Bis-Grid: Business Workflows for the Grid*

S. Gudenkauf[1], W. Hasselbring[2], F. Heine[3], A. Hoeing[3], G. Scherp[1], O. Kao[3]

[1] OFFIS Institute for Information Technology, R&D-Division Business Information Management, Escherweg 2, 26121 Oldenburg, Germany
email: [stefan.gudenkauf, guido.scherp]@offis.de
[2] Univ. Oldenburg, Software Engineering Group, PO Box 2503, D-26111 Oldenburg, Germany
email: hasselbring@informatik.uni-oldenburg.de
[3] TU Berlin, Faculty IV - Electrical Engineering and Computer Science, Dept. of Telecommunication Systems, Complex and Distributed IT Systems, Einsteinufer 17, 10587 Berlin, Germany
email: [felix.heine, andre.hoeing, odej.kao]@tu-berlin.de

Abstract

The BIS-Grid project, a BMBF-funded project in the context of the German D-Grid initiative, focusses on realising Enterprise Application Integration using Grid technologies, enabling small and medium enterprises both to integrate heterogeneous business information systems and to use external D-Grid resources and services with affordable effort. The goal is to provide proof of concept that Grid technologies are feasible for information systems integration, and to enable business users to utilise and participate in the D-Grid, facilitating a more sustainable Grid infrastructure in Germany. One technical part of BIS-Grid is to provide a WS-BPEL-based workflow engine that is based upon the Grid Middleware UNICORE 6, and that is capable to integrate stateful Web Services, so-called Grid Services. In this paper we present a generic extension to UNICORE 6 that allows to use an arbitrary WS-BPEL engine, in our scenario the WS-BPEL engine ActiveBPEL, and standard WS-BPEL to orchestrate stateful Web Services in a transparent and scalable manner.

1 Introduction

Enterprise Application Integration (EAI) has become a well-established way to integrate heterogeneous business information systems. This provides a basis to map business processes considered as workflows to the technical system level. Often, this is accomplished via service orchestration in service-oriented architectures (SOA). By doing so, Web Services provide means to enable service orchestration as well as to hide the underlying infrastructure. Modern Grid technologies such as the Grid middlewares UNICORE 6[1] and Globus Toolkit 4[2] are based on the Web Service Resource Framework (WSRF) [1], a standard

*This work is supported by the German Federal Ministry of Education and Research (BMBF) under grant No. 01IG07005 as part of the D-Grid initiative.

[1] http://www.unicore.eu

[2] http://www.globus.org/toolkit/

that extends classical, stateless Web Services to be stateful in a standard manner. Such WSRF-based Web Services, also called Grid Services, provide a basis to build SOA using Grid technologies. Thus, Grid technologies and EAI have much in common since both technologies focus on integration problems within a heterogeneous environment, Grid technologies on resource level and EAI on application level. In BIS-Grid[3], a BMBF-funded project in the context of the German D-Grid[4] initiative, we intend to enable Grid technologies to be used for the integration of decentralised business information systems. The project especially addresses small and medium enterprises (SMEs). BIS-Grid will enable these enterprises to design and run workflows realised as Grid Service orchestrations to develop and provide dynamic solutions for arising business challenges.

Within BIS-Grid the application scenarios are motivated by our industrial partners: CeWe Color, the number one services partner for first-class trade brands on the European photographic market supplying both stores and internet retailers with photographic products, and KIESELSTEIN Group, the global market leader in the field of wire drawing and draw-peeling for the automotive industry. Both CeWe Color and KIESELSTEIN have needs for Enterprise Application Integration which they are willing to realise with Grid technologies: CeWe Color to integrate enterprise data for unified access for call centre agents, and KIESELSTEIN Group to improve access to, and retrieval and maintenance of product and project data. Thereby, service orchestrations have to be provided unified and with hard constraints to service quality. The goals of the two application scenarios are to investigate the feasibility and expenses of Grid-based EAI, and to investigate additional benefits on applying Grid technologies. Enterprises shall be enabled to switch between enterprise-specific EAI and Grid utilisation dynamically. While in the first case resources and application providing are both located within the enterprise (inhouse providing), Grid-enabled solutions allow application providing and resources to be outsourced to the Grid (Grid application providing). In this way, it becomes possible to integrate external Grid Services in enterprise-specific workflows.

In Section 2 we present related work that is relevant to the development of a workflow engine capable of orchestrating Grid Services. Then, in Section 3, we present the architecture of the BIS-Grid workflow engine, focussing on load balancing and security. Finally, in Section 4, we conclude the paper and briefly present our future work.

2 Related Work

Orchestrating stateful services is in the focus of several papers. Leymann [5] describes the appropriateness of using BPEL4WS as a basis for Grid Service orchestration since it already fulfils many requirements of the WSRF standard. He concludes that a Grid-specific extension of BPEL4WS is more appropriate than creating new Grid-specific standards. The appropriateness of BPEL4WS

[3] http://www.bisgrid.de

[4] http://www.d-grid.de

for Grid Service orchestration is also confirmed in [4], [9], and [2]. Inter alia, Emmerich et al. [4] describe the evaluation of reliability, performance, and scalability issues of the open source workflow engine ActiveBPEL on executing a complex scientific workflow. In [3], Dörnemann et al. discuss composing Grid Services by using BPEL4WS. They present a solution that is based on extending the BPEL4WS specification. Amongst other things, Dörnemann et al. see the seamless integration of security mechanisms, for example WS-SecureConversation [7] as well as Virtual Organisation management in the BPEL engine as a topic of further research. Orchestrating stateful services is in the focus of many German and international projects. For example, the German D-Grid projects TextGrid[5] and InGrid[6], and the European projects A-WARE[7], Chemomentum[8], and EGEE[9]. These projects, as well as the previously described papers, mainly focus on scientific workflows instead of business workflows relevant to BIS-Grid. Concerning the use of BPEL (BPEL4WS or WS-BPEL), the described papers focus on extending or adapting the language, thus creating BPEL dialects. In BIS-Grid we instead focus on proper BPEL modelling by using existing language elements to orchestrate WSRF-based Grid Services[10]. We are in contact with some of the mentioned projects, for example A-WARE, to work together on common problems and to exploit other synergy effects.

3 Architectural Solution

One technical goal of the project is to provide a WS-BPEL-based workflow engine capable of integrating Grid Services. The current state of work regarding this engine is that the architecture is almost defined and the implementation start is scheduled for the first quarter of 2008. The engine is intended as a composition of the WS-BPEL engine ActiveBPEL[11] and UNICORE 6. Upon ActiveBPEL, a WSRF Grid wrapper based on UNICORE 6 enables workflows both to use Grid Services and to be offered as Grid Services as well, thereby providing compatibility to Grid middlewares, and to utilise additional functionalities such as role-based access control (RBAC) and monitoring capabilities. The technical issues relevant to Grid Services, in particular security, are completely managed by this wrapper, and are therefore transparent to ActiveBPEL. Since our extensions will only affect UNICORE 6, the underlying WS-BPEL engine is replaceable with any other WS-BPEL engine. Thus we expect to encounter only minor upgrade problems when switching to a different BPEL engine. The WSRF wrapper itself is implemented as a set of Grid Services. This decision is based on the following aspects:

[5] http://www.textgrid.de/
[6] http://www.ingrid-info.de/
[7] http://www.a-ware-project.eu/
[8] http://www.chemomentum.org
[9] http://www.eu-egee.org/
[10] We will create appropriate WS-BPEL patterns, for example to invoke Grid Services
[11] http://www.active-endpoints.com/active-bpel-engine-overview.htm

- The WS-BPEL specification states that WS-BPEL service orchestrations should be available as Web Services itself. Therefore the chosen workflow engine, ActiveBPEL, was written to run in a Web Service container. The BIS-Grid workflow engine per se has not necessarily to run in a service container instead, but project results have to be integrated as Grid Services in UNICORE 6. The UNICORE 6 WSRFLite environment provides a secure and fast basis for a seamless UNICORE 6 integration.
- Without UNICORE 6 integration, we would have to deal with challenges that are already solved. For example the UNICORE 6 security implementation supports a X.509 certificate-based security layer. A stand-alone solution would have to provide an own Grid Service-compliant security layer.

The BIS-Grid workflow engine consists of three main components: the *Workflow Deployment Service*, the *Workflow Factory Service*, and the *Workflow Service*. The Workflow Deployment Service provides means for workflow management such as deployment and undeployment. For each deployed workflow, there exists a Workflow Factory Service and a Workflow Service. The Workflow Factory Services are responsible for creating workflow instances for the corresponding Workflow Services. These services are placed in the UNICORE/X service container as Grid Services. To provide flexibility concerning the WS-BPEL engine, the WS-BPEL engine is placed as an own module in its own service container. The WS-BPEL engine manages the actual workflow execution while the BIS-Grid services encapsulate the engine and represent the WSRF Grid wrapper. The communication between the BIS-Grid services and the WS-BPEL engine is message-based using a secured HTTPS communication channel. To avoid unauthorised access, the engine only accepts connections from the UNICORE/X component, checked by its certificate. This is similar to the connection of UNICORE/X and the UNICORE 6 user database (XUUDB). Figure 1 shows the general architecture of the BIS-Grid workflow engine. Continuous arrows indicate communication channels between the elements of the architecture, and dashed arrows emphasise internal functionalities. The dotted rectangles represent WSDL interfaces. These interfaces provide access to the functional range of the BIS-Grid services, if appropriate permissions are granted to the service client.

The generic scenario is as follows: In cooperation with a *process designer* a *business analyst* designs a workflow creating a deployment package which includes a WS-BPEL description of the workflow, ActiveBPEL-specific information, and additional information specific to the BIS-Grid services, and sends it to the Workflow Deployment Service. This service processes the given information and deploys the workflow in ActiveBPEL. If this deployment is successful the Workflow Deployment Service creates a new Workflow Service Factory for the deployed workflow, and a new Workflow Service. *Process owners* may now create instances of the Workflow Service by calling the *create* method of the Workflow Factory Service. The Workflow Factory Service then returns the endpoint reference of the new instance. Workflow execution is started by calling the *start*

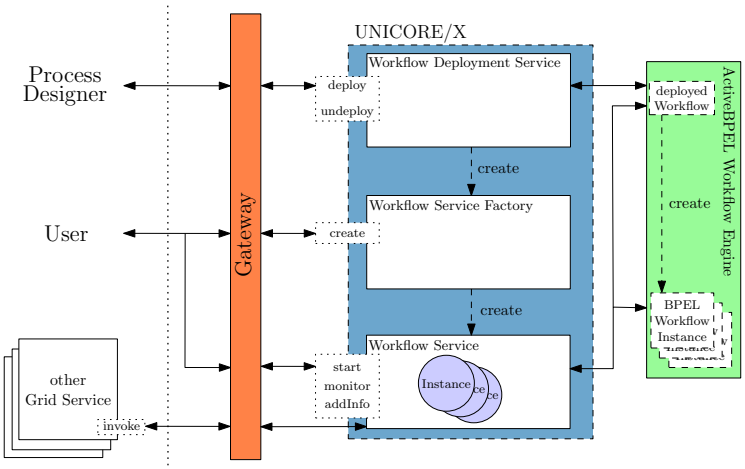

Fig. 1: Overview on the BIS-Grid architecture

method of the Workflow Service, which causes the WS-BPEL engine to create a new process instance (cp. [8]). During workflow execution, the BIS-Grid workflow engine is used as a two way proxy server. Grid-specific information, e.g. credentials, is added to outgoing messages and removed from incoming messages, whereas the latter must pass access permission checks. Additionally, accounting and billing information is evaluated. Workflow instances may be reconfigured as far as allowed. After execution, a *process owner* may explicitly destroy his Workflow Service instance or it destroys itself when its lifetime expires.

3.1 Load Balancing

Performance tests have shown that the UNICORE 6 environment is very fast if a simple test service is called[12]. Since the BIS-Grid services are complex latency may perhaps become too high. However, latency is a hard constraint in the business domain, especially in the CeWe Color call centre scenario. Beside UNICORE 6 and the BIS-Grid services, the ActiveBPEL engine may be the second bottleneck due to performance issues, for example if several hundreds of workflow instances are running concurrently. Therefore we kept the architecture flexible enough to deal with possible upcoming performance problems. The WS-BPEL engine runs in its own service container and the BIS-Grid services forward messages to the engine using secure HTTPS connections. If the WS-BPEL engine does not scale well, it is possible to move the engine from the frontend

[12]Emmerich et al. conducted performance test with ActiveBPEL, concluding that "the ActiveBPEL engine is unlikely to be a performance bottleneck [...]" [4]. We also performed simple performance tests on which the statement in the beginning of the section is based. However, documented performance tests are an open issue.

node to an exclusively reserved backend node. Furthermore, load balancing can be conducted by running several WS-BPEL engines on different backend nodes. This solution requires minor changes: workflows must be deployed on each WS-BPEL engine on the backend nodes, and the engines' locations have to be stored when workflow instance creation is scheduled to one of the engines. If UNICORE 6 is the bottleneck a more sophisticated solution is needed. If one or more engines on backend nodes process hundreds of workflow instances and workflow instances make invocations of other Grid Services the UNICORE 6 frontend node may turn out to be a bottleneck as well. We suppose checking security permissions and BIS-Grid-specific operations will cause the most time consumption. Our approach is to place BIS-Grid-specific balancing services on the frontend node which is registered at the UNICORE 6 gateway, and to install the UNICORE/X environment on several backend nodes. The balancing services have to provide the combination of all interfaces of backend nodes so that they are propagated by the gateway. Since it is unlikely that system administrators allow opening ports of backend nodes the instances of the Workflow Services must also be known to the frontend node, and messages must be dispatched by employing an appropriate mapping algorithm. Figure 2 depicts the load-balanced architecture. Each workflow will be deployed on every UNICORE/X backend node. Except from the Workflow Deployment Service other frontend BIS-Grid services do not provide access control, but dispatch incoming messages while access control is performed at the target backend nodes. The frontend's UNICORE/X additionally starts a new *Load Information Service* which collects load information of all backend nodes, enabling the frontend to calculate the less loaded machine for dispatching factory calls. The frontend node stores the returning endpoint reference and creates a workflow instance corresponding to the instance on the backend. Each call on this instance will be forwarded to the corresponding backend instance, and both instances will be destroyed simultaneously when their lifetime expires or their otherwise terminate. By shifting most security checks to backend nodes load is also transferred to them.

3.2 Security

Security is one of the most important challenges in BIS-Grid. To provide sustainability, enterprises must be convinced that BIS-Grid provides a highly secured infrastructure and that using Grid Services does not implicate a risk for their business. Beside data security the identities of the users and their rights must be checked strictly. An adequate authentication infrastructure and appropriate authorisation mechanisms must be introduced. In the following we present our Role-Based Access Control (RBAC) mechanism that clearly identifies a user and manages his permissions, leaving data security aside, although being an important topic in the project. Instead of directly mapping users to permissions, users are mapped to one or more roles, describing company membership, company department membership or a certain position. These roles are then mapped to permissions, for example to access resources. One benefit of RBAC is that managing permissions is reduced to assigning roles to a

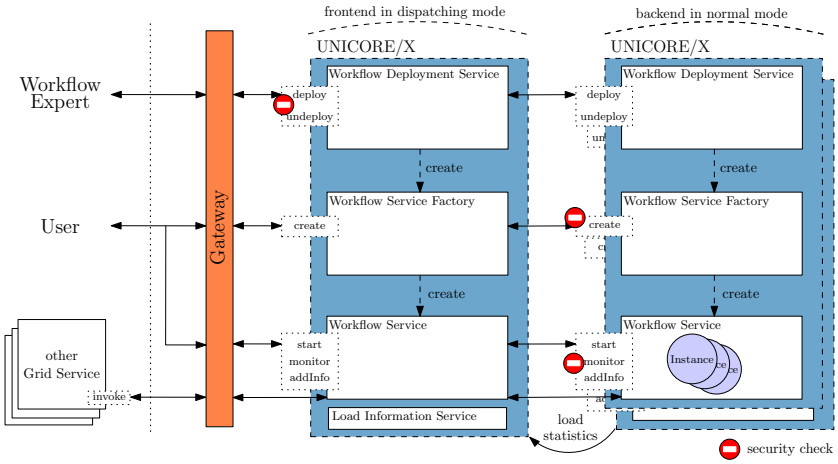

Fig. 2: BIS-Grid load balancing for UNICORE 6, backend nodes omitted

user, thereby facilitating common tasks such as changing a user's department or adding a new user. UNICORE 6 already provides means for authorisation. It can handle X.509 certificates and SAML assertion, which are used for credential delegation. Security is realised as handlers which can be placed in the ingoing handler chain of each Grid Service. Handlers use a common *Security Manager* which checks credentials against polices for the desired service operation. Permissions are described as XACML [6] policies and must be completely defined when starting UNICORE 6. However, when a new Workflow Factory Service and a new Workflow Service are created (hot deployment), a security policy would normally not be available for the services. Therefore the deployment descriptor of the workflow includes a default policy for the services, if the services on the whole or only selected operations of the services shall be access controlled. We will extend the UNICORE 6 security manager to allow policies to be instantiated when a new service is deployed without needing to restart UNICORE 6. Normally, once initialised, authorisation policies stay the same. But there may arise situations where policy changes are unavoidable, e.g. when the responsibility of a human activity shall be delegated to another user. An operation to change a policy is provided by the BIS-Grid services. This operation is secured by an XACML policy, too, which defines the changeable tags of a policy as its target. The changeable parts of a policy are well defined in the default policy. This strategy allows the recursive definition of policies and permissions to change the policies. The exchange of a default policy without redeploying the affected service is strictly forbidden.

4 Conclusion and Future Work

In this paper we presented the architecture of the BIS-Grid workflow engine as the integration of a standard WS-BPEL engine with the UNICORE 6 Grid middleware. It is shown how WS-BPEL-based service orchestration is extended to integrate Grid Services, and how load balancing and security issues are addressed. The key benefit of our solution is the transparent integration of a standard workflow engine, making it both exchangeable and scalable, and the utilisation of standard WS-BPEL in Grid Service orchestration instead of extending the language. Relying on UNICORE 6 and standard Grid technologies, we gain interoperability with other Grid solutions. Beside the refinement of the BIS-Grid engine's architecture described in the paper our future work focusses on additional issues that are relevant for business workflows. This includes accounting and billing, monitoring, and the modelling of human tasks in workflows to support human interaction within workflow enactment.

References

1. Tim Banks. Web Services Resource Framework (WSRF) - Primer v1.2. http://docs.oasis-open.org/wsrf/wsrf-primer-1.2-primer-cd-02.pdf, May 2006.
2. Kuo-Ming Chao, Muhammad Younas, Nathan Griffiths, Irfan Awan, Rachid Anane, and C-F Tsai. Analysis of Grid Service Composition with BPEL4WS. In *Proceedings of the 18th International Conference on Advanced Information Networking and Application (AINA'04)*, volume 01, page 284, Los Alamitos, CA, USA, 2004. IEEE Computer Society.
3. T. Dörnemann, T. Friese, S. Herdt, E. Juhnke, and B. Freisleben. Grid Workflow Modelling Using Grid-Specific BPEL Extensions. German e-Sience Conference 2007, May 2007.
4. Wolfgang Emmerich, Ben Butchard, Liang Chen, Sarah L. Price, and Bruno Wassermann. Grid Service Orchestration Using the Business Process Execution Language (BPEL). In *Journal of Grid Computing (2006)*, number 3, pages 283–304. Springer, 2006.
5. Frank Leymann. Choreography for the Grid: towards fitting BPEL to the resource framework: Research Articles. *Concurr. Comput. : Pract. Exper.*, 18(10):1201–1217, 2006.
6. Tim Moses. eXtensible Access Control Markup Language (XACML) Version 2.0. `http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf`, Februrary 2005.
7. OASIS Web Services Secure Exchange TC. WS-SecureConversation 1.3. http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-os.pdf, March 2007.
8. OASIS WSBPEL Technical Committee. Web Services Business Process Execution Language (WSBPEL) Primer. http://www.oasis-open.org/committees/download.php/23974/wsbpel-v2.0-primer.pdf, May 2007.
9. Aleksander Slomiski. On using BPEL extensibility to implement OGSI and WSRF Grid workflows: Research Articles. *Concurr. Comput. : Pract. Exper.*, 18(10):1229–1241, 2006.