Sharing Product Data among Heterogeneous Workflow Environments

Markus Bon, Norbert Ritter, Theo Härder

Department of Computer Science
University of Kaiserslautern
67653 Kaiserslautern, Germany
e-mail: {bon | ritter | haerder}@informatik.uni-kl.de

Abstract

Nowadays, we increasingly face the situation that possibly heterogeneous workflow environments must be integrated in order to support company-internal business processes as well as cooperation among different enterprises more effectively. Thus, interoperability of heterogeneous workflow management systems (WfMSs) is the major goal of one of our projects conducted with industrial cooperation partners. In this paper, we, on one hand, report on how our approach supports integration of heterogeneous WfMSs in general, and, on the other hand, detail the very important aspect of allowing workflow applications associated with different, possibly heterogeneous workflow environments to share product data. Traditionally, the management of product data is beyond the scope of a WfMS and is left to the individual workflow applications. However in a multi-WfMS environment, additional control facilities are needed enabling multiple and potentially different WfMSs to share data for cooperation purposes. We introduce different approaches for product data control in heterogenous WfMS environments. As an important result of our work, global dataflow dependencies between workflows in different environments may be properly modeled and automatically controlled by extending the local workflows by activities, which provide the transparent supply of data. Fortunately, in most cases only few adaptations of the local workflow types are necessary to achieve this goal.

Keywords: WfMS, Heterogeneity, Integration, Interoperability, Product Data Management.

1 Motivation

During the last few years, WfMSs were successfully used to automate and to improve the execution of classic business processes [Sche94]. Further on, there are also attempts to use workflow techniques supporting engineering environments [BDS98]. Since modern enterprises often consist of various more or less independent companies, heterogenous workflow environments are a natural consequence of the technological development. All the participating companies contribute to the overall enterprise's business goal, and the closer the separate workflows have to interact with each other, the stronger different kinds of dependencies between them are formed. Unfortunately, maintaining these inter-workflow dependencies exceeds the capabilities of the WfMS used. Therefore, necessary work to satisfy these dependencies has to be done ‘by hand’.

In our project, we examine possibilities for maintaining these dependencies automatically. The different parts of an enterprise (with potentially different WfMSs) are called ‘islands'. Not only the WfMS used is associated with an island, but also resources like users (identified by roles), applications, or database and product data management systems (PDMSs). A new way has to be explored to specify knowledge about existing dependencies between the islands and to replace the manual maintenance by some kind of automated (or at least semi-automated) mechanism. The islands have to ‘obey' some kind of authority – a global system control. Rather than defining new global workflows to include all dependencies in question (the ‘top down’ method), we intend to identify these dependencies between existing workflows, describe them in an explicit way and handle them automatically (‘bottom up’ integration). As a result, the necessary adaptation of the underlying local workflows should be as minimal as possible.

One of the main aspects within this problem area is the flow of data between islands. After describing our general approach for combining heterogenous workflows in Section 2, we will focus on dataflow dependencies and how they may be handled. In Section 4, we will have a closer look to the specification of data granules in order to extract them from a PDMS and to transfer them to the requesting location. Section 5, describes the key part of our proposal, that is the different aspects of automated data supply, thereby assuming only minimal workflow adaptation. After showing how SOAP may be used as a transport protocol we present a short summary of our ideas and conclusions.

2 General Approach to Combine Heterogeneous Workflows

To accomplish island-spanning workflows, we have to use global knowledge about the process as a whole; the information available at a specific island is not sufficient. Furthermore, we want to keep the changes affecting the local workflows as small as possible. Because of lack of global knowledge, distributed solutions for global workflow control require complex protocols and massive adaptations of the participating WfMS [Schu99]. Therefore, we recommend a logically centralized 'omniscient' component which we denote coordinator [BRZ00]. The coordinator has to be informed about dependencies between local workflows in order to be able to control and support the resulting actions between the associated islands. In particular, the coordinator has to be provided with knowledge about:

- *global controlflow*: causal dependencies of activities in different workflow types;
- *global dataflow*: dependencies between workflow types specifying the flow of data, especially originating workflow activities and corresponding destinations;
- *execution locations* of (local) workflows;
- *execution progress*: monitoring information about the status of every workflow instance as well as of the global process;
- *temporal restrictions*, e.g., deadlines, resulting from inter-workflow dependencies, *and exceptions*, e.g., special recovery steps to be performed if corresponding restrictions are violated;
- *authentication* of systems participating in the global process in order to guarantee that none of the remotely accessed data is abused.

This knowledge helps the coordinator to fulfill the tasks of a reliable and neutral mediator. Thus, the coordinator is responsible for:

- registration of the local workflows which together establish the global process;
- island-spanning coordination of local workflows, i.e., identification/authentication of partners for information exchange as well as control of the corresponding communication processes;
- monitoring the global process by monitoring each of the participating local workflows;
- suspending/resuming local workflows in order to fulfill (island-spanning) temporal restrictions;
- performing (island-spanning) exception handling in the case of failures.

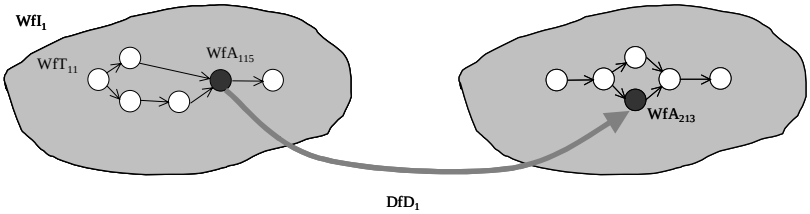
After having presented major ideas of our approach, we focus on the automatic handling of dataflow dependencies in the remainder of this paper.

3 Dataflow Dependencies

The main problem we have to face is providing workflow applications with data produced by some external source. In this paper, we will focus on engineering activities, but this makes no major difference compared with the 'common case'. We assume that the source is an island with workflow support, too. As shown in Figure 1, dataflow dependencies affect two different 'layers' of processing (Figure 1). Initially, at the type layer the dependencies between workflow types at different islands are identified; later on these dependencies have to be materialized at the instance layer.

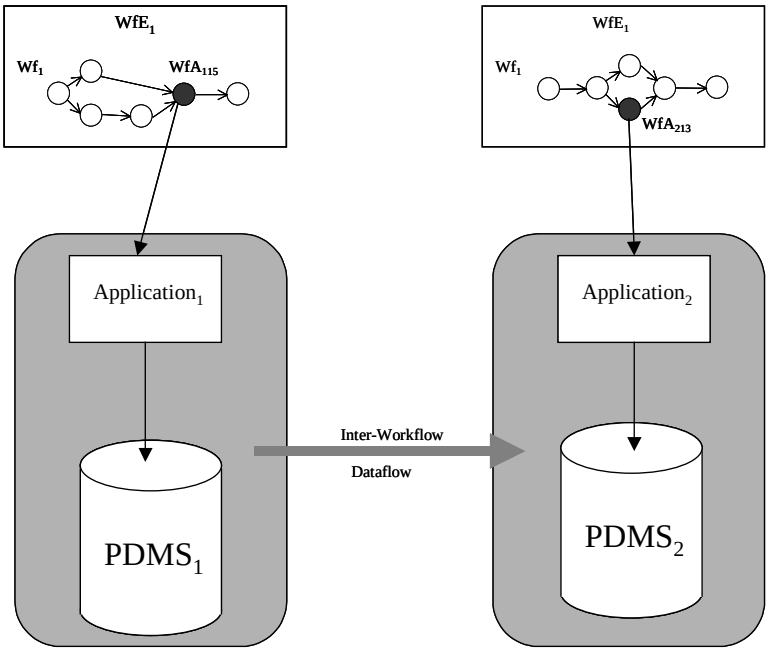
In our example, we consider the workflow types WfT_1 and WfT_2 (Figure 1A). A dataflow dependency may be viewed as a special relationship crossing workflow type borders. For example, dataflow depend-

A. Type Layer



WfI_i: Workflow Island i
WfT_{ij}: Workflow Type j at Island i
WfA_{ijk}:Workflow Activity k in Workflow Type j at Island i
DfD_i: Dataflow Dependency i

B. Instance Layer



WfE_i: Workflow Engine at Island i
Wf_i: Workflow at Island i (Instance of appropriate Type)
$Application_i$: Application called by Instance of Workflow Activity
$PDMS_i$: Product Data Management System used by Application i

Figure 1 Dataflow Dependencies

ency DfD_1, as shown in Figure 1, links two workflow activities of the two different workflow types by a directed arc. The directed arc indicates that output data of activity WfA_{115} is required as additional input for activity WfA_{213}. 'Additional' means that the appropriate data (called cooperation data) is strongly required as input by some application used in WfA_{213} without being directly registered in the internal dataflow specification of WfT_2. The specification of an inter-workflow dataflow dependency consists of:

- the workflow types concerned;
- the workflow activities linked by the dataflow dependency;
- a suitable specification describing the cooperation data;
- the name of the data source (in general a PDMS) and a specification of the data access (database query or PDMS function) to be performed in order to extract the cooperation data;
- the mechanism to be used to transport the cooperation data from the source island to the target island;
- the data target (in general a PDMS, too) and a specification of the operation to be performed in order to integrate the cooperation data.

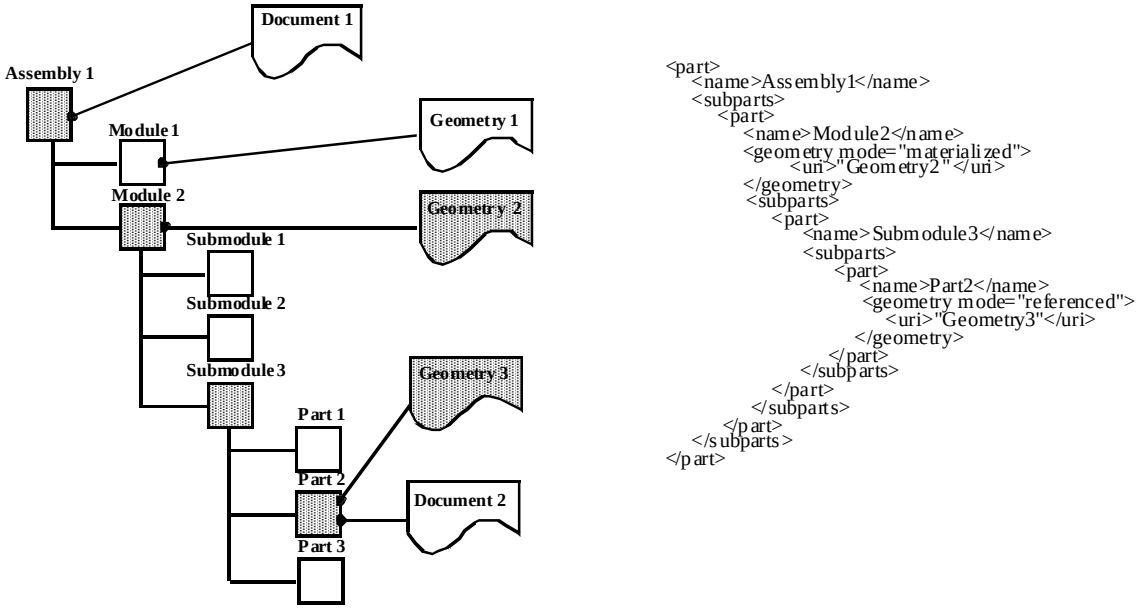


Figure 2 Identification and Specification of Cooperation Data

At the instance layer, workflow instances (shortly workflows) are to be considered, as Wf_1 and Wf_2 in our example (Figure 1B). Since there may be multiple instances of a single workflow type, the identification of cooperation pairs, each associating exactly one source workflow (Wf_1 in our example) with exactly one target workflow (Wf_2 in our example) w. r. t. exactly one dataflow dependency specification (DfD_1 in our example), is a prerequisite for the proper handling of dataflow dependencies. Enabling the coordinator to maintain cooperation pairs requires:

- local workflow management systems to *register* newly initiated workflow instances at the coordinator's registry and
- *human assistance* in order to associate workflow instances with each other.

The mentioned human assistance can help in the following way. Whenever a workflow instance is created and registered, the coordinator, first, creates a list of specified dataflow dependencies referring to the workflow type of the newly created workflow instance in the role 'source' or 'target'. For each pair (dataflow dependency, role), a list of potential cooperation partners can be generated regarding the set of currently running workflow instances which have not yet been assigned to a cooperation partner. Thus, for each pair (dataflow dependency, role), the user may select a cooperation partner from the list offered by the coordinator or can indicate that the corresponding partner has not yet been initiated.

4 Specification of Data Granules

As stated in Section 3, specifications of the cooperation data and corresponding means for extraction are parts of the dataflow dependency specification. Thus, a (specification) language is needed, powerful enough to specify exactly that amount of data needed for cooperation purposes. As indicated in Figure 2, product data is typically modeled in a hierarchical way. The root of the product tree represents a kind of handle for the product as a whole and may consist of composite parts ('assemblies'), which are simple parts ('components') or assemblies themselves [Step92]. By applying a nesting of components/assemblies, a tree structure emerges (we call it the 'product tree'). Its inner nodes are formed by assemblies, the leaves by components. At each level of the tree, additional objects may be attached, for example, documents describing a part in detail or geometry data needed to process a part by CAD tools. These elements may be very large. Hence, for better performance, only data which is really needed should be transferred. Furthermore, access control privileges may limit the selection of components/assemblies during the extraction process (as will be further detailed in the following sections). By performing the data granule specification, we typically have to select a significant subset of the product tree describing the elements to be transferred. For instance, the gray parts in Figure 2 are chosen as cooperation data for some dataflow dependency. The selection of such a granule is not a simple task, since specific knowledge

about the product structure at the source side as well as at the target side is mandatory. Furthermore, the processing characteristics of the target application also have to be considered. Therefore, major parts of the specification have to be supplied manually by a human expert providing all this knowledge and cannot be generated automatically. After identification of the relevant parts, an XML-like language can be used to specify the cooperation data required. Figure 2 shows an XML-based specification example. For this purpose, we have identified the following properties a specification language should embody:

- *descriptive specification*: powerful constructs allowing to describe the relevant data resp. the desired sub-tree have to be offered;
- *hierarchical data structures*: in general, product data is build up in a hierarchical way; therefore, the description language should allow to separately describe every hierarchy level as well as the hierarchical structure of the entire tree;
- *extensibility*: specifications must be mapped to various, heterogeneous systems (e.g., in order to extract and integrate cooperation data from/into PDMSs); although most PDMSs support similar models for storing product data, adaptations may help to reduce mapping overhead;
- *simplicity*: for ease of specification as well as efficient and, as far as possible, automated processing, the language should be easy to parse.

Obviously, it might not be convenient for the human expert to use this kind of language directly. Therefore, assistance by graphical tools is certainly helpful.

5 Automation of Data Supply

After having introduced the notion of dataflow dependencies as well as having clarified the necessity of a language flexibly supporting the specification of exactly that amount of data needed for cooperation purposes in the previous sections, we now want to tackle the problem of data supply. This problem has several dimensions to be discussed in the following subsections.

Level of Automation

In general, we see two completely different approaches of data supply automation. First, actions can be taken at the *level of workflow types* by extending the source workflow type and the target workflow type (of a dataflow dependency) by additional, possibly generated (workflow) activities performing the data supply. Second, automated actions can be taken at the *level of workflow applications* by directing data access operations of the target workflow application to the source PDMS for cooperation data access.

Pursuing the first approach (automation at the level of workflow types) requires to take the following actions:

- identifying and extracting the cooperation data from the source PDMS by mapping the cooperation data specification (as contained in the dataflow dependency specification) to source PDMS access operations;
- converting, packaging and transferring the cooperation data from the source side to the target side;
- integrating the cooperation data by mapping the cooperation data specification (as contained in the dataflow dependency specification) to access operations of the target PDMS.

To what extend corresponding activities can be provided automatically will be the subject of the following section. We will further see that identification, extraction, conversion, packaging and transfer actions are performed by workflow activities which have to be newly made available and integrated into the source workflow type. Integration activities are to be provided by corresponding adaptation of the target workflow type. Note that all these actions are subjected to temporal restrictions, which can be partially fulfilled by integrating new workflow activities into the original workflow types at the right places, but also require some kind of global workflow synchronization the coordinator is in charge of.

Regarding the second approach (automation at the level of workflow applications) we assume the cooperation data to remain in the source PDMS. In this case, suitable mechanisms for remote data access have to be provided [AGL98]. Note that it is in general not feasible to modify the target application in a way that cooperation data access is directly routed to the source PDMS. Beyond, there are two feasible possibilities of enabling the target application to access remotely stored cooperation data objects. First, some kind of proxy objects can be stored within the local PDMS providing interfaces of cooperation data objects and transparently performing the remote access. Second, data access operations of the target application are filtered and each operation addressing cooperation data is redirected to the remote PDMS at the source side. Although feasible, the second approach is not as elegant as the first one, since an external filter is needed. Furthermore, there are the following problems w.r.t. both approaches:

- mechanisms must be provided translating API calls of the target PDMS into semantically equivalent calls to the source PDMS;
- these calls lead to possibly frequent system border crossings, which, in turn, might be difficult, since many islands are protected by security measures such as firewalls;
- as the cooperation data remains at the source side, especially access to large attachments (see Section 4) must be expected to lead to increased waiting times for remote users.

Before we can take a decision about which level of automation is the better choice w.r.t. to data supply, we have to take into account the following aspects.

Product Data Management Systems

A favorable setting is the usage of identical PDMSs at both islands. Thus, both systems offer the same API and the same model for representing the product data structures and there is no need for a cooperation data mapping. If the systems differ, however, things are substantially more complex. The cooperation data has to be mapped from the source PDMS data model into the one used by the target system. This mapping is obviously a prerequisite of data transfer (which is the major characteristic of the workflow type level of automation). It is also helpful for determining proper actions of access propagation (which is the major characteristic of the workflow application level of automation). However, it seems to be more feasible to provide a generator, which automatically generates data transfer activities from the dataflow dependency specification and a mapping given by a human expert than to provide a generator for rewriting the data access operations of the target PDMS for access propagation purposes.

Cooperation Data Access

Another problem dimension is whether the cooperation data is accessed at the target side in a read-only or read/write manner. The simplest solution is achieved by restricting accesses to read-only. Thus, no data changes have to be propagated back. Nevertheless, suitable arrangements have to guarantee that the target application accesses the right version of the cooperation data. This problem primarily arises in the case of access propagation, since the version seen by the target application has to be frozen until it is no longer needed. Changes performed by applications running at the source island must be isolated from the target application. Fortunately, management of multiple versions is a standard functionality of most PDMSs.
Read/write access is much more difficult to be handled, since the modifications performed by the target application have to be propagated back to the source system. This update propagation, however, can only be performed in a safe and consistent way, if appropriate synchronization mechanisms across island borders are provided. Such a distributed concurrency control component would require considerable changes in the participating PDMSs. Hence, this does not seem to be a feasible approach. A more practical solution exploits the versioning mechanisms supported by most PDMSs allowing to propagate the changes as new versions of the product. This, however, requires that other applications can be prevented to modify the product as long as the target application has not propagated its changes, since merge operations are usually not feasible due to complexity.

Access Control

Another important issue for all kinds of data supply is access control. Since product data is crucial for the company's work, unauthorized access has to be prevented, no matter if it is initiated at the source or at the

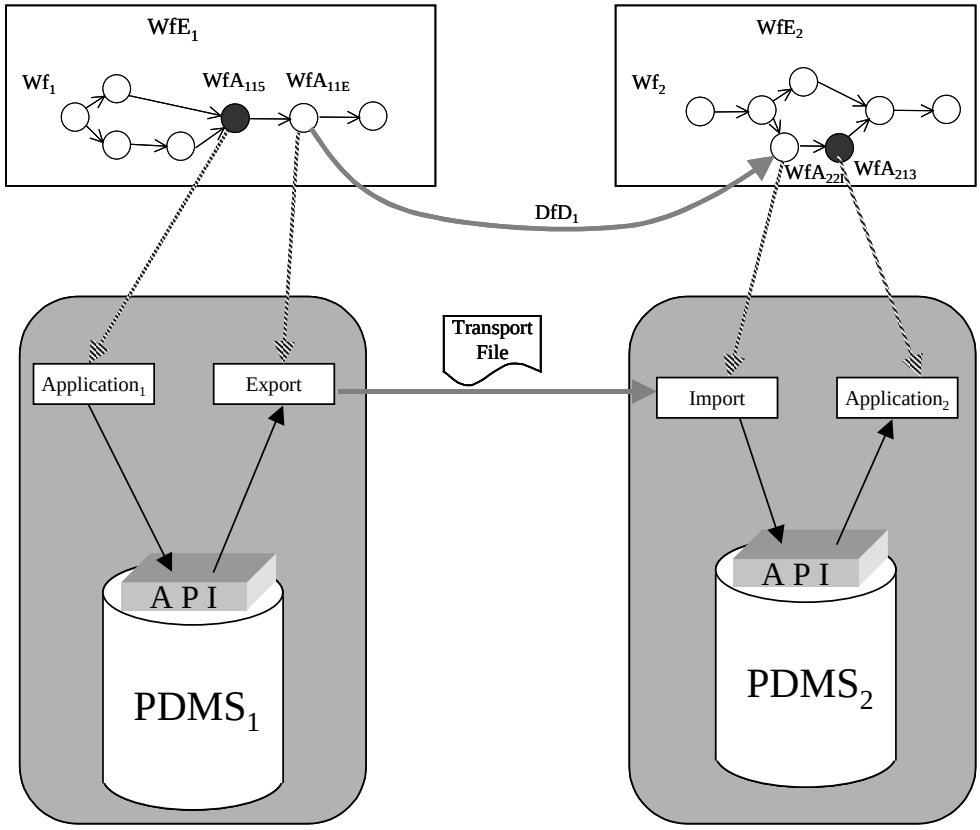


Figure 3 Transfer of Cooperation Data (Instance Layer)

target side. In the case of data transfer this, on one hand, means that in the adjusted source workflow type a staff member must be assigned to the newly provided activity for data extraction, who has at least read permissions to the target PDMS. On the other hand, in the adjusted target workflow type a staff member must be assigned to the newly provided activity for data integration, who has at least insert permissions to the target PDMS, and, additionally appropriate access rights must be granted to the person/role, who is responsible for the target application. In the case of access propagation, the staff member responsible for the target application has to be registered at the source side and to be granted sufficient access rights to the source PDMS. However, since the islands are assumed to be different environments, this again seems to be the worse solution.

Coming back to the still open question, if the workflow type level or the workflow application level is the better choice w.r.t to data supply automation, the previous discussions show that the first approach (automation at the level of workflow types) is less problematic, since workflow management systems are generic by nature and, therefore, can be more easily adjusted than workflow activities. For this reason, we favor the first approach and detail it in the following section.

6 Automation by Workflow Type Adaptation

In this section, we examine how workflow types can be adjusted to support automated processing of dataflow dependencies. Because of space restrictions we can only consider the scenario of identical PDMSs at source and target side and read-only access by the target application. Remember, workflow type adaptation means cooperation data transfer where all original workflow activities and corresponding data access operations remain unchanged.

For explanation purposes, we refer to the scenario illustrated in Figure 1 and corresponding notions in the following. Since the dataflow dependency DfD_1 refers to activity WfA_{115} of workflow type WfT_{11} as source activity and to activity WfA_{213} of workflow type WfT_{21} as target activity, the data extraction activity to be newly provided has to be incorporated into WfT_{11} right after WfA_{115} and the new integration activity has to be integrated into WfT_{21} before WfA_{213}. The resulting adapted workflow types can be

imagined from the illustration in Figure 3, although only the instance layer is shown. WfA_{11E} represents the extraction activity and WfA_{22I} the integration activity, both newly created in order to automatically perform data supply actions at workflow runtime. Obviously, the process of adapting the workflow types is very easy and can be performed on every WfMS.

After the adaptation of the workflow types, we now consider the transfer of cooperation data at the instance layer. Here, besides the involved WfMSs, the coordinator plays an important role, too. There are 6 phases to be processed (cf. Figure 3):

1. Wf_1 and Wf_2 are (not necessary simultaneously) initiated. The coordinator is informed about the creation of the new instances and gets the information needed to relate cooperation partners.
2. The source application ($Application_1$) produces data and stores it into its local PDMS ($PDMS_1$).
3. The export application identifies/extracts the cooperation data using the cooperation data specification contained in the dataflow dependency specification and creates a transport file.
4. The transport file is transferred from the source island (WfI_1) to the target island (WfI_2). The coordinator is involved to provide information about the target location.
5. After having received the transport file, the import application extracts the cooperation data and stores it into the local PDMS ($PDMS_2$).
6. The target application starts its work and accesses data using the local copy.

Dynamic Specification of Cooperation Data

As already mentioned, the specification of the cooperation data is part of the overall dataflow dependency specification. Unfortunately, this *static* specification is not really satisfying in all situations. In fact, we also have to consider scenarios in which the concrete set of cooperation data cannot be determined before runtime. For example, a certain result status of the source application may determine the cooperation data. Such situations require some kind of generic data export and import activities which are able to process dynamic cooperation data specifications.

Import and Export Applications

Import and export applications perform the actual work of data extraction, transfer, and integration. We see three different approaches for data extraction and transfer.

- Many PDMSs support the extraction of objects in some kind of proprietary format. This can be exploited by an export activity performing the following three steps:

  1. extraction of the product tree into a file (proprietary format);
  2. filtering the tree w.r.t the cooperation data specification;
  3. contacting the coordinator to get the target address and sending the file.

  The critical point within this processing is the filter to be used in step 2. It must at least be able to 'understand' the proprietary data format used by the PDMS for data export as well as the specification of the cooperation data. It must also be able to remove objects from the file created in step 1 without invalidating the file. Note that for each export data format of a PDMS a separate filter is needed. Information about applying the cooperation data specification to the contents of the extracted file is 'hard coded' within the filter.

- Since most PDMSs use a relational database management system (DBMS) as data store it is also possible to extract data using the SQL interface. Based on the tree structure described by the cooperation data specification, a set of SQL statements may be generated. Database schema information needed for the generation of these SQL statements can be selected by metadata lookups in the database system. After the evaluation of the generated SQL statements by the DBMS, the resulting data can be packaged into a transport file in some neutral data format (for example by using XML). Using the SQL interface has proved to be quite an efficient way for our purposes [MDJF01].

- It is also possible to combine both approaches. Using the SQL interface, an internal representation of the whole product tree can be created, for example using the DOM [W3C01]. Afterwards, it can be reduced to the desired volume quite effectively, because all subsequent operations are directly performed using the product tree created in main memory. Since the SQL statements for creating the whole product tree rarely change, they may be pregenerated.

The application used to insert the cooperation data into the target PDMS is easier to provide. There is no additional mapping to perform, since the two PDMSs are assumed to be of the same type. If the proprietary file is available, the data can directly be integrated by calling the corresponding API function of the PDMS. Otherwise, the structural information provided by the transport format may be exploited to create appropriate INSERT statements. Especially XML seems to be well suited for such purposes because of its hierarchical structures. The schema information needed to generate the mentioned INSERT statements can again best be selected by performing metadata lookups at the relational DBMS used by the target PDMS.

Access Control

As already mentioned, cooperation data may be critical from the viewpoint of the enterprise. Therefore, it is, on one hand, important to grant sufficient access rights for the newly generated export and import activities as well as for the target application. On the other hand, access rights must be as strict as possible in order to prevent unauthorized access. However, since the data is shifted between different environments, some amount of trust is also required between the human partners. Only in some cases, precautions like disabling parts of data while keeping the relevant parts are possible [Naw01].
For coding (protection during transmission) and authentication, techniques using asymmetric key pairs may be used. To each island a private key is assigned. A public key ring may be accessible through the coordinator as trusted 'key broker'. Before sending the cooperation data, the sender gets the target island's public key. After having encoded the transport file with this key, only the target island is able to decode it again. In the case the sender is supposed to identify himself, too, the file is also encoded using the source island's private key. The target island may now use the matching public key and identify the source securely. The problem with this approach is the high costs in time. Especially encoding large files, e.g., containing CAD geometries, may last very long. Therefore, according to the security needs the encoding may be skipped for less critical data.
After successfully importing the cooperation data, the import application has to adjust proper access rights for data access at the target side. For this purpose, the organizational manager provided by most WfMSs can be exploited. Since we restricted our considerations to read-only access, it is sufficient to grant read access to the corresponding group of people/applications.

7 Transfer Protocols

Communication between workflow islands requires communication protocols. There are various possibilities, for example some kind of binary protocol (using TCP) for synchronous communication or message queuing for asynchronous communication [SZ98]. Furthermore, using remote procedure calls (RPC) may be a good choice, too. Unfortunately, the security procurements such as firewalls complicate things for protocols like CORBA or RMI, but using SOAP seems to help in this matter.
SOAP messages are transferred using HTTP. Upon arrival a servlet provided by an application server takes care of the further processing. The application server may be integrated quite easily with most of the common web servers available (for example, TOMCAT may be docked to an APACHE web server). Therefore, SOAP messages may be delivered using port 80, a port which is unlocked at almost every firewall. Afterwards, the message is handed over to the responsible servlet (the SOAP message handler).
A sample scenario may look as follows. An export application (as introduced in Section 5) prepares the cooperation data for transfer. Then, a servlet wraps the data as 'payload' into a SOAP message. This message is transmitted using HTTP. At the target island, another servlet 'unwraps' the cooperation data for further processing. Using SOAP is substantially slower than using CORBA or RMI [GSC+00]. On the other hand, the possible use of port 80 avoids many firewall problems and is nevertheless a big pro for using SOAP.

8 Conclusion

In this paper, we examined how dataflow dependencies in distributed and heterogeneous workflow environments may be handled. After a short motivation, we introduced our general approach towards providing interoperability of heterogeneous workflows. Since control information existing in the local systems is not sufficient, we propose the use of a logically centralized component dealing with global dependencies. This coordinator has several duties like supervising global flow of control, resolve dataflow dependencies, monitor the current state of a global process, observe the meeting of deadlines, provide recovery actions, and enable the islands to authenticate themselves. In this paper, we focused on dataflow dependencies. A dataflow dependency specification encompasses: the workflow types, the concerned workflow activities, the data source, the data target, the transport mechanism, and a specification of the cooperation data to be actually transferred. Since product data may be very large in size, it is desirable to minimize the data volume to be transferred. Unfortunately, much knowledge is needed to achieve this goal, so human participation becomes necessary. Furthermore, for purposes of specifying the cooperation data some kind of language is needed. Therefore, we examined the typical structure of product data and described the characteristics required for such a language. Thus completing the definition of dataflow dependencies, we examined how their instantiation may be handled automatically. We identified several 'dimensions': physical transfer of cooperation data (automation at the level of workflow types) vs. access propagation (automation at the level of workflow applications); read-only access vs. read/write; use of identical PDMSs vs. different PDMSs. Furthermore, we examined the adaptation of workflows for automated handling of dataflow dependencies. We found out that there has to be some kind of 'pair management' adding further duties to the coordinator. We detailed the approach of automation at the workflow type level by considering the simple scenario characterized by identical source and target PDMSs and read-only accessed to the cooperation data. For this scenario, we proposed different approaches for exporting and importing data. Fortunately, most extensions to the local workflows can be made available without exploiting WfMS-dependent features. For example, the actual import and export process may be achieved by applications embedded into workflow activities. Hence, modifications of the workflow engine can be avoided. Finally, we introduced SOAP as a protocol usable for communication between islands which are protected by firewalls.
As future work, we want to detail and to prove our concepts to work in a realistic environment. Therefore, we will realize the proposed components for different scenarios and, thereby, hope to gain further experience with this kind of inter-workflow operability.

9 Literature

[AGL98] Abramovici, M., Gerhard, D., Langenberg, L.: Supporting Distributed Product Development Processes with PDM., in: Krause, Heimann, Raupach.(Editors), New Tools and Workflows for Product Development - Proceedings of the CIRP Seminar STC Design, May 1998, Berlin, Fraunhofer IRB Verlag, 1998, pp. 1-11

[BDS98] Beuter, T., Dadam, P., Schneider, P.: The WEP Model: Adequate Workflow-Management for Engineering Processes, Proc. European Concurrent Engineering Conf., Erlangen, 1998

[BRZ00] Bon, M., Ritter, N., Zimmermann, J.: Interoperabilität heterogener Workflows. Grundlagen von Datenbanken 2000: 11-15

[Bur99] Burkett, W.: PDML - Product Data Markup Language - A New Paradigm for Product Data Exchange and Integration, 30.04.1999, www.pdml.org/whitepap.pdf

[GSC+00] Govindaraju, M., Slominski, A., Choppella, V., Bramley, R., Gannon, D.: Requirements for and Evaluation of RMI Protocols for Scientific Computing, in: Proc. Supercomputing 2000 (SC2000), Dallas, 2000

[MDJF01] Müller, E., Dadam, P., Enderle, J., Feltes, M.: Tuning an SQL-Based PDM System in a Worldwide Client/Server Environment, Proc. 17th Int. Conf. on Data Engineering (ICDE2001), Heidelberg, 2001, pp. 99-108

[Naw01] Nawotki, A.: Eine selektive Methode zur Verschlüsselung von Konstruktionsdaten mit Wavelets, Dissertation, Kaiserslautern, logos Verlag, 2001

[Sche94] Scheer, A.-W.: Business Process Engineering: Reference Models for Industrial Enterprises, 2nd ed., Springer, 1994

[Schu99] Schulze, W.: Workflow-Management für CORBA-basierte Anwendungen, Springer-Verlag, Berlin Heidelberg 2000

[Step92] Subcommittee 4 of ISO Technical Committee 184, "Product Data Representation and Exchange - Part 11: The EXPRESS Language Reference Manual," ISO Document, ISO DIS 10303-11, August 1992

[SZ98] Steiert, H.-P., Zimmermann, J.: JPMQ – An Advanced Persistent Message Queuing Service, in: Advances in Databases, Proc. 16th Nat. British Conf. on Databases (BNCOD16), LNCS 1405, Springer, 1998, pp. 1-18

[W3C01] W3C, Document Object Model (DOM) Level 3 Core Specification, W3C Working Draft 1, 3 September 2001, http://www.w3.org/TR/2001/WD-DOM-Level-3-Core-20010913/