periodica polytechnica

Electrical Engineering
53/3-4 (2009) 123–138
doi: 10.3311/pp.ee.2009-3-4.05
web: http://www.pp.bme.hu/ee

RESEARCH ARTICLE

SOA based web service adaptation in enterprise application integration

Péter Martinek / *Béla* Szikora

Received 2009-05-03

Abstract

Enterprise Application Integration (EAI) is a permanent need since various information systems are employed at companies. Numerous standard systems must be aligned to new business processes. There are participant systems older than 10 years, and others developed only 1-2 years ago. This implicates a wide technological variance making the integration problem a real challenging issue. The widespread of the Service Oriented Architecture (SOA) seems to be one of the most promising approaches in EAI. Although this is already supported by solid technology and tools, deploying executable processes, predicting and optimizing their non-functional performance is still an open issue. In this paper we propose a technological solution for the adaptation of standard enterprise services into SOA integration scenarios providing support for applying data transformation to bridge data incompatibilities. To evaluate our approach three other possible solutions are designed and implemented. An in detailed analytic and experimental comparison of the approaches is also presented.

Keywords

SOA · run-time performance · web service adaptation · data transformation

Péter Martinek
Béla Szikora
Department of Electronics Technology, BME, H-1111 Budapest, Goldmann Gy. tér 3. building V2, Hungary

Introduction

Organizations today are hard to imagine without complex information systems. Software applications are applied for managing customer orders, timing the procurement and production, supervising shipping and billing, performing the financial booking, maintaining employee data etc. [9]. Adapting to the always changing (or growing) needs of customers, newer products and newer

services enforce the creation of new business processes and services. Unfortunately these do not conform to the boundaries of the different applications at the organization. Requested information and capabilities can come from many different information systems of the company or even from other organizations. This implicates the need for integrating our systems and realizing business processes over multiple enterprise applications. Based on the boundaries of integration we can talk about intra- and inter-organizational enterprise application integration.

The Service Oriented Architecture has appeared 4-5 years ago. It envisions an architecture where provided capability is offered by service providers and applied by service consumers. The services can be characterized by their short descriptions which are stored in service registries. Using the registry available services are easy to browse, search and find [11]. For example one can have a service offering booking at hotels. The characterizing description of this service should contain information about its capabilities e.g. the service is able to find an available hotel at a given place for a given time interval, and to perform or cancel booking. The complex capability of this service is divided into operations like *check availability*, *perform booking* and *cancelling*. The requested input and output data of each operation is also required in the description. Furthermore some technical details like the definition of service endpoints are needed in order to invoke the service. Having this description one is able to adopt the service and communicate with it properly.

SOA provides not only the possibility of simple one-to-one, request-answer communication. Whole business processes can be defined upon services. A combination that integrates the invocation of two or more services into a complex process is called

orchestration [23]. An orchestration makes it possible to create new, more complex services. For example, if there is a service capable of booking flights, combining it with the previous hotel booking service a new complex service can be created. This will be already able to organize the whole trip – both journey and accommodation. The complex orchestration process should describe the right order of operation invokes while paying attention to given constraints for example the flight dates and the hotel booking dates should be the same, or if there is no available flight for a given time period the hotel cannot be booked (must be cancelled).

Organizations have early recognized the applicability of SOA methodology and technologies for solving integration issues [9, 25, 34]. Orchestrated processes can be directly aligned to business processes – the goal of the orchestration is to create the requested new business functionality – and participating services are interfaces of enterprise application systems. However there are several difficulties and preconditions in designing and applying such an integration solution in a business environment.

Different systems and their services provide heterogeneous data semantics. The provided data can differ in simple naming conventions e.g. name of a person entity can be depicted as "guest" by the hotel booking service and "customer" by the flight booking service. It is obvious that both services cannot be invoked by the same entity naming convention. This means, that we can not choose a "standard" representation for this entity for the whole trip organizing service – instead of using the same entity (name of a customer) the request data must be different invoking the two services. There can also be other differences between provided data semantics e.g. differences in data structure or messaging protocol. These must be handled by several transformations, see [14, 15]. However applying transformations can slow down the designing issues and strongly influences the non-functional capabilities of the new complex services (processes).

In this paper we present a methodology and technical solutions for creating the necessary transformations and present an approach which conserves the applicability of solid and standard SOA technologies and tools while the run-time performance of the system remains still predictable. The focus of the paper is on the proposed run-time framework and on the prediction of non-functional capabilities.

The remainder of the paper is the following. In the next section we introduce related researches and work comparing them with our results. A SOA integration methodology and the creation of semantic aligned services and requested transformations are presented in Section 2. Section 3 compares our proposed run-time architecture with 3 other proposals. Moreover the performance of the different approaches is analyzed detailed. In Section 4 we introduce some experiments and results and finally section 5 draws conclusions and suggests future areas of work and study.

1 Related work

There are numerous researches in literature about predicting non-functional QoS parameters of composite services [2, 3, 6, 7, 16, 18, 38]. Some of them are based on the usage of intelligent agents to evaluate and increase the reliability and performance of orchestrated processes, for example see [16, 38]. Another current topic is the dynamic resource management in grid systems to optimize resource allocation in service oriented architectures [22, 35]. Unfortunately most of these approaches pass by the existence of interoperability problems.

The theoretical approach of the data heterogeneity problem is called schema matching in the literature. Unfortunately most of these works are rather academic. They propose improved analytical methods and algorithms to build common global schemas or to match different data structures without designing and implementing tools and technologies to realize it [5, 8, 13, 17].

On the other hand numerous researches are dealing with data heterogeneities during enterprise application integration. To overcome the problem the meaning of entities is examined and compared instead of the name of the concepts. These approaches add semantic description to operations and services based on specific semantic taxonomies. For example see [10, 12, 19, 33, 37].

The authors of [24] use a stochastic model to analyze the performance and reliability bottleneck of composite processes. A continuous-time Markov chain formulation including failure states is developed to compute both performance and bottlenecks. However this approach is based on the run-time monitoring of participating services and consumes a lot of resources for dynamic prediction of overall performance and modification of the orchestration at run-time. Because our integration approach focuses on the integration of enterprise applications, unreliable and instable services from the internet are not involved in processes and resource requirements of services and processes can be predicted directly upon documentation about system specification, features and development. Furthermore if a given function of a given enterprise application is required at a given point of the orchestrated process replacing it is simply out of the question since there is no other available service performing the same functionality in the same system.

Cardoso [7] states that high complex composite services tend to be less flexible and cause decreased performance compared to optimized and carefully designed processes. The paper analyzes the control flow complexity of processes and proposes several methods for simplification. Thus this approach can extend our work by the structural analysis of composite services and help us predict the resource consumptions of complex process constructions. On the other hand, data heterogeneities should be considered and resolved as proposed in this paper.

The vieDAME framework presented in [18] provides a complete solution for performance optimization and for overcoming interoperability problems. The robust architecture contains sev-

eral components for intercepting service invokes, for monitoring operation performance, for storing and executing transformations, for communicating with the run-time engine etc. However overall performance of a configuration can not be predicted and bottlenecks remain hidden from system operators.

2 SOA integration methodology and architecture

Vendors of enterprise applications are already prepared for SOA. Information systems offer their capabilities in the form of services. The service interfaces are mostly matched to worldwide standards which results in easy to use services in a technical sense.

Unfortunately this is not enough for creating real, working processes because services of different application systems offer their capabilities in heterogeneous data semantics [15]. This means that the output data provided by a participating service can not be fed directly into another service as an input. Data transformation must be applied within the chain of services to adjust input and output data of different services to each other.

There are many ways to adopt the required transformations into a composite process. The simplest solution is the following:

1 Create the process ignoring data heterogeneity problems. Participating services are composed by defining their relations e.g. invoke order, logical expressions for branches etc.

2 Search for incompatibilities caused by data heterogeneity. Design and apply transformation at every service invoke where entity mismatches appear.

Although this approach is easy to understand and can help with every kind of data heterogeneity, it offers a non-optimal solution. The unique transformations applied before service invokes are totally customized for a given environment. Hence they can not be adopted to another process or not even for another invoke of the same service within the same process. This implies that by updating the process or by creating a new one, the possibility of transformation re-using is excluded.

Instead of creating custom transformations by process orchestration applying a global data schema can ensure data compatibility on the process level. The global data schema can define every related entity of a given business domain regularizing the proper usage of given business concepts. For example in the global schema we can define the usage of term *guest* for the tourism domain excluding the usage of synonyms like client, customer, etc. in the domain. Thus applying regularizations defined by global schemas can prevent data mismatches on the process level. One application methodology for this scenario can be the following:

1 Define global schema for the domain – determine the structural and linguistic terms for every possible entity of a given domain.

2 Define transformations to every participating service. The transformation(s) are strictly connected to the service, which results in an encapsulated service behaving the same way as the original but communicates using the right concepts and terms of the global schema.

3 Design composite service (process) defining the relationships among participating services.

This solution already makes possible to re-use the defined transformations. The encapsulated services containing the necessary transformations are applied in new processes or in newer version of the same process. The next chapter presents the steps detailed above.

Note that the main advantage of this approach is that the transformations must be created to each participating service to each domain only once. The encapsulated service containing the transformations and the original service can be created on demand (right before we need it at a given point of the orchestration for the first time) or all participating services of given enterprise applications can be encapsulated before starting SOA based integration. If large enterprise software vendors agree on given global schemas the creation of encapsulated services can already be handled globally by them preventing any kind of data heterogeneity by service orchestration at companies. Moreover software vendors have the necessary experience and developer team to easily come over the task of transformation creation and fitting service schemas even to more widely used global schemas. The most applied global schemas in the field of online business are [21, 36].

2.1 Preparing for orchestration

The aim of this step is to align the service input and output to the global schema. In other words we would like to create a service that behaves in the same way as the original service but communicates with the concepts of the global schema towards the outside world. More precisely, the applied interfaces should conform to the global schema but the original service of the given application should be invoked in the background.

For example our flight reservation service requires the name of the client in a complex data structure containing two fields: *firstname* and *lastname*. In contrast to this the hotel booking service requires one field containing the full name (firstname and lastname separated with a space) called *guestname*. The process (composite service) communicates with the global schema concept and the corresponding global schema *tourism* standardizes the person entity in the separated *firstname*, *lastname* form. In the case of the flight service the process invoke data can simply be forwarded to the service call. However, invoking the hotel booking service requires a transformation to be glued to the service input which concatenates the firstname and lastname fields. After the transformation, the hotel booking service receives the required input in the right form and is able to serve the request.

Another direction of transformation may also be necessary. For example after a successful booking the flight reservation service responds with a confirmation message containing a field

called *smoking restrictions.* However our global schema contains the same entity called *smoking regulations*. Hence the given concept of the service output should be renamed to ensure later reusability and common understanding at the process level. To perform this, a transformation is glued to the output of the flight reservation service.

Upon these there are 2 types of transformations:

– *Down-cast* transformations transform the concepts of the global schema into the required form of a given service input before invoking the service and

– *Up-cast* transformations transform the concepts of the service output into the corresponding form of the global schema right after the service response.

There are numerous proposals to detect data incompatibilities, design and attach transformations in current researches [4, 19, 23, 25, 27]. Unfortunately most of them lack the tasks of composite service deployment and run-time architecture. In the next chapter we present our proposed architecture.

2.2 Executing encapsulated services

Designing and attaching transformation rules to services does not results that our composed service is ready to run after the orchestration. The complex service should be deployed into a run-time environment. The run-time engine is responsible for accepting service invokes, create a process instance and perform the tasks of the given instance. These tasks are invoking participating services, evaluating logical expressions and performing any business logic defined on the process level.

There are lots of solid process run-time environments available. However they are based on standards and are not prepared for executing transformations glued to participating services. To overcome this problem our proposed architecture ensures a standard service interface for the encapsulated services. This means, that the run-time engine can interact with the services in the standard way without knowing anything about its inner architecture and glued transformations. The standard service interface is provided by a special proxy service. The proxy service intercepts the service invoke, determines which service is willing to be invoked, relays the service request and response to the given service and performs required transformations on the service input and output. So the proxy service acts as a single service towards the outside world. It represents the original services of enterprise applications, behaves the same way as them but communicates with the concepts of the global schema towards the process instance (e.g. towards the process run-time engine). The strict matching of transformations and service inputs and outputs are managed also by the proxy architecture.

This approach also implies, that there is no need to modify the standard services and endpoints of enterprise applications: all data incompatibilities can be handled and prevented with aligning services to the global schema by designing and attaching up-

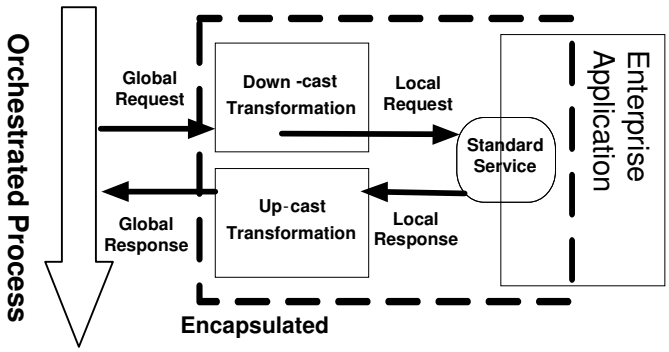


Fig. 1. Encapsulated service invoke

and down-cast transformations. Furthermore this architecture is able to hide real services from the business partners, which makes it possible to create an additional private layer in the collaborative business process. Fig. 1 shows one encapsulated service answering a service call.

The next chapter briefly presents some technical details of our solution.

2.3 Applied standards and technology

To be compatible with standard process run-time engines our solution follows current open standards.

The de facto standard to design executable business processes is the Business Process Execution Language (BPEL) [20]. Thus the focus was set to design processes in BPEL and apply solid BPEL run-time engines.

The main building elements of a SOA are usually web services communicating mostly with Simple Object Access Protocol (SOAP) [30]. Created services of enterprise applications are also equipped with web service interfaces. Thus our proxy implementation provides standard web service interfaces towards the process run-time engine and is prepared for invoking standard web services of enterprise information systems. The standard web service invocation is for sure an available and widely used option for interacting with collaborative business partners in BPEL processes.

Standard web services are described by the Web Service Description Language (WSDL) standard [31]. This defines one or more operations to the service containing requested input and output data structure per each operation. Hence our transformations must be connected to the operations. The number of attached transformations is equal to the number of operations multiplied by 2 at maximum (1 up-cast transformation to the output and 1 down-cast transformation to the input can be defined to each operation). Every data incompatibility between the global schema and a given service input or output is covered with one complex transformation. In other words, simple transformation rules eliminating one given data mismatch of one given data concept are collected by one complex transformation.

To align to widely used standards the XSL Transformation standard was selected to realize transformations [32]. The ex-

pression power of the XSL transformation language is NP full, which implicates that it is able to cover every kind of incompatibilities. Furthermore there are lots of solid XSLT designer applications and run-time engines available which can be applied in our architecture.

The creation of the proxy service interface to each service makes it possible to extend the WSDL description of standard enterprise application services with additional semantic information. This makes service discovery and selection easier which can be a great help for the service orchestration. See [21, 36] for further details.

It is obvious that our solution provides some additional functionality by solving data incompatibility problems. But what cost should we pay for that? Apart from the additional design cost of transformations, proxy services must be created and transformations must be performed during the process run-time. Because the creation of proxy services is done in design time it does not really influences the performance of the running system. But what additional cost do we have for handling web service invokes, relaying them to and from the web services of enterprise applications and executing transformations? The next chapter provides some analysis about the prediction of run-time overhead of our architecture. Moreover some configuration and installation issues are also presented.

2.4 Performance prediction and configuration analysis

Current researches often deal with QoS parameters of complex services. There are two types of QoS parameters:

- Functional parameters describe service capabilities. In other words, they present what and how the service does. For example for our flight reservation service these parameters contain the fact that there are 3 operations *search for flight*, *book flight*, *cancel booking* and the search for flight operation requests *departure*, *destination*, and *date of journey* input parameters and provides a confirmation as output containing the *number of reservation*, *number of flight*, *departure time, arrival time,* etc.
- Non-functional parameters describe all kinds of non-technical information about the service. These are availability, average response time, error-rate etc.

In this paper the focus is set to the prediction and optimization of non-functional parameters. Description of the functional parameters of our encapsulated services can be found in [15].

There are working methods and algorithms to predict system parameters – like response time and throughput – of composite processes [16, 38]. Most of them are based on the idea of dividing the process into components whose performances can be directly evaluated and calculating the parameters of the whole system based on the defined connections (e.g. sequences, branches with logical conditions of predictable likelihoods, parallel runtime threads, etc.) among these components. Because the basic building blocks of processes are services they are the smallest undividable parts which actually determine the system performance. Thus we focus on performance analysis of our encapsulated service in the paper.

The questions to be answered are the followings: How does the usage of transformations influence the response time of composite processes? What is the maximum throughput of our architecture compared to other approaches? Is this the optimal solution for handling interoperability problems at process composition from the point of view of overall performance?

Response time and throughput are determined by resources of system hardware components. More precisely services and processes require a given amount of resources and performance depends on the ratio of resource requirements to available resources. Extension of available hardware resources can result in faster response times and higher throughput while application of non-optimal solutions in process compositions may lead to significant decreasing of these parameters. Thus prediction of expected resource consumption is critical when designing composite processes.

There are numerous resources needed by a process at runtime. Central processing unit (CPU), memory, input/output operations (I/O) and network bandwidth are some of them. Nevertheless there is always one resource which is critical in a given environment determining the actual values of response time and throughput. This is called the bottleneck. Strengthening or increasing of bottleneck resource yields better performance while expansion of other resources may be ineffective. After consuming all of the bottleneck resource the response time of the system is decreasing by serving further requests or this can even lead to malfunctions like refusing of requests or other errors sent in the response. This means that an optimal efficiency (minimal response time) can be achieved only if the load of the bottleneck resource is lower than the available level of the resource. On the other hand system throughput reaches its maximal value (maximal number of performed transactions per a given time unit) if the consumption of the bottleneck resource is maximized. Upon these in the rest of the paper performance is analyzed through these two different (conflicting) performance parameters considering the actual usage rate of the bottleneck.

The next chapter analyzes our proposal comparing its efficiency with other possible approaches.

3 Analysis and comparison of our approach

Our approach is based on the encapsulation of the standard services of enterprise applications. One upcast and onedowncast transformation may be glued to every operation of every service. In the rest of the paper we assume that there are always both up- and downcast transformation required to be attached. Because this is the worst case from the point of view of added overhead in resource requirement, the efficiency of our solution will surely not be overestimated.

The resource requirement of a given enterprise application

service (operation) is determined by the application itself. Since it can not be changed during the process composition and it will determine the future performance of the composite process, its resource requirement and non-functional parameters in the given environment can serve as reference values for our tests.

Consider that a given operation (e.g. the *book flight* operation of our service) requires R_{service} unit of the bottleneck resource. The optimal (minimum) response time of the system for that is $T_{service}$ unit of time. There is a certain amount of available bottleneck resource denoted by R_{avail}.

Service request may come from more clients (or from more process instances) at the same time. Thus total resource requirement comes to $\sum_{n\_conc} R_{\text{service}}$ where *n_conc* is the number of concurrent service requests. The *response time* can be calculated as follows:

$$T_{\text{response}} = T_{\text{service}} \text{ if } \sum_{\text{n\_conc}} R_{\text{service}} \leq R_{\text{avail}} \text{ and}$$
$$T_{\text{response}} = T_{\text{service}} \cdot \frac{\sum_{\text{n\_conc}} R_{\text{service}}}{R_{\text{avail}}} \text{ if } \sum_{\text{n\_conc}} R_{\text{service}} > R_{\text{avail}}. \quad (1)$$

After the resources and response times are determined for all participating operations the performance of the whole process can be predicted. The requested resource of the process (R_{process}) consists of two separable components:

– the calculated cost of the orchestrated service (R_{orchest}) built of encapsulated services which depends on the given orchestration e.g. structure and business logic of the process [16], [38] and

– the constant cost of process management ($R_{\text{constructs}}$), which includes process instance, variable creation and process maintenance.

Our experiments show that at reasonable process complexity and available resources (<50 participating service, 5-10 branches and logical expressions, etc.) the calculated orchestrated service cost (R_{orchest}) is almost equal to the cost of invoked participating services. Thus the overhead of applying a process is almost equal to the cost of process management ($R_{\text{constructs}}$). Moreover $R_{\text{constructs}}$ seems to be not dependent on the given instance of processes and remains to be a constant value in a given hardware environment even for different type of processes.

Eq. (2) shows the calculation of the amount of requested resource of the process.

$$R_{\text{process}} = R_{\text{constructs}} + R_{\text{orchest}} = R_{\text{constructs}} + \sum_{\text{m\_conc}} R_{\text{service}} \quad (2)$$

where m is the number of service invokes during the execution of a process.

By substituting Eq. (2) into Eq. (1) the response time of a process instance can be calculated as follows:

$$T_{\text{response}} = \sum_{\text{path}} T_{\text{service}} + T_{\text{constructs}} \text{ if } \sum_{\text{m\_conc}} R_{\text{process}} \leq R_{\text{avail}}$$

and

$$T_{\text{response}} = \left(\sum_{\text{path}} T_{\text{service}} + T_{\text{constructs}} \right) \cdot \frac{\sum_{\text{m\_conc}} R_{\text{process}}}{R_{\text{avail}}} \text{if} \quad (3)$$
$$\sum_{\text{m\_conc}} R_{\text{process}} > R_{\text{avail}}$$

where the response times of the services participating in the given execution path of the process must be summarized $\left(\sum_{\text{path}} T_{\text{service}} \right)$. Because $R_{\text{constructs}}$ is constant, the response time overhead $T_{\text{constructs}}$ caused by it is a constant value as well.

In our interpretation the *throughput* is the number of served operation invoked in a given time unit. This can be an invoked operation of a simple or a complex service (Tp_{service},Tp_{process} respectively). After having used up the entire bottleneck resource (in other words the bottleneck is fully loaded) the throughput reaches its maximal value, denoted by $Tp_{\text{service\_max}}$. So the maximal throughput for a given service invoke describes the pure efficiency of a given hardware-software environment. Such hardware-software environments consisting of several computers connected into a network are called *configurations* in the rest of the paper.

The throughput of a configuration for a given service (process) can be predicted as follows:

$$Tp_{\text{service}} = Tp_{\text{service\_max}} \cdot \frac{\sum_{\text{n\_conc}} R_{\text{service}}}{R_{\text{avail}}} \text{ if}$$
$$\sum_{\text{n\_conc}} R_{\text{service}} \leq R_{\text{avail}},$$
$$Tp_{\text{service}} = Tp_{\text{service\_max}} \text{ if}$$
$$\sum_{\text{n\_conc}} R_{\text{service}} > R_{\text{avail}} \text{ and}$$
$$Tp_{\text{process}} = Tp_{\text{process\_max}} \cdot \frac{\sum_{\text{m\_conc}} R_{\text{process}}}{R_{\text{avail}}} \text{ if}$$
$$\sum_{\text{m\_conc}} R_{\text{process}} \leq R_{\text{avail}},$$
$$Tp_{\text{process}} = Tp_{\text{process\_max}} \text{ if}$$
$$\sum_{\text{m\_conc}} R_{\text{process}} > R_{\text{avail}} \quad (4)$$

Furthermore there is a trivial connection between the maximal throughput and response time at a status where increasing the exploitation of the bottleneck resource has just reached the 100% limit. If we also assume that the concurrent requests all came from the same type of service (or process) the maximal throughput can be predicted as follows:

$$Tp_{\text{service\_max}} = \frac{n}{T_{\text{response}}}, \text{where} \sum_{\text{n\_conc}} R_{\text{service}} = R_{\text{avail}}$$

and

$$Tp_{\text{process\_max}} = \frac{m}{T_{\text{response}}}, \text{where} \sum_{\text{m\_conc}} R_{\text{process}} = R_{\text{avail}} \quad (5)$$

Now we have the necessary formalism to analyze the performance of complete configurations of given approaches.

3.1 Evaluation of proposals

Four different approaches are compared on the same configuration in the paper. Because the configurations are the same, the efficiency of the approaches can be directly compared. Furthermore other parameters of these proposals (like component re-usability, modularity, transparency etc.) are also evaluated. The following sub-chapters contain the detailed analysis of each proposal.

3.1.1 Proposal XSLT

Our already presented proposal is evaluated first. To implement our approach on the same configuration (see later) the transformations are stored and executed on the same computer (or grid of computers) as the enterprise services are executed on. The proxy service is also installed here, so invocation of an encapsulated service can be served without adding new resources (computers) to the existing configuration of the enterprise application system. This proposal is called as *proposal_XSLT* (because of the applied XSL transformations) in the rest of the paper. The configuration and typical process execution can be viewed in Fig. 2.

The two transformations (up- and downcast) require R_{transf} resource from the bottleneck resource (the worst case scenario is assumed again) and the work of the standard web service interface requires R_{ws_interf}. So the additional resource consumption of an encapsulated service is:

$$R_{\text{add}} = R_{ws\_interf} + R_{\text{transf}} \quad (6)$$

Assuming the worst case scenario, the bottleneck resource of these operations is the same as the bottleneck of the original operation. In this case the invocation of the encapsulated service costs:

$$R_{\text{encaps}} = R_{\text{service}} + R_{\text{add}} \quad (7)$$

By substituting Eq. (7) into Eq. (1) the response time can be calculated as follows:

$$T_{\text{response}} = T_{\text{service}} + T_{\text{add}} \quad \text{if} \sum_{\text{n\_conc}} R_{\text{encaps}} \leq R_{\text{avail}} \text{ and}$$

$$T_{\text{response}} = (T_{\text{service}} + T_{\text{add}}) \cdot \frac{\sum_{\text{n\_conc}} R_{\text{encaps}}}{R_{\text{avail}}} \quad \text{if}$$

$$\sum_{\text{n\_conc}} R_{\text{encaps}} > R_{\text{avail}} \quad (8)$$

where T_{add} is the additional response time compared to the stand-alone enterprise service.

Accordingly to Eq. (5) the maximum throughput can be calculated as follows:

$$Tp_{\text{encaps\_max}} = \frac{n}{T_{\text{response}}}, \text{ where} \sum_{\text{n\_conc}} R_{\text{encaps}} = R_{\text{avail}} \quad (9)$$

During an orchestration our encapsulated services are built in into a complex service (process). However the process runtime engine is placed separated from the enterprise application server(s). This determines the distribution of process runtime costs i.e. resource requirements of process management ($R_{\text{constructs}}$) and orchestrated process (R_{orchest}) between enterprise application and process management server. As already mentioned R_{orchest} is almost equal to the resource requirement of invoked participating services in a certain environment. Thus R_{orchest} is mainly processed by the application server(s). On the other hand process management requirements ($R_{\text{constructs}}$) are served by the process management server(s). The two run-time components are connected with network which is responsible for the communication during the execution of the composite process. While the network connection may influence the response time, it barely modifies the throughput of the system because it is unlikely to be the bottleneck resource in any systems applied for process execution today. Hence dedicating separated resources for enterprise services and process run-time is reasonable and it results greater performance than applying the enterprise service environment also for process management.

Eqs. (6)-(9) described the resource requirements, expected response times and throughput at the enterprise application server(s). Eq. (2) must be separated due to the separated resource allocation for enterprise services and process management. Furthermore Eq. 4 should also be modified because of the two possible bottleneck points i.e. the bottleneck of the whole system can be either the enterprise or the process server. Based on Eqs. (2) and (4) the expected response time of the configuration loaded by a given process is the following:

$$T_{\text{response}} = \sum_{\text{path}} T_{\text{service}} + \sum_{\text{path}} T_{\text{add}} + T_{\text{constructs}}$$

$$\text{if} \sum_{\text{n\_conc}} R_{\text{orchest}} \leq R_{\text{avail\_enterprise}} \quad \text{and}$$

$$\sum_{\text{n\_conc}} R_{\text{constructs}} \leq R_{\text{avail\_process}},$$

$$T_{\text{response}} = \sum_{\text{path}} T_{\text{service}} + \sum_{\text{path}} T_{\text{add}} + T_{\text{constructs}} \cdot \frac{\sum_{\text{n\_conc}} R_{\text{constructs}}}{R_{\text{avail\_process}}}$$

$$\text{if} \sum_{\text{n\_conc}} R_{\text{orchest}} \leq R_{\text{avail\_enterprise}} \quad \text{and}$$

$$\sum_{\text{n\_conc}} R_{\text{constructs}} > R_{\text{avail\_process}},$$

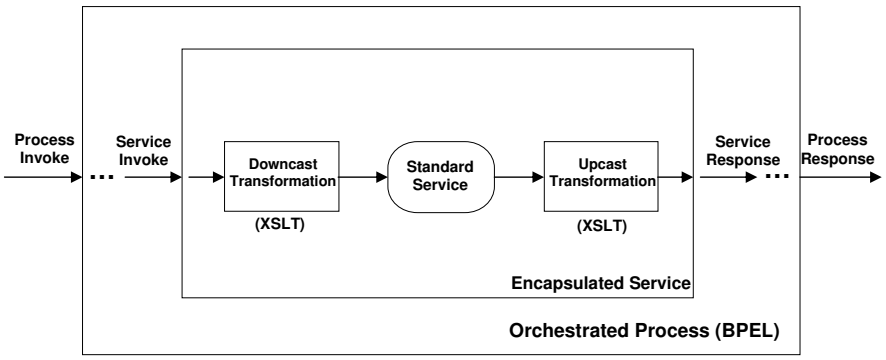


Fig. 2. Execution of a process request at proposal XSLT

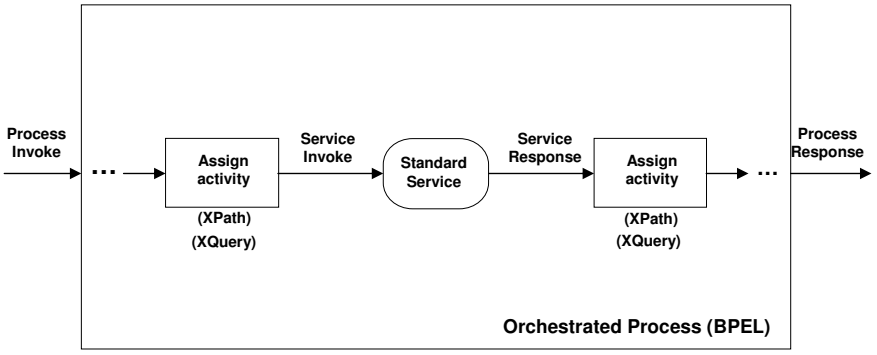


Fig. 3. Execution of a process request at proposal Direct

$$T_{\text{response}} = \left(\sum_{\text{path}} T_{\text{service}} + \sum_{\text{path}} T_{\text{add}} \right) \cdot \frac{\sum_{\text{n\_conc}} R_{\text{orchest}}}{R_{\text{avail\_enterprise}}} + T_{\text{constructs}}$$

$$\text{if} \sum_{\text{n\_conc}} R_{\text{orchest}} > R_{\text{avail\_enterprise}} \quad \text{and}$$

$$\sum_{\text{n\_conc}} R_{\text{constructs}} \leq R_{\text{avail\_process}},$$

$$T_{\text{response}} = \left(\sum_{\text{path}} T_{\text{service}} + \sum_{\text{path}} T_{\text{add}} \right) \cdot \frac{\sum_{\text{n\_conc}} R_{\text{orchest}}}{R_{\text{avail\_enterprise}}}$$

$$+ T_{\text{constructs}} \cdot \frac{\sum_{\text{n\_conc}} R_{\text{constructs}}}{R_{\text{avail\_process}}}$$

$$\text{if} \sum_{\text{n\_conc}} R_{\text{orchest}} > R_{\text{avail\_enterprise}} \quad \text{and}$$

$$\sum_{\text{n\_conc}} R_{\text{constructs}} > R_{\text{avail\_process}}. \tag{10}$$

The proper calculation of actual and maximal throughput (based on Eqs. (4) and (5)) are left for the reader.

Summarizing this proposal loads the process server by $R_{\text{constructs}}$ at each process invoke. Moreover the enterprise server is loaded with an extra R_{add} by each participating service in a process execution path by each process invoke. Note that based on our experiments this additional cost is negligible compared to the cost of the original enterprise services. Thus one may focus on calculating the resource requirements $R_{\text{constructs}}$ and predicted performance of the process server.

Remember that the encapsulated services are re-useable in subsequent process orchestration scenarios. Furthermore applying encapsulated services modularity can be achieved and it hides possible business interests within the non-transparent encapsulated service.

3.1.2 Proposal Direct

Another proposal for dealing with data heterogeneity in integration consists of the following steps:

1 Create the composite process ignoring data heterogeneities

2 Design and add necessary transformation at the process level.

As already mentioned besides defining relationships among participating services and setting run-time branches by logical equations it is possible to define some business logic at the process level. (For example the BPEL standard offers the usage of XPath and XQuery standards for defining copy rules in data assignments [20, 28, 29].) Thus both up- and downcast transformations can be designed and added at the process level. One upcast transformation is added before each operation (service) request and one downcast transformation is added after each operation response in worst case. This proposal is called as *proposal_Direct* (because data heterogeneities are bridged directly at the process level during the orchestration) in the rest of the paper. The configuration and typical process execution can be viewed in Fig. 3.

The resource consumption is increased by the needs of transformation at the process server but requested resource remains intact at the enterprise server. The extra cost added to each op-

eration (service) invoke is:

$$R_{\text{add\_Direct}} = R_{\text{transf}} \tag{11}$$

Our experiences concluded that pure transformation costs are almost the same at the process level as they are in the case of our XSLT execution within the encapsulated services. However with this proposal the creation and employment of an additional service interface (the service interface of the proxy service) is avoidable. Comparing Eqs. (11) and (7):

$$R_{\text{add\_XSLT}} = R_{\text{add\_Direct}} \quad \text{and}$$
$$R_{\text{add\_XSLT}} = R_{\text{add\_Direct}} + R_{ws\_\text{interface}} \tag{12}$$

Furthermore $R_{\text{add\_XSLT}}$ loads the process server and not the enterprise application server. (Note that this may be an advantage or disadvantage of this proposal depending on the appearance of the bottleneck on the process server or the enterprise server.) By using this and Eqs. (10) and (11) the response time can be calculated as follows:

$$\begin{aligned}
&T_{\text{response}} = \sum_{\text{path}} T_{\text{service}} + \sum_{\text{path}} T_{\text{add}} + T_{\text{constructs}} \\
&\text{if } \sum_{\text{n_conc}} R_{\text{orchest}} \leq R_{\text{avail_enterprise}} \quad \text{and} \\
&\sum_{\text{n_conc}} R_{\text{constructs}} \leq R_{\text{avail_process}} \\
&T_{\text{response}} = \sum_{\text{path}} T_{\text{service}} + \left(\sum_{\text{path}} T_{\text{add}} + T_{\text{constructs}} \right) \cdot \\
&\frac{\sum_{\text{n_conc}} R_{\text{constructs}}}{R_{\text{avail_process}}} \\
&\text{if } \sum_{\text{n_conc}} R_{\text{orchest}} \leq R_{\text{avail_enterprise}} \text{ and} \\
&\sum_{\text{n_conc}} R_{\text{constructs}} > R_{\text{avail_process}}, \\
&T_{\text{response}} = \sum_{\text{path}} T_{\text{service}} \cdot \frac{\sum_{\text{n_conc}} R_{\text{orchest}}}{R_{\text{avail_enterprise}}} + \sum_{\text{path}} T_{\text{add}} + T_{\text{constructs}} \\
&\text{if } \sum_{\text{n_conc}} R_{\text{orchest}} > R_{\text{avail_enterprise}} \text{ and} \\
&\sum_{\text{n_conc}} R_{\text{constructs}} \leq R_{\text{avail_process}}, \\
&T_{\text{response}} = \sum_{\text{path}} T_{\text{service}} \cdot \frac{\sum_{\text{n_conc}} R_{\text{orchest}}}{R_{\text{avail_enterprise}}} + \\
&\left(T_{\text{constructs}} + \sum_{\text{path}} T_{\text{add}} \right) \frac{\sum_{\text{n_conc}} R_{\text{constructs}}}{R_{\text{avail_process}}} \\
&\text{if } \sum_{\text{n_conc}} R_{\text{orchest}} > R_{\text{avail_enterprise}} \quad \text{and} \\
&\sum_{\text{n_conc}} R_{\text{constructs}} > R_{\text{avail_process}}.
\end{aligned} \tag{13}$$

where instead of $R_{\text{orchest}} R_{\text{constructs}}$ contains the additional resource requirement of transformations compared to proposal_XSLT.

To sum up, some increase in the performance (lowering the response time) is possible compared to the proposal_XSLT in given cases. However additional cost of maintaining a service interface are not significant compared to the service (R_{service}) and process running ($R_{\text{constructs}}$) costs, so possible performance increase is not significant.

This little improvement probably is not worth the price we have paid for it. Because our transformations are defined at the process level they are process dependent. Hence probably they can not be reused in other process compositions. Furthermore by adding data transformations to the business logic we have messed up the clear structure and view of the orchestration. (The transformations were added to given copy rules of assign elements in the BPEL process where also the structural assignments of services – connection of global BPEL and local service variables – were defined. These can not be separated and differentiated easily later by a process upgrade or modification.) So the modularity of the orchestration was almost ruined and the possibility of creating an additional private business layer upon the enterprise service was largely reduced.

3.1.3 Proposal BPEL

The next presented proposal also defines the necessary transformations at the process level, but it eliminates upcoming low modularity and non-reusability problems of proposal_Direct. The methodology of creating working composite processes is the following:

- Define a collection of possible concepts for the given domain (global schema).
- Adopt each participating services to the process level each of them into a new process.
- Design and implement transformations in these processes to align complex service (process) interface to the global schema.
- Orchestrate business process using complex services created in the previous step as building blocks.

Similarly to proposal_Direct the transformations are defined at the process level (Copy rules are defined in assign elements in the BPEL process). On the other hand complex services containing only one operation (service) invoke and glued up- and downcast transformations are equivalent with the encapsulated services of proposal_XSLT.

This proposal is called *proposal_BPEL* (because additional BPEL processes are also applied to bridge data heterogeneities at each service invoke) in the rest of the paper. The configuration and typical process execution can be viewed in Fig. 4.

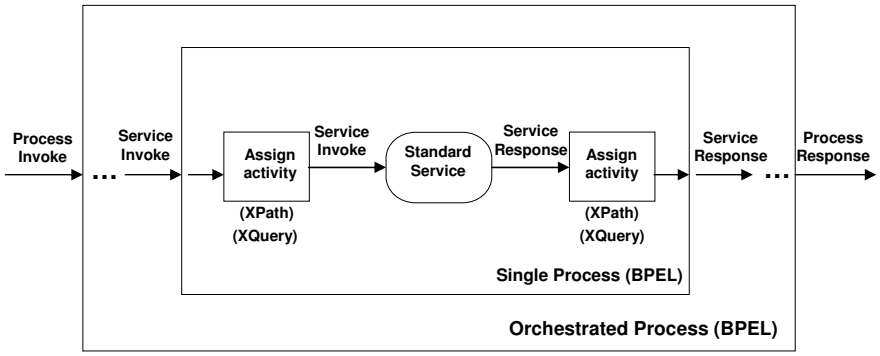


Fig. 4. Execution of a process request at proposal BPEL

The additional resource requirement is significant. Besides the additional cost of transformations (R_{transf}) there is a process management overhead ($R_{\text{constructs}}$ and R_{orchest}) at every participating service invoke. Although the orchestration costs are negligible (each "encapsulated" servicer contains only 2 assign elements and 1 service invoke) the creation of process instance and managing process interface adds significant overhead ($R_{\text{constructs}}$to each participating operation) in the run-time.

Similarly to proposal_Direct additional performance requirements are loaded to the process server. Based on Eq. (13) the response time can be predicted as follows:

$$T_{\text{response}} = \sum_{\text{path}} T_{\text{service}} + \sum_{\text{path}} T_{\text{add}} + T_{\text{constructs}}$$

$$\text{if} \sum_{\text{n\_conc}} R_{\text{orchest}} \leq R_{\text{avail\_enterprise}}$$

$$\text{and} \sum_{\text{n\_conc}} R_{\text{constructs}} \leq R_{\text{avail\_process}},$$

$$T_{\text{response}} =$$

$$\sum_{\text{path}} T_{\text{service}} + \left(\sum_{\text{path}} T_{\text{add}} + T_{\text{constructs}} \right) \cdot \frac{\sum_{\text{n\_conc}} R_{\text{constructs}}}{R_{\text{avail\_process}}}$$

$$\text{if} \sum_{\text{n\_conc}} R_{\text{orchest}} \leq R_{\text{avail\_enterprise}} \text{ and}$$

$$\sum_{\text{n\_conc}} R_{\text{constructs}} > R_{\text{avail\_process}}$$

$$T_{\text{response}} =$$

$$\sum_{\text{path}} T_{\text{service}} \cdot \frac{\sum_{\text{n\_conc}} R_{\text{orchest}}}{R_{\text{avail\_enterprise}}} + \sum_{\text{path}} T_{\text{add}} + T_{\text{constructs}}$$

$$\text{if} \sum_{\text{n\_conc}} R_{\text{orchest}} > R_{\text{avail\_enterprise}} \text{ and}$$

$$\sum_{\text{n\_conc}} R_{\text{constructs}} \leq R_{\text{avail\_process}},$$

$$T_{\text{response}} = \sum_{\text{path}} T_{\text{service}} \frac{\sum_{\text{n\_conc}} R_{\text{orchest}}}{R_{\text{avail\_enterprise}}} + \left(T_{\text{constructs}} + \sum_{\text{path}} T_{\text{add}} \right) \cdot \frac{\sum_{\text{n\_conc}} R_{\text{constructs}}}{R_{\text{avail\_process}}}$$

$$\text{if} \sum_{\text{n\_conc}} R_{\text{orchest}} > R_{\text{avail\_enterprise}}$$

$$\text{and} \sum_{\text{n\_conc}} R_{\text{constructs}} > R_{\text{avail\_process}} \tag{14}$$

where $\sum_{\text{path}} T_{\text{add}}$ contains also $T_{\text{constructs}}$ which is the cost of each built in complex service invoke.

The extra cost added to each operation (service) invoke is:

$$R_{\text{add\_BPEL\_service}} = R_{\text{transf}} + R_{\text{constructs}} \tag{15}$$

which is significantly greater than the extra costs of proposal_XSLT and proposal_Direct (based on our experiences $R_{\text{constructs}} \gg R_{\text{ws\_interface}}$). Thus performance of this proposal is expected to be lower (higher response times, and lower throughput) than the previous proposals.

On the other hand applying transformations at the process level has advantages too. The solid tools for BPEL process design and run-time can be reused and no further equipment is required to create and run the transformations. Furthermore the same process run-time environment can be applied for deploying and running both stand-alone "encapsulated" processes and orchestrated composite services.

Similarly to proposal_XSLT the "encapsulated" processes (services) are reusable in later orchestration scenarios. Moreover the proposal involves modularity and makes it possible to create an additional private layer beyond the enterprise services at the process level of stand-alone processes.

The reader may also note, that extending our configuration with an additional process server environment the "encapsulated" processes can be separated from the orchestrated processes which probably results significant improvement in the throughput at higher loads.

3.1.4 Proposal Native

The realization of the last presented proposal consists of the steps as follows:

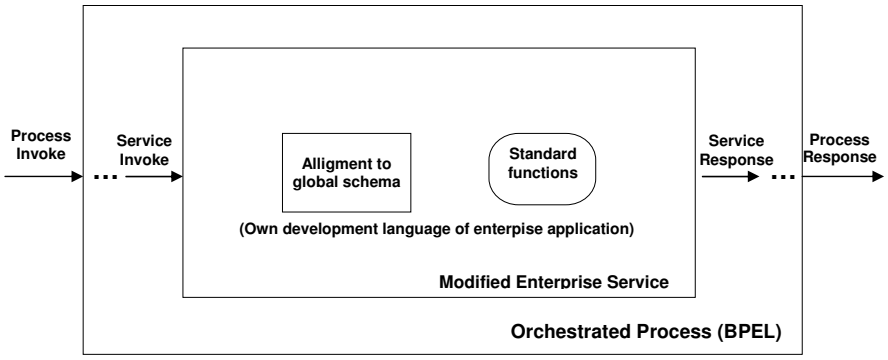


Fig. 5. Execution of a process request at proposal Native

- Define a collection of possible concepts for the given domain (global schema),
- Modify the enterprise application service and/or its interface directly at the application level to assign participating operations to the global schema,
- Orchestrate business process using the revised services of enterprise applications created in the previous step as building blocks.

To modify an enterprise application the special development environment of the application is required. This is mostly unavailable at the customer of the system. Furthermore employees of the organization are usually not allowed (or able) to modify standard functions of enterprise applications they use. Thus alignment of services and their interfaces to a global schema remains to the software vendor. Because integration (the creation of collaborative business process) can concern more application systems at more companies with various vendors, the process orchestration requires the contribution of numerous developers and can slow down the whole integration progress. Contrary to this, the creation of transformations and processes can be performed by integration experts responsible directly for process orchestration.

This proposal is called as *proposal_Native* (because bridging of data heterogeneities is solved at the native, enterprise application implementation level) in the rest of the paper. The configuration and typical process execution can be viewed in Fig. 5.

After all participating services are aligned to the global schema the performance of this proposal is convincing. The additional resource requirements of the modified enterprise service are not significant and no more additional resource consumption is required nor by the preparation nor by the orchestration at all. The enterprise server is loaded by the resource costs of invoked operations and the process server is loaded by the costs of the management of orchestrated composite service ($R_{\text{constructs}}$). Due to the additional cost of proposals the following relations hold:

$$R_{\text{add\_Native\_service}} < R_{\text{add\_Direct\_service}} < R_{\text{add\_XSLT\_service}} < R_{\text{add\_BPEL\_service}} \tag{16}$$

$$R_{\text{add\_Native\_service}} \approx R_{\text{add\_Direct\_service}} - R_{\text{transf}} \approx R_{\text{add\_XSLT\_service}} - R_{\text{transf}} - R_{\text{ws\_interface}} \approx R_{\text{add\_BPEL\_service}} - R_{\text{transf}} - R_{\text{constructs}} \tag{17}$$

The response time of the configuration can be predicted by the following equation:

$$T_{\text{response}} = \sum_{\text{path}} T_{\text{service}} + T_{\text{constructs}}$$

if $\sum_{\text{n\_conc}} R_{\text{orchest}} \leq R_{\text{avail\_enterprise}}$ and $\sum_{\text{n\_conc}} R_{\text{constructs}} \leq R_{\text{avail\_process}}$,

$$T_{\text{response}} = \sum_{\text{path}} T_{\text{service}} + T_{\text{constructs}} \cdot \frac{\sum_{\text{n\_conc}} R_{\text{constructs}}}{R_{\text{avail\_process}}}$$

if $\sum_{\text{n\_conc}} R_{\text{orchest}} \leq R_{\text{avail\_enterprise}}$ and $\sum_{\text{n\_conc}} R_{\text{constructs}} > R_{\text{avail\_process}}$,

$$T_{\text{response}} = \sum_{\text{path}} T_{\text{service}} \cdot \frac{\sum_{\text{n\_conc}} R_{\text{orchest}}}{R_{\text{avail\_enterprise}}} + T_{\text{constructs}}$$

if $\sum_{\text{n\_conc}} R_{\text{orchest}} > R_{\text{avail\_enterprise}}$ and $\sum_{c\_conc} R_{\text{constructs}} \leq R_{\text{avail\_process}}$,

$$T_{\text{response}} = \sum_{\text{path}} T_{\text{service}} \cdot \frac{\sum_{\text{n\_conc}} R_{\text{orchest}}}{R_{\text{avail\_enterprise}}} + T_{\text{constructs}} \cdot \frac{\sum_{\text{n\_conc}} R_{\text{constructs}}}{R_{\text{avail\_process}}}$$

if $\sum_{\text{n\_conc}} R_{\text{orchest}} > R_{\text{avail\_enterprise}}$ and

$$\sum_{\text{n\_conc}} R_{\text{constructs}} > R_{\text{avail\_process}}. \tag{18}$$

where $R_{\text{constructs}}$ is the process management cost of the orchestrated service and R_{orchest} is the run-time cost of invoked participating enterprise services (operations).

Based on resource requirements this proposal may yield an optimal (maximal) throughput for a given composite service (process). On the other hand it is hard to be carried out because upcoming modification requests (alignment of service to the global schema) can only be treated at the enterprise application development level.

Concerning modularity and transparency this proposal does not differ from the orchestration of standard enterprise services. Updated services remain black-boxes, and are reusable in subsequent orchestration scenarios.

Table 1 summarizes the main aspects (advantages and disadvantages) of the 4 presented proposals.

The next chapter presents our experimental results. All proposals were implemented for the same integration scenario (same data heterogeneity problem) on the same configuration. The results have confirmed the analytic methods and predicted values described in this section.

4 Experiments

The applied configuration contains two computers in our experiments. One was used as the enterprise server the other one as the process server. Both work with a PentiumCore2Duo processor at 2.1Ghz and is equipped with 2GigaBytes of memory, uses high speed disks and internet connection. As the tests will show later, the bottleneck resource of the configuration is the computational requirement (CPU usage). Thus further technical parameters of the configuration are not described in details.

To run the test (simulate concurrent service requests) an additional computer was also necessary. Because the test machine only sends requests and receives responses to and from the orchestrated service its resource consumption is significantly lower then the load on the servers. None of the resources was consumed totally, so the bottleneck of the system was not the test computer. Note, that in real usage scenarios client requests are mostly distributed on several computers. Thus performance on the side of the client is usually not critical.

To simulate real integration scenario and data heterogeneities a toy example was used in our tests. Our virtual organization offers all kinds of healthcare and wellness services to the clients. To help estimate customers' demands and to give advice on healthy living the company decided to implement a portal based on SOA architecture and methodology. For example, a composite process is able to ask (retrieve) a client's parameters and propose optimal sport activity and personalized dietary. Since there are numerous enterprise systems at the organization, composite processes often communicate with several systems. To overcome data heterogeneities the leadership decided to align all services (and service interfaces) to a global schema of the domain sports and healthcare. In our test only the alignment of one service is presented. The standard Body-Mass-Index calculater (shortly *BMIcalculator*) service of our enterprise system is equipped with necessary up- and downcast transformations and the performance of its application in composite services is measured during several tests. Based on the proposals described in the previous section, the performance of 4 different implementations is compared during the tests. Moreover basic analytic equations of the previous section are also confirmed.

The standard BMI calculator service requests *full name*, *height* and *weight* as input parameters and responds the *full name*, the *BMI* and a *category* of a person as an output. Because the application was made in the UK, the service requests the height given in feet, the weight in pounds and awaits the full name given within one field – first name and last name separated with a space. However our applied global schema (and most of other enterprise applications of ours) calculates the height in centimeters and the weight in kilograms. Moreover the name of the clients is stored in a structure containing two fields: one for the first name and one for the last name. The responded category of BMI status should also be translated for Hungarian customers and other applications: categories *underweight*, *normal*, *overweight* and *obesity* are transformed into *sovány*, *normál*, *túlsúlyos* and *kórosan elhízott*. Thus up-cast transformation is required to change between different metrics and concatenate separated last name and first name and down-cast transformation is required to translate the value of category and split full name string value into two separated fields.

The standard BMI service and all necessary transformations and constructs of all proposals were implemented. For the implementation .Net framework, Visual C# and Active BPEL [1] designer were used. The running was performed on Internet Information Services (IIS) and on Active BPEL run-time engine. Native level enterprise transformations (proposal_Native) were also implemented in C# and XSLT transformation and the proxy service were executed on the .Net framework as well (proposal_XSLT). Process level transformations were defined using the Xpath and XQuery standards in the Active BPEL designer.

To determine basic efficiency of the given experimental configuration the standard BMI service were evaluated. First the bottleneck resource was searched. With 9 concurrent clients sending request continuously the CPU usage has reached 95% usage while the usage of other resources was significantly below the available maximum. (In our experiments the usage level of the bottleneck resource (CPU) has never reached higher values then 95-96%). Fig. 6 shows the usage level of all resources at this load.

Changing the number of concurrent client requests the response time and throughput moved closely enough to the expected values which confirms our Eqs. 1-5. Fig. 7 shows the changing of response time and throughput according to the changing in the number of concurrent clients. All tests were performed for 3-5 minutes until the average results have stabilized while values of the setup period were excluded from the results.

In the same way as the enterprise server environment the process server was tested, the process server was tested as well. A stand-alone process containing no external operation invokes was implemented for this purpose. Fig. 8 shows the usage level of resources at the reached 100% bottleneck resource load and Fig. 9 shows response time and throughput reflecting the changes in the number of concurrent process requests.

The implementation based on proposal XSLT was evaluated first. Similarly to previous test results the bottleneck resource

Tab. 1. Comprehensive table of proposals

Proposal\Aspect	*Resource overhead per enterprise operation invoke*	*Modularity of the orchestrated process*	*Reusability of transformations & encapsulated services*	*Applied technologies*
Proposal_XSLT	R_transf+ R_wsinterface	Modular encapsulated services	Encapsulated service are reusable	Process (BPEL)* and transformation (XSLT)
Proposal_Direct	R_transf	Orchestration logic is messed up by transformations	Transformations are NOT reusable	Process (BPEL)*
Proposal_BPEL	R_transf + R_constructs	Modular processes containing 1 operation and 2 transformations	Single processes containing 1 operation and 2 transformations are reusable	Process (BPEL)*
Proposal_Native	None	Same as in standard SOA architecture	Same as in standard SOA architecture	Process (BPEL)

* Process (BPEL) means that included technologies of the BPEL standard (e.g. XPath, XQuary) are used for transformation creation and execution as well.

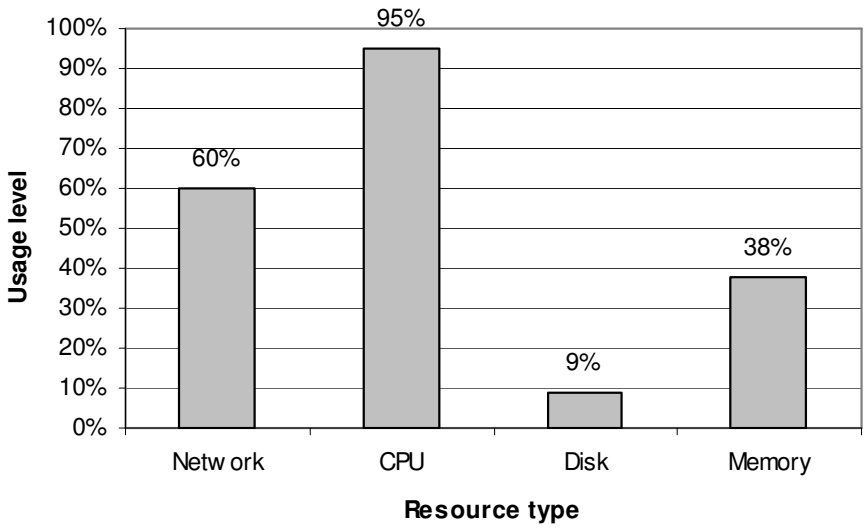
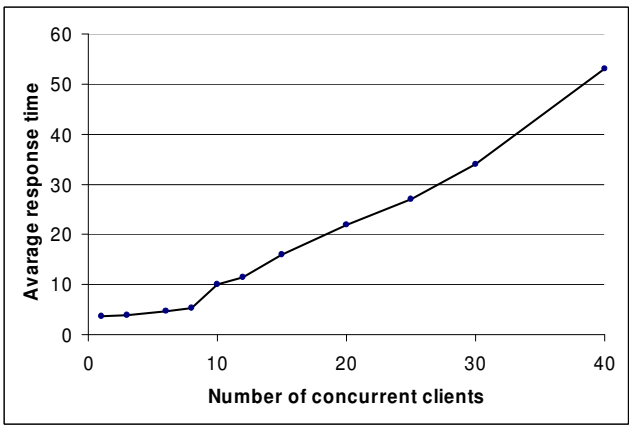


Fig. 6. Resource usage reaching maximum bottleneck resource usage by standard service invoke



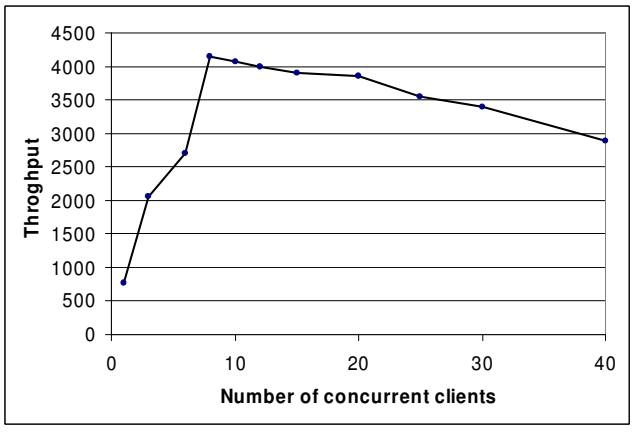


Fig. 7. Evaluation of response time and throughput invoking standard service

was the computational efficiency of the system, more precisely the process server has reached its maximum consumption of the CPU usage at a level of about 96%. Thus the CPU usage of the enterprise server was about 30% we conclude that $R_{\text{constructs}}$ is significantly greater than R_{orchest}. Fig. 10 show evaluation of response time and throughput reflecting to the changes in the number of concurrent requests. The experiments provide evidence on our Eqs. (1-5) and (10) to be found in the analytical section.

The evaluation of proposal_Direct showed the expected dif-

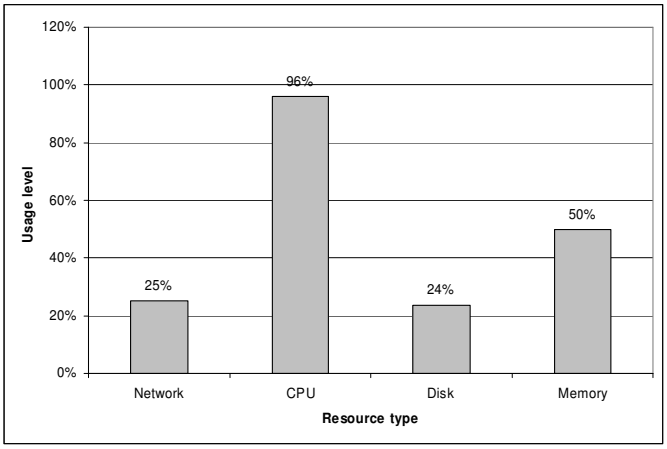


Fig. 8. Resource consumption reaching maximum bottleneck resource usage at the process server

ferences compared to proposal XSLT. However differences in response time and throughput are disappearing between the two proposals corresponding to a continuously increasing load over the maximum loaded interval. Response time and throughput are showed in Fig. 10. Based on the test results the following relations can be stated: $R_{\text{ws\_interface}}$ influences the overall performance not much and $R_{\text{add\_Direct}}$ is not significant compared to $R_{\text{constructs}}$. Furthermore measured values of response time confirmed the Eq. 13.

Proposal BPEL was observed to be the slowest approach according to the test results. Adding $R_{\text{constructs}}$resource overhead to every participated operation invoked caused a significant increase in response time and a decrease in throughput compared to other proposals. Fig. 11 shows test results.

Proposal Native has slightly outperformed the other 3. On one hand the reducing of costs by eliminating transformations was yet not significant compared to $R_{\text{constructs}}$, on the other hand some computational resources are still required because of the extended functionality of the standard service. The resource need caused by $R_{\text{constructs}}$is determining again which exhausts the bottleneck resource (CPU) on the process server. Fig. 12 shows response time and throughput according to changes in the number of concurrent requests (clients).

The reader may recognize that the response time also increases in the period before maximum load has reached. Although it seems to be conflicting to the statements of the previous section, regarding the hardware specification of our configuration the explanation is trivial. The CPU of the computers contains two (or even more) computational units today. The system optimizes the resource allocation dedicating different tasks for parallel execution resulting a performance growth up to the number of additional processor cores. However the response time can only be decreased this way until the load has reached its maximum. The advantage of having two (or more) computational units is fully consumed when maximum usage of the CPU is hit. In our case the one additional core caused 2 times shorter response time at low load which has decreased continuously till the maximum throughput has been reached.

The performed tests have shown the applicability of each proposal and proved that the analytic results presented in the previous section were right.

Conclusions and possible areas of future work are presented in the next section.

5 Conclusion and future work

This paper has presented an approach to bridging semantic heterogeneities during the enterprise application integration. The framework is based on the concepts, standards and technologies of the SOA methodology. To evaluate the efficiency of our solution analytical tools were developed and presented in the paper. Apart from our approach, three further proposals were implemented and compared to each other. The analytical results were reinforced by extensive experiments and the test results have confirmed the performance values expected from formal calculations. Hence, the paper has presented a complete methodology and detailed technological proposal to implement integration configurations and predict non-functional QoS parameters of orchestrated services.

The approach has presented solution for attaching and executing transformations before (and after) standard service invoke (and after service response). However the paper gives no promotion to detect data inconsistencies and to create up- and downcast transformations. The integration environment and the number of participating enterprise services can be huge at large organizations making the detection of semantically related concepts and the designing of proper transformations rather inconvenient for human integration experts and developers. Tools realizing (semi)-automated assistance of this process may be needed.

The experiments were only performed on one given configuration and implemented on one given task (integration of the BMI calculator service). More complex tests and more refined configurations would probably lead to more detailed results. Real services of standard enterprise application could also be involved into the analysis and experiments in the future.

References

1 *Active Endpoints – Active BPEL*, available at `http://www.activevos.com/community-open-source.php`. viewed 12. nov. 2008.

2 **Al-Masri E, Mahmoud Q H**, *Investigating Web Services on the World Wide Web*, Proceeding of the 17th international conference on World Wide Web, 2008, pp. 795-804.

3 **Baresi L, Ghezzi C, Guinea S**, *Smart Monitors for Composed Services*, Proceedings of the 2nd international conference on Service oriented computing, 2004, pp. 193-202.

4 **Blazona B, Koncar M**, *HL7 and DICOM based integration of radiology departments with healthcare enterprise information systems*, International Journal of Medical Informatics **76S** (2007), 425-432.

5 **Buttler D**, *A Short Survey of Document Structure Similarity Algorithms*, Proceedings of the 5th international conference on internet computing, 2004.

6 **Canfora G, Di Penta M, Esposito R, Villani M L**, *A framework for QoS-aware binding and re-binding of composite web services*, Journal of Systems and Software **81** (2008), 1754-1769.

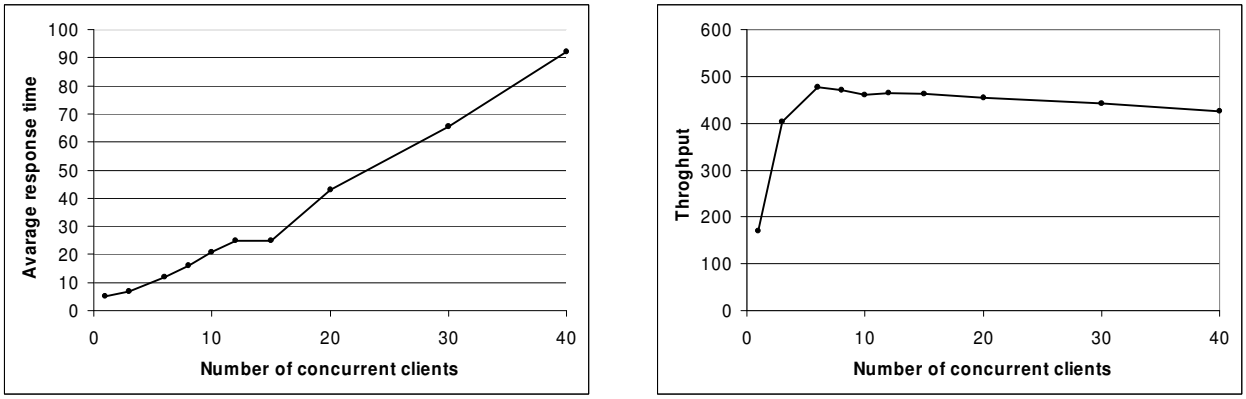


Fig. 9. Evaluation of response time and throughput by invoking empty process

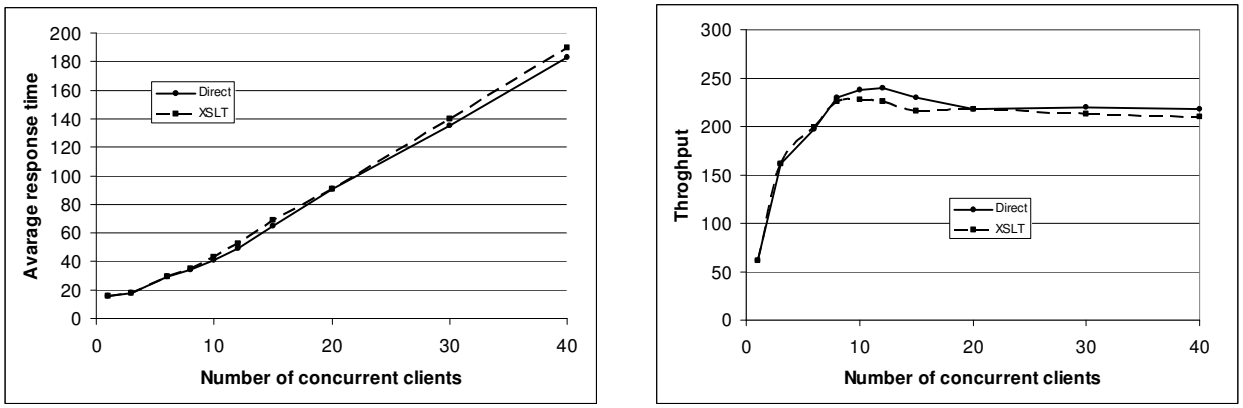


Fig. 10. Evaluation of response time and throughput by invoking processes of proposals XSLT and Direct

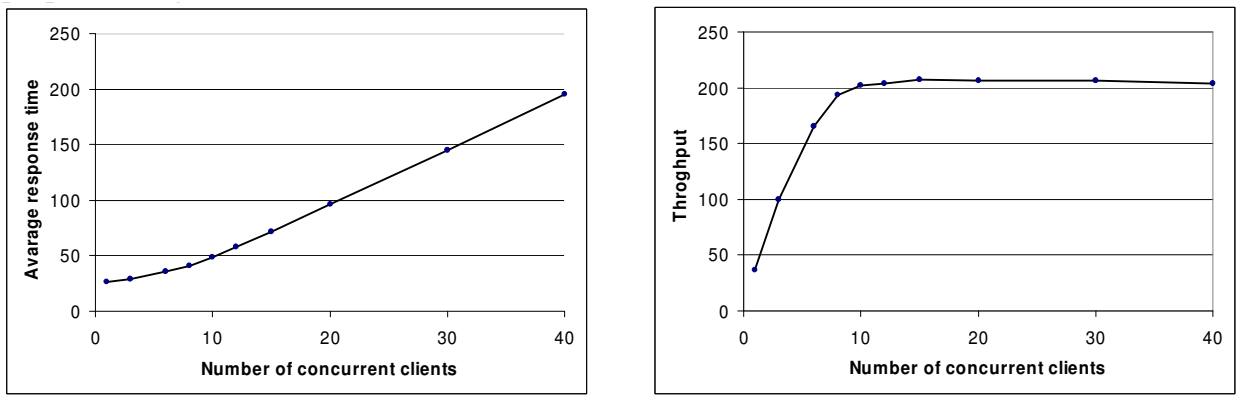


Fig. 11. Evaluation of response time and throughput by invoking processes of proposal BPEL

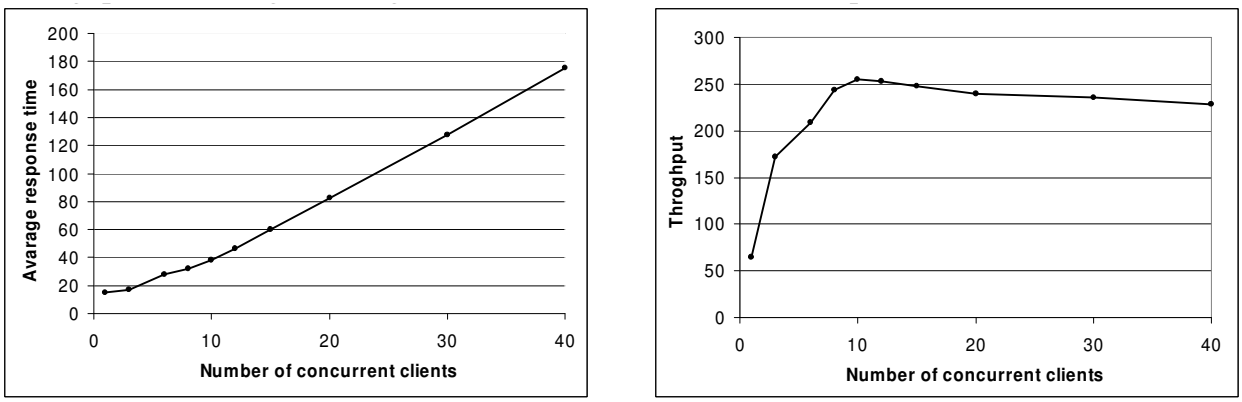


Fig. 12. Evaluation of response time and throughput by invoking processes of proposal Native

7 **Cardoso J**, *Complexity analysis of BPEL web processes*, Software Process: Improvement and Practice **12** (2006), 35-49.

8 **Hong-Hai Do, Erhard Rahm**, *Matching large schemas: Approaches and evaluation, Information Systems* **32** (2007), 857-885.

9 **Goodchild M F, Max J. Egenhofer, Robin Fegeas, Cliff Kottman**, *Changing focus on interoperability in information systems: from system, syntax, structure to semantics*, Changing focus on interoperability in information systems: from system, syntax, structure to semantics (Amit P. Sheth, ed.), Springer, 1999. chapt. 2, pp. 5-30.

10 **Guglielmina C, Kiauleikis M, Tolle K, Morkevicius N**, *Performance and Architecture Modeling of Interoperability System for SME's*, Lecture Notes in Business Information Processing **7** (2008), 345-356, DOI 10.1007/978-3-540-79396-0_30.

11 **Kavianpour M**, *SOA and Large Scale and Complex Enterprise Transformation*, Proceedings of the 5th international conference on Service-Oriented Computing, 2007, pp. 530-545.

12 **Lamparter S, Ankolekar A, Oberle D, Studer R, Weinhardt C**, *Semantic specification and evaluation of bids in web-based markets*, Electronic Commerce Research and Applications **7** (2008), 313-329.

13 **Madhavan J, Bernstein P A, Rahm E**, *Generic Schema Matching with Cupid*, Proceedings of the 27th International Conference on Very Large Data Bases, 2001, pp. 49-58.

14 **Martinek P, Szikora B**, *Detecting semantically related concepts in a SOA integration scenario*, Periodica Polytechnica. in press.

15 **Martinek P, Tóthfalussy B, Szikora B**, *Implementation of Semantic Services in Enterprise Application Integration*, WSEAS Transactions on computers **7** (2008), 1658-1668.

16 **Maximilien E M, Singh M P**, *A Framework and Ontology for Dynamic Web Services Selection*, IEEE Internet Computing, 2004, pp. 84-93.

17 **Melnik S, Garcia-Molina H, Rahm E**, *Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching*, Proceedings of the 18th International Conference on Data Engineering, 2002, pp. 117-128.

18 **Moser O, Rosenberg F, Dustdar S**, *Non-Intrusive Monitoring and Service Adaptation for WS-BPEL*, Proceeding of the 17th international conference on World Wide Web, 2008, pp. 815-824.

19 **I. Navas-Delgado, M. del Mar Roldán-García, J. Francisco Aldana-Montes Kreios**, *Towards Semantic interoperable systems*, Lecture notes in computer science **3261** (2004), 161-171.

20 *OASIS - Web Services Business Process Execution Language WS-BPEL (ver. 2.0)*, available at `http://docs.oasis-open.org/wsbpel/2.0/wsbpel-specification-draft.html`. viewed 3. febr. 2009.

21 *Advanced Open Standards for the Information Society (OASIS) - e-business XML (ebXML)*, available at `http://www.ebxml.org/`. viewed 10. febr. 2009.

22 **Pautasso C, Heinis T, Alonso G**, *Autonomic resource provisioning for software business processes*, Information and Software Technology **49** (2007), 65-80.

23 **Peltz C**, *BWeb services orchestration and choreography*, IEEE Computer **36** (2003), no. 10, 46-52.

24 **Sato N, Trivedi K S**, *Stochastic Modeling of CompositeWeb Services for Closed-Form Analysis of Their Performance and Reliability Bottlenecks*, Lecture Notes In Computer Science **4749** (2007), 107-118.

25 **Shen J, Grossmann G, Yang Y, Stumptner M, Schrefl M, Reiter T**, *Analysis of business process integration in Web service context*, Future Generation Computer Systems **23** (2007), 283-294.

26 **Theodoratos D**, *Semantic Integration and Querying of Heterogeneous Data Sources Using a Hypergraph Data Model*, Lecture Notes In Computer Science **2405** (2002), 166-182.

27 **Thompson S, Giles N, Li Y, Gharib H, Duong Nguyen T**, *Using AI and semantic web technologies to attack process complexity in open systems*, Knowledge-Based Systems **20** (2007), 152-159.

28 *W3C – XML Path Language (XPath)*, available at `http://www.w3.org/TR/xpath`. viewed 22. dec. 2008.

29 *W3C – XML Query Language (XQuery)*, available at `http://www.w3.org/TR/xquery/`. viewed 21. dec. 2008.

30 *W3C – Simple Object Access Protocol (SOAP)*, available at `http://www.w3.org/TR/soap/`. viewed 27. jan. 2009.

31 *W3C, Web Service Description Language*, available at `http://www.w3.org/TR/wsdl,2001.03.15.`.

32 *W3C – XSL Transformations (XSLT)*, available at `http://www.w3.org/TR/xslt`. viewed 12. jan. 2009.

33 **Wahler A, Schreder B, Balaban A, Gomez J M, Niederacher K**, *MIKSI - A Semantic and Service Oriented Integration Platform*, The Semantic Web: Research and Applications, 2004, pp. 459-472, DOI 10.1007/978-3-540-25956-5_32.

34 **Junbiao Wand, Hu Deng, Jianjun Jiang, Binghong Yang, Bailing Wang**, *EAI-oriented information classification code system in manufacturing enterprises*, Frontiers of Mechanical Engineering in China **3** (2008), no. 1, 81-85.

35 **BangYu Wu, Chi-Hung Chi, Zhe Chen, Ming Gu, JiaGuang Sun**, *Workflow-based resource allocation to optimize overall performance of composite services*, Future Generation Computer Systems **25** (2009), 199-212.

36 *XML Comman Business Liberary (XCBL)*, available at `http://www.xcbl.org/`. viewed 10. febr. 2009.

37 **Ye Y, Yang D, Jiang Z, Tong L**, *Ontology-based semantic models for supply chain management*, The International Journal of Advanced Manufacturing Technology **37** (2008), no. 11-12, 1250-1260.

38 **Liangzhao Zeng, Benatallah B, Ngu A H H, Dumas M, Kalagnanam J, Chang H**, *QoS-aware middleware for Web services composition*, IEEE Transactions on Software Engineering **30** (2004), 311 - 327.