Supporting Virtual Software Projects on the Web

Kari Alho, Reijo Sulonen
Helsinki University of Technology
TAI Research Centre
PL 9555
FIN-02015 TKK, Finland
Email: Kari.Alho@hut.fi, Reijo.Sulonen@hut.fi

Abstract

A growing share of all software development project work is being done by geographically distributed teams. To satisfy shorter product design cycles, expert team members for a development project may need to be recruited globally. Yet to avoid extensive travelling or replacement costs, distributed project work is preferred.
Current-generation software engineering tools and associated systems, processes, and methods were for the most part developed to be used within a single enterprise. Major innovations have lately been introduced to enable groupware applications on the Internet to support global collaboration. However, their deployment for distributed software projects requires further research. In particular, groupware methods must seamlessly be integrated with project and product management systems to make them attractive for industry.
In this position paper we outline the major challenges concerning distributed (virtual) software projects. Based on our experiences with software process modeling and enactment environments, we then propose approaches to solve those challenges.

1. Introduction

For a number of reasons, a growing share of all software development work is being performed by geographically distributed organizations, teams, or individuals. The production of competitive software products today requires one to skillfully put together talented work of a large number of professionals from various fields. In many cases, it is not reasonable to assume that all project team members can be found from the local office. Many strategies exist to save costs and shorten development cycles, such as utilizing off-the-self components, subcontracting, or distributed and joint projects. To make these new strategies effective, new management and information technology approaches are needed.

Current distributed projects often incur substantial overhead in time and cost due to increased communications and management effort, misunderstandings, rework, or lack of standards. Often this even results in lower quality in the results. In order to really save costs, (e.g., through decreased travel or increased parallelism in the development process), better methods and tools to manage distributed development projects are needed. We will call such projects *virtual projects*, since they utilize resources in many organizations, thus forming one kind of a virtual enterprise.

We do acknowledge that many of the problems and challenges associated with virtual software projects are not primarily of technical nature, but instead deal with human behavior. For instance, the organizational processes enabling team formation and the management of a distributed process during its lifetime have significant impact to the project results. In this paper, however, our treatment is technical, and process issues are only discussed in the context of modeling and enactment techniques.

The software development and project management tools used today in companies were designed to operate well within a single organization. Development and management processes are also typically modeled to occur within one enterprise; perhaps excluding customer processes and simple subcontracting.

Innovative collaborative applications for the Internet are emerging increasingly, so that supporting global collaboration is now enabled in simple application areas. However, the deployment of groupware collaboration for software engineering requires further research. In particular, collaboration must seamlessly be integrated with project and product management systems to make them attractive for industry.

Workflow systems have made it possible to tie together simple applications to accomplish more complicated tasks [1]. However, most current workflow suffer from a number of deficiencies, which have hindered their widespread adoption in the product and software development domain. First, the workflow model, once specified and started for execution, cannot typically be modified on the fly. This makes it impossible to model and execute ad-hoc processes, a necessity in many dynamic businesses. We firmly believe that a process model

should be allowed to be modified and specialized even after it has started execution.

Second, current workflow environments typically include a built-in process modeling tool with it's own formalism, often the only means to specify the process. It is unrealistic to expect that one tool or formalism could offer all the features needed to model all processes of say, a large multi-national or virtual enterprise. In practice, there already exists a large number of process models in the organization, and it would be a waste of effort to re-model most of these.

Third, many workflow systems are implemented so that the resulting workflow can only be executed in a closed homogeneous computing environment, typically a local area network. This makes it difficult to successfully apply workflow technology in large organizations, where heterogeneous environments do exist. With current workflow tools it would be quite difficult to define and execute processes spanning several organizations, like in virtual enterprises. However, recent advances in networking and computing technologies (e.g., Java and agent technologies) have enabled new approaches for extending the capabilities of workflow execution.

This paper introduces our approach for finding solutions to these challenges. Our research group has already implemented some software components enabling the definition and execution of dynamic workflows (ones that can change during their execution) [2]. We have also ported the process engine to Java, so that it is now able to support distributed process execution in the Internet through a central server and client applet architecture.

Now, our aim is to extend these components so that process models from different sources (e.g. commercial project planning tools) can be utilized in the process support environment through model converters. To further support the definition and execution of workflows on autonomous computing environments (like in virtual enterprises) is another major goal of our work.

The rest of this paper is organized as follows: in Section 2 we position our work among other efforts to manage virtual software development process. We also propose our project integration model. Section 3 outlines the conceptual framework we use for process modeling and enactment. Section 4 gives an overview of a system architecture in which such a framework has been implemented. Section 5 discusses related work, and finally Section 6 gives conclusions and outlines our future work.

2. Background

Managing and performing distributed projects requires various skills from the personnel, including communications, management, teamwork, and technical skills. Likewise, an effective technical solution consists of many individual technologies, tools and methods woven together. Background for these individual solutions may well come from different scientific fields, so virtual software projects are a good candidate for cross-disciplinary research. At least the following fields could be involved:

- Software Engineering (process modeling),
- Artificial Intelligence (agent technologies),
- Product Data Management, and
- Data Communications (Internet).

The parties taking part in a virtual project can be autonomous and geographically distributed. They can utilize different hardware platforms, operating systems, communication protocols, and software tools. This implies that tightly integrated, technology-specific IT solutions are not acceptable.

We propose a technology model for collaboration in virtual projects which is based on loose integration of process, product and the collaboration platform. Integration is needed to support collaborative work and enable the transfer of product data and project information. Yet the individual organizations taking part in the collaboration are autonomous, and we cannot assume that they would drastically change their internal working methods to support virtual projects.

2.1. Process integration

Process models and their execution (or enactment) continues to be an active research area worldwide. A great deal of the work has been invested on devising improved process model formalisms for expressing various aspects of a business or engineering process. However, since process models have so many different uses (e.g., to understand a process, simulate, analyze, or execute it with a computer system), we feel that even today no single model formalism is good enough for all purposes.

Our approach is to develop a "common denominator representation" (CDR) of many process modeling formalisms and use it as a platform for joining different modeling formalisms, such as Petri Nets, statecharts, or Gantt charts. To integrate specific models, one has to map the specific formalisms to the CDR (and vice versa in some applications). The CDR model—which we discuss more in Chapter 3—includes abstractions of *activities*, *artifacts*, and *agents* with associated behavior and relationships.

Some specific problems will arise when we try to map process formalisms not intended for execution (such as an IDEF0 chart) into the CDR, which does include execution semantics. Typically, we must use heuristics to

include a predefined execution model (a statechart) into the converted process model.

On a higher level, it is important that all parties understand and interpret a process in a similar manner. A useful method to improve the understanding is to develop a shared ontology of the needed concepts.

2.2. Product integration

A shared understanding of the structure and properties of the developed product (i.e., software) is vital to the success of a virtual development project. However, as stated earlier, we cannot assume tight integration, like a common file system. In the presented Software Workmate architecture we propose that the process engine only knows about URL-kind references of the artifacts, and specific *artifact servers* would handle the specifics of data access [3]. In a virtual project environment it would be natural to assume that the artifacts would mostly be accessed from the WWW (through a WWW artifact server).

2.3. Platform integration

Workflow models need a strong supporting infrastructure to enable their controlled evolution and execution and integration to other computing resources. In our earlier effort, a preliminary version of such an infrastructure was built based on CORBA architecture [3]. Recent advances in Java and other internet technologies have made them viable solutions for integrating and utilizing computing resources company-wide or even across organizations. We do believe that in the future these technologies will provide a good basis for distributed process management.

Based on the earlier work, we have re-implemented the process engine (PE) component in Java. Client applications (or applets) can communicate with the PE through Java Remote Method Invocation (RMI) protocol. This means that it is now simpler to integrate process support into Java applications and applets since separate middleware is no longer needed.

3. Process Modeling and Enactment Framework

In this section we present in more detail the common denominator representation (CDR) model for process modeling and enactment. The model, as introduced in Chapter 2.1, consists of three basic entity classes:

- *Activities*: the basic units of work. These units can be assigned, ordered, scheduled, etc.
- *Artifacts*: the things created, modified and used by the activities.
- *Agents*: the "doers", that is, who perform the activities. These can be human beings or software programmed to perform specific activities.

Each entity class has class-specific attributes and relationships as well as methods for manipulating both the attributes and relationships. In addition to these, the framework implements a general attribute model allowing the user to define his own attributes (per instance) as needed.

When representing a real process with the aid of the entities in the framework, the entity classes are instantiated into objects that represent specific real-world entities.

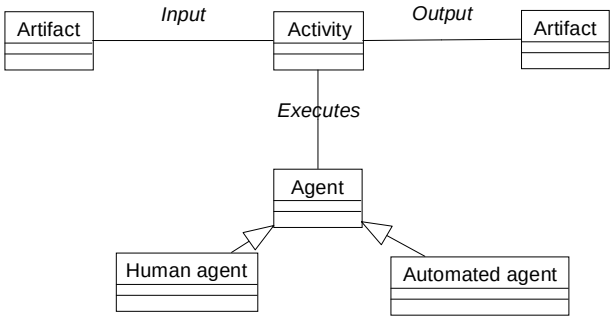

Figure 1: Conceptual Framework

The entities and their relationships are depicted as a UML class diagram in Figure 1. The rectangles represent the entity classes, and the lines named relations. In the UML notation, an arrow represents specialization, so in our case, both Human agent and Automated agent are specializations (i.e., subclasses) of Agent.

An entity class instance may also have named relations to other specific classes with a general relationship mechanism. For example, this can be used to model causal activity relationships.

To model behavior of the entities, all the three basic entities can have an associated FSM, which describes the life cycle model of the entity. In most cases these can be freely defined, but in order to support automated enactment, we do use a system-defined FSM to model a fixed part of the activity behavior.

The basic entities can be grouped together to form more meaningful structures for process modeling, enactment, and re-use. We define *process* as a uniquely named set of activity, artifact, and agent instances. Processes can form hierarchies, so that one process has a set of subprocesses. Another kind of collection of the three basic entities is called a *project*. A project represents a

process, which is ready to be enacted, so it must fulfill the following conditions:

- There must be at least one agent associated with every activity of the project.
- Every artifact of the project must have a physical representation.

A process can be *activated* to make a new project of it. The activation method checks and, if necessary, fulfills the above conditions. One basic entity can belong to many processes or projects, so that shared activities can be represented. Processes and projects can form hierarchies, so that one process has a set of subprocesses, and one project has a set of subprojects. The activation method preserves a defined hierarchy.

4. Architecture

The main architectural components of the system are an (optional) Object Request Broker (ORB), the artifact servers, the process engine, and the client applications. The components are depicted in Figure 2. Server components are displayed below the ORB and Java RMI level and client components above it.

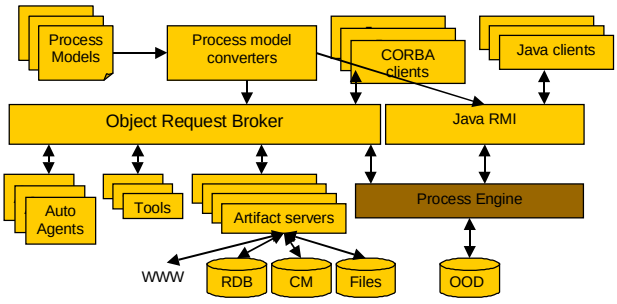

Figure 2: Components of the Architecture

An ORB is the central middleware component which makes the locations of system servers largely invisible to the clients within a network. It also can make the services available on heterogeneous networks and client computers. The Java RMI can also supply these services, provided the affected components are implemented with the Java language.

The artifact servers take care of storing the data of the artifacts in a system-specific way and serving access requests of the other system components through a standard protocol.

The process engine is the central service component, which is used to store and execute process models in the CDR format. It receives service requests through its API (defined in CORBA IDL or Java) and stores process descriptions and enactment data into its internal object database.

Client applications include model converters from other process formalisms and clients used to track and monitor process enactment. In a virtual project environment client applications might also represent agents communicating and negotiating with remote parties.

When storing into one process engine processes that span multiple parties, we must keep in mind that the process engine quickly comes a very critical component in the virtual project. Autonomous parties typically also want to administer their own systems, so ideally we would like to have multiple coordinating process engines. One possible model for decentralized process enactment is proposed in [4], and it seems feasible to utilize the proposed Summit protocol in our enactment environment.

5. Related Work

Armitage and Kellner describe the need for a similar Common Denominator Representation in [5]. Their conceptual schema has many similarities to our model.

Our approach to the architecture of a process support system has many similarities with the ideas presented in [6], where he proposes that the state of software processes should be stored separately from the applications that created that state.

Han has proposed a general architecture for integrating workflow models and resources, e.g., application programs, agents, and documents [7]. The formalism to describe the models is called HOON (Higher-Order Object Nets). The central idea is to arrange Petri net models and their surrounding environment in a client/server manner and model the interfaces explicitly as a set of special places, called interface places. We are currently investigating how the ideas of HOON and the CDR representation can be used together.

6. Conclusions and Future Work

We have outlined ways to extend an existing software process support system so that virtual projects spanning several autonomous parties could be supported. These include:

- Utilizing process models expressed in different formalisms, coming from different parties and integrating them into a common one.
- Using the WWW as a primary medium for information exchange.
- Distributing the project coordination into autonomous agents and process engines.

We expect to utilize our experiences from the earlier systems to gain insight into the virtual project support problem. In addition, we will work closely with our industrial partners.

Testing these ideas will take place within a newly proposed research project titled “GECOS—Global Engineering COrdination Support”. The project is to be funded by the Technology Development Centre of Finland and severeal Finnish product development companies.

References

[1] Georgakopoulos, D., Hornick, M., Sheth, A.: “An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure”, *Distributed and Parallel Databases*, (3)1995, pp. 119-152.

[2] Alho, K., Lassenius, C., Sulonen, R.: “Process Enactment Support in a Distributed Environment”. *Computers in Industry*, 29 (1996), pp. 5-13.

[3] Lassenius, C.: *The Design of the Software Workmate Process-Centered Software Engineering Environment*. Master's Thesis, Helsinki University of Technology, Department of Industrial Management, October 1996.

[4] I. Z. Ben-Shaul and G. E. Kaiser. A Paradigm for Decentralized Modeling and its Realization in the Oz Environment. In *Proceedings of the 16th International Conference on Software Engineering*, May 1994, pp. 179-188.

[5] Armitage, J., Kellner, M. I.: “A Conceptual Schema for Process Definitions and Models”. In Proceedings of the *3rd International Conference on the Software Process, ICSP-3*, pp. 153-165, Reston, VA, October 1994.

[6] Heimbigner, D.: “The ProcessWall: A Process State Server Approach to Process Programming”. In *5th ACM SIGSOFT Symposium on Software Development Environments*, pp. 159-168, December 1992.

[7] Han, Y.: *Software Infrastructure for Configurable Workflow Systems—A Model-Driven Approach Based on Higher-Order Object Nets and CORBA*. Ph.D. Thesis, Technical University of Berlin, July 1997.