Software Process in Virtual Software Corporations

Zoran Constantinescu
zoran@idi.ntnu.no

Abstract

This paper tries to outline some of the challenges concerning software development in virtual software corporations. The communication infrastructure provided by the Internet allows team members to participate in a development effort without being physically collocated. We will try to present some of the requirements for the software process systems for virtual corporations, and make a survey of some of the existing systems supporting this kind of software development over the Internet.

1. Introduction

With recent developments, there is an increased pressure on software engineers to make more competitive software products, to create larger software systems and to deliver them faster, ie. to have shorter design and production cycles. Such projects requires work of a large number of professionals from various fields. Nowadays it is impossible for one organization to possess all the expertise required for their development. It is harder and harder to find these skilled people locally. These led to the idea that major software developments could be carried out by partnerships of organizations, all bringing to the project their own areas of expertise. This new emerging concept in software engineering is called Virtual Software Corporation (VSC). Usually, such a Virtual Software Corporation, or Virtual Enterprise, is composed of development teams from different organizations, who collaborate on specific projects, then possibly form other alliances for other projects. These teams can be seen as a big Virtual Team, which works on the the project. Also these kind of projects are often called Virtual Projects.

One of supporting technology is the Internet, which made it possible to remotely access information across organizational and national boundaries. It supports globally distributed product development so that work can proceed around the clock in different locations. Some of the characteristics of a Virtual Software Corporation are:

- partnerships form dynamically: major difference between this kind of partnership and strategic business alliances, which take a long time to set up
- each partner brings to the partnership a core area of expertise
- large diversity of organization al structures and styles
- the customer could be part of the corporation, enabling it to react quickly to customer needs

- the partners will be based at many different geographically distributed sites (possibly different countries and time–zones)
- participants in the virtual corporation retain a high degree of autonomy over their own development activities, product data, tools and environments.

This paper is structured as follows: Section 2 presents some of the problems and challenges associated with Virtual Softwate Projects development and concludes with some requirements for process modeling in VSCs. A summary of some existing systems is given in section 3. For each system, the most important features are presented (general idea), a brief description of the architecture and the use of the Internet is given, and solutions to the requirements. The section ends with a table showing how each system fulfils the requirements presented in section 2. The last section gives some conclusions.

2. Requirements

Software projects that involve people from physically distributed locations are a challenge both for projects managers and developers. The problems that people are confronted with in such projects involve many different aspects of software engineering as well as technical issues related to Internet. However, many of the problems and challenges associated with virtual software projects are not primarily of technical nature, but instead deal with human behaviour [3]. Some of the most important problems are briefly described in the following.

Human nature problems:

- Team formation: especially difficult if dealing with such geographically distributed members
- Team–work: the interpersonal relations are more difficult to be maintained in larger, distributed environments – formal procedures are required for coordination (lack of coordination: over–committed developers , late delivery)
- Coordination:
- Communication: communication acts which are implicit in small teams, must be made explicit for larger teams (lack of communication: unclear understanding of the process, so inefficient allocation of human resources)
- Management of process: (planning the project, scheduling the project, monitoring the execution of the project) – these processes become distributed – so very complex and hard to manage; these human problems tend to have a more important role than in the current (non–distributed) projects.

Technical nature problems:

- Today's tools are mainly for a single–organization: software development and project management tools used today were design to work in a single organization; the same with the development and management processes
- Existing workflows have deficiencies:

    - they are not at all widespread in the development domain
    - cannot be modified on the fly: can't exec. adhoc processes needed in many dynamic busin. today; and they are not flexible enough to adapt to the changing processes
    - there is no one–tool or formalism with all needed features: different organizations use different process models, and it is unrealistic to think they would change it
    - close environment for executing the workflow: usualy a small, homogeneous computer environment, like a LAN
    - lack of support for knowledge intensive tasks: the knowledge needed for executing the process is not explicitely described in the workflow model
- Use of different hardware and software: the parts involved in a virtual project usualy use different hardware platforms, different operating systems, communication protocols, software tools
- People located in different locations and different timezones, different ontologies, different work ethics

Some of the requirements [8] imposed on software process support for the virtual software corporation are listed and briefly discussed in the following:

R1. Asynchronous work support: people work asynchronously in different locations and at different times. Need for a support system to facilitate the coordination and management of the process.
R2. Synchronous work support: the results of a working day need to be efficiently communicated to the coworkers in other time zones. Need for an easy understandable model of the process, so that an overview on the changes since the last day could be easily provided.
R3. Ubiquitous communication infrastructure: every team member has to be able to connect to the development process without effort. Needed to reduce the set–up problems of the virtual team.
R4. Transparent and fast access to process knowledge: easy access to all relevant project data and knowledge. Should be transparent to the user, but bandwidth and other limitations should be considered.
R5. Distributed configuration management: to maintain several versions of the system and all the informations related to it.
R6. Repository for ontologies: a project repository that defines commonly used terms.
R7. Flexibility of development processes: support for refinement of the project plan and the process model while the development is already in progress (dynamic workflow).
R8. Proactive change notification: changes introduced into the process (because of new/changed requirements) should be proactively sent to team members whose work is influenced by those changes (proactive=acting in anticipation of future problems, needs, or changes).

A few other requirements can be added:

R9. User friendly visual environment: provide a easy visual programming environment to teamwork managers and visual interface to normal team members to carry out the teamwork, without understanding the underlying technology.

R10. Support for process enactment: automation of the software process, or guide the user through the process.

R11. Support for the representation of the software process: a model of the process, by using a PML.

3. *Existing systems*

Some of the existing systems that support software development in a virtual corporation are shortly described in the following. A more comprehensive description of each of them can be found in the references.

3.1 *GECOS*

Approach: to develop a "common denominator representation" (CDR) of many process formalisms and use it as a platform for joining different modeling formalisms, such as Petri nets, statecharts, or Gant charts. The CDR model includes abstractions of activities, artifacts and agents with associated behaviour and relationships. For a specific model, it is necessary to map the specific formalism to the CDR, and vice versa. This is performed using some model converters. Some problems arise when trying to map process formalisms not intended for execution into the CDR, which includes execution semantics.

Earlier versions were implemented using CORBA architecture, but the latest one uses recent Java and Internet technologies. Client applications (applets) can communicate with the process engine (PE) through the Java Remote Invocation protocol (RMI). This offers a very good flexibility, allowing to integrate process support into Java applications and applets very easy.

The model consists of three basic entity classes: *activities*, which are the basic units of work; *artifacts*, the things created and used by the activities; and *agents*, which are the elements performing the activities (could be human beings or software programmed to perform specific activities). To model behaviour of the entities, each of the basic ones can have an associated finite state machine, which describe the life cycle model of the entity. In order to support automated enactment, a system–defined FSM is provided to model a fixed part of the activity behaviour. The basic entities can be grouped together to form more meaningful structures for process modeling, enacting and reuse. In this context, a *process* is defined as a uniquely named set of activity, artifact and agent instances. Processes can also form hierarchies, so that one process can have a set of subprocesses. A process which is ready to be enacted is called a *project*. A process can be activated to make a new project of it. One basic entity can belong to many processes or projects, so that shared activities can be represented.

Support for dynamic workflows, that can change during their execution. Use of Java allows the support for distributed process execution over the Internet, through a central server and client applet architecture.

3.2MILOS

A new process modeling language, which allows to represent knowledge about software development processes. It supports a process−centered structuring of knowledge. It integrates process modeling with project planning and enactment. A flexible workflow engine allows refining and changing process models during project execution. Support for change notification.

The MILOS environment consists of several components:

- the resource pool component: manages agents, roles and agent properties. It allows to represent company structures (eg. in organization charts or hierarchical skill structures).
- the process modeling component: maintains formal process definitions. These include control flow and data flow specifications, as well as pre− and postconditions, process refinement and required skills.
- the project plan management component: supports the project manager in planning and customizing the project.
- the workflow management component: is responsible for executing the project plan and managing products. It generates todo lists for agents and maintains the current project state. The workflow engine is able to react dynamically to project plan changes during execution, without interrupting the enactment flow.

MILOS is implemented in Java using an object oriented database server, providing transaction management and persistency services. The architecture is based on a multi−tier approach: many servers act as components of the system. Clients of the system are web based Java applets using Java RMI, which, together with replication and conflict solving mechanisms, allow distribution of workflow execution. There is implemented a connection to MS−Project, for planning support. Other applications (MS−Word, Rational−Rose) can be invoked within the execution framework.

3.3DHT

A distributed semantic hypertext framework (DHT) that support software development projects within virtual enterprises. A way to achieve software process enactment within a virtual enterprise without centralized mechanisms. The process description is represented as a user−navigable hypertext graph of nodes that links process steps, staff roles and associated tools with designated software products. The approach to project coordination and sharing of project artifacts is implemented in a framework that employs two complementary forms of information integration: *a) logical integration* provides a view of the shared information space based on a virtual central artifact repository to facilitate project coordination; *b) physical integration* provides transparent access to objects that appear in the virtual repository, but are actually stored and managed in autonomous, distributed, heterogeneous repositories. The structure of the virtual repository is described with a semantic hypertext data model.

A software process model has a natural representation as a hypertext graph. Nodes in the graph represent tasks or process steps, while links specify both the order in which the tasks should be performed and the products on which they should be performed. The resulting nodes, links and contexts can be browsed and followed just like other hypertext graphs. In enacting a DHT–based process model, developers perform the assigned task/step using the tool(s) integrated and bound to the task/step, accessing and processing the designated software development artifact(s) as specified according to the developer's role. The mechanism for process enactment is embedded entirely within the process representation as a hypertext graph, together with the source predicates and resolution functions of available–task links. In contrast to other enactment systems which employ an environment accessed through a process–based user interface, process–aware tools, or process state databases to execute process specifications, DHT process models can be enacted simply by browsing the process hypertext using any DHT–compatible Web browser or tool.

The architecture is based on a client–broker–server model that provides transparent access to heterogeneous repositories through intermediary information brokers called *transformers*. Transformers for new repositories can be developed with modest effort (i.e. hours to days), based on reusable server templates. Clients implement all application functionality that is not directly involved in a server's storage management. Thus, a client is typically an individual tool, but may be a complete software development environment. It is impractical to expect developers and organizations to abandon their favorite tools to use new tools that can access a DHT corpus. Therefore, DHT includes a strategy for migrating existing and new tools to the DHT environment, and a configurable cache mechanism to enable alternative approaches for creating access to, and controlling concurrent updates to, collaborative information spaces.

3.4ProcessWeb

It is a system designed to provide active process modeling and co–ordination via the World Wide Web. At its core is the ProcessWise Integrator system, which is a general purpose process enactment system. It provides an environment for enacting process models written in a high level language, PML (Process Management Language). One of the aspects of ProcessWise models is that they are persistent, and will keep running whether or not a particular user is connected. Another aspect is that they are dynamic, and can respond to change by changing themselves. Parts of a process program can be instructed to suspend execution of another part, or even themselves, apply modifications, then resume execution of the model. The aim of the system is to use the World Wide Web as a convenient interface to the ProcessWise Integrator. The user interface to PWI is designed to be simple to program. It is very prescriptive about what the user is permited to do at any time, and is very much oriented towards driving the user.

3.5VISCOUNT

The objective of the project is to develop configuration management tool and process support for use in a virtual software corporation. The project uses the Lifespan configuration management tool, which is based on a tailorable process which ensures that appropriate controls are imposed on all changes to configuration items. Hence an essential part of the development of Lifespan within VISCOUNT will be to enhance its current process support to make it suitable for multi–site, multi–organisational procedures.

3.6*OSS*

The idea behind Open Source Software (OSS) is very simple: when programmers on the Internet can read, redistribute, and modify the source for a piece of software, it evolves. People improve it, people adapt it, people fix bugs. And this can happen at a speed that, if one is used to the slow pace of conventional software development, seems astonishing. Open source doesn't just mean access to the source code. The distribution terms of open–source software must comply with the following criteria: Free Redistribution, Source Code, Derived Works, Integrity of The Author's Source Code.

This model has been used for years, and is responsible for many software projects which holds the Internet together (Apache, Sendmail). The people that develop that software are distributed across the world, almost never seeing each other, only communicating through e–mail and newsgroups. And yet they produce software that is vital to the Internet's functioning.

Some of the most important elements in these projects are given in the following.

- *Official Site*: each project has an official site (almost invariably a web site), where the most up–to–date information exists (there may be mirror sites around the world).
- *Source Code*: the most important aspect of the Open Source Development Model, and is where it gets its name from. "Open Source" means anyone, anywhere, can download the source for the project and look at it, modify it, and resubmit it. A Revision Control System is almost always used in large projects

to manage the source.

- *Official Maintainers*: small projects may just have one maintainer who handles all the modules. Large projects may have one or more maintainers for each module. Design decisions are often made through consensus. These people control what code gets included into the official source tree.
- *Documentation*: every form of documentation related to the project is available, whether it be user documentation, installation guides, or a known issues document.
- *Frequently Asked Questions*: a set of FAQs of varying complexity exist and can be browsed. Some projects just provide FAQs for casual surfers, while others provide detailed FAQs for developers. Often, questions and answers are taken from the project's mailing list and newsgroup. Sophisticated sites can have interactive pages that allow a person to post a question or answer. This cuts out contact with the maintainers and lets anyone in the world ask a question or answer one.
- *Mailing List and/or Newsgroup*: no matter how small the project is, it almost always has a mailing list, even if it is only a group of people who CC each other. The bigger projects can have both a mailing list and a newsgroup, or even multiple mailing lists and newsgroups, ranging from the general to the specific.

In the following table we summarize how each of the presented systems fulfills the list of requirements.

Current. Implement.	***GECOS***	***MILOS***	***DHT***	***Process Web***	***VISCOUNT****	***OSS***
R1 (async.wrk.)		KB workflow managemt.				RCS, Web
R2 (sync.wrk.)		Available technologies				Email, Newsgroup
R3 (communic.)	Internet (Java)	Only LAN (Corba,Java)	WAN (Corba)	Internet (CGI)	Internet (Web)	Internet
R4 (access PK)			Existing tools! Web			
R5 (distr.CM)	Central server	Servers (OODBMS)	Virtual repository		SCM (Lifespan)	Repository
R6 (repos.ontl.)		Web	Web			Web site
R7 (dyn.wrflw)	OO descr.	OO descr.	Web	PWI		

Current. Implement.	***GECOS***	***MILOS***	***DHT***	***Process Web***	***VISCOUNT****	***OSS***
R8 (proact.ntf.)		Y				Mailing list
R9 (user frndy.)	Web?	Internet* (Java*)	Web	Web		N
R10 (enactment)	FSM	Y	Browsing			
R11 (proc.modl.)	Y	Y	Hypertext	PWI		

* under development

4. Conclusions

Our purpose has been primarily to outline the challenges of software development faced by virtual corporations, and to give a survey of some existing systems which provide support for virtual projects. We have done this from a software process modeling point od view. We hope that this survey, by collecting references and descriptions of existing projects, will give some help to others.

5. References:

1. Haag, Zsolt; Foley, Richard; Newman, Julian – Software Process Improvement in Geographically Distributed Software Engineering: An Initial Evaluation. 23rd EUROMICRO Conference '97 – New Frontiers of Information Technology.
2. GECOS – http://mordor.cs.hut.fi/gecos.
3. (1)Alho, Kari; Sulonen, Reijo – Supporting Virtual Software Projects on the Web. IEEE 7th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET–ICE), 1998.
4. MILOS – http://wwwagr.informatik.uni–kl.de/~milos.
5. (2a)Maurer, Frank; Dellen, Barbara – Internet Based Software Process Support. IEEE 7th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET–ICE), 1998.
6. (2b)Maurer, Frank; Dellen, Barbara; Holz, H.; Kötting, B.; Goldmann, S.; Dellen, B. – Software Process Support over the Internet. ICSE'99 ...
7. (2c)Goldmann, Sigrid; Münch, Jürgen; Holz, Harald – MILOS: A Model of Interleaved Planning, Scheduling and Enactment. ICSE'99 Workshop on Software Engineering over the Internet (http://sern.cpsc.ucalgary.ca/~maurer/ICSE99WS/ICSE99WS.html).
8. (2d)Maurer, Frank; Dellen Barbara – An Internet Based Software Process

Management Environment. ICSE 98 Workshop on Software Engineering over the Internet (http://sern.cpsc.ucalgary.ca/~maurer/ICSE98WS/ICSE98WS.html).
9. (2e)Maurer, Frank; Dellen Barbara; Holz, Harald – Process Support for Virtual Software Organizations. 11th International Conference on Software Engineering and Knowledge Engineering (SEKE'99).
10. DHT – http://www.usc.edu/dept/ATRIUM.
11. (3a)Noll, John; Scacchi, Walt – Suppporting Distributed Configuration Management in Virtual Enterprises. Lecture Notes in Computer Science vol. 1235.
12. (3b)Noll, John; Scacchi, Walt – Suppporting Software Development in Virtual Enterprises. Jurnal of Digital Information vol. 1 no. 4 February 1999 (http://jodi.ecs.soton.ac.uk/Articles/v01/i04/Noll).
13. ProcessWeb – http://sweetpea.cs.man.ac.uk.
14. (4a)Greenwood, Mark; Warboys, Brian – ProcessWeb: Process Support for the World Wide Web. Lecture Notes in Computer Science vol. 1149.
15. (4d)Yeomans, Benjamin – Enhancing the World Wide Web. Project Report 1996.
16. Open Source Software – http://www.opensource.org.
17. VISCOUNT – http://www.baesema.co.uk/stg/viscount (http://www.viscount.gcal.ac.uk).
18. Boldyreff, Cornelia; Newman, Julian; Taramaa, Jorma – Managing Process Improvement in Virtual Software Corporations. Proceedings of WET–ICE '96.
19. Warboys, B.C; Balasubramaniam, D.; Greenwood, R.M.; Kirby, G.N.C.; Mayes, K.; Morrison, R.; Munro, D. – Collaboration and Composition: Issues for a Second Generation Process Language. Lecture Notes in Computer Science vol. 1687.
20. Yang, Yun; Wojcieszak, Paul – Visual Programming Support for Coordination of Web–based Process Modelling. SEKE'99 11th International Conference on Software Engineering and Knowledge Engineering, 1999.