A Delegation Based Model for Distributed Software Process Management

Simon Becker, Dirk Jäger, Ansgar Schleicher, Bernhard Westfechtel

Aachen University of Technology
Department of Computer Science III
D-52056 Aachen, Germany
{sbecker|jaeger|schleich|bernhard}@i3.informatik.rwth-aachen.de

Abstract. Complex development processes which cross organizational boundaries require specialized support by process management systems. Such processes are planned in a top-down manner. A suitable cooperation model for these processes is the delegation of process parts. Because the client and the contractor of a delegation may be independent organizations they may have diverging interest concerning autonomy of process execution, information-hiding, control, etc. We propose a concept for delegating process parts which takes these interests into account and describe how delegation is implemented in the process management system AHEAD.
Keywords: process modeling, process management, interorganization cooperation, delegation, distributed processes

1 Introduction

Developing complex technical products has become a challenging task which can only be handled by the cooperation of many individuals. A process management system (PMS) supports the manager to control such a development process which involves a large number of developers. The skills required for developing a product can be quite diverse and thus the employees involved in the development processes can belong to different organizational units. Those can be different branches of a company or even different companies which may be geographically dispersed. Each organization conducts its part of the process, which is guided by the organizations' local PMS. These local PMS must be integrated to establish the interorganizational processes.

Integration of processes can follow different models [19, 1]. Which of these is appropriate depends on the application domain. For companies that e.g. aim at integrating processes along a supply chain, chained execution is a good model. In contrast to this, we regard a top-down approach more suitable for modeling development processes in which one organization acts as a supervisor and delegates parts of the process to other organizations.

Again, several models can be applied here. Taking a black-box approach, the process at the delegating organization simply triggers the execution of a process at the contractor organization without having knowledge about the internals of this process or receiving feedback on its progress. This approach is hardly applicable because it has been recognized, that the client-contractor relationship is more complex in reality. The delegation of a task should also include guidelines how the task has to be performed, milestones to monitor the progress of the project, etc. An adequate model for the delegation of processes must take into account additional requirements like security, information-hiding, and autonomy of process execution [18, 14].

In this paper we present an approach that allows for the delegation of parts of development processes which are modeled as process nets in a PMS. The parts can be selected on the fly which gives the manager a great flexibility in deciding which tasks he wants to delegate. The manager can monitor the execution of the delegated parts without seeing the details of their realization. In this way, the approach satisfies the needs of the client as well as of the contractor of a delegation. Moreover, the system automatically ensures the integration of the process parts and the consistency of the overall model which results in a tighter coupling of the participating PMSs than it is usually realized in federated process centered software engineering environments (PSEE) [1].

The rest of this paper is structured as follows: In Section 2 we will introduce the scenario of delegation in development processes and discuss the requirements which have to be fulfilled by a PMS to support delegation. Section 3 briefly introduces the process modeling language of dynamic task nets. In Section 4 we describe how delegation is supported by the AHEAD management system which uses dynamic task nets as process modeling language. Section 5 discusses related work and Section 6 concludes the paper.

2 A Concept of Delegation

Interorganizational processes emerge whenever organizations agree to cooperate. The subject of the cooperation and the relationship between the cooperating organizations determine the nature of the interorganizational process and how it should be supported by a PMS. Therefore, we will start with a short discussion of possible cooperation scenarios and point out in which of these delegation is a suitable cooperation model.

Our approach does not focus on routine *business processes*. Organizations which want to cooperate by integrating their business processes, e.g. along a supply chain, usually do not delegate process parts. The processes to be integrated already exist and have to be chained. Most of the work on interorganizational workflows deals with the chaining of existing workflows.

In contrast to this, we are interested in supporting *development processes* which are highly dynamic and can not be planned in advance in every detail but only on a coarse-grained level. Which activities have to be performed in detail depends on the results of previous activities. For example, which coding activities are needed in a software project depends on the design document which is the result of a design activity.

This kind of planning can only be done in a top-down manner. Therefore, one of the cooperating organizations is responsible for it and acts as a supervisor of the interorganizational process. Such a situation naturally arises when an organization is the initiator of a development process and acts as the *client* of other organizations which are hired as *contractors*. The client organization delegates parts of the process to its contractors, which enact these process parts. We will use the terms *client* and *contractor* in the following to refer to the roles involved in a delegation. An organization may participate in more than one delegation in different roles. A contractor can delegate parts of its process to subcontractors. In this case the contractor acts as a client of his subcontractors.

In the most general case, the client and the contractor are independent organizations which are only connected by the delegation. Both may have different interests, e.g. the client wants the contractor to produce a product as good as possible while the contractor tries to produce a product which meets but not exceeds the agreed specification. Likewise, the client is interested in the overall result of the process while the contractor does not care about it, etc. Our goal was to create a model of delegation which takes into account the interests of client and contractor. This proved to be hard because these interests are in some cases conflicting. As a result, we have defined the following set of requirements along which we constructed our system:

- *Delegation of processes fragments.* The system should be able to delegate process fragments consisting of several tasks including their relationships. This means that the client should be able to specify the steps of the processes on a coarse-grained level. For example, software development projects for German government organizations have to use the V-Model. A client can enforce this by specifying the phases of this model as top-level activities of a delegated process.
- *Delegation as a contract.* The delegation of a part of a process should be viewed as a contract. Once the client and the contractor have agreed to it, none of them is entitled to change the delegated process independently.
- *Control of process execution.* The client should be able to monitor the execution of the delegated tasks in his own PMS. The delegated tasks are milestones of the project and their states provide the client with feedback concerning the progress of the delegated part of the process.
- *Refinement by contractor.* The contractor can refine each of the delegated tasks as long as the delegated coarse-grained process remains unchanged. This requirement reflects that each organization has its specific standards and processes to realize certain tasks.
- *Autonomous execution.* The contractor should autonomously execute the delegated process fragment. This requirement is to be understood in two ways. Firstly, on the conceptual level it means that the contractor has full control over the execution of the delegated part of the process. The client PMS merely provides a read only view on it but cannot be used to influence it (e.g. by performing operations on the delegated tasks). Secondly, with regard to the coupling of the management tools involved, it means that the

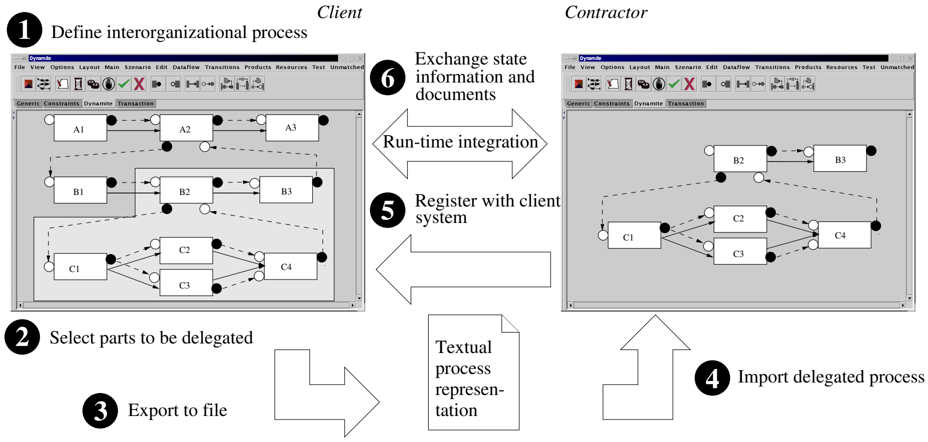

Fig. 1. Steps of delegation

contractor PMS should be able to execute its part of the process without contacting the client PMS.

– *Information hiding*. While the client wishes to monitor the delegated process part, the contractor usually does not want him to see all the details of process refinement. Therefore, the contractor PMS should be able to hide levels of process refinement and, in the extreme case, provide the client PMS only with information concerning the coarse-grained tasks of the original delegation process.
– *Preservation of execution semantics*. An important requirement of delegation is that in the contractor PMS the delegated process should be executed exactly as it would be in the client PMS. This requirement is naturally fulfilled if both systems use the same PML. However, there are many different process modeling formalisms around, and in general, systems may cooperate which use different PMLs. In this case, a translation step is required. However, the translation usually is not trivial. Though each net-based PML has modeling elements for activities, documents, control flow, etc. their semantics may be different. A naive and direct translation between two PMLs might result in a translated net having the same topology as the original one but exhibiting a completely different behavior. Thus, our approach as far as it is presented in this paper is restricted to the homogeneous case in which client and contractor use the same PML.

Figure 1 shows the steps in which a delegation is performed. At first, the process manager of the client specifies the interorganizational process including the parts to be delegated to contractors in the following. In Step 2, parts which are suitable for delegation are selected. Here, suitable is to be understood with regard to the real-world process. In addition, the selected parts have to satisfy some conditions with regard to the semantics of the employed modeling formalism, e.g connectedness, which will be discussed in Section 4.

The selected part of the process is exported to a file in Step 3. The exported elements of the process net are not removed form the client's PMS but are tagged as being exported. These parts of the net cannot be changed anymore by the client, except for that he can revoke the export at all. We think that an asynchronous way of communicating the delegated process via a file should be preferred, because the contractor can be unknown at this time. For example, the exported process description can be published at electronic market places along with a call for tender. As soon as a contractor is found, the process description becomes part of the contract and serves as a formal definition of the work to be carried out.

In Step 4 the contractor imports the process definition. The process model created there is a copy of the net in the client system. Next, the contractor registers with the client (Step 5). For this purpose the exported net includes the address of the client. A direct communication is established and the interorganizational process is ready for enactment.

At runtime, the contractor is responsible for executing the exported activities. The state of the execution is reported to the client which updates his copy of the exported net. Likewise, input and output documents of the delegated activities have to be propagated over the network as well.

3 Dynamic Task Nets

The concept of process delegation which is presented in this paper was developed for dynamic task nets. Dynamic task nets are a visual language for modeling development processes [9]. The concept is implemented within the PMS AHEAD (Adaptable Human-centered Environment for the Administration of Development processes) which uses dynamic task nets as process modeling formalism [12, 11]. In AHEAD there are basically two user roles: the *process manager* who is responsible for planning and the *developers* who are guided by the PMS. A process manager uses the *management tool* of AHEAD which displays a view on the complete process net. All screenshots throughout this paper show the main window of the management tool. The developers interact with the system via the *developer front ends*. The front ends provide each developer with an agenda which contains those tasks currently assigned to him.

The language of dynamic task nets is formally specified as a graph-based meta model in the executable specification language PROGRES [13, 16]. The specification includes the types of modeling elements and their possible relationships as well as complex commands for constructing, analyzing and enacting a task net.

Figure 2 shows a sample task net instance. Tasks are shown as boxes. They have *input parameters* (empty circles) and *output parameters* (filled circles). A task's parameters are placeholders for the documents the task is working on. Tasks are connected by control flow relationships, denoted by arrows, which describe the order of execution. Data flow relationships, denoted by dashed arrows, can be viewed as a refinement of control flow relationships. Data flows specify

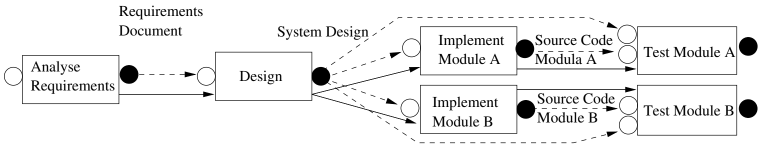

Fig. 2. Sample task net

which output document of one task is consumed as an input document by another task. The actual flow of documents is model by passing tokens which are produced by output parameters and consumed by input parameters.

Another important feature of dynamic task nets is the distinction between *complex tasks* and *atomic tasks*. Complex tasks can be refined by a net of subtasks. A complex tasks can have two kinds of parameters: *external parameters* are used for passing documents between tasks on the same hierarchy level while *internal parameters* serve to pass documents between a task and its refining subtasks.

To make process models executable, not only the structure but also the behavior of tasks has to be defined. This is achieved by assigning a state to each task, e.g. `Waiting`, `Active` or `Done`. State transitions are either triggered by certain events within the task net or are requested by the user, e.g. a developer changes a task's state from `Waiting` to `Active` when he starts working at this task, later he may have successfully finished the tasks and changes the state to `Done`. Whether the developer may perform these state changes depends, among others, on how the task is embedded in the net. For example, if there is an incoming control flow edge specifying that the task and its predecessor have to be executed sequentially, the task can only be started if the predecessor is in state `Done`.

Because development processes cannot be fully planned in advance, dynamic task nets allow for the intertwined execution and modification of process models. Operations changing the net structure can be applied to a task net in execution. The formal definition of all these operations ensures that the consistency of the net is always maintained.

In general, the executability of a certain operation at one task depends on the state of its predecessor, successor, parent and child tasks, and their parameters. We refer to these model elements as the *context* of the task.

4 Delegation in Dynamic Task Nets

So far, we have presented the concept of delegation on a rather abstract level and we have pointed out in which situations it can be applied. In this section we will present the implementation of delegation in the process management system AHEAD. For this purpose we will refer to Figure 1 and describe each step in detail from a technical point of view with help of a small example process.

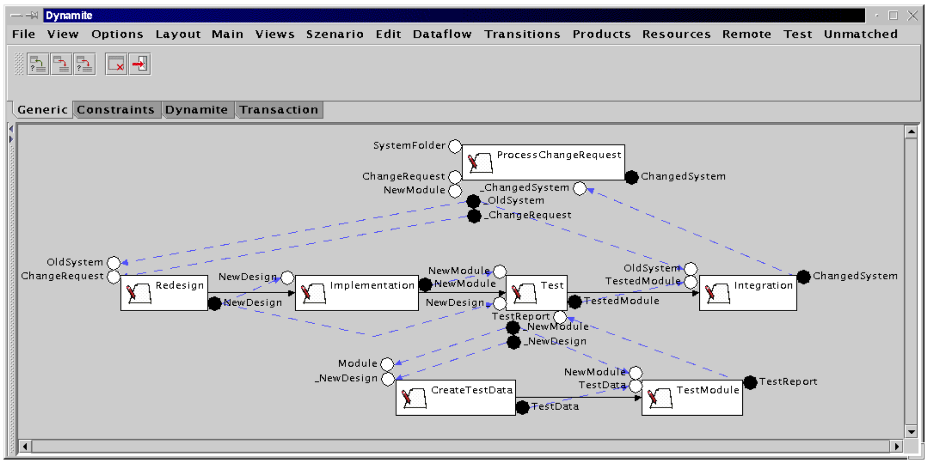

Fig. 3. Definition of the interorganizational process

4.1 Definition of the Interorganizational Process

The definition of an interorganizational process (Step 1 of Figure 1) does not differ from the definition of a local process at this stage. Figure 3 shows a screen shot of AHEAD's manager view. The process manager has constructed a model of a process for processing a change request. We can see three levels of refinement. The task `ProcessChangeRequest` at the top represents the process as a whole. It is refined into the sequence of `Redesign`, `Implementation`, `Test` and `Integration`. The task `Test` is again refined into `CreateTestData` and `PerformTest`. At this point, the manager decides that it may be a good idea to hire a contractor for doing the implementation and the subsequent test, while the redesign of the system remains with his own organization.

4.2 Selecting the Tasks to be Delegated

To start a delegation, the manager has to select the tasks to be delegated and invokes a command on these which prepares them for export (Step 2 of Figure 1). In terms of our graph-based process model, preparation for export means that a special node of type `ExportLink` is introduced into the net which connects the delegated tasks. The command also enforces a number of conditions which ensure the consistency of the model:

- The delegated top-level tasks have to be connected by control flows. Unconnected nets should not really cause problems from a technical point of view but we think they should be treated a two seperate delegations.
- Each element of the net may only be exported once, because a task can only be delegated to one contractor.
- The refining subnet of each task has to be exported as well if it exists. Defining a subnet means to specify how the contractor should perform the delegated task. Therefore, the subnet must be part of the delegation.

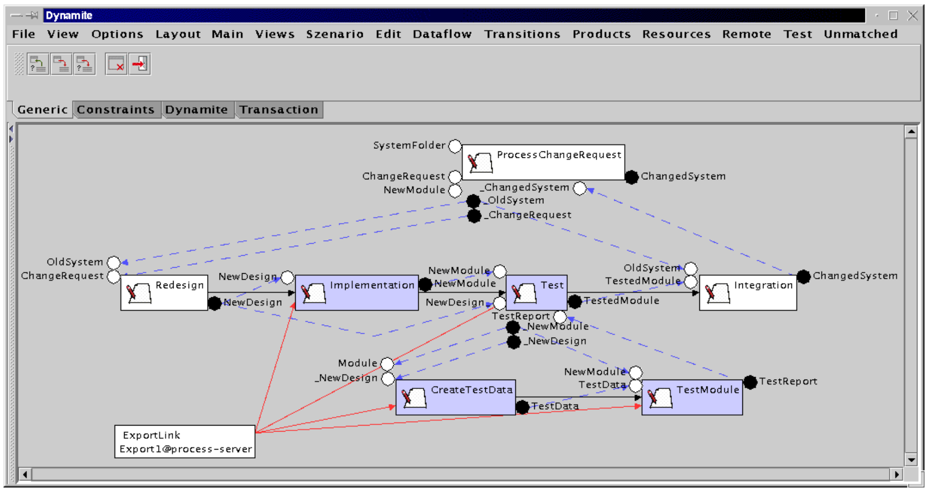

Fig. 4. Selecting the tasks to be delegated

The status of the net after preparation for export is shown in Figure 4. Normally, the node of type `ExportLink` in the window's lower left is an internal node of the model and not shown to the tool's user. Rather, the now prepared and later exported tasks are visualized by a different, here darker color. The export link also serves as the data structure in which all information concerning the delegation is stored, including the URL which identifies the client system and the delegation. Through this URL the contractor can contact the client after having imported the delegated net and establish a connecting for coupling the parts of the interorganizational process.

4.3 Exporting the Delegated Net

After having selected the delegated tasks, the manager invokes the command for the actual export (Step 3 of Figure 1). This command at first retrieves the parameters of the exported tasks. Then, by adding an export edge to each parameter, they are marked for export as well (Figure 5). Though parameters are modeled as nodes of the net, they are subordinated model elements which are only meaningful together with their tasks.

Next, the command retrieves the context of the delegated net. The context is the set of those nodes which are connected by edges to the nodes marked for export. These nodes are added to the exported net via edges of type `context` (dashed edges originating from the node `ExportLink` in Figure 5). By including the context of the exported net, the enactment of the process in the contractor's PMS is greatly simplified as we will point out in Section 4.5.

After these preparing steps, a textual representation of the exported net is written to a file. The task net is represented as an attributed graph according to GXL [10], a graph-based exchange format based on XML. The textual rep-

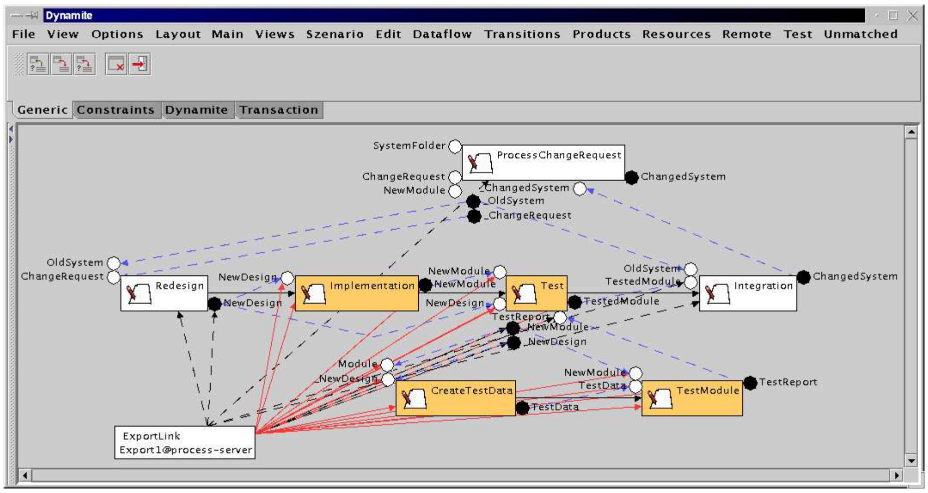

Fig. 5. Exporting the selected tasks

resentation also includes the export link of this delegation through which the contractor can identify the client system.

As soon as the export is finished the PMS rejects all changes to the exported net by the client process manager. The execution of the other parts of the net may continue normally. The PMS now waits for being contacted by a contractor which has imported the task and claims to be responsible for them. However, the client process manager can call a command which revokes the export, removes the export link and thus invalidates the file containing the process description. After the connection between client and contractor has been established, net changes are possible if both agree to them (see Section 4.6).

4.4 Importing the Delegated Net at the Contractor

To import the delegated net (Step 4 of Figure 1), the contractor reads the textual representation and constructs a task net accordingly (see Figure 6). This net also contains the tasks and parameters of the context. Just as the delegated net is in a read-only state in the client PMS, the context of the delegation is read-only in the contractor PMS, which is shown to the user by a different color of the tasks. Internally, there is an import link node (not shown here) which corresponds to the export link node in the client PMS. The contractor system now contacts and registers with the client (Step 5 of Figure 1). In the following, each system reports changes of the execution state of model elements on which the remote system contains a read-only view.

Note that the context of the net in Figure 6 is really limited to those nodes which are directly connected to the delegated net by an edge[1]. The predecessor task `Redesign` and its output parameter `NewDesign` are part of the context

[1] Not all edges of the internal model are shown to the user. The edges connecting tasks and their subtasks are e.g. visualized by the arrangement of tasks in different layers.

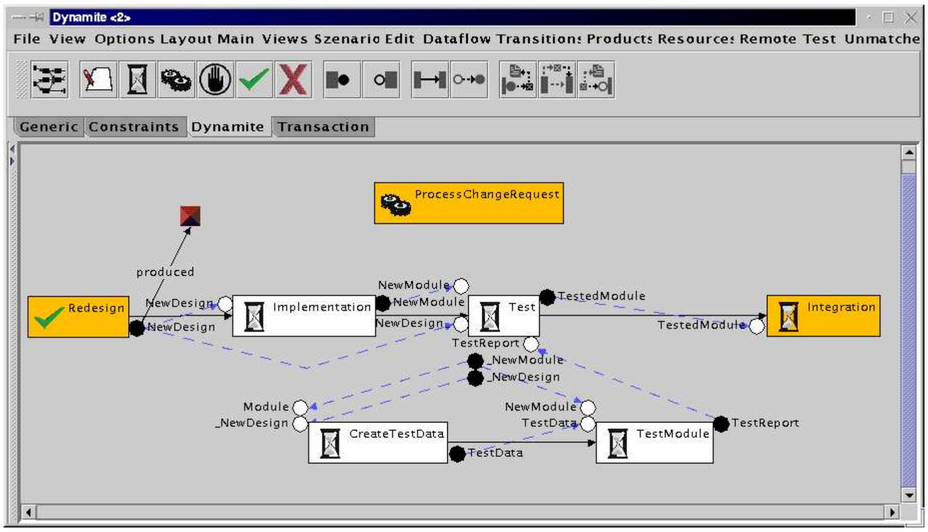

Fig. 6. Contractor PMS executes the delegated process

but its input parameters `OldDesign` and `ChangeRequest` are not. From the contractors point of view, the context is the interface to the client process. The executability of the task `Implementation` in the contractor system depends on the execution state of its predecessor task and the availability of the input document passed through the parameter `NewDesign`. The contractor does not have to know what inputs are required for `Redesign`.

However, the parameters of the delegated task net are not suited as the only specification of the deliverables of a delegation. There have to be accompanying specification documents, which are outside the scope of the process model just as the transfer of the process description itself can naturally not be part of the process description.

4.5 Executing the Interorganizational Process

The presence of the delegated net's context at the contractor greatly simplifies the realization of the integration mechanism between the systems (Step 6 of Figure 1). From the enacting process engine's point of view, read only tasks are handled like ordinary tasks. The difference is that no local user of the system, developer or process manager, has the permission to change these tasks. Instead, a component of the PMS, the *link manager*, acts as a user of these tasks by invoking the same operations on them as normal users invoke on local tasks.

Unlike a human user, the link manager does not call operations on the process model to reflect the progress of the real-world process. It merely reacts on messages sent by a peer system concerning changes in its part of the model. Likewise, the link manager monitors the local process model. If it detects a change

Therefore, the parent task of a delegated net, in this case `ProcessChangeRequest`, is always part of the context, though the user does not see an edge.

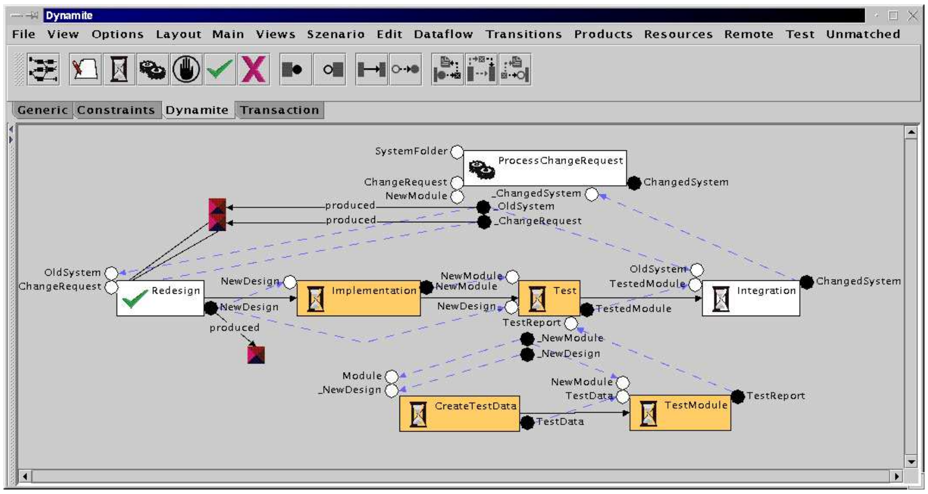

Fig. 7. Client view on the delegated process

at a tasks that is part of the context of a delegation, it sends a message to the contractor PMS.

Figure 7 shows the task net in execution at the client. By looking at the icon of the task `ProcessChangeRequest` we can tell that the task's state is `Active`, i.e. the task is executed right now. Through its internal output parameters `OldSystem` and `ChangeRequest` the task has passed the respective documents to its subtask `Redesign`. The task `Redesign` has read the documents, and it has produced a new design, denoted by the token which is attached to the output parameter `NewDesign`. After this, the task was finished and the developer to whom the task was assigned has switched its state to `Done`.

Now let us have a look at the contractor PMS as it is shown in Figure 6. The contractor sees the state of the parent task `ProcessChangeRequest` and of the predecessor task `Redesign` which are part of its context. He does not see the tokens passed between them. The token produced by the output parameter of `Redesign` is also present in the contractor PMS. It was created by the link manager after the client system informed it about this change in the context of the delegation. Through the connection between the link managers, the actual design document, let us e.g. assume some UML model, was transferred from the client's document repository to the contractor's repository.

The developer to whom the task `Implementation` is assigned can now see in his front end that the design document is available. He can invoke the command for consuming the token which causes the actual document to be transferred to his local workspace where he can work on it.

4.6 Changing an Exported Net

Throughout this paper, we have emphasized that the delegation of a process net is a contract. Neither the client nor the contractor is therefore entitled to change the delegated net. The parts of the net to be delegated should thus be chosen very carefully. However, due to the unpredictability of development processes changes to delegated nets might become necessary. The PML of dynamic task nets allows for changes at process enactment time. They are performed by the process manager and they are not unusual.

But in case of a delegation, there are two process managers and, because client and contractor are independent organizations, both have to agree to change the delegated part of the net. How this agreement is reached is beyond the scope of our management system. As soon as an agreement is reached, the manager at the contractor can switch the state of the delegated tasks to `Planning`. In that state, the client PMS permits the manager at the client to modify the delegated net. Each change is propagated by the link manager to the contractor where it is executed first, because the complete information, especially concerning the refinement of delegated tasks, is only available at the contractor. Therefore, attempts to change the net may as well fail and in this case the contractor's manager has to do some preparing modifications at first. Only if the change succeeds in the contractor system, it is also performed in the client system. Altogether, one can say that changing a delegation during enactment is difficult not only with regard to the process model, but also in real-life. It certainly requires frequent interaction of the involved people.

5 Related Work

Distribution and distributed enactment of processes and workflows have been studied to some extent and implemented in several systems, e.g. Oz [3], PROSYT [6] or FUNSOFT Nets [7]. Most of these approaches focus on connecting already existing processes. With regard to interorganizational cooperations this means that each organization's part of a process is modeled separately including the connection points to the other parts. These systems neglect the requirements of interorganizational cooperations, which are formulated in [18, 14, 1, 20], and have their strength in supporting a distributed team of developers who are members of the same organization. The loose coupling is typical of federated PSEEs where the overall process which is established through the federation remains implicit. In contrast to this, Tiako proposes in [17] to explicitly model the federation process instead and introduces the notion of delegation.

In [8] Dowson reports on the IStar environment in which single tasks can be delegated to contractors. For each delegation, a contract specification describes the contractor's obligations. It is in the contractors responsibility to decide how these obligations are fulfilled. The approach does not allow for the delegation of a whole subnet. Therefore, guidelines how to perform a delegated task can only be given as an annotation to the task and do not integrate with the process model as they do in our system.

Research by van der Aalst [20, 19] provides valuable insight into the interoperability of workflow management systems. He identifies several forms of interaction among which we can find *subcontracting*. Again, subcontracting is limited to single activities of the workflow. There are other WfMS which offer distribution of workflows, e.g. ADEPT [2] or Mentor [15]. In these systems, distribution is used to cope with a huge number of workflows as it might occur in a large enterprise. Critical issues are the migration of workflow execution between different workflow servers and the reduction of communication overhead. Organizational boundaries and the requirements resulting from them are not considered.

Delegation requires the transfer of process descriptions and their subsequent enactment by the target PMS. In the area of workflow management the Workflow Management Coalition has specified the Workflow Process Definition Language (WPDL) [5] and an API for connecting workflow servers at runtime [4]. The interfaces defined by the WfMC do not support delegation as introduced in this paper but merely provide low-level support for system integration. We did not use the WPDL for process exchange in our system because it is confined to the workflow domain, and mapping dynamic task nets to WPDL would result in a loss of semantics.

6 Conclusion

In this paper, we have presented a concept by which a client in an interorganizational development process can delegate parts of net-based process models to contractors. The approach takes into account the special circumstances which arise when different organizations agree to cooperate in a development project. The achieved integration of the process parts is tighter than it is in comparable approaches for distributed process enactment. Nevertheless, client and contractor may autonomously execute their process parts as long as the delegated net, which is part of the contract between them, remains unchanged.

We have implemented the concept of delegation within the process management system AHEAD for the modeling language of dynamic task nets. To perform an operation at one model element, the context of this element has to be known. The realized export and integration mechanism therefore includes the context of the delegated net. This greatly simplifies the implementation of the coupling because the underlying process engine can treat delegated or imported tasks just like ordinary local tasks. To update the read-only copies of tasks the process link managers have been introduced which monitor the process execution and inform their remote counterparts if changes at relevant parts of the process occur.

The system, as it is implemented so far, does not take into account issues like authentification and security. Moreover, it has to be ensured that no conflicting update messages cross each other on the network and thus the interorganizational process gets into an inconsistent state. And finally, the requirement of autonomous execution means that PMSs which participate in a cooperation may be temporarily down without affecting the other systems. If a system is down, its local process state cannot change and therefore there is no need to update the read-only view which is maintained by other systems on that process. But a PMS must be able to distinguish between a broken network connection and a peer system that is currently inactive, so a more elaborated communication protocol is needed here. It is not our aim to solve all these problems. Distributed system research has dealt with these problems and came up with a number of well known solutions for them. In our work, we have concentrated on elaborating the requirements which a PMS should meet to support delegation, and we have shown how delegation mechanisms can be implemented.

References

1. C. Basile, S. Calanna, E. Di Nitto, A. Fuggetta, and M. Gemo. Mechanisms and policies for federated PSEEs: Basic concepts and open issues. In Carlo Montagnero, editor, *Proceedings of the 5th European Workshop on Software Process Technology*, LNCS 1149, pages 86–91, Nancy, France, October 1996. Springer.
2. Thomas Bauer and Peter Dadam. A distributed execution environment for large-scale workflow management systems with subnets and server migration. In *Proceedings of the second IFCIS conference on Cooperative Information Systems (CoopIS'97)*, pages 99–108, Kiawah Island, South Caroline, USA, June 1997.
3. Israel Ben-Shaul and Gail E. Kaiser. Federating process-centered environments: the OZ experience. *Automated Software Engineering*, 5:97–132, 1998.
4. Workflow Management Coalition. Workflow client API specification. Technical report, Workflow Management Coalition, http://www.aiim.org/wfmc/standards, July 1998.
5. Workflow Management Coalition. Process definition interchange. Technical report, Workflow Management Coalition, http://www.aiim.org/wfmc/standards, October 1999.
6. Gianpaolo Cugola and Carlo Ghezzi. Design and implementation of PROSYT: a distributed process support system. In *Proceedings of the 8th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, California, USA, June 1999. Stanford University.
7. Wolfgang Deiters and Volker Gruhn. Process management in practice, applying the FUNSOFT net approach to large-scale processes. *Automated Software Engineering*, 5:7–25, 1998.
8. Mark Dowson. Integrated project support with IStar. *IEEE Software*, 4:6–15, November 1987.
9. Peter Heimann, Gregor Joeris, Carl-Arndt Krapp, and Bernhard Westfechtel. DYNAMITE: Dynamic task nets for software process management. In *Proceedings of the 18th ICSE*, pages 331–341, Berlin, March 1996. IEEE Computer Society Press.
10. Ric Holt, Andreas Winter, and Andreas Schürr. GXL: Toward a standard exchange format. In *Proceedings of the 7th Working Conference on Reverse Engineering (WCRE 2000)*, Brisbane, Australia, November 2000.
11. Dirk Jäger. Generating tools from graph-based specifications. *Information and Software Technology*, 42:129–139, 2000.
12. Dirk Jäger, Ansgar Schleicher, and Bernhard Westfechtel. AHEAD: A graph-based system for modeling and managing development processes. In Manfred Nagl and Andy Schürr, editors, *AGTIVE — Applications of Graph Transformations with Industrial Relevance*, LNCS 1779, Castle Rolduc, The Netherlands, September 1999. Springer-Verlag.
13. Carl-Arndt Krapp. *An Adaptable Environment for the Management of Development Processes.* PhD thesis, RWTH Aachen, Aachen, Germany, 1998.

14. Heiko Ludwig and Keith Whittingham. Virtual enterprise co-ordinator: Agreement-driven gateways for cross-organisational workflow management. In Dimitros Georgakopoulos, Wolfgang Prinz, and Alexander L. Wolf, editors, *Proceedings of the International Joint Conference on Work Activities, Coordination and Collaboration (WAC-99)*, pages 29–38, N.Y., February 22–25 1999. ACM Press.
15. P. Muth, D. Wodtke, J. Weissenfels, A. Kotz Dittrich, and G. Weikum. From centralized workflow specification to distributed workflow execution. *JIIS – special issue on workflow management*, 10(2), March 1998.
16. Andy Schürr. Introduction to the specification language PROGRES. In Manfred Nagl, editor, *Building Tightly-Integrated Software Development Environments: The IPSEN Approach*, LNCS 1170, pages 248–279, Berlin, Heidelberg, New York, 1996. Springer Verlag.
17. Pierre F. Tiako. Modelling the federation of process sensitive engineering environments: Basic concepts and perspectives. In *Proceedings of the 6th European Workshop on Software Process Technology (EWSPT'98)*, Weybridge, UK, September 1998.
18. Pierre F. Tiako and Jean-Claude Derniame. Modelling trusted process components for distributed software development. In *Proceedings of the International Process Technology Workshop (IPTW)*, Grenoble, France, September 1999.
19. Wil M. P. van der Aalst. Interorganizational workflows – an approach based on message sequence charts and Petri nets. *Systems Analysis, Modeling and Simulation*, 34(3):335 – 367, 1999.
20. Wil M. P. van der Aalst. Process-oriented architectures for electronic commerce and interorganizational workflow. *Information Systems*, 24(8):639–671, 1999.