A Unified Process Support Framework for Global Software Development

Jin Sa and Elena Maslova
Faculty of Computing, Engineering and Mathematical Sciences
University of the West of England
Bristol BS16 1QY
UK
{jin.sa, elena.maslova}@uwe.ac.uk

Abstract

There is an increasing trend in global software development, where different parts of a software product are being developed in different organisations. The process aspect of such a distributed development environment is particularly important in order to ensure communication and coordination between the teams in different organisations. However different organisations may use different notations in modelling their software development processes. This paper addresses the issue on how to integrate models defined using different notations. The paper presents the initial results from a prototype of a unified process support framework that allows teams to define their process models using different notations. The framework integrates the models into a unified representation to facilitate communication and coordination.

1. Introduction

Software development processes have been studied for many years. There has been a recognition that studying and modelling the process aspects of a software development helps to improve the quality of the product. Much work has been done in this area, for example there are many conferences in the area of software processes [1,10,11]. However most of the work in this field concentrates on traditional software development processes in the sense that the software is developed within one organisation using one development method, and the process model used to support the development process is defined using one notation.

With the rapid growth of the Internet, software development is changing. In recent years, there is an increased globalisation of software development. A software system may be jointly developed by different organisations (or teams) with cultural differences on software engineering practices. We believe that the process aspect is very important in global software development because of the needs for communication and coordination between the development teams. However, it is very likely that the preferred notations of different teams for describing their parts of the process model might be different. This phenomenon raises the challenge of how to facilitate the communication and coordination between different players in the process of the global software development without forcing organisations to change their preferred practices.

Many process support and CSCW systems have been developed to support distributed software development. While most of the work is concerned with supporting shared access and coordination, very little work has addressed the issue of allowing different organisations to use different modelling notations. The aim of our work is to produce a unified process support framework. The framework allows users to create process models. Different parts of a model can be defined in different notations allowing different teams using their preferred notations to define their parts of the model. The framework facilitates the communication and coordination of the different parts of the model by translating them to a unified model.

This paper describes the basic ideas of the unified process support framework. An initial prototype of such a framework, called SPASE, has been developed. Details of SPASE are described in [8]. In this paper we use SPASE to illustrate the concepts of the unified process support framework. The rest of the paper is organised as follows: section 2 explains the basic concepts in the unified process support framework; section 3 describes a prototype of the unified process support framework with an example; section 4 looks at some related work; and finally section 5 draws the conclusion and outlines the future work.

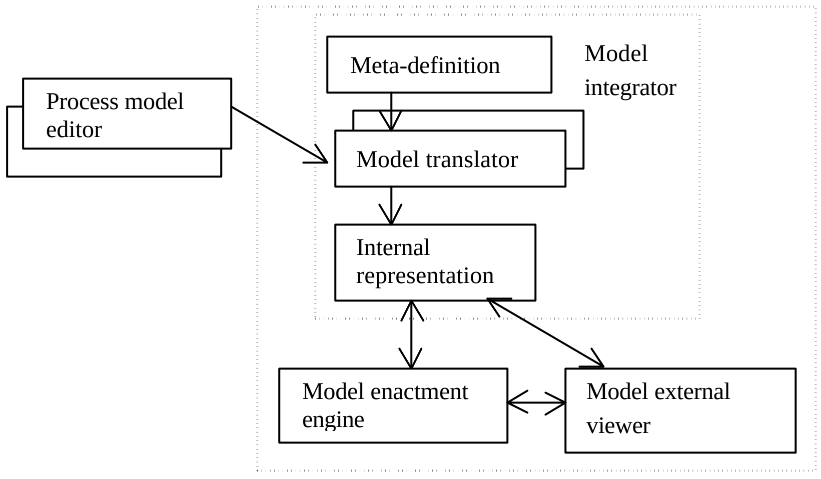

Figure 1. The architecture of the unified process support framework

2. Concepts of the unified process support framework

There are essentially three parts in the unified process support framework: model integrator, model enactment engine, and model external viewer as shown in Figure 1.

The unified process support framework (hereafter referred to as 'the framework') interfaces with a number of process model editors. These process model editors are external to the framework. An example of a process model editor is Rational Rose when it is used to create process models defined in some UML notations. The model integrator itself consists of three parts: a meta-definition, a number of model translators and an internal representation. Model translator takes models defined using external process model editors and translates them into the unified internal representation. The meta-definition defines process concepts such as role, activity and interaction. Each model translator contains a mapping between the concepts in a notation supported by a particular process model editor, e.g. UML class diagram, to concepts in the meta-definition. The model integrator interfaces with the external process model editors such as Rational Rose. For example, if we want to define part of a model using UML class diagram, we can use Rational Rose to create the required class diagram. The model translator for UML class diagram contains a mapping of concepts defined in UML class diagram and concepts defined in the meta-definition. The model translator generates the internal representation based on the mapping. The model enactment engine executes the internal representation and interfaces with the model external viewer. The model external viewer is an interface for displaying the status of the process model.

3. A prototype of the unified process support framework – SPASE

The SPASE project was a feasibility study to demonstrate a possible architecture of the framework.

SPASE only includes a subset of concepts in process modelling. The SPASE architecture maps directly onto the concepts described in the framework in section 2. SPASE is able to interface with two external process model editors: UML class diagram editor (Rational Rose) and a simple Role Activity Diagram (RAD) [9] editor. A full description of SPASE can be found in [8]. In this section, we briefly explain the features of SPASE with a simple scenario.

3.1. A scenario

We have two teams working together: the design team and the review team. The design team has the following activities: 1. produce a design; 2. send it to the review team; 3. receive comments from the review team; 4. repeat from 1 if necessary. The review team has the following activities: 1. receive the design from the design team; 2. review it; 3. send comments to the design team; 4. repeat from 1 if necessary.

3.2. Meta-definition

The SPASE meta-definition is very simple. It only contains the following concepts: process, node (or action), condition for action, links to traverse to other actions, and communication.

3.3. Model definitions and model translators

The SPASE model translator has two plug-in components. One interfaces with Rational Rose, the other interfaces with a simple RAD editor. Consider the scenario described above, the process of the design team is defined using (a subset of) class diagrams; the process of the review team is defined using (a subset of) RADs.

For example, in the model for the design team, the action 'produce a design' is modelled as a specialised node class; 'send it to the review team' is modelled as a specialised communication class; and the link between the two is modelled as a specialised link class. See Figure 2.

The plug-in components use the mappings shown in table 1 and table 2 to translate UML class diagrams and the RADs diagrams to the internal representation.

Meta-definition	UML class diagram
Process	A class diagram
Node	A specialised class
Link	A specialised class
Condition	A specialised class
Communication	A specialised class

Table 1. Mappings between the meta-definition and UML class diagram

Meta-definition	RAD
Process	Role
Node	Activity
Link	State line
Condition	Event
Communication	Interaction

Table 2. Mappings between the meta-definition and RAD

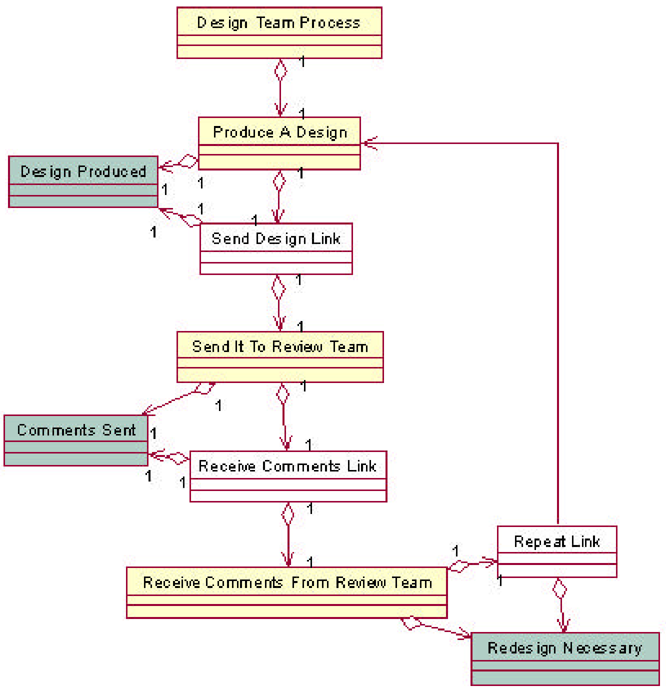

Figure 2. The design team's process modelled using Rational Rose as an external model editor

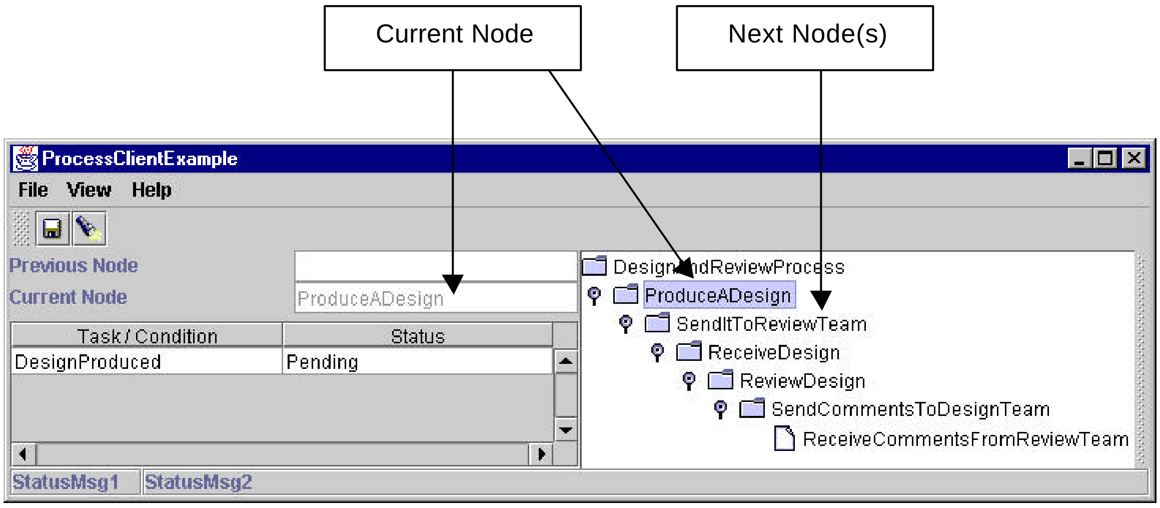

Figure 3. External view of the enactment – snapshot 1: produce a design

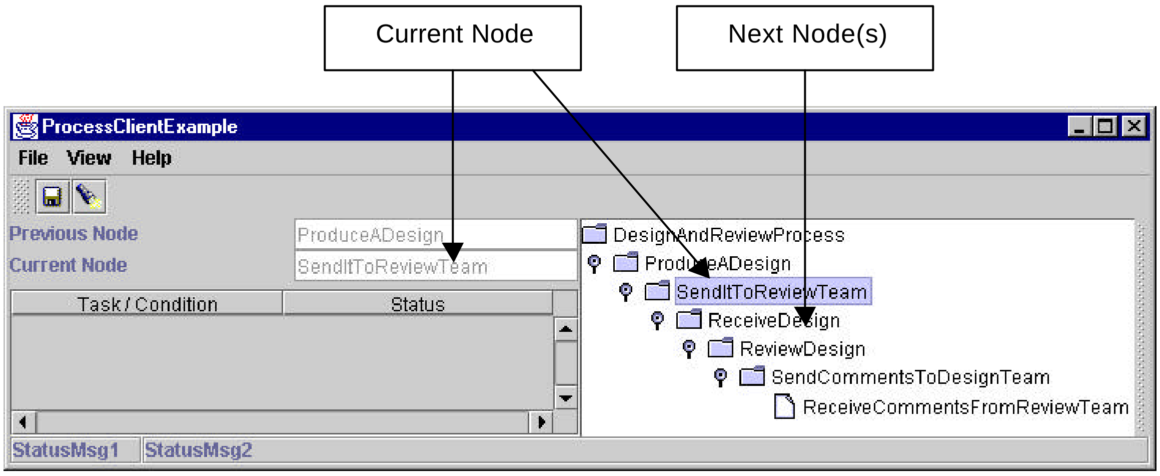

Figure 4. External view of the enactment – snapshot 2: send design to review team

3.4. Internal representation and integration of sub-processes

The output of the model translators is an integrated model defined in the internal representation containing both the design team and the review team. Consider the scenario described above again, following are the steps they would take in order to integrate their sub-processes: Design team's sub-process is modelled in UML using Rational Rose

1 The UML model of the design team sub-process is then loaded into SPASE which translates it into internal representation.

2 Review team's sub-process is modelled in RAD using a RAD editor.

3 The RAD model of the review team is then loaded into SPASE that translates it into internal representation.

We now have two separate internal representations for the design team sub-process and the review team sub-process respectively. The next step is to integrate them by connecting the communications between the two sub-processes. To establish the connection between them, the process participants select one participant to take the responsibility. Assuming in this case, the design team is responsible for making the connections between the two sub-processes, SPASE changes the two sub-processes from its internal representation to the UML representation.

1 Both sub-processes are translated into UML so that the design team can view/edit them to establish the required connections between the design team and the review team.
2 The integrated UML model is then loaded into SPASE that translates it into internal representation.
3 Both teams can follow the integrated model through the external view.

3.5. Enactment engine

The integrated internal representation can be executed by the SPASE enactment engine. Processes are persistent within the SPASE environment. Because of the technology and architecture (includes Enterprise Java Beans) used to develop SPASE, SPASE also supports distributed process enactment.

Consider our scenario again, let the design team be located in Peking and the review team be located in Moscow. Assume that the main SPASE server (which includes the enactment engine and the process model administrator) runs in UK. The teams in Peking and Moscow can interact with the process model administrator and the process during the enactment phase through SPASE external viewers that run locally on their machines. See Figures 3 and 4.

3.6. External viewer

The SPASE external viewer interacts with the internal representation and the enactment engine to display the current status so that the process participants can interact with their parts of the process model. The external viewer user interface contains information about the previous action, the currently enactable actions and their conditions. See Figures 3 and 4. The multi-tiered architecture in SPASE allows the external viewers to be portable. Therefore process participants can have the viewers locally on their machines.

4. Related work

A number of approached have addressed some issues associated with global software development.

Work described in [4,5] address problems with distributed software development coordination. However it does not support multiple process modelling notations.

The processes described in P2E [2] and HDEV [3] allow users to dynamically change to different methods for dealing with different tasks. Although the frameworks in both approaches allow the use of different software development methods within one project, the process model for each project still needs to be defined in one notation.

ProcessWeb [12] allows users to enact process models via the Web. This addresses the distributed issues, but it uses the same notation for all models.

In [6,7] customisation of process modelling languages is tackled. It defines process metamodelling, which allows process modelling languages to be specified and implemented in a process support environment. Therefore it allows users to change to a different process modelling language if necessary. The work however does not address the integration issue, which is the concern of our work here.

5. Conclusion and future work

In this paper, we have illustrated by using SPASE how the concepts in the unified process support framework can be implemented. SPASE only serves the purpose of a feasibility study. The process modelling concepts included in it are very limited. However we have demonstrated that it is possible to provide support to teams with different practices in software development.

Work in the unified process support framework is on going in the following areas. The meta-definition needs to be revisited. It is the notation to which all the external process modelling notations are mapped. Therefore it needs to have a much more comprehensive coverage of process issues. Given that the meta-definition is an internal representation, readability and simplicity are not the main issue. We recognise that it is not possible to include all process modelling concepts in the meta-definition so that any process modelling notation can be mapped and integrated. Our aim is to define a unified framework so that many of the commonly used notations can be integrated.

The design of the model integrator can be compared with CORBA in the sense that it provides a backbone for supporting many different notations. The challenge is to make the part that interacts with the external process modelling editors as small as possible so that it is relatively easy to plug in a component for a specific process modelling editor. We are currently investigating a number of technologies. One of the issues that we are exploring at the moment is to use XML to represent the meta-language. This will de-couple the implementation language of the translators for the external editors from the implementation language of the unified framework. Investigation into the work on process interoperability in the area of workflow management will also be carried out.

The external viewer for SPASE is very simple. It only shows the previous activity and the next possible activities. The design needs to be revisited to include a bigger picture of the process.

The overall conclusion is that we believe it is important to provide process support for global software development. This paper has illustrated that such an objective is achievable.

6. References

[1] EWSPT 2001, The 8th European Workshop on Software Process Technology Witten, Germany, June 2001.

[2] Greenwood M., Robertson I., and Warboys B. C., A Support Framework For Dynamic Organisations, In the proceedings of the 7th Europoean Workshop on Software Process Technology, LNCS 1780. Springer-Verlag, Kaprun Austria, February 2000.

[3] Greenwood M., Warboys B.C. and Sa J., Cooperating Evolving Compoentns – A Rigorous Approach to Evolving Large Software Systems, In the Proceedings of 18th IEEE International Conference of Software Engineering, Berlin, Germany, March 1996.

[4] Grundy J., Hosking J. and Mugridge R., Serendipity: Integrated Environment Support for Process Modelling, Enactment and Work Coordination, Automated Software Engineering, vol. 5, no. 1, 1998.

[5] Grundy J., Hosking J. and Mugridge R., Coordinating Distributed Software Development Projects with Integrated Process Modelling and Enactment Environments, In Proceedings of WET ICE '98: IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Stanford, California, 1998.

[6] Koskinen M. 2000. Process Metamodelling – Conceptual Foundations and Aplication. Jyvaskyla Studies in Computing 7. University of Jyvaskyla. Jyvaskyla Uniersity Press. PhD. Dissertation.

[7] Koskinen M. & Marttin P. 1998. Developing a Customisable Process Modelling Environment: Lessons Learnt and Future Prospects. In V. Gruhn (Ed.) Proceedings on the 6th European Workshop on Software Process Technology, EWSPT'98, LNCS 1487. Springer-Verlag, 13 – 27.

[8] Maslova E. Process Support Environment, MSc Dissertation, University of the West of England, 2001.

[9] Ould M. A. Business Processes – Modelling and Analysis for Re-engineering and Improvement. Wiley. 1995.

[10] ProFes 2001, Improving Software and Software Process, Kaiserlautern, Germany, September 2001.

[11] ProSim/ISPW 2002, International Workshop on Software Process Simulation and Modelling/ International Software Process Workshop, Silver Creek Canyon, July, 2000.

[12] Warboys B. C., Kawalek P., Robertson I., and Greenwood R. M., Business Information Systems: a Process Approach, McGraw-Hill, Information Systems Series, 1999, ISBN 0-07-709464-6.