Haifa Research Lab.

OGSA-based Problem Determination
An Use Case

Benny Rochwerger
Research Staff Member

Nov. 24, 2003

Agenda

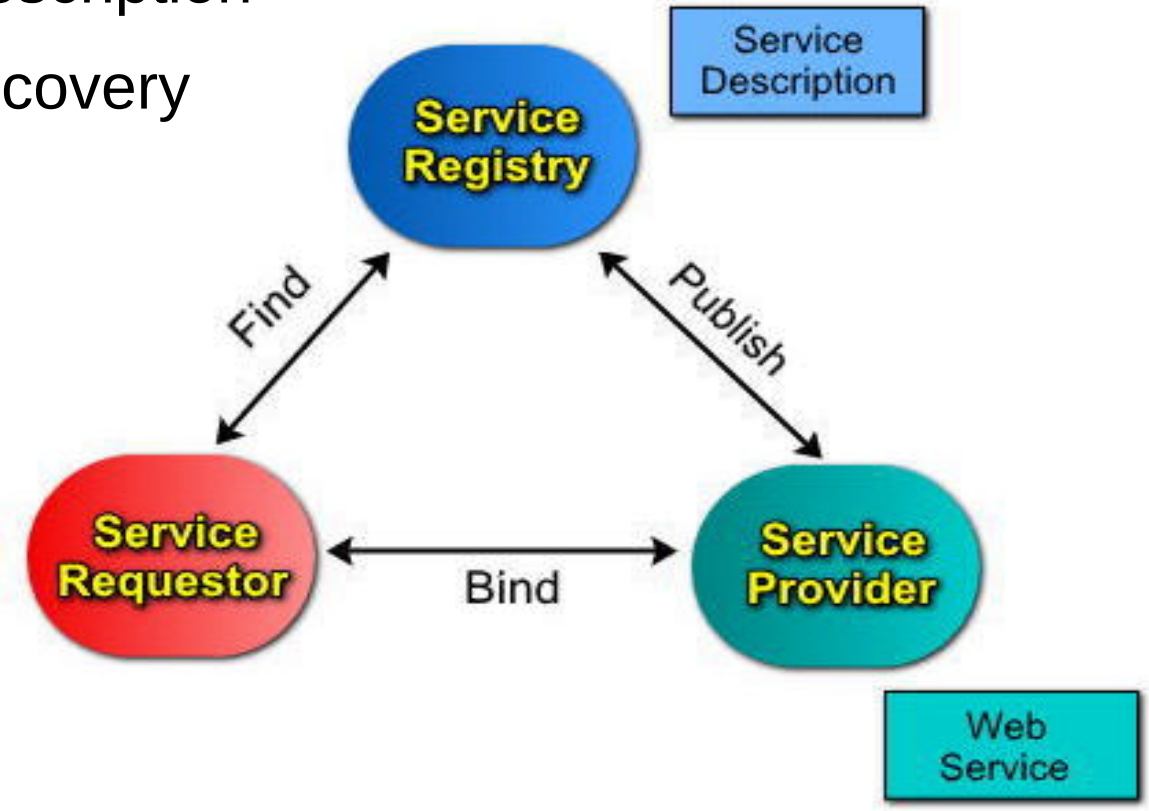
- The Open Grid Services Architecture
- Autonomic Computing
- The End to End Problem Determination Toolkit
- Lessons Learned

Web Services in a Nutshell

- A Web Service is an interface that describes a collection of operations that are network accessible through standardized XML messaging
- A Web Service is described using a standard, formal XML notion, called service description, which covers message formats, transport protocols and location
- The interface hides the implementation details of the service

The Web Services Model

- SOAP: Communication protocols
- WSDL: Service description
- UDDI: Service discovery



What is missing in the Web Services Model ?

- Service description captures interface syntax
  - The service accepts an operation request with a particular signature
- Service description does not capture semantics
  - Operation should respond as expected

    "As expected" is usually defined offline in specifications
  - Use names as basis for reasoning about semantics
  - Standardize common services and operations

What is missing in the Web Services Model ?

getQuote("BMW")

Used Motorcycle dealer

1. BMW K 1200 RS Mileage: 8266
   Year: 1999 Price: £6950
2. BMW R 1100 S Mileage:10347
   Year: 1998 Price: £5750
3. BMW R 1100 S Mileage : 7284
   Year: 1999 Price: £6250
4. BMW R 1100 S Mileage: 9000
   Year: 2000 Price: £6350

Stock Quotes Provider

BMW (ISE:BMW) Time: 7:11
Last Price: 1961.00
Change: +18.00
Open: 1942.00
High: 1977.00 @ 5:41 ET
Low: 1927.00 @ 3:16 ET
Currency: GBX

The Open Grid Services Architecture

- Extensions to existing Web Service technologies
  - Transient stateful service instances
    - Significant implications for how services are managed, named, discovered and used.
  - Standard interfaces and behaviors that address key distributed system issues
    - Naming and bindings
    - Information model
    - Lifecycle
    - Notification

OGSA: Transient Service Instances

- "Web services" address discovery & invocation of persistent services
  - Interface to persistent state of entire enterprise
- Must also support transient service instances, created/destroyed dynamically
  - Interfaces to the states of distributed activities
  - E.g. workflow, video conf., dist. data analysis

OGSA: Standard Interfaces & Behaviors

- Naming and bindings
  - Every service instance has a unique name, from which can discover supported bindings
- Information model
  - Service data associated with Grid service instances, operations for accessing this info
- Lifecycle
  - Service instances created by factories
  - Destroyed explicitly or via soft state
- Notification
  - Interfaces for registering interest and delivering notifications

Service Data

- A Grid service instance maintains a set of service data elements
  - XML fragments encapsulated in standard <name, type, TTL-info> containers
  - Includes basic introspection information, interface-specific data, and application data
- findServiceData operation (GridService interface) queries this information
  - Extensible query language support
- See also notification interfaces
  - Allows notification of service existence and changes in service data

Lifetime Management

- GS instances created by factory or manually; destroyed explicitly or via soft state
  - Negotiation of initial lifetime with a factory (=service supporting Factory interface)
- GridService interface supports
  - destroy operation for explicit destruction
  - setTerminationTime operation for keepalive
- Soft state lifetime management avoids
  - Explicit client teardown of complex state
  - Resource “leaks”in hosting environments

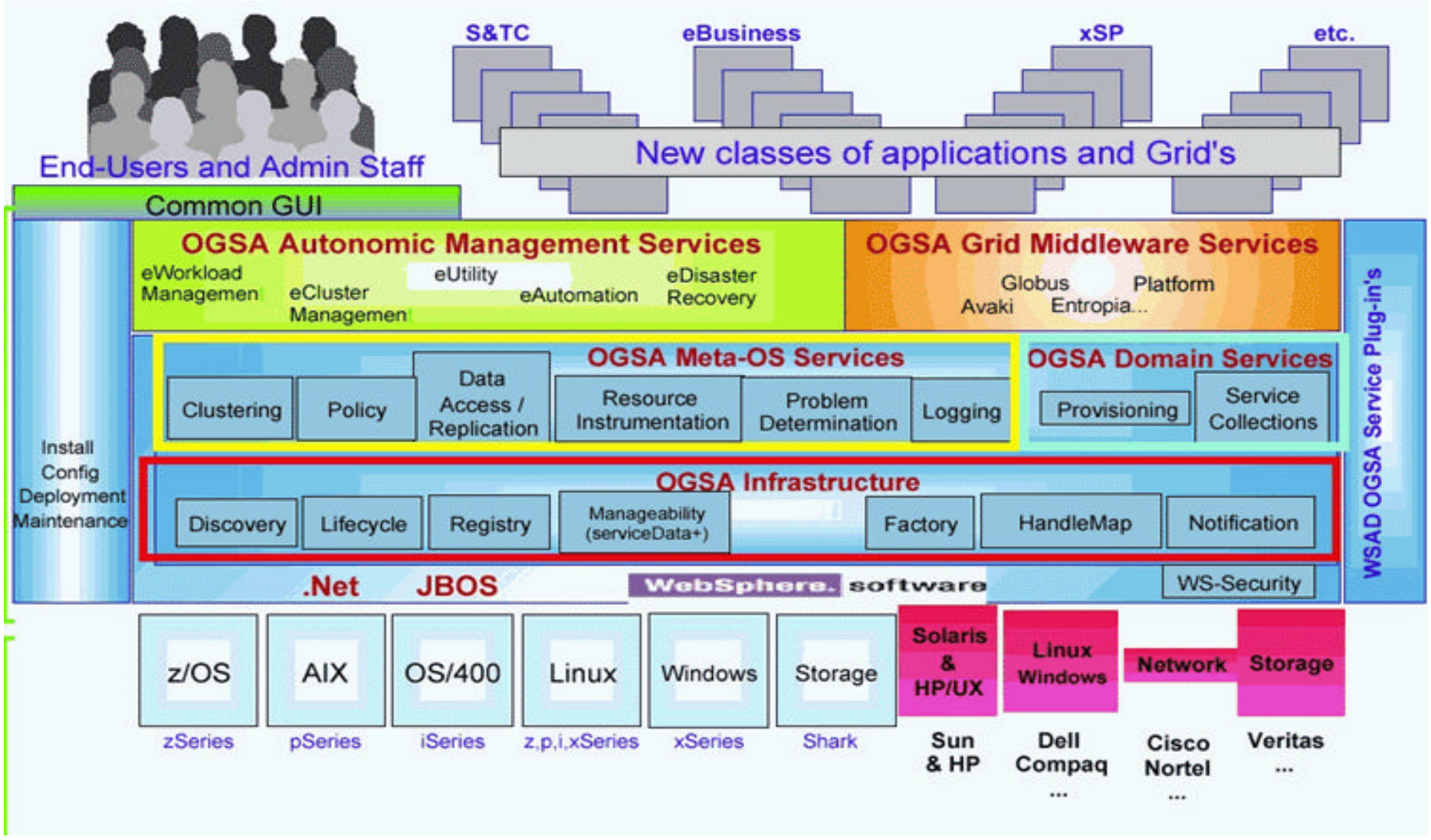
Factory

- Factory interface's createService operation creates a new Grid service instance
  - Reliable creation (once-and-only-once)
- createService operation can be extended to accept service-specific creation parameters
- Returns a Grid Service Handle(GSH)
  - A globally unique URL
  - Uniquely identifies the instance for all time

Notification Interfaces

- NotificationSource for client subscription
  - One or more notification generators
    - Generates notification message of a specific type
    - Typed interest statements: E.g., Filters, topics, …
    - Supports messaging services, 3rd party filter services, …
  - Soft state subscription to a generator
- NotificationSink for asynchronous delivery of notification messages
- A wide variety of uses are possible
  - E.g. Dynamic discovery/registry services, monitoring, application error notification, …

The OGSA Framework



Agenda

- The Open Grid Services Architecture
- Autonomic Computing
- The End to End Problem Determination Toolkit
- Lessons Learned

Autonomic Computing Attributes

Self-managing Systems that deliver...

Increased Responsiveness

Adapt to dynamically changing environments

Business Resiliency

Discover, diagnose, and act to prevent disruptions

Operational Efficiency

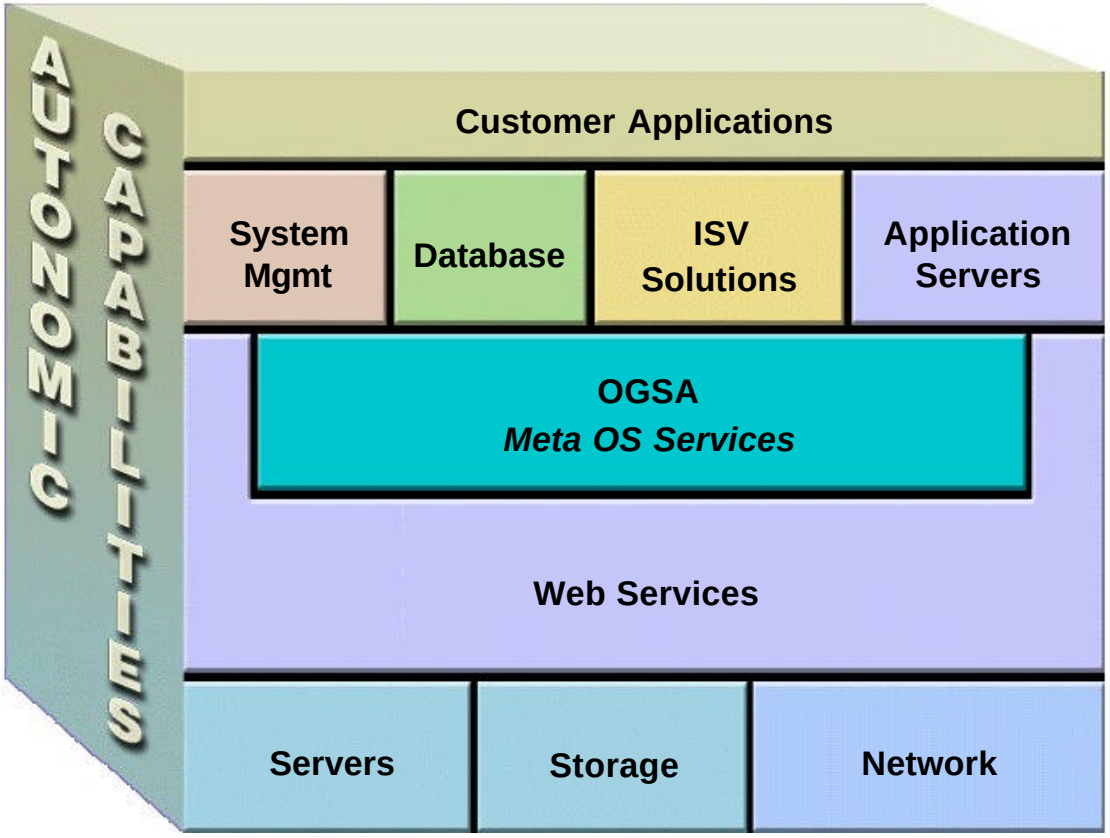
Tune resources and balance workloads to maximize use of IT resources

Secure Information and Resources

Anticipate, detect, identify, and protect against attacks

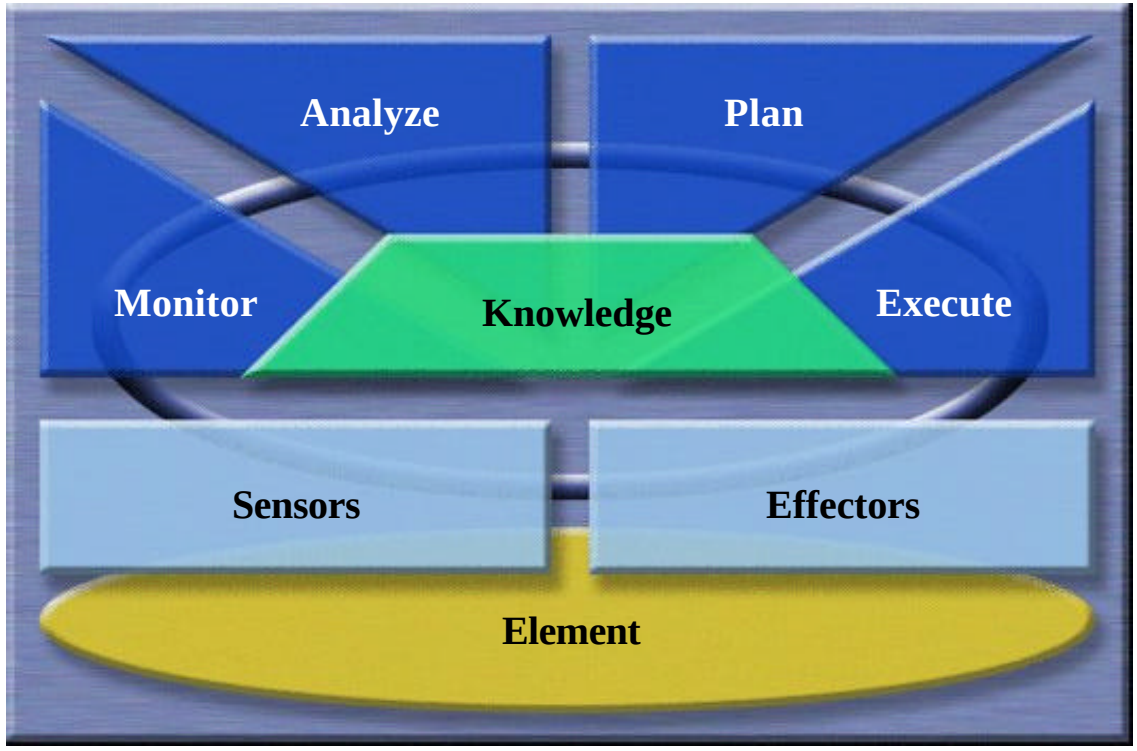
A Holistic Approach to Autonomic Computing

- Service oriented architecture
- Virtualization of resources
- Automation & manageability enablement for each system element



Architecture Framework

Autonomic Computing Building Blocks



An autonomic element contains a continuous control loop that monitors activities and takes actions to adjust the system to meet business objectives

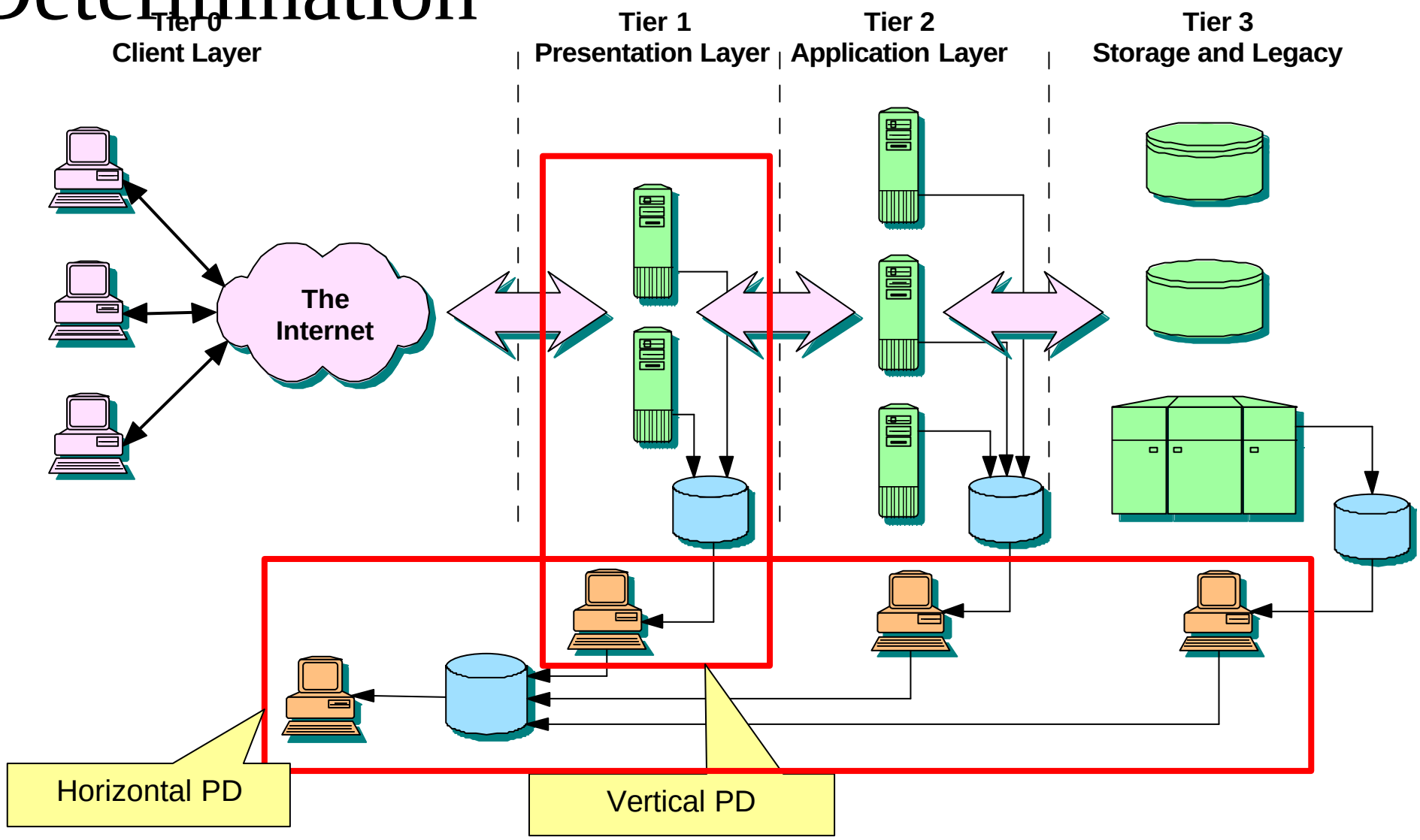
Autonomic elements learn from past experience to build action plans

Managed elements need to be instrumented consistently

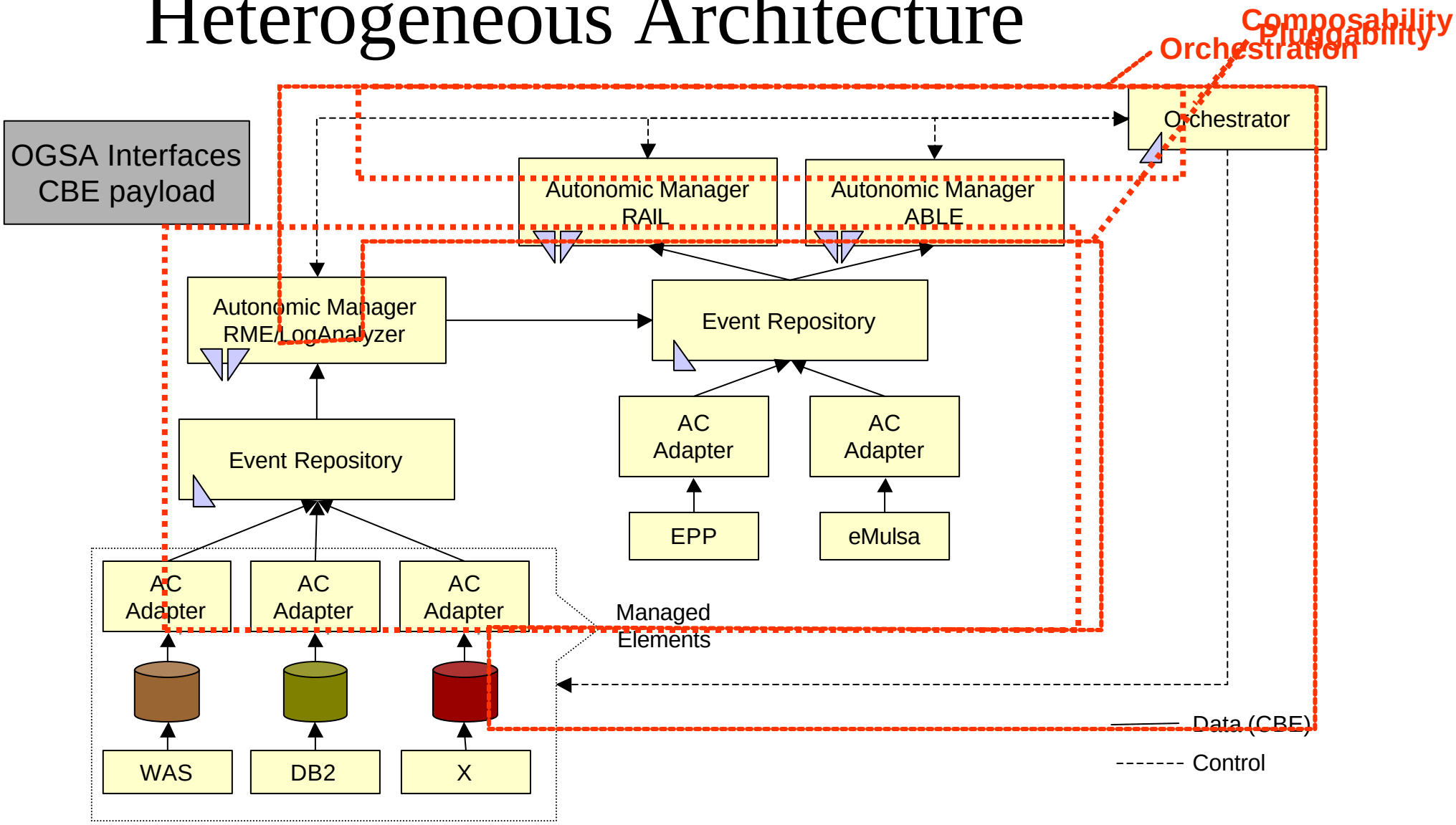
Agenda

- The Open Grid Services Architecture
- Autonomic Computing
- The End to End Problem Determination Toolkit
- Lessons Learned

Hierarchical End-to-end Problem Determination



The ePD Toolkit: Hierarchical, Heterogeneous Architecture



The ePD Toolkit Guiding Concepts

- Compose-ability
  - The ability to bind PD components into hierarchal end-to-end PD system
- Plug-ability
  - The ability to switch components with similar functionality in a complex PD system, e.g.,

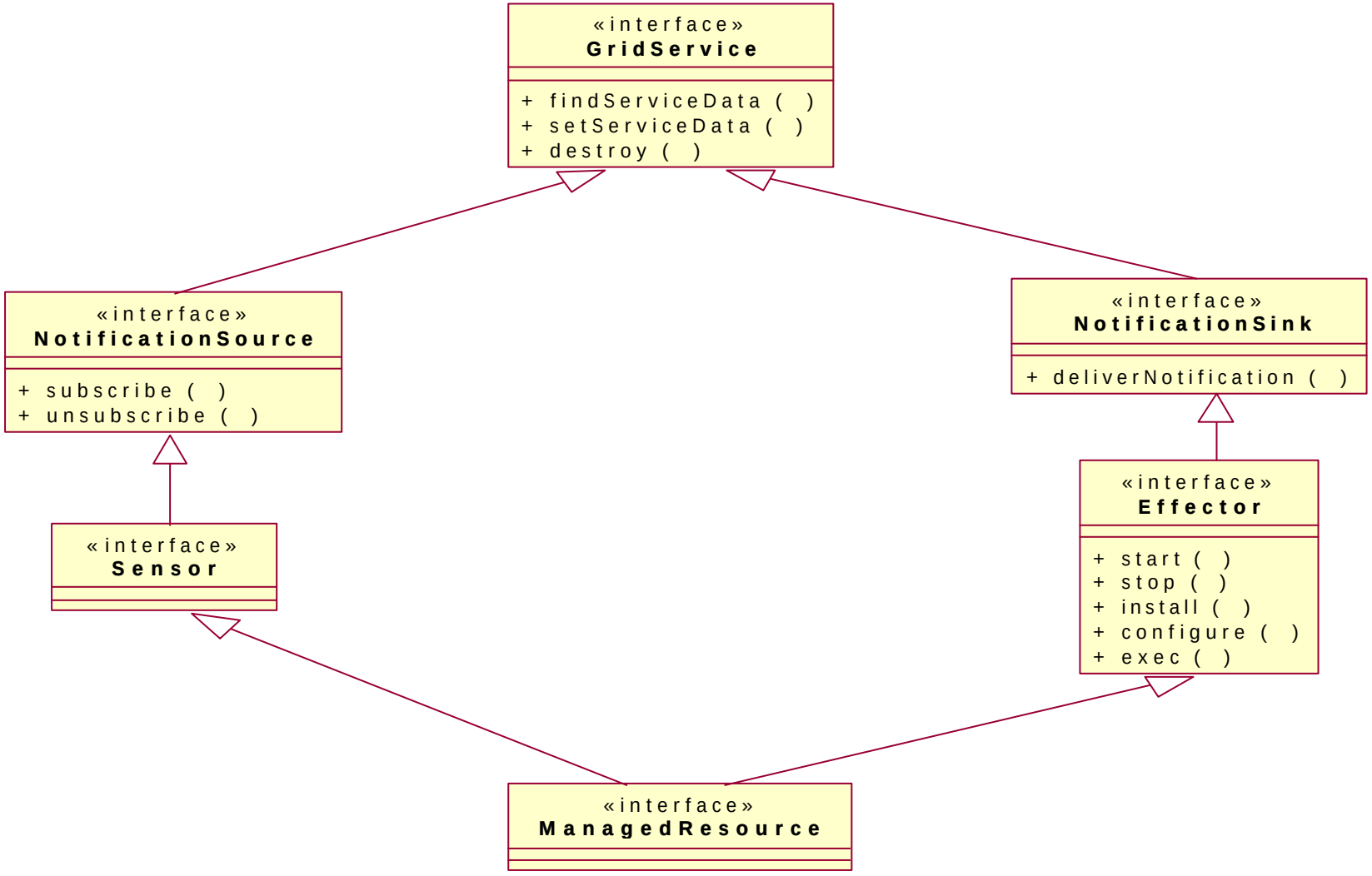
    Switch between LogAnalyzer and ABLE based analyser
- Orchestration
  - The mechanism for automatic dynamic configuration and wiring of the hierarchical PD system
  - The capability to instantiate and control, on demand, components/tools as a result of autonomous decisions of the PD system, e.g.,

    Collects all messages coming from server that looks faulty

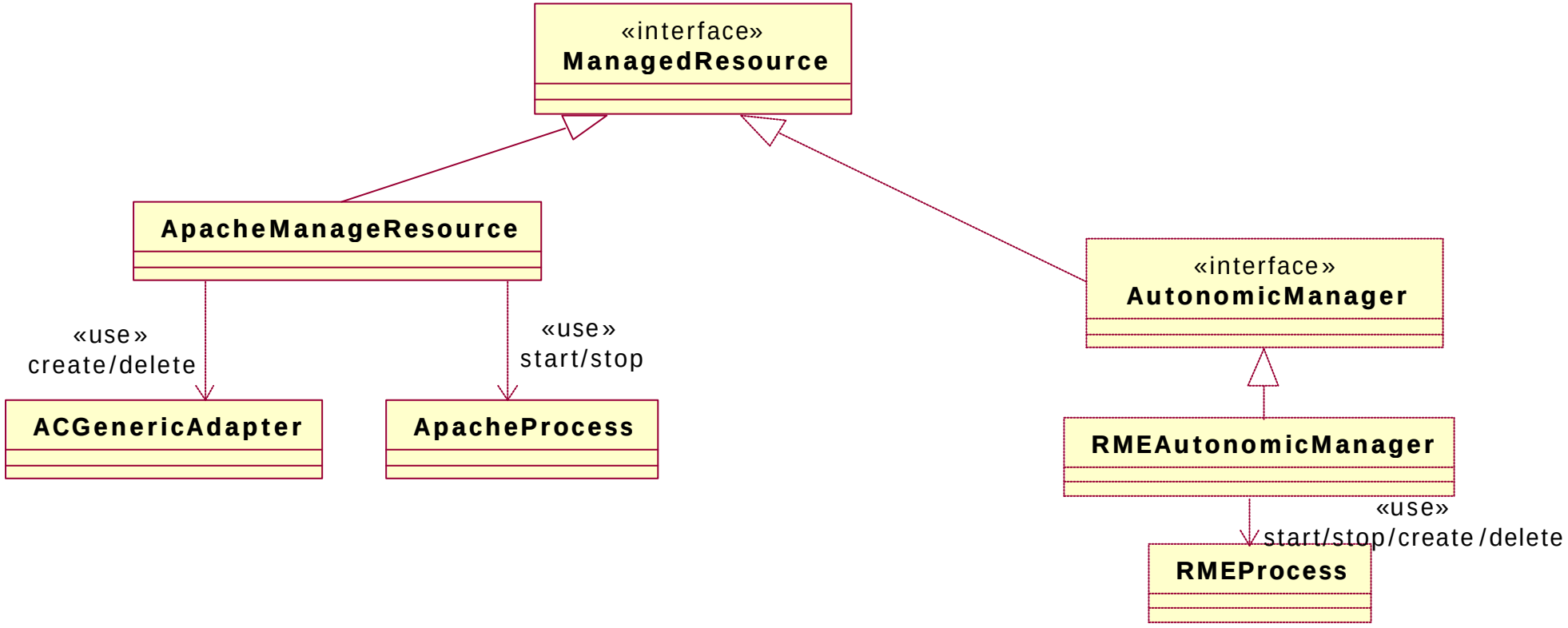
The ePD Toolkit Highlights

- Hierarchical architecture
  - First level analysis identifies something wrong based on local information; it fires an event. Second level analysis can correlate this event with others events in the system (composability)
- Feedback mechanism
  - Allows dynamic wiring and zooming in a particular problem (orchestration)
  - Demonstrate independent “Execute” component of MAPE
    - Plan decomposition, action distribution
- Heterogeneous
  - Tools are combined to achieve a powerful system through standard interfaces and common data representation
- Componentized Autonomic Manager
  - “Standard” interfaces (plugability)
    - RME control, input and output thru Grid Services

OGSA Based Problem Determination



Everything is a Managed Element



Agenda

- The Open Grid Services Architecture
- Autonomic Computing
- The End to End Problem Determination Toolkit
- Lessons Learned

OGSA Summary/Lessons Learned

- The OGSA Programming Model
  - ☑ Reuse portTypes when possible
    - ☑ Extensibility types
    - ☒ Service data elements
  - ☑ Use of Factories
  - ☒ Explicit Lifecycle Management
- The Notification Model
  - ☒ Two entities: sink and source
  - ☒ Notifications on state changes only

Haifa Research Lab.

OGSA-based Problem Determination
An Use Case

Benny Rochwerger
Research Staff Member

Nov. 24, 2003