XQuery Discussion

Ravi Subramaniam, Intel Corporation

OGF-20 Manchester, May 7th 2007

History

- Work done in 2003-2004
- First production in 2005
- Larger objectives still work in progress

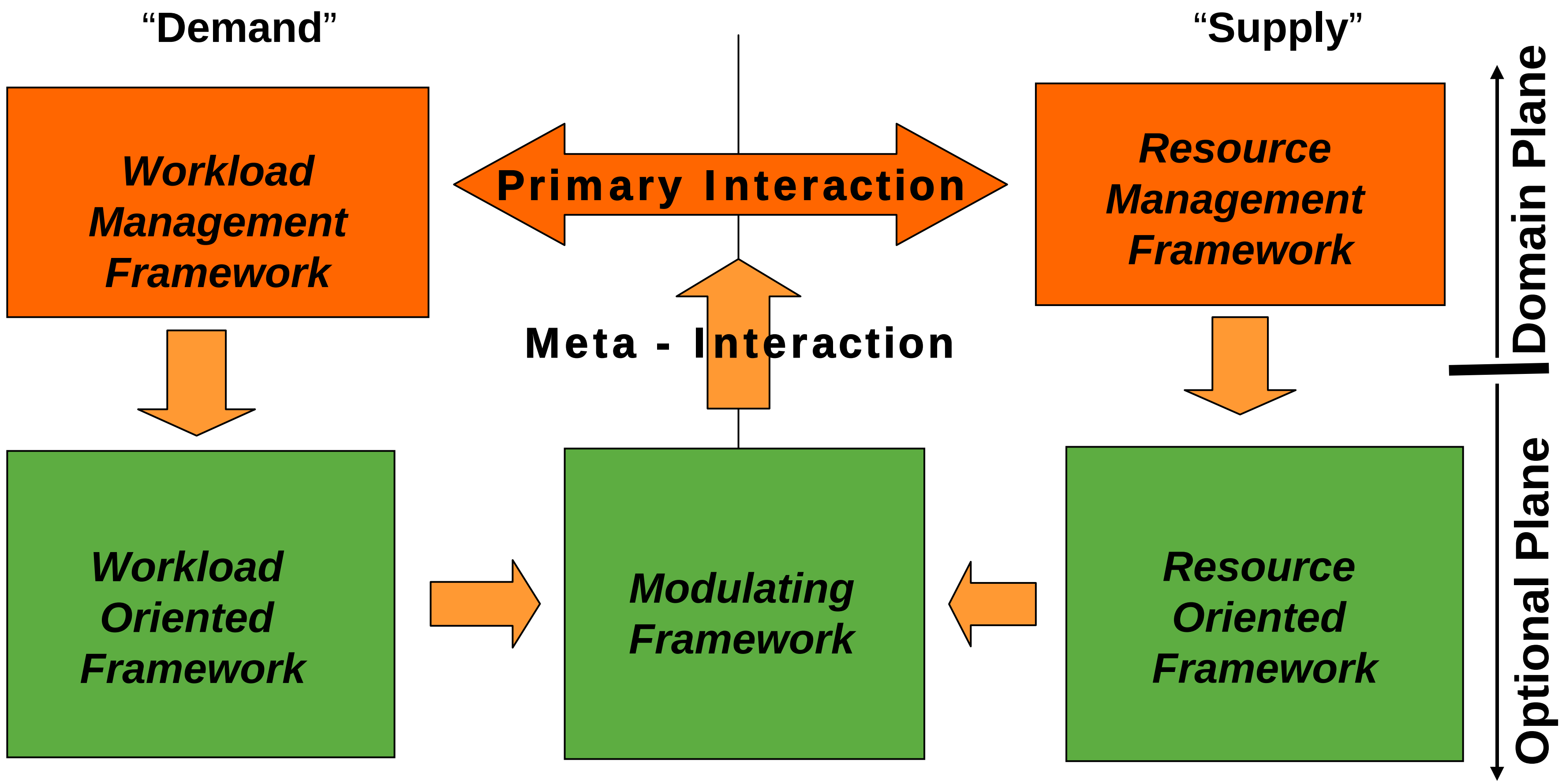
Background –
Set context for our work

- As Grids become large they become increasingly complex systems
  - Current imperative models will not scale – (as we try to identify and use specific system – actually such details would not be necessary too)
  - Need to move to *decentralized control and interactions*
  - Need to migrate to *declarative* modes – specify what we want to achieve rather than what we have to do; different elements in a complex system have different objectives
  - Need to deal with *virtual entities* – such entities may or may not exist aprori (redefines what we understand say provisioning)
  - Policy is distributed – may *not* be able specify all the policies
  - Hierarchies and aggregation should not ideally be determined aprori
- We therefore had to re-think the current paradigms and investigate new ones

Background – job execution example

- Current models for job execution
  - Submit to a “scheduler”
  - “User/Initiator” can only specify requirements – policy only as determined by “central” authority
  - Execution cannot be steered or modified by user once dispatched
- Initiated effort to investigate alternatives to this “portal” based model

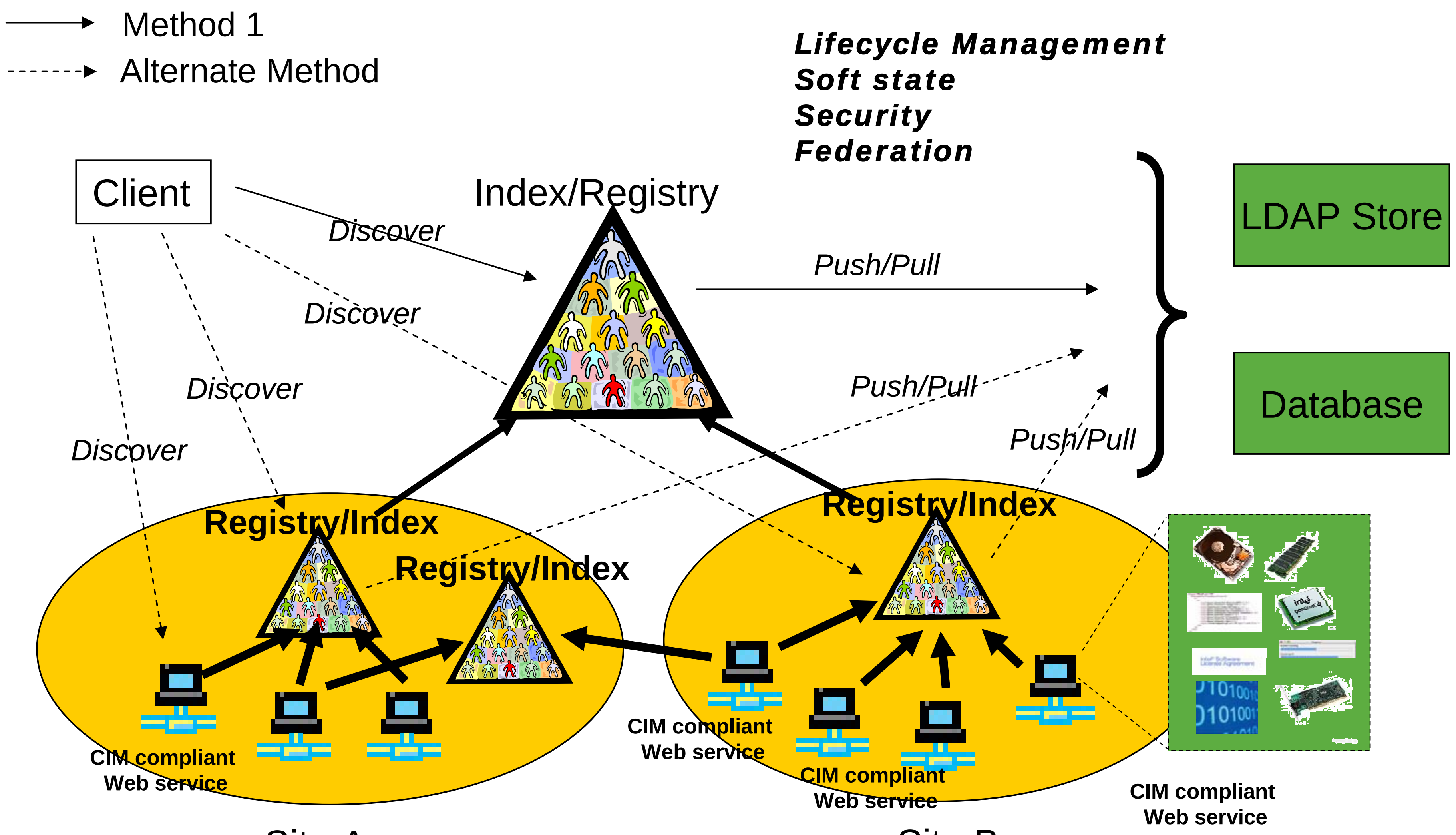
Paradigm


"Demand"
"Supply"
Workload Management Framework
Primary Interaction
Resource Management Framework
Meta - Interaction
Domain Plane
Workload Oriented Framework
Modulating Framework
Resource Oriented Framework
Optional Plane

- Demand-Supply model is *not* another way of describing a request/response
- Describes the dynamic nature of complex systems – system equilibrium – involves selective cooperation and competition

What does this mean?

Registration -> Discovery



Site A

Site B

Registration & Discovery

- ***Registry/Index:***
  - *Web service*
  - *XML based primary store*
  - *Persistence (Relational)*
  - *XML and SQL queries*
  - *Synchronous and asynchronous updates*
  - *Aggregation and Indexing*
  - *Lifecycle Management*
  - *Soft state*
  - *Security*
  - *Federation*
  - *Open Standards-oriented*

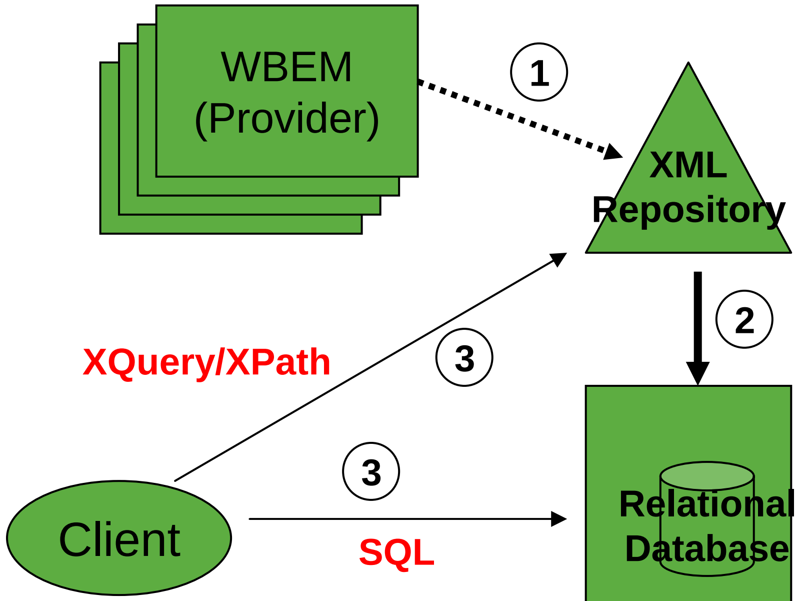
- **Can organize and manage information that is**
  - *Static, quasi static or dynamic*
  - *Software and hardware assets*
  - *Dynamic and/or ephemeral entities like processes and jobs*
  - *State changes*
  - *Configuration changes*
  - *Thresholds*
- **Extensible data model – can be determined at deployment**
- **Self Organizing (by topic)**
- **Can be highly distributed and federated**

Resource Properties

- Based on CIM – use the CIM vocabulary
- Expressed/encoded as CIM-XML (xmlCIM).
- Can be accessed at the source or aggregated – the highly distributed nature mandates a method for consistent querying – hence XQuery
- Aggregation done at the client or using "server-based" aggregation services

Repositories

- Two parts
  - Repository to store native XML
  - Database built from XML data (CIM objects as tables)
    - XML repository modifications synchronized to a relational database (both adds, deletes and updates) – queried using SQL
    - Used XQuery to drive the same ability against native XML



Queries

- Primary usage of XQuery
  - Need to locate the elements representing desired “object(s)” – XPath – parsing and selection can be done on the clients/consumers
  - XQuery allows the finer selection of the required resources (all the resources including data) – modeled on SQL queries
  - Evaluation in a particular context (more flexible than a specific information model)

Queries (contd.)

- Queries can encode policies – both explicit and implicit
- Queries made by clients – pull
- Queries can be stored – evaluated at events – push

Query Examples

- Find a linux machine whose load is < 15%:

```
//INSTANCE[@CLASSNAME='Linux_Processor']/PROPERTY[@NAME='L
  oadPercentage']/VALUE[15>number()]
```

- Find a running job_id on a linux machine whose load > 75%:
  - XQuery:

```
//INSTANCE[@CLASSNAME='Linux_Processor']/PROPERTY[@NAME='L
  oadPercentage']/VALUE[number()>=75]/parent::*/parent::*/
  parent::*/INSTANCE[@CLASSNAME='Job']/PROPERTY[@NAME='sta
  tus']/VALUE[text()='running']/parent::*/parent::*/PROPER
  TY[@NAME='job_id']/VALUE"
```

  - A SQL query would be something like:

```
Select job_id from Job, Linux_Processor
Where Job.host = Linux_Processor.host
And Job.status = “running”
And Linux_Processor.LoadPercentage>75
```