DRAFT – DRAFT – DRAFT

October 21, 2004

Information Technology Needs

InSAR Community Workshop Breakout Group

Nettie LaBelle-Hamer
Alaska SAR Facility
John Rundle
University of California, Davis

Introduction

This document outlines the problems identified at the InSAR Community workshop held at Oxnard, California, October 20-22, 2004. We briefly summarize the material and conclusions reached in the discussions of the breakout group on Information Technology. Note that the technology referenced here as the "international Solid Earth Research Virtual Observatory" (iSERVO) is discussed elsewhere [Donnellan et al, 2004].

Recommendation:

Rather than constructing only a data and code repository with limited functionality, we advocate the development of an InSAR data, analysis, and simulation environment based on a web service portal architecture. In such an environment, data from satellites can be downloaded, accessed, validated, simulated, interpreted, fused with data from other sources, and visualized, all without leaving the web-based portal environment. Portals such as these are the object of active research in computational and computer science, and examples of such portals and environments are now numerous.

General Topics and Requirements

The general topics and requirements developed by the IT group are shown in Table 1.

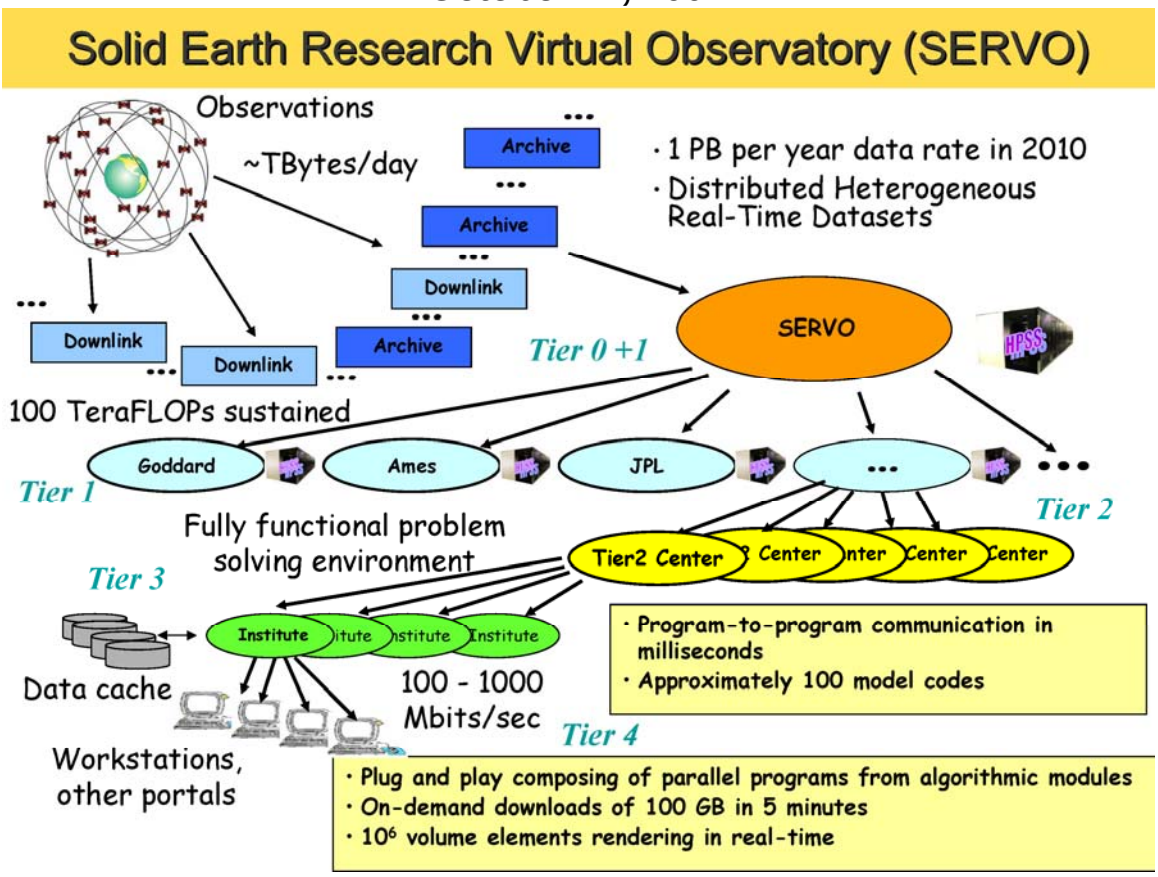
Topic	Requirements
Technology Needs Identified	1. Access to data through data bases and the web 2. Software for data mining and data exploration 3. Software for simulating, interpreting, visualizing data
Potential End Users	1. Engineering 2. Education 3. Government/Policy decision makers 4. Operational/Hazards 5. Commercial

Issues Associated with Access to Data Through Data Bases	1. Standards for domestic and international missions 2. Security 3. Quality and Validation 4. Bandwidth 5. Interoperability – XML & GML 6. Content addressable storage 7. Near real time or real time 8. Rapid response tasking of satellite
S/w and H/w	1. Data exploration and analysis - Higher level functionality is desirable - Portal architecture - Peer to peer - Collaborative - Error quantification - Graduated authentication 2. S/w development - Curation - Reliability and robustness - Fusion
Simulating, interpreting, and visualizing data	1. Large scale simulation & modeling capability is needed & should be part of mission plan 2. Centralized and Grid computing resources 3. Code sharing (open source) 4. Code validation and curation 5. Reliability and robustness
Remaining questions	1. Commercial component? 2. Operational versus Science mission?

We show below a figure illustrating the multi-tiered Solid Earth Research Virtual Observatory concept that was first proposed at the NASA Computational Technologies workshop held in Washington, DC during April 30-May 2, 2002.

DRAFT – DRAFT – DRAFT

October 21, 2004



Review: CCE Coupling Scenarios and Requirements

We now consider a Community Component Environment (CCE) whose object is to enable mapping of the distributed computing coupling technologies (services for managing distributed geophysical applications and data) to problems in data transmission and archival, data mining/pattern informatics, and multi-scale geophysical simulation.

The following is an outline of the major topics that will need to be fleshed out and further developed in the full InSAR proposal.

1. Data requirements for applications, including database/data file access as well as streaming data.
2. Service coupling scenarios: composing meta-applications out of several distributed components.
3. Limits and appropriate time-scales for this approach.
4. CCE data sources and characterizations (type of data, type of access).
5. Pattern informatics techniques
6. Multi-scale modeling techniques
7. Coupling scenarios that may be explored by the CCE project

Within CCE applications, we will adopt the "loose" or "lightly" coupling approach that is suitable for distributed applications that require millisecond (or perhaps much longer) communication latencies.

Tightly coupled communication is out of scope for the CCE. We will instead adopt (if appropriate) existing technologies for this. Prominent projects include the DOE's Common Component Architecture (CCA) and NASA's Earth Systems Modeling Framework (ESMF). These are complements to our approach. In the lightly coupled CCE, applications built from these technologies are service nodes that may be coupled with data nodes and other services.

CCE Requirements and Solutions

The following table summarizes the CCE architecture requirements and approaches that we will follow in building this system. Sections that expand on these solutions are identified.

Requirement	Description	CCE Solution or Approach
Maximize interoperability with world.	Allow for easy adoption and integration of third party solutions: service instances and service frameworks, client tools, etc.	Adopt Web Service and Portal standards using the WS-I+ approach. See "Managing Web WS-<any> Specification Glut."
Minimize lifecycle costs	Keep the cost of maintenance and training needed to keep the system running following the end of the project.	Adopt standard implementations of third party tools for Web services and portals where available and appropriate.
Security: Protect computing resources.	Computing centers have account creation and allocation policies that we cannot change. We must support their required access policies	Support as needed SSH, Kerberos, GSI security. Leverage community portal efforts through NMI, DOE Portals and similar. See "Security Requirements."
Security: Protect community data	Need a authorization model for controlling access to data sources	In short term, implement solutions using portal authorization model. Investigate authorization solutions from Web Service community; integrate with NaradaBrokering framework. See "Security Requirements."
Map multiscale models into workflow and metadata	Modeling applications must be described with metadata to identify where they fit in	CIE approach will be used to maintain metadata. Workflow will be mapped to scripting techniques (HPSearch). See "Core CCE Infrastructure:

support.		Context and Information Environment" and "Controller Environments for CCE: Portals and Scripting."
Storage requirements	CCE tools will need three types: volatile scratch, active, and archival storage.	Hardware resources necessary to run CCE applications will be obtained from NASA JPL, Goddard, and Ames. CCE architectures will be compatible with these. We estimate mass storage requirements (terabytes).
Data source requirements.	Must support current community data sources for GPS, Fault, and Seismic data	Adopt standards (such as OGC standards for geospatial data) where they are available. See [Parker2004].
Computational requirements	The system must support computational	We will leverage NASA computational resources. The CCE system will be compatible with these sites.
Visualization requirements	The CCE must support earth surface modeling of both input data sources and computational results. Analysis techniques will use IDL and Matlab tools wrapped as services.	We will adapt OGC tools such as the Web Map Server to provide interactive maps with data sources and computational results as overlays. See "Visualization Requirements" for more information. Services to support wrapped IDL and Matlab will be developed.
Data modeling and query requirement	Must support standard models wherever they exist; must support schema resolution and meta-queries to resolve differing data models.	We will develop and integrate ontology management tools. See "CCE Data Models and Tools."
Network Requirements	The CCE must take into account available network speeds required to connect	We will design the CCE to scale to a potentially Global deployment in cooperation with ACES partners. As described in [Parker2004] we will adjust the network dependence of our services to be compatible with standard internet latencies. Higher performance for some interactive visualizations and data transfers may be required. Our approach to this is detailed

		in "Core CCE Infrastructure: Internet-on-Internet (IOI)."
Scalability	The system as a whole should scale to include international partners.	Fault tolerance, redundancy, and service discovery/management are critical if the system is to work on the international scale. We describe our approaches to these problems in the IOI and CIE sections of this report.

Table 1: CCE system requirements and solution approaches.

In the following section we review applications and scenarios that we are pursuing.

Key Capabilities

Managing WS-<any> Specification Glut

With the key Grid Service architectural debate apparently resolved, we must now face the consequences: proliferation of would-be standard extensions to the basic approach. Web Services (and Grid Services) may be spawned out of a number of specification bodies, including primarily the World Wide Web Consortium and OASIS. At the time of writing, there are over sixty different specifications, many of them with overlapping and competing concerns, many others strategically linked in "roadmaps" that attempt to ensure adoption through specification dependency.

In order to resolve this situation, we advocate a conservative approach that follows the WS-I interoperability approach. WS-I's charter is to promote interoperability in various implementations of Web Service specifications from different vendors running on different platforms and supporting different languages. Many of the WS specifications are not written with any simultaneous implementation/interoperability tests, and without requirements for completely independent implementations, so there are inevitably uncertainties, ambiguities, and plain old bad decisions in the specifications. WS-I exists to clarify/refine these specifications after the fact.

Core CCE Infrastructure: Internet-on-Internet (IOI)

Web Services are beginning to mimic the capabilities of the lower level network within their messages and messaging implementations. Reliability and fault tolerance are two prominent examples.

We think this development is important for reasons previewed above. We refer to this as the "Service-Internet-on-Bit-Internet," or IOI. IOI is essentially a reimplementation of standard low-level networking capabilities at the higher application level.

Typical IOI capabilities include several items listed previously:

1. Support for multiple transport protocols
2. Support for many different message delivery protocols, such as reliable delivery, once-only delivery, ordered delivery, and persistent delivery/delivery replay.
3. Application-level performance optimization through compression/decompression.
4. Fragmentation/coalescence of messages, which may be delivered over separate routes, in parallel. One may use this to do higher performance file transfers and to increase the reliability of large file transfers.
5. Security services, such as message encryption and authorization.
6. Time stamping services to assist with ordered delivery and replay.
7. Congestion control and dynamic best-route determination.
8. Performance monitoring.
9. Ad-hoc network support
10. Broker discovery for internal NB network management.
11. Topic discovery
12. Native NB support for SOAP

All of these are traditional "low-level" networking capabilities that can be re-implemented in the NB fabric layer, on top of traditional networking.

High Performance SOAP: Interestingly, the implication of Web Service standards is that we can mimic the TCP/IP layer directly in the SOAP (Simple Object Access Protocol, a protocol enabling objects to be defined and accessed) header and use the much higher performance UDP (User Datagram Protocol, which is a messaging protocol) for exchanging SOAP messages. We are investigating this development. We see it as extremely useful for interactive applications that demand higher performance out of services, although it is not a requirement in any of our application scenarios.

We may also use broker topologies to mimic network topologies, creating overlay networks, "virtual private grids", firewalls, and demilitarized zones.

Core CCE Infrastructure: Context and Information Environment (CIE)

In addition to the IOI capabilities, we have a number of other requirements needed to manage grids and services. That is, if implemented correctly, the IOI fabric may be invisible to the applications that run in it. Although an application developer may conceivably want to directly touch this layer, this would not be standard. Instead, they would specify the desired Quality of Service and let the IOI fabric implement this.

There are a number of higher level services and capabilities that do not belong in the IOI layer. As a general rule, these are services that extend (rather than mimic) the lower level networking features and are more specifically needed for Web service management. Typical examples include service information and metadata management. We refer to this collection of capabilities as the Context and Information Environment.

The basic problem is the following: which service or sequence of services actually accomplishes my desired calculation? In a "Grid of Grids" approach, there are various service collections that provide the basic capabilities of the CCE. There are execution

services for running remote applications and orchestrating cooperating services, there are data grid services that provide access to remote data, there are collaboration services, etc. These services are used to build "useful" grids for CCE applications and are maintained in an IOI fabric that is responsible for the messaging infrastructure and the related qualities of service.

In the Grid of Grids approach, the services must share "higher level" information about themselves. This is commonly called metadata. We may extend this to the problem of building "information grids" and "knowledge grids" that build on the more traditional "execution" and "data" grids.

The problem in the Grid and Web Service world is that the metadata/information problem for describing services is very confused.

- The WS-I (Web Service Interoperability) has effectively endorsed UDDI (Universal Description Discovery Integration standard), but it has a number of problems (rigid data models that don't describe science Grids very well, no mechanisms for dynamic service discovery and clean up, to name two.)
- The Semantic Web has worked for a number of years on metadata descriptions and more sophisticated knowledge management, but tends to get ignored by the Web Service community, at least in the US. DARPA has tried to refocus its part of the Semantic Web work into service descriptions.
- The Grid community has two competing concepts (the WSRF specification suite and WS-GAF) on managing metadata, particularly when it concerns dynamically evolving resource state information.
- From OASIS we have the WS-DistributedManagement suite of specifications and WS-MetadataExchange.

Controller Environments for CCEs: Portals and Scripting

The CCE Portal will allow universal access and minimize requirements on the user's desktop. However, portals will not be the only way of interacting with the system. The CCE system needs a more suitable framework for development by application scientists. Experience indicates that developers need a scripting environment for interacting with services: they need to couple their applications to other services, but the actual workflow chain may need to be altered frequently. Once scripts have been finalized, they need to be incorporated into the portal.

Portals

The portal field has undergone important changes since the portal was implemented. Two important specifications, JSR168 and WSRP, have emerged that may standardize much of the portal component work so that it may be shared across vendor containers. In summary,

- JSR168 defines a standard *local* portlet API in Java. JSR 168 compatible portlet engines can load each other's portlets. Examples of JSR168-compatible portlet containers include WebSphere, Jetspeed2, uPortal, and GridSphere.

- WSRP defines a standard *remote* portlet API in WSDL. That is, the portal runs separately from its container.

These two standards are compatible: JSR168 compatible portlets may act as proxies/web service clients to the remote WSRP portlets. Both standards have shortcomings (some possibly serious) and both have reference implementations that will need improvement, but we expect portlet containers to support them nonetheless.

The CCE should migrate to JSR168 and/or WSRP portlets. We will follow the lead of other projects, particularly the NMI OGCE portals project and the DOE Portal Web Services project.

Scripting Support for Web Services

The Service Oriented Architecture makes a clean separation between services, messages, and user interfaces. It is possible to build more than one client interface to the same remote services. This has a very clear value in the CCE project, since typical CCE applications require many different services to be orchestrated in a single application.

Security Requirements

The CCE system has two main categories:

- Securing data, application, results, and other intellectual resources
- Securing hardware resources

Applicable security concepts are

- Authentication or proof of identity.
- Authorization or access rights associated with an authenticated identity.
- Privacy and encryption
- Data integrity

References

[Parker2004] Jay Parker and Marlon Pierce, "Exploring Coupling Methodologies for the Development of Cross-Scale Tools." Tech Report for AIST CCE Project, 2004.

[Donnellan2004a] Andrea Donnellan, Jay Parker, Geoffrey Fox, Marlon Pierce, John Rundle, Dennis McLeod Complexity Computational Environment: Data Assimilation SERVOGrid 2004 Earth Science Technology Conference June 22 - 24 Palo Alto.