Dieses Dokument ist eine Zweitveröffentlichung (Postprint) / This is a self-archiving document (accepted version):

Dirk Habich, Sebastian Richly, Wolfgang Lehner, Uwe Assmann, Mike Grasselt, Albert Maier, Christian Pilarsky

**Data-aware SOA for Gene Expression Analysis Processes**

Erstveröffentlichung in / First published in:


DOI: [https://doi.org/10.1109/SERVICES.2007.28](https://doi.org/10.1109/SERVICES.2007.28)

Diese Version ist verfügbar / This version is available on:

[https://nbn-resolving.org/urn:nbn:de:bsz:14-qucosa2-788490](https://nbn-resolving.org/urn:nbn:de:bsz:14-qucosa2-788490)
Abstract

In the context of genome research, the method of gene expression analysis 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. Having access to this variety of repositories, researchers would like to incorporate this data in their analyses processes to increase the statistical significance of their results. Such analyses processes are typical examples of data-intensive processes. In general, data-intensive processes are characterized by (i) a sequence of functional operations processing large amount of data and (ii) the transportation and transformation of huge data sets between the functional operations. To support data-intensive processes, an efficient and scalable environment is required, since the performance is a key factor today. The service-oriented architecture (SOA) is beneficial in this area according to process orchestration and execution. However, the current realization of SOA with Web services and BPEL includes some drawbacks with regard to the performance of the data propagation between Web services. Therefore, we present in this paper our data-aware service-oriented approach to efficiently support such data-intensive processes.

1. Introduction

In the context of genome research, the method of gene expression analysis 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. Those microarray experiments are popular in biological and medical research as well to address a wide range of problems. One prominent example is cancer research, where microarrays are used to study the molecular variants among tumors with the aim of developing better diagnostics and treatment strategies.

To increase the statistical significance of analysis results, researchers would like to incorporate various microarray experiments in their analysis process. In [6], we have proposed a two-phase clustering strategy for gene expression data sets considering this fact.

Gene Expression Analysis Process

Our developed analysis process with three different data sets is illustrated in Figure 1. We suggest that each microarray data set should be normalized and clustered separately (first phase) and that the combination of the local clustering results to a global result (second phase) yields better results than other approaches. Advantages of our approach are: (i) normalization and clustering can be adjusted for each data set separately, (ii) we integrate various local results instead of data and then analyze the integrated data, and (iii) for each local result, a statistical weighting factor can be determined for quantifying the local results proportion within a global results based on technical and biological quality measures.

An advantage of our two-phase clustering strategy is that the analysis process fits perfectly in a distributed environment. Therefore, we look for an environment/platform supporting an efficient execution of such processes in a distributed environment. Furthermore, the approach should allow a flexible process orchestration to adapt the environment to other biological processes. Therefore, the service-oriented architecture (SOA) is of special interest for this area.

Service-Oriented Architecture

Web services represent an innovative architecture paradigm for applications in a service-oriented architecture (SOA). They are XML-based, independent and modular software components, which are published and usable via Internet.
That implies that applications in general can use services without explicitly integrating them. For this purpose, a collection of protocols and standards are defined. Fundamentally, these are SOAP and WSDL. The Simple Object Access Protocol (SOAP) is a protocol over which Web service communicate and exchange structured information. The Web Service Description Language (WSDL) is an XML-based language for the interface description of Web services.

To facilitate the composition of Web services, the Business Process Execution Language for Web services (BPEL4WS or short BPEL) has been proposed. BPEL provides a comprehensive syntax for describing workflow logic based on Web service invocations. An important aspect of BPEL is that the data flow is implicitly defined by variables [13]. Therefore, BPEL is pursuing the concept of centralized control flow inseparable combined with a centralized data flow aspect [13].

Challenges for Combination

Our previously described gene expression analysis process is a motivating example of a data-intensive process. Fundamentally, data-intensive processes are characterized by (i) a sequence of functional operations and (ii) the transportation and transformation of huge data sets between the functional operations. The combination of data-intensive processes with the service-oriented architecture is beneficial with regard to process orchestration and execution. However, the service-oriented approach includes some drawbacks regarding the performance of the data propagation between Web services.

In service-oriented environments, the Simple Object Access Protocol (SOAP) is commonly used for the communication with Web services. According to the client/server paradigm, clients (service requestors) perform a Web service invocation by passing SOAP messages to a server hosting the corresponding service. These messages include a reference to the target service to be invoked as well as any number of parameters and data to be transmitted to the service. Reply messages may be transmitted either synchronously or asynchronously, from the server back to the client. The SOAP protocol defines an XML-based format for the messages to be used in a Web service invocation.

Database systems still play an important role in service-oriented environments. Either, Web services are used to offer well-defined views on data stored in a database system (e.g. publishing gene expression data sets) or a database system is used as temporary storage system within services, e.g. in a analysis process. The latter case is essential, since it cannot be taken for granted that the received data can be managed in the main memory unit during processing time. If both the service requestor and the service use database systems, the data exchange between them using SOAP is possible but not a suitable solution from the performance perspective. Instead of using SOAP, database technologies for data propagation like ETL (Extract-Transform-Load) [15] or database replication should be applied. Such specialized data transfer approaches outperform the SOAP approach as demonstrated in the evaluation section.

Since the performance including memory and scalability issues is a key factor in real applications, an integration of such data-specialized propagation approaches would be beneficial. In this case, the following challenges arise: (i) the creation of data-grey-box Web services offering more information on the data persistence aspect within the service, (ii) the integration or triggering of data propagation tools in the Web service invocation process, and (iii) the composition and execution of data-grey-box Web services with BPEL. In order to efficiently use our developed data-grey-box Web services with BPEL, we developed a seamless extension to BPEL: the 'BPEL data transition.'

To put it in a nutshell, we present our data-aware service-oriented approach to efficiently support gene expression analysis processes. In detail, we provide general contributions in the following areas:

- We developed data-grey-box Web services allowing us to use specialized data infrastructures for the data propagation between service requestors and Web services. The conducted experiment (see Section 5) indicates that a high performance gain can be achieved. A more detailed description can be found in [5].
Based on this foundation, we introduce our seamless extension to BPEL: the ‘BPEL data transitions.’ These data transitions are a novel approach for explicit data flows in the orchestration language BPEL. With the help of these data transitions, we are able to (1) efficiently support complex data transformations and (2) conduct data exchanges between participating Web services in a process. These data transitions are the main focus of this paper.

The remainder of the paper is structured as follows. In the following section, we present related work. Afterwards, we briefly introduce our data-grey-box Web services as foundation of the data transitions. In Section 4, we propose our extension to BPEL. A short evaluation of our approach is presented in Section 5. The paper concludes with a short conclusion.

2. Related Work

In this section, we review related work for the areas (i) SOAP protocol, (ii) Business Process Execution Language and (iii) data propagation approaches.

SOAP Protocol

Recently, the performance of the SOAP protocol has received a lot of attention from the research community as well as from the industry. One prominent technique to enhance the performance is caching at the client site, server site or in dedicated hardware components [1]. Other techniques try to reduce the network bandwidth requirement by using compression [3], Binary XML or binary metadata [16]. All applied compression techniques, like gZip, XMll, and Millau [4], produce high compression ratios. In [11], a Table-Driven XML approach (TDXML) is proposed offering a more compact message size, a simpler message structure and easier access to individual elements when compared to SOAP. The drawback of such methods in data-centric environments is that structured data is exchanged on a functional application level instead of triggering a data-level transfer method.

Pactas et al. [12] propose a non-intrusive approach for achieving data distribution among Web services that are engaged in compositions. Their extension consists of the integration of a protocol called Data-Flow Distribution Protocol for Web Services (DFDP-WS). This protocol encapsulates both control flow information and data in the messages it carries. In general, the DFDP-WS protocol is interposed between the application level protocols and the transport level protocols. The proposed protocol is a lightweight protocol and assures the infrastructure for component services to exchange data directly with each other. The focus is similar to us, but the data propagation is still done with an SOAP-like protocol approach, while we present an flexible approach to integrate various existing data propagation tools.

Business Process Execution Language

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. Primitive workflow functions are: receive, invoke, reply, throw, wait, and assign. The first three (receive, invoke, reply) are responsible for the interaction between the process and partner services. Exception handling can be done with the throw activity, while a scheduled execution is possible using the wait activity.

In addition to these primitive activities, a set of structured activities are available, which allow the creation of more complex workflow logic: sequence, flow, switch, and while. The structured activities determine how the basic activities are orchestrated. The flow construct contains activities that should be executed concurrently, while activities in the sequence construct are executed sequentially. In addition, the switch and while activities provide conditional logic.

From the above description, it is recognizable that BPEL offers various language constructs to express control flow patterns. The ever necessary data flow is defined implicitly by specifying variables that basically represent input and output data messages of activities. That means, the only way to cope with data flows in BPEL processes is with the variables concept. In this case, the assign activity is of special interest because assignments are used within BPEL to manipulate variables. A small example is illustrated in Figure 2. The output data message of Web service WS1 is stored in variable outputWS1. The subsequent assign activity manipulates this variable for the switch activity and the result is stored in variable inputCheck. The switch decision is then executed by the content of the inputCheck variable. As a conclusion, we are able to deduce that BPEL prefers control flow over data flow.

To execute BPEL processes, a corresponding execution engine, e.g. IBM’s WebSphere Business Integration Server Foundation, is required. Such a BPEL server controls the Web service invocations and coordinates the data message exchange between Web services. Therefore, BPEL follows the concept of a centralized control flow and a centralized data flow. The centralized data flow arises from the implicit data flow handling by variables because the BPEL server is the mediator for the data exchange for the participating Web services in a process. In the context of data-intensive processes such centralized data flow is not applicable with regard to performance and scalability. Using the BPEL server as main data broker point is not a good solution.
To increase the power of BPEL, a variety of extensions have been proposed. One well-known extension is BPELJ [14], which allows to include JAVA snippets (code) in BPEL definitions. Such JAVA snippets are expressions or small blocks of Java code that can be used for things such as (1) loop conditions, (2) branching conditions, (3) variable initialization and (4) Web service message preparation. The JAVA snippets are included in the process definition as normal activities, whereas the BPEL server executes the specified JAVA code.

Maier et al. [9] proposed a similar extension to BPEL, BPEL4SQL. BPEL4SQL supports SQL snippets as BPEL activities and in BPEL conditions. This approach can be seen as an embedded SQL approach for BPEL. However, BPEL4SQL does not support the full ETL technology.

**Data Propagation**

The problem of data access and data sharing across various Web services has also been examined in the Grid approach. In this special environment, the OGSA-DAI framework has been established to produce common middleware allowing uniform access to data resources using a service-based architecture [10]. The produced Grid Data Services allow consumers to discover the properties of structured data stores and to access their contents. Aside from the access to structured data, the OGSA-DAI framework offers the possibility of moving data between services with gridFTP [10]. However, FTP or gridFTP is just one possibility.

Moreover, 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, loading data from different data sources into a common data warehouse [15]. Such ETL processes consist of three parts: (1) extraction of data from the different source systems, (2) application of a series of rules and functions to the extracted data to derive the data to be loaded, and (3) 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 [7]. In database systems, this is used to provide redundancy or to balance the load across multiple database servers. Both methods propagate the data between two parties in a streaming mode and they are oriented to database system.

**3. Data-Grey-Box Web Services**

Fundamentally, Web services are a standard method for sharing (i) data as well as (ii) functionality among loosely coupled systems. In both cases, a database system, hidden from the service requestor, is normally in use. In the first case, the service providers manage their own data in a database system and with the Web service technology, the service providers offer well-defined views on the data for external requestors. In our application scenario, gene expression data sets are our data sources and therefore they are published over Web service interfaces. In the second case, a database system is used, since it cannot be taken for granted that the received data can be managed in the main memory unit during the whole processing time. Especially, for data-intensive functions like normalization or clustering a database is temporarily stored in a database system during the data processing.

Therefore, database systems play an important role in service-oriented environments. Based on this observation, we developed the concept of data-grey-box Web services [5]. In contrast to original black-box Web services, we enhanced the Web service interface with an explicit data aspect. Aside from the separation of parameters and data in the interface description, we introduced a new binding format for data. Through this new data binding, the Web service signals that the data are not to be transferred via SOAP but that there is a separate data layer instead. As before, parameters are handed over via SOAP when calling the Web service.

To handle our newly introduced data binding, we extended the SOAP framework by the integration of a novel data layer component. The entire invocation process of our data-grey-box Web services is illustrated in Figure 3. On the client side, an enhanced Web service call semantic is necessary. Aside from the transmission of the endpoint and regular parameters in the SOAP message, the client has to deliver access information as references or pointers for (i)
Figure 3. Data-Grey-Box Web Service Invocation Process

where the input data is available and (ii) where the output data should be stored. 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.

On the service side, our SOAP framework receives the SOAP message and conducts a separation into the functional aspect and the data aspect. The associated data layer can now call an appropriate mediator for the data propagation based on the access information of the client and the service. The data access information of the requestor can be found in the received SOAP message, while the data access information for the service must be queried from our extended service infrastructure [5]. Such a mediator may be a general ETL or data replication tool which is accessible over a standard Web service interface. At the moment our focus is oriented to database system and specialized data propagation tools for databases. The next research step is enhance this approach to be more general. We believe that this enhancement is possible through the possibility to integrated specialized propagation tools using standard Web service technology.

Advantages of this invocation principle are (i) that our data-grey-box Web services are still stateless, (ii) that the data propagation is transparent for both partners, and (iii) that new data exchange tools can be seamlessly integrated. As we show in Section 5, such data-specialized data propagation tools outperform the SOAP data transmission. A more detailed description of our data-grey-box Web services can be found in [5].

4. BPEL Data Transitions

The process definitions of our gene expression analysis processes are based on the standardized orchestration language BPEL. On top of Web services, BPEL has been established as the de-facto standard language to orchestrate business processes. In general, BPEL offers a standardized way to describe the functional composition of single services to create comprehensive processes. The data flows within BPEL processes are nowadays defined implicitly by specifying variables that basically represent input and output messages of services.

In order to allow an efficient composition of our data-grey-box Web services and an efficient process execution, we have developed a seamless extension to BPEL: 'BPEL data transitions (BPEL^{DT})'. These data transitions are an orthogonal data flow concept to the control flow; similar to the data aspect in BPMN [2]. Such data transitions are explicitly placed between two functional services and they wrap the process inherent to the data flow. One advantage is that Web services are now able to exchange data directly without involving the BPEL server. Furthermore, the data exchange is done with specialized data propagation tools.

An example process with two services and one data flow between them is shown in Figure 4. While Figure 4(a) shows the orchestration according to BPEL, Figure 4(b) de-
picts our BPEL<sup>DT</sup> approach. In this example, the output data of Web service $WS_1$ is used as input for Web service $WS_2$. $WS_1$ returns data included in a data source (gene expression data set); Web service $WS_2$ is responsible e.g. for the data normalization. Due to the very restrictive influence on Web services, extended selection and projection, for example, cannot be pushed down to services. In our example, not all data offered from service $WS_1$ should be processed by $WS_2$. Therefore, we have to filter out tuples which do not satisfy a user’s selection conditions. Furthermore, $WS_2$ cannot process the data in the actual schema, and a projection operation must be applied.

In the original BPEL approach (Figure 4(a)), two activities on the functional level have to be placed between services $WS_1$ and $WS_2$. The drawbacks of this solution are (1) that the functional order is enriched with data operations and (2) the inefficient execution with regard to the centralized data flows in BPEL. With our newly introduced data transition in BPEL, we separate functional activities from data activities, as illustrated in Figure 4(b). Furthermore, we are able to push down data operations directly in the data exchange procedure between Web services.

**Modelling Aspects**

From the modelling point of view, the previous BPEL approach follows a two-level programming model [13]. The first level (lower level) consists of Web Services as executable software components realizing the basic activities. The upper level is a process model defining the activation order of activities, where data and control flow are merged.

With our BPEL data transitions, we extend the programming model to a three-level model. The three levels are now:

- **Lower Level:** Web services as executable software components. These services represent the basic functional activities including a explicit data aspect (data-grey-box Web services).

- **Functional Orchestration Level:** In this level, a domain expert models the pure domain logic without considering data flow semantics. The main advantage is that the domain expert can focus on the process logic and the data flow is only modelled by data placeholders. The result of this level is a description of pure domain process without any infrastructural information. Such process descriptions are not essentially executable.

- **Data Flow Level (extension):** In the third level, a data management expert takes the process and annotates this process with all necessary data flows by BPEL data transitions. By this data flow annotation concept, the functional description of the BPEL process is not changed.

In this case, the modelling of the data flows can be done with corresponding suitable tools like ETL, because those tools are specialized for data flows. The result of this modelling step is a BPEL process with an explicit control as well as explicit data flows.

**Execution Aspects**

In the previous section, we presented the invocation principle of our data-grey-box Web service. In this invocation principle, data mediators are used for the data propagation. These data mediators are now extended to execute additional data transformation which are specified in our data transitions. In this case, we push down data operations as far as possible to the data sources represented by data-grey-box Web services.

In the presented invocation principle, the client has to deliver access information as references for (i) where the input data is available and (ii) where the output data should be stored. In our considered BPEL case, those access information must be delivered from the BPEL server during the invocation of data-grey-box Web services. Therefore, the BPEL server can now decide where necessary massive data sets should be temporarily stored during the process execution. Either the data is managed centralized by the BPEL server or decentralized by distributed databases which are registered at the BPEL server. In case of centralized data flows, the BPEL server must own a database system. For the other case, various distributed data sources have to be registered at the BPEL server. The advantage of decentralized data flows is that a load balancing across the distributed data sources can be conducted. This possibility offers some optimization strategies with regard to scalability and performance.

To summarize, with our data transition we have separated control and data flow activities in BPEL. The introduction of an explicit data flow concept with BPEL<sup>DT</sup> influences the execution of BPEL processes. The explicit data
flows are a new control instance for BPEL processes. The execution of a data flow is an essential precondition to invoke the data-consuming Web service. We call this property the PreInvoke condition. The BPEL server can only invoke a data-consuming service WS\(_2\) when the corresponding data flow from WS\(_1\) to WS\(_2\) is finished. To meet this PreInvoke condition, the BPEL server has to monitor the execution of the associated data flows.

Moreover, BPEL\(_{DT}\) extends the power of BPEL. The control flow is specified in the appropriate orchestration language BPEL, while the data flows are modeled with data-specialized approaches like ETL. Aside from the establishment of a decentralized data propagation in BPEL, our approach is able to execute necessary data operations in the propagation procedure. Our proposed data transitions are realized in our own BPEL server, called UseBPEL.

5. Evaluation

In this section, we present some evaluation result of our data-grey box Web services with the proposed invocation process regarding data transfer performance against the traditional SOAP approach. In this evaluation scenario, Web service WS\(_1\) extracts gene expression data from a local database system and Web services WS\(_2\) conducts a local clustering on the received data set. In the experiment, we transmitted 50,000 tuples with different numbers of columns, where each column included double values between WS\(_1\) and WS\(_2\) with SOAP (not other a BPEL server as broker). The measured transfer times include the time for (1) the data extraction from the database system, (2) the transfer of the SOAP message from the service to the requestor, and (3) the insertion of the received data into the database system of the requestor. Figure 5(a) depicts the resulting transmission times. As Figure 5(c) illustrates, we obtain a transmission speedup of up to 16 compared to the SOAP approach. More evaluation results can be found in [5].
ure 6. With our BPEL data transitions, the data propagation is done with ETL and over a distributed database system. That means, two ETL jobs were executed. The times are illustrated in Figure 6. As the experiment indicates, our approach optimize two SOAP transmissions and therefore the performance gain is twice of a single SOAP transmission.

6. Conclusion

In the context of genome research, the method of gene expression analysis 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. Having access to this variety of repositories, researchers would like to incorporate this data in their analyses processes to increase the statistical significance of their results. In this paper, we have given a short description of a gene expression analysis process as typical example of a data-intensive process. Fundamentally, data-intensive processes are characterized by (i) a sequence of functional operations and (ii) the transportation and transformation of huge data sets between the functional operations.

To support those data-intensive processes in general, an efficient and scalable environment is required. Our developed environment is based on the service-oriented architecture with an extension to be data-aware. In particular, our developed concept includes (i) data-grey-box Web services offering more information on the data persistence aspect within the service, (ii) the integration or triggering of data propagation tools in the Web service invocation process, and (iii) the composition and execution of data-grey-box Web services with BPEL—BPEL data transitions.

In order to efficiently use our developed data-grey-box Web services with BPEL, we developed a seamless extension to BPEL: the 'BPEL data transition.' In this way, BPEL data transitions are an orthogonal concept to the control flow. More detailed, with BPEL transitions we are able to (1) perform format conversion of individual data fragments, (2) perform decentralized data flows based on our data-grey-box Web services. As a result BPEL transitions provide the significant benefit of separating business logic from infrastructural aspects and from a solid foundation to efficiently support data-intensive processes based on Web service infrastructures. Our next research focus will explore the optimization approaches of BPEL<sup>DT</sup> in more detail.

References


