Interorganizational Management of Development Processes*

Markus Heller, Dirk Jäger

RWTH Aachen University, Department of Computer Science III,
Ahornstrasse 55, 52074 Aachen, GERMANY
{heller|jaeger}@cs.rwth-aachen.de

Abstract. The AHEAD system supports the management of development processes for complex end products in engineering disciplines. AHEAD is based on nearly ten years of ongoing research on development processes in different engineering disciplines and the underlying concepts have been applied to the above application domains.
Today, development processes tend to be distributed across organization boundaries. As the organizations which execute a shared process, usually have different goals and interests, a balance between all interests has to be achieved in a supporting concept.
This paper illustrates by a short sample scenario, how the AHEAD system supports distributed development processes thereby taking the interests of all cooperation partners into account.

1 Introduction

Development processes for complex end products in engineering disciplines are difficult to plan and manage in advance due to their highly creative character. It is very likely, that some product specifications are modified or some planned activities are reorganized during development. Therefore, development processes also have a very dynamic character.

The process management system AHEAD supports the management of development processes in an organization, e.g. a company or a department of a company. The AHEAD system has been developed within a long-term running Collaborative Research Center IMPROVE (Information Technology Support for Collaborative and Distributed Design Processes in Chemical Engineering)[1]. A net-based approach is used to model development activities and their relationships, resources and products, like technical documents or plans. A detailed description of AHEAD can be found in [2].

Today, the development of an end product often is not accomplished by one organization alone. Instead, cooperating organizations execute a common development process together.

A common pattern for cooperation of organizations is a *delegation relationship*, where selected fragments of the overall development process are delegated from one

* The authors gratefully acknowledge financial support of Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center 476 "Informatische Unterstützung übergreifender Entwicklungsprozesse in der Verfahrenstechnik".

contractor organization to another *subcontractor* organization. The subcontractor executes the delegated process and returns a specified product to the contractor. Both parties usually have different interests. Contractor organizations often try to gain full control over the delegated process fragments, while subcontractor organizations are interested in hiding the internal details of the delegated parts of the process.

The AHEAD system provides a management tool to manage distributed development processes. The proposed concept, called *delegation of process fragments*, takes the divergent interests of contractor and subcontractor organizations into account.

This paper describes a demonstration scenario for the support of distributed development processes in the AHEAD system. For sake of brevity, the underlying concepts and the internal realization of the concept are omitted here, they can be found in [3], [4] and [5].

2 Demonstration scenario

In this section, the main features of the AHEAD system for the support of distributed development processes are illustrated. Figure 1 shows a schematic view on a delegation relationship between a contractor and a subcontractor organization. It is assumed that in both organizations AHEAD is used for the process management. In the demonstration session a detailed concrete sample scenario will be used that covers the steps described below.

The upper part of the figure shows a *task net* that is maintained in the management system of a contractor organization. The top-level task T is decomposed into four subtasks T1-T4. Furthermore, the task T3 has been refined by the tasks T3.1-T3.3. The tasks are connected by control flow edges, as indicated.

2.1 Export and import of delegated process fragments

The contractor selects a fragment of the process for delegation, in the example tasks T2 and T3 (shown in the dark grey box). The refining subnet of T3 is added to the delegated process fragment automatically.

Then the selected fragment is exported to a file. Copies of the delegated tasks remain in the contractor's management system. The file also includes contextual information about the delegated part of the process.

Next, the subcontractor imports the file and a corresponding task net is set up in his management system. In the lower part of Figure 1 the imported task net is shown in the grey box (the surrounding context information of the delegated process fragment is omitted).

After inspection of the imported task net, the subcontractor accepts the order of the contractor to execute the transferred process and to deliver the specified results to the contractor. In this way, the delegation of process fragments is interpreted as a contract between contractor and subcontractor. In contrast to the delegation of a single task, entire task nets can be delegated. Thereby, the subcontractor has to agree to the specified structure of the delegated process (milestone activities). Violations of this treaty may happen, but need not have legal consequences.

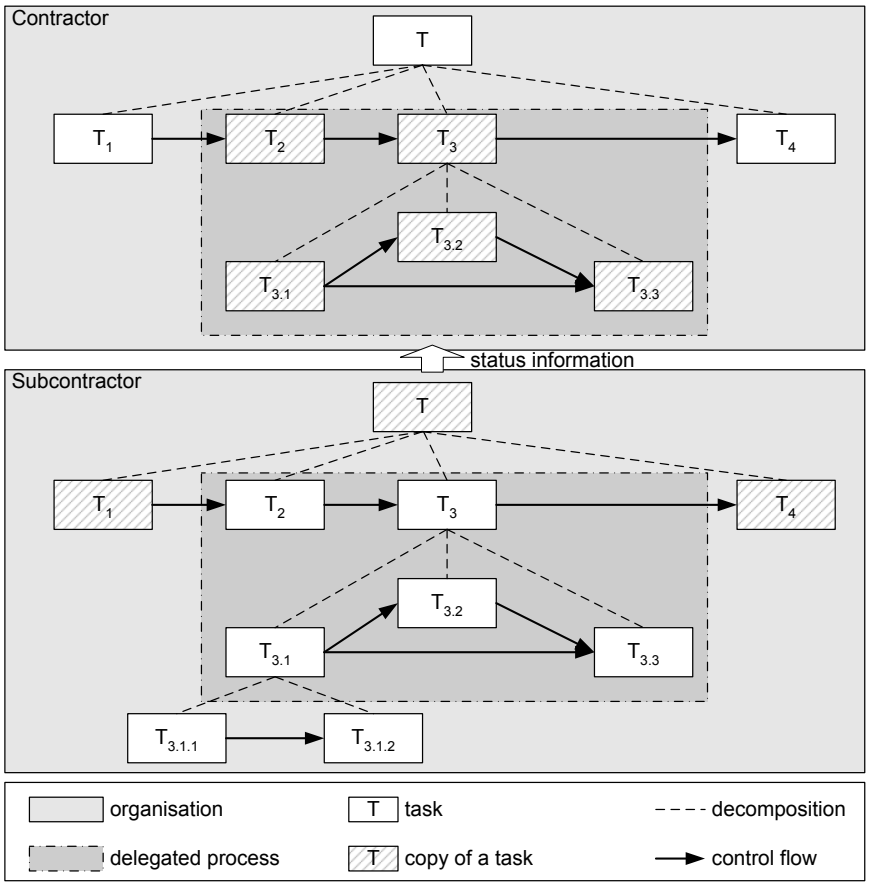

Fig. 1. Schematic view of a delegation relationship between two coupled systems (after [4])

2.2 Run-time coupling

Both management systems are coupled at run-time and exchange *messages* to inform each other about *changes*. All delegated tasks can only be manipulated in the subcontractor system. Changes of these tasks are signaled to the contractor system where these changes can be monitored, too. Vice versa, only the contractor can manipulate tasks in the context of the delegated process fragment. Changes of these tasks are signaled to the subcontractor system.

Both management systems can work independently, if a communication server handles the message transfer. The server is in between the two management systems, receives messages from one management system and sends them to the other. If one of the management systems becomes temporarily unavailable, the messages are stored in the server until this management system has registered with the server again.

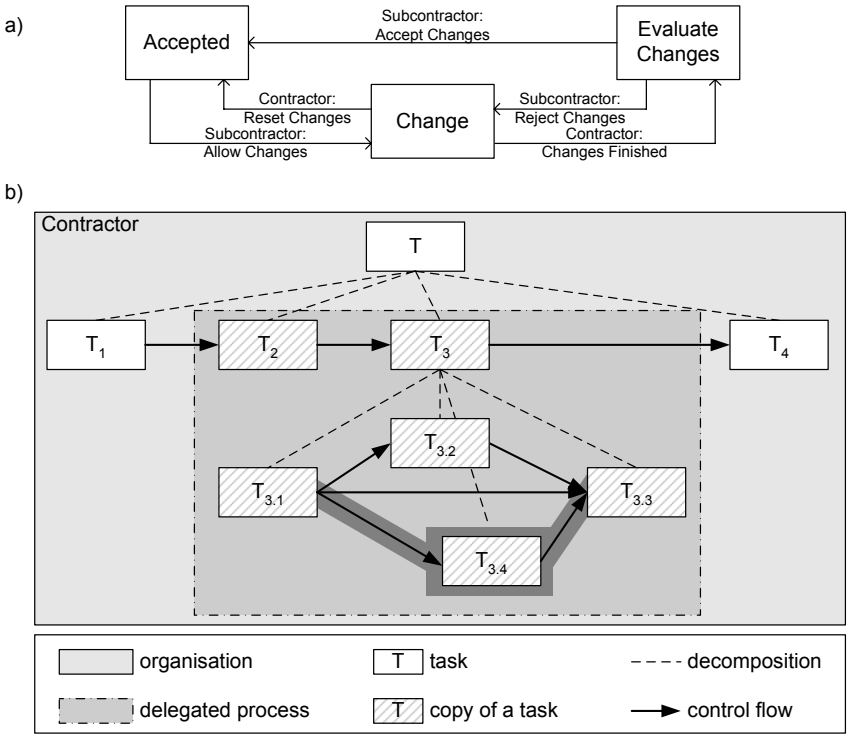

Fig. 2. Structural changes of the delegated process fragment

2.3 Information Hiding

The subcontractor can refine the delegated tasks by private task nets. In the example, the delegated task T.3.1 is refined with the private tasks T3.1.1 and T3.1.2.

These private refinements are not visible to the contractor. The subcontractor can change the visibility property of private tasks, so that the contractor can see them. In this case, copies of these tasks are sent to the management system of the contractor. Every change of these tasks in the subcontractor system is propagated by change messages to the copies of these tasks maintained in the contractor system.

Analogously, the contractor can refine tasks, for which he is responsible for, by private sub-nets, which are not visible to the subcontractor.

2.4 Structural changes of the delegated process

After the run-time coupling has been established, the delegated process fragment can be structurally changed in order to change the contract between contractor and subcontractor. For example, addititional tasks can be added to the delegated task net or new control flow relationships between delegated tasks can be inserted.

Both organizations have to communicate with each other about an intended change of the delegated task net, and a consensus has to be reached. This phase of negotiation

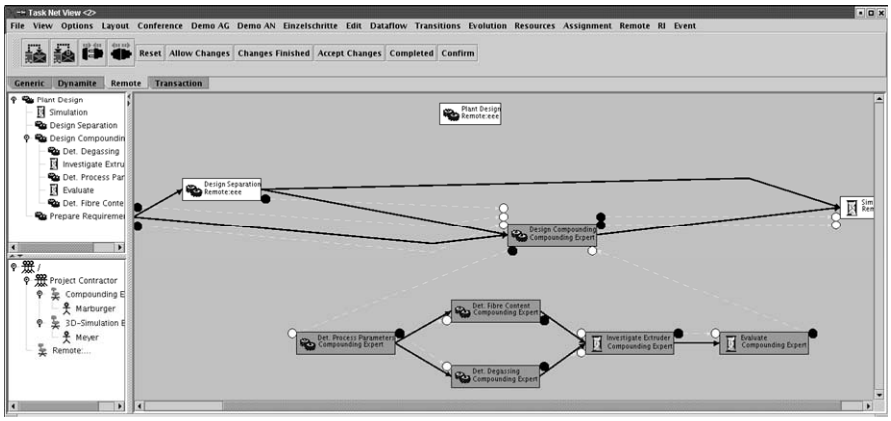

Fig. 3. Screenshot of AHEAD system on subcontractor side

is not supported in the AHEAD system. A formal change protocol is enforced, where the contractor carries out all changes, while the subcontractor can accept or refuse these changes (Figure 2).

The subcontractor starts the change procedure to allow changes of the delegated process fragment by executing the command `Allow Changes` within his instance of the AHEAD system . The delegated process fragment changes its state from *Accepted* to *Change* (according to the state transition diagram in Figure 2.a).

In the change phase only the contractor can change the delegated process fragment structurally. All changes are indicated in both AHEAD systems at the same time, while the contractor changes the task net. These changed parts of the task net are displayed by colored markings to indicate that they are just intented changes, since these changes are proposed by the contractor and are not yet accepted by the subcontractor. Figure 2.b shows an example, where a new task T3.4 is added to the delegated process fragment as a sub task of T3, together with two new control flow relationships between T3.1, T3.3, and T3.4. After all intended changes have been done, the contractor executes the command `Changes Finished`. The cancellation of all changes is possible by executing the command `Reset Changes`.

In an evaluation phase the subcontractor inspects the changed process fragment and decides, if he accepts these changes of the contract or not. To simplify the change protocol, all intended changes can only be accepted or rejected as a whole. In the example, the subcontractor accepts the changes and executes the command `Accept Changes`. The delegated process fragment changes its state from *Change* to *Accepted*. All intended changes are then executed in both AHEAD systems and the delegated task net is updated. Now the changed process fragment is visible in both systems.

A screenshot of the AHEAD system on the subcontractor side is shown in Figure 3. In the graphical view on the right-hand side the task net is presented to the subcontractor. All tasks of the delegated process fragment presented in the demonstration (task `Design Compounding` and its refining tasks) are coloured dark grey. In the different views on the left-hand side all available tasks and resources are presented.

2.5 Finishing the delegation

The delegated process fragment has been delegated to the contractor in order to receive a specific product as the result of the delegated process activities. After the subcontractor has produced the desired result, it is sent back to the contractor organization. The contractor evaluates the product and accepts the result by executing a special command. In this case, the delegation relationship is finished and the necessary coupling of the two management systems is terminated. Otherwise the contractor rejects the produced result by executing a special command.

3 Summary

The AHEAD system supports the management of development processes that are distributed across multiple organizations[4]. The cooperating organizations work together in a delegate relationship and are able to execute a shared process. In his instance of AHEAD, the contractor exports delegated parts of a development process into a file and a subcontractor imports this file into his management system. Next, the management systems of the cooperating partners are coupled at run-time and exchange messages to inform each other about changes relevant for the delegation relationship. The use of a communication server in between both management systems ensures that they can operate independently. Contractor and subcontractor can refine tasks with private task nets, which are not visible to the partner. If wanted, these private tasks can be made visible for the other organization.

Structural changes of the delegated process after enactment are possible. A fixed change protocol enforces that only the contractor can change the delegated process in its structure, and the subcontractor can only reject these changes.

The proposed delegation concept for supporting distributed development processes has been applied in software and chemical engineering within the IMPROVE project[1]. In this demonstration session, the coupling of two instances of the AHEAD system will be demonstrated.

References

1. Nagl, M., Westfechtel, B., eds.: Integration von Entwicklungssystemen in Ingenieuranwendungen. Springer, Heidelberg (1998)
2. Westfechtel, B.: Models and Tools for Managing Development Processes. LNCS 1646. Springer, Heidelberg (1999)
3. Becker, S., Jäger, D., Schleicher, A., Westfechtel, B.: A delegation-based model for distributed software process management. In Ambriola, V., ed.: Proceedings 8th European Workshop on Software Process Technology (EWSPT 2001). LNCS 2077, Witten, Springer (2001) 130–144
4. Jäger, D.: Unterstützung übergreifender Kooperation in komplexen Entwicklungsprozessen. Volume 34 of Aachener Beiträge zur Informatik. Wissenschaftsverlag Mainz, Aachen (2003)
5. Heller, M., Jäger, D.: Graph-based tools for distributed cooperation in dynamic development processes. In Nagl, M., Pfaltz, J., eds.: Proc. AGTIVE 2003. LNCS, Springer (2004) In this volume.