advisory board (a committee proposed by ITIL to decide on changes [8]) gives authorisation to perform a change and can thus be integrated into the system as well. Finally, the role of a tester needs to be filled.

The service provider needs roles for submitting offers and another role for the actual realisation of a change. Additionally a role for internal testing can be added. Implementation of the changed software onto the operating system can be performed by the client organisation itself, the developing company or another service provider, depending on the configuration given.

The mapping of roles to users is done via an administrative interface not covered here, the actual decision which users to present a certain work item is based on rules defined with the tasks.

Business Objects. The central object in a CM system is the change request (CR). It holds all information regarding a certain change and is the base for all tasks in the process.

A CR object holds a) basic information like title and description, b) classification like priority, type, and severity, and c) additional information gathered throughout the process, like comments by the users involved, associated documents etc.

Which information exactly is needed here depends very much on the organisation using the system, since some of this information might concern organisational specifics like the integration with an ERP system. Some information might also be only accessible to certain types of users, like rating and billing information.

4.3 Developing an Example Application

The organisational environment in which Change Management occurs, can be very diverse. Therefore one CM process that fits everyone does not exist. In the last section we have described a CM application that might be used in a production context, to keep things simple however, a smaller, simpler example will be presented here.

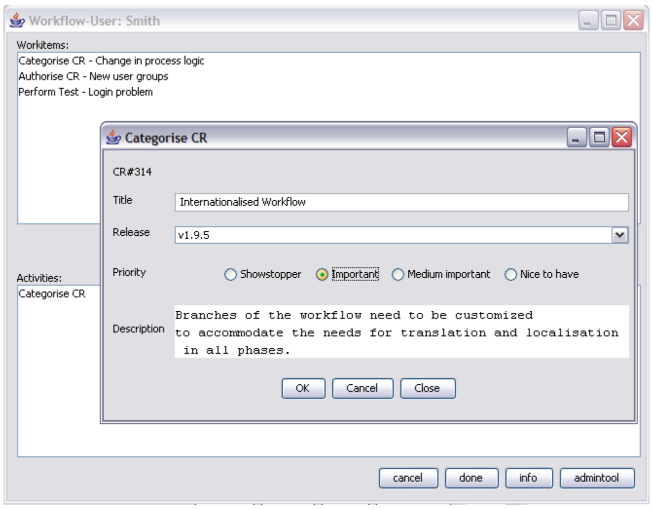
CM Process. The example application for Change Management is realised as a basic prototype, the structure of the agent-based WFMS allows easily adding further complexity as needed later. Therefore the used workflow is initially very simple.

The user of a software product detects some issue in this software he thinks should be resolved. He starts a new instance of the workflow and enters the basic data of the change. Once that task is finished, the workflow continues to the developer who gets the task of resolving the issue. Once that is done, the first user is notified again of the resolving and the workflow completes.

Roles. The process contains only a small number of roles, the customer, developer and the IT department. As tasks get more refined also new roles can be defined, for example representatives from the departments using the software, IT coordination or the change advisory board in the subprocess of generating the requirements of a change.

Ontology. For the example the CR object needs only a couple of fields, this has to be enhanced for any real life applications of course. But for now, the CR object can be limited to the fields: CR-Id, title, description, severity and CR-type. Severity and CR-type can be implemented as types of their own to limit the possibilities here and provide some common values to refer to.

Client Applications. For the simple example here, no new client applications have been defined. Instead the existing standard applications can be used to model all aspects needed here. Figure 2 shows a form task in execution. The definition of the form is done using the standard form ontology, which describes the elements of the form. The mapping of form elements to information in the CM ontology is done by the WFMS Agent.


Workflow-User: Smith

Workitems:
Categorise CR - Change in process logic
Authorise CR - New user groups
Perform Test - Login problem

Activities:
Categorise CR

Categorise CR

CR#314

Title: Internationalised Workflow

Release: v1.9.5

Priority: ( ) Showstopper [✓ Important] ( ) Medium important ( ) Nice to have

Description:
```
Branches of the workflow need to be customized
to accommodate the needs for translation and localisation
 in all phases.
```

Fig. 2. The personal agenda of one user with a form task.

4.4 Building on the Example

As stated above, each organisation has their own requirements towards a CM system. Therefore one of the most important criteria to measure against is the flexibility and the ease of adaptation to different requirements. It has been shown how a sample CM application could look like. In the following we sketch how to adapt such a system to the specific needs of another organisation.

Each of the tasks in the process definition could be further refined, introducing new activities and more complex processes with choices, iterations and concurrent tasks. It is even possible to fragment the workflow onto different WF engines (see [2]).

Process Definition. The first aspect to focus on is the CM process needed for the organisation. Process definitions can be edited with the workflow plug-in of the Renew tool and uploaded to the workflow definition agent, if the user has the appropriate rights.

Roles. New roles to use in workflow processes can be defined using the administration interface of the User Agent. Special protocols exist that make these known to the administration agent.

Client Applications. For more complex applications it will not be sufficient to just use simple tasks like those shown in the example. New client applications can be developed as Task Agents[6] and deployed onto the running platform. Once they are made known to the directory service, they can be used in process definitions.

5 Conclusion

In this paper we have presented a Multi-Agent System based WFMS as a framework for the development of process-based applications. The procedure for this kind of development has been shown on the example of a Change Management system.

The general goal of a distributed software development environment has been illustrated by this description of a special part of the software development process which has to be supported in such an environment.

Further challenges are still on the one hand the conceptual aspects of distributed software systems and especially distributed software development environments. On the other hand the practical requirements in constantly evolving infrastructures like the Internet, Semantic Web, Grids etc. the development of usable software will require new ways to handle central aspects of software development like distribution, concurrency, non-determinism, mobility, adaptability. In future work we will extend our tool set in the direction of a distributed development environment to cover critical parts of software development and maintenance.

References

1. Claudio Bartolini and Matthias Sallé. Business driven prioritization of service incidents. In Akhil Sahai and Felix Wu, editors, *Utility Computing: 15th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, DSOM 2004, Davis, CA, USA, November 15-17, 2004. Proceedings*, volume 3278 of *LNCS*, pages 64–75. Springer, 2004.
2. Timo Carl. Entwicklung eines agentenbasierten verteilten Workflow-Management-Systems mit Referenznetzen. Diplomarbeit, Universität Hamburg, Fachbereich Informatik, 2004.
3. Thomas Jacob, Olaf Kummer, Daniel Moldt, and Ulrich Ultes-Nitsche. Implementation of Workflow Systems using Reference Nets – Security and Operability Aspects. In Kurt Jensen, editor, *Proc. of CPN*, 2002. DAIMI PB: Aarhus, Denmark, August 28–30, number 560.
4. Nick R. Jennings. On agent-based software engineering. *Artificial Intelligence*, 117(2):277–296, 2000.
5. Olaf Kummer. *Referenznetze*. Logos, Berlin, 2002.
6. Kolja Lehmann and Vanessa Markwardt. Proposal of an Agent-based System for Distributed Software Development. In Daniel Moldt, editor, *Proc of MOCA 2004*, pages 65–70, Aarhus, Denmark, October 2004.
7. Sven Offermann, Jan Ortmann, and Christine Reese. Agent based settler game, 2005. Part of NETDEMO, demonstraion at international conference on Autonomous Agents and Multi Agent Systems, AAMAS-2005.
8. Office of Government Commerce (OGC), editor. *IT Infrastructure Library (ITIL)*. The Stationary Office, Norwich, UK, 2000.
9. Protégé homepage. `http://protege.stanford.edu/`, 2006.
10. Christine Reese, Kolja Markwardt, Sven Offermann, and Daniel Moldt. Distributed business processes in open agent environments. In *Accepted at: International Conference on Electronic Information Systems (ICEIS) 2006*, 2006. Accepted paper.
11. Heiko Rölke. *Modellierung von Agenten und Multiagentensystemen – Grundlagen und Anwendungen*, volume 2 of *Agent Technology – Theory and Applications*. Logos Verlag, Berlin, 2004.