SERVICE-ORIENTED ARCHITECTURE (CONTD..)

DR. MANJUNATH V HEGDE AND DR. VIDYA RAO

L20-L21

PORTALS AND SCIENCE GATEWAYS

- Science Gateway Exemplars
- HUBzero Platform for Scientific Collaboration
- Open Gateway Computing Environments (OGCE)

PORTALS AND SCIENCE GATEWAYS

Gateways provide user-centric environments for interacting with remote computational resources through user interfaces that are typically (but not exclusively) built with web technologies.

TurnKey Solution

HUBzero

Provide end-to-end Solution

Tool Box Solution

Open Gateway Computing Environment (OGCE)

Provides tools to solve a specific problem

PORTALS AND SCIENCE GATEWAYS

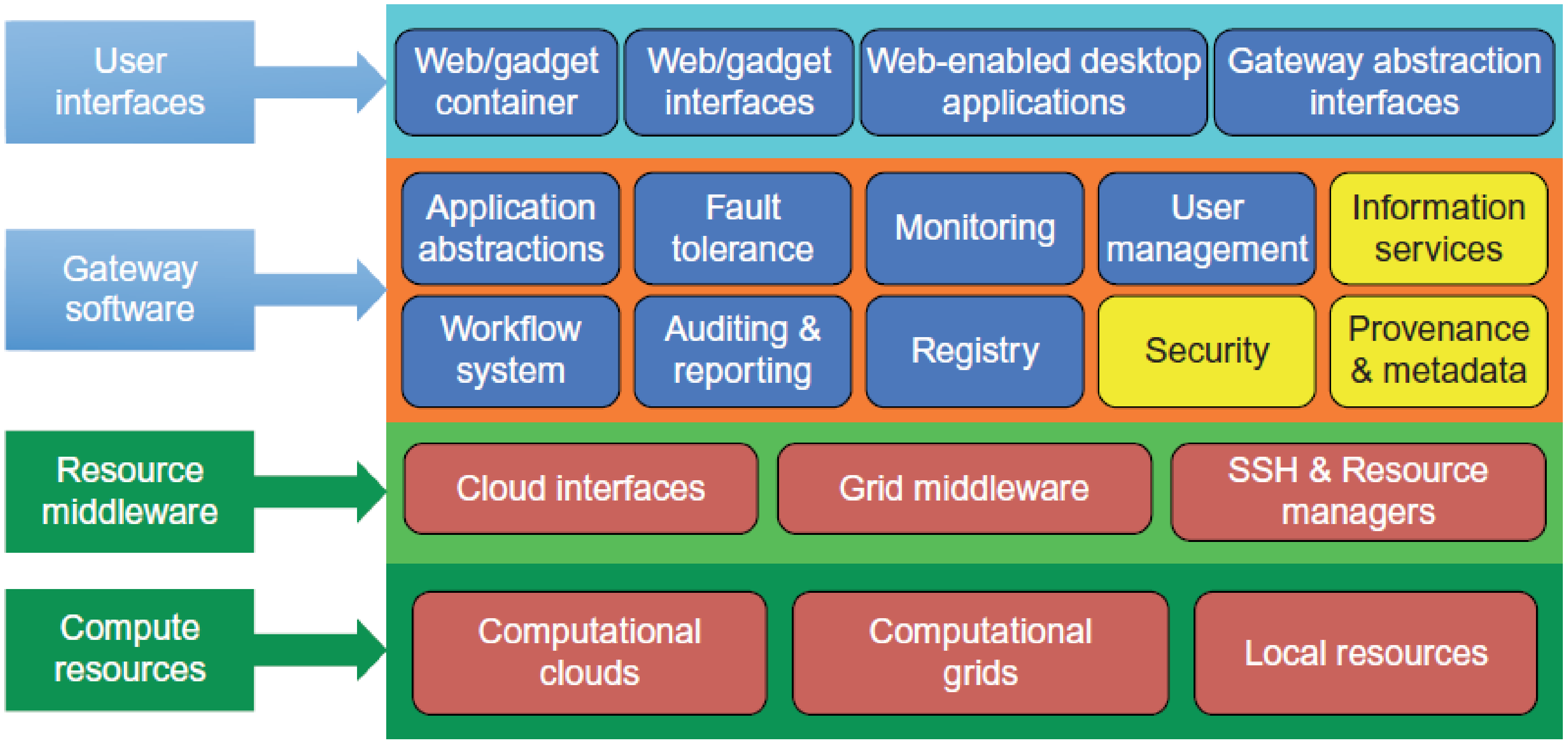


FIGURE 5.8

A gateway component software stack for scientific applications.

HUBzero PLATFORM FOR SCIENTIFIC COLLABORATION

- HUBzero is an open source software platform used to create web sites or “hubs” for scientific collaboration, research, and education.
- Similar to:
    - YouTube.com, HUBzero allows people to **upload content and “publish”** to a wide audience, but instead of being restricted to short video clips, it handles many different kinds of scientific content.
    - Google Groups, HUBzero lets **people work together** in a private space where they can share documents and send messages to one another.
    - SourceForge.net, HUBzero allows researchers to work **collaboratively** on the source code of their simulation programs and share those programs with the community.

HUBzero PLATFORM FOR SCIENTIFIC COLLABORATION

HUBZero Architecture

Linux Apache MySQL PHP

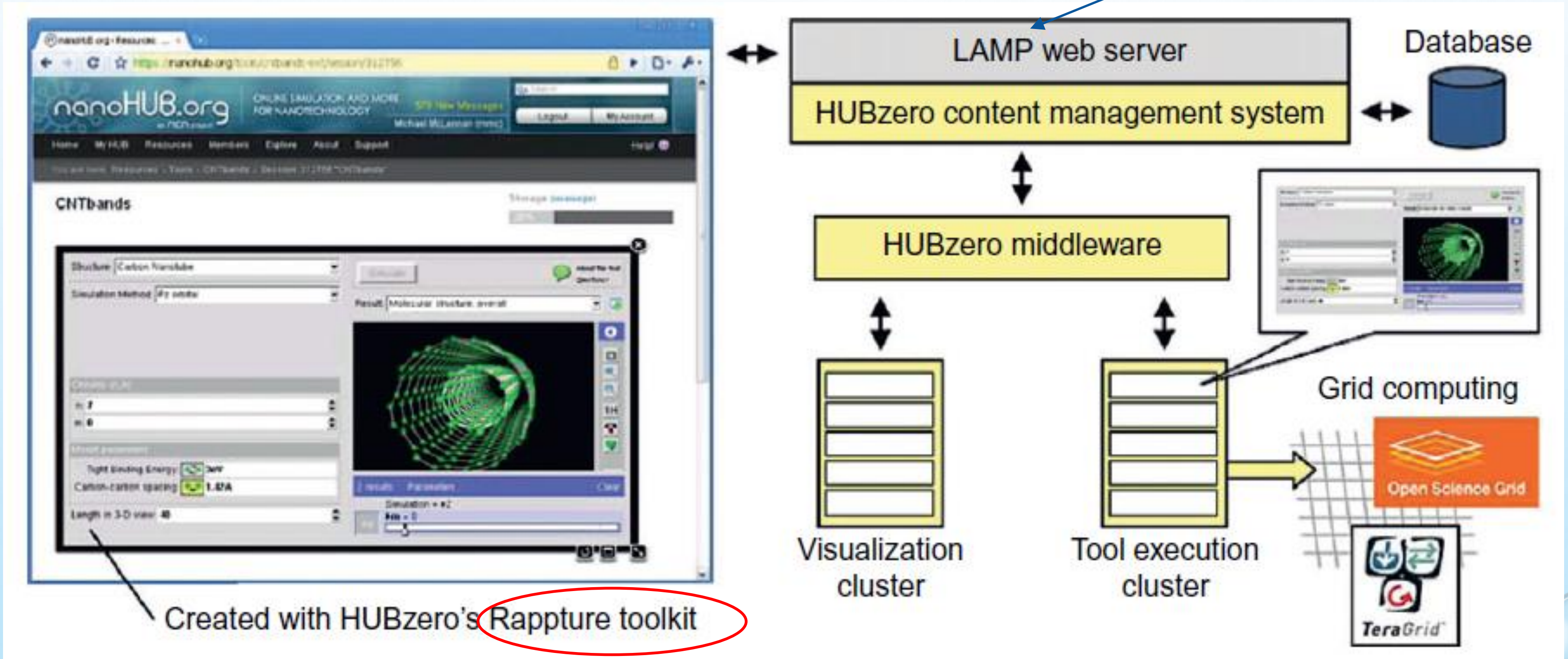


FIGURE 5.9

The HUBzero architecture and its major functional components.

HUBzero PLATFORM FOR SCIENTIFIC COLLABORATION

Rappture toolkit generates GUI containing:

- Operational Features
- Rating and citations
- Content tagging
- User support are
- Wikis and blogs
- Usage metric
- Future design

OPEN GATEWAY COMPUTING ENVIRONMENTS (OGCE)

- Gateways are tools that enable interactive, web-based science, education, and collaboration.
- OGCE open source gateway software that is used in several collaborating gateways.
- OGCE component tools include the following:
  - **OGCE Gadget container**, a Google tool for integrating user interface components
  - **XRegistry,** a registry service for storing information about other online services and workflows
  - **XBaya**, a workflow composer and enactment engine
  - **GFAC**, a factory service that can be used to wrap command-line-driven science applications and make them into robust, network-accessible services
  - **OGCE Messaging Service supports** events and notifications across multiple cooperating services

OPEN GATEWAY COMPUTING ENVIRONMENTS (OGCE)

- OGCE's strategy is based on the toolkit model.
- This strategy has been shaped by the TeraGrid science gateway program and its wide variety of gateways.
- There are obviously many frameworks, programming languages, and tools for building web-based gateways and providing advanced capabilities.
- The OGCE tools focus on scientific application and workflow management and defer issues such as data and metadata management to other projects

OPEN GATEWAY COMPUTING ENVIRONMENTS (OGCE)

Workflows:

The OGCE scientific workflow system provides a programming model that allows the scientist to program experiments using application web services.

Scientific Application Management:

Wrapping scientific applications as remotely accessible services

Gadget Container:

Workflows, registries, and service wrappers have both client- and server-side pieces. The OGCE builds most of its default user interfaces as gadgets.

Packaging:

Applications are packed with required packages that ensure the application is running smooth.

DISCOVERY, REGISTRIES, METADATA, AND DATABASES

- In SOA, business services need to discover appropriate services to use and to integrate with.
- A registry requires a set of data structure specifications for the metadata to be stored in the registry.
- Registry operations: **Create, Read, Update, and Delete (CRUD)** for storing, deleting, and querying the data to store metadata for ownership, containment, and categorization of services.
- Registries usually contain three categories of information:
  - **White pages** contain name and general contact information about an entity.
  - **Yellow pages** contain classification information about the types and location of the services the entry offers.
  - **Green pages** contain information about the details of how to invoke the offered services

DISCOVERY, REGISTRIES, METADATA, AND DATABASES

UDDI and Service Registries

- UDDI : Universal Description, Discovery And Integration
- UDDI specifications define a way to describe, publish, and discover information about web services by creating a platform-independent, open framework.

PUBLIC

Public registry is a logically centralized distributed service

PRIVATE

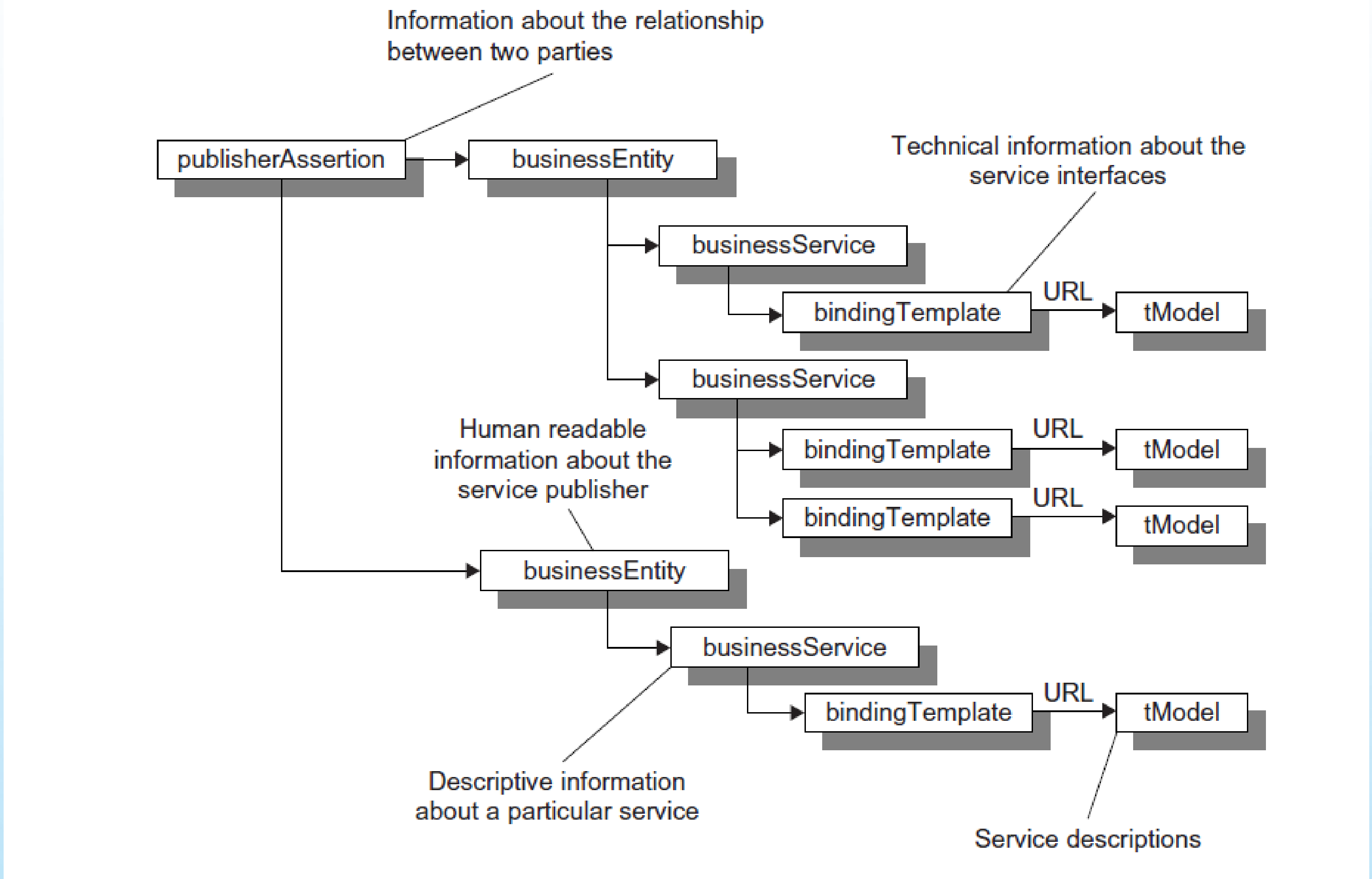
A private registry is only accessible within a single organization or is shared by a group

DISCOVERY, REGISTRIES, METADATA, AND DATABASES

Data in a UDDI registry is organized as instance types:

- ***businessEntity*** Describes an organization or a business that provides the web services, including the company name, contact information, industry/product/geographic classification,
- ***businessService*** Describes a collection of related instances of web services offered by an organization, such as the name of the service, a description, and so forth
- ***bindingTemplate*** Describes the technical information necessary to use a particular web service, such as the URL address to access the web service instance and references to its description
- ***tModel*** A generic container for specification of WSDL documents in general web services
- ***publisherAssertion*** Defines a relationship between two or more businessEntity elements

DISCOVERY, REGISTRIES, METADATA, AND DATABASES



Information about the relationship between two parties
publisherAssertion
businessEntity
Technical information about the service interfaces
businessService
bindingTemplate
URL
tModel
businessService
Human readable information about the service publisher
bindingTemplate
URL
tModel
bindingTemplate
URL
tModel
businessEntity
businessService
bindingTemplate
URL
tModel
Descriptive information about a particular service
Service descriptions

FIGURE 5.10

UDDI entities and their relationship.

DISCOVERY, REGISTRIES, METADATA, AND DATABASES

UDDI provides a set of APIs:

- **UDDI Inquiry API** - to find the set of registry entries such as business, service, binding, etc.
- **UDDI Publishers API -** enables add, modify, and delete entries.
- **UDDI Security API** Allows users to get and discard authentication tokens
- **UDDI Custody and Ownership Transfer API** Enables registries to transfer
- **UDDI Subscription API** Enables monitoring of changes in a registry by subscribing to track new, modified, and
- **UDDI Replication API** Supports replication of information between registries so that different registries can be kept synchronized

DISCOVERY, REGISTRIES, METADATA, AND DATABASES

Databases and Publish-Subscribe

- Publish-subscribe is a design pattern that enables asynchronous interaction among distributed applications.
- Many high-level applications regularly query the database in order to adapt their execution according to this information.
- Such periodic data polling is **not only inefficient and unscalable but also resource-demanding on both sides.**
- The publish-subscribe mechanism, already largely adopted in the implementation of today’s applications, solves this issue.
- In a publish-subscribe interaction, event subscribers register to particular event types and receive notifications from the event publishers when they generate such events.

DISCOVERY, REGISTRIES, METADATA, AND DATABASES

Databases and Publish-Subscribe(contd..)

- Publish-subscribe systems are classified as either as:

Topic-based

- In topic-based systems, publishers generate events with respect to a topic or subject.
- Subscribers then specify their interest in a particular topic and receive all events published on that topic.
- Defining events in terms of topic names only is inflexible and requires subscribers to filter events belonging to general topics.

Content-based

- Content-based systems solve this problem by introducing a subscription scheme based on the contents of events.
- They give users the ability to express their interest by specifying predicates over the values of a number of well-defined attributes.
- The matching of publications (events) to subscriptions (interest) is done based on the content.

DISCOVERY, REGISTRIES, METADATA, AND DATABASES

Databases and Publish-Subscribe (contd..)

- Database systems provide many features that a messaging-based architecture can exploit, such as reliable storage, transactions, and triggers.
- It integrates with publish-subscribe capabilities in the database account for information-sharing systems that are simpler to deploy and maintain.
- PostgreSQL open source database system to include publish-subscribe middleware functionality.

DISCOVERY, REGISTRIES, METADATA, AND DATABASES

Databases and Publish-Subscribe (contd..)

PostgreSQL:

- Based on the integration of active databases and the publish-subscribe communication model to form a global event-based system.
- Databases define and advertise change events, and clients subscribe to events of interest, and can refine their subscriptions through content-based filter expressions.
- This allows a database system in the local domain to function as an event broker (broker), reliably routing events among publishers, subscribers, and other brokers.
- This integration simplifies information management by grouping security, configuration (e.g., type schema), and recovery tasks for database and pub/sub operations under the same interface.

DISCOVERY, REGISTRIES, METADATA, AND DATABASES

Databases and Publish-Subscribe (contd..)

Oracle Publisher-Subscriber

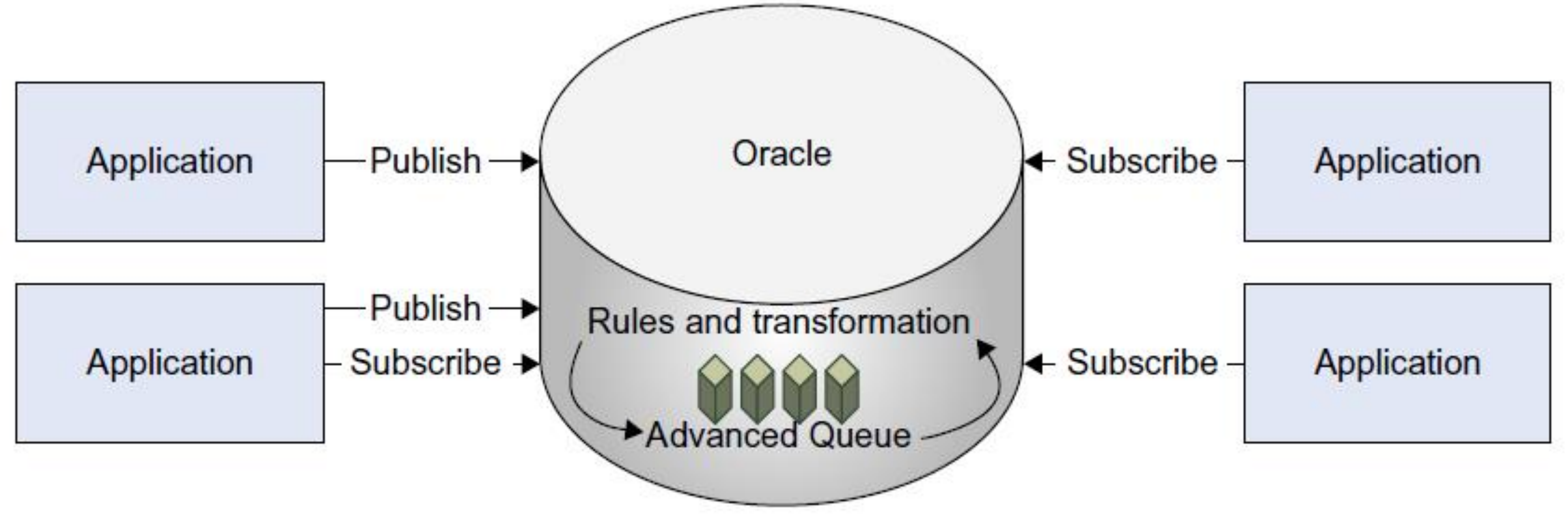


FIGURE 5.11

Oracle publish-subscribe model.

DISCOVERY, REGISTRIES, METADATA, AND DATABASES

Metadata Catalogs

- Metadata is information about data.
- Metadata is important, since it adds context to the data, in order to identify, locate, and interpret it.
- Play a vital role in distributed heterogeneous environments such as grids by providing users and applications the means to discover and locate the desired data and services among lots of sites in such environments.
- Key metadata on the grid includes the name and location of the data resource, structure of the data held within the data resource, data item names and descriptions, and user information.

DISCOVERY, REGISTRIES, METADATA, AND DATABASES

Metadata Catalogs

- Metadata services groups
  - Metadata Catalog Service (MCAT)
  - Replica Location Service (RLS)
  - AMGA (the ARDA Metadata for Grid Applications)

DISCOVERY, REGISTRIES, METADATA, AND DATABASES

Semantic Web and Grid

- The grid strives to share and access metadata in order to automate information discovery and integration of services and resources in a dynamic, large-scale distributed environment.
- Meanwhile, the Semantic web is all about automation discovery and integration.
- The Semantic web aims to provide an environment where software agents are able to dynamically discover, interrogate, and interoperate resources and perform sophisticated tasks on behalf of humans, which is not far from the ambition of grid computing.

DISCOVERY, REGISTRIES, METADATA, AND DATABASES

Semantic Web and Grid (contd..)

- **Resource Description Framework (RDF)**, agreed ontologies expressed in a common language.
- RDF is the first language developed for the Semantic web, using XML to represent information (including metadata) about resources on the web.
- RDF uses web identifiers (URIs), and describes resources in terms of simple properties and property values.
- Semantic web services describe and annotate various aspects of a web service using explicit, machine-understandable semantics, that facilitating the discovery, execution, monitoring, and aggregation of resources.

DISCOVERY, REGISTRIES, METADATA, AND DATABASES

Semantic Web and Grid (contd..)

- **Web Ontology Language (OWL):** enables web services to be described semantically and their descriptions to be processed and understood by software agents.
- Three OWL-S subontologies:
  - **Service profile Expresses** what a service does in order to enable service advertisement and discovery.
  - **Service model Describes** how the service works in order to enable service invocation, composition, monitoring, and recovery.
  - **Service grounding Specifies** the details of how to access the service.

DISCOVERY, REGISTRIES, METADATA, AND DATABASES

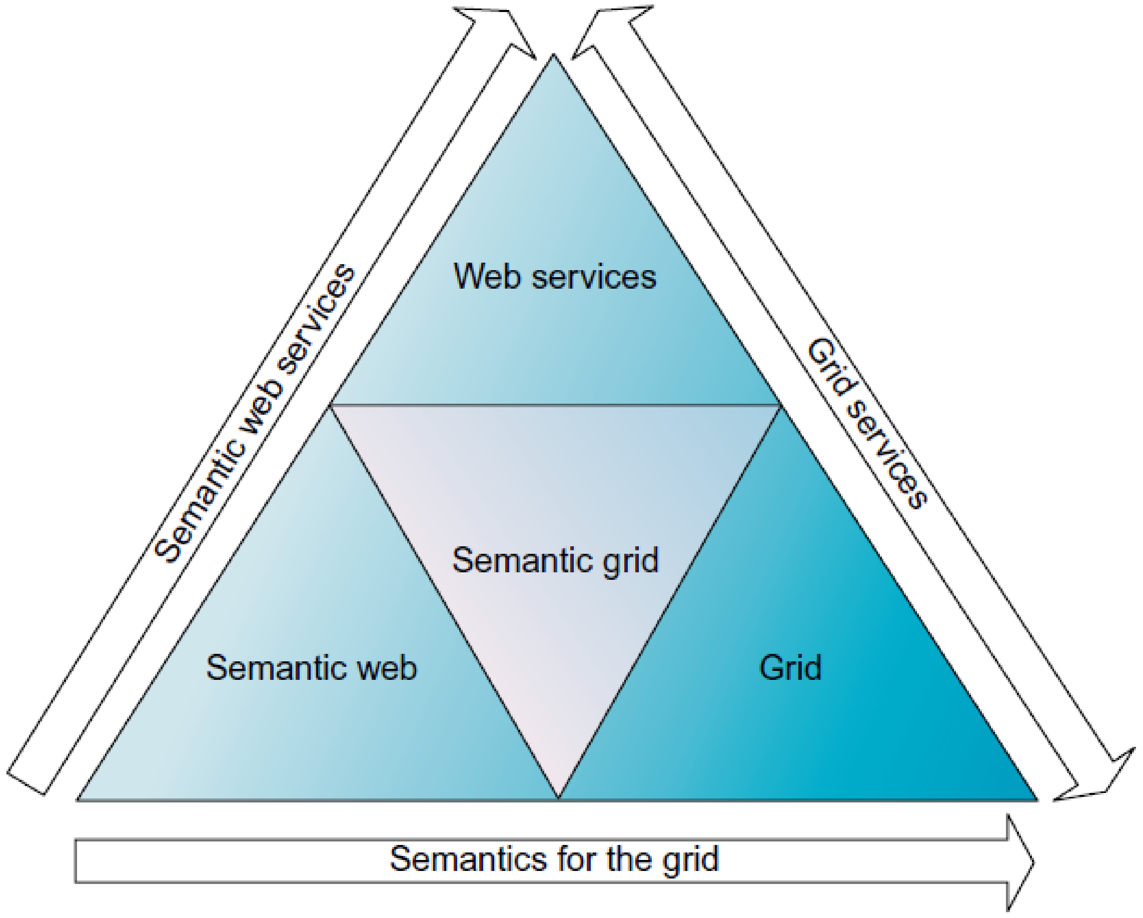


FIGURE 5.12

Semantic grid-related concepts and technologies.

(Courtesy of Goble and Roure, (ECAI-2004), Valencia, Spain, [95])

DISCOVERY, REGISTRIES, METADATA, AND DATABASES

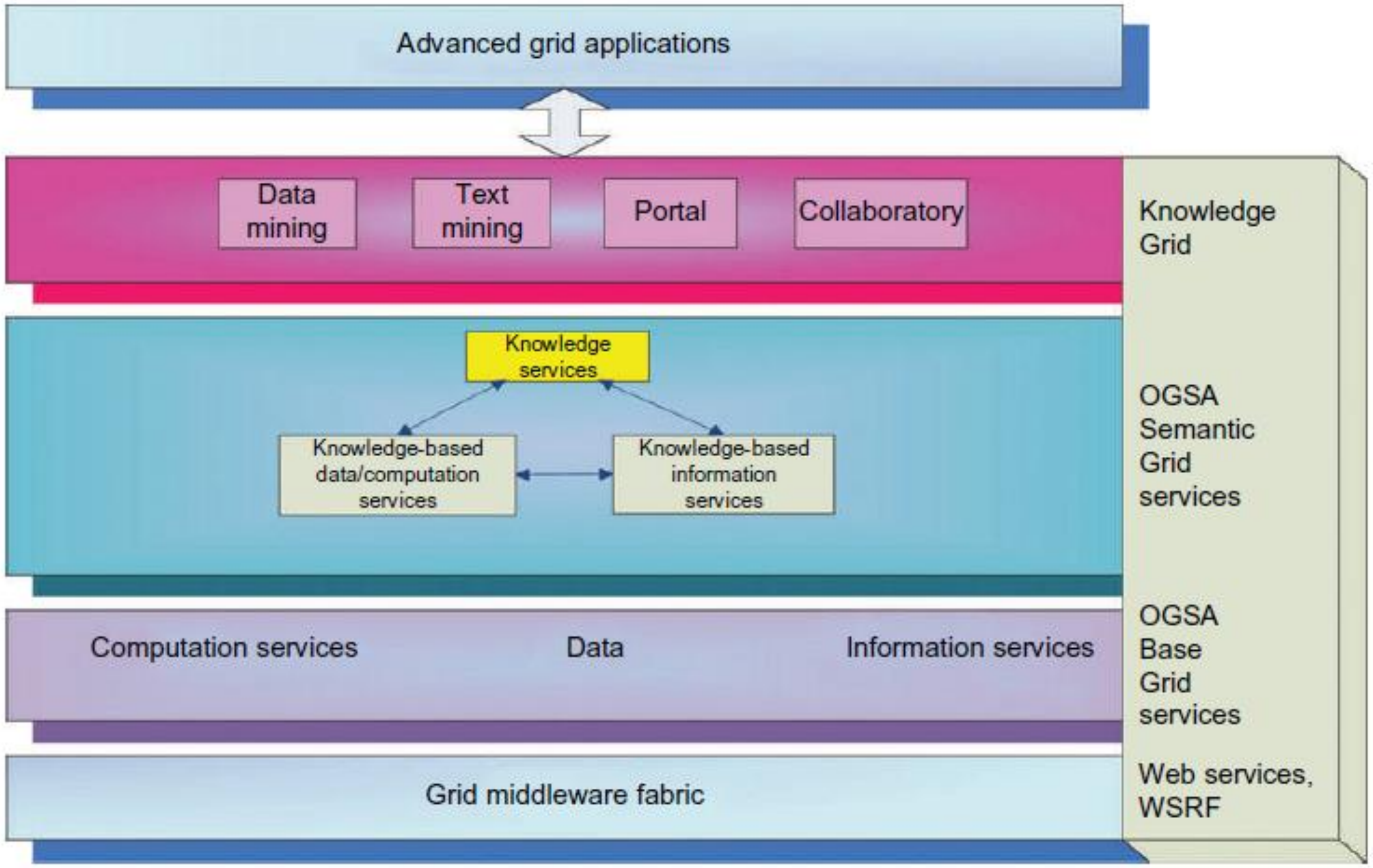


FIGURE 5.13

Semantic grid architecture.

(*Courtesy of Goble and Roure, (ECAI-2004), Valencia, Spain, [95]*)

DISCOVERY, REGISTRIES, METADATA, AND DATABASES

Job Execution Environments and Monitoring

- **A distributed job execution environment often consists of two components:**

Job execution engine

Distributed data management system

- **Job scheduling, resource allocation, and other issues such as fault tolerance**

Provides an abstraction for jobs to access distributed data

Example:

- Google MapReduce,
- Microsoft Dryad

DISCOVERY, REGISTRIES, METADATA, AND DATABASES

Job Execution Environments and Monitoring (contd..)

- **MapReduce:**
- MapReduce was primarily designed to support Google applications that use and generate large data sets.
- It generalizes a map/reduce abstraction from these applications and provides a simple programming model for distributed execution of programs.
- The scheduling mechanism in MapReduce considers data locality by using the data location information provided by the GFS metadata server.
- Similar scheduling strategy is exploited in location aware request distribution algorithms (LARD) for web server clusters

DISCOVERY, REGISTRIES, METADATA, AND DATABASES

Job Execution Environments and Monitoring (contd..)

- **Dryad:**
- Dryad has a similar scheduling mechanism.
- Each execution of a vertex in a DAG has an execution record that contains the state of the execution and the versions of the predecessor vertices providing its input data.
- The scheduler then does the matchmaking by allocating the vertex to the resource.
- The approach has a long history and can be traced back to the Linda programming model.
- As jobs can be dispatched to different nodes, a job execution environment typically needs the support of a distributed data management system for a job to access remote data sets and, sometimes, exchange data with other jobs.

NEXT CLASS….