TOWARDS DEPENDABLE GRIDS

Anh Nguyen-Tuong, Andrew S. Grimshaw, Glenn Wasson, Marty Humphrey, and John C. Knight
{nguyen, grimshaw, wasson, humphrey, knight}@virginia.edu

Technical Report CS-2004-11
Department of Computer Science, University of Virginia

ABSTRACT

To date, grids (a form of *distributed system*) have been used to aggregate resources for performance-starved applications typically resulting from scientific enquiry. Grids should not just be facilitating advances in science and engineering; rather they should also be making an impact on our daily lives by enabling sophisticated applications such as new consumer services and support for homeland defense. For example, grids providing financial services should be seamlessly interconnected with grids that support telecommunications, and grids providing health care services should be seamlessly interconnected with both financial services and telecommunications. This is not possible today because the poor grid dependability—which is tolerated by scientific users—would be unacceptable in critical infrastructure applications. This project aims at correcting this problem by developing technology that will allow grids to be used to provide services upon which society can depend. Grids must be engineered both to achieve high dependability and to permit assurance that high dependability has been achieved.

1 BACKGROUND AND MOTIVATION

To date, grids have been designed to aggregate resources for performance-starved applications. The goals are easy-to-use remote execution, easy-to-use access to large data sets, single sign on, etc. The prototypical user needs to execute an MPI application, perhaps for particle physics, molecular dynamics, and weather forecasting, and perhaps without caring where it executes. Or maybe the user needs to execute 1000 jobs on idle desktops. These goals are the basis of emerging grids such as GEON, LEAD, iVDGL, NEES, TeraGrid, and others. The community is doing an excellent job of providing these capabilities via software such as the NSF Middleware Initiative (NMI) software, VDT, and the NPACKage software packages.

Grids should not be facilitating just "Big Science"; rather they should be making an impact on our daily lives by enabling sophisticated applications such as new consumer services and support for homeland defense. This is simply not possible today because the goals and requirements are different. Current large-scale scientific endeavors are able to depend on the grid because scientific users are largely tolerant of the grid's: (1) often idiosyncratic behaviors; (2) unexplained or obtuse failures; and (3) inability to provide a highly available cyber infrastructure.

Current grid systems provide services that are important to society, but our dependence will grow considerably over time reaching a point where they will become interwoven within the fabric of society. They will then have become critical infrastructure. Grid systems are dependent upon one another and will become more so. Grids providing financial services, for example, will depend on grids that support telecommunications, and grids providing health care services will depend on both financial services and telecommunications. The common need throughout all the various interdependent grid user communities is high levels of dependability; users must be able to depend upon the service they receive. This dependability will include reliability in some cases, availability in others, data integrity and confidentiality (security) in essentially all services, and possibly safety in some circumstances.

There are many threats to dependable services, for example: (1) failure of a power system leading to power loss; (2) physical damage to the grid computing fabric as a result of natural events (hurricanes) or

terrorist acts; and (3) failure of system or application software, errors by system operators or malicious cyber attacks leading to the loss of services. All of this is compounded by the interdependencies between grid systems since the consequences of a specific failure might cascade throughout a set of grids.

To engineer a grid to meet the expected high dependability requirements is likely to be technically infeasible or prohibitively expensive or both. Infeasibility derives from technical issues such as the reliance on commodity operating systems that are known to be vulnerable to cyber attacks. The expense derives from the very high cost of the replication of the basic components of the system, such as those providing communications, storage and computing, that will be required if continuous availability is demanded.

The alternative approach proposed here recognizes explicitly this fundamental trade-off between dependability and the resources and technology needed to provide it by making the next generation of grids *survivable*. Informally, a survivable system is one that provides one or more alternate services (different, less dependable, or degraded) in a given operating environment if the primary service cannot be provided because of attacks or failures. The key approach behind a survivable system is that each of these alternate services is designed to cope with potentially a different class of faults. By constructing or incorporating elements of the system (such as those implementing the primary functionality) with less provision for coping with faults than normally might be preferred, a survivable approach *may take the implementation of such a system from a complexity and cost level that is infeasible to one that is feasible*. The potential loss of service that ensues is dealt with by providing alternate services when the primary implementation is unavailable. Users receive a higher value from the system either in the form of reduced cost or increased options for functionality.

A survivable approach has the added advantage that it maps naturally to a development environment in which applications are built in part by incorporating existing grid elements from *different administrative domains*, an inherent characteristic of the grid paradigm. The specification of the alternate services advocated by a survivable approach provides an intellectual framework by which to analyze and incorporate existing grid elements with the inevitably different characteristics in multiple dimensions, e.g., different policies on the availability of the provided grid facilities, different expected average and peak load demands, different security policies, and different service availability guarantees. In such an environment, the dependability requirements for a specific application will be met in part by a process of resource discovery combined with the composition of services with the requisite levels of dependability. Note that oftentimes the straightforward composition of existing services will not be sufficient to meet the stated dependability requirements. Thus, part of the proposed research will be to investigate techniques and transformations to increase the dependability of a composite set of services.

1.1 APPROACH

The future of grid technology, the implementation of which has already begun, lies in the merging of current grid services and current Web services. The result will be a powerful mechanism for the exploitation of grid concepts to support novel user services, including those that are characterized as critical infrastructure applications. The focus of grid technology today is the Open Grid Services Architecture (OGSA) being developed in the Global Grid Forum (GGF). OGSA is a Service Oriented Architecture (SOA), predicated on the use of web service standards such as the Web Service Definition Language (WSDL), the Web Service Resource Framework (WSRF), and many others [WSDL, Foster-WSRF, OASIS].

By a service-oriented architecture we mean: “a specific type of distributed system in which the discrete software agents—that must work together to implement some intended functionality—are called *services*...the description of a service in a SOA is essentially a description of the messages that are exchanged. This architecture adds the constraint of stateless connections, that is where the all the data for a given request must be in the request.” [Booth2003].

Our approach to the development of survivable grids is illustrated in Figure 1. At the bottom are the techniques commonly used to construct dependable systems: fault avoidance, fault elimination, and fault-

tolerance. These techniques need to be enhanced to deal with the service-oriented architecture world-view that pervades anticipated grid systems and to implement the survivability concept.

In the middle of the figure is the SOA. One of the hallmarks of SOA's is that of service composition: the construction of new services by composing, often dynamically, existing web services into new services. Available grid facilities change over time as circumstances change within administrative domains. Thus, for example, the hardware resources that can be provided by a specific administrative domain will depend on demands within the domain, failures that have occurred, and so on. We propose to adapt the implementation of service provided to a given user at the time that the requirements are stated. To do so, we propose to require that grid elements specify their dependability characteristics accurately via metadata or attributes. These characteristics and the associated fault model will be represented in machine interpretable documents in the Dependability Exchange and Specification Language (DESL).

At the top of Figure 1 are the applications. It is the applications that have dependability requirements. Those requirements need to both be specified and tested against the dependability characteristics of the services the applications use.

The unique characteristics of SOA's - particularly SOA's that span multiple organizational, administrative, and trust domains will require new techniques. For example, it may not be possible to query and manage a service via its managing "container" due to security issues at the site where the service is running.

To test the efficacy of our approach, we propose to develop a prototype system that will provide an opportunity to test the techniques we will develop both in a "live" environment as well as in a test harness where anticipated faults can be injected into the system. We intend to construct a dependability architecture and implementation, the Grid Dependability and Survivability Architecture (GDSA), based on a classic *data-driven* control system model. In GDSA a series of instruments will continuously monitor application and system behavior. Monitoring may be either direct polling, or subscribing to events of interest. The raw monitoring data will be processed, collected and distributed to appropriate dependability services, analyzed, and as needed both appropriate corrective actions may be taken and availability characteristics of components will be updated.

GDSA will build on our earlier work with Legion [Grimshaw1999,2003], Willow [Knight2002, Rowanhill2004] and OGSI.net [WassonG2004], as well as the standards being developed in GGF. In particular, we will be constructing the prototype as a set of Web Services that are named with WS-References, and that fit within the context of the OGSA architecture.

In summary, the proposed work will take place in the context of SOA's, web services and OGSA, and our approach to survivability will consist of five broad aspects:

- *Dependability and Survivability*
  We will bring both the rigor and techniques developed in the dependable computing field to service oriented architectures (SOA) and applications. This has three significant parts: i) identify the charac-

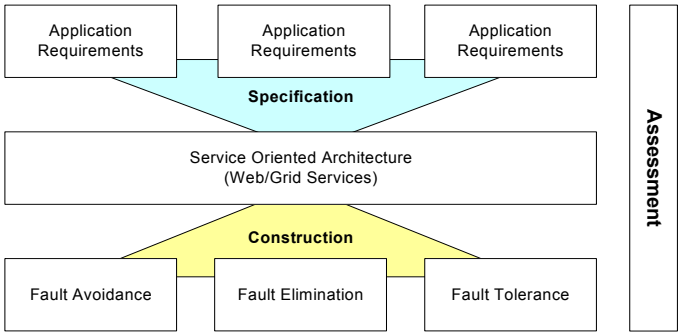


Figure 1. Specification, construction and assessment of dependable applications in service-oriented architectures

teristic service styles, their dependability requirements, their acceptable survivability trade-offs, in SOA's and exploit known techniques from the dependability literature to address the requirements; ii) develop a language for the specification of both dependability requirements and characteristics along with dependability services that exploit and "understand" the language; and iii) develop an architecture (within OGSA) for dependability and survivability.

- *Key Service Identification*
  We will identify key services within OGSA and the application use-cases that need enhanced dependability in order to meet their requirements that can use the techniques and language constructs identified in aspect one.
- *System Architecture Development*
  We will develop an open implementation of the architecture (iii above).
- *Evaluation*
  We will evaluate the resulting architecture and system in the context of real applications from the life sciences and medical informatics domains. This evaluation will be both in a live environment and in a test environment where we can control and inject faults. The application of our techniques to medical informatics will be guided through our collaboration with Dr. William A. Knaus, M.D., of the University of Virginia Health System.
- *Standards Bodies*
  We will continue to work closely with the standards communities (GGF, OASIS, W3C, DMTF) to keep abreast of standards and to offer our results and requirements to the standards process.

These five aspects will not be performed sequentially; rather we will engage in an interactive process in which we introduce the lessons learned into the research process.

1.2 MOTIVATING EXAMPLE

Consider an example from the life sciences. There is a wealth of data available, genomic data, protein data, tissue data, and clinical data on outcomes and patient background data (demographics, lifestyle, family history). To do research on multi-factorial disease requires accessing and mining the data followed by interpreting the results. Further, researchers want to use tools that work when they want them to. Unfortunately, even within a single institution the data is kept in many different departments, and much of the data is not at the same institution but at hundreds of peer institutions that do not share a large degree of trust. And making matters still worse, there are various institutional and national rules and laws that govern the use of the data which must be enforced.

To provide the highly available, cross department, cross organizational information processing environment the researchers want is a challenge. Data access must be automatic, applications powerful yet intuitive, and the system must meet high availability, confidentiality, and data integrity requirements.

This sort of requirement is not unique to the life sciences. We see similar applications and challenges as grids are deployed in industry, and in particular in homeland defense. Instead of searching for multi-factorial disease using multiple data types, the user is looking for threats or patterns of activity that might indicate or predict attack. Instead of data scattered across campus and other peer institutions, the data is scattered across thousands of police departments, law enforcement agencies, hospitals, shipping companies, intelligence agencies, and public data bases around the world. As with the life sciences, these organizations do not always trust one another, and they may have complex rules and laws that govern who can see what data for what purpose and with what priority.

Continuing the example, we note that the life-sciences grid will be extremely important in the event of a terrorist attack. Whereas a homeland-defense grid supports law enforcement, a terrorist attack might demand that the life-sciences grid provide information such as medical records of the injured, the detailed effects of pathogens, or the immediate availability of drugs to homeland defense. Since this would be tak-

ing place in the face of an attack, both the life-sciences grid and the homeland-defense grid would have to provide crucial services even when damaged or overloaded. This is precisely the type of capability that survivability will ensure.

2 Related Work

2.1 Grid Projects

The two major grid software projects in academia are Legion [Grimshaw1999, 2003, 2004] and Globus [Globus, Foster]. Both projects have their genesis in the early 1990's. Legion has extensive mechanism designed in and implemented to address fault-tolerance [NguyenTuong1996,1999] [NguyenTuong2000]. Mechanisms included automatic recovery of failed objects, object migration away from failing or overloaded hosts, stateless replication, fault-tolerant MPI libraries, and K-replication of stateful objects with automatic failover. Further the system would automatically adapt to resource availability. Early versions of Globus had a service called a "heart beat monitor" that would poll a service and notify a client if the service being monitored failed. Later versions of Globus eliminated the service.

Both the Legion team and the Globus team are now behind the Open Grid Service Architecture (OGSA) being developed in the Global Grid Forum [Foster2002]. OGSA was proposed by IBM and the Globus PI's. OGSA is a community effort with input from both the public and private sectors.

In the industrial space there are a number of products that claim to be grid. These include products from Avaki (data grid), Platform (compute grid), Sun (compute grid), United Devices (compute grid), Veridian (compute grid), HP (adaptive infrastructure), and IBM (IBM resells and has compute grid products). As a rule compute grid products provide at least rudimentary fault-tolerance by re-starting failed computations. The more sophisticated also provide management control over resource provisioning and allow the establishment of SLA's with guarantees of QoS for certain user classes. e.g., "fair share scheduling".

2.2 Industry Initiatives

The "grid" has become a hot buzzword in IT. All of the major vendors, IBM, HP, Sun, Fujitsu, and NEC have embraced and incorporated grid technology and concepts in their product lines. These vendors are also actively participating in the various standard bodies (Global Grid Forum, OASIS, W3C, DMTF) that are establishing grids standards such as OGSA and WSRF (Web Service Resource Framework). WSRF, a recently released specification proposed by IBM, HP, Tibco, Akamai and the Globus Alliance represents the convergence of concepts and compromises from the grid and web services communities.

This convergence is important to our goal of having impact on society. By basing our experimental work on such standards, we dramatically increase the odds that the fruits of our research, i.e., methodologies, concepts, tools and software artifacts, will be incorporated into the next generation of vendor products. Synergistically, the adoption of common standards by major vendors accelerates the pace of our research as it allows us to focus on the issue at hand, namely achieving dependability and survivability in grids. In addition, we will continue to both track the output of the standard bodies as well as actively participate in the shaping of grid standards.

2.3 Quality of Service in Grids

A commonly accepted definition of a grid is a system that delivers service by coordinating distributed resources using standard, open, general-purpose protocols and interfaces [Foster2002b]. Scientific grid applications often require the coupling of processing, storage, and network capabilities for good performance. Today these runs typically require manual reservation, configuration, and allocation of resources. Quality-of-service research in the grid community is focused on providing mechanisms for supporting these applications through resource provisioning and co-allocation. However, little attention in the grid community is being paid to the relationship between dependability and quality of service, i.e., the expectation that the agreed-upon quality of service will be delivered.

The DUROC [DUROC] broker provided a first step toward advanced quality of service (QoS) in Globus by implementing opportunistic co-allocation of distributed resources using the Globus Resource Allocation Manager (GRAM) [GRAM]. MPICH-G2 [MPICH-G2] uses DUROC to synchronize MPI jobs across GRAM resources. The General-purpose Architecture for Advance Reservation and Allocation (GARA) [GARA] was later developed with support for QoS reservations for different resource types. GARA has been demonstrated for network, CPU, and disk allocations but does not provide an agreement protocol or co-allocation protocol. The proposed Service Negotiation and Acquisition Protocol (SNAP) [SNAP] meets this need with support for SLA negotiation for tasks and resources.

2.4 DEPENDABILITY AND SURVIVABILITY

Many terms have been applied to grids in an informal or even misleading sense. Three of the most common are reliable, available and fault-tolerant. The intuitive notion is that one can rely on a system to do what it is supposed to do, that a system is available to perform its work, and that a system somehow tolerates faults. Of these, fault tolerance is the term that is most often misused; it narrowly draws attention on mechanisms instead of high-level requirements. For example, fault avoidance and fault elimination are techniques that can also be used to support dependability requirements. In fact, in many cases, these are preferable to fault tolerance techniques. In discussing requirements for grids (and any other systems), it is essential to have a precise set of definitions of the terms being used.

Avizienis, Laprie, and Randell have created a taxonomy of facets of what they term dependability [Avizienis2001]. Dependability in their sense, "the ability to deliver service that can justifiably be trusted," replaces informal notions of reliability and availability and include other facets that must be considered for many systems if the systems' users are to depend on them. This taxonomy has become a *de facto* standard as well as a *de jure* standard in progress through IFIP Working Group 10.4. The facets Avizienis *et al.* define are availability, reliability, safety, confidentiality, integrity, and maintainability.

The concept of *survivability* is a special case of dependability. Informally, a survivable system is one that has facilities to provide one or more alternate services (different, less dependable, or degraded) in a given operating environment despite the presence of attacks or failures. An alternate service would be required to be in effect if an event (such as some form of damage) precludes or hinders provision of the system's normal service. Several definitions of survivability can be found in the literature [Ellison1997, Knight2003]. We will use the definition of survivability as proposed by Knight et. al. [Knight2003] as it provides a rigorous and precise engineering definition that explicitly focusses on engineering trade-offs.

2.5 SECURITY

The Grid Security Infrastructure (GSI) [Foster1998] focuses on an authentication infrastructure for the Grid that is based on a Public-Key Infrastructure (PKI) [Butler2000]. GSI provides a standard programming interface for authentication, message integrity, and message confidentiality (GSSAPI), a mutual-authentication proposal based on SSL/TLS, and a delegation protocol by which a user can temporarily empower a software service to act on his/her behalf. While GSI focuses on the human-to-machine mutual authentication, the Community Authorization Service (CAS) [Pearlman2002] extends this so that a person can obtain and exercise authorization rights based on the group to which they belong. MyProxy [Novotny2001] is an on-line credential repository, which enables a user to upload a credential (such as a PKI credential for use with GSI) to a secure service, retrieving the credential later as necessary. MyProxy is particularly valuable in that it ameliorates some of the problems associated with a PKI, essentially making a PKI easier to use and manage.

While the Grid community has been developing security mechanisms [Siebenlist2002], the commercial sector has been working on a number of important security approaches based on Web Services. In April, 2002, IBM and Microsoft jointly authored a roadmap [IBM-Microsoft 2002] that identified many of the component security specifications that they believe will be the foundation of Web Services. The core specification is WS-Security, which defines basic XML for confidentiality and integrity on SOAP mes-

sages. Other specifications identified in this roadmap have either been published but have not been introduced into a standards organization yet (e.g., WS-Policy, WS-Trust, WS-Federation, WS-SecureConversation) or have not been released in public yet (e.g., WS-Privacy, WS-Authorization).

During the time that IBM and Microsoft have been working on this roadmap, the Liberty Alliance Project was formed to create open, interoperable standards for federated network identity [Liberty2003]. The goal of the Liberty Alliance is to create the open protocols by which a user can easily "link" his/her various network identities together, thereby establishing a "single sign on" (SSO). A cornerstone in the Liberty Alliance is the Security Assertion Markup Language (SAML), which is being standardized in OASIS [OASIS-SAML2003]. At this time it is not clear to what extent WS-Federation, as identified by the joint IBM-Microsoft white paper mentioned above, will interoperate with the specifications of the Liberty Alliance Project (the same can be said for WS-Authorization and the eXtensible Access Control Markup Language (XACML) [OASIS-XACML2003], which is an OASIS standard that describes both a policy language and an access control decision request/response language (both written in XML). The architecture for security in OGSA has been the focus of the OGSA Security Working Group of the GGF [Nagaratnam2003].

3 TECHNICAL APPROACH

Our goal is to enable the construction of survivable grid systems and applications via an architecture for specifying, constructing, monitoring, analyzing, and controlling the behavior of grid services under a variety of fault conditions (including attacks). Below we briefly describe our approach. We begin with a presentation of the conceptual model and its realization in the Grid Dependability and Survivability Architecture - GDSA. Once the conceptual model is clear we will dive into a description of DESL, the Dependability Exchange and Specification Language. From there we will look at the programming model presented to application developer and how we will both specify and realize survivable and dependable grid services.

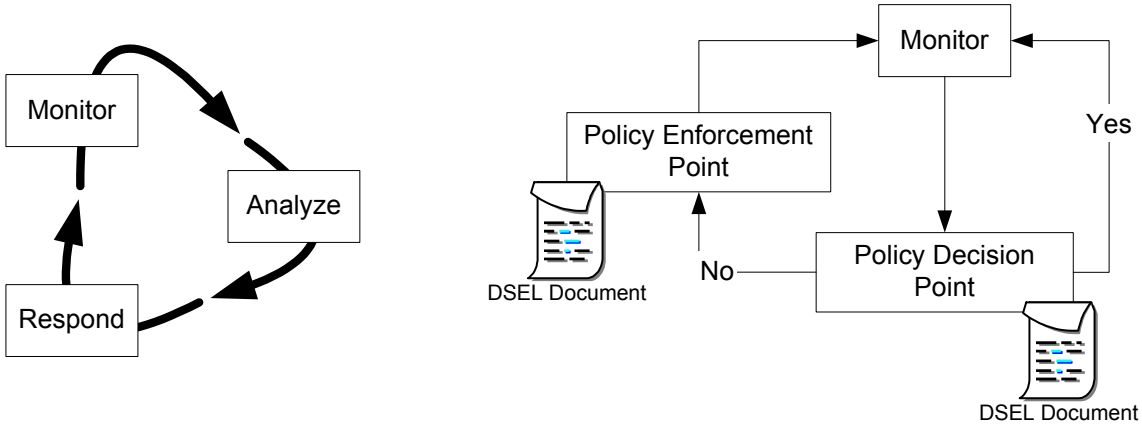
3.1 GRID DEPENDABILITY AND SURVIVABILITY ARCHITECTURE (GDSA)

GDSA is a data-driven control system architecture for dependability/survivability developed in the context of existing and evolving grid and Web Services standards. Specifically the OGSA standards from the Global Grid Forum (OGSA-Program-Execution, OGSA-Core, OGSA-Security, OGSA-logging, etc.), and from the W3C and OASIS (WS-Security, WS-Context, WS-Notification, WS-Transaction, etc.).

To achieve dependability and survivability in a SOA given a dependability specification requires an architecture for detecting faults and responding to them. The problem can be thought of as a classic control system with sensors that monitor and provide status information to an analysis module that makes decisions on how to respond using an underlying control mechanism. Think of it as dials, levers, and an agent that reads the dials and pulls the levers.

The conceptual model is shown in Figure 2. Using various forms of monitors, the system detect errors, faults, state changes, or anomalous conditions. For example, message transmission fails because the receiver is not there, or a host fails, or an un-authorized user attempts access to critical data, or a monitor detects that the load on a host is above or below some threshold. Then analyze the events in the context of DESL specifications to see if the state with respect to the specification has changed. If conditions warrant, take some form of action to recover from the situation or exploit the change in state.

The basic conceptual model by itself is insufficient. Let us dive deeper into the "analysis" loop.



(a) The conceptual model as a classic data driven control loop.

(b) More detail with the role of DESL as a specification language.

Figure 2. The conceptual model for GDSA.

The PDP is a "policy decision point" [Yavatkar2000]. We utilize this common terminology shared by many projects and approaches, for example XACML [OASIS-XACML]. The PDP evaluates the current state based on information collected by the monitors or held in databases against policies and requirements carried in DESL documents. The PDP makes a "yes/no" decision. If the decision is "yes", no action is taken and the process continues. If the decision is "no" then the current state does not meet the specified requirements and action must be taken, i.e., a "policy enforcement point" (PEP) has been reached.

The PEP has as inputs the current state and a set of specifications, rules, trade-offs, and actions to take specified in a DESL document. The objective is to take an action to bring the state back to some correct state. We stress "some" because new correct state may not be what is was before, a degraded level of service may have been selected, or perhaps even no level of service at all.

A few comments on the model before we continue. First, the PDP and PEP combination can be thought of as a sort of application or service "manager" that is responsible for keeping an application or service delivering its specified QOS. Indeed below we will often call the combination an *application manager*. Keep in mind that the "application" may in fact be the enforcement of system level property, e.g., that no priority 2 jobs run anywhere if priority 1 jobs are waiting.

Second, there will likely be many application managers running concurrently in any real system. The application managers will have different, and often conflicting, objectives. Thus, like any large set of intertwined control loops some interference is to be expected, and unexpected behaviors may emerge. This is, we believe, a fundamental property of large scale, multi-organizational, grids. Different organizations will have different objectives. We will develop this technology from the existing prototype system that is in the Willow architecture.

The monitoring aspects of GDSA are provided by the classic publish/subscribe mechanism of WS-Notification and the OGSA-Grid-Monitoring-Architecture; polling techniques that acquire system metadata and check component "liveness"; and by the use of ExoEvents [NguyenTuong2000]. Pub/sub is a widely used mechanism that needs no introduction. Similarly, polling for liveness is a time-honored technique. ExoEvents need some explanation.

ExoEvents are a mechanism for subscribing to a set of events that may occur in the call chain of a service invocation. For example, notify monitor_A if there is a "no such service" fault. The advantage to ExoEvents over classic publish/subscribe in a grid is that the set of services that may be used in executing a particular service may not be known *a priori*, making setting up the required subscriptions difficult. Further, the subscription is often needed only during the context of a particular call - not before and not after.

The "levers" (control mechanism) in GDSA consist of the additional porttypes defined services to support dependability and survivability techniques as well as the standard services provided by OGSA. Examples of additional porttypes are save-state, transfer-state, migrate, replicate, begin-epoch, end-epoch, etc. An example of a standard OGSA service call is a call on an OGSA-container to check the status of a service or instantiate a new service instance.

3.2 DEPENDABILITY EXCHANGE AND SPECIFICATION LANGUAGE (DESL)

The essence of our approach is the specification and construction of a dependability framework in the context of grid SOA's. The first critical aspect is determining how dependability requirements and characteristics will be represented, inspected, and transformed, i.e. defining a language and a set of transforms on documents in that language. We define and XML-based language, DESL, that will be used for the specification and exchange of dependability and survivability characteristics, requirements and actions. We refer to services that process DESL documents as "DESL engines".

Although the exact structure and use of DESL is part of our investigation, we imagine two use cases for DESL documents. First, DESL may be used as a means to export a service or application's dependability and survivability characteristics. Clients may examine these documents to determine if they wish to engage a service. Service composition engines may use these documents to determine which services to connect in order to meet the overall dependability properties of an application. DESL may also be used to express the dependability and survivability requirements that a service requires of its hosting environment or its clients. For example, a DESL document may describe how its hosting environment should manage a group of replicants or when to rejuvenate a service [Trivedi2000]. In addition, DESL can describe required properties of a service's client, e.g. clients must be in the same survivability mode as the service to invoke methods on the service[1].

We can further refine the DESL language as used to export properties of a service. Consider the DESL document used to describe the grid service MyProxy (or a MyProxy-like service that might be available in the near term) [Novotny2001]. The DESL document in Figure 3 describes the supported "survivability modes" of the services as well as dependability properties of the service, both static (e.g. the software process used to build this service) and dynamic (e.g. the availability of the service, perhaps measured by the underlying container) in the `<Supports>` section. The `<SurvivabilityModes>` section of the document contains tags to describe the operations/actions that take place when the service is in that mode, the trigger event(s) that cause a service to get into that mode, and some description of the effects that users of the service can expect when the service is in that mode, i.e. what is the "cost" of being in a given mode. In the above example, when a national emergency is declared by the Department of Homeland Security, the MyProxy service will issue limited credentials to non-essential personnel (the effect being to prioritize grid operations from essential clients). The `<Requires>` section, shown for completeness, would contain any information needed by the MyProxy service from the fabric on which it depends (e.g. other services, or its underlying container) or from clients that access it.

Two important security concerns should be noted with respect to DESL documents. First documents or sections of documents may need to be signed (possibly by multiple authorities) to prove their legitimacy. Second, certain aspects services that may be expressed in DESL are subject to privacy concerns. Not all sections of a DESL document would be made available for public consumption.

Documents in DESL will be exposed, collected, transformed, and analyzed by "dependability services" such as the application managers above that are responsible both for making dependability assertions (based on the compositions of services that are used), for reading and interpreting assertions made by other services, and for making choices about trade-offs among multiple service implementation options.

---

1. In this use case, DESL can be seen as a more detailed version of WS-Policy [Box2003] (as WS-SecurityPolicy [Della-Libera2002] is for the security domain).

```
<DESL xmlns:desl="http://gcg.cs.virginia.edu/DESL/02-21-04">
   <desl:Supports>
      <desl:SurvivabilityModes>
         <desl:Mode name="Normal">
            <desl:Operations>... normal delegation of credentials ... </>
         </desl:Mode>
         <desl:Mode name="NationalEmergency">
            <desl:Operations>
               ... limited delegation credentials supplied to non-essential personnel
            </desl:Operations>
            <desl:Triggers> ... order from Dept. Homeland Security </>
            <desl:Effects>
               ... description of "cost" of this mode, e.g. reduced service for some
            </desl:Effects>
         </desl:Mode>
      </desl:SurvivabilityModes>
      <desl:DependabilityInformation>
         <SoftwareProcess> ... </SoftwareProcess>
         <Availability> ... </Availability>
      </desl:DependabilityInformation>
   </desl:Supports>
   <desl:Requires>
         ... requirements of the service in terms of sub-services, performance, etc.
   </desl:Requires>
</DESL>
```

Figure 3. A sample DESL document for MyProxy illustrating the basic document structure and the sub-components.

In Figure 4 we depict a situation where a number of applications each of which uses different subsets of services in their implementation.[1] Each application has a set of dependability requirements and dependability assertions. These are represented as meta-data for the services in DESL documents.

The DESL documents are used by dependability services to asses whether requirements can be met, to aid in service selection when different capabilities are available, and to interact with the overall control architecture. These dependability services may be directly embedded in the client application as shown, or may be completely separate services.

3.2.1 SUPPORT FOR GRID PROGRAMMERS

A key issue in the use of DESL documents is how they are generated. Automation can be used to assist grid service programmers both in generating the DESL document that a service exports (the `<Supports>` section) and in responding to DESL-described requirements from, for example, clients or those who deploy services (the `<Requires>` section).

Many of today's web and grid service programming environments provide a container which "holds" services and allows them to take advantage of base functionality that is outside the scope of the service logic (e.g. handling transport protocols or message serialization) [Microsoft2004] [Tomcat2004] [WassonG2004] [JBoss2004]. This functionality of the container can be extended to support dependability and survivability requirements defined in DESL. In our work on OGSI.NET [WassonG2003,2004], we provided grid service programmers with a mechanism to annotate their service logic with meta-data that the container could use to determine how to expose that service logic to the grid. For example, service state could be marked such that the container would automatically expose it using OGSI's well-defined SDE interface [Tuecke2003].

1. In general, rather than a two-tier architecture as shown here for simplicity, there will be n-tiers, where n will vary between different applications - and sometimes vary at run-time.

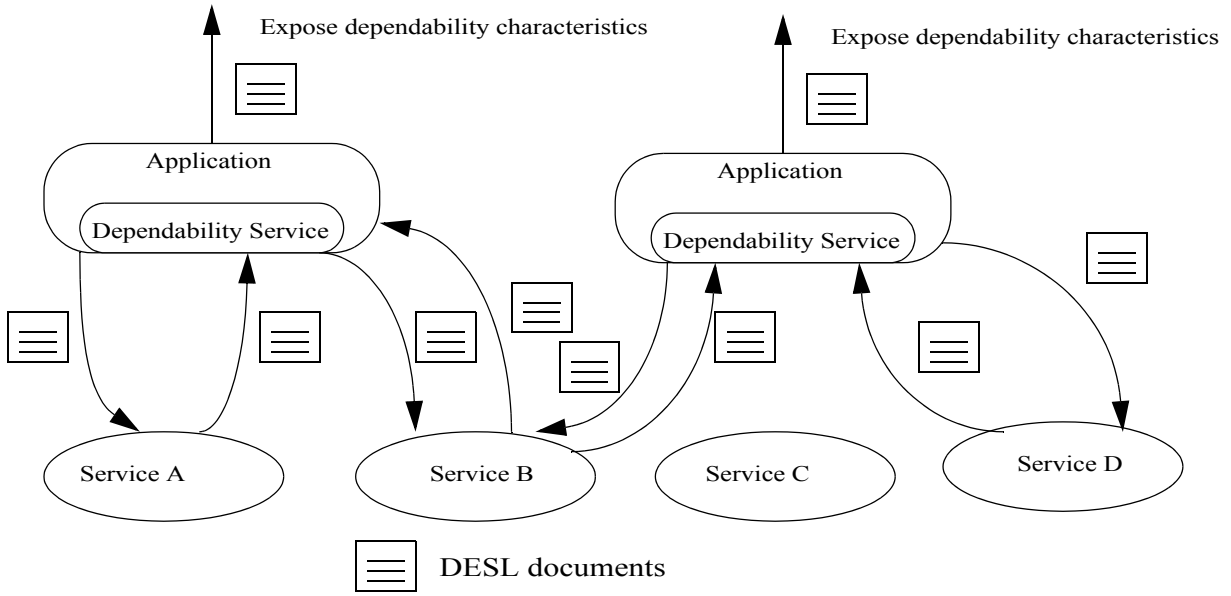


Figure 4. DESL documents are used to specify requirements, characteristic, and action rules. Note that each service may be accessed my many different applications, and export a different DESL document to

We propose to extend the meta-data annotation ideas from OGSI.NET to support the dependability and survivability requirements expressed in a service's DESL document. For example, programmers may provide meta-data "hints" to the container about how to conduct a software rejuvenation process on a particular service. The container may decide to being this process because of a DESL document that requires the service to have a certain availability.

A service's container can also help with the generation of a DESL document that describes a service. The `<DependabilityInformation>` section may contain static properties of resources used by the service that impact dependability (e.g. the OS of the service's host), but which the service author will not know. The container can easily determine these from the deployment environment. In general, the container may be able to monitor the service to determine dynamic information, such as availability, and automatically add this to a service's DESL document as well.

Some portions of the DESL generation process may be amenable to static tooling. For example, annotations in a service's logic may allow tools to generate the `<SurvivabilityModes>` section of a DESL document in much the same manner as today's WSDL generation tools create a service's WSDL document from the service code. While we do not believe it will be possible to express the full richness of the to-be-devised DESL language by static inspection of service code (even with embedded programmer meta-data), tools may be useful in generating an outline of a DESL document that the service programmer and/or deployer can flesh out.

3.3 Engineering Dependable and Survivable Grids

A critical obstacle to constructing dependable and survivable grids is that grid service developers are experts in neither dependable systems nor the protocols and messaging standards that run today's service-oriented grids. A supporting cast of tools, libraries and other grid services with which developers can turn their service logic into dependable and survivable grid services must be provided. To the extent possible, developers should only need to implement their application logic. However, the introduction of survivability concepts will require the mapping of survivability modes to grid services. In some cases, this mapping can be automatically performed by the container on behalf of the service (cf. Section 3.2.1), e.g., to enforce policies that limit external resource consumption or policies that can be enforced at the message level. In other cases, the developer will need to program the service to understand the various modes supported. In

yet other cases, the mapping can be implemented by placing a proxy in front of existing services; this proxy would intercept a request to transition to mode X and map it to messages on the back end services.

Figure 5 illustrates a possible structure for a survivable grid application. The application logic is implemented by a set of communicating grid services labelled S1 through S4. The shaded region surrounding the services S1-S4 denotes that they are contained within a hosting environment, such as .net or J2EE. All other services represent the supporting cast needed to create and execute the application.

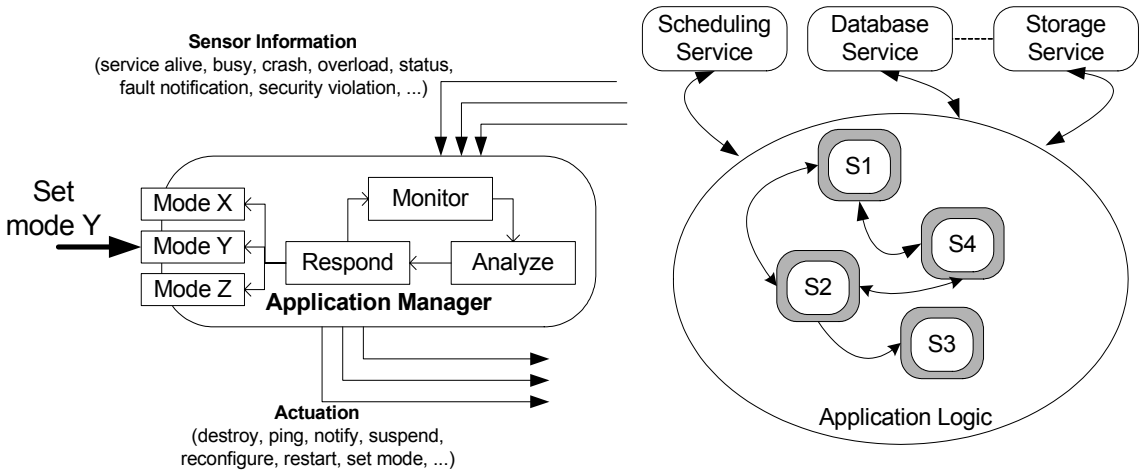


Figure 5. Structure of a survivable grid application. Services S1-S4 represent the actual application logic. The other services provide support essential to the execution of the underlying application.

The application manager plays a vital role in the management of the application. It implements a DSEL engine and is responsible for: (a) receiving information from sensors, e.g., status information such as network and host load, network bandwidth prediction, liveness notification for S1-S4, fault notifications, security violations; (b) analyzing the stream of sensor information and taking appropriate actions, e.g., transitioning from one survivability mode to another; and then (c) issuing actuation commands, i.e., sending messages to either application services S1-S4 or other services. Note that transitions between survivability modes can be the result of external (and authorized) requests, themselves possibly issued as a result of another control loop, or as the result of an administrative decision to transition all applications within a grid domain to a given operating mode, e.g., "National Emergency".

4 RESEARCH AGENDA

Building a real, working system provides perhaps the best way to find out what really works, and what crucial details were overlooked. Our own experience with Legion, Willow and OSGI.net demonstrated the difference between theoretical models and what can be achieved on live systems.

The amount of work to construct a new system from scratch is significant. We prefer to stand on the shoulders of others rather than their toes. We will build our demonstration and evaluation system as a set of OGSA grid services using public implementations of OGSA services and tooling such as Globus Toolkit 5 (GT5). We will further leverage our rather extensive grid and survivability experience gained from Legion, OGSI.Net, GT3, Willow, and the PI's commercial grid experience at Avaki.

Because we are working within OGSA we do not have to build a whole system. Instead we can focus our efforts developing the fault model, developing DESL, developing specifications and requirements for critical services, implementing dependability and survivability techniques, and developing application managers that implement the DESL engines. The development process will be iterative, developing DESL syntax and semantics, developing and testing implementations, and then back again for more complex cases. Below we expand briefly on each.

Develop and Evaluate Grid Fault Models. The first step is to extend fault models developed for traditional distributed systems to service-oriented grid architectures. These will be extended to include new classes of faults, such as service failure arising out of actions undertaken by local administrators, threats arising out of an environment in which grid users both compete and collaborate, rogue services that implement man-in-the-middle attacks, and security attacks that target grid services and coopt their behavior. We will evaluate our models by using them to drive survivability specifications for a range of grid applications deployed over a variety of grid fabrics, e.g., grids over the public Internet, grids over private networks, grids with wireless components, to approach a degree of completeness. We will then use these fault models to drive the development of survivability specifications for a selected set of grid services, e.g., MyProxy.

Develop and Evaluate DESL. DESL is not complete, nor are the mechanisms and negotiation protocols well understood at this stage. To make DESL work we intend to address the following questions:

- What are the nouns, verbs, adjectives, properties, and semantics of DESL? How are DESL documents structured? How are trade-offs represented?
- How do we automatically reason about DESL documents? For example, if multiple services are being composed, how do we compose a joint document? How do we determine if a service meets the requirements of a client beyond a straight textual match?
- How are DESL documents generated? By hand? Automatically by tools? By observing historical behavior?
- How do we measure and discover availability, reliability and other dependability facets?
- How do we ensure that policy and mechanism are clearly separated?
- How do these documents fit in the emerging WS-Agreement standards [Czajkowski2004]]?

The development of DESL and effective DESL-engines will be one of the most significant accomplishments of this work. The development will include syntactic development efforts, and more importantly, the semantic model for describing and handling dependability and survivability requirements.

Develop and Evaluate Application Managers. We will develop application managers, along with their associated libraries and DESL documents, and evaluate the extent to which they can be made generic. Our conjecture is that only a small set of application managers will be needed to manage grid applications. We will test our conjecture against such key OGSA services as MyProxy and the OGSA data services.

Build dependable and survivable grid services. We will develop containers to support dependability and survivability. In particular, we will provide support for: (a) the automatic generation of DSEL documents from source code annotations, and (b) the enforcement of policies contained in DSEL documents. We also envision building a set of common services, either from scratch or by extending existing services, to support dependability and survivability requirements. Examples of such services include DSEL support tools such as analysis engines, failure detectors, intrusion detectors and resource discovery services.

We will initially incorporate techniques taken from the dependability literature and our earlier work in Willow and Legion, and then investigate additional techniques as warranted by the dependability requirements:

Restart and Rebind. When a service instance fails (ceases processing messages) it is detected either by a client or by an explicit monitoring service. The application manager for that service then starts a new service instance and generates a new WS-address for the new instance to give to clients that can no longer use the old WS-address. In the case of stateful services the new service is given a handle to the service state.

Availability and Degraded Service. A service is replicated to provide higher availability. Synchronization is performed periodically between the primary and its replicas. On failure of the primary a *degraded* service is made available to clients. The service is degraded in one or both of two ways, the data held by the replicant may be slightly out of date, or, the service may transition to a “read-only” mode until the primary is once again available.

Replication for Performance. In the case of stateless services there are no consistency issues. However, the application manager can instantiate new instances of the service to handle performance specification errors, i.e., the service is overloaded and not responding fast enough.

Security Enforcement (MayI [Chapin1999b]). Not all errors are caused by equipment or process death. Security policy specification enforcement must also be monitored. MayI and associated security events can be used as policy enforcement points for local access control as well as a mechanism for monitoring for anomalous events or attacks against a set of services. For example, a set of services being monitored send information on all access to a application manager that looks for anomalous patterns of access. The patterns may include both unauthorized accesses, as well as abnormal patterns for authorized users that may indicate a compromise.

4.1 TEST AND EVALUATION

Testing is crucial to ensure that our techniques deliver what is specified. We will construct a test lab that spans UVA and NCSA to test the resulting service implementations to see how they respond to both injected/simulated errors in the test lab and in the deployed infrastructure.

Testing requires a test plan to ensure good coverage and to ensure that the right things are tested. We have divided our test plan into four types of tests. Of course as the project develops, the test plan will be both more fully developed and modified as GDSA and DESL change. These are fault-recovery tests, survivability tests, and the construction of an automatic test generator.

Fault-recovery. We will test the fault-recovery capabilities by injecting faults (failed hosts, large synthetic loads, certificate errors, and attacks on the security infrastructure, etc.) into applications and OGSA system components in our testbed. The resulting behavior will be compared against the specified behavior.

Survivability testing. Similarly to fault-recovery testing we will inject faults into applications and components that have defined survivability properties, e.g., degraded service, and compare the resulting behavior against the specifications.

Auto test generation. An interesting approach to testing is presented by the use of DESL to specify dependability and survivability characteristics under different circumstances. We will experiment with using the DESL documents to automatically generate and run tests (faults to generate, expected behavior).

5 SUMMARY

As society increasingly becomes dependent on grid systems, it is imperative that grids be engineered both to achieve high dependability and to permit assurance that high dependability has been achieved. We have outlined a program and framework for realizing our goal of engineering dependable service-oriented grid systems, based on the concept of *survivability* as a means to achieve dependability. The salient feature of a survivable approach is that it recognizes and exploits the fundamental trade-off between dependability and the resources and technology needed to provide it. We have proposed an architecture to implement survivability concepts—the Grid Dependability and Survivability Architecture (GDSA). GDSA is based on a classic data-driven control system model in which the state of the grid is continuously monitored and actions effected to achieve the requisite levels of dependability. In GDSA, dependability policies and requirements are represented and manipulated in machine-interpretable documents, in a proposed novel XML-based language, the Dependability Exchange and Specification Language (DESL).

6 REFERENCES

[Alliance2002] Alliance: National Computational Science Alliance. 2002. http://www2.ncsa.uiuc.edu/About/Alliance/

[Avizienis2001] A. Avizienis, J. Laprie, and B. Randell, “Fundamental Concepts of Computer System Dependability,” IARP/IEEE-RAS Workshop on Robot Dependability: Technological Challenge of Dependable Robots in Human Environments, Seoul, Korea, May 2001.

[Basney2003] J. Basney, W. Yurcik, R. Bonilla, and A. Slagell. The Credential Wallet: A Classification of Credential Repositories Highlighting MyProxy. 31st Research Conference on Communication, Information and Internet Policy (TPRC 2003), Arlington, Virginia, September 19-21, 2003.

[Booth2003] D. Booth et. al. Web Services Architecture. W3C Working Draft 8 August 2003. http:// www.w3.org/TR/ 2003/WD-ws-arch-20030808/

[Box2003a] D. Box et. al. Web Services Policy Attachment (WS-PolicyAttachment). Version of 28 May 2003. http:// www-106.ibm.com/developerworks/library/ws-polatt/

[Box2003b] D. Box, et. al. Web Services Policy Framework (WS-Policy). Version of 28 May 2003. http://www-106.ibm.com/developerworks/library/ws-polfram/

[Butler2000] R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer, V. Welch. A National-Scale Authentication Infrastructure. IEEE Computer, 33(12):60-66, 2000.

[Chapin1999a] S.J. Chapin, D. Katramatos, J.F. Karpovich, and A.S. Grimshaw, "Resource Management in Legion," Journal of Future Generation Computing Systems, vol. 15, pp. 583-594, 1999.

[Chapin1999b] S.J. Chapin, C. Wang, W.A. Wulf, F.C. Knabe, and A.S. Grimshaw, "A New Model of Security for Metasystems," Journal of Future Generation Computing Systems, vol. 15, pp. 713-722, 1999.

[Clarke2002] B. Clarke and M. Humphrey. Beyond the "Device as Portal": Meeting the Requirements of Wireless and Mobile Devices in the Legion Grid Computing System. In 2nd International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing (associated with IPDPS 2002), Ft. Lauderdale, April 19, 2002.

[Czajkowski2004] K. Czajkowski, A. Dan, J. Rofrano, S. Tuecke, and M. Xu. Agreement-based Service Management (WS-Agreement). Global Grid Forum draft-ggf-graap-agreement-1. Version as of Feb 8, 2004.

[Della-Libera2002] G. Della-Libera et. al. Web Services Security Policy (WS-SecurityPolicy). Version of 18 December 2002. http://www-106.ibm.com/developerworks/library/ws-secpol/

[DUROC] Karl Czajkowski, Ian Foster, and Carl Kesselman. Co-allocation services for computational grids. In Proc. 8th IEEE Symp. on High Performance Distributed Computing. IEEE Computer Society Press, 1999.

[Ellison1997] B. Ellison et. al., “Survivable Network Systems: An Emerging Discipline,” Technical Report CMU/SEI-97-TR-013, Software Engineering Institute, Carnegie Mellon University, November 1997.

[Ferrari1999] A.J. Ferrari, F.C. Knabe, M. Humphrey, S.J. Chapin, and A.S. Grimshaw, "A Flexible Security System for Metacomputing Environments," 7th International Conference on High-Performance Computing and Networking Europe (HPCN'99), April 1999, Amsterdam: 370-380.

[Foster-WSRF] I. Foster, et. al. Modeling Stateful Resources with Web Services. http://www.globus.org/wsrf/ModelingState.pdf

[Foster1998] I. Foster, C. Kesselman, G. Tsudik, S. Tuecke. A Security Architecture for Computational Grids. Proc. 5th ACM Conference on Computer and Communications Security Conference, pg. 83-92, 1998.

[Foster2002a] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Draft of 6/22/02. http://www.gridforum.org/ogsi-wg/drafts/ ogsa_draft2.9_2002-06-22.pdf

[Foster2002b] I. Foster. What is the Grid? A Three Point Checklist. GRIDToday, July 20, 2002.

[GARA] I. Foster, A. Roy, V. Sander. A Quality of Service Architecture that Combines Resource Reservation and Application Adaptation. 8th International Workshop on Quality of Service, 2000.

[Globus] Globus Project. http://www.globus.org

[GRAM] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, S. Tuecke. A Resource Management Architecture for Metacomputing Systems. Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing, pg. 62-82, 1998.

[Grimshaw1999] A.S. Grimshaw, A.J. Ferrari, F.C. Knabe and M.A. Humphrey, "Wide-Area Computing: Resource Sharing on a Large Scale," IEEE Computer, 32(5): 29-37, May 1999.

[Grimshaw2000] A.S. Grimshaw, M.J. Lewis, A.J. Ferrari and J.F. Karpovich, "Architectural Support for Extensibility and Autonomy in Wide-Area Distributed Object Systems", Proceedings of the 2000 Network and Distributed Systems Security Conference (NDSS'00), San Diego, California, February 2000.

[Grimshaw2003] A.S. Grimshaw, A. Natrajan, M.A. Humphrey and M.J. Lewis, A. Nguyen-Tuong, J.F. Karpovich, M.M. Morgan, A.J. Ferrari, "From Legion to Avaki: The Persistence of Vision", Grid Computing: Making the Global Infrastructure a Reality, eds. Fran Berman, Geoffrey Fox and Tony Hey, 2003.

[Grimshaw2004] A.S. Grimshaw, M. A. Humphrey, and A. Natrajan. A philosophical and technical comparison of Legion and Globus. IBM Journal of Research & Development, vol 48 no.2 March 2004.

[Humphrey2000] M. Humphrey, F. Knabe, A. Ferrari, and A. Grimshaw, "Accountability and Control of Process Creation in Metasystems," in Proceedings of the 2000 Network and Distributed Systems Security Conference (NDSS'00), pp. 209-220, San Diego, CA, February 2000.

[Humphrey2002] M. Humphrey and M. Thompson. Security Implications of Typical Grid Computing Usage Scenarios. Cluster Computing, Vol. 5, pp 357-364, 2002.

[Humphrey2003] M. Humphrey. From Legion to Legion-G to OGSI.NET: Object-based Computing for Grids. In Proceedings of the IPDPS NSF Next Generation Software Workshop Nice, France, April 2003.

[Humphrey2004] M. Humphrey. Web Services as the Foundation for Learning Complex Software System Development. Proceedings of Technical Symposium on Computer Science Education (SIGCSE 2004), Norfolk, Virginia, March 3-7 2004.

[IBM-Microsoft2002] IBM and Microsoft. Security in a Web Services World: A Proposed Architecture and Roadmap. April 7, 2002,version 1.0. http://www-106.ibm.com/developerworks/library/ws-secmap/

[JBoss2004] JBoss Group. http://www.jboss.org/

[Knight2000] J. C. Knight et. al., "Survivability Architectures: Issues and Approaches," DARPA Information Survivability Conference and Exposition (DISCEX 2000), January 2000.

[Knight2002] J. C. Knight et. al., "The Willow Architecture: Comprehensive Survivability for Large-Scale Distributed Applications," Intrusion Tolerance Workshop, DSN-2002 The International Conference on Dependable Systems and Networks, June 2002.

[Knight2003] J. C. Knight et. al., "Towards a Rigorous Definition of Information System Survivability," DISCEX 2003, Washington DC, April 2003.

[Lewis2003] Michael J. Lewis, Adam J. Ferrari, Marty A. Humphrey, John F. Karpovich, Mark M. Morgan, Anand Natrajan, Anh Nguyen-Tuong, Glenn S. Wasson and Andrew S. Grimshaw, "Support for Extensibility and Site Autonomy in the Legion Grid System Object Model" Journal of Parallel and Distributed Computing, Volume 63, pp. 525-38, 2003.

[Liberty2003] Liberty Alliance Project. Introduction to the Liberty Alliance Identity Architecture. Revision 1.0. March, 2003. http://www.projectliberty.org

[Lorch2004] M. Lorch, J. Basney, and D. Kafura, "A Hardware-secured Credential Repository for Grid PKIs," 4th IEEE/ACM International Symposium on Cluster Computing and the Grid, April 2004.

[Microsoft2004] Microsoft ASP.NET. 2004. http://www.asp.net/

[MPICH-G2] N. Karonis, B. Toonen, and I. Foster. MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface. Journal of Parallel and Distributed Computing, 2003.

[MyProxy] MyProxy Online Credential Repository. http://grid.ncsa.uiuc.edu/myproxy/

[Nagaratnam2003] N. Nagaratnam et. al. Security Architecture for Open Grid Services. Global Grid Forum Working Draft. Revision as of 6/5/2003.

[Natrajan2001a] A. Natrajan, A. Nguyen-Tuong, M.A. Humphrey, M. Herrick, B.P. Clarke and A.S. Grimshaw, "The Legion Grid Portal", Grid Computing Environments, Concurrency and Computation: Practice and Experience, 2001.

[Natrajan2001b] A. Natrajan, A. Fox, M. Humphrey, A. Grimshaw, M. Crowley, N. Wilkins-Diere, "Protein Folding on the Grid: Experiences using CHARMM under Legion on NPACI Resources," Interna-

tional Symposium on High Performance Distributed Computing (HPDC), pp. 14-21, San Francisco, California, August 7-9, 2001.

[Natrajan2001c] A. Natrajan, M.A. Humphrey and A.S. Grimshaw, "Grids: Harnessing Geographically-Separated Resources in a Multi-Organisational Context", High Performance Computing Systems, June 2001.

[Natrajan2001d] A. Natrajan, M. Humphrey, and A. Grimshaw, "Capacity and Capability Computing using Legion," Proceedings of the 2001 International Conference on Computational Science, pp. 273-283, San Francisco, CA, May 2001.

[Natrajan2003a] Natrajan, A., Crowley, M., Wilkins-Diehr, N., Humphrey, M. A., Fox, A. D., Grimshaw, A. S., Brooks, C. L. III, "Studying Protein Folding on the Grid: Experiences using CHARMM on NPACI Resources under Legion," to appear Grid Computing Environments 2003, Concurrency and Computation: Practice and Experience.

[Natrajan2003b] A. Natrajan, M.A. Humphrey, and A.S. Grimshaw, "Grid Resource Management in Legion", in Resource Management for Grid Computing, eds. Jennifer Schopf and Jaroslaw Nabrzyski, 2003.

[NguyenTuong1996] A. Nguyen-Tuong, A. S. Grimshaw and M. Hyett, “Exploiting Data-Flow for Fault-Tolerance in a Wide-Area Parallel System”, 15th International Symposium on Reliable and Distributed Systems, pp. 1-11, October 1996.

[NguyenTuong1999] A. Nguyen-Tuong and A.S. Grimshaw, "Using Reflection for Incorporating Fault-Tolerance Techniques into Distributed Applications," Parallel Processing Letters, vol. 9, No. 2 (1999), 291-301.

[NguyenTuong2000] A. Nguyen-Tuong, “Integrating Fault-Tolerance Techniques into Grid Applications,” Department of Computer Science, Doctoral Dissertation, University of Virginia, August 2000.

[Novotny2001] J. Novotny, S. Tuecke, and V. Welch. An Online Credential Repository for the Grid: MyProxy. Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10), IEEE Press, August 2001.

[OpenSAML] OpenSAML - an Open Source Security Assertion markup Language implementation. Internet2. http:// www.opensaml.org/

[OASIS] Organization for the Advancement of Structured Information Standards, http://www.oasis-open.org/

[OASIS-SAML] Organization for the Advancement of Structured Information Standards (OASIS). Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1. OASIS Standard, 2 September 2003.

[OASIS-XACML] Organization for the Advancement of Structured Information Standards (OASIS). Extensible Access Control Markup Language (XACML) Version 1.0. OASIS Standard, 18 February 2003. http://www.oasis-open.org/committees/xacml/

[OASIS-SOAPSec] Organization for the Advancement of Structured Information Standards (OASIS). Web Services Security: SOAP Message Security. Working Draft 17, Wednesday, 27 August 2003.

[OASIS-PKI] Organization for the Advancement of Structured Information Standards (OASIS) Public Key Infrastructure (PKI) Technical Committee (TC). PKI Action Plan. Ed: Steve Hanna, Sun Microsystems. Nov 24, 2003. http://www.oasis-open.org/committees/pki/pkiactionplan.pdf

[Pearlman2002] L. Pearlman, V. Welch, I. Foster, C. Kesselman, S. Tuecke. A Community Authorization Service for Group Collaboration. Proceedings of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks, 2002.

[Rowanhill2004] J. C. Rowanhill, P. E. Varner, J. C. Knight. Efficient Hierarchic Management For Reconfiguration of Networked Information Systems. To appear in proc. of Dependable Systems and Network, June 2004.

[Siebenlist2002] F. Siebenlist et. al. OGSA Security Roadmap: Global Grid Forum Specification Roadmap towards a Secure OGSA. Global Grid Forum Working Draft. July 2002.

[SNAP] K. Czajkowski, I. Foster, C. Kesselman, V. Sander, and S. Tuecke. SNAP: A Protocol for negotiating service level agreements and coordinating resource management in distributed systems. Lecture Notes in Computer Science, 2537:153-183, 2002.

[Strassner2001] J. Strassner, Ellesson, E., Moore, B. and Westerinen, A. 2001. Policy Core Information Model -- Version 1 Specification. RFC 3060.

[Sundaram2002] B. Sundaram, and B. Chapman. XML-Based Policy Engine Framework for Usage Policy Management in Grids. Proceedings of the Third International Workshop on Grid Computing (Grid 2002). Baltimore, MD, November 2002.

[Tuecke2003] S. Tuecke et. al. Open Grid Services Infrastructure (OGSI) Version 1.0. Global Grid Forum. GFD-R-P.15. Version as of June 27, 2003.

[Tomcat2004] Apache Tomcat. Apache Jakarta Project.http://jakarta.apache.org/tomcat/index.html.2004.

[Trivedi2000] K. S. Trivedi, K. Vaidyanathan and K. Goseva-Popstojanova, "Modeling and Analysis of Software Aging and Rejuvenation, " IEEE Annual Simulation Symposium, April 2000.

[Verma2002] D. Verma, S. Sahu, S. Calo, M. Beigi, and I. Chang. A Policy Service for GRID Computing. Proceedings of the Third International Workshop on Grid Computing (Grid 2002).

[WassonG2003a] G. Wasson and M. Humphrey. Attribute-based Programming for Grid Services. Workshop on Designing and Building Grid Services. The Ninth Global Grid Forum. Chicago, IL. Oct 5-8, 2003.

[WassonG2003b] G. Wasson and M. Humphrey. Policy and Enforcement in Virtual Organizations. In 4th International Workshop on Grid Computing (Grid2003) (associated with Supercomputing 2003). Phoenix, AZ. Nov 17, 2003.

[WassonG2004] G. Wasson, N. Beekwilder, M. Morgan, and M. Humphrey. OGSI.NET: OGSI-compliance on the .NET Framework. In Proceedings of the 2004 IEEE International Symposium on Cluster Computing and the Grid. April 19-22, 2004. Chicago, Illinois.

[WassonK2003] Wasson, Kimberly S., John C. Knight, Elisabeth A. Strunk, and Sean R. Travis, Tools Supporting the Communication of Critical Application Domain Knowledge in High Consequence Systems Development, SAFECOMP 2003, The 22nd International Conference on Computer Safety, Reliability and Security, Edinburgh, Scotland (September 2003)

[White2000] B. White, A. Grimshaw, and A. Nguyen-Tuong, "Grid Based File Access: The Legion I/O Model," to Proceedings of the Symposium on High Performance Distributed Computing (HPDC-9), Aug 2000, Pittsburgh, PA.

[White2001] B. White, M. Walker, M. Humphrey, and A. Grimshaw "LegionFS: A Secure and Scalable File System Supporting Cross-Domain High-Performance Applications", Proceedings SC 01, Denver, CO. www.sc2001.org/papers/pap.pap324.pdf

[WSI] Web Services Interoperability Organization (WS-I). http://www.ws-i.org/

[WSDL] Web Service Definition Language, W3C, http://www.w3.org/TR/wsdl

[WSRF] Web Service Resource Framework, http://www-fp.globus.org/wsrf/default.asp, January 2004.

[Welch2003] V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman, S. Tuecke. Security for Grid Services. Twelfth International Symposium on High Performance Distributed Computing (HPDC-12), June 2003.

[Yavatkar2000] Yavatkar, R., Pendarakis, D. and Guerin, R. 2000. A Framework for Policy-based Admission Control. IETF RFC 2753. http://www.faqs.org/rfcs/rfc2753.html.