Literature search on SOA, Web Services, OGSA and WSRF

Martin Kuba[1], Ondřej Krajíček[1]

[1]*Institute of Computer Science, Masaryk University,*
Botanická 68a, 602 00 Brno
{makub,krajicek}@ics.muni.cz

April 17, 2007

Abstract

This text is a result of literature search on topics of Service Oriented Architecture (SOA), Open Grid Services Architecture (OGSA) and Web Services Resource Framework (WSRF). It is not meant as a complete resource on the mentioned topics, rather it is a survey of materials encountered while working with these technologies.

1 Motivation

The Grid, as it was envisioned in the classic articles *Computational Grids* [1], *The Anatomy of the Grid* [2] and *What is the Grid? A Three Point Checklist* [3], should be an infrastructure for broad (dependable, consistent, pervasive, inexpensive, flexible, secure, coordinated – to cite the classics precisely) sharing of diverse resources while solving problems. But such infrastructure meets big challenges in interoperability and platform-independence, as many if not all hardware platforms, operating systems and programming languages need to be included.

A way out of this problem, which is not specific only to the Grid, promises service orientation, expressed in *The Physiology of the Grid* [4] and *Grid Services for Distributed System Integration* [5]. The Grid should be now based on Open Grid Services Architecture (OGSA).

This paper thus provides overview of concepts which now seem to be fundamental to future evolution of the Grid.

2 Service Oriented Architecture

2.1 Motivation for SOA

Computer systems are evolving since their inception, to be able to tackle more and more complex problems. Methodologies for programming and system architecture must evolve accordingly. The evolution can be described from one point of view as from centralized monolithic single-threaded systems to distributed componentized parallel systems.

Currently a new methodology for building such complex distributed systems is being discussed, named Service Oriented Architecture. Its definition and principles are the topic of this section.

The basic idea of SOA is not new. The idea is roughly that it would be nice to create complex systems from simple separate parts. The idea was probably first expressed in *modular programming* (programming languages like Pascal and Modula-2). Later it was expressed as *component oriented programming*, which suggests that software should be composed from parts in similar way as in mechanical industry a bicycle can be assembled from prefabricated components made by independent vendors. The idea was also expressed in object-oriented systems, both nondistributed and distributed ones. All of these programming paradigms support separation of interface and implementation of a given basic part (module, component, object), where the interface is all that the rest of the world needs to know about the part.

The recent interest in the SOA was spurred by the emergence of Web Services [6], which are described in detain in section 3.

2.2 Service Definition

Let's begin with defining what a service is. Etymologically it comes from Latin *servitium* - "slavery", which is derived from *servus* - "slave". The term **service** is very overloaded, Webster's Dictionary lists 31 meanings. An interesting point is that in the definitions, which make sense for our purpose, service is defined as a process, as "an act of helpful activity", "the performance of any duties or work for another", "useful labor that does not produce a tangible commodity". However in computer-oriented texts, a service usually means an entity:

- From [4] (Physiology): a network-enabled entity that provides some capability through the exchange of messages
- From [7] (IBM): a functional unit
- From [8] (W3C): an abstract resource that represents a capability of performing tasks that form a coherent functionality from the point of view of providers entities and requesters entities; to be used, a service must be realized by a concrete provider agent
- From [9] (Wikipedia): a self-contained, stateless business function that accepts one or more requests and returns one or more responses through a well-defined, standard interface
- From [10] (GGF): a software component participating in a service-oriented architecture that provides functionality and/or participates in realizing one or more capabilities
- From [11] (WS-I+): the logical manifestation of some physical or logical resources (like databases, programs, devices, humans, etc.) and/or some application logic that is exposed to the network; and Service interaction is facilitated by message exchanges
- From [12] (WS-GAF): a well-defined set of actions, it is self-contained, stateless, and does not depend on the state of other services
- From [13] (RDNL): the provision of, or system of supplying, one or more functions of interest to an end-user or software application.
- From [14] (OASIS RM draft): The performance of work (a function) by one for another.
- From [14] (OASIS RM draft): A service is a mechanism to enable access to a set of one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description.
- From [15] (ZapThink): Service is a contracted interface to software functionality and data that communicates via messages

The closest dictionary definition to this is "a facility supplying some public demand (telephone service, bus service)", however a service in the computer oriented meaning needs not to be public.

We can conclude that a service is a software component performing some work for others. However there are some corner cases. Some services may not be accessible over network for performance reasons, but then it is debatable whether a simple subroutine invocation inside a program can be too considered invocation of a service.

2.3 SOA Definition

It is difficult to define what a Service Oriented Architecture is. The term is being used in an increasing number of contexts with conflicting understandings of implicit terminology and components. The problem got even so far that the OASIS standard organization formed a committee in May 2005 to develop a *SOA Reference Model* [16] for "preserving a common layer of understanding about what SOA is". The first draft is available since February 2006 [14].

Let's list some of SOA definitions:

- From [17] (XML.com): SOA is an *architectural style* whose goal is to achieve loose coupling among interacting software agents. A service is a unit of work done by a service provider to achieve desired end results for a service consumer.
- From [7] (IBM, alphaWorks) : SOA is a *component model* that inter-relates an application's different functional units, called services, through well-defined interfaces and contracts between these services. The interface is defined in a neutral manner that should be independent of the hardware platform, the operating system, and the programming language in which the service is implemented.
- From [18] (IBM, Joseph et al.): SOA is a loosely coupled *architecture* with a set of abstractions relating to components granular enough for consumption by clients and accessible over the network with well-defined policies as dictated by the components.
- From [19] (BEA): SOA is a *design methodology* aimed at maximizing the reuse of application-neutral services to increase IT adaptability and efficiency.
- From [8] (W3C): SOA is a *set of components* which can be invoked, and whose interface descriptions can be published and discovered.
- From [20] (Microsoft) SOA is usefully defined as the *policies, practices, frameworks* that enable application functionality to be provided and consumed as sets of services published at a granularity relevant to the service consumer. Services can be invoked, published and discovered, and are abstracted away from the implementation using a single, standards-based form of interface.
- From [10] (GGF): SOA refers to an *architectural style* of building reliable distributed systems that deliver functionality as services, with the additional emphasis on loose coupling between interacting services.
- From [21] (Artima): SOA is an *architecture* based on services, where a service is a distributed resource that is accessed in accordance with its interface
- From [14] (OASIS RM draft): SOA is a *paradigm* for organizing and utilizing distributed capabilities that may be under the control of different ownership domains

We can conclude that there is an agreement on the basic level of SOA:

SOA is a methodology for building distributed systems, which recommends using independent components, named services, which communicate by exchanging messages, and whose interfaces are described in machine-processable way in terms of which messages can be exchanged.

It should be noted, that there is a clear tendency to distinguish SOA from Distributed Object systems, by accenting loose coupling, as summarized in section 2.3.1. Loose coupling should help in building distributed systems which are robust and which can evolve over time in absence of closely administered environment.

2.3.1 SOA versus Distributed Objects

Previous methodology for building distributed systems was Distributed Objects, which were tightly coupled RPC (Remote Procedure Call)-oriented systems. They have met only a limited success in closely administered homogeneous environments and failed to scale to Internet-sized distributed systems. Experience suggests that they have some inherent problems because of the tight coupling that they require between distant systems [22, 23, 24]. Distributed Objects systems were for example DCOM, CORBA, Java EJB/RMI.

SOA may be in contrast with Object-Oriented systems [17, 14], which tend to bind data and its processing together, especially if our view of object oriented technologies is limited to the imperative approach adopted by C++ and similar languages (Java, C# 1.0, etc.). The contrast is aptly described in [17] as "a CD player offers a CD playing service", while "in object oriented programming style, every CD would come with its own player and they are not supposed to be separated". This view of object oriented concepts is however limited to systems where objects interact with passing references and invoking methods. This approach leads to non-trivial dependencies of object references. However, object systems where objects are interacting by passing messages (such as in Smalltalk or Self programming languages), exhibit interesting similarities to SOA.

The distinction between Service-orientation and Object-orientation is important, so let's state it explicitly here: Service Oriented Architecture is not another level of Distributed Objects technology like CORBA or DCOM. In [14], the distinction is described as: "Unlike Object Oriented Programming paradigms, where the focus is on packaging data with operations, the central focus of Service Oriented Architecture is the task or business function getting something done. This is a more viable basis for large scale systems because it is a better fit to the way human activity itself is managed – by delegation."

A detailed discussion of the differences is given in *Web Services Are Not Distributed Objects*[24], we will summarize them here.

The common features of SOA and Distributed Objects (DO) are:

- both have explicit description of their external interface
- both have well-defined network interactions
- both provide means for component discovery, usually by means of component registry

However, there are significant differences between SOA and DO:

- DO have object life cycle – a factory instantiates an object, later it is destroyed; services do not have instances or factories
- DO can contain references to other objects; services are independent

- DO instances with mutual references require sofisticated reference handling and distributed garbage collection; services don't have them
- an object instance has internal state; services should be stateless
- DO are RPC-oriented, they have interfaces with methods for remote invocation; services communicate by document-oriented messages
- richness of information flow in DO is given by the interface of objects; in services it is given by the documents that can be exchanged
- DO technology is matured and robust, designed for closely administered environments; SOA is aimed at Internet-style distributed computing for interoperability in heterogeneous environment

The reasons given for the failure of Distributed Objects system are [24, 22, 25]:

- synchronous interactions over wide-area networks are not scalable
- large-scale versioning of procedure interfaces is extremely difficult (RPC interfaces don't support evolution of distributed systems)
- vendors compete on ORB (Object Request Broker) implementations and have no motivation for interoperability

The common misconception that SOA is another DO probably comes from the fact, that the most widely used form of SOA are web services, and the web services' protocol SOAP was originally named Simple Object Access Protocol and was indeed intended for RPC calls on remote objects. However over time web services and SOAP were aligned with SOA, their RPC-oriented features were suppressed and now they focus on exchange of documents.

This is not to say that object-orientation is wrong, it is still a valuable methodology for implementing the internals of services. But for distributed systems, it has been argued in a classic article *Note on Distributed Computing* [26] that it it imposible to treat local and remote objects in the same way, and SOA does not try to hide this distinction. On the other hand, there are research projects which try to devise new object-based programming models and methods for distributed systems (such as Proactive toolkit described in [27]) which may bring new interesting results in this field.

2.4 Features of Service Orientation

While there is no agreement on exact SOA definition, we can complement the definitions listed in section 2.3 by listing SOA properties and design principles.

SOA is a form of distributed systems architecture that is typically characterized by following properties [28]:

- **Logical view:** The service is an abstracted, logical view of actual programs, databases, business processes, etc., defined in terms of what it does, typically carrying out a business-level operation.
- **Message orientation:** The service is formally defined in terms of the messages exchanged between provider agents and requester agents, and not the properties of the agents themselves. The internal structure of an agent, including features such as its implementation language, process structure and even database structure, are deliberately abstracted away

in the SOA: using the SOA discipline one does not and should not need to know how an agent implementing a service is constructed. A key benefit of this concerns so-called legacy systems. By avoiding any knowledge of the internal structure of an agent, one can incorporate any software component or application that can be "wrapped" in message handling code that allows it to adhere to the formal service definition.

- **Description orientation:** A service is described by machine-processable meta data. The description supports the public nature of the SOA: only those details that are exposed to the public and important for the use of the service should be included in the description. The semantics of a service should be documented, either directly or indirectly, by its description.
- **Granularity:** Services tend to use a small number of operations with relatively large and complex messages.
- **Network orientation:** Services tend to be oriented toward use over a network, though this is not an absolute requirement.
- **Platform neutral:** Messages are sent in a platform-neutral, standardized format delivered through the interfaces. XML is the most obvious format that meets this constraint.

In SOA, there are a number of design principles that should be followed [29]:

- **Boundaries are explicit:** The boundaries of a service are well-defined when they are incorporated into a distributed application. Other services do not see the internal workings, implementation details, or resource representations of a service.
- **Services are autonomous:** Service implementations are developed and evolve independently from one another.
- **Services share schema and contract, not classes:** In service-oriented architectures, no single set of abstractions (classes) spans an entire application. Services share schemas (contracts) that define the structure of the information that they exchange, not information about their underlying type systems.
- **Policies determine service compatibility:** Services interact with one another only after it has been determined based on policy assertions that they can meaningfully exchange information.

2.4.1 Temporal simultaneity

Services orientation has some implications. As stated in *Data on the Outside Versus Data on the Inside* [30], an intrinsic part of SOA is that different services live in their own temporal domains, there is no simultaneity at distance, so by the time a service sees a message from a distant service, the data the message is carrying may have changed inside the distant service. "Going to SOA is like going from Newton's physics to Einstein's physics" [30]. Services must be able to cope with that. This is different from previous RPC systems, which pretended that many systems look like a single system with observable simultaneous state.

One suggested way how services can cope with the temporal problem is *immutable versioned data* with stable unchanging interpretation across space and time [30].

2.4.2 State in services

Services can be *stateful* or *stateless*, with stateless services being the preffered choice. The precise definition what stateless means is also difficult. We can distinguish three types of services according to state [31].

1. service which makes no use of information not contained in input message
2. service which keeps no state of conversation with a particular client, but can make permanent changes in outside systems, like a database
3. service which keeps state of conversation with a particular client, i.e. result of one operation depends on prior operations

The first case is obviously stateless service, the third case is stateful service.

In the second case, the service's *implementation* can be seen as stateless, as processing each message is independent on processing of other messages. This helps in scalability, for example, in a multi-step transaction scenario, a client doesn't have to use the same service for all the steps [32]. Also such service can be safely restarted and new copies can be created for handling high load.

3 Web Services

The definition of SOA is technology independent. However the most widely used implementation of SOA is in Web Services, where services are communicating using established Internet standards, namely XML (eXtensible Markup Language) and HTTP (Hyper Text Transfer Protocol), which makes services interoperable.

It should be noted that web services *can* be used in a way that mimics RPC-oriented distributed objects systems, but then it is likely that such systems will repeat the problems encountered by distributed objects systems. It is therefore preferred that web services use Service Oriented Architecture principles.

Definition of a Web Service is given in a document "Web Services Architecture" [28] produced by W3C:

> **A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.**

Other definitions are listed in [33].

3.1 WSDL

The language for describing web services is based on XML and is called WSDL (Web Services Description Language), the current version at the time of writing is 1.1 and is defined in [34]. Currently, the next major version of the standard (probably numbered 2.0) is in the works and currently has the status of candidate recommendation, for more information see: [35].

WSDL models a service interface as sets of *operations*, where each operation consists of zero or one input message, zero or one output message, and any number of fault messages.

This approach was criticized as being too object-oriented, and a pure message-oriented description language was proposed, named SSDL (SOAP Service Description Language) [36].

3.2 SOAP

SOAP was originally defined by Microsoft and IBM, as version 1.1, which was accepted as W3C Note [37], together with WSDL version 1.1. Then W3C developed a next version of SOAP, endorsed as W3C Recommendation [38].

In previous years much attention was paid to the overal performance of the SOAP communication protocol. Several issues were identified, mostly related to the achievable performance. SOAP protocol imposes significant overhead in terms of processing power required by necessary SOAP stack implementation and bandwidth for transmitting SOAP messages. These issues are described in [39], with references to other works devoted to this issue.

Solutions proposed to address these issues fall mostly follow two approaches. Either, they propose usage of different data presentation mechanism (as in [39], [40], [41]) or optimisations or modifications to the current encoding of messages based on XML (as in `DIME` [42], `MTOM` [43], `XOP` citeXOP). The efforts based on the first approach usually provide significant performance improvements, but are not generally interoperable and would require implementation of new standards. The efforts based on the second approach are relatively new. Their advantage is that they are supported by international standard bodies, such as W3C or IETF. In case of the three mentioned representants, the opinion of the authors is that they will probably fail to provide any viable solution for Grid Services, since:

- The `DIME` standard is being implemented in several tools, but the standard proposal itself has expired and is currently only unofficial. Hence it fails to provide good foundations for interoperability of implementations done by different vendors.
- The `MTOM` and `XOP` standards are proposed by W3C and currently have status of W3C Recommendation. However, they only suggest that content, which is natively represented as binary data (and would impose an performance drawback if encoded in XML and transferred as SOAP messages) should be transferred by other means. This solution is not acceptable since it would lead into usage of different communication protocols and would increase the communication complexity of SOAP based applications.

.

3.3 WS-Interoperability

Interoperability problems between SOAP implementations, caused by vague formulations in SOAP and WSDL specifications , led industry to form another organization, named *Web Services Interoperability Organization (WS-I)*, which produced so called *profiles*, which further refine the specifications. It produced WS-I Basic Profile[44] in versions 1.0 and 1.1. The main point of this profile is that only literal encodings are allowed in WSDL, so that SOAP messages can be easily validated against XML Schema. The original SOAP and WSDL specifications allowed also so called "SOAP encoding" which introduced a whole new type system and its encoding in XML. It happened because development of SOAP started before XML Schema was developed.

It should be noted, however, that even SOAP messages validated with pure XML schemas can still cause interoperability issues, as some XML Schema features are so complex, that they are implemented inconsistently across tools, as reported recently (May 2005) during W3C Workshop on XML Schema 1.0 User Experiences [45] [46]. So in the long run, using SOAP in accordance with WS-I Basic Profile and XML Schema should provide universal interoperability together with platform independence, however this goal is yet to be fully achieved.

3.3.1 Interoperability of Web Services

The current recommended practise for creating truly interoperable web services is to create WSDL description first, using document/literal style, and use tooling to generate implementations. This si called "contract-first design". A common practise, where WSDL is generated from implementation classes, is discouraged, and is called "contract-last design" in [47].

It is worth noting that the type system of XML schema cannot be fully mapped to any object-oriented language, as it allows derivation by restriction and other features. The details are described in [47].

3.4 UDDI

UDDI [48] stands for Universal Description, Discovery and Integration. On the contrary to SOAP and WSDL, the UDDI specification was never a W3C recommendation or note. It began in the year 2000 as an industry initiative published at uddi.org, with UDDI specification version 2 defined in June 2001, later version 3 defined in July 2002. Then the effort moved to OASIS standard organization, which ratified the version 3.0.2 of UDDI in February 2005 as OASIS standard.

In the begining, UDDI was a hot technology, however the original enthusiasm faded over years. All three main vendors – IBM, Microsoft and SAP – switched off their public UDDI registries in January 2006 [49]. The given reason is that most entries in such public registries were incorrect. Some texts suggest that UDDI is being used for non-public intra-enterprise registries, however this cannot be verified.

The [11] paper reports that most projects in e-Science programme designed their own service registries independent of UDDI, which suggests that UDDI is not well suited for the task of being a web service registry.

The [50] paper evaluates UDDI as grid registry, and finds that UDDI's limited query model and lack of data typing (compares only strings, not numbers or timestamps) make it possible only for small grids where performance cost of workarounds would be acceptable.

(The reasons for UDDI failure are not described anywhere. Personal experience suggests that UDDI is designed very generally, so that even information not related to web services can be stored, like that a local pizza delivery business has some phone number, because UDDI aimed to be a universal business directory. But the specialized task of being a directory for WSDL-described SOAP-based web services is not handled very well.)

3.5 The WS-* jungle

The SOAP standard was seen as too simple for complex scenarios by many parties. This conclusion led to proliferation of many so called WS-* standards, which we list here sorted alphabeticaly [51]:

1. **WS-Addressing**
2. **WS-Agreement**
3. **WS-AtomicTransaction**
4. **WS-Attachments**
5. **WS-BusinessActivity**
6. **WS-Choreography**

7. **WS-Context**
8. **WS-Coordination**
9. **WS-CoordinationFramework**
10. **WS-Discovery**
11. **WS-Enumeration**
12. **WS-Eventing**
13. **WS-EventNotification**
14. **WS-Federation**
15. **WS-Management**
16. **WS-MessageDelivery**
17. **WS-MetadataExchange**
18. **WS-Notification**
    - **WS-BaseNotification**
    - **WS-BrokeredNotification**
    - **WS-Topics**
19. **WS-Policy**
20. **WS-PolicyAssertions**
21. **WS-PolicyAttachment**
22. **WS-Reliability**
23. **WS-ReliableMessaging**
24. **WS-ResourceFramework**
    - **WS-Resource**
    - **WS-ResourceLifetime**
    - **WS-ResourceProperties**
    - **WS-RenewableReferences**
    - **WS-ServiceGroup**
    - **WS-BaseFaults**
25. **WS-ResourceTransfer**
26. **WS-SecureConversation**
27. **WS-Security**
28. **WS-Transaction**
29. **WS-TransactionManagement**

30. **WS-Transfer**
31. **WS-TransferAddendum**
32. **WS-Trust**
33. **Web Services Distributed Management**

Some are competing specifications from different vendors [11], namely:

- WS-Notification and WS-Eventing
- WS-Coordination and WS-CoordinationFramework
- WS-AtomicTransaction/WS-BusinessActivity and WS-TransactionManagement
- WS-Reliability and WS-ReliableMessaging
- WS-Addressing and WS-MessageDelivery

Tim Bray, co-creator of XML and Director of Web Technologies at SUN Microsystems, says aptly in his blog entry named *The loyal WS-Opposition* [52] (cited in [53, 54]): "No matter how hard I try, I still think the WS-* stack is bloated, opaque, and insanely complex. I think its going to be hard to understand, hard to implement, hard to interoperate, and hard to secure."

On the other hand, enterprise system architects argue that they need such specifications to be able to match the level of complexity of their applications[54]. They don't use all of the WS-* specifications in the same time, they choose only the ones they need. However OntoGrid's overview of state-of-the-art[53] expresses concern whether real interoperability can be achieved in such "mix-and-match world".

In March 2006, main players (IBM, HP, Intel and Microsoft) announced [55] a roadmap for converging the WS-* standards which are important for grids. Details are in section 4.3.

3.6 Alternatives: REST and XML-over-HTTP

Critics of the SOAP and WS-* stack often give the following statistics. Amazon.com offers its API in two forms - a SOAP one and simple XML-over-HTTP one. The XML-over-HTTP one accounts for 80% of traffic, while the SOAP one only 20% [53, 56].

A hot debate emerged around this. The net outcome seems to be that for simple web-based request-response applications, SOAP and WS-* are an overkill. XML processing and HTTP transport are now available in nearly all environments, while SOAP processing limits programmers to much narrower set of tools. On the other hand, SOAP and WS-* are suited for application where messages need to be processed by intermediaries [57, 54] which is often needed in enterprise applications.

The term REST (REpresentional State Transfer) [58] is frequently used in such debates. The term was created by Roy Fielding, a co-author of HTTP and co-founder of the Apache Software Foundation, in his dissertation. In essence, REST is an architectural style, based on the assumption that every resource on the web can be identified by a URI and represented by an XML document. The document can be created, retrieved, updated and deleted using the four basic HTTP operations PUT, GET, POST and DELETE on that URI, or more generaly, new HTTP methods can be used as *verbs* specifying what to do with the resources. Thus REST is very HTTP centric (which is not suprising, as Roy Fielding is HTTP co-author), arguing that HTTP made World Wide Web succesful.

The proponents of REST, called RESTafarians for their agressiveness, recommend that this is all that is needed for web services. However the *Web Services Architecture* [28] document produced by W3C provides an example where REST assumption is not met – a web service adding two numbers together is a web resource, but has no representation as an XML document.

Sometimes the term REST is used loosely to refer to any simple XML-over-HTTP interface, albeit this usage is wrong.

4 OGSA

Principles of OGSA were layed out in *Physiology of the Grid* [4], now it is being defined further by Global Grid Forum (GGF) working group [59].

OGSA is actualy two things in the same time:

- collection of higher-level services needed for a functioning Grid
- basic low-level architecture based on web services for handling state, notifications and registries of services

The list of higher-level services, like data management services, workflow services, monitoring services, is still work in progress in GGF [60]. Some argue, that OGSA should focus only on those higher-level services, which are Grid-specific, while the requirements for the low-level plumbing are not Grid-specific and should be left to the Web Services community to solve [12].

On the low-level, OGSA has originaly seen the Grid as a collection of *grid services*. Each grid service can be created and destroyed, and has an internal state which can be observed. Furthermore notifications can be sent among grid services when the internal state is changed. OGSA also requires *registries* of grid services to be able to find grid services and group them.

The OGSA low-level architecture was first defined in OGSI (Open Grid Services Infrastructure), implemented in Globus Toolkit 3, but OGSI met with resistance from the web services community [61, 62], because OGSI used web services as objects with internal state and limited life span, thus trying to create a CORBA-like architecture on top of web services [12].

For the reasons described in section 2.3.1, web services community want web services to follow service-orientation and not object-orientation. It was proposed that state can be managed is service-orientation compatible way using contextualization, which means passing context identifiers in message headers [12]. Then OGSI was abandoned, and WSRF (Web Services Resource Framework) taken its position.

WSRF keeps web services stateless, but adds so called WS-Resources, which have internal state and limited life span, and can be manipulated thru web services. Each WS-Resource has a set of *properties*, which can be expressed as XML document, and these properties can be read and eventualy set. Notifications can be sent from one WS-Resource to other WS-Resources when WS-Resource properties are changed.

We can summarize that originally OGSA has taken object-oriented approach with OGSI, but later it was made more aligned with service-orientation by separating services and state in WSRF.

It is not certain that WSRF will become a de facto standard for grid computing, as evaluated in [53]. The main problem is that WSRF depends on many WS-* standards, many of them being inter-dependent, and interoperability is difficult to achieve.

An alternative is so called **WS-I+** initiative [11] launched by UK e-Science Programme, which uses more evolutionary approach by using only carefully selected web service specifications. As stated in [53], even approaches radically different to SOAP, like XML-over-HTTP and REST, should not be overlooked.

4.1 Is OGSA Object-Oriented or Service-Oriented ?

OGSA adopted web services "to capitalize on desirable Web services properties, such as service description and discovery; automatic generation of client and server code from service descriptions; binding of service descriptions to interoperable network protocols; compatibility with emerging higher-level open standards, services and tools; and broad commercial support" [4].

However the authors of OGSA added "dynamic creation, lifetime management, notification of state change" to web services, in OGSI directly to the services, in WSRF to stateful resources behind the services. They don't agree with critics that say that web services should be stateless [31],

This list of added features is very similar to the list of properties of Distributed Objects systems (like CORBA, DCOM). This rised the question whether OGSA is trying to reconstruct the failing Distributed Objects systems on top of web services, thus going in fact against Service Orientation [12].

The authors of WS-GAF [29] recommended to use contextualization to manage state when OGSA was implemented as OGSI. WSRF partially followed this suggestion by using resource ID in EPR (Endpoint Reference defined in WS-Adressing) in SOAP headers of messages. However the EPR is still treated as an opaque entity, like an object pointer, which encourages applications to be built around interactions between resources rather than services. So even WSRF is still not completely SOA compliant [63] and is not supported by all major players in web services community.

4.2 WSRF

*(**TODO**: This section is not finished.)*

WSRF [64]. Evolution of grid computing architecture and grid adoption models [18]. An Early Evaluation of WSRF and WS-Notification via WSRF.NET [65]. A Comparison of Five WSRF Implementations (Java, C, Python, Perl, .NET) [66]. Message-level security is 200 times slower than transport-level security in GT3 according to [67].

In article *Alternative Software Stacks for OGSA-based Grids* [68] Humphrey et. al. publish their experience with developing implementation of WSRF and WS-Notification family of standards on .NET with implementation of competing standards WS-Transfer and WS-Eventing. They conclude, that WSRF and WS-Notification are more complex than WS-Transfer and WS-Eventing, so they are harder to implement and implementations are less likely to be interoperable. However WS-Transfer does not distinguish between a resource and resource properties, both being the same XML document. If a resource is an active entity, like a running job, this difference is important. In WS-Notification, subscription is associated with a resource. In WS-Eventing,a subscription is associated with a service.

"Evaluation of UDDI as a Provider of Resource Discovery Services for OGSA-based Grids [50] Benchmark Suite for SOAP-based Communication in Grid Web Services [69].

4.3 Beyond WSRF - Resources and Events

In March 2006, IBM, HP, Intel and Microsoft announced a roadmap for convergence of some WS-* standards [55], which are important for grids. The WS-ResourceFramework/WS-Notification families of standards backed by IBM, HP and others, were competing with WS-Transfer/WS-Eventing family of standards backed by Microsoft, Intel and others. New standards will be developed, layered on WS-Transfer/WS-Eventing family, which will incorporate ideas from WS-ResourceFramework/WS-Notification. Namely

- new **WS-TransferAddendum** extends WS-Transfer, which defines a SOAP-based protocol for accessing XML representations of Web service-based resources, by adding Get', Put' and Create' messages, which allow to specify a subset of a resource. WS-Transfer is a SOAP version of HTTP protocol.
- new version of **WS-MetadataExchange 1.1** which better integrates with WS-Transfer.
- new **WS-ResourceTransfer** provides some concepts from WS-ResourceFramework, namely partial updates, partial retrieval, metadata and lifetime of resources, but is build on top of WS-Transfer, WS-TransferAddendum, WS-Enumeration and WS-MetadataExchange 1.1.
- new **WS-EventNotification** extends WS-Eventing by adding ideas from WS-Notification and references WS-ResourceTransfer

Thus WSRF (WS-ResourceFramework) will be replaced by WS-ResourceTranfer and its underlying standards. And WS-Notification will be replaced by WS-EventNotification.

Ian Foster stated in a comment on that announcement [70], that Globus Toolkit will implement these new standards and provide a migration path for current GT4-based applications.

5 P2P

In the future, the grid may adopt the approach of peer-to-pper networks, which promise robustness, as is outlined in the article *Why Grid and Agents Need Each Other* [71].

6 Conclusion

This document is the result of a literature search done as a part of our PhD studies. It provides an overview of the current state of Service Oriented Architecture, web services and service oriented grid. However the topic is so large and quickly evolving, that this document cannot claim to be complete.

References

[1] Ian Foster and Carl Kesselman. Computational Grids. In *The Grid: Blueprint for a New Computing Infrastructure*, chapter 2. Morgan-Kaufman, 1999. `http://www.globus.org/alliance/publications/papers.php#chapter2`.

[2] Ian Foster, Carl Kesselman, and Steven Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International J. Supercomputer Applications*, 15(3):200–222, 2001. `http://www.globus.org/alliance/publications/papers.php#anatomy`.

[3] Ian Foster. What is the Grid? A Three Point Checklist. *GRIDToday*, 7 2002. `http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf`.

[4] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, 6 2002. `http://www.globus.org/alliance/publications/papers.php#OGSA`.

[5] Ian Foster, Carl Kesselman, Jeffrey Nick, and Steven Tuecke. Grid Services for Distributed System Integration. *Computer*, 35(6), 2002.

[6] Yefim V. Natis. Service-Oriented Architecture Scenario. http://www.gartner.com/DisplayDocument?doc_cd=114358, 2003.

[7] IBM alphaWorks: SOA and Web services: New to SOA and Web services. http://www-128.ibm.com/developerworks/soa.

[8] W3C Web Services Glossary. http://www.w3.org/TR/ws-gloss/.

[9] Wikipedia, the free encyclopedia: Service-oriented architecture. http://en.wikipedia.org/wiki/Service-oriented_architecture.

[10] OGSA Glossary of Terms. http://www.ggf.org/documents/GWD-I-E/GFD-I.044.pdf, 1 2005.

[11] Web Service Grids: An Evolutionary Approach. http://www.nesc.ac.uk/technical_papers/UKeS-2004-05.pdf, 7 2004.

[12] Savas Parastatidis, Jim Webber, Paul Watson, and Thomas Rischbeck. A Grid Application Framework based on Web Services Specifications and Practices. http://www.cs.ncl.ac.uk/research/pubs/trs/papers/825.pdf, 2003. School of Computing Science, University of Newcastle upon Tyne, technical report.

[13] RDN/LTSN resource type vocabulary. http://www.rdn.ac.uk/publications/rdn-ltsn/types/, 2003.

[14] OASIS Reference Model for Service Oriented Architecture, Committee Draft 1.0, 7 February 2006. http://www.oasis-open.org/committees/download.php/16587/wd-soa-rm-cd1ED.pdf, 2006.

[15] ZapFlash: The Third Conversation. http://www.zapthink.com/report.html?id=ZAPFLASH-2007416, 2007.

[16] OASIS SOA Reference Model Technical Committee. http://www.oasis-open.org/committees/soa-rm, 2005.

[17] Hao He. What is Service-Oriented Architecture? http://webservices.xml.com/pub/a/ws/2003/09/30/soa.html, 2003. O'Reilly webservices.xml.com.

[18] Joshy Joseph, Mark Ernest, and Craig Fellenstein. Evolution of grid computing architecture and grid adoption models. *IBM Systems Journal*, 43(4), 2004. http://www.research.ibm.com/journal/sj/434/joseph.html.

[19] BEA dev2dev: Service-oriented Architecture. http://dev2dev.bea.com/soa/.

[20] David Sprott and Lawrence Wilkes. .NET Architecture Center: Service Oriented Architecture: Understanding Service-Oriented Architecture. http://msdn.microsoft.com/architecture/soa/default.aspx?pull=/library/en-us/dnmaj/html/aj1soa.asp.

[21] Sean Landis. Service Oriented Architectures - Separating Hype From Reality. http://www.artima.com/weblogs/viewpost.jsp?thread=59975, 8 2004.

[22] Paul Prescod. Second Generation Web Services. http://webservices.xml.com/pub/a/ws/2002/02/06/rest.html, 2002. O'Reilly webservices.xml.com.

[23] European Conference on Web Services ECOWS 2005, scope. http://wscc.info/ecows2005/research/.

[24] Werner Vogels. Web Services Are Not Distributed Objects. *IEEE Internet Computing*, 7 (6), 2003. http://csdl.computer.org/dl/mags/ic/2003/06/w6059.htm.

[25] Dan Gisolfi. Web services architect, Part 3: Is Web services the reincarnation of CORBA? http://www-128.ibm.com/developerworks/library/ws-arc3/, 2001. IBM developerWorks.

[26] Samuel Kendall, Jim Waldo, Ann Wollrath, and Geoff Wyant. Note on Distributed Computing. http://research.sun.com/techrep/1994/abstract-29.html, 1994.

[27] D. Caromel and H. Ludovic. *A Theory of Distributed Objects*. Springer, 2005. ISBN 3-540-20866-6.

[28] Web Services Architecture. http://www.w3.org/TR/ws-arch/, 2004. W3C Web Services Architecture Working Group.

[29] Savas Parastatidis, Jim Webber, Paul Watson, and Thomas Rischbeck. WS-GAF: a framework for building Grid applications using Web Services. *Concurrency and Computation: Practice and Experience*, 17(2-4), 2004. http://aspen.ucs.indiana.edu/CCPEwebresource/c816webber/c816wsgaf.pdf.

[30] Pat Helland. Data on the Outside Versus Data on the Inside. In *CIDR, Second Biennial Conference on Innovative Data Systems Research*, pages 144–153, Asilomar, CA, USA, 1 2005. http://www-db.cs.wisc.edu/cidr/cidr2005/papers/P12.pdf.

[31] Ian Foster, Jeffrey Frey, Steve Graham, Steve Tuecke, Karl Czajkowski, Don Ferguson, Frank Leymann, Martin Nally, Igor Sedukhin, David Snelling, Tony Storey, William Vambenepe, and Sanjiva Weerawarana. Modeling Stateful Resources with Web Services. http://www-128.ibm.com/developerworks/library/ws-resource/ws-modelingresources.pdf, 2004.

[32] Latha Srinivasan and Jem Treadwell. An overview of service-oriented architecture, Web services and grid computing. http://devresource.hp.com/drc/technical_papers/grid_soa/index.jsp, 11 2005. HP Software Global Business Unit.

[33] Project Artemis Deliverable 3.1.1.1: Review of the state-of-the-Art - Web Service Technologies. http://www.srdc.metu.edu.tr/webpage/projects/artemis/documents/D3.1.1.1-SOAv1.2_WebServiceTechnologies.doc, 2004.

[34] Web Services Description Language (WSDL) 1.1, W3C Note. http://www.w3.org/TR/wsdl, 2001.

[35] Web Services Description Language 2.0: Core Language.

[36] Savas Parastatidis, Simon Woodman, Jim Webber, Dean Kuo, and Paul Greenfield. Asynchronous Messaging between Web Services Using SSDL. *IEEE Internet Computing*, 10(1): 26–39, 1 2006. http://csdl.computer.org/dl/mags/ic/2006/01/w1026.pdf.

[37] Simple Object Access Protocol (SOAP) 1.1, W3C Note. http://www.w3.org/TR/2000/NOTE-SOAP-20000508/, 2000.

[38] Simple Object Access Protocol (SOAP) 1.2, W3C Recommendation. http://www.w3.org/TR/soap12/, 2003.

[39] Ondřej Krajíček. Invocation Architecture for Scalable High-Performance Web Services. Master's thesis, Faculty of Informatics, Masaryk University, Botanická 68a, Brno, Czech Republic, 2004.

[40] Hessian Binary Web Service Protocol.

[41] Fast Web Services.

[42] Direct Internet Message Encapsulation.

[43] SOAP Message Transfer Optimization Mechanism.

[44] Web Services Interoperability (WS-I) Basic Profile Version 1.1. `http://www.ws-i.org/Profiles/BasicProfile-1.1.html`, 2004.

[45] W3C Workshop on XML Schema 1.0 User Experiences: IBM's Position. `http://www.w3.org/2005/05/25-schema/IBM.html`, 5 2005.

[46] W3C Workshop on XML Schema 1.0 User Experiences: Microsoft's Position. `http://www.w3.org/2005/05/25-schema/microsoft.html`, 5 2005.

[47] Steve Loughran and Edmund Smith. Rethinking the Java SOAP Stack. In *IEEE International Conference on Web Services (ICWS) 2005*, 5 2005.

[48] Brent Sleeper. The Evolution of UDDI: UDDI.org White Paper. `http://uddi.org/pubs/the_evolution_of_uddi_20020719.pdf`, 2002.

[49] UDDI is a Dead Parrot. `http://weblog.infoworld.com/realworldsoa/archives/2005/12/uddi_is_a_dead.html`, 12 2005.

[50] Edward Benson, Glenn Wasson, and Marty Humphrey. Evaluation of UDDI as a Provider of Resource Discovery Services for OGSA-based Grids. In *2006 International Parallel and Distributed Processing Symposium (IPDPS 2006)*, Rhodes Island, Greece, 4 2006. ISBN 1-59593-061-2. `http://www.cs.virginia.edu/~humphrey/papers/UDDI_Grids.pdf`.

[51] Paul Denning. Annotated List of Web Services Specs. `http://lists.w3.org/Archives/Public/www-ws-arch/2004Feb/0022.html`, 2004.

[52] Tim Bray. The Loyal WS-Opposition. `http://www.tbray.org/ongoing/When/200x/2004/09/18/WS-Oppo`, 2004.

[53] Jon MacLaren, Maria Hernandez, Ioannis Kotsiopoulos, and Colin Puleston. Ontogrid Project: Deliverable D1.1: Grid Fabric and State-of-the-Art. `http://www.ontogrid.net/`, 2 2005.

[54] Jon Udell. Building SOA your way. `http://www.infoworld.com/article/05/09/12/37FEsoaevolve_1.html`, 9 2005.

[55] Kevin Cline et al. Toward Converging Web Service Standards for Resources, Events, and Management. `http://devresource.hp.com/drc/specifications/wsm/wsm.pdf`, 3 2006. A Joint White Paper from HP, IBM, Intel and Microsoft.

[56] Greg Goth. Critics Say Web Services Need a REST. *IEEE Distributed Systems Online*, 5 (12), 2004. `http://csdl2.computer.org/comp/mags/ds/2004/12/oz001.pdf`.

[57] Don Box. Transcript: SOAP: Service-Oriented Architecture and Programming, Pt. 2. http://msdn.microsoft.com/msdntv/transcripts/20030902SOAPDBTranscript.aspx, 2003.

[58] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*, chapter 5 Representational State Transfer (REST). PhD Dissertation, University of California at Irvine, 2000. http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm.

[59] Global Grid Forum (GGF) Open Grid Services Architecture Working Group (OGSA-WG). https://forge.gridforum.org/projects/ogsa-wg.

[60] Defining the Grid: A Roadmap for OGSA Standards. http://www.gridforum.org/documents/GFD.53.pdf, 9 2005.

[61] From Open Grid Services Infrastructure to WS-Resource Framework: Refactoring & Evolution. http://www.globus.org/wsrf/specs/ogsi_to_wsrf_1.0.pdf, 2004.

[62] Ian Foster, Karl Czajkowski, Donald Ferguson, Jeffrey Frey, Steve Graham, Tom Maguire, David Snelling, and Steven Tuecke. Modeling and Managing State in Distributed Systems: The Role of OGSI and WSRF. In *Proceedings of the IEEE*, volume 93, pages 604–612, 2005. http://www.unigrids.org/papers/wsrf.pdf.

[63] Savas Parastatidis. comments about WSRF and WS-GAF. http://savas.parastatidis.name/2005/08/23/f25de0b1-68a1-4edf-a2b3-0e835b3ca2d2.aspx, 2005.

[64] Web Services Resource Framework. http://www.globus.org/wsrf/, 2004.

[65] Marty Humprey, Gelnn Wasson, Mark Morgan, and Norm Beekwilder. An Early Evaluation of WSRF and WS-Notification via WSRF.NET. In *Fifth IEEE/ACM International Workhop on Grid Computing (associated with Supercomputing 2004)*, Pittsburgh, PA, 11 2004. ISBN 0-7695-2256-4.

[66] Marty Humphrey, Glenn Wasson, Keith Jackson, Joshua Boverhof, Matt Rodriguez, Jarek Gawor, Joe Bester, Sam Lang, Ian Foster, Sam Meder, Stephen Pickles, and Mark McKeown. State and Events for Web Services: A Comparison of Five WS-Resource Framework and WS-Notification Implementations. In *14th IEEE International Symposium on High Performance Distributed Computing (HPDC-14)*, Research Triangle Park, NC, 7 2005.

[67] Satoshi Shirasuna, Aleksander Slomiski, Liang Fang, and Dennis Gannon. Performance Comparison of Security Mechanisms for Grid Services. In *Fifth IEEE/ACM International Workhop on Grid Computing (associated with Supercomputing 2004)*, Pittsburgh, PA, 11 2004. ISBN 0-7695-2256-4.

[68] Marty Humprey, Glenn Wasson, Yuliyan Kiryakov, Snag-Min Park, David Del Vecchio, Norm Beekwilder, and Jim Gray. Alternative Software Stacks for OGSA-based Grids. In *IEEE/ACM SC'05 Conference*, Seattle, WA, 11 2005. ISBN 1-59593-061-2.

[69] Michael R. Head, Madhusudhan Govindaraju, Aleksander Slominski, Pu Liu, Nayef Abu-Ghazaleh, Robert van Engelen, Kenneth Chiu, and Michael J. Lewis. A Benchmark Suite for SOAP-based Communication in Grid Web Services. In *IEEE/ACM SC'05 Conference*, Seattle, WA, 11 2005. ISBN 1-59593-061-2.

[70] Ian Foster. The Holy Grail: Industry-Wide System Management Standards at Last? http://www.globus.org/wsrf/convergence.php, 3 2006.

[71] Ian Foster, Nicholas Jennings, and Carl Kesselman. Brain Meets Brawn: Why Grid and Agents Need Each Other. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, 2004.

[72] David Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15(12), 12 1972.

[73] Raissa Medeiros and Jacques Sauvé Walfredo Cirne, Farancisco Brasileiro. Faults in Grids: Why are they so bad and What can be done about it? In *Fourth International Workshop on Grid Computing*, 2003.

[74] Vijay Dialani, Simon Miles, Luc Moreau, David De Roure, and Michael Luck. Transparent Fault Tolerance for Web Services Based Architectures. *Lecture Notes in Computer Science*, 2400, 2002.

[75] Borja Sotomayor. Towards a service-oriented Grid. `http://people.cs.uchicago.edu/~borja/lectures/towards_service_oriented_grid.pdf`.

[76] Michael Uschold. Where are the Semantics in the Semantic Web? *AI Magazine*, 24(3):25–36, September 2003. `http://www.starlab.vub.ac.be/WhereAreSemantics-AI-Mag-FinalSubmittedVersion2.pdf`.

[77] Vojtěch Svátek. Ontologie a WWW. In *DATAKON*, 2002. ISBN 80-210-2958-7.

[78] Felix Heine, Matthias Hovestadt, and Odej Kao. Towards Ontology-Driven P2P Grid Resource Discovery. In Rajkumar Buyya, editor, *Fifth IEEE/ACM International Workshop on Grid Computing*, pages 76–85, Pittsburgh, Pennsylvania, November 2004. ISBN 0-7695-2256-4.

[79] Matthew Horridge. A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools, August 2004. `http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf`.

[80] Preeda Rajasekaran, John Miller, Kunal Verma, and Amit Sheth. Enhancing Web Services Description and Discovery to Facilitate Composition. *Lecture Notes in Computer Science*, 3387:55–68, January 2005. `http://lsdis.cs.uga.edu/lib/download/swswpc04.pdf`.

[81] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, May 2001.

[82] Jorge Cardoso and Amit Sheth. Introduction to Semantic Web Services and Web Process Composition. *Lecture Notes in Computer Science*, 3387, January 2005. `http://lsdis.cs.uga.edu/library/download/CS05-SWSWPC-chapter.pdf`.

[83] Glen Wasson and Marty Humphrey. Policy and Enforcement in Virtual Organizations. In *Proceedings of the FOurth International Workshop on Grid Computing*, 2003. ISBN 0-7695-2026-X.

[84] Glen Wasson, Norm Beekwilder, Mark organ, and Marty Humphrey. WS–ResourceFramework on .NET. In *Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing (HPDC'04)*, 2004. ISBN 0-7803-2175-4.

[85] Marty Humphrey. An Early Evaluation of WSRF and WS-Notification via WSRF.NET. In *Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04)*, pages 172–181, 2004.

[86] Glenn Wasson and Marty Humphrey. Exploiting WSRF and WSRF.NET for Remote Job Execution in Grid Environments. In *19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) – Papers*, 2005.

[87] Glenn Wasson and Marty Humphrey. Architectural Foundations of WSRF.NET. *International Journal of Web Services Research*, 2(April–June):83–97, 2005.

[88] M. Humphrey, G. Wasson, K. Jackson, J. Boverhof, M. Rodriguez, J. Gawor, S. Lang, J. Bester, I. Foster, S. Meder, S. Pickles, , and M. McKeown. State and Events for Web Services: A Comparison of Five WS-Resource Framework and WS-Notification Implementations. Presentation at The 14th IEEE International Symposium on High Performance Distributed Computing (HPDC-14), 2005. The corresponding paper was not found to be publicly avaiable yet. Still it is referenced from the homepage.

[89] Marty Humphrey, Glenn Wasson, Yuliyan Kiryakov, Sang-Min Park, David Del Vecchio, and Norm Beekwilder. Alternative Software Stacks for OGSA-based Grids. In *Proceedings of the Fifth IEEE/ACM Internation Workshop on Grid Computing (GRID'04)*, 2005.

[90] Sangmi Lee Pallickara, Beth Plale, Scott Jensen, and Yiming Sun. Monitoring Access to Stateful Resources in Grid Environments. 2005.

[91] Anis Charfi and Mira Mezini. An Aspect-based Process Container for BPEL. In *Proceedings of the 1st workshop on Aspect oriented middleware development*, 2005.

[92] Xiang Fu, Tevfik Bultan, and Jianwen Su. Analysis of interacting BPEL Web Services. In *Proceedings of the 13th international conference on World Wide Web*, 2004.

[93] G.Conte M. Amoretti, F. Zanichelli. SP2A: a service-oriented framework for P2P-based Grids. In *Proceedings of the 3rd international workshop on Middleware for grid computing*, 2005.

[94] M. Zhao, V. Chadha, and R. Figueiredo. Supporting Application-Tailored Grid File System Sessions with WSRF-Based Services. In *Proceedings of 14th IEEE International Symposium on High Performance Distributed Computing (HPDC'05)*, 2005.

[95] I. Foster. Ian Foster on Recent Changes in the Grid Community. *IEEE Distributed Systems Onlone*, 5(2), 2004.

[96] H. Hosoya, A. Frisch, and G. Castagna. Parametric Polymorphism for XML. In *Proceedings of the 32nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 50–62, 2005.

[97] XML Optimized Packaging.