BRINGING TOGETHER WHAT TOGETHER BELONGS
Applying Web Services to Couple SOA and Grid in Smaller Environments

Carsten Kleiner and Arne Koschel
University of Applied Sciences and Arts
Ricklinger Stadtweg 120, 30459 Hannover, Germany

Keywords: Grid computing, SOA, Web services, Case studies, SME.

Abstract: This paper describes practical experiences from a project to couple Grid and SOA technologies in smaller environments. Web services have been applied in two structurally different case studies to solve tasks with a Grid that is integrated into a SOA and vice versa. The case studies have revealed important insight on how and when to couple SOA and Grid technologies including monitoring aspects. Some interesting general rules are derived on what has to be observed when combining SOA and Grid in smaller environments. Performance and software technical analysis have been used in validating the results. They also clearly showed the benefits gained by employing SOA and Grid concepts form both a performance as well as an architectural perspective.

1 INTRODUCTION

1.1 Motivation

Today's IT systems are often comprised of quite different technical parts. A heterogeneous communication infrastructure is thus used to connect them. To exchange information between such systems XML has established itself as some lingua franca. Based on the XML fundament more specialized languages became popular for service description and communication between systems, A well known example is the combination of service interface description based on WSDL and abstracted service communication with SOAP, in this paper jointly called SOAP/Web Services (SOAP/WS).

No wonder, that SOAP/WS is examined for communication nowadays within enterprise software architectures as well. Especially enterprise IT tries to utilize SOA to become more flexible and to allow for better control. At the same time distributed, parallel problem processing across heterogeneous hardware became more and more popular recently; Grid technology, which couples such systems to become virtual super computers is here the means of choice. Since in both cases (SOA and Grid) communication is required, a common usage of XML-based SOAP/WS ((W3C, 2007), (Conrad et al., 2005), (Erl, 2005)), SOA ((Conrad et al., 2005), (Erl, 2005), (Krafzig, Banke and Slama, 2005)) and Grid ((Foster and Kesselman, 2003), (Conrad et al., 2005)) seems interesting for further investigation. Moreover this combination can be seen as an advanced project experience for students.

This experience report thus describes two case studies from student projects in the SOA/Grid space and presents experiences gained within them. While the first case study focuses on Grid computing technology for parallel problem solving with some smaller SOA part, the second study uses an SOA combined with a simulated data Grid. Both case studies commonly use SOAP/WS as communication base. Another requirement was the usage of Open Source Software only. For example the Grid technology Globus Toolkit (Foster, 2006) was used and the Nagios (Nagios, 2007) monitoring tool was extended to allow for initial SOA-Grid monitoring. Some performance and software technical analysis was performed as well.

1.2 Related Work

Related work comes from different areas: Coupling of SOA and Grid in very large scale IT landscapes is examined in (Holmes, Johnson and Miller, 2003) and (Chen et al., 2006). There is even a widely accepted proposed architecture (OGSA; (Foster et al., 2005)) for these kind of systems. It is however a

Kleiner C. and Koschel A. (2008).
BRINGING TOGETHER WHAT TOGETHER BELONGS - Applying Web Services to Couple SOA and Grid in Smaller Environments.
In *Proceedings of the Fourth International Conference on Web Information Systems and Technologies*, pages 30-37
DOI: 10.5220/0001524900300037

very general architecture concept (similar in (Huang, 2003)) not focused on a specific problem context.

More specific work just occurs sporadically in some kind of individual solution, e.g. for astronomy (Wang et al., 2006), bioinformatics (Xu et al., 2006), or geosciences ((Fraser, Rankine and Woodcock, 2007), (Patra and Das, 2005)). This paper thus adds more specific work that especially addresses smaller IT environments in the combined GRID/SOA space. The fact that many of the subject related articles are quite recent, supports the claim that there is still much work needed in this area.

For distributed search in Grids a data structure is proposed by (Tadepalli, 2006), not combined with SOA however. Prototypes for easier WS development in Grids are shown in ((Bocchi et al., 2005), (Kwon, Choi and Cho, 2006)), but they assume an SOA to communicate among the Grid nodes only.

2 SOA, GRID AND WEB SERVICES

To utilize the benefits of SOA and Grid within one single IT system jointly, a concept is required to couple both technologies. Two options are obvious: Option one utilizes Web services as communication technology between Grid nodes; option two is an SOA, which uses a Grid as some super service for calculation or data storage.

Within option one the nodes of a Grid can be seen as services within an SOA. Each node can typically be used via SOAP/WS and together the services form a Grid. All communication within the Grid is thus based on Web services. Part of the case study for this option is to check, whether the flexibility gained by the usage of Web services help significantly or whether the overall Grid performance is influenced negatively instead. Possibly the decision about the practical relevance of this option is even depending on the specific application and the type of Grid (computing Grid, data Grid, etc).

In the second option the whole Grid as one (service) element within an SOA is examined. The Grid offers its service to other components within the SOA; again SOAP/WS is used, but only to communicate with the node which controls the Grid but not necessarily within the Grid itself. Internally the Grid could instead use any kind of communication protocol. This architectural option seems well suited if an existing computing or data Grid shall be utilized within an SOA. From an SOA viewpoint the Grid is just another service like other services. Grid specifics are not used.

The following chapter shows a case study for both options. Based on the gained experienced both architecture options are conclusively valued.

3 CASE STUDIES

For a practical evaluation of the combined Grid and SOA concepts in smaller environments (smaller with respect to hardware in use), two prototypes where developed as case studies from Q3/2006 till mid 2007. Both prototypes used open source Grid- and SOA technology (Globus IV-Toolkit (Foster, 2006), Celtix 1.x (Celtix, 2007) or Apache CXF Enterprise Service Bus (CXF, 2007)). The development was executed by 11 bachelor students in their final study year, supervised by 2 professors. Effort was 1 day per week per student. Both prototypes were based on medium size Linux PC workstations as well as 2 multi processor Linux servers.

3.1 Focus Grid: “RSA Key Challenge”

3.1.1 Description

The idea behind the first case study was to utilize a computing Grid to break (reasonable easy) RSA keys based on factorization of prim numbers. Motivation for this prototype was the ”RSA Key Challenge“ (RSA, 2007). Within this case study a simple “brute force” factorization algorithm using division by prim numbers was used. The focus was on getting started with Grid and SOA technology with a reasonable easy problem domain.

Figure 1 shows the architecture of the first case study. Monitors mean end users, pc symbols mean system components possibly distributed across different computing nodes. SOAP/WS is used for communication between nodes.

Starting point in figure 1 is the presentation server component. It is a servlet, which takes user-initiated tasks for factorization and passes them on to the management server. In return it gets back results and presents them to the end user. Core component is the management server WS. It distributes sub tasks for prim factorization to different Grid nodes.

For some more technical detail, listing 1 in appendix A shows an excerpt of the management server’s WSDL. It contains operations to start and delete (Grid node) jobs, get a list a of current jobs etc. Note the relation to resource descriptions and

stateful services, which occasionally occur in Grid service architectures.

Each Grid node is a Web service itself, which takes a number to be factorized and an interval to be tested. If the factorization is successful, the result is returned, otherwise an error. The Grid nodes process their number intervals in parallel. Using the monitoring tool Nagios runtime supervision of the overall SOA-Grid is performed and visualized.

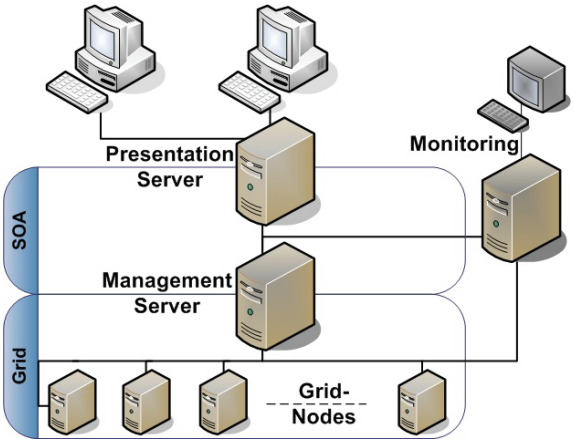

Figure 1: Case study 1 – "RSA Key Challenge".

3.1.2 Experiences

Development of the software for this case study was performed just as a typical standard application would have been. No major problems were encountered and the students had no problems to develop software for this specific area. Even the new technologies such as web services and Grid computing did not pose any major challenges apart from the standard setup time required to be familiar with a new technology.

The project team of students was divided into sub teams which were used as domain experts during the course of the project. Domain experts were concerned with tackling the following work packages: project management, detailed specification of the scenario, Grid technology, enterprise service bus (ESB) technology, hardware and system software as well as monitoring.

Some minor problems have been encountered in the Grid software team. These were due to the somewhat unhandy web services layer of the Grid software used (Globus toolkit 4). Consequently this layer has not been used directly in the case study but rather the more basic GridFTP has been used for data transfer on the Globus side. Since the case study's goal was to examine web services in a Grid context the team developed its own web service wrapper for the GridFTP base technology. This wrapper has been based on another open source product, namely the Celtix ESB implementation.

With this case study focusing on using a Grid to solve a computationally complex problem it should also be evaluated how and to what extent a Grid will help to solve the problem faster than a single machine could. Our lab environment for performing this quantitative analysis consisted of up to 4 completely identical Grid nodes. Standard PCs with exactly the same hardware and software components have been used. Since only relative times are relevant in this study the simple hardware is sufficient as long as all nodes use the same.

Table 1: "RSA Key Challenge" – Grid speedup.

Number of Nodes	Very small numbers	Small numbers	Large numbers
1	4.5 s	4.4 s	1254 s
2	4.4 s	4.5 s	618 s
3	4.4 s	4.5 s	419 s
4	4.7 s	3.6 s	120 s

As shown in Table 1 we can see that for very small numbers only the Grid setup time dominates and this using multiple Grid nodes is not reasonable. With a little bit larger numbers we can detect the first significant speedup, but still using a Grid does not gain enough. This situation changes completely with using really large numbers (in our study these numbers consisted of 23 digits, but this is definitely dependent on the hardware used). Here we can observe a significant speedup by using a Grid. Since the setup time of the Grid is negligible here (compare the 4s for very small numbers to the absolute time required now), even two Grid nodes already lead to a near linear speedup. In the case of 4 nodes we even observed super-linear speedup which is due to some specifics of our implementation. In general at least relatively linear speedup can be expected. Note, that we did not test speedup which goes widely beyond 4 nodes, e.g. to 100 or 1000 nodes. However, due to the "linear design" of our RSA scenario, we still expect at least very reasonable speedup for larger scale environments as well. This should hold at least as long as we do not hit network speed or latency boundaries significantly.

Figure 2 illustrates that the good performance results by adding more Grid nodes are not dependent on the existence of a solution to the factorization problem (red line) or non-existence (green line). In both cases similar speedup can be achieved. This test has been using large numbers only.

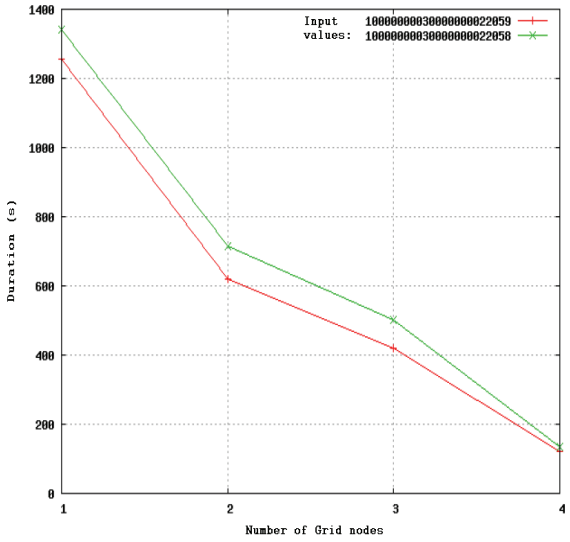

Figure 2: "RSA Key Challenge" – Speedup with/without solution (large numbers).

3.2 Focus SOA: Specialized Web Search Engine – "IT Web Indexer"

3.2.1 Description

Figure 3 illustrates the second case study. It shows an "IT" specialized Web search engine, an "IT Web Indexer". The components work jointly in a SOA, which uses a simulated data Grid as a storage service. For time reasons, this data Grid was implemented as a distributed, replicated database.

The IT Web Indexer works as follows: a Web crawler WS manages a list of URLs to be examined in the data Grid. Each loaded Web page is reduced to its text content only and passed to an analysis WS, which examines the page for "IT relevance" by means of a pre-defined IT glossary. If a certain score is exceeded, the page is seen as relevant and stored using the data Grid service. The value is passed to an index server WS, which indexes the word from the Web page, eliminates duplicates etc.

End users access the results by means of a Web page, which interacts with the user (input gathering and validation, result preparation) as well as with the request server WS. The latter separates the user query into IDs, which are prepared as queries for the data Grid storage service. The query result is a list of page IDs, from which the user can pick, like in popular Web search engines.

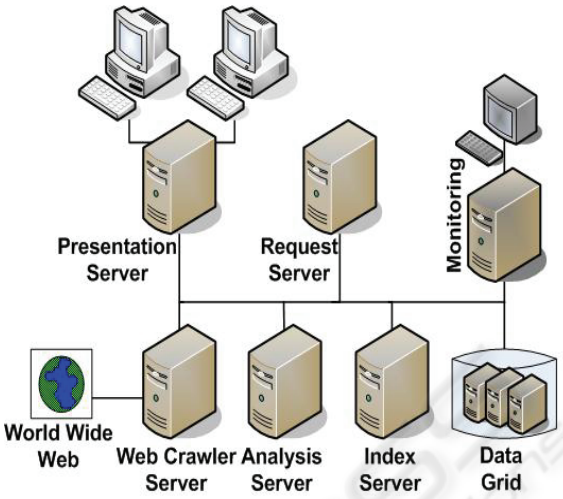

Figure 3: Case study 2 – "IT Web Indexer".

3.2.2 Experiences

As expected using a SOA with Web Services for communication lead to an extremely flexible system architecture. On one hand the separation of the whole software system into several functional components induces a very well structured software system. Single components (where a component can be identified by the functionality it provides to the system as a whole) can be replaced by different implementations as desired. On the other hand the different software components within the case study could be distributed arbitrarily among the available resources. This leads to a well improved usage of resources which can even be adjusted dynamically depending on the current need of the system. Thus it was possible to achieve a pretty good system throughput (measured in terms of web sites scanned and hits found per time unit) even with the very limited resources available in a student project.

As an example for the good flexibility, which the SOA-based design provides, one could exchange the front end easily with a self-developed application, which just calls the appropriate Web service. Other options would be specialized front-ends for mobile devices like PDAs or highly interactive Web GUIs based on Web 2.0 technologies like Ajax. Similarly, the crawler Web service could be implemented in different versions, e.g. to search other data sources like local documents rather than the Web. Figure 4 shows such SOA-based architectural variations of the "IT Web Indexer".

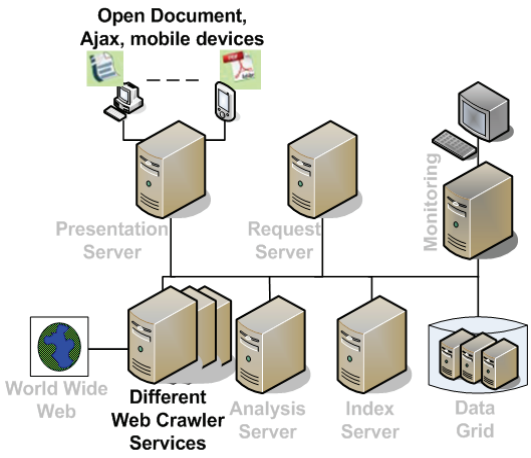

Figure 4: Architectural variations – ”IT Web Indexer”.

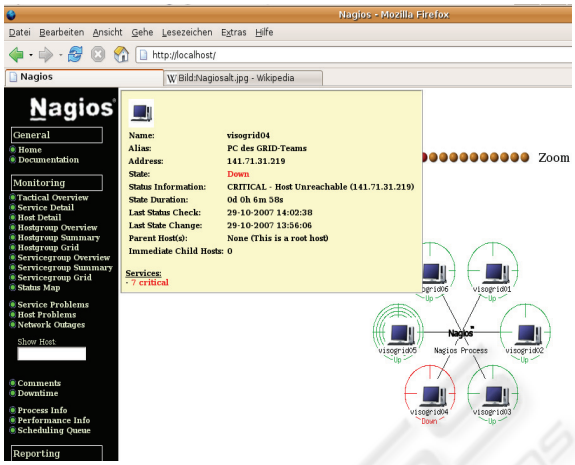

Figure 5: SOA-Grid-monitoring in action.

The data Grid providing database storage and access capabilities for the search engine has been included into the SOA as a “super service”, meaning that communication with the data Grid is performed just as communication with any service in the system (thus making it a service); on the other hand the data Grid provides much more functionality and its internal complexity is much larger compared to a classical service (making it a super service). Nevertheless viewing the data Grid as a service made it possible to include it smoothly into the system: There was e.g. no need for any specific interaction methods.

In total, the service-based design of the system resulted in a highly flexible architecture, as promised by SOA.

Similar to our study only limited resources will be available in a realistic setting where Grid and SOA are to be employed in smaller environments, e.g. small companies. Our second case study shows that even then a combination of the two technologies can lead to a considerable performance boost due to the flexible architecture and improved resource usage.

As in the first case study a specifically extended version of the open source tool Nagios provided integrated monitoring capabilities of both SOA and Grid. Such integrated capabilities proofed to be essential for both the development as well as the operation phase. During the development phase monitoring is important for locating errors originating from the distributed nature of the application. During system operation, monitoring is required for availability control as well as for system tuning issues. The following Figure 5 shows a screenshot of the running monitoring system, which monitors the SOA-Grid. As can be seen, all nodes run fine except node visogrid04 which is currently down.

In our experience without proper monitoring of the different nodes it is quite difficult to automatically detect available resources which may be employed by a different component for optimized system performance. Without monitoring, the overall SOA-Grid would be much harder to develop and almost impossible to manage even in small environments like ours.

4 CONCLUSION AND FUTURE WORK

4.1 Lessons Learned from Case Studies

Both case studies showed that the combination of SOA and Grid technologies is possible without any problems. We can also conclude that both architectural variants may be employed for implementation. Which of the two variants should be preferred is strongly application dependent.

The first architecture where a computing Grid uses web services for internal communication has been proven to provide an easy implementation of a computing Grid. Without using web services (and by using a proprietary communication protocol instead) the system would have been much less flexible. I.e. using different hardware and operating system on some of the Grid nodes would not have been possible without additional development effort. This architecture is particularly beneficial, if the messages exchanged among the Grid nodes are diverse and the hardware, operating system and/or software on the Grid nodes varies. Even though web services incur a certain communication overhead we observed a significant speedup of the Grid application and the improved flexibility outweighs

the overhead in this context. The performance analysis, which we performed, showed the significant speedups, which are possible especially even for small SOA-Grids like ours.

The architecture in the second case study, where a database Grid was integrated into a distributed application as a super web service, has also proven to be efficient for this kind of application. In this case the flexibility of the SOA has been used in favor of implementation of the distributed application. The advantage of flexible resource allocation based on a possibly heterogeneous computing infrastructure facilitates an efficient implementation of this complex application. Communication within the data Grid is proprietary in this case, since the messages to be exchanged are fixed and well-known. This holds since they are defined by the database system used for implementing the data Grid. Nevertheless Web services are used to access the data Grid as a whole in order to be able to use the data Grid as flexible and wide spread as possible. The highly increased flexibility shows, that SOA usage makes sense, even for comparable small applications. The easy integration of a Grid as a super-service, as well as flexible exchange or additions of services within the second case study pointed this clearly.

In both studies we observed that a loose coupling of the two concepts Grid and SOA proved beneficial. The Grids have been integrated into the distributed application by means of services in both cases. This makes flexible usage of the Grids possible. Within the Grids services have been used for the computing Grid but not for the data Grid. As a general rule we can state that services within the Grid provide an advantage as long as many diverse services are used within the Grid and a heterogeneous computing infrastructure is used. If that is not the case proprietary communication should be used within the Grid.

Distributed applications running on a heterogeneous infrastructure require advanced monitoring capabilities of the system as a whole. In our studies the tool Nagios with specific extensions developed within the scope of the studies proved to be extremely valuable. It is possible to monitor hardware as well as software status of the system. Monitoring for both Grid and SOA could be nicely integrated and a good overview of the system as a whole is achieved.

All the software developed within the case studies has been based on open source products. The fact that the applications have been successfully implemented based on this kind of software with acceptable effort show that even in such complex application scenarios open source software is an alternative. Especially in the case of SMEs where financial investments in IT have to be quite limited it is important to be able to use open source software. Cheap software complements the potential to use a heterogeneous computing infrastructure and flexible resource allocation very well. Jointly it is possible to implement a complex and highly productive software system at comparable low cost. This is especially important for smaller environments in SMEs.

Eventually, the results showed as well, that such relatively complex integration projects are quite feasible for teaching purposes in (advanced) student courses respectively projects.

4.2 Future Work

The quantitative evaluations of the distributed applications developed in the case studies should be extended further. Especially many more different hardware and software foundations on the Grid nodes should be examined. This could potentially lead to more insight into the influence of heterogeneity for the Grid and SOA applications. Of course interesting as well would be to add significantly more Grid nodes, to validate the scalability results. We plan to explore this in the future. In this case care must be taken however, that our targeted “smaller environments” are still addressed here primarily.

If the customized extensions to Nagios would be extended somewhat further, a tool for general Web Service and Grid management and monitoring could be obtained. Such a tool would be of great benefit to many different SOA-based software systems and has many potential use cases.

The combination of SOA and Grid should also be employed for implementation of real-world applications used within a SME. Practical experiences gained from this kind of applications and from the integration of the different technologies into an actual IT landscape of a SME could reveal further perception of the applicability of the architectural variants. Finally more benefits of the combination of Grid and SOA could be deduced and the necessity for improvements could be detected.

ACKNOWLEDGEMENTS

We would like to thank our students from the ViSoGrid project team for their highly productive and enthusiastic work in this project. Special thanks goes to the team members E. Friedrich, B. Hellmann, A. Reich, M. Schaaf, and J. Salzwedel.

REFERENCES

Bocchi, L., Ciancarini, P., Moretti, R., Presutti, V., and Rossi, D., 2005. *An OWL-S based approach to express Grid services coordination*. Proc. 2005 ACM Symp. on Applied Computing. Santa Fe, New Mexico. L. M. Liebrock, Ed. SAC '05. ACM Press, New York, 1661-1667.

Bunn, J., van Lingen, F., Newman, H., Steenberg, C., Thomas, M., Ali, A., Anjum, A., Azim, T., Khan, F., Rehman, W. u., McClatchey, R., and In, J. U., 2005. JClarens: *A Java Framework for Developing and Deploying Web Services for Grid Computing*. In Proc. IEEE Conf. on Web Services (ICWS'05) - Volume 00 (2005). ICWS. IEEE CS, Washington, DC, 141-148.

Celtix, 2007. *Celtix Enterprise Service Bus, 1.x.* objectweb.org. Acc. July 2007.

Chen, X., Cai, W., Turner, S. J., and Wang, Y., 2006. SOAr-DSGrid: *Service-Oriented Architecture for Distributed Simulation on the Grid*. In Proc. 20th Workshop on Principles of Advanced and Distributed Simulation. Workshop on Parallel and Distributed Simulation. IEEE CS, Washington, DC, 65-73.

Conrad, S., Hasselbring, W., Koschel, A., Tritsch, R., 2005. *Enterprise Application Integration*, Spektrum Akademischer Verlag, Germany.

Apache CXF Enterprise Service Bus, 2007. apache,org. Acc. July 2007.

Erl, T., 2005: SOA: *Concepts, Technology, and Design,* Prentice-Hall.

Fang, L. 2006 *A Scalable Capability-Based Authorization Infrastructure for Web Services in Grids*. Doctoral Thesis. UMI Order No. AAI3215173., Indiana University.

Foster, I., Kesselman, K. (eds), 2003. *The Grid 2: Blueprint for a New Computing Infrastructure. Morgan Kaufmann*.

Foster, I., Kishimoto, H., Savva, A., Berry, D., Djaoui, A., Grimshaw, A., Horn, B., Maciel, F., Siebenlist, F., Subramaniam, R., Treadwell, J., Von Reich, J., 2005. *The Open Grid Services Architecture, Version 1.0*. Informational Document, Global Grid Forum (GGF).

Foster, I., 2006. *Globus Toolkit Version 4: Software for Service-Oriented Systems*. IFIP Intl. Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, 2-13.

Fraser, R., Rankine, T., Woodcock, R., 2007. *Service oriented grid architecture for geosciences community*. In: Proc. 5th Australasian symposium on ACSW frontiers (ACSW07), Australian Computer Society, Inc., Darlinghurst, Australia, 19 – 23.

Friese, T., Smith, M., and Freisleben, B. 2004. *Hot service deployment in an ad hoc Grid environment*. Proc. 2nd Intl. Conference on Service Oriented Computing (New York, NY, USA, November 15-19, 2004).ICSOC '04. ACM Press, New York, NY, 75-83.

Holmes, V., Johnson, W., Miller, D., 2003. *Integrating Web Service and Grid Enabling Technologies to Provide Desktop Access to High-Performance Cluster-Based Components for Large-Scale Data Services*. Proc. 36th annual symposium on Simulation (ANSS03), IEEE Computer Society Press, Washington, DC. 167.

Huang, Y., 2003. JISGA: *A Jini-Based Service-Oriented Grid Architecture*. International Journal of High Performance Computing Applications, Sage Publ., Thousand Oaks. 317-327.

Krafzig, D., Banke, K., Slama, D., 2005. *Enterprise SOA*. Prentice Hall.

Kwon, S., Choi, J., and Cho, K., 2006. *Light-weight service-oriented Grid application toolkit*. Proc. of the '06 ACM Symp. on Applied Computing (Dijon, France, April 23-27, '06). SAC '06. ACM Press, New York, NY. 1482-1486.

Lomow, G., Newcomer, E., 2005. *Understanding Service-Oriented Architecture (SOA) with Web Services*. Addison-Wesley.

Nagios, 2007. *Nagios Monitoring Toolkit*, Nagios.org. Acc. July 2007.

Patra, M., Das, R., 2007. SORIG: *A service-oriented framework for rural information grid -- an implementation viewpoint*. Proc. 1st Intl. Conf. on Theory and practice of electronic gov. (ICEGOV07), ACM Press, New York, NY. 49 – 52.

RSA, 2007. *RSA Key Challenge.* rsasecurity.com. Acc. July 2007.

Starke G., Tilkov S. (Edts.), 2007. *SOA-Expertenwissen*, dpunkt. Germany.

Tadepalli, P., 2006. *Grid-based distributed search structure*. In: Proc. of the 44th annual ACM Southeast regional conf., New York. 752-753.

WWW Consortium (W3C), 2007. Web Services specifications, W3C.org. Acc. July 2007.

Walker, D. W., 2003. *Grid Computing: Infrastructure and Applications. Intl. Journal High Performance Compu*ting, Appl. 17, 3, 207-208.

Wang, M., Du, Z., Chen, Y., and Cheng, Z., 2006. *A SOA Based Pipeline System to Deal with Astronomy Telescope Data*. In Proc. of the 2nd IEEE Intl. Symposium on Service-Oriented System Engineering (Sose'06)-Vol.00 (Oct. 25-26,'06). SOSE. IEEE Computer Society, Washington, DC, 156-166.

Xu, G., Luo, Y., Yu, H., and Xu, Z. 2006. *An Approach to SOA-Based Bioinformatics Grid*. In Proc. of the 2006 IEEE Asia-Pacific Conf. on Services Computing (December 12 - 15, 2006). APSCC. IEEE Computer Society, Washington, DC, 323-328.

APPENDIX A

The following listing 1 shows an excerpt of the WSDL for the management server from the RSA key challenge case study.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
  xmlns:soap= "http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns= "http://visogrid ... /Rsakc/"
  <wsdl:types>
      . . .
      <xsd:element name="addJob">
        <xsd:complexType>  <xsd:sequence>
            <xsd:element name="name" type="string" />
            <xsd:element name="key" type="string" />
        </xsd:sequence>    </xsd:complexType>
      </xsd:element>
      <xsd:element name="addJobResponse">
        <xsd:complexType>  <xsd:sequence>
            <xsd:element name="id" type="int" />
        </xsd:sequence>    </xsd:complexType>
      </xsd:element>
      <xsd:element name="deleteJob">
        <xsd:complexType>  <xsd:sequence>
            <xsd:element name="id" type="int" />
        </xsd:sequence>    </xsd:complexType>
      </xsd:element>
      . . .
  </wsdl:types>

  <wsdl:message name="addJobRequest">
    <wsdl:part name="in" element="tns:addJob" />
  </wsdl:message>
  <wsdl:message name="addJobResponse">
    <wsdl:part name="out" element="tns:addJobResponse"></wsdl:part>
  </wsdl:message>
   . . .

  <wsdl:portType name="Rsakc">
    <wsdl:operation name="addJob">
      <wsdl:input message="tns:addJobRequest" />
      <wsdl:output message="tns:addJobResponse" />
    </wsdl:operation>
      . . .
  </wsdl:portType>

  <wsdl:binding name="RsakcSOAP" type="tns:Rsakc">
    <soap:binding style="document"
          transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="addJob">
      <soap:operation soapAction="http://visogrid .../Rsakc/NewOperation" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
        . . .
    </wsdl:operation>
      . . .
  </wsdl:binding>

  <wsdl:service name="Rsakc">
    <wsdl:port  binding="tns:RsakcSOAP" name="RsakcSOAP">
      <soap:address location= "http://visogrid .../SoapContext/SoapPort" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Listing 1: WSDL excerpt for RSA key challenge.