BPEL^{DT} - Data-Aware Extension of BPEL to Support Data-Intensive Service Applications

Dirk Habich[1], Sebastian Richly[2], Mike Grasselt[3], Steffen Preissler[1], Wolfgang Lehner[1], and Albert Maier[3]

[1] Dresden University of Technology
Database Technology Group
{dirk.habich,steffen.preissler, wolfgang.lehner}@inf.tu-dresden.de
[2] Dresden University of Technology
Software Engineering Group
{sebastian.richly, ua1}@inf.tu-dresden.de
[3] IBM Boeblingen Lab, Germany
Information Server SWG SOA Integration
{grassel,amaier}@de.ibm.com

Abstract. Aside from business processes, the service-oriented approach—currently realized with Web services and BPEL—should be utilizable for data-intensive applications as well. Fundamentally, data-intensive applications are characterized by (i) a sequence of functional operations processing large amounts of data and (ii) the delivery and transformation of huge data sets between those functional activities. However, for the efficient handling of massive data sets, a significant amount of data infrastructure is required and the predefined 'by value' data semantic within the invocation of Web services and BPEL is not well suited for this context. To tackle this problem on the BPEL level, we developed a seamless extension to BPEL—the 'BPEL data transitions.'

1 Introduction

Web services and the Business Process Execution Language for Web Services (BPEL4WS, BPEL for short) [20] are of interest to both software vendors and researchers. In this paradigm, the functionality provided by business applications is enclosed within Web service software components. Those Web services can be invoked by application programs or by other Web services via internet without explicity binding them. On top of that, BPEL has been established as the defacto standard for implementing business processes based on Web services.

Fundamentally, a process consists of a series of activities. Therefore, BPEL offers a standardized way to describe the functional composition of services to create comprehensive process definitions. A typical service-oriented process example is the *booking a trip* application. Aside from such traditional business processes, the service-oriented approach should also be utilizable in application scenarios with special properties, since such applications can benefit from the service-oriented idea as well.

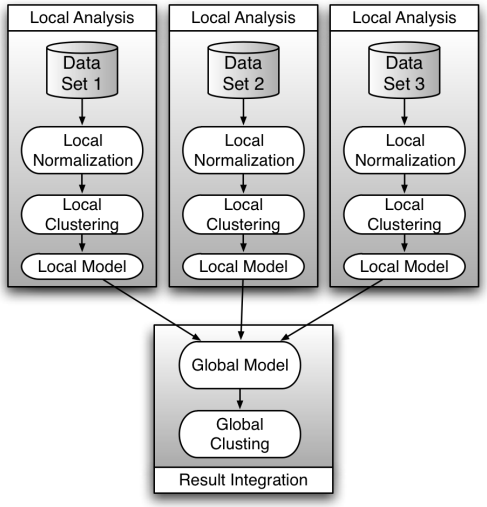


Fig. 1. Two-Phase Clustering Approach for Gene Expression Data Sets

1.1 Data-Intensive Service Applications in SOA Environments

In the context of genome research, the method of gene expression has been used for several years. Related microarray experiments are conducted all over the world, and consequently, a vast amount of microarray data sets are produced, e.g. in biological and medical research addressing a wide range of problems.

After the conduction of the pure experiments, a data analysis process follows to extract useful knowledge. To increase the statistical significance of the extracted knowledge, researchers would like to incorporate various microarray experiments in their analysis processes. In [9], we proposed a *two-phase clustering strategy* for gene expression data sets considering this fact in a special way. A simplified view of our developed concept is illustrated in Figure 1. This analysis process offers some advantages concerning the result quality. A further important aspect is that a lot of work can be done in parallel, thus allowing the efficient process execution.

In general, the utilization of the service-oriented approach as execution environment for such processes provides some advantages with regard to process orchestration and distributed computing. The realization of the presented analysis process would include Web services acting (i) as data providers for microarray data sets and (ii) as functionality providers for normalization and clustering strategies. The microarray data sets are usually n x m matrices, where n is the number of genes and m is the number of samples. Typical values for n and m are: $n > 20,000$ and $m > 100$. A special property of this process is that such large data sets have to be exchanged between participating Web services in the process. A further property is that it cannot be taken for granted that a partic-

ipating Web service can manage the received data in the main memory during the whole processing time. Therefore, database systems are used for the internal processing of such large amounts of data.

1.2 Contribution

The current standards of Web services and BPEL are not efficiently applicable in this special context because both expose some weaknesses concerning the data aspects. One drawback is the predefined 'by value' handling of data within the service invocation of Web services. In service-oriented environments, the Simple Object Access Protocol (SOAP) is commonly used for communication with and between Web services. Embedding massive structured data sets in SOAP is possible but not a suitable solution from the performance perspective, in particular with regard to memory and scalability issues [6, 8, 18, 22]. To overcome this problem in data-centric environments, we developed the concept of *Data-Grey-Box Web Services*[8] which allow the transparent integration of specialized data propagation tools in the service invocation procedure.

The second drawback evolves from the implicitly defined data flows in BPEL and the 'by value' handling. Through the variables concept within BPEL, the concept of centralized data flows is pursued. That means the BPEL server is directly involved as a broker in the exchange of massive data sets between two participating Web services in a process. This may lead to some scalability problems on the BPEL server. To tackle this issue on the BPEL level, we propose a seamless extension to BPEL—the 'BPEL data transitions (BPEL^{DT})'—in this paper. With the help of these data transitions, the data flows are explicitly specified within BPEL processes. More detailed, our proposed BPEL data transitions represent an orthogonal data flow concept to the control flow. Furthermore, these data transitions are optimally exploited during process execution, especially in combination with our *Data-Grey-Box Web services*. Therefore, *BPEL^{DT}* and *Data-Grey-Box Web Services* from a solid foundation to support data-intensive service applications.

The remainder of the paper is structured as follows: In the next section, we go through related work, including an introduction to our developed concept of *Data-Grey-Box Web Services*. In Section 3, we describe our BPEL data transitions from a modeling and execution perspective. Implementation details and an evaluation are provided in Section 4 and Section 5. The paper closes with a summary.

2 Related Work

In this section we give a structured overview of related work. Relevant works come from the fields (i) Web services, SOAP and BPEL, (ii) Data-Grey-Box Web services and (iii) specialized data propagation tools.

2.1 Web Services, SOAP, and BPEL

Web services are an innovative architecture paradigm for applications in a service-oriented architecture (SOA). Fundamentally, Web services are considered as black-box components, since they do not offer any information on how they work; they only expose information on the structure of parameters and data they expect as input and return as result.

The Simple Object Access Protocol (SOAP) is commonly used for communication with and between Web services. The SOAP protocol defines an XML-based format for messages to be used in a Web service invocation. Such messages include a reference to the target service to invoke as well as any number of parameters and data to be transmitted to the service ('by value' semantics). Recently, the performance of the SOAP protocol has received a lot of attention. Proposed techniques try to reduce network bandwidth through compression [4, 7] or other approaches like [24, 18]. Furthermore, serialisation and de-serialisation of XML messages have been in the focus of optimization approaches [1, 2]. Furthermore, SOAP messages with attachements are an option for binary data in the form of image files or encapsulations of other XML documents [10].

On top of Web services, the Business Process Execution Language for Web Services (BPEL4WS, or BPEL for short) provides a comprehensive syntax for describing workflow logic. The BPEL language offers a number of predefined activities to express control flow patterns. The ever necessary data flow is defined implicitly by specifying variables that basically represent input and output messages of activities. In this case, the *assign* activity is of special interest because assignments are used within BPEL to manipulate variables.

To execute BPEL processes, a corresponding execution engine is required. Such a BPEL server controls the service invocations and coordinates the message exchange between Web services. Therefore, BPEL follows the concept of a centralized control flow and a centralized data flow. In this case, the BPEL server is the broker for all SOAP message exchanges between participating Web services in a process. To improve the capabilities of BPEL, a variety of extension have been proposed. One well-known extension is *BPELJ* [21] allowing to include JAVA snippets (code) in BPEL definitions. Furthemore, Maier et al. [14] proposed a similar extension to BPEL, *BPEL4SQL* supporting SQL snippets as BPEL activities and BPEL conditions. This approach can be seen as an embedded SQL approach for BPEL.

Aside from such extensions, there exist methods for (i) data-flow distribution (DFDP-WS) [19] and (ii) decentralizing the execution of composite Web services [5, 16, 17]. The DFPD-WS approach [19] extends the Web service stack with a *Data-Flow Distribution Protocol for Web Services (DFDP-WS)* to exchange data directly without the requirement of a central broker. Nanda et al. [16] propose an approach for partitioning centralized BPEL descriptions into smaller parts that are executed by distributed BPEL engines.

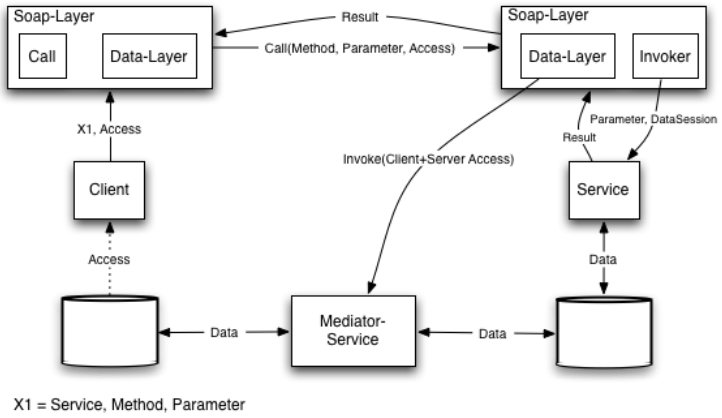


Fig. 2. Invocation Process of Data-Grey-Box Web Services

2.2 Data-Grey-Box Web Services

In [8], we introduced the concept of *Data-Grey-Box Web Services*. In contrast to the original black-box Web services, we enhanced the Web service interface with an explicit data aspect offering more information about the data semantics. Aside from the separation of parameters and data in the interface description, we introduced a novel binding format for structured data. Through this new data binding, the Web service signals that data has not been transfered via SOAP but that there is a separate data layer instead. As before, regular parameters are handed over via SOAP when calling the Web service.

To handle our newly introduced data binding, we extended the SOAP framework with the integration of a novel *data layer* component, as illustrated in Figure 2 that shows the whole invocation procedure. On the client side, enhanced Web service call semantics are necessary. Aside from the transmission of the endpoint and regular parameters in the SOAP message, the client has to deliver access information as references for (i) where the input data is available (input reference) and (ii) where the output data should be stored (output reference). That means our new data binding is translated into no more than two additional parameters for access information for input and output data on the client side. These new parameters are included in the SOAP message for the invocation of Web services. That means instead of propagating the pure data in an XML-marshaled SOAP message, we only deliver access information as data pointers in SOAP.

On the service side, our extended SOAP framework receives the SOAP message and conducts a separation into the functional aspect and the data aspect. As illustrated in Figure 2, the associated data layer calls an appropriate mediator for the data propagation based on the access information of the client and the service. While the data access information of the client can be found in the re-

ceived SOAP message, the data access information for the service instance must be queried from our extended service infrastructure [8]. Fundamentally, a mediator is a neutral partner which is responsible for the data propagation between client and Web service. Examples of mediators are ETL, e.g. IBM DataStage, or data replication tools (see Section 2.3). Those mediators have to be accessible over a standard Web service interface. Advantages of the proposed mediator concept are (i) the optimized data propagation through specialized tools, (ii) the availability of an independent concept enabling an exchange of the mediators, and (iii) the release of data propagation to a third party.

If a Data-Grey-Box Web service receives and returns a data set, two data propagation tasks will be initiated. The first propagation task for the input data is conducted before the pure functionality of the service is invoked. The correlation of this input data to the Web service instance is done by our extended service infrastructure. If the input data propagation task is finished, the functionality is automatically invoked. The last step is the initiation of the data propagation task to deliver the output data to the client.

2.3 Specialized Data Propagation Tools

Fundamentally, the database research community has paid a lot of attention to the field of data exchange between different database systems. A well-known method is the ETL (Extract-Transform-Load) approach, where data from different data sources are loaded into a common data warehouse [23]. Such ETL processes consist of three parts: (i) extraction of data from the different source systems, (ii) application of a series of rules and functions to the extracted data to derive the data to be loaded, and (iii) loading of the data into a data warehouse system. This ETL approach is a data-specialized technique to efficiently transmit structured data to various different data management systems, e.g. relational or XML database systems. A further popular data propagation method is replication [13]. In database systems, this is used to provide redundancy or to balance the load across multiple database servers.

It is already possible today to include such tools in the service-oriented environment. With IBM's information server [12], for example, pre-defined ETL jobs can be provided as Web services. These Web services can then be included in the process orchestration. Disadvantages of this approach lie in the fact that (i) such data operations are explicitly integrated in the control flow and (ii) whenever such an operation is to realize the data exchange between two Web services (source and target WS) in a process, these 3 Web services are then no longer loosely coupled but can only be used together. The reason is that the source and target WS have to be designed appropriately, i.e. the source WS do not deliver and the target WS do not receive the data via the service interface.

3 BPEL Data Transitions

Our proposed Data-Grey-Box Web Services (DGB Services) [8] are only one step in the right direction towards the efficient support of data-intensive service ap-

plications. The subsequent step is the orchestration of DGB Services to create comprehensive processes. Therefore, we present our novel BPEL data transitions (BPEL^{DT}) as a data-aware extension of BPEL in this section. These data transitions are a new explicit link type connecting several DGB Services on the data level. Fundamentally, our newly introduced BPEL data transitions are an orthogonal data flow concept to the control flow, similar to the data aspect in BPMN [3].

3.1 Modeling Perspective

From the modeling perspective, the original BPEL approach follows a two-level programming model [14, 20]. The first level (*lower level*) consists of Web services as executable software components realizing the basic activities. The upper level is often called 'programming in the large;' there, the order of the activities is orchestrated. With our explicit BPEL data transitions, we extend this programming model to a three-level approach. The three levels are:

1. *Lower Level:* This level includes Web services as executable software components; in our case, as *Data-Grey-Box Web Services* with an explicitly published data aspect.
2. *Functional Flow Modeling Level:* On this level, a domain expert models the pure functional process logic without considering data flows. The main advantage is that the domain expert can focus on the functional logic, and the data flow is modeled by data placeholders. The result on this level is a comprehensive functional process description. Such descriptions are essentially not executable.
3. *Data Flow Modeling Level:* This third level represents our extensions, where a data management expert takes the functional process description and annotates this process with all necessary data flows using our BPEL data transitions. The functional description of the process is not changed by this data flow annotation concept.

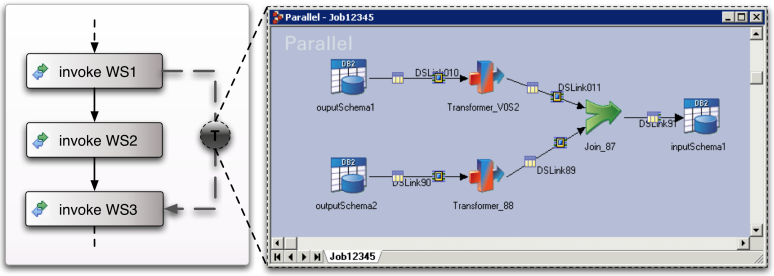


Fig. 3. ETL Process Inside BPEL Data Transition

In Figure 3, a simple process with an explicit data transition between two services is depicted. In this case, the modeling of the illustrated data flow is done with the specialized data propagation tool ETL. As illustrated in the figure, the output of $WS1$ consists of two data sets (*outputSchema1*, *outputSchema2*). Then, two different transformation operations are separately applied on the data sets (*Transformer_V0S2*, *Transformer_86*). Since service $WS3$ expects only one input data set (*inputSchema1*), the transformed data sets are joined together (*Join_87*). Fundamentally, the following data flow relationships between functional operations exist: $1:1$, $1:N$, $N:M$, $N:1$. For example, a $1:N$ relationship signifies that the output data of a Web service is used as input data for a set of Web services.

The result of the three modeling steps is a process definition with an explicit control flow and with an explicit data flow as well.

3.2 Execution Perspective

In Section 2.2, we reviewed our Data-Grey-Box Web Services. These services offer more information on the data aspect than standard Web services do. As we demonstrated in [8], the usage of Data-Grey-Box Web Services creates an obvious performance benefit in the classic client-server scenario. The composition of Web services with BPEL generates more dependencies. These dependencies are built during the additional modeling phase in the form of data transitions.

Standard Web services do not dispose of the qualification to handle these explicit data flows. Up to now, implicit data flows have been used in BPEL engines, resulting in centralized data flows where the BPEL engine is used as a central data broker. With this principle, BPEL engines do not scale well on data-intensive processes. The additional data aspect information in Data-Grey-Box Web Services now enable us to handle these data transitions with specialized tools. Thereby, every data transition joins the data output reference of the producing service with the data input reference of the consuming services. The data propagation is conducted by a neutral mediator. In the composition scenario, the mediator may now handle the propagation and data transformation as well. The additional transformation execution by the mediator has several advantages: (i) the service does not lose its autonomy, (ii) better load balancing is possible, since the BPEL engine is not responsible any longer for the data transformation with BPELJ or for mediation flows, and (iii) already allocated resources are used, since mediators have to handle the data anyway.

If mediators are used to realize our BPEL data transitions with Data-Grey-Box Web Services, then three service invocation protocols are possible. These protocols can be categorized as *pessimistic* ones and *optimistic* ones. Pessimistic means that the allocation of the data input reference of the consuming (target) service is done after the producing service has finished. In case of an optimistic approach, the data input references are allocated before the producing service is executed. The resulting three service invocation protocol approaches are described in more detail in the next section. This description is oriented at a small example BPEL process: two Data-Grey-Box Web Services $WS1$ and $WS2$ are

connected on the control level and the output data of $WS1$ is used as input data for $WS2$. That means $WS1$ and $WS2$ are connected by a control flow and a data flow.

Pessimistic and Control-Flow-Oriented With this method, a decentralized data flow is realized in consideration of the control flow. That means that the control flow triggers the execution of the data flows. As a pessimistic approach, the data propagation is not started until it is really needed, that means until the consuming service is started (see Figure 4). Based on our example, WS1 commits its output reference to the BPEL engine. In the invocation procedure of $WS2$, this reference will be transmitted to $WS2$, calling the mediator (the mediation invocation is done by the introduced data layer in the SOAP framework; see Section 2.2 or [8]). The mediator finally transforms the data and delivers them to $WS2$. After the data propagation has been finished, the functional part of $WS2$ can be executed. The advantage of this approach is that data is not transfered over a broker in vain but the data is stored at $WS1$ until $WS2$ triggers the mediator. In the worst case, the duration can amount to hours or days.

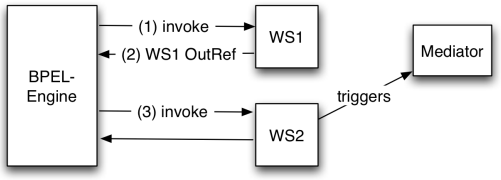


Fig. 4. Pessimistic and Control-Flow-Bound Service Invocation

Semi-Pessimistic With this method, centralized as well as decentralized data flows are possible. Figure 5 illustrates the service invocation protocol approach. In this case, the BPEL server pre-defines all input and output references for the participating Web services independent from control flow or data flow information. That means the data is temporarily stored at a third-party site. In the centralized case, the BPEL server must own a data source to temporarily store the data, while in the distributed case, the BPEL server uses arbitrarily distributed data stores as temporary storage devices.

Fundamentally, this service-invocation approach is compatible to the existing procedure (centralized data flow). However, instead of handling the data 'by value', the data is coordinated by reference. If necessary, the data can be propagated to the BPEL server to get 'by value' access. Disadvantages are (i) the interlocking of the engine with the data flow, and (ii) the fact that two data propagation operations (two mediator calls) are necessary to exchange data between two services. An advantage of this invocation principle is that it is always applicable.

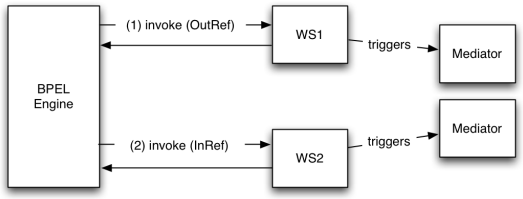


Fig. 5. Semi-Pessimistic Service Invocation

Optimistic and Data-Flow-Oriented The third service invocation protocol approach is optimistic. That means the output data is transfered immediately after its creation in $WS1$. Therefore, the input reference of $WS2$ is committed during the invocation of $WS1$. The data propagation through a mediator is done before $WS2$ is invoked. To realize this approach, the invocation protocol has to be changed in general. We introduce a *preinvoke*, which is illustrated in Figure 6. During this *preinvoke*, the data resources for the input data will be allocated in $WS2$. This input reference will be returned to the BPEL engine and used during the invocation of $WS1$. This *preinvoke* allocation implies a policy mechanism to ensure that only authorized invocations are processed. This approach is efficient if Web services are invoked asynchronously. Thus, a parallel control and data flow is possible.

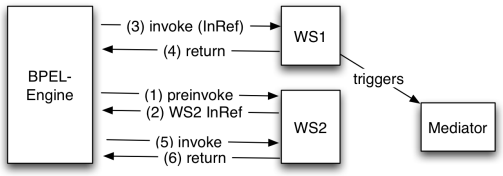


Fig. 6. Optimistic and Data-Flow-Bound Service Invocation

3.3 Summary and Further Investigations

The result of the combination of *Data-Grey-Box Web Services* with our proposed BPEL data transitions is a data-aware SOA environment as illustrated in Figure 7. In such an environment, a functional component (BPEL server) and a data component interact together. Data-Grey-Box Web Services are still loosely coupled as regular Web services. Moreover, the data exchange between participating Web services happens with specialized data propagation tools.

The presented BPEL data transitions with the three service invocation principles create some interesting questions for further research. The additional data-flow modeling phase for data transitions is the starting point of several optimization approaches (see Figure 8). It is desirable to have the possibility to model the data propagation and transformation in an abstract way. From this, it

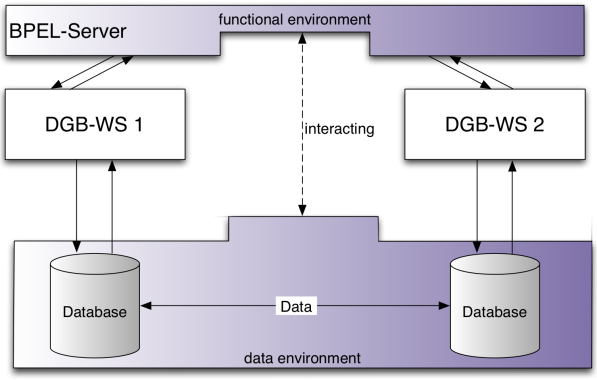


Fig. 7. Data-Aware SOA Environment

should be possible to choose the best strategy to transfer the data, for example, through ETL, stored procedures or other specialized approaches. Depending on the concrete underlying scenario, several optimized process execution plans can be derived from this abstract process model.

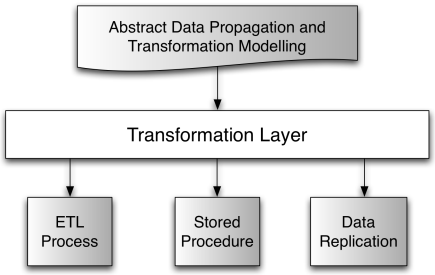


Fig. 8. Optimization Approach

The second point deals with the execution of the process, in particular with the selection of one out of the three presented service invocation principles. The optimistic as well as the pessimistic approach have the disadvantage that one side has to store the data for a long time (in the worst case). The advantage, however, is that in both approaches, only one data propagation operation is required to deliver data from the source Web service to the target Web service, while the semi-pessimistic principle uses two data propagation operations. To tackle this problem from a process perspective, we want to introduce a Data-Grey-Box Web Service policy indicating the time span before and after a certain event during which data can be stored on the service's side. With the help of some process

statistics, we want to determine the best invocation principle. Moreover, some runtime adaptation techniques have to be developed.

4 Implementation Details

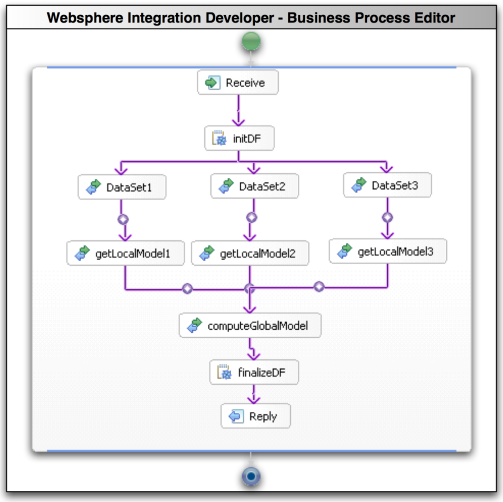


Fig. 9. Two-Phase Clustering with BPEL^{DT}

In this section, we describe our prototypical realization of the entire concept within the Websphere Integration Developer (WID). The BPEL^{DT} process for our two-phase clustering strategy is depicted in Figure 9. Again, we used three different microarray data sets as starting point. These microarrray data sets are persistently stored in a relational database system (IBM DB2), and *Data-Grey-Box Web Services* allow access to them. The normalization and clustering are realized as single *Data-Grey-Box Web Service* (*getLocalModel* Service). The aggregation of the local clustering result to a global clustering result is done by the data-grey-box Web service *computeGlobalModel*. All data-grey-box Web services use a relational database system in order to allow the efficient and scalable processing of incoming data and some tasks within those services are pushed down to the database system [11].

In contrast to the more general implementation presented in [8], *Data-Grey-Box Web Services* are realized in a slightly different way within WID. The separation of parameters and data in the interface description is done by embedding schema descriptions in the DBM format for input and output data in the *types* section. Through naming conventions, a service requestor is able to determine which database-oriented schemas for input and output data are assigned

to each functional operation. Moreover, instead of integrating of the introduced data-layer component within the SOAP framework, each *Data-Grey-Box Web Service* interface includes various administrative operations. These administrative operations are normally hidden from the service requestor by the data-layer component.

BPEL data transitions are prototypically realized using <*flow*>-link types with transition conditions. These transition conditions include the data flow task descriptions. As execution strategy, we use the optimistic and data-flow-oriented approach. We think this strategy offers some advantages with regard to performance compared to the other execution approaches. To enable the optimistic approach, two additional activities are necessary. The first activity, *initDF*, consists of the initialization of all data flow sessions at the participating *Data-Grey-Box Web Services* one th process has been started. That means necessary data resources are allocated at the services. The last activity of the process, *finalizeDF*, closes all data sessions and deallocates all data resources at the services. However, the implementation of the semi-pessimistic strategy is straightforward.

The abstract definition of the data transformation is done with the XML-based Mapping Specification Language (MSL) [15]. The Rational Data Architect (RDA) can be used to specify such transformations based on schema information. For each data flow, an MSL template will be generated containing the data output schema of the data-producing Web service (source schema) and the input schema of the data-consuming service (target schema) (see Figure 10). Those templates have to be refined by the user.

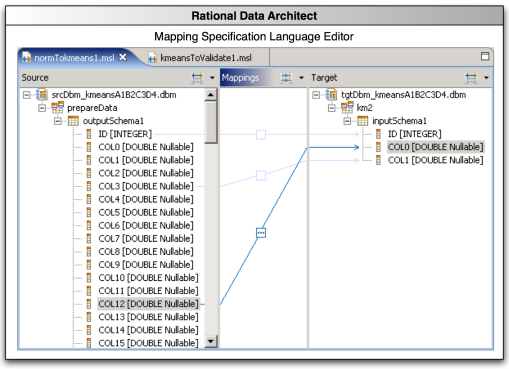


Fig. 10. MSL Template in Rational Data Architect

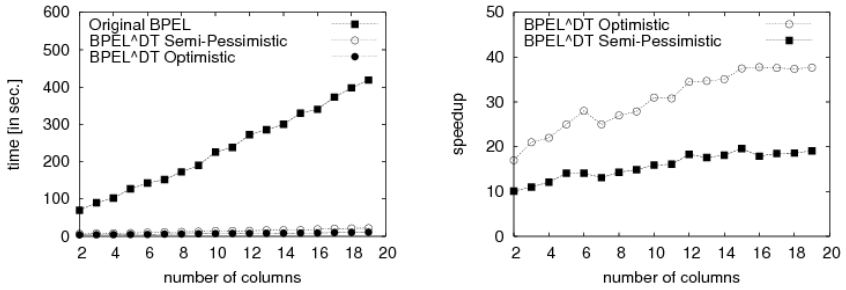
The result of the entire modeling part is a functional process definition with explicit data flows containing transformation specifications in MSL form. During the process deployment, the BPEL process definition is investigated and included data flows will be mapped to parametrized DataStage ETL job descriptions. In

this mapping step, the MSL specification will be considered. That means the data propagation is restricted to ETL for now.

At process runtime, the data flows are executed according to our optimistic and data-flow-oriented strategy. The parametrized DataStage ETL job descriptions are invoked with the current information regarding the data source from our *Data-Grey-Box Web Services*.

5 Evaluation

In this section, we evaluate our proposed BPEL^{DT} approach regarding the performance gain. An evaluation of the data-grey-box Web services is presented in [8]. In this BPEL^{DT} evaluation, we delivered microarray data matrices from a provider service to an analysis service. In the experiment, we measured the time for the data exchange between these two services (ii) with the original BPEL approach, where the BPEL server is the broker, (ii) with BPEL^{DT} with the optimistic execution strategy and (iii) with BPEL^{DT} with the semi-pessimistic approach. The resulting times are depicted in Figure 11(a). In the conducted experiments, we changed the number of columns of the microarray data matrices, whereas the number of rows remained fixed (rows=20, 000).



(a) Evaluation Times BPEL and BPEL^{DT} (b) Speedup of BPEL^{DT}

Fig. 11. Evaluation of BPEL^{DT}

As illustrated in Figure 11(a), both BPEL^{DT} execution strategies are faster than the original BPEL approach. The data flows in case of BPEL^{DT} are executed with the ETL tool DataStage. The resulting speedups compared to the SOAP transmission in case of the original BPEL approach are depicted in Figure 11(b). As expected, the semi-pessimistic execution strategy is slower than the optimistic as well as the pessimistic approach. In the semi-pessimistic case, an additional data propagation is necessary, since the data is temporarily stored at a third position. Therefore, this strategy is slower than the other two execution approaches. In general, this experiment confirms the evaluation results of our *Data-Grey-Box Web Services*. *Data-Grey-Box Web Services* and BPEL^{DT} are

more suitable for data-intensive service applications with regard to performance issues than the original approach.

6 Conclusion

Up to now, the current standards of SOAP, WSDL and BPEL define a 'by value' handling of data in the service-oriented architecture. However, this 'by value' handling is not suitable for data-intensive application. In this paper, we have illustrated our whole service-oriented solution for data-intensive applications. This solution includes *Data-Grey-Box Web Services* [8] which allows the integration of specialized data propagation tools in the invocation process. In this case, the data is not longer handed 'by value'. The main focus of this paper is the introduction of BPEL data transitions to enable an efficient composition and execution of *Data-Grey-Box Web Services*. Aside from theoretical background, we present our prototypical realization, and some evaluation result. Furthermore, we highlight further reseach aspects in this direction.

References

1. Nayef Abu-Ghazaleh and Michael J. Lewis. Differential deserialization for optimized soap performance. In *Proceedings of the ACM/IEEE SC2005 Conference on High Performance Networking and Computing (SC 2005, November 12-18, 2005, Seattle, WA, USA)*, 2005.
2. Nayef Abu-Ghazaleh, Michael J. Lewis, and Madhusudhan Govindaraju. Differential serialization for optimized soap performance. In *Proceedings of the 13th International Symposium on High-Performance Distributed Computing (HPDC 2004, 4-6 June, Honolulu, Hawaii, USA)*, pages 55–64, 2004.
3. Business Process Modeling Notation (BPMN) Information. http://www.bpmn.org/.
4. Min Cai, Shahram Ghandeharizadeh, Rolfe R. Schmidt, and Saihong Song. A comparison of alternative encoding mechanisms for web services. In *Database and Expert Systems Applications, 13th International Conference*, 2002.
5. Girish Chafle, Sunil Chandra, Vijay Mann, and Mangala Gowri Nanda. Decentralized orchestration of composite web services. In *Proceedings of the 13th International Conference on World Wide Web-Alternate Track Papers and Posters(WWW 2004, New York, NY, USA, May 17-20)*, page 134143, 2004.
6. Kenneth Chiu, Madhusudhan Govindaraju, and Randall Bramley. Investigating the limits of soap performance for scientific computing. In *11th IEEE International Symposium on High Performance Distributed Computing*, 2002.
7. Marc Girardot and Neel Sundaresan. Millau: an encoding format for efficient representation and exchange of xmlover the web,http://www9.org./w9cdrom/154/154.html.
8. Dirk Habich, Steffen Preissler, Wolfgang Lehner, Sebastian Richly, Uwe Assmann, Mike Grasselt, and Albert Maier. Data-grey-box web services in data centric environments. In *Proceedings of the 2007 International Conference on Web Services (ICWS 2007)*, pages 976–983, 2007.

9. Dirk Habich, Thomas Wächter, Wolfgang Lehner, and Christian Pilarsky. Two-phase clustering strategy for gene expression data sets. In *Proceedings of the 2006 ACM Symposium on Applied Computing - Bioinformatics Track (SAC 2006, Dijon, France, April 23-27)*, pages 145–150, 2006.
10. Steffen Heinzl, Markus Mathes, Thomas Friese, Matthew Smith, and Bernd Freisleben. Flex-swa: Flexible exchange of binary data based on soap messages with attachments. In *Proceedings of the IEEE International Conference on Web Services (ICWS'06)*, Washington, DC, USA, 2006. IEEE Computer Society.
11. Alexander Hinneburg, Wolfgang Lehner, and Dirk Habich. Combi-operator: Database support for data mining applications. In *Proc. of 29th International Conference on Very Large Data Bases*, 2003.
12. IBM. Ibm information server, 2007. http://www-306.ibm.com/software/data/integration/info_server/.
13. Bettina Kemme and Gustavo Alonso. A new approach to developing and implementing eager database replication protocols. *ACM Trans. Database Syst.*, 25(3):333–379, 2000.
14. Albert Maier, Bernhard Mitschang, Frank Leymann, and Dan Wolfson. On combining business process integration and etl technologies. In *Datenbanksysteme in Business, Technologie und Web, 11. Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme" (BTW 2005, Karlsruhe, 2.-4. März)*, pages 533–546, 2005.
15. Mapping Specification Language. http://www.research.ibm.com/journal/sj/452/roth.html.
16. Mangala Gowri Nanda, Satish Chandra, and Vivek Sarkar. Decentralizing execution of composite web services. In *Proceedings of the 19th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2004, October 24-28, Vancouver, BC, Canada)*, page 170187, 2004.
17. Mangala Gowri Nanda and Neeran M. Karnik. Synchronization analysis for decentralizing composite web services. In *Proceedings of the 2003 ACM Symposium on Applied Computing (SAC03, Melbourne, FL, USA, March 9-12)*, page 407414, 2003.
18. Alex Ng. Optimising web services performance with table driven xml. In *Proc. of the 17th Australian Software Engineering Conference*, 2006.
19. Lucian-Mircea Patcas, John Murphy, and Gabriel-Miro Muntean. Middleware support for data-flow distribution in web service composition. In *Proceedings of the combined Doctoral Symposium and 15th PhDOOS Workshop at the 19th European Conference on Object Oriented Programming(PhDOSS, Glasgow, Scotland, July 25)*, 2005.
20. Specification of BPEL. http://www-128.ibm.com/developerworks/library/specification/ws-bpel/.
21. Specification of BPELJ. http://www-128.ibm.com/developerworks/library/specification/ws-bpelj/.
22. Robert van Engelen. Pushing the soap envelope with web services for scientific computing. In *Proc. of the International Conference on Web Services (ICWS'03)*, 2003.
23. Panos Vassiliadis, Alkis Simitsis, and Spiros Skiadopoulos. Conceptual modeling for etl processes. In *Proc. of the 5th ACM international workshop on Data Warehousing and OLAP*, pages 14–21, New York, NY, USA, 2002. ACM Press.
24. Patrick Widener, Greg Eisenhauer, and Karsten Schwan. Open metadata formats: Efficient xml-based communication for high performance computing. In *10th IEEE International Symposium on High Performance Distributed Computing*, 2001.