University of Stuttgart
Germany

Institute of Architecture of Application Systems

Making Scientific Applications on the Grid Reliable through Flexibility Approaches Borrowed from Service Compositions

Dimka Karastoyanova, Frank Leymann

Institute of Architecture of Application Systems, University of Stuttgart, Germany
{Karastoyanova, Leymann}@iaas.uni-stuttgart.de

BibTeX:

```
@inbook {INBOOK-2010-03,
    author = {Dimka Karastoyanova and Frank Leymann},
    title = {{Making Scientific Applications on the Grid Reliable
              through Flexibility Approaches Borrowed from Service
              Compositions}},
    series = {Handbook of Research on P2P and Grid Systems for
              Service-Oriented Computing: Models, Methodologies
              and Applications. Volume II.},
    publisher = {IGI Global},
    pages = {635--656},
    year = {2010},
    doi = {10.4018/978-1-61520-686-5},
    language = {English}
}
```

Making Scientific Applications on the Grid Reliable through Flexibility Approaches Borrowed from Service Compositions

Dimka Karastoyanova, Frank Leymann
IAAS, University of Stuttgart, Germany
e-mail: {Karastoyanova, Leymann}@iaas.uni-stuttgart.de

ABSTRACT

The current trend in Service Oriented Computing (SOC) is to enable support for new delivery models of software and applications. These endeavours impose requirements on the resources and services used, on the way applications are created and on the QoS characteristics of the applications and the supporting infrastructure. Scientific applications on the other hand require improved robustness and reliability of the supporting Grid infrastructures where resources appear and disappear constantly. Enabling business model like Software as a Service (SaaS), Infrastructure as a Service (IaaS), and guaranteeing reliability of Grid infrastructures are requirements that both business and scientific application nowadays impose. The convergence of existing approaches from SOC and Grid Computing is therefore an obvious need. In this work we give an overview of the state-of-the-art of the overlapping research done in the area of SOC and Grid computing with respect to meeting the requirements of the applications in these two areas. We show that the requirements of business applications that already exploit service-oriented architectures (SOA) and the scientific application utilizing Grid infrastructures overlap. Due to the limited extent of cooperation between the two research communities the research results are either overlapping or diverging in spite of the similarities in requirements. Notably, some of the techniques developed in each area are needed but still missing in the other area and vice versa. We argue therefore that in order to enable an enterprise-strength service-oriented infrastructure one needs to combine and leverage the existing Grid and Service middleware in terms of architectures and implementations. We call such an infrastructure the Business Grid. Based on the Business Grid vision we focus in this work on presenting how reliability and robustness of the Business Grid can be improved by employing approaches for flexibility of service compositions. An overview and assessment of these approaches are presented together with recommendations for use. Based on the assumption that Grid services are Web services, these approaches can be utilized to improve the reliability of the scientific applications thus drawing on the advantages flexible workflows provide. This way we improve the robustness of scientific applications by making them flexible and hence improve the features of business applications that employ Grid resources and Grid service compositions to realize the SaaS, IaaS etc. delivery models.

1. INTRODUCTION

Today, Grid infrastructures are mainly used for scientific computations dealing with considerable amounts of (experimental) data and performing extensive, time-consuming computing tasks. Usually, the hardware and software resources used to run the computations are out of the control of the application owners. The resources used for the computations may be used only for a concrete time period and may appear and disappear in an unpredictable fashion. It has been documented in research publications that the major challenge to be overcome on the Grid is its reliability (Fox & Gannon, 2006).

This book chapter will focus on approaches for making the Grid more reliable. We argue that Grid applications can be made more reliable through making them more flexible. We will present approaches towards flexibility of applications on the Grid. An objective of this work is to show that approaches and infrastructure created by the SOA community can be used for the benefit of the Grid community. Moreover, since approaches from the Grid are needed on the SOA infrastructures, such leverage of concepts and techniques will allow for a mutual amplification of these benefits for both communities. The major assumption in the chapter is that Grid and Web Service technologies can be combined to enable more reliable infrastructure for both business and scientific applications. We argue that only through this combination an infrastructure for supporting both business and scientific applications can be created, which fosters leverage of existing technologies, mechanisms and techniques, and can provide added value to the users in both domains. We are convinced that the two domains have a lot to leverage and learn from each other and employ the research results to the advantage of both communities.

The chapter will start with an overview of the SOA (Service Oriented Architecture) paradigm, its role model and the operations an SOA infrastructure supports. We also present the Web Service technology as the only available implementation of the SOA paradigm. It is only through service composition that complex business and scientific computations can be enabled, therefore we present an overview of the process-based approach for service composition known from the field of Business Process Management (BPM). In particular, we identify the Business Process Execution Language (BPEL) as the one with the greatest potential to facilitate the creation of complex applications for business and scientific computations. We present the architecture of a (Web) Service Middleware, also known as the Bus or the Enterprise Service Bus (ESB).

Afterwards we shift the focus onto Grid services and existing technologies, whereas we pay the greatest attention to the Web Service Resource Framework (WSRF), which is the latest technology for Gird services and represents the convergence of research efforts of the Grid and SOA communities. WSRF is a framework that allows for representing stateful resources (hardware: like processors, hard disks, software: applications, processes etc.) as services, whose state, also called properties, can be retrieved, manipulated, and processed in a meaningful way.

We compare the Web Service and Grid technologies, their overlaps and the potential for synergies between them, to facilitate better understanding.

We continue with a discussion about service-based business applications and their most important characteristics, as required by the application domains. We also investigate the characteristics of the existing infrastructures supporting them and identify missing but required functionality. In particular we stress on the need for reliable and scalable applications for enacting business transactions in a dynamic and competitive market. We also present the characteristics of applications currently run on the Grid. Most of the existing applications on the Grid are scientific applications that deal with huge amounts of data and long time to compute or analyse results. Additionally, we emphasise the fact that reliability of the Grid still needs improvements to meet the needs imposed by scientific applications, in particular because of the fact that resources appear and disappear on the Grid constantly. Another drawback of the existing approaches for enabling Grid applications we identify is that the reusability of applications is not fostered to the full and they are not flexible to changes in the environment and fault tolerant. Usually a scientist would create an application from scratch, or just execute the steps needed for a complex computation himself, and wait every time a piece of this computation is run, gather the results, prepare them for the next computation step and start this step. Existing computations implementations, from the areas of e.g. astrophysics, high energy nuclear physics and computational genomics, are not reused across organizations, steps of computations are not pre-defined and cannot be used multiple times with different configurations.

We discuss the vision of Business Grid as the way to follow for enabling reliable, robust and scalable applications. The Business Grid is a composable middleware that combines techniques available on the Bus and in Grid infrastructures. The main characteristics of the Business Grid will be summarized and its architecture is presented. We will show that this architecture addresses the requirements posed on Grid applications from the field of scientific computing and simulation technologies. We argue that instead of creating new concepts, architectures and technologies, the existing once from Grid and SOA can be combined in an advantageous way to support both business and scientific applications. This combination of technologies will enable reliability and scalability for Grid applications and ability to perform long, data intensive computations for business applications. In the same section we will list

the requirements on the Business Grid imposed by both business applications and Grid computations. The requirements will be classified according to the characteristics needed and to the supporting infrastructure.
We shall identify the reliability of the Business Grid as the characteristic that needs to be urgently enabled, since both the ESB and the Grid are not yet capable to perform reliably for the needs of application operating on huge data sets for a longer time in a multi-tenant environment. We advocate the support for flexibility of applications on the Business Grid as the instrumental approach towards enabling reliability of such applications.
In order to facilitate the comprehension of the approaches toward flexibility that we present later in this chapter, we start with a classification of approaches to flexibility of applications on the Business Grid. These encompass flexibility on the level of the middleware as well as on the level of the applications. Based on this classification we present approaches towards flexibility of Business Grid applications in detail. These include approaches that enable reaction to failing services like dynamic binding and re-binding, adaptation of process-based applications using new constructs/extensions to existing process composition languages, using the AOP paradigm, using ontological descriptions of services and processes, and others. Even though most of these approaches have been applied for business applications, we will show that they are applicable and can be leveraged for scientific applications. For each of the approaches we will present the enhancements that need to be made in order to customize them for scientific applications, if necessary. We compare these approaches and identify their advantages and disadvantages. Based on this we produce recommendations about in which cases to apply these approaches and what the effort to employ them would be. The main principle we follow in this comparison is their non-intrusiveness and compliance to standards, as well as reuse of investment in technology and skills.
This chapter will be concluded by an overview of what is missing to enact the Business Grid for it to hold to its promise. At the end we provide a summary and conclusions.

2. SOA, WEB SERVICES AND GRID

2.1 The Service Oriented Architecture (SOA) - Overview

SOA is one natural continuation in the evolution of integration technologies (Weerawarana & Curbera & Leymann & Storey & Ferguson, 2005). It aims at enabling interoperability within and across organizations using the *service* abstraction, loose coupling, composability, and fostering technology and investment leverage. An additional goal of SOA is the automation of business processes, which has a great practical importance. Services are self-contained and self-describing units of functionality exposing *stable interfaces* in a unified format independent of implementation specifics and interaction formats and protocols. They render a heterogeneous environment homogeneous. Thus services allow for standardized access and identification of functionality/applications and thus facilitate application integration. *Loose coupling* is the basic principle of service orientation (Kaye, 2003), (Weerawarana & Curbera & Leymann & Storey & Ferguson, 2005), (Gehlert & Hielscher & Danylevych & Karastoyanova, 2008). In a service-oriented environment the assumptions a service consumer has about a service are reduced; the knowledge about the architectural style, implementation, programming language, native protocols used to realize a service is not needed by the service consumer. Service clients know only the stable interface exposed by the service. The client applications using this service will not break if the implementation of the service is changed as long as the service interface is maintained stable. Loose coupling is also enabled by means of *messaging*, which supports asynchronous mode of communication and enables time and space decoupling of applications.
The SOA architectural style specifies three major *roles* and three *operations* (see Figure 1). Service providers are entities (organizations, persons, departments etc.) that provide services for use by service consumers/clients. The services may be developed by the organization providing them or by any other organization. To enable discoverability of services a discovery component (also called registry or sometimes called broker), is defined as a role in SOA. The provided services are registered with the discovery component and thus made available for discovery by potential service consumers. An entity may play any of the three roles. The *operations* defined by SOA are: find, register and bind. The find

operation is used by service consumers to discover services in a service registry. The service providers use the register operation to provide information about their services in the discovery component. Once a client discovers an appropriate service he binds to the service and starts interacting with it. These operations are realized by the service middleware, the so-called service bus (Chappell, 2004), (Leymann, 2005), and are made available to the entities playing any of the SOA roles. The invocation of services is performed by the bus. For this it receives as input from the service consumer the input data for the service invocation and the service description, more precisely a service operation; as a result it returns the output produced by the service. If the service consumer has not provided the location of the service and the access mechanism, it is up to the bus to discover a service on behalf of the client that meets his requirements with respect to functionality and quality characteristics of the service, usually known as quality of service (QoS).

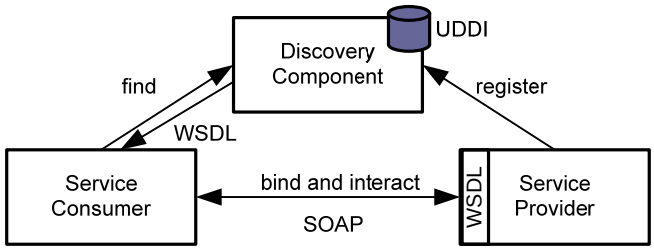

Figure 1. The SOA paradigm and its implementation in terms of the Web Service technology

2.2 Web services

Web services (WSs) are the only existing standardized implementation of SOA. This technology provides a model for use of (existing) applications, not for programming applications. Web services are about virtualisation of applications, which also explains their already huge success in industrial applications and in utility and cloud computing (Mietzner & Leymann, 2008). Services can be implemented in any programming language for any platform, as well as they can be exposed for use via any of the existing interface description languages (IDLs). The novelty in Web services is the notion of binding, which is reflected by the standard language for WS interface description, the Web Service Description Language (WSDL) (W3C, 2001). Unlike any other existing IDL, WSDL keeps the information about the transport protocol and message encoding separate from the actual interface description (signature). This allows for exposing the same functionality/program/implementation on different endpoints/ports by combining a service interface description with multiple bindings. This improves modularity and facilitates loose coupling between service providers and consumers.

The WS technology draws on experience in application integration from both academia and industry and meanwhile enjoys a huge support on behalf of industry. The technology has been created for the purposes of application integration and therefore it possesses the features inherent for such a technology (Kaye, 2003), (Weerawarana & Curbera & Leymann & Storey & Ferguson, 2005). WSs enable machine-to-machine interaction across heterogeneous systems/environments by rendering them homogeneous using a unified service interface description format. WSs do not preclude the use of multiple communication modes and transport protocols with different QoS. Of great practical importance is the fact that WSs allow the leverage of existing software and its reuse in new applications in a very flexible manner. WSs can be combined in complex compositions, which can in turn be provided as WS too. The Web Service technology enables composability of technology and specifications. There are many specifications in the WS protocol stack dealing with different concerns of application development and integration. These specifications need not be all used in all WS applications if the features they specify are not needed. Standardization is also one of the major goals of WSs with the purpose of interoperability.

Apart from WSDL, SOAP and UDDI are also considered part of the basic WS protocol stack. SOAP specifies a message format and message processing model independent of the transport protocols used on the message path. UDDI (Universal, Description, Discovery and Integration) defines a data model for storing information about services in service registries and an API for interaction with UDDI registries.

Due to the advantages WSs possess, industry has already invested heavily in creating supporting infrastructures for WSs. There are already even commercial implementations of service buses by all major software vendors.
Recently there have been developments in the area of applying semantic information in the field of WSs, where meta-information about WSs is described in terms of ontologies. This research area targets the improvement of automation of discovery of WSs and their use in business processes across multiple application domains.
The WS technology *virtualizes applications* in a unified manner and thus fosters interoperability and integration. Recently, the need has arisen to *virtualize hardware resources* in particular in the area of scientific computing and simulation, where the computations are lengthy and resource-consuming due to the nature of the computations and the huge amounts of distributed data used and produced. This need was addressed partly by the Grid, and its name stands for a distributed IT infrastructure for advanced science and engineering applications (Foster, 2002), (Foster & Kesselman & Tuecke, 2001).

2.3 Grid and Grid Services

Initially the intent of the *Grid* was to virtualize computing resources and thus supply scientific collaborations with more computing power than the computing resources each concrete domain provides (i.e. the local IT infrastructures of computing centres). In other words, the Grid is viewed as a solution to the problem of "flexible, secure, coordinated resource sharing among dynamic collections of individuals, institutions and resources" (Foster, 2002). These dynamic collections of individuals, institutions and resources are referred to as "virtual organizations" (Foster, 2002) and can comprise different groups of scientists working on a common goal but can as well be different companies that join forces or are in a supplier-requestor relationship in order to reach a certain (mostly computing intensive) business goal. A fundamental principle virtual organizations follow is the sharing of files or documents, and also of access to programs and even hardware resources.
The organizations that form a virtual organization can only share computing resources if a standardized architecture for the Grid as well as interfaces is at hand to enable the interoperability among Grid resources. The Grid community introduced the Open Grid Services Architecture OGSA (Foster & Kesselman, 2004), which follows the principles of SOA and provides a standardized set of services important in a Grid environment (such as registry, authorization monitoring, data access and others) (Foster & Kesselman & Tuecke, 2001). The technology to realize the OGSA is Web services, thus allowing Grid services to be described and used in a standardized and interoperable manner. The Open Grid Services Infrastructure (OGSI) (Tuecke et al., 2003) and its successor the Web Service Resource Framework (WSRF) (Czajkowski K. et al., 2004), (OASIS WSRF TC, 2008) are specifications that define how to specify stateful Web services that represent stateful resources (such as computers or storage) as they are present in Grid environments. WSRF specifies the so-called *WS-Resource*, which is a WS that allows a unified access to computing resources on the Grid. Apart from possessing a WS interface that exposes the functions a resource implements in terms of WS operations, the WS-Resource also references a resource property document (see Figure 2**Fehler! Verweisquelle konnte nicht gefunden werden.**). The properties document contains a description of the resource properties and allows for viewing the resource state that may include information like storage capacity, number of discs, processor frequency, resource type, current workload etc.

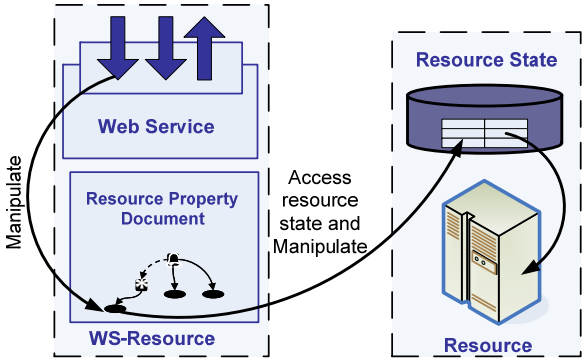

Figure 2. Web Service Resource

Additionally, WSRF specifies standardized interfaces for lifecycle control, standardized faults, notifications (WS-Notification) in separate composable specifications, quite in the sense of the WS technology. Compare to the OGSI, where all these orthogonal features were part of one and the same specification, and all OGSI compliant services had to implement all the features, regardless of the requirements.

2.4. Composing WSs and Grid Services

WS and Grid services (i.e. WS-Resources) can be combined in more complex applications using some composition approach. The workflow- or process-based approach (Leymann & Roller, 1999), (van der Aalst & van Hee, 2002) is the most favoured one for creating compositions of services. The process-based approach to service composition, also known as service orchestration, allows composing services in a very flexible manner by means of improved modularity, separation of concerns and configurability. Processes are created in terms of (i) control logic that specifies the sequencing of tasks and the data they exchange, (ii) the functions/services that implement the tasks and (iii) the human participants in an organization that participate in carrying out the tasks. These entities form the so-called workflow/process dimensions. In (Web) service compositions, due to the fact that services do not involve people or hide their participation, the organizational dimension is not present. The definition of a workflow or process used here is based on those presented in BPM (Business Process Management) literature like (Leymann & Roller, 1999) and (van der Aalst & van Hee, 2002). It is important to note that the workflow technology distinguished between a workflow model and workflow instances. A model can be run multiple times, i.e. multiple workflow instances can be executed simultaneously, which brings significant time savings. To the best of our knowledge, based on our participation in multiple projects together with the scientific computing community, the definitions of the notion of workflow first differ from group to group in the scientific world, and second sands for only the sequencing of tasks and the data dependencies. According to these definitions workflows do not include the features from the workflow technology, like: interruptible sequences of tasks, modelling of constraints to transitioning from task to task in a global model of the overall computation, modelling parallel execution of independent tasks, forward and backward recovery, fault handling, staff assignment per task. The available scientific workflow infrastructures do not always distinguish among workflow model and workflow instances; moreover, some of them are tailor-made for a concrete scientific domain, which the workflow engines employed in business applications indeed avoid by implementing a generic workflow meta-model suitable for all application domains. In this work we have as a starting point the workflows or processes as defined by the BPM community.
Usually processes are modelled using some graphical notation, however in order to be executed (on a workflow engine/system) they need to be transformed into some executable format. There are many workflow languages and BPEL (OASIS BPEL TC, 2008) is the standard language for describing service orchestrations. It contains constructs for control flow and data manipulation, as well as the so-called interaction activities which model the interaction with WSs that implement tasks in a workflow, e.g. book hotel, calculate credit risk, generate a mesh in a FEM computation etc. The interaction activities contain references to the service port type and operation and specifies the input and output data. In BPEL the control flow and the functions (WSs) used are predefined during the modelling of the composition and remain unchanged during the process execution. Currently, flexibility of the processes is enabled due to the fact that concrete service implementations (called also ports) can be assigned to tasks during the process deployment onto the execution environment or during run time. For the latter case, it is up to the service bus to discover, select and invoke a concrete service that complies with the specification of the process task and the desired QoS characteristics. The types of services used are however the same during the whole life cycle of the process. Processes are executed by process/orchestration/workflow engines and currently these engines make use of the service middleware to invoke services (see Figure 3). The use of Grid infrastructures for involving WS-Resources into BPEL service orchestrations has great potential and will allow for flexible mixing and interchange of WSs and WS-Resources from the point of view of service orchestrations. This should be addressed by research and engineering efforts in this direction.

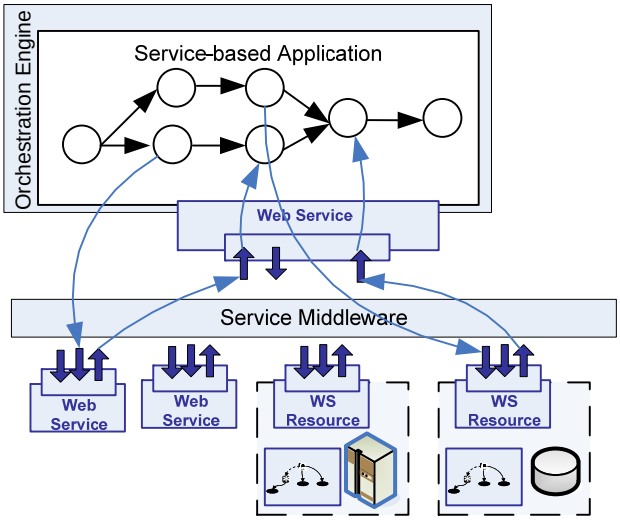

Figure 3. Interplay among Service compositions, WSs and Grid services

BPEL is being extensively used in business applications and all big software vendors have rolled out products with BPEL support. There are preliminary attempts to utilize BPEL for composing scientific computations using existing algorithm and computation implementations (Emmerich & Butchart & Chen & Wassermann & Price, 2005), (Fox & Gannon, 2006), (Leymann, 2006). Admittedly, there are many open issues that still have to be considered in the area of applying the SOA paradigm, Web services and Grid services for scientific applications (SimTech, 2008), (Gannon, 2007). The major critical issues are in particular the robustness, scalability and availability the infrastructure and resources to maintain the long-running scientific application up and running.

A current trend in business applications and supporting infrastructures is to use Grid infrastructures. It is observed that there is a significant lack in support for the use of huge data sets in business applications and in the scenarios where extensive computations are required, like e.g. analysis of customer data, predictions, monitoring of business activities and correlation of such activities, text and multimedia processing and search (Clarin, 2008). While SOA has great success in improving the support for business applications, techniques from Grid can be utilized to enhance this support further. Some of the reasons for the modern SOA infrastructures to not yet support such features can be explained by the requirements business applications up till now had on technologies. Modern business applications however are much more complex and require most of the features mentioned above.

Indeed, both business and scientific applications miss certain features which the other domain has already implemented. Usually, each industry stays in its own confines and keeps developing its own technology and standards. With the advent of service-orientation, which is employed by both communities, now there is a chance to discontinue such practices and unify the efforts in developing composable infrastructures comprising concepts, techniques and implementations from different domains for the mutual benefit. The vision of the service bus (Chappell, 2004) has laid the foundations of the architecture of such a composable infrastructure. It however has considered only application or software services as part of the service landscape. The Grid community has shown that the SOA paradigm can be applied to Grid infrastructures. In the next section we introduce the architecture of the so-called Business Grid, which is meant to combine these two separate developments in a unified infrastructure for business and scientific applications.

3. THE BUSINESS GRID

With the advance of SOA and WSs in particular, the internet has transformed into a ubiquitous integration platform. The Grid resources are also WSs and this provides the opportunity to utilize more, increasingly powerful computing resources across organisations from science and business domains. The enormous potential of the Grid to improve the utilization of idle resources to perform computing intensive tasks or even sell them to any organization becomes evident. The Grid therefore transforms from an infrastructure for scientists into an infrastructure for enterprises, too. It is this unique combination of SOC and Grid paradigms that makes this possible. We argue that the currently

divergent infrastructures and research developments can mutually amplify their applicability, features and added-value characteristics if combined in a unified infrastructure. We call such an infrastructure the *Business Grid*. The Business Grid is a vision introduced in our recent research work (Mietzner & Karastoyanova & Leymann, 2008). He we present only an overview of the Business Grid to allow for the positioning of the approaches for flexibility discussed in the next sections.

Based on the analysis of requirements imposed on business and scientific applications (Mietzner & Karastoyanova & Leymann, 2008) we can state that business and scientific applications have similar requirements with respect to multiple features like composability of services, modularity, ease of modelling and execution of applications, seamless and transparent support for service discovery, selection and invocation, availability of multiple transport protocols and encoding formats, flexibility, reliability and robustness. These similar requirements have been addressed by the existing infrastructures created in both domains. We also observed that the infrastructures built by these two communities still lack support for features that the opposite community has already enabled. Furthermore, we have also observed that because of the changed application requirements in these two domains the business applications need features available on the Grid and scientific applications require techniques from service-oriented business infrastructures. The Business Grid has been introduced with the purpose of addressing the collective requirements of these two fields. Our motivation is based on the fact that apart from avoiding the development of overlapping approaches and technologies and duplicate infrastructure implementations, some of the approaches from the Grid may be successfully used to address open issues in the business applications domain, like handling of big data sets, improvements in the long-running execution of applications using provisioning techniques (Keller & Badonnel, 2004), optimization of resource utilization, whereas techniques available in the business applications domain can be utilized to address challenges still prevalent on the Grid, like for instance scalability, flexibility, robustness, composition, and resource/service discovery. The architecture of the Business Grid is presented in Figure 4; for more details consider (Mietzner & Karastoyanova & Leymann, 2008). This architecture is meant to be implemented as a composition of existing implementations of Grid and Service Bus implementations, and the necessary add-ons. It has been meant to provide a framework for integrating existing infrastructures and approaches from the two different domains under the SOA principles of reuse, composability and interoperability.

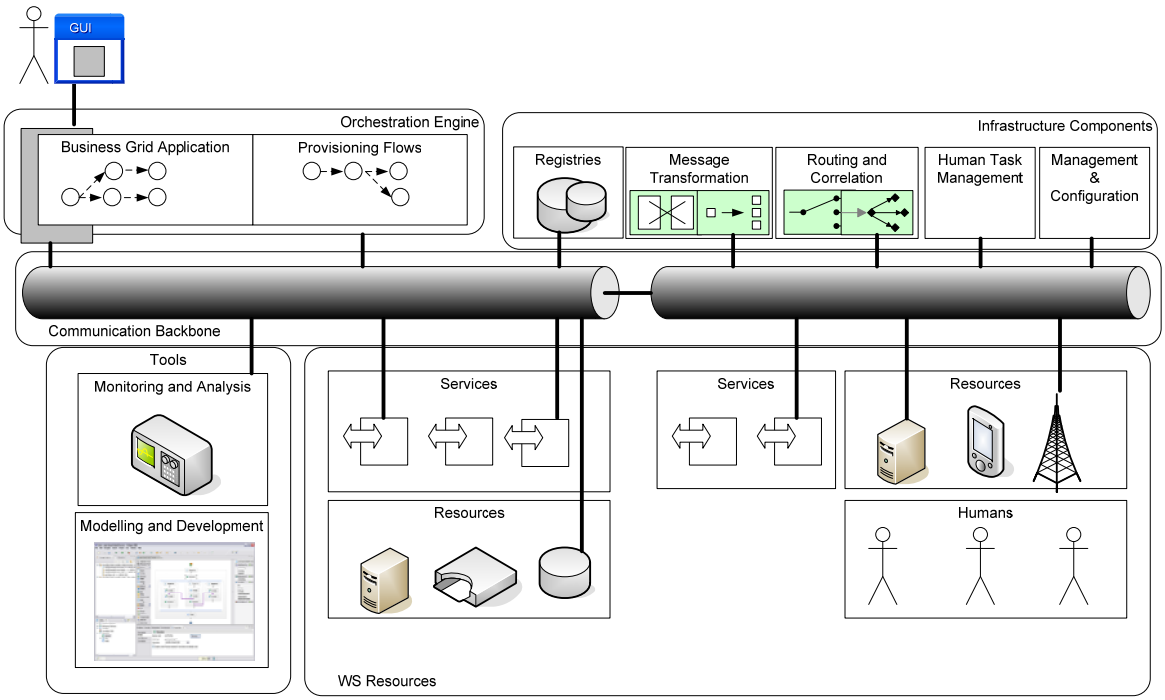

Figure 4. Business Grid architecture (Mietzner & Karastoyanova & Leymann, 2008).

One example that shows the need to combine techniques from both Grid and SOC is the recognized fact that Grid applications are not yet robust and reliable enough. The service middleware and the

workflow engines in an SOA environment provide these features, and since Grid services are Web services it is only natural to reuse the existing techniques for scientific applications that use Grid services. The Business Grid is designed with the idea to reuse as much as possible from the available techniques, one of which is the ability to maintain robustness and reliability of applications using the approaches for flexibility of service compositions.
The ability to react to faults in the environment is flexibility. The faults to which the applications must remain resistant are service faults, failures in the middleware services like discovery and selection, faults in the execution of service compositions either due some unforeseen exceptional situations, or because extensive testing has recommended to exchange services, and others. The assumption is that the applications in a service-oriented environment are service compositions, while the mostly employed approach is the one known from workflow management. Flexibility is needed in order to make business and scientific applications robust even when Grid resources/services are used. We would like to stress once again that only by combining advantageous features from both areas can the Business Grid infrastructure become an enterprise strength one. In the next section we present a classification of approaches for flexibility of service compositions and describe several existing techniques that can be employed on the Business Grid.

4. APPROACHES TO FLEXIBILITY OF BUSINESS GRID APPLICATIONS

Workflows have been used in the last decades for implementing reliable and robust applications. Due to their flexible programming model for separating control logic from the actual functionality to compose, workflows are forward and backward recoverable and can be guaranteed to complete in finite time successfully. Note that a completion of a process, where all the effects of all performed tasks are reversed is also considered successful; success is defined by the business process and the application domain. Parallel execution of tasks, which do not exhibit dependencies, is possible using workflows. Furthermore, workflows are by design capable of performing in heterogeneous environments, especially because the make extensive use of middleware. Note again that multiple instances of the same workflow can be executed, which saves enormous amount of time and human resources. This is identified as a need in scientific computing, where currently only a single run of a computation is performed at a time. Logical units of work can be defined and executed with all-or-nothing semantics – a feature that can be traced back in the history of production workflows (Leymann & Roller, 1999) that evolved from transactional workflows and research in transaction processing.

4.1 Classification of Approaches to Process Flexibility

Even though workflows are a very flexible model for creating applications by predefining a process model optimized for a domain, in some scenarios the process model needs to be adjusted to some unexpected situations, called also changes in the environment. The ability of processes to adapt to the changes in the environment is called flexibility. Adjustments in processes, called also adaptations or changes, can be made for one or more process instances during their execution, or the process model itself may need to be adjusted if the adaptations are relevant for all process instances. Another reason for the necessity of adaptation of processes or instances is the continuous process of business process re-engineering, which targets the continuous improvement and optimization of processes run by enterprises. All process adaptation approaches are applicable for enabling flexibility of service compositions; some additional approaches however are needed because of features inherent for service compositions. These different types of changes are classified in (Karastoyanova, 2006) and the summarized in the following table:

Table 1. Classification of adaptation approaches for service orchestrations.

Life cycle phase	*Modification level*	*Approach*
Design time	Model	Any kind of change possible
	Instance	Flexibility by configuration
Run time	Model	Change in: • functions (port types, operations) • control flow
	Instance	Change in: • functions (port type, operations) • control flow • WS ports or instances

During the design time phase of service compositions, the phase in which the process specifications/definitions are created, any kind of changes in the models can be made. Some of the changes done include the definition of alternative execution paths in the control flow to accommodate different potential situations to which each instance must react as predefined in the control flow; the instance runs along only that alternative path that tackles the situation valid for that instance – this approach is known by the name "flexibility by configuration" (van der Aalst & Jablonski, 2000). Adaptation of service compositions during their execution, or run time, is complex and requires an infrastructure with special features. Whenever a process model is changed the instances that will be created after this change will follow the new model.
For the running process instances first it has to be checked if these instances can at all be modified to reflect the new model depending on their current status/position in the execution of the process. The instances that can be modified according to the new model are said to be migrated to this model. This may require changing the internal workflow engine representation of the instances and the data they use. Additionally, the instanced that cannot be migrated may either be terminated or let complete as specified in the original model. This means that one and the same model would have two versions for a considerable period of time and version management is one of the implied infrastructure features. Instance migration and the supporting infrastructure are complex areas of current research and are out of the scope of this work. In service compositions only the fixed dimensions of the specification may need to be changed as reaction to environment changes. The control flow specification/model can be adapted by including new tasks or groups of tasks or deleting tasks. Additionally, transition conditions and data can be changed. The functions implementing these tasks may also be changed. All these changes can be performed for both models and one or more instances. Service composition instances may also incorporate a reaction to changes in the environment, e.g. failing WS, the appearance of a new service with better QoS characteristics, adaptation request due to recommendation of a test run (Gehlert & Hielscher & Danylevych & Karastoyanova, 2008), by exchanging a concrete WS implementation that has originally been chosen to perform a task in an instance (Karastoyanova & Houspanossian & Cilia & Leymann & Buchmann, 2005). Approaches enabling all of these types of changes in service compositions already exist and the next section represents an overview of the most advantageous of them.

4.2 Existing Approaches towards Flexibility of Service Compositions

This section provides an overview of existing approaches towards flexibility of service compositions and assigns them to the concrete group of approaches in the classification presented above.
Consider the following observations (see Figure 5):

- Control logic dimension is fixed in the process models and definitions
- Port types and operations of partner WSs are hard-coded in WS composition models, i.e. the functions dimension is fixed
- A single port type name may identify a simple or a complex functionality
- Concrete participants are typically resolved upon deployment

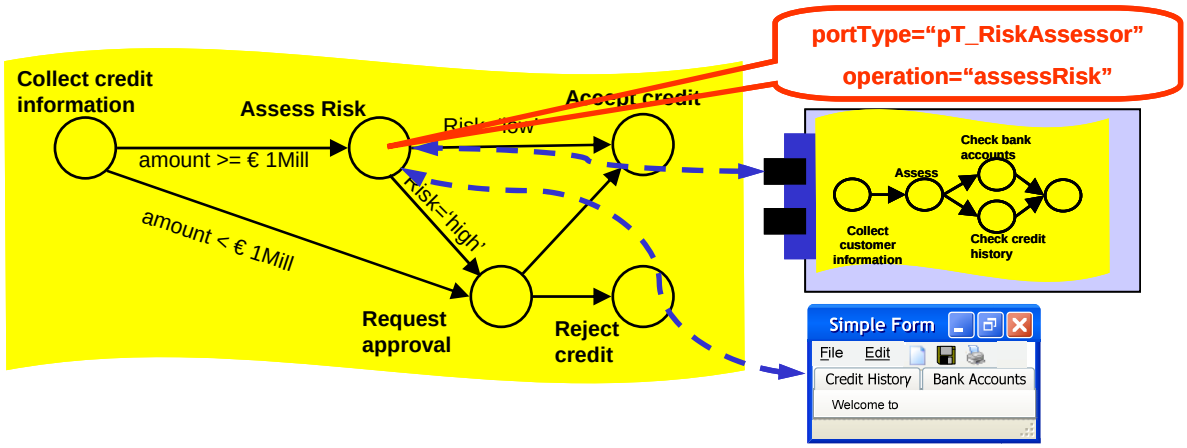

Figure 5. Service composition specifications reference only port types; the concrete implementations may differ for each of the instances of the same task in a process model.

Since service compositions reference only port types of services, there must be a mechanism to resolve these port types to concrete service endpoints/ports. The mechanism is called *service binding* (see Figure 6A) and there are several strategies defined in literature (Weerawarana & Curbera & Leymann & Storey & Ferguson, 2005) for controlling this mechanism, called *binding strategies*. The static binding strategy prescribes concrete service ports for each of the tasks in a process during either design time or deployment time; the statically prescribed service ports are used during process execution. Dynamic service binding is a strategy that postulates that the concrete services are to be discovered during process instance execution. It is enabled during both design time and run time. During modelling or deployment time the port types of services that implement a task are specified; these are the functional requirements toward a service. Additionally, the QoS selection criteria, usually provided in terms of WS-Policies, may also be provided. During process execution the process engine delegates the discovery and selection of the concrete services for each of the tasks to the service bus, which uses as input the functional and non-functional properties as provided by the process specification. The dynamic binding is an approach used to enable flexibility for each process instance with respect to the functional dimension of service composition.

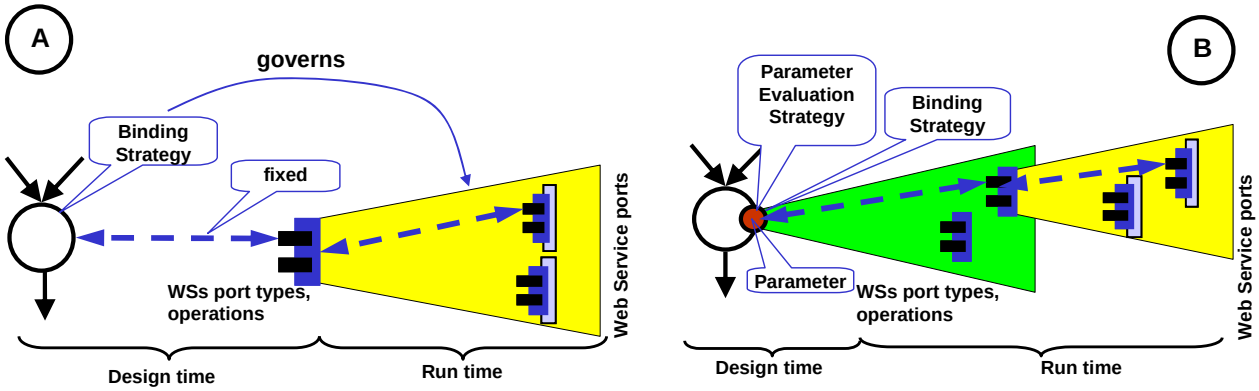

Figure 6. (A) The role of binding strategies in service compositions. (B) Improving flexibility using parameterization.

4.2.1. Exchanging Concrete Service Implementations during Run Time

The mechanism for discovery, selection and invocation of WS is known by the name *find and bind mechanism* and is utilized during the normal execution of service compositions. It can also be used to exchange services that have been bound for use with a service composition, regardless of the strategy used for the original binding, with another port. This may be necessary in order to handle a fault of a service and the inability of the service middleware to discover a new alternative but compliant service. For the discovery of the new service, the selection criteria must be altered to allow for a discovery at all. Therefore, to enable this, the alternative selection criteria must be provided in the original process

model. There are several ways to do this: (a) extend the process definition language to allow for specification of such criteria directly in the interaction activity elements; (b) extend the deployment information by alternative selection policies and use it during execution for the discovery of a substitute for a failed service, which will provide a much more configurable approach.
Approach (a) has already been defined and implemented and detailed report on it can be found in (Karastoyanova & Houspanossian & Cilia & Leymann & Buchmann, 2005). The alternative selection criteria can be provided either via an extension element to the BPEL <invoke> activity automatically, where the alternative policy is available, or manually, where a user provides the alternative selection criteria for the extended <invoke> activity in which the attribute modelling the alternative criteria is left empty. To the best of our knowledge, this approach has not been implemented yet.

4.2.2. Parameterized Processes

One approach to make service compositions flexible on the functions dimension in their reaction to changes is enabled by the parameterized processes (Karastoyanova et al., 2006). The approach extends BPEL and provides an extension element in the interaction activities to allow for providing or discovering a port type for the service to be bound using a parameter evaluation strategy (see Figure 6B). Such a strategy is specified during design time but is executed during run time to resolve parameter values. The parameter values may be provided manually by a human participant, or may be provided as a result from a discovery according to service criteria specified in terms of semantic description of services in any of the existing semantic Web services frameworks, or may be copied from a variable. The approach implies particular characteristics of the execution environment since it extends the BPEL core model and hence is not supported by the standard BPEL engines. Additionally, it also requires the integration with an execution environment for semantic Web services (Karastoyanova et al., 2006), (Karastoyanova, 2006), the so-called semantic service bus (Karastoyanova et al., 2007). The use of the approach implies also the use of the dynamic binding strategy, in order to discover the concrete implementation of the discovered port type for each instance of the task, too (see Figure 6B). This approach provides a primitive support for control flow change in service composition since by discovering or specifying a port type and following one of the above observations that a port type may also be implemented by a process, the control flow of the overall service composition is changed.

4.2.3. BPEL^{light}

BPEL^{light} (Nitzsche & van Lessen & Karastoyanova & Leymann, 2008) is an extension to BPEL that decouples the process logic from the WSDL descriptions and enable the use of any kind of service interface descriptions. This approach has been implemented prototypically and described in (van Lessen at al., 2007). The BPEL^{light} specification defines an extension activity which separates completely the process logic from the descriptions of services that allows for associating any kind of service description to the process definitions. During process deployment, the service descriptions are simply supplied together with the process logic. This approach removes the hard-coding to service interfaces into the process models and facilitates the adaptability and reusability of service composition definitions. It is a standard compliant extension of BPEL and can be executed on BPEL compliant engines with the respective extensions. The prototype supports the use of conventional WSs and semantic Web services described in WSMO. For this the BPEL4SWS specification has also been produced and the corresponding engine implementation provided (van Lessen at al., 2007). BPEL4SWS specifies how BPEL^{light} is used in combination with semantic Web services. Once the service type is discovered based on its semantic description, the discovery of service ports/endpoint is a must. Any other semantic service description framework, like OWL-S and SAWSDL, can also be applied with BPEL4SWS. Through the use of semantics BPEL4SWS improves the discovery of services (port types and ports) through an additional level of abstraction provided by the semantic description of interfaces. This improves further flexibility and reusability of BPEL processes. The approach supports flexibility of the functions dimension of WS compositions and may in some cases enable control flow adaptations similar to the way it is enabled by parameterized processes.

4.2.4. BPEL'n'Aspects

The BPEL'n'Aspects approach for flexibility (Karastoyanova & Leymann, 2009) draws on the ideas of the aspect-oriented programming (AOP) paradigm. The approach enables control flow adaptation of running instances of a process model. It is a non-intrusive approach in the spirit of the AOP paradigm; this means that the original process models are kept unchanged, but using the approach the actual execution of each of the instances of such models may be adapted as needed.
The analogy used to enable the approach follows the AOP paradigm which postulates that upon events of a specified type additional functionality, which has not been part of the original code, can be executed or weaved into the control flow of the original program. We observe a similarity to event notification and reaction to events used in reactive database systems and complex event processing systems.
Following this analogy we treat the BPEL processes the original programs that are executed on BPEL engines; the BPEL engines interpret the BPEL process models and publish navigation events. WSs implement functionalities and can be treated as the functionalities to be weaved in upon a navigation event happening, hence WSs are the advices in AOP terms (see Figure 7). Aspects are specified in terms of WS-Policy (W3C) and aspect references any Web Service (advice) that will represent the additional functionality.

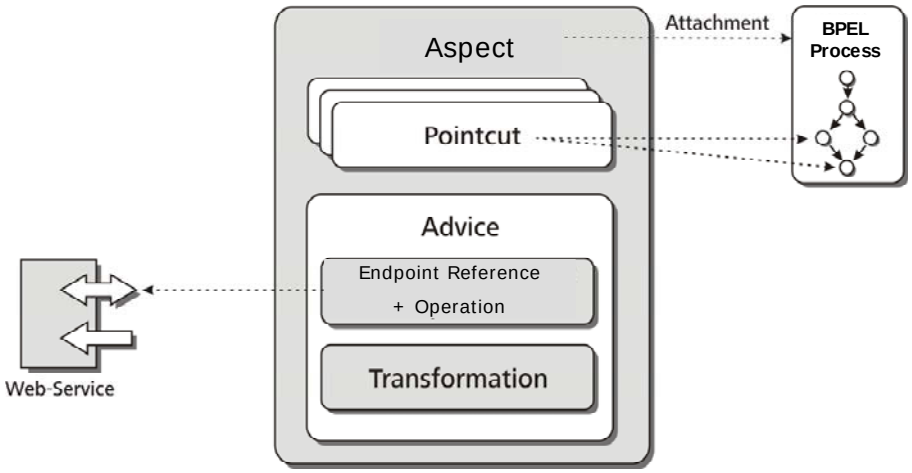

Figure 7. Aspects in the BPEL'n'Aspects approach.

An aspect also specifies upon which event in the process navigation the WS should be executed by means of the so-called pointcut and advice type (before, after and instead). For example a policy, i.e. an aspect, referencing the "Store Debit" operation can be assigned to a process instance of the trip booking process and be executed after the completion of the activity/task "Charge Credit Card" (see Figure 8). Similarly, aspects may be defined to incorporate additional functionality in the process flow before or instead a task, to modify the value of variables or transition conditions on links.
The aspects/WS-Policies are attached to the process models and/or process instances using WS-PolicyAttachment (see Figure 8). This facilitates the better configurability of the approach and allows for combining any aspect definition with any process model.

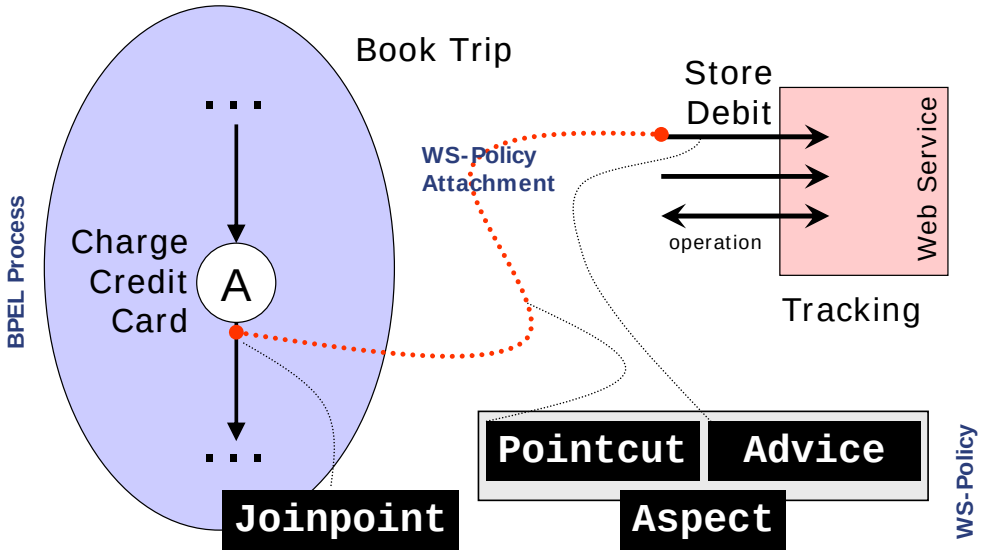

Figure 8. BPEL'n'Aspects approach – example.

The implementation of the approach follows the architecture presented in Figure 9. A BPEL engine that is capable of publishing navigation events is needed. The so called custom controller (Khalaf & Karastoyanova & Leymann, 2007) implements the functionality of the weaver. It is responsible of including the additional functionality for each process instance as defined by the aspects attached to that process instance. The weaver lets only the navigation events for which there is an aspect deployed to be notified. Upon such a notification the WS that constitutes the advice of the corresponding aspect is invoked. This communication (between the process instances and the WSs/advices) is based on the publish/subscribe paradigm and in this particular case uses WS-Notification (WS-N, 2004).

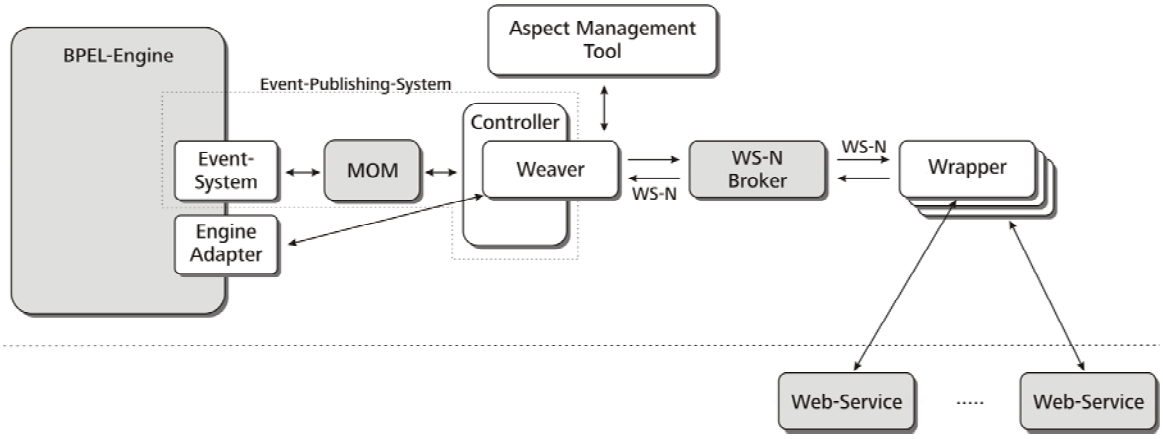

Figure 9. Architecture of the infrastructure implementing the BPEL'n'Aspects approach

The way to proceed when using the approach is the following:

- Define and deploy the aspect on the supporting infrastructure using the Aspect design and deployment tool (Figure 9)
- Upon subscription the Weaver component subscribes to the navigation event (published by the engine) specified in the pointcut of the aspect, e.g. to the event notifying that the activity "Charge Credit Card" is finished. Additionally, a wrapper is created for the invocation of the WS referenced into the deployed aspect. The wrapper is the component that handles the invocation of the WS/advice.
- Upon this event the Weaver publishes it, and thus the corresponding wrapper is notified (via a predefined topic) and the WS in the aspect is executed, e.g. the WS implementing the operation "Store Debit". The result produced by the WS invocation is returned to the process instance. The effect of the aspect weaving is the inclusion of a new task (i.e. the "Store Debit" task) into the process logic.

The advantages of the BPEL'n'Aspects approach include improvements in modularity and configurability of service compositions, maintaining the standardization endeavour of the WS technology as a whole, and support for both model evolution and ad-hoc changes in service compositions in a non-intrusive manner. The non-intrusiveness is a very valuable characteristic, since it allows for adapting/extending even existing and running BPEL processes without the necessity to change and redeploy them. Thus the approach tackles a major drawback of other approaches to flexibility presented in this and other works; for further references please consult (Karastoyanova, 2006).

4.3. Assessment of the Flexibility Approaches

The approaches to flexibility of service compositions presented in this chapter enable different types of changes as classified in Table 1. The exchange of concrete services/ports during the execution of service compositions allows for adaptations on the level of functions in processes on per-instance basis. The functions dimension can be modified with respect to the port types used on both models and instances using the approach of process parameterization. This approach can in some cases enable modifications of the control flow of orchestrations. The BPEL'n'Aspects approach has been designed with the purpose to support adaptation of control flows of service compositions and has been demonstrated as a feasible mechanism for adapting BPEL processes. It can also be used to render the functions dimension flexible; the aspects one defines for this must be of the "instead" type, which states that the original task is substituted with another one.

In the scenarios where existing running processes need to be adapted, the best approach to use is BPEL'n'Aspects, since it utilizes existing technology and standard specifications and does not alter the process definition language. It however requires extending the execution environment but all the needed extensions leverage existing very well known technologies and concepts like the WS stack and publish/subscribe.

The rest of the approaches are based on changes in the process definition language and hence imply more changes to the execution engine; on the other hand, tool support and infrastructure for them also exists which facilitates employing them as is.

When choosing an approach for improving the flexibility of service compositions we advise evaluating thoroughly the concrete types of changes needed in the concrete application domain and scenario and use the result of the evaluation during the decision-making process.

5. OPEN ISSUES IN THE FIELD OF THE BUSINESS GRID

The section summarizes some of the open research issues that need to be considered in order to combine the efforts of the SOC and Grid communities and thus supply the foundations of an infrastructure applicable for both business and scientific applications.

Handling huge amounts of data produced and consumed by both scientific and business applications in an SOA is still an open topic. Research done in the field of database management systems, data warehousing and the simulation technology should be leveraged to produce fast results in a standard based manner reusing as much of the existing technology as possible.

Fault handling in business and scientific applications, especially in a domain specific manner, requires not only standardization of the business processes in each of the domains, but also utilizing the advantages of the existing technologies. For example, scientific applications on the Grid are encouraged to take advantage of the workflow technology to enable not only parallel or alternative task execution, but also utilize the inherent backward- and forward-recoverability of workflows supported by workflow management systems. Workflows in combination with SOA enable the creation of fault-tolerant scientific applications in a heterogeneous environment, which directly matches the requirements of scientific computing landscapes.

The available provisioning techniques on the Grid can be utilized to provision enough resources for the execution of business applications in the cases of high workloads and in support of the new application delivery models like Software as a Service (SaaS), Infrastructure as a Service (IaaS), Middleware as a Service (MaaS) etc.

In this relation, it is of utmost importance to design and expose computations as Grid services in a unified manner; note that currently most scientific computations and algorithms are monolithic applications and incorporate features like security, data handling, integration and others, which are orthogonal to the core concern of the applications.
The approached for improving flexibility of applications/compositions and hence their robustness may need to be tuned as to address domain specific requirements. For example, the requirements for robustness in simulation (SimTech), (Taylor et al., 2003) might be different from those in computer linguistics (Clarin, 2008). These requirements still need to be investigated and analysed.
The integration of Grid infrastructures and Bus implementations is not seamlessly possible, since they do not comply with the same general architecture; the introduction of the Business Grid vision provides the necessary framework for this and endeavours to facilitate convergence of the SOC and Grid communities, and the research done by them. Tool support is an obvious must and can be provided once the architecture of a unified architecture is agreed upon. Such support depends on the existence of Domain Specific Languages, and the corresponding extensions to BPEL or other process definition languages. Support for human interaction is still a research issue in SOC and the potential solutions need to take into account the requirements from the scientific domains, too. Reusability of process models and fragments is typically one of the major requirements in both domains and needs to be taken care of. This implies incorporating support for reusability in the modelling tools and the design of process model libraries, including support for debugging and testing. This should all lower the barrier to entry for users.
The appropriate composable execution infrastructure must be put into place. It needs to ensure concurrency, reliability of the middleware and processes running on it, while reusing of existing approaches to ensure leverage of investment in technology and skills.
The performance of this infrastructure must meet the requirements of both domains, and the corresponding performance models, test procedures, evaluation and monitoring infrastructure must be put in place too.

6. SUMMARY AND CONCLUSIONS

In this work we discussed the potential benefits of combining the SOA and Grid paradigms and the techniques available in these two domains. We are convinced that only the combination of the techniques from the two domains can support enterprise strength applications for business and scientific use. We observe that existing techniques from scientific computing based on Grid infrastructures can be exploited in business applications. Additionally, mechanisms available on the service bus and known from service compositions can be used to address the lack of reliability and robustness in scientific applications on the Grid. This observation is the motivation for introducing a combined infrastructure, called the Business Grid, which exhibits beneficial features applicable in both domains. In this paper we have concentrated on one example of how Grid and SOC techniques can be combined to improve the state-of-the-art. In particular, we have presented approaches for enabling flexibility of service compositions known from business driven research and that they can be applied for scientific Grid applications to tackle reliability and robustness problems under the assumption that Grid services are utilized. We have identified some further synergy opportunities between the business and scientific application domains and argued that the appropriate combination and thorough exploitation of existing approaches can improve the state of the art enormously and grant benefits to the SOC and Grid communities altogether.

REFERENCES

1. van der Aalst, W.M.P., Jablonski, S. (2000) Dealing with workflow change: identification of issues and solutions. *International Journal of Computer Systems Science and Engineering, Vol. 15, No. 5* CRL Publishing Ltd.
2. van der Aalst, W., van Hee, K. (2002) *Workflow Management. Model, Methods and Systems. MIT Press.*
3. Chappell, D. (2004). *Enterprise Service Bus.* O'Reilly Media, Inc.
4. CLARIN Project. (2008). http://www.clarin.eu/

5. Czajkowski K. et al. (2004). From Open Grid Services Infrastructure to WS-Resource Framework: Refactoring & Evolution. Global Grid Forum Draft Recommendation.
6. Emmerich W., Butchart B., Chen L., Wassermann B., Price S. (2005). Grid Service Orchestration Using the Business Process Execution Language (BPEL), *Journal of Grid Computing, vol. 3*, pp. 283-304.
7. Foster, I. (2002). What is the Grid? - a three point checklist. *GRIDtoday, vol. 1*, http://www.Gridtoday.com/02/0722/100136.html.
8. Foster, I., Kesselman, C., und Tuecke, S. (2001). The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *Int. J. High Perform. Comput. Appl., vol. 15* 2001, pp. 200-222.
9. Foster, I., Kesselman, C. (2004). *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers.
10. Foster I. et al. (2005) The Open Grid Services Architecture, Version 1.0. Global Grid Forum, Lemont, Illinois, USA, GFD-I, vol. 30.
11. Fox G. C., Gannon D. (2006). Workflow in Grid Systems. *Concurrency and Computation: Practice and Experience, vol. 18,* pp. 1009-1019.
12. Gannon, D. (2007) A Service Architecture for eScience Grid Gateways. Grid Computing, High-Performance and Distributed Applications (GADA'07).
13. Gehlert, A., Hielscher, J., Danylevych, O., Karastoyanova, D. (2008). Online Testing, Requirements Engineering and Service Faults as Drivers for Adapting Service Compositions. In Proceedings of MONA+ at ServiceWave2008.
14. Kaye, D. (2003). *Loosely Coupled: The Missing Pieces of Web services*. RDS Press.
15. Karastoyanova, D., Leymann, F. (2009) BPEL'n'Aspects: Adapting Service Orchestration Logic. In Proceedings of ICWS 2009.
16. Karastoyanova, D., Lessen, T. Van, Nitzsche, J., Wetzstein, B., Wutke, D., Leymann, F. (2007). Semantic Service Bus: Architecture and Implementation of a Next Generation Middleware. In Proceedings of the 2nd International Workshop on Services Engineering (SEIW) 2007, in conjunction with ICDE 2007.
17. Karastoyanova, D. (2006). *Enhancing Flexibility and Reusability of Web Service Flows through Parameterization. PhD Thesis*. TU-Darmstadt, Shaker Verlag.
18. Karastoyanova, D., et al. (2006). Parameterized BPEL Processes: Concepts and Implementation. Proc. of BPM 2006.
19. Karastoyanova, D., Houspanossian, A., Cilia, M., Leymann, F., Buchmann, A. (2005). Extending BPEL for Run Time Adaptability. In Proceedings of EDOC 2005.
20. Keller, A., Badonnel, R. (2004) Automating the Provisioning of Application Services with the BPEL4WS Workflow Language. Proceedings of the 15th IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM 2004). LNCS 3278, Springer Verlag
21. R. Khalaf, D. Karastoyanova, F. Leymann, Pluggable Framework for Enabling the Execution of Extended BPEL Behavior. (2007) Proceedings of the 3rd ICSOC International Workshop on Engineering Service-Oriented Application: Analysis, Design and Composition (WESOA 2007) Springer.
22. Lessen, T. van; Nitzsche, J.; Dimitrov, M; Konstantinov, M; Karastoyanova, D.; Cekov, L. (2007). An Execution Engine for Semantic Business Process. 2nd International Workshop on Business Oriented Aspects concerning Semantics and Methodologies in Service-oriented Computing (SeMSoC), in conjunction with ICSOC 2007.
23. Leymann F. (2005). The (Service) Bus: Services Penetrate Everyday Life. In Proceedings of ICSOC'2005, Amsterdam, LNCS 3826 Springer-Verlag.
24. Leymann, F. (2006). Choreography for the Grid: towards fitting BPEL to the Resource Framework: Research Articles. *Journal of Concurrency and Computation: Practice & Experience, Volume 18*, pp. 1201-1217.
25. Leymann, F., Roller, D. (1999). *Production Workflow: Concepts and Techniques.* Prentice Hall PTR.
26. Mietzner, R., Karastoyanova, D., Leymann, F. (2009). Business Grid: Combining Web services and the Grid. *In Special Issue of ToPNoC on Concurrency in Process-Aware Information Systems*. Springer. (to appear)

27. Mietzner, R., Leymann, F. (2008). Towards Provisioning the Cloud: On the Usage of Multi-Granularity Flows and Services to Realize a Unified Provisioning Infrastructure for SaaS Applications. In Proceedings of the International Congress on Services, SERVICES 2008.
28. Nitzsche, J., Lessen, T. van, Karastoyanova, D., Leymann, F. (2008). BPEL^{light}. 5th International Conference on Business Process Management (BPM 2007).
29. OASIS BPEL Technical Committee. (2008). http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel
30. OASIS WS-BPEL Extension for People (BPEL4People) Technical Committee. (2008). http://www.oasis-open.org/committees/bpel4people/charter.php
31. OASIS Web services Resource Framework (WSRF) TC. (2008) http://www.oasis-open.org/committees/documents.php?wg_abbrev=wsrf
32. OpenQRM. (2008). http://www.openqrm.org/
33. Slomiski, A. (2006). On using BPEL extensibility to implement OGSI and WSRF Grid workflows. *Concurrency and Computation: Practice & Experience, vol. 18*, pp. 1229-1241.
34. Taylor, I. J. et al. (2006). *Workflows for e-Science: Scientific Workflows for Grids*. Springer.
35. Tuecke S. et al. (2003). Open Grid Services Infrastructure (OGSI) Version 1.0, Global Grid Forum Draft Recommendation.
36. SimTech: Stuttgart Research Centre for Simulation Technology at the University of Stuttgart. http://www.simtech.uni-stuttgart.de/
37. W3C, W3C Note: Web services Description Language (WSDL) 1.1, http://www.w3.org/TR/wsdl, 2001
38. Web services Notification. (2004) http://www-106.ibm.com/developerworks/library/specification/ws-notification/
39. W3C. W3C Member Submission: Web services Policy Framework. http://www.w3.org/Submission/WS-Policy/
40. Weerawarana S., Curbera F., Leymann F., Storey T., Ferguson D.F. (2005). *Web services Platform Architecture.* Prentice Hall.