Chris A. Mattmann, Ph.D.
Jet Propulsion Laboratory, California Institute of Technology /
University of Southern California / Apache Software Foundation

Big Data Architecture: Fundamentals

Sponsored by the Jet Propulsion Laboratory and by the Keck Institute for Space Studies, California Institute of Technology

Outline

- Introduction to Software Architecture
- Styles, Patterns, Reference Architectures
- Architectural Modeling, Visualization
- Architectural Drift and Recovery
- Case Study: Grid Computing
- Conclusions

Now for an application of architecture to Big Data

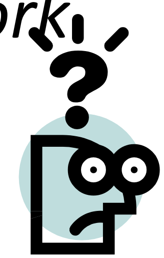
- Next section of the lecture covers a specific example that will bring together Software Architecture and Big Data and illustrate the principles

Introduction: Grid Computing

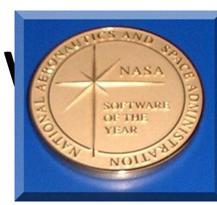
- The goal is to provide an infrastructure **AND** architecture for providing seamless utilization and access of heterogeneous resources
- The big picture
  - Two predominant types of resources
    - Data (large data sets, heterogeneous storage methods, difficult quality of service goals)
    - Computational (heterogeneous computing systems, environments, security, types of jobs)
  - As such, two predominant types of grids
    - Data Grids and Computational Grids

Motivation

- Workshop paper
  - (Paraphrased): "The grid technology you are studying is nothing more than a **simple** *Object Oriented Framework*"



- Ask ourselves the question
  - Was the reviewer right?
- Alleged *Object Oriented Framework* was 2003 Runner-up NASA Softw… of the Year



Motivation (cont)

- State of the practice in grid computing
  - Little understood or known about the *as-implemented* architectures of current grid technologies [FINK04]
  - Little understood or known about understanding the *requirements* of grid computing
    - Requirements normally written as "high level" objectives, rather than your everyday software engineering style requirement
- Risk of architectural drift
  - myriad of grid technologies that all generally claim to have the same capabilities
    - but are clearly implemented using different approaches, technologies, with different requirements in mind

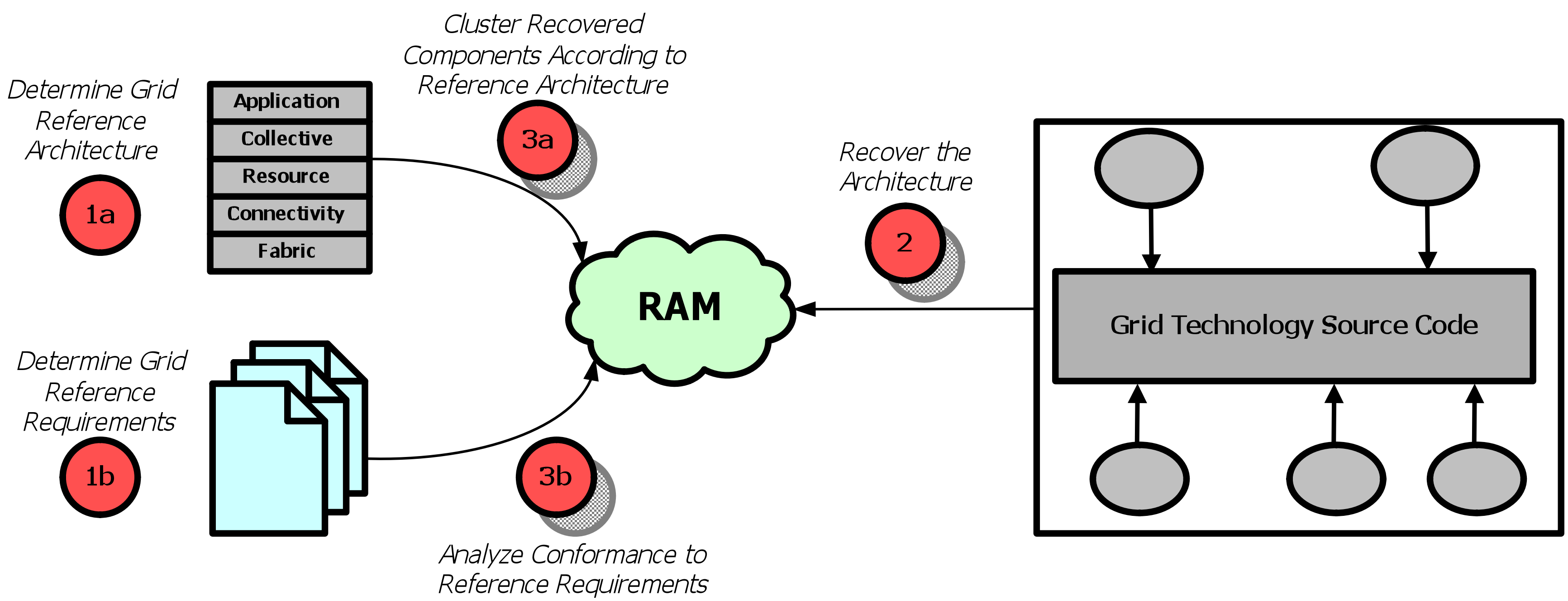
Our Approach

- Object Oriented Grids
  - Thanks to the reviewer, we wanted to study OO style grids
  - Also a by product of the chosen architectural recovery approach
- Open Source
  - Needed source code as input, because many of the grid technologies had somewhat limited documentation
- Off-the-shelf, useful systems
  - **Globus**, the *de facto* grid technology
  - **OODT**, NASA's and National Cancer Institute's grid technology
  - **DSpace**, rapidly becoming pervasive in the digital library community as a "data grid" technology
  - **GLIDE**, a lightweight, data-intensive grid technology
  - **JCGrid**, open source computational grid technology from Sourceforge.net

Our Approach

- Inputs:
  - Detailed literature review among 4 seminal grid papers
    - Chervenak (data grid), Foster (3 point checklist), Kesselman (anatomy), Foster (physiology)
    - Allowed us to distill a set of "reference" requirements for grid technologies
      - Along with a "reference" architecture for grid technologies, the famous layered architecture proposed by Kesselman et al.
  - Source code/documentation from the 5 selected "representative" grid technologies
- Recover the architecture of the 5 technologies using the *Focus* [JASE-MED] architecture recovery approach
  - Gives you "Recovered Architectural Model", or *RAM*

Our Approach



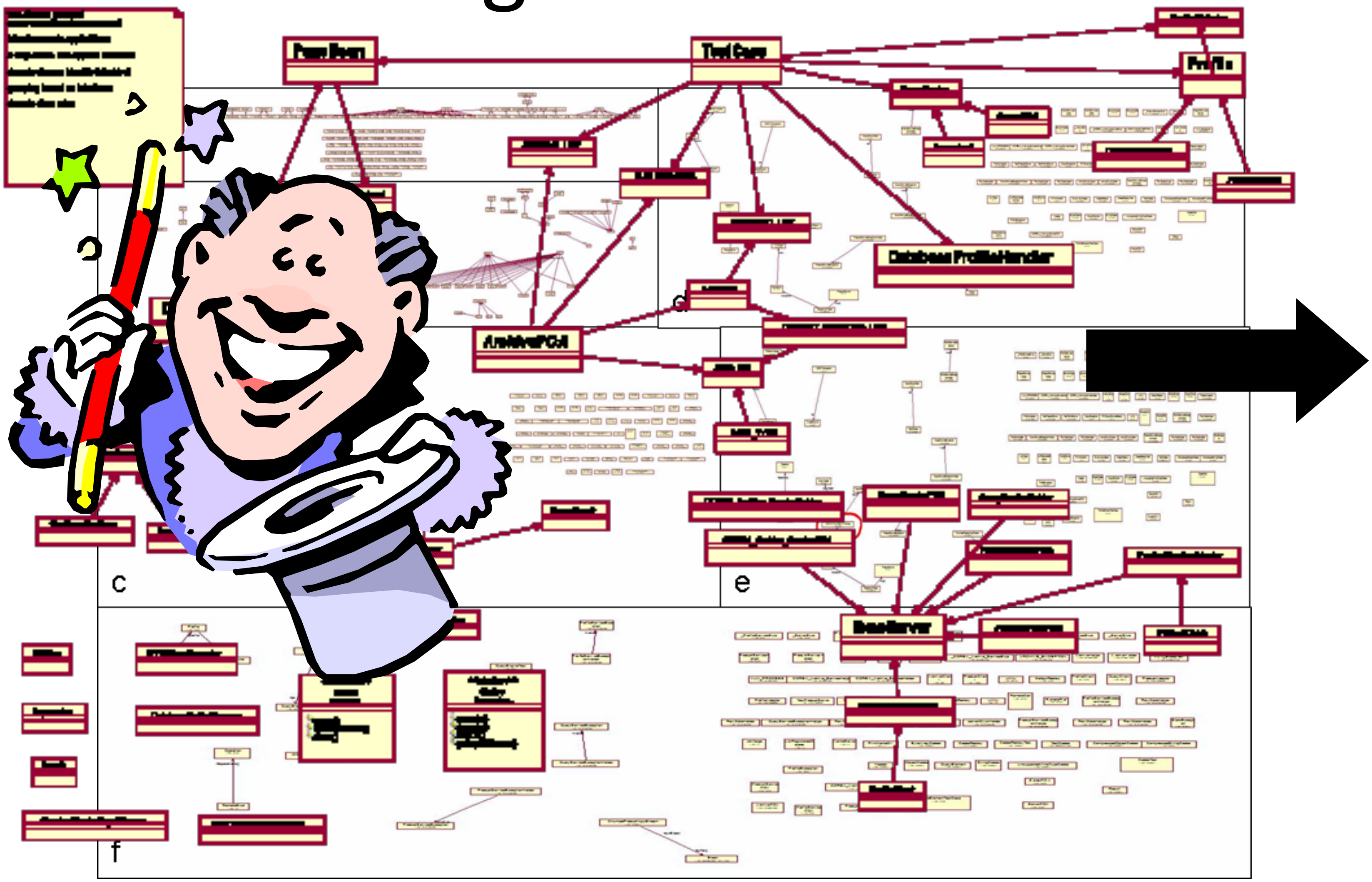
Our Approach

- "Cluster components according to reference architecture"
  - i.e., try and "shoe horn" the components into their appropriate layer of the 5 layer grid ref. arch
  - Used:
    - Documentation of grid technology (e.g., web site, txt files included with distribution, white papers and research papers, javadocs)
    - Reference requirement table, maps layer(s) to requirements
    - Observation of the component's role in the running system (instrumentation)
- "Analyze conformance to reference requirements"
  - Use the following table to map requirements/desired capabilities to architectural layers

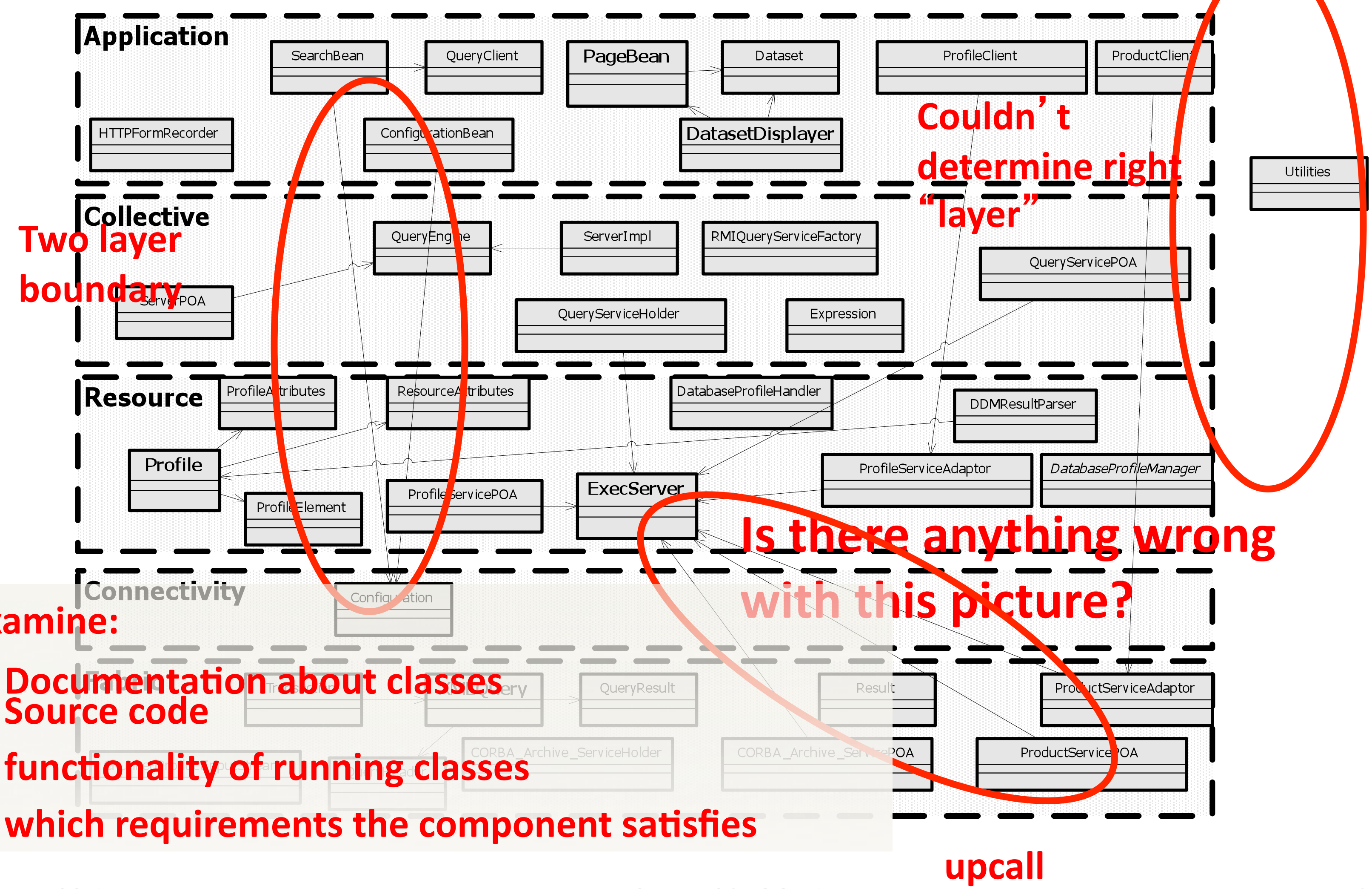
Reference Requirements

Requirement		Impacted Layer
1	Share resources across dynamic and geographically dispered organizations	Collective
2	Enable single sign-on	Connectivity
3	Delegate and authorize	Connectivity
4	Ensure access control	Connectivity
5	Ensure application of local and global policies	Fabric
6	Control shared resources	Collective
7	Coordinate shared resources	Collective
8	Ensure "exactly once" level of reliability service	Connectivity, Application Resource
9	Use standard, "open" protocols and interfaces	Collective, Resource, Connectivity
10	Provide ability to achieve non-trivial QoS	Application, Resource, Collective
11	Ensure neutrality of data sharing mechanism	All layer's implementation
12	Ensure neutrality of data sharing policy	Collective, Resource, Connectivity, Fabric
13	Ensure compatibility with Grid infrastructure	Possibly all layers
14	Provide uniform information infrastructure	Application, Resource, Collective
15	Support metadata management	Resource
16	Interface with heterogeneous storage systems	Fabric
17	Provide the management of data replicas	Fabric, Resource, Collective

Recovering the RAM of OODT



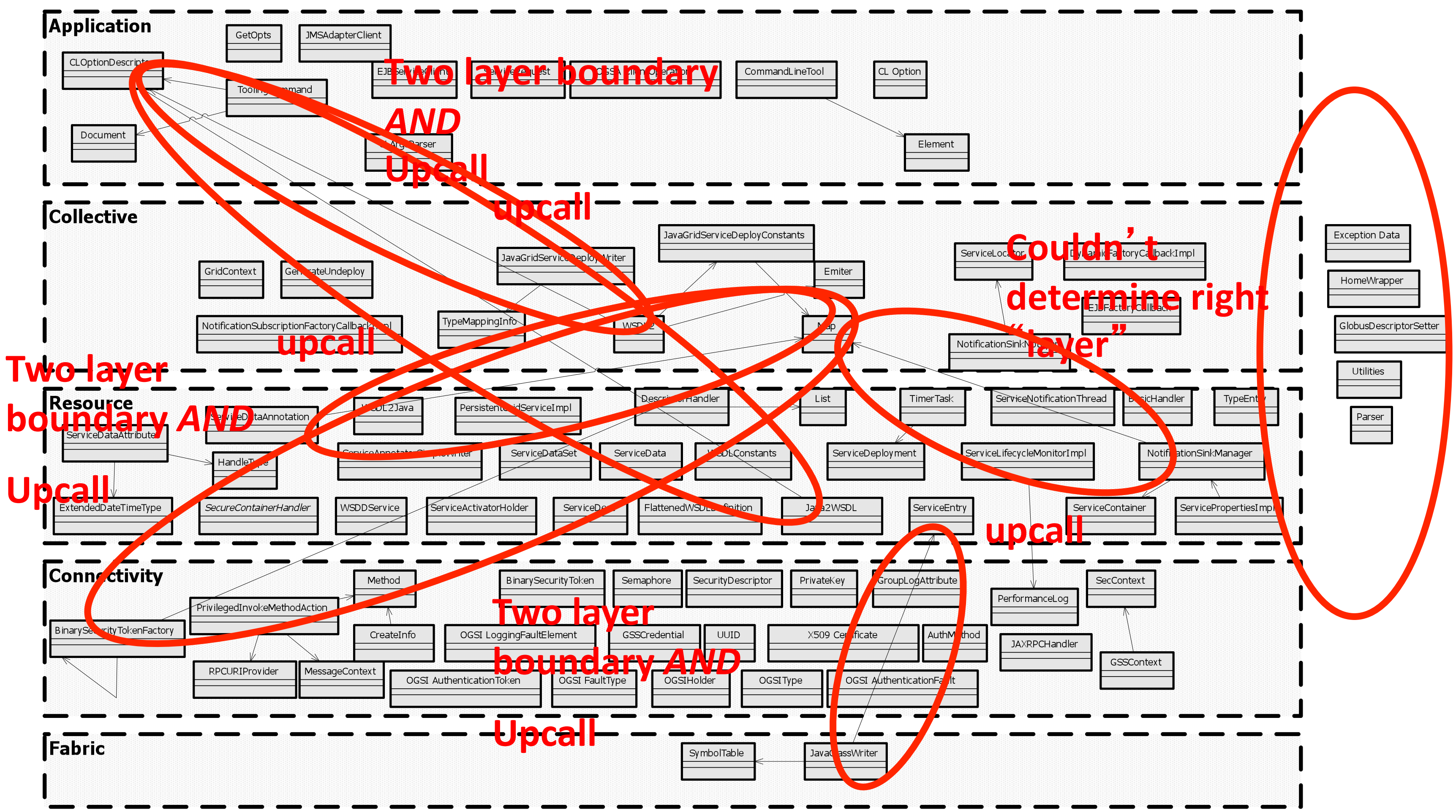
“Shoe horn” components into Grid Reference Architecture



Examine:

- Documentation about classes
- Source code
- functionality of running classes
- which requirements the component satisfies

What about this one?
(Globus's RAM)



Numerous violations of the reference architecture

- Component upcalls
  - Violation of the layered architectural style
  - Possible architectural "drift"
- Crossing two (or more) layer boundaries
  - Worst of these was a cross of *all four layers* in the reference architecture!
  - Possibly a "refactorization" problem
    - Too many needed capabilities in lower level componet, not enough abstraction
- Can't determine what *layer* the component goes in
  - Not enough documentation, or component with capabilities that seemed to "span" several layers

Discussion

- Grid technologies appear to be DSSA's for the domain of grid computing
  - Each has its own instantiated subset of the core grid components
- Size and Complexity of the Grid technologies
  - Large variation
  - 2000 SLOC in GLIDE to over 55,000 SLOC in Globus
  - 61 classes in GLIDE to over **14 times** as many in Globus (around 900)
  - Each technology has its own design foci
    - Pervasive grids versus computational grids versus data grids and so on…

Identification of "optional" requirements

- Identified through careful examination of study results
  - Grid Technology A supports Requirement X and claims to be Y-type "grid"
- Optional requirements
  - Enable single sign on
  - Delegate and authorize
  - Ensure access control
  - Ensure "exactly once" level of reliability service
  - Data Grid requirements

Distinction between "Computational" Grids and "Data" Grids

- In terms of identified requirements
  - Distilled from the results of our study
- Data Grid requirements
  - Ensure neutrality of data sharing mechanism
  - Ensure neutrality of data sharing policy
  - Provide uniform information infrastructure
  - Support metadata management
  - Interface with heterogeneous storage systems
  - Manage data replicas

Related Work

- Study by Finkelstein et al. [FINK04]
  - Journal of Grid Computing
  - Studies architectural style requirements and patterns for data grid systems
  - Level of "conformance" to architectural style
  - Identification of 83 data grid requirements
- Numerous works in computational grid computing, data grid computing by Foster, Chervenak, Kesselman, et al.
- Architecture recovery of middleware by Ivkovic et al.

Conclusions

- Grid systems regularly violate the reference architecture
- Overlap between grid layers
  - Components appear to belong in “multiple” layers
- Grid requirements are under-specified and very high level
  - It’ s difficult for most middleware to “not” provide at least some of the grid requirements
- Grid interoperability poses a problem
- An ADL for grid systems would be very useful

Current Studies and Work

- Possibly "forward" architecting grid system
  - Use technique such as CBSP to distill, from recovered requirements, the appropriate grid components
  - Compare and contrast results with that of *as-implemented* architectures
- Discuss more with grid technology experts as to "reasons" behind the level of non-conformance
  - Perhaps grid technologies are indeed such project-specific DSSA's, that a "reference" architecture for grid computing may be *impossible* to define
- Address informality of grid requirements
  - Investigate formal requirements specification
- Address informality of grid architecture, and violations
  - Conformance to an ADL?

Some Pointers

- CSCI 578: Software Architectures at USC
  - http://sunset.usc.edu/classes/cs578_2014b/
- Mattmann Home Page
  - http://sunset.usc.edu/~mattmann/
- Study of Grid Architectures
  - Original Study: http://sunset.usc.edu/~mattmann/GridMiddlewares/
  - Longer version (submitted to J. Grid Computing) http://sunset.usc.edu/~mattmann/grids/