Reference Model
and
Use Cases
▪
Part 1 of 2

Developed by:
Enterprise Grid Alliance
Reference Model Working Group

*** Version 1.5 ***

As Approved by EGA

10 March 2006

This Page Intentionally Blank

Copyright Notice

Author Information

Contributor	***Organization***
Mike Beckerle	Ascential Software, Inc.
Mukund Buddhikot	Sun Microsystems, Inc.
Debu Chatterjee	Oracle Corporation
Alan Clark	Novell, Inc.
Tim Coulter	Qlusters, Inc.
Michael Enescu	Cassatt Corporation
Brajesh Goyal	Network Appliance, Inc.
Julian Hammersley	Advanced Micro Devices
Masato Kataoka	NEC Solutions (America), Inc.
Raj Kumar	Hewlett-Packard Company
David Pearson	Oracle Corporation
Parviz Peiravi	Intel Corporation
Surendra Reddy	Optena Corporation
Tony Rodriguez	EMC Corporation
Junaid Saiyed	EMC Corporation
Ross Schibler	Cisco Systems, Inc.
Stephen Schleimer	Cisco Systems, Inc.
Michael B Schmitz	Oracle Corporation
Ron Sheen	Fujitsu Siemens Computers
Harald Skardal	Network Appliance, Inc.
Benny Souder	Oracle Corporation
Paul Strong	Sun Microsystems, Inc.
Kazunori Sudo	NEC Solutions (America), Inc.
Bob Thome--editor	Oracle Corporation
Vinod Viswanathan	Optena Corporation

Revision History

Version	Date	Comments
1.0	13 April 2005	Approved by EGA board as a public document.
1.5	10 March 2006	Added extensive set of use cases and created a two part document – EGA Reference Model and Use Cases Parts 1 and 2.

Contents

Figures

Preface

This document is intended to provide a consistent context for the various Enterprise Grid Alliance (EGA) working groups when discussing enterprise grids, their components and the interactions with and between these components. As such, it may also provide a useful reference for those who wish to understand the various components that comprise the modern data center, or *Enterprise Grid*, how they relate to one another, their life cycles and how these may be managed.

The document is organized into two parts. Part 1 contains the **Introduction, EGA Reference Model, Generic Use Cases, Glossary of Terms** and **Appendix A.** Part 2 contains Fifty three (53) specific **Provisioning Use Cases** [1].

In part 1, the ***Introduction*** section provides a simple summary of the purpose of the EGA and it's scope with respect to the set of problems to be addressed in managing enterprise grids and thus driving the adoption of enterprise grid enabling technologies and products. This is included so as to provide a clear scope for the working groups and as background for the rest of the document. This section also includes an overview of the scope of the reference model and how the model is expected to evolve.

This is followed by the ***EGA Reference Model*** section, which provides a simple definition of the *grid component*, classification of the various types of grid component to be found within an enterprise grid, their relationship with each other, their life cycle(s) and how they are managed.

The ***Generic Use Cases*** section, using the previous sections as its foundations, captures the set of generic use cases around the management of the life cycle of the grid component.

The ***Glossary of Terms*** section provides a reference of terms to be used within the EGA and EGA documents, which ensures that when discussing enterprise grids, and the EGA reference model in particular, there is a common understanding of their meaning. Also included are a number of extended discussions providing additional context for certain terms and clarifying or illustrating the relationships between related terms. Terms that have their standard, English meaning are not included. Terms which have a standard meaning within the information technology community and which have been defined appropriately elsewhere are merely referenced.

Appendix A – Grid Component Life Cycle Examples, provides two specific examples of grid component life cycles that build on the generic life cycle already described.

Comments regarding content and format are welcome, and may be directed to the EGA Reference Model working group at ega_referencemodelwg@mail.gridalliance.org.

1 Introduction

1.1 The Enterprise Grid Alliance

The purpose of the *Enterprise Grid Alliance (EGA)* consortium is to drive the adoption of grid computing and the technologies that enable it within enterprise data centers. The EGA is pragmatic and is focused on short and medium term goals that accelerate adoption, as well as the long term objectives in grid computing.

Grid computing is typified by a focus on sharing pools of network-distributed resources to deliver applications and services. Grid computing environments may be typified by -

- The use of network distributed, shared pools of discrete resources to achieve greater performance, scaling, resilience and utilization
- A focus on managing services, rather than on managing discrete components, as grids turn networks into arbitrarily rich/complex fabrics of resources.
- Flexibility or mutability, as components may be regularly repurposed or re-provisioned in line with the goals for the services that run on them
- Application or service architectures that are disaggregated or distributed in nature and which leverage the properties of the fabric of resources, for example traditional multi-tiered applications such as ERP or CRM, Service Oriented Architectures (SOAs) or decomposable compute intensive workloads.
- The consolidation of computing components into [typically] a smaller number of larger resource pools to enable easier provisioning and greater resource utilization.
- The standardization of components and/or their interfaces, configurations, processes and applications.

Enterprise Grid Computing is specifically the use of grid computing within the context of a business or enterprise, rather than perhaps for academic or research purposes. The requirements and challenges associated with its adoption within an enterprise differ, especially in the operational sense.

Enterprise Grids are typically managed by a single enterprise - i.e. an entity, the business, is responsible for managing a networked pool of resources, and a set of services, and for managing the assignment of resources to services in order to meets its goals. The resources and services may or may not be owned by that business, for example in the case of managed services or a service provider/outsourcer. What defines the boundaries of the enterprise grid is management responsibility and control. An enterprise grid may be confined to a single data center or it may extend across several.

Grid computing is compelling in many ways from the perspective of an enterprise – especially with respect to the potential for greater efficiency and agility. However there are often a number problems or issues which prevent the adoption of grid computing technologies or prevent their true potential from being realized. Examples include, but are obviously not limited to –

- Lack of trust in new technologies, fear of being the first.
- The need to support existing core applications.
- The need for interoperability between disparate vendor’s components within an enterprise grid

It is the purpose of the EGA to help address these issues by –

- Identifying key, high priority problems that highlight these issues, for example –
  - How to standardize and automate the provisioning of compute resources or of data.
  - How to provide cross platform, service based, i.e. utility, billing.
  - What is different about security within the context of an Enterprise Grid?
- Driving and proving interoperable/heterogeneous, vendor neutral solutions through
  - Creating demonstrations or reference implementations/architectures.
  - Recommending appropriate use of standards (profiling).
  - Working with relevant standards bodies to identify gaps and overlaps.

The EGA aims to drive interoperability through the use standards, but it does not necessarily aim to be the producer of such standards. Rather the intention is to –

- Identify appropriate existing standards and recommend their use where appropriate.
- Drive enterprise grid requirements into the bodies that own existing standards that may need extending or modification.
- Collaborate with existing standards bodies, in order to find an appropriate venue for defining new standards for which the EGA has determined a need.

Only if all of these options are exhausted will the EGA consider defining it's own standards.

As a consortium that consists of many/most of the key industry vendors who supply the components that comprise the enterprise grid, the EGA is in an ideal position to actually deliver on these goals. It will achieve this through chartering working groups to solve very discrete problems. During the first 12 to 24 months (2004-2005) the working groups will focus on solving discrete problems associated with *commercial enterprise applications* within a single data center.

Commercial enterprise applications are those that tend to be multi-tiered, may have both batch and interactive components, and are often packaged, but may also be heavily customized. Examples include, but are not limited to CRM, ERP, BI etc. Such applications may or may not be grid enabled. The EGA expects to extend the scope of it's working groups into *technical enterprise applications*, i.e. those that tend to be more compute intensive and less interactive, and beyond single data centers in due course.

Priority with respect to working groups is given to the most pressing problems, so that pragmatic, short and medium term solutions and consequent benefits may be delivered, whilst working within the context of the larger picture. The current (as of March 2005) set of working groups includes –

- Reference Model – chartered to deliver context to the other EGA working groups. This is described more fully in section 1.2 below.
- Component Provisioning – initially looking at the issues around server provisioning.
- Data Provisioning – looking at service or application centric management of data.
- Utility Accounting – exploring the issues around reconciled resource utilization with application value, and integration with billing and accounting systems. These are key to enabling utility pricing and delivery.
- Security – exploring what is different when thinking about security in an enterprise grid, i.e. an environment in which components may be shared or rapidly repurposed.

Each of these groups is described in more detail on the EGA web site – http://www.gridalliance.org.

The charter of the Reference Model Working Group, and the aim of this document, is to provide the shared context for all of the EGA's other working groups for their initial phase of work, that is within a single data center and for commercial enterprise applications.

1.2 The Goals Of The Reference Model Working Group

The goal of the reference model working group is simple – to provide a single, shared context for the various EGA working groups. This shared context includes –

- A common set of terms. The goal is not to define lots of new terms. Rather it is to provide some precision around terms in common use but which are directly applicable to the work of the EGA working groups. Of course, some new terms are also defined which have very specific meaning within the EGA, nonetheless the goal is to use, leverage and cross-reference as much as possible. The set of terms is strongly aligned with, indeed driven by, the reference model itself.
- A reference model. This provides a common context in terms of enumerating and classifying the set of components to be managed within an enterprise grid, their life cycle and how they are managed. Again some of the work will be very EGA specific, but the goal is not to reinvent the wheel, so to speak, but rather to leverage wherever possible. Thus the initial, high level, informal specification will be quite EGA specific, but the more formal definition to follow may well simply cross reference other standards or work, for example CIM (the Common Information Model) from the DMTF (Distributed Management Task Force) or OGSA (Open Grid Services Architecture) from the GGF (Global Grid Forum). At present, no other bodies adequately capture the nature of the components within an enterprise grid, their life cycle and how this is managed within a single model. There is an opportunity for the EGA to make a start here and to provide impetus to the bodies that already own parts of the picture (DMTF, GGF and SNIA), to either unify their models or at least make them consistent with each other.
- A set of use cases. These provide the overall customer or user centric view of the sets of problems that the EGA intends to address. This document captures the generic set of use cases, essentially around managing the life cycle of services and the resources they depend on or consume. By taking a general approach, the use cases provide the wider context for the other working groups, who are then expected to express or describe the specific problems they wish to solve within specific, more detailed extensions or versions of those provided by the reference model working group. Part 2 of this document includes a large set of specific provisioning use case examples.

1.3 Reference Model Development Plan

As outlined above, version 1 of the EGA reference model is quite basic and informal. Version 1.5 includes an extensive set of provisioning uses cases. Version 2 will include a more formal model as appropriate, will include additional use cases, and will include changes and/or modifications resulting from use of the model elsewhere within the EGA and externally.

Approved revisions to the document, will be incorporated into the updated versions of the document. Updates will be published as and when required, specifically when either a major change or a sufficient volume of smaller changes is incorporated.

2 The EGA Reference Model

2.1 The Enterprise Grid

An inferred goal of the reference model is to provide as simple a model of the enterprise grid, or data center, as possible, yet one that allows the complexity inherent in modern data centers to be adequately represented.

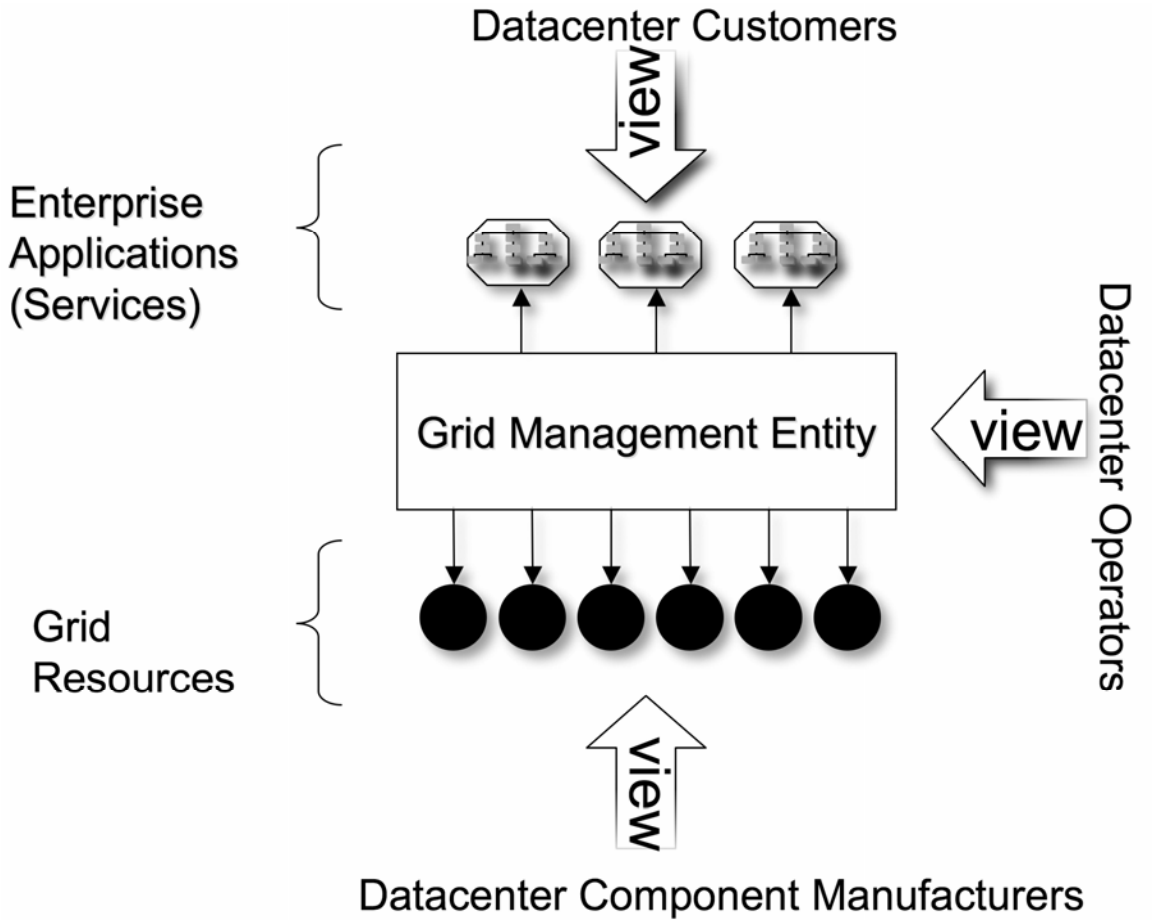

Figure i - Basic View Of A Data Center

Figure i - Basic View Of A Data Center – shows one way of breaking down the various components within an enterprise grid - into services, the resources that they consume and the entity that manages the association or mapping of services onto resources. However, upon close examination it can become confusing as to what is a service and what is a resource. For example services can often be broken down into what are considered constituent services. Thus a bookstore service may perhaps be broken down into database and business logic components or services. The database component could likewise be broken down into database instance, operating system instance and server components. Are these services or resources? The answer usually depends on one's perspective, thus to avoid confusion the EGA reference model defines the *grid component*.

This now allows a more precise definition of an enterprise grid -

An ***enterprise grid*** is a collection of interconnected (networked) ***grid components*** under the control of a ***grid management entity***.

Enterprise grids are typically differentiated from more traditional data centers by management practices and technology which –

- Enable service or application centric management rather than component centric management.
- Enable pooling and sharing of networked resources.

Of course there are other differentiators, as enumerated above (see section 1.1 above) but these two tend to lie at the core of enterprise grids.

The next sections describe –

- The grid component, its life cycle and the classification of types of grid component.
- The grid management entity, what it is and what it does.
- Generic grid component use cases.

2.2 Grid Components

2.2.1 Definition

The ***grid component*** is a super class of object from which all of the components that are managed within an enterprise grid are descended or derived. This may include, but is not necessarily limited to –

- Traditional resources such as servers, network switches, routers, SAN switches, disk arrays and so forth.
- Applications or services such as databases, an ERP service, an online bookstore etc.
- Everything in between which could be considered to be a meta-resource or a component of a service.

NOTE *– This model is complementary to others such as the Distributed Management Task Force's (DMTF)[1] Common Information Model (CIM)[2]. Thus each grid component could be modeled as a CIM_ManagedSystemElement if more compositional detail were required.*

The grid component is also analogous the Office of Government Commerce's (OGC) IT Infrastructure Library's (ITIL) Configuration Item (CI)[6]..

2.2.2 Recursion/Dependency Mapping

A grid component such as a Business Intelligence service may be broken down into other grid components, such as a database, application servers and so forth. Likewise grid components may be combined or associated, resulting in other grid components. Grid components are thus potentially recursive.

Ultimately decomposition or mapping resolves to a set of physical grid components that cannot be broken down or decomposed from a management perspective. This is illustrated in Figure ii through Figure iv below.

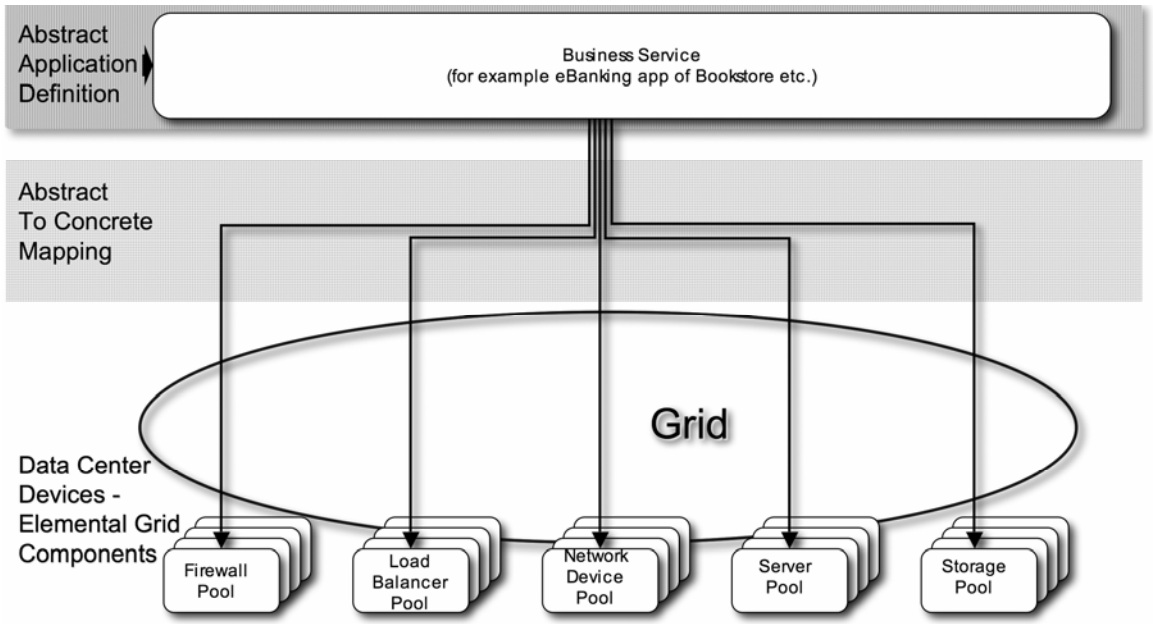

Figure ii - Abstract To Concrete Component Mapping 1

In Figure ii - Abstract To Concrete Component Mapping 1 – the high level mapping between a service such as a bookstore or electronic banking service and the underlying resources, in this case pools of servers, network devices and so forth is shown.

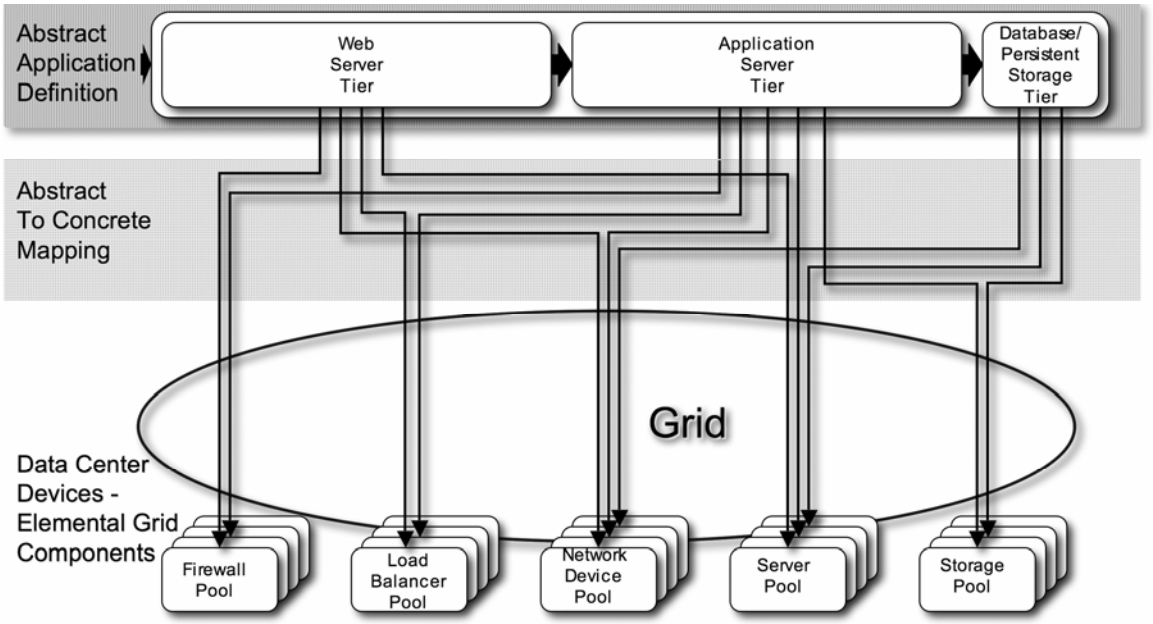

Figure iii - Abstract To Concrete Component Mapping 2

In Figure iii - Abstract To Concrete Component Mapping 2, this is broken down into relationships between tiers, in this case web server tier, application server tier and database/persistent storage tier, and sets of servers, networks devices.

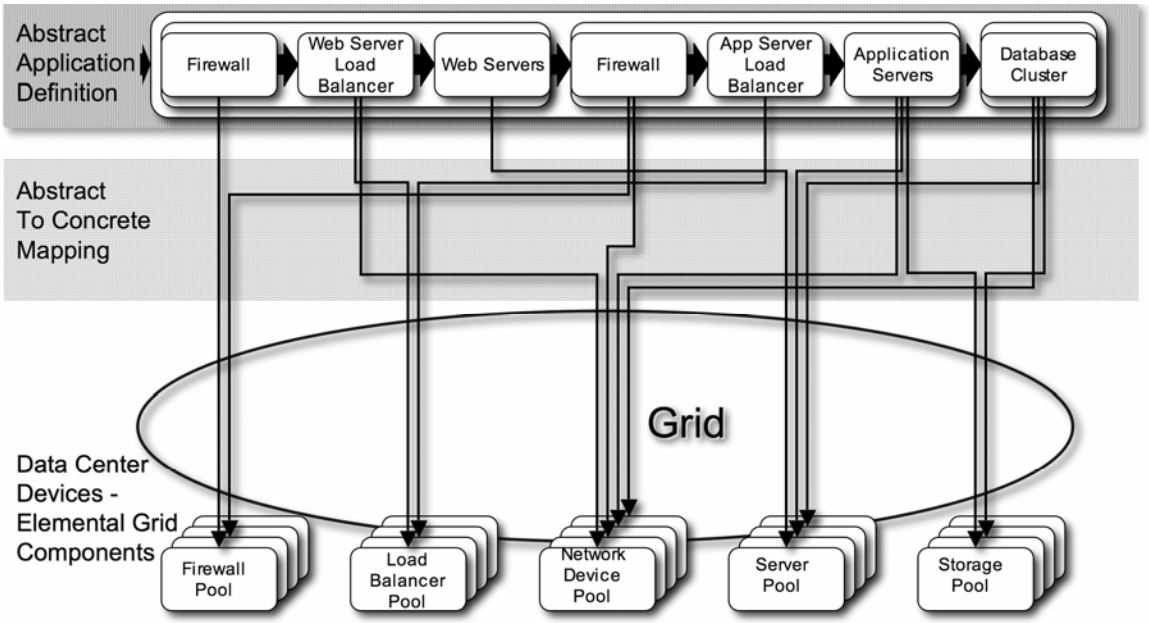

Figure iv - Abstract To Concrete Component Mapping 3

Finally, in Figure iv - Abstract To Concrete Component Mapping 3, more discrete components are mapped onto elemental grid components.

Figures ii through iv are 3 among many different possible renderings of mappings of abstract grid components (typically thought of as applications or services), onto underlying, concrete grid components (typically thought of as resources).

In fact the dependencies or associations that define a business service or application may be represented in the form of a tree, or more correctly (if each connection had an arrow head pointing down to the component depended upon) as a directed acyclic graph (DAG) or acyclic digraph (the two terms are synonymous) – see Figure v - A Grid Component Dependency Graph – below. This particular rendering of the graph, with the connections (edges) between the grid components (nodes) representing instantiation or installation dependencies, is useful when considering grid component provisioning. Thus provisioning of the grid component that is the bookstore requires the provisioning of the persistent storage, business logic and presentation tiers. Provisioning the presentation tier grid component requires the provisioning of the web server farm grid component, and so on, until all of grid components in the DAG are provisioned.

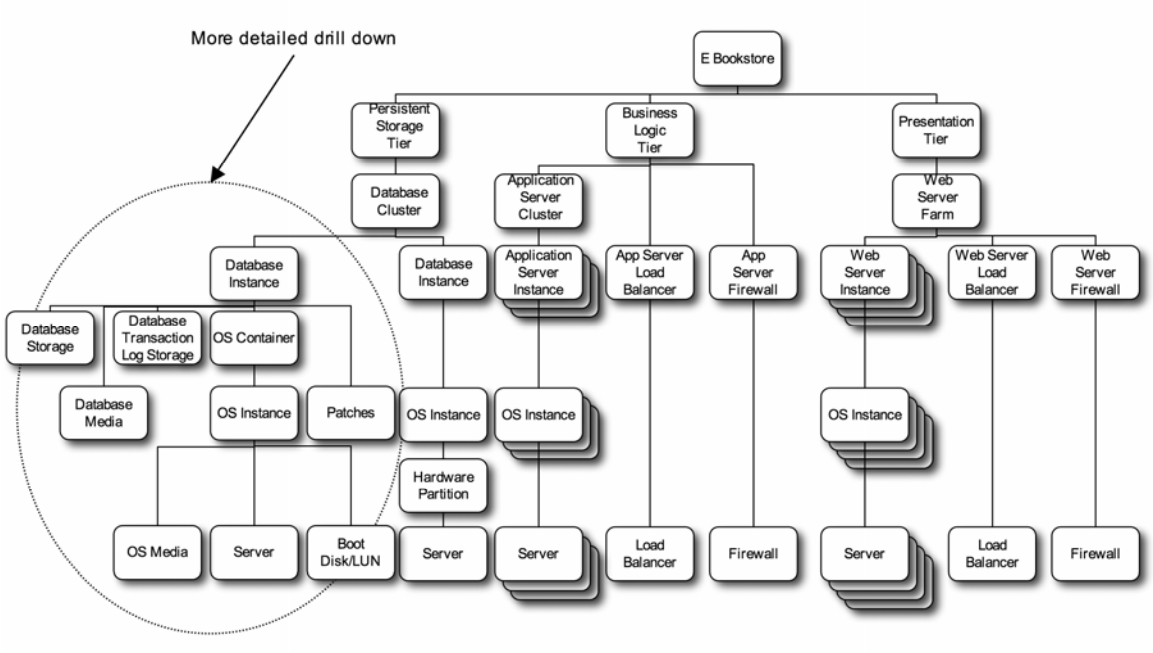

Figure v - A Grid Component Dependency Graph

Other renderings of the graph, with different connections (edges) between the grid components (nodes) may be used to represent performance, scaling, availability and security dependencies.

2.2.3 Attributes And Properties

Grid components have attributes or properties associated with them, such as

- Dependencies or constraints for instantiation, performance, scaling, availability, security and so forth, which may be represented by or deduced from one or more DAGs.
- Service Level Objectives (SLOs) which enumerate the desired attributes of the grid component instance, that would allow the business service or application of which it is a part, to satisfy its Service Level Agreement (SLA). SLOs and SLAs are discussed in more detail in section 2.3.5.2 below.
- Configuration information, which may include versioning, configuration parameters and options, and so forth.
- Data and metrics associated with its state and progress against SLOs.

These attributes may be implemented as internal attributes of a realized grid component or they may only be explicitly implemented as attributes associated with a managed grid component within the Grid Management Entity. The EGA reference model does not make a distinction. It is implementation neutral. – This is discussed in more detail within section 2.3 - The Grid Management Entity.

2.2.4 Life Cycle

2.2.4.1 Definition

The grid component is not a static entity. It has state that may change. Within an enterprise grid, or indeed a traditional data center, certain states, or perhaps state transitions, may be considered to define the life cycle of a grid component, including –

- Provisioning
- Ongoing management
- Decommissioning

This life cycle, at least at a high level, is common to all grid components. This is illustrated in Figure vi - The General Grid Component Life Cycle.

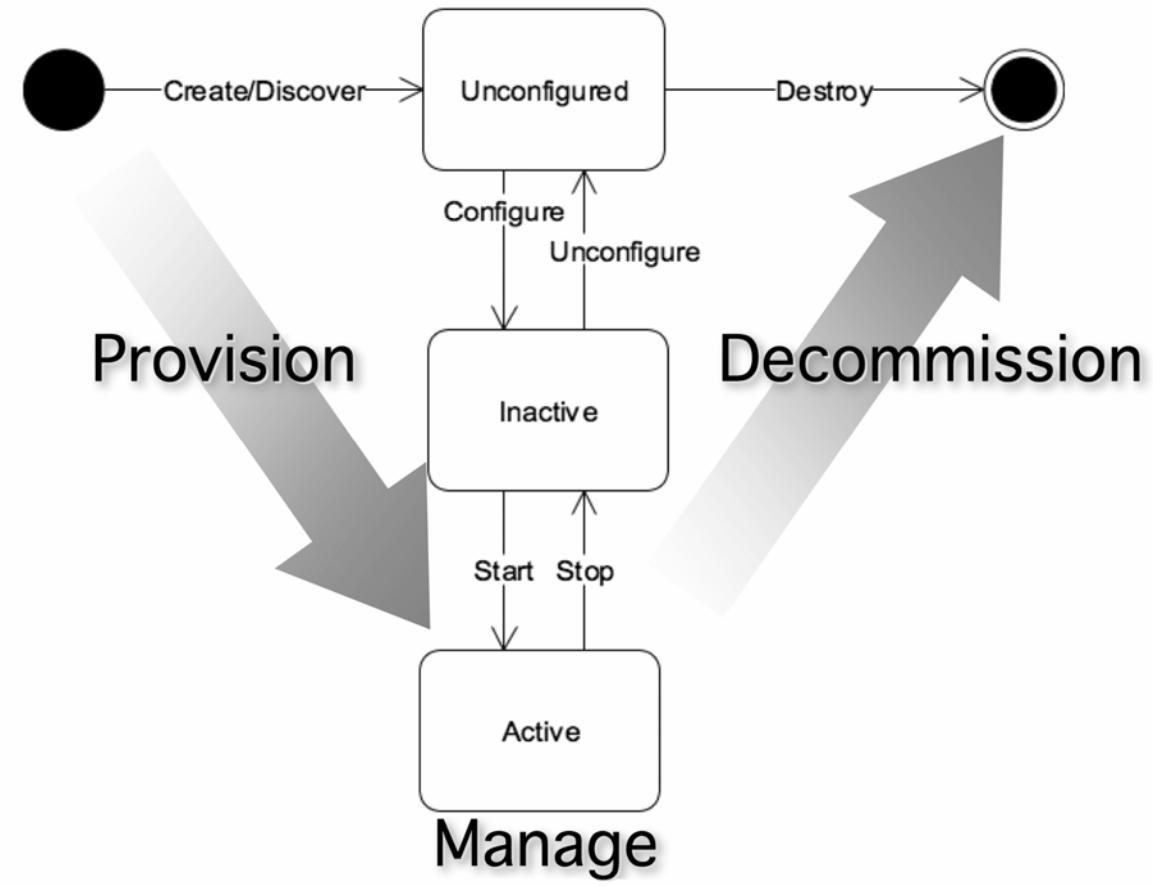

Figure vi - The General Grid Component Life Cycle

Note that the diagram only captures those high level states most pertinent to provisioning. Each state is a macro-state and may encapsulate many micro-states. For example the *Inactive* state could have sub-states such as *Ready* or *Failed*, with a transition from *Failed* to *Ready* required before the grid component can be started again.

As mentioned in section 2.2.3, Attributes And Properties, the grid component has attributes or properties associated with it, such as

- Dependencies or constraints.
- Service level objectives (SLOs).
- Configuration information.
- Data and metrics associated with its state and progress against SLOs.

Which attributes and properties are set during which state or state transitions is left unspecified as these may vary by implementation. Indeed specific implementations may not explicitly include all of the states or transitions – they may be implied.

2.2.4.2 More Complex Life Cycles

Combining the state model with DAGs, as discussed in section 2.2.2 (Recursion/Dependency Mapping), allows the life cycles of complex or more abstract grid components, such as a bookstore, to be expressed in terms of the life cycles of it's constituents or those components upon which it depends.

Referring to Figure vii, provisioning the Database Instance, consists of provisioning each of the grid components upon which it depends. going from left to right and from leaf nodes up. Thus the order for provisioning of the grid components that would result in the provisioning of the database instance is: 1, 2, 3.1.1, 3.1.2, 3.1.3, 3.1, 3.2, 3, 4, 5 and finally the database instance itself.

Note that the dependency of a grid component on others may be more complex, including dependencies on specific micro-states of the *Active* state and on specific attributes.

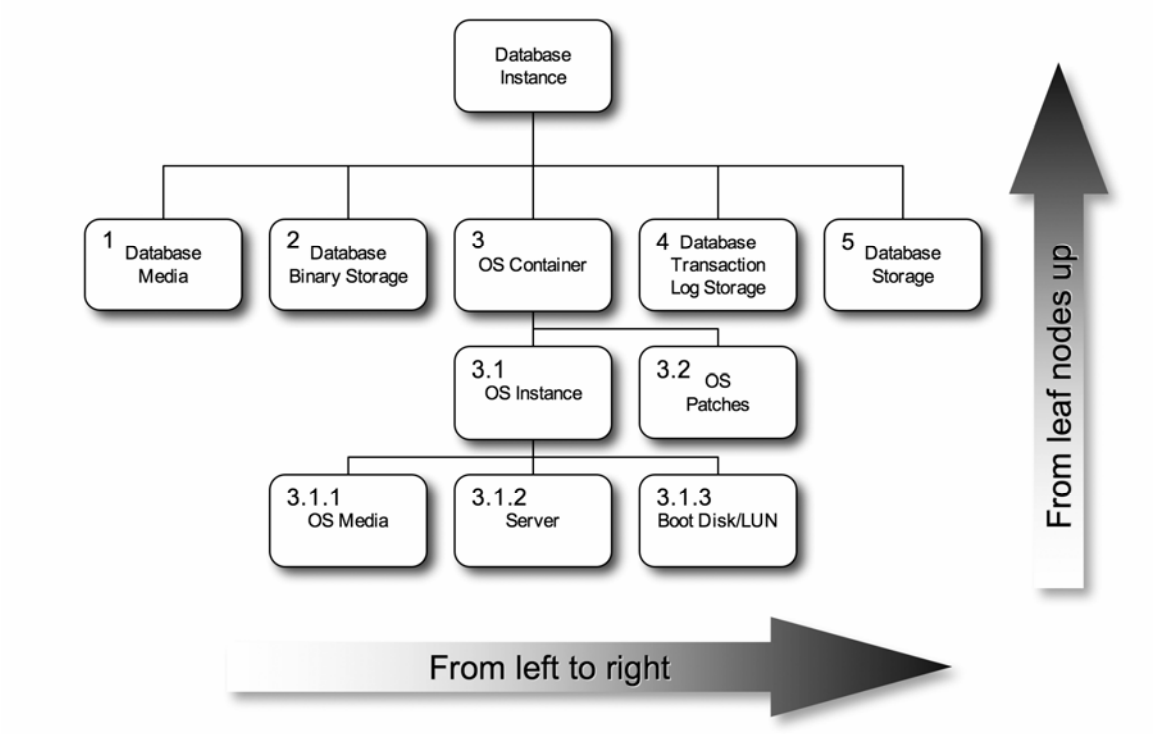

Figure vii - Applying Verbs To The DAG

2.2.5 Classification

In addition to being able to connect grid components together into DAGs to allow for the resolution of various dependencies, it is also possible to group various grid components together into similar groupings or classes that share similar properties.

Figure viii - Some Grid Component Classes – below, shows a possible breakdown of grid components into various classes. The layering of classes and naming is to some degree arbitrary and reflects a view based on provisioning. Thus one would provision from the "bottom up" so to speak.

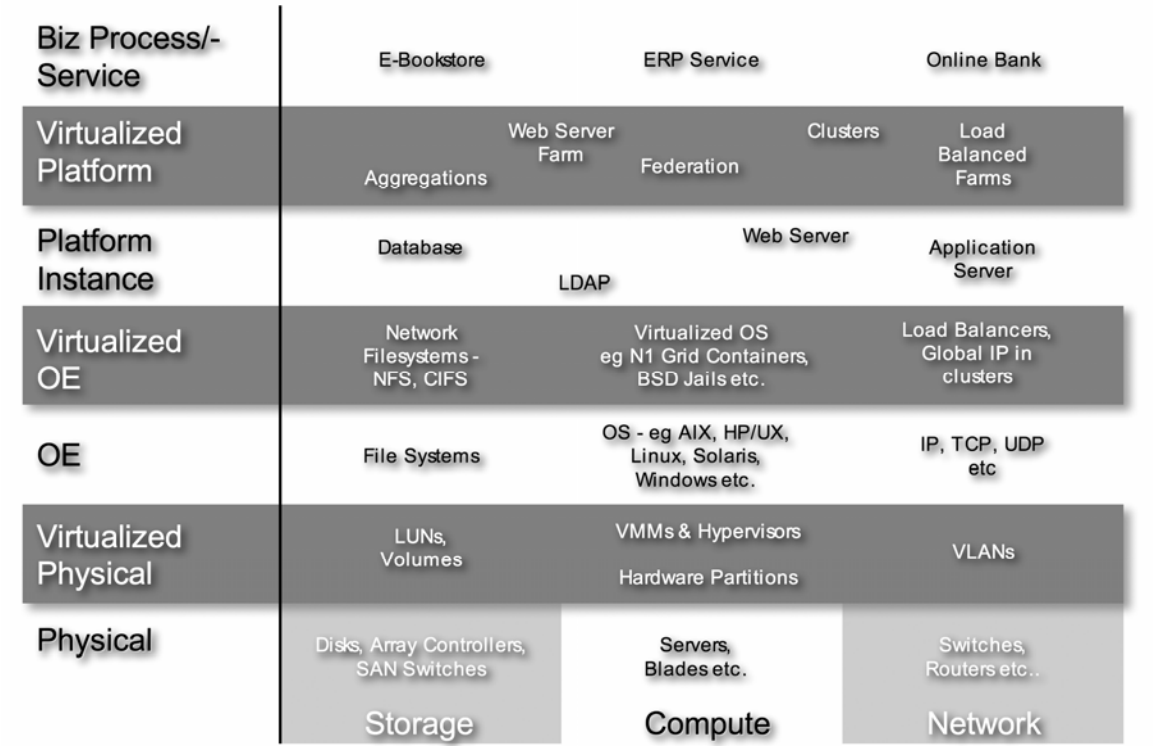

Figure viii - Some Grid Component Classes

The classes of grid components range from the basic physical or elemental grid components to the business processes, which are ultimately realized as services that run on those physical components, as well as everything in between. These include both the managed and managing components.

The various grid components within the data center are shown as layers of increasingly abstract objects. Each layer virtualizes, or abstracts, the one below to enable greater flexibility, efficiency and manageability. At the bottom are physical grid components, which are ultimately, through the layers, related to the business processes that are realized as services, which run on them.

The layers are, from the bottom –

1) Physical – Physical grid components are the basic physical building blocks, which comprise the enterprise grid fabric of resources. Examples include, but are not limited to discrete, traditional servers, network switches, disks and so forth.

2) Virtualized Physical – Virtualized Physical grid components are those that present essentially the same interface as the physical grid component but which may be realized or implemented differently. For example, a storage LUN presents the

interface and properties of a disk, yet the underlying physical component may be a part of a disk, a whole disk or an aggregation, such as a RAID stripe, presented by a volume manager or storage controller. Similarly both hardware partitions and virtual machine managers (VMMs) allow a single rack of resources or a server to host multiple operating environments, each of which behaves as if it had it's own dedicated server.

3) Operating Environment (OE) – OE layer grid components abstract physical grid components, whether virtualized or not, presenting them as resources and making them available to grid components such as services or applications, as well as enabling higher level and more scalable management. These components include such things as file systems, operating environments, IP based LAN segments etc.

4) Virtualized Operating Environment – Virtualized OE grid components again present the same essential interfaces and functionality as the OE components, whilst hiding their implementation. As with the virtualized physical devices, the implementations tend to be partitions or aggregations of the original platform component. Thus the virtualized operating environment is realized through technologies such as Solaris Containers or BSD Jails, which essentially partition in some way, a single instance of an operating environment and present each hosted grid component (service) with a customized environment which behaves as if it were a discrete operating system instance. Likewise IP load balancers and global IP addressing in clusters virtualize real IP addresses.

5) Platform Instance – The platform instance layer grid components are what one would think of as traditional single instance services or applications, such as an executable binary – e.g. a database server. This may resolve to multiple threads or processes but it may probably be best characterized as an application or service running as an executable on a single instance of an OS. Examples may include, but are not limited to database instances, application server instances, LDAP server instances, mail server instances and so forth.

6) Virtualized Platform – Virtualized platform grid components again present the same interface and functionality but may again be a partition of, or an aggregation of platform instance grid components. Examples include load balanced web server farms, application server clusters and database clusters.

7) Business Process (Service) – Business processes are complete services realized as aggregations of platform and virtualized platform grid components to fulfill a business function. Examples could be an online bookstore or perhaps an electronic banking service.

Note that any given realization need not include grid components in all layers, and indeed may include many components from within the same layer.

Also note that the various grid components within the physical and virtualized physical layers may also be broken down into components that are typically compute, network or storage focused. This is both potentially useful in that these categorizations are fairly familiar, but also fraught because as one goes up the stack, components may fit into multiple categories. Thus a traditional server (a compute element) may provide routing services or perhaps storage management services. This is represented by the fact that the silos become blurred in the diagram.

2.2.6 Grid Component Summary

The Grid Component concept provides a simple model of the various components within the modern data center, together with their relationships and dependencies on one another. Any

given component may be classified in a number of ways depending upon its context or use, yet those components share an essential set of properties and a life cycle.

2.3 The Grid Management Entity

2.3.1 Definition

Briefly recapping, the definition of an enterprise grid is

An ***enterprise grid*** is a collection of interconnected (networked) ***grid components*** under the control of a ***grid management entity***.

The ***Grid Management Entity*** (GME) is that ***logical*** entity that manages –

- The grid components.
- The relationships between various grid components.
- Their life cycles.

It is responsible for ensuring that the various grid components meet their goals, i.e. that the high level services and applications (e.g. ERP, BI, CRM etc.), deployed on the enterprise grid, meet their service level objectives within their operational constraints.

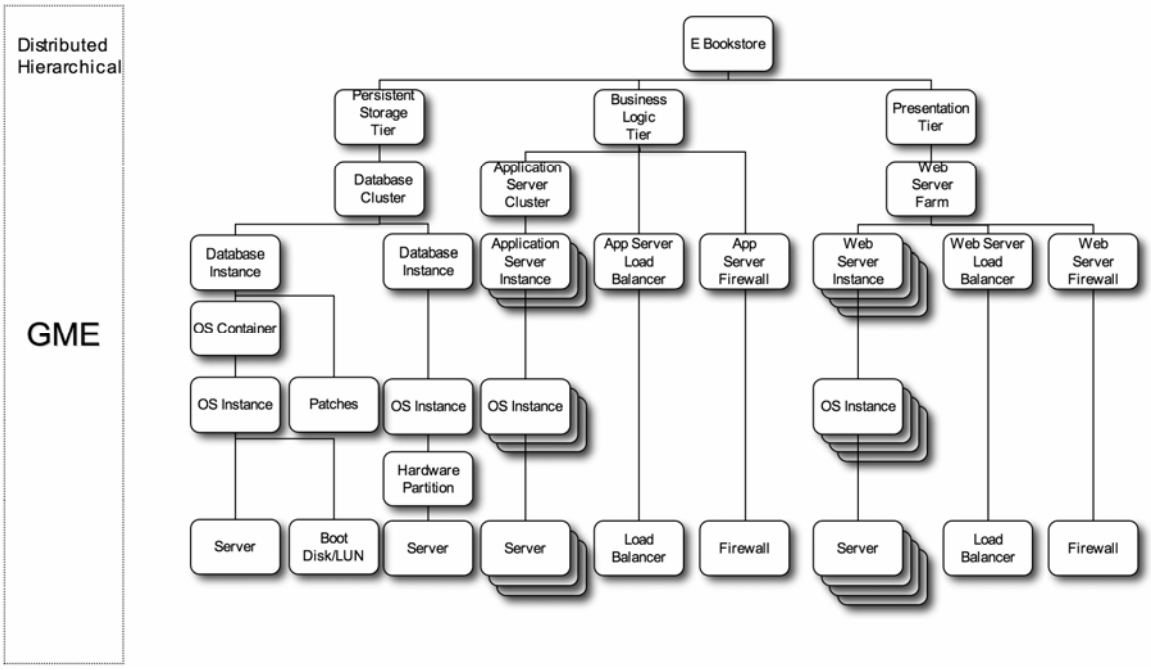

Figure ix - The GME As Logically Separate From The Grid Components

It is extremely important to note that the GME is a logical entity into which all management functionality is grouped. This is illustrated in Figure ix above, which shows the GME as being logically distinct from the set of managed grid components, in the form of an instantiation dependency graph, which it manages.

The GME may be realized by any combination of people and software tools. Today, in a typical enterprise data center, some of the GME functionality, especially in the areas of policy based

provisioning, management and monitoring, is realized in the form of processes undertaken by people. Through time the expectation is that most, if not all, of the processes will move into the technology domain in the form of automated tools. It may be realized by many pieces of software, coupled with people or process, or it could be realized in a single piece of software. The latter is somewhat unlikely given the nature of enterprise grids and the need to scale.

Whilst the GME is logically distinct from the managed grid components, the realization of GME functionality may not be as clearly separate from the realization of the grid components themselves. Thus a given grid component implementation could include management functionality, used to manage other grid components. This management functionality logically resides in the GME even though it is implemented in the grid component. This is illustrated in Figure x below. In this example, a portion of the dependency graph in Figure ix above has been expanded to show that the realized database cluster grid component contains GME functionality.

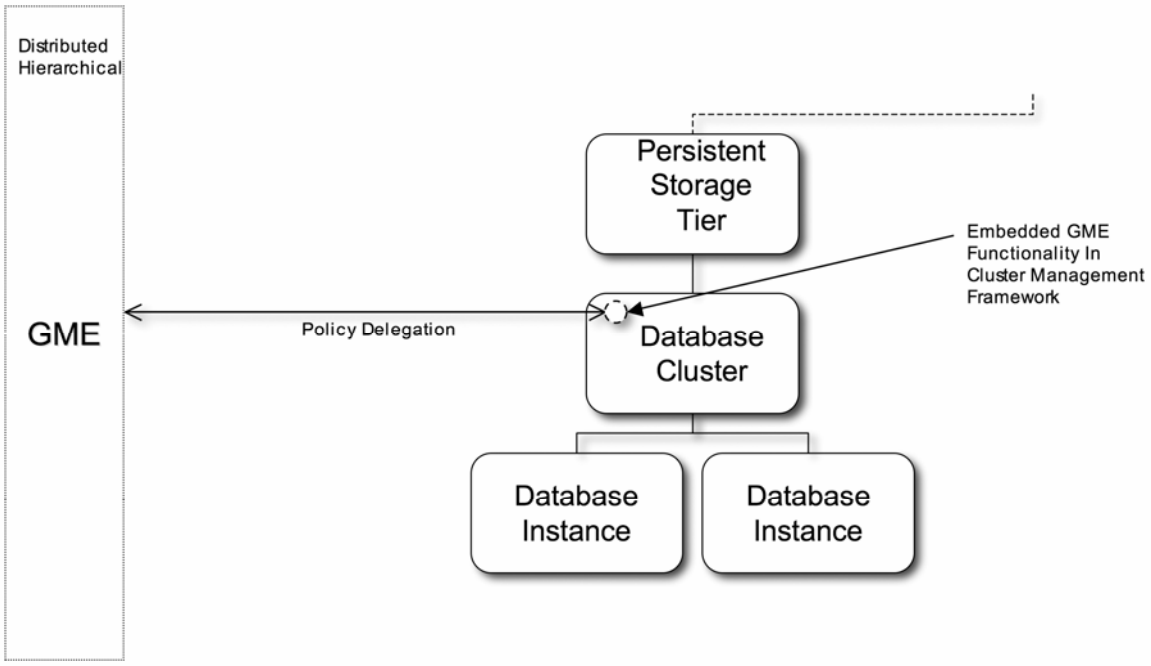

Figure x - GME Functionality Embedded In Grid Components

Thus the database cluster may be provided with SLOs and constraints with regards to managing the database(s) for which it is responsible. Once the GME has provisioned these database instances and made them available to the database cluster, the cluster may then be responsible for deriving the SLOs for each individual database instance and ensuring that they are met. The database cluster may then interact with the GME to provide updates with respect to progress against objectives or if it needs additional database instances provisioned on its behalf.

This is one of many possible examples showing that the GME is essentially hierarchical and distributed in nature. It also implies that the GME itself may be decomposed in the same way as applications and services, i.e. into grid components.

This example also illustrates the fact that many grid component realizations will include managed attributes/functionality and management functionality.

2.3.2 The Role Of The GME

Having described the general function and form of the GME, its role is now discussed.

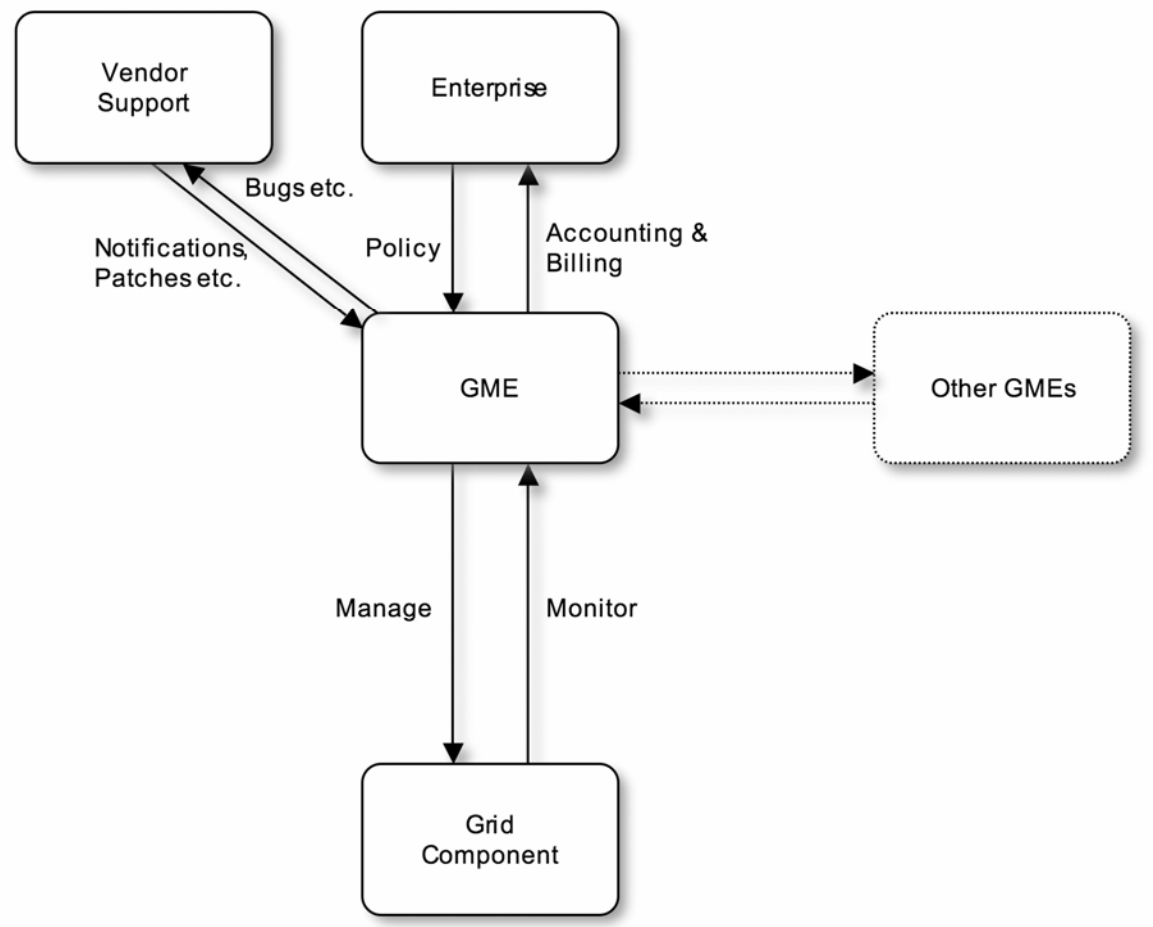

Figure xi - The Grid Management Entity

Figure xi illustrates the role of the GME and it's interaction with grid components. Whilst the figure shows the interaction with one grid component, the logical GME obviously interacts with all of the logical components within an enterprise grid. Its interaction with a specific grid component is determined by its interaction with the most abstract of grid components, such as a business service, and how that specific grid component contributes or enables that service. This may be deduced through the use of DAGs (see section 2.2.2).

Referring to Figure xi, the GME is responsible to the enterprise for the application of policy, for example service level objectives (SLOs) and constraints, governing the grid components under its control and for providing the enterprise with feedback, such as accounting and billing information. The higher the level of abstraction of the grid component actively managed, the more efficient the management of the enterprise grid is likely to be.

The GME implements these policies through the management and monitoring of grid components. Management and monitoring includes that of all of the macro phases of the component life cycle (including provisioning and decommissioning), together with more component specific microstates.

The GME also receives external input from entities such as vendors. Examples of this could include security alerts or perhaps patch update notifications, which may modify the policies or constraints applied to a given grid component.

Whilst the interaction of one GME and another is shown, this is only implied and is beyond the scope of this document at present. It is unclear whether the GMEs would constitute an aggregate GME that exhibits the same essential properties and behavior as a single GME, or whether inter GME boundaries exist, where the interactions differ from those within a GME.

The specific policies applied by the GME to a grid component will vary from one grid component to another. In some cases, for example with more abstract applications or services, the policies will be those explicitly specified by the enterprise and may include service level objectives, together with constraints. For simpler grid components, such as a server, some of the policies may be derived by the GME, based on its understanding of how that grid component is related to the more abstract grid components it enables. The role of policy is more fully described in section 0 below.

2.3.3 The Grid Component Life Cycle From The Perspective Of The GME

As already mentioned, the GME is responsible for the provisioning, monitoring, managing and decommissioning of all logically manageable grid components in order to meet the SLAs and SLOs of the business applications or services which they comprise or which depend upon them. This section explores the role of the GME in the life cycle of a grid component in more depth. The role is illustrated in Figure xii

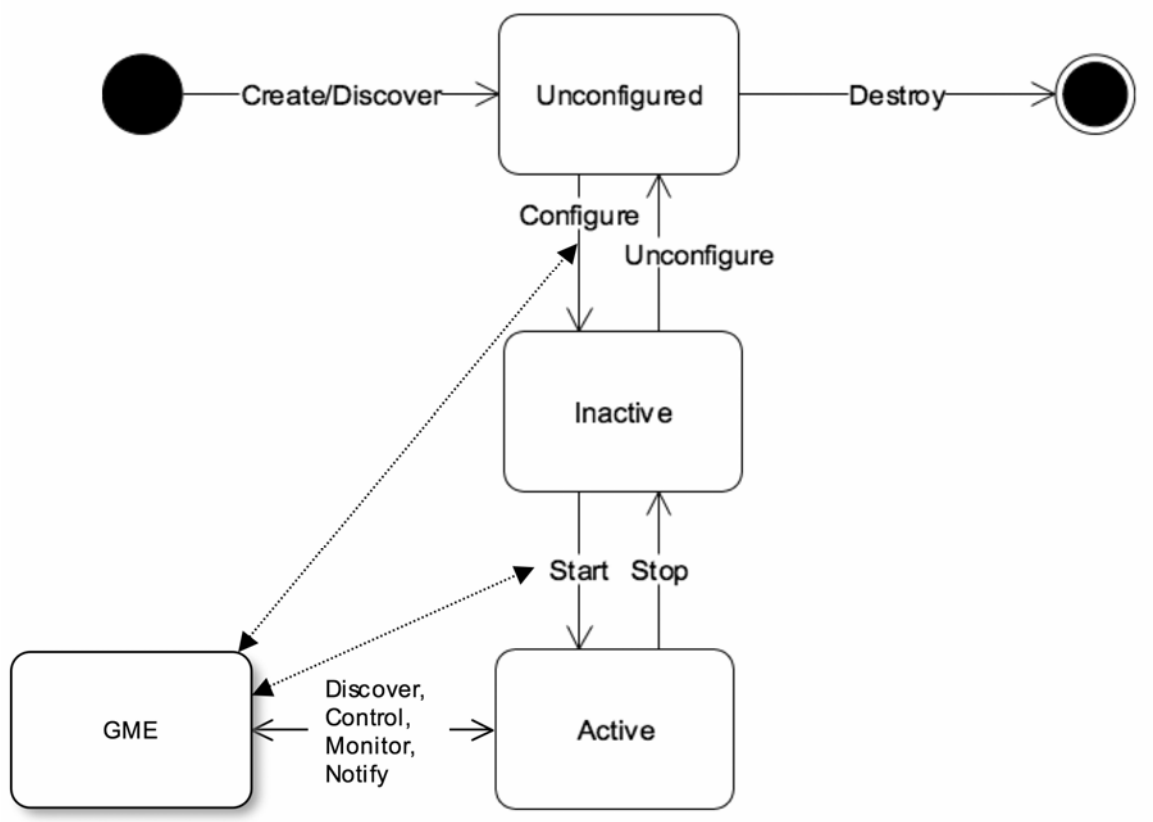

Figure xii - The GME And Grid Component States

The role of the GME includes –

- Setting and getting the various attributes or properties of the grid component, including but not limited to
  - Attributes or properties around performance, scaling, availability, security etc.
  - Dependencies or constraints for instantiation, performance, scaling, availability, security and so forth.
  - SLOs or SLO derived configuration parameters that enumerate the desired attributes of the grid component instance, that would allow the business process or service of which it is a part, to meets its SLOs. For more details please refer to sections 0 (Policy Based Management) and (Service Level Management)
  - Configuration information.
  - Data and metrics associated with its state and progress against SLOs.

  These attributes may be the result of direct operator input or may be derived by the GME via service characterization and sizing (see section 2.3.5.6).
- Managing the various state transitions. These may involve –
  - Setting or getting any of the attributes enumerated above.
  - Sending notifications.
  - Binding or associating a grid component with one or more grid components on which it depends – for example starting a database server on an operating system instance.
  - Stopping and unbinding a grid component from others upon which it depends.
- Monitoring the grid component once it is in the active state. This may include
  - Receiving notifications.
  - Getting any of the attributes enumerated above.

2.3.4 Orchestrating Grid Components

So far the GME discussion has really been limited to defining the GME within the context of enterprise grids and describing its interactions with individual grid components. On a component by component basis this is conceptually simple. However an individual grid component which is a service or business application is actually realized by orchestrating the set of grid components which comprise or are depended upon by it. Some sense of the complexity involved here is illustrated by Figure v - A Grid Component Dependency Graph, which shows just some of the instantiation (installation/provisioning) dependencies for a service or business application.

Thus the GME is responsible for orchestrating all of these components, their life cycles and their relationships with one another. It must manage the installation or creation of a service within the enterprise grid. It must manage that service so that it meets some set of business goals and constraints, and it must provide billing information and other information back to the owners of the service and/or the enterprise grid within which it is deployed.

The GME is analogous to an operating system in that it is responsible for managing the mapping and remapping of workload onto resources in line with goals. A traditional operating environment does this by managing a process and mapping it onto some set of processors based on scheduling priorities. In doing so it allows efficient resource sharing (multiple tasks can share the server), enables flexibility and simplifies management (you don't need to know which processors

a process is using, just how many and the specific processors used can be changed at any time). The GME's task is similar but somewhat more complex in that it must manage the mapping of complex business applications or services (grid components) onto a networked fabric of resources (enterprise grid) to meet a set of goals (SLOs, business constraints and so forth).

In order to successfully undertake this task, the GME needs a model and internal representation(s), and functionality that captures –

- The set of components to be managed - from servers, disks and switches to complete business services. The grid component provides a simple, generalized component model which may be extended into various classes of component as discussed in section 2.2.5 and illustrated in Figure viii - Some Grid Component Classes.
- The attributes and properties of the grid components, for example type, versions, capacity and so forth.
- The life cycle of these components. A generalized life cycle is presented in section 2.2.4 above.
- The relationships and dependencies between these components. These may be represented in a number of ways, one of which is through DAGs, see sections 2.2.2 and 2.2.4.2 above.
- Goals such as SLOs that determine the desired performance criteria for each grid component, from business applications to disks or servers.
- Policies that determine what management actions to undertake in order to meet the goals of the service, within the constraints of the enterprise gird, in response to various events or status changes.
- Status information indicating how well a grid component is performing against it's goals.
- A set of actions or verbs to be applied to the various grid components, to manipulate their life cycles as appropriate in order to meet their goals. These may include some of the following –
    - Create
    - Configure
    - Start
    - Update
    - Notify
    - Monitor
    - Control
    - Stop
    - Unconfigure
    - Destroy

The GME also requires a set of mechanisms that enable it to undertake each of the actions enumerated above.

All of the above are required so that the GME can manage the life cycle of a service or business application within an enterprise grid. In combination they enable the GME to use policy and heuristics to manage the life cycle of those applications and meet the goals and constraints of the enterprise(s) that own and host them.

2.3.5 Policy Based Management

2.3.5.1 General

As discussed in section 2.3.1, the GME essentially applies policies defined by the enterprise to the enterprise grid. The policies applied to the individual grid components may be those articulated by the enterprise or may be derived from them, depending on the nature of the grid component and how it is used. Today, most of the policies associated with managing an enterprise grid are applied by people. It is anticipated that in the future they will be applied by software tools.

These policies may be broken down into –

- Goals
- Constraints
- Configurations
- Rules

2.3.5.2 Goals - SLOs And SLAs

Goals are typically associated with more abstract grid components, such as a service or application instance, and express the desired operational attributes of that grid component.

Typically goals capture the SLOs of the grid component. For example, the grid component that is a business application, such as an electronic bookstore, could have the following set of SLOs –

- Average book purchase transaction response time of 0.5 seconds
- 80% of book searches return a response in under 2 seconds
- 200,000 concurrent logged in users
- 100,000 concurrent searches
- 30 minutes outage per month for security and software patching
- No unauthorized access to persistent storage or business logic tiers
- Only the bookstore manager may change the SLOs of the bookstore

SLOs may change through time or for pre-determined periods of time, for example the number of concurrent searches or logged on users may be increased if a popular new book is released or a sale is advertised.

In a more general sense, SLOs may also be broken down into the following, non-exhaustive list of categories –

- Response Time requirements
- Capacity requirements
- Volume or Throughput requirements
- Availability requirements – total availability (including both planned and unplanned outage) expressed as a percentage or as real time per month or year for example.
- Reliability requirements – some measure of the reliability, such as MTTF (mean time to failure)
- Recoverability requirements – some measure of recoverability, such as MTTR (mean time to recover) or desired post recovery state.

- Maintainability requirements – some measure of how maintainability, such as maximum required outages for upgrades etc.
- Safety requirements
- Security requirements

These are the typical business level SLOs that would form part of a Service Level Agreement (SLA) for that service. An SLA moves beyond the specification of some set of measurable parameters that define acceptable or desired performance to a legal document that may also articulate the rewards and penalties associated with success or failure in meeting those goals.

Less abstract grid components may also have SLOs. For example a database server that is a part of the bookstore may have a 0.1 second desired transaction response time for book searches.

Note that the role of the GME does not include the explicit interpretation of licenses and SLAs, which are out of the scope of this document. The GME may be provided with SLOs and constraints that may result from such SLAs or licenses, but it is the responsibility of the business that owns the enterprise grid to interpret the terms of any SLAs and licenses, and derive appropriate SLOs and constraints. Otherwise if the GME is responsible for such interpretation and the GME is realized through software then the software vendor may become liable for ensuring that SLAs and license terms are adhered to. It may be the case that this is desirable in certain circumstances. However, for the time being it is easier to simply declare SLA/license interpretation as out of bounds and that the enterprise is responsible for such interpretation and for configuring the GME and appropriate grid components so as to meet their SLA/license obligations.

2.3.5.3 Constraints

Constraints typically bound the goals of individual applications or services, and for the more abstract grid components may comprise some of the SLOs for the entire enterprise grid. For example constraints for a component could include its priority versus other grid components. Or perhaps define the maximum or minimum set of grid components (for example servers, or licensed instances of a database) that may be used in trying to achieve the SLOs of the service.

2.3.5.4 Configurations

Each grid component has configuration parameters, even if they are just derived from higher level SLOs rather than explicit SLOs themselves.

2.3.5.5 Rules

The GME also requires sets of rules so that it can handle events appropriately. These rules may prompt an automatic response or they may simply provide feedback to the enterprise or to an operator. For example rules are required to handle failure to meet an SLO, and which applications or services may be denied resources in order to improve the performance against goal of another application or service.

2.3.5.6 Applying And Deriving Policy

Ultimately the GME is responsible for ensuring that each grid component meets its goals and is appropriately configured within the operational constraints of the enterprise grid. When a business service is configured within an enterprise grid, the GME is made aware of its general composition (through a DAG or similar) and the SLOs and constraints that apply to it (license or other limits etc.). This is part of creating the grid component that is a business application.

Once the application has been defined or characterized within the GME, the GME then uses its understanding of the application and constraints to determine which other grid components to utilize for the application implementation, (based on their constraints/attributes setup when they were added to the grid by the GME) and to derive the SLOs (*derived component-level SLOs*) or configuration parameters for them, as well as perhaps the required number of instances. For example the derived SLO for the database component of the bookstore could include –

- 80% of bookstore search requests complete in under 0.15 seconds
- 100,000 concurrent searches

This process may be extended throughout the DAG for a business service, ultimately deriving the requirements for leaf node grid components in terms of processors, memory and network bandwidth for servers and so forth. These requirements are typically configurations rather than goals. Examples of how these derived values may be obtained is discussed in section 2.3.6.2 below.

Note that whilst all of this is done by the logical GME, this does not mean that it is centralized or that there exists a single global policy inference engine. This too may be distributed, with realized grid components that include GME functionality actually deriving some policies for those grid components it manages.

2.3.5.7 Policy Implementation

It should be noted that just as the general GME functionality is essentially distributed and hierarchical, so too is the policy decision making and enforcement. Using the previous example of a clustered database (see section 2.3.4). The cluster framework may be responsible for determining when a database instance has failed and for requesting that a new instance is started on another operating system instance under its control.

2.3.6 Service Level Management

2.3.6.1 General

The previous section was focused on policy (see section 2.3.5) and this section is focused on service level management (SLM), the thing that automated policy interpretation and application can enable. SLM is focused on automatically managing services and applications through high level SLOs and is key to enabling the efficient use of enterprise grids. It is SLM that is the process that drives the management of grid components to deliver business value.

Figure xiii expands on the role of the GME to emphasize its role in dynamically managing the life cycle and attributes of grid components as part of closed feedback loop SLM. The GME enables a grid component to deliver value by provisioning it or assigning/binding those grid components it depends upon to it (see also Figure xii). In the general or traditional sense the GME assigns resources (i.e. other grid components), which are then consumed (in some sense) by the grid component. Telemetry associated with the grid component usually allows metrics to be generated. These are then used as input by the GME, which reconciles them with its policies, as well as additional input (for example patch notifications from a vendor) before doing any necessary reassignment or re-provisioning.

NOTE in this section the term resource means any grid component on which the main grid component depends. This could range from an abstract entity such as a database to something physical like processors or a server.

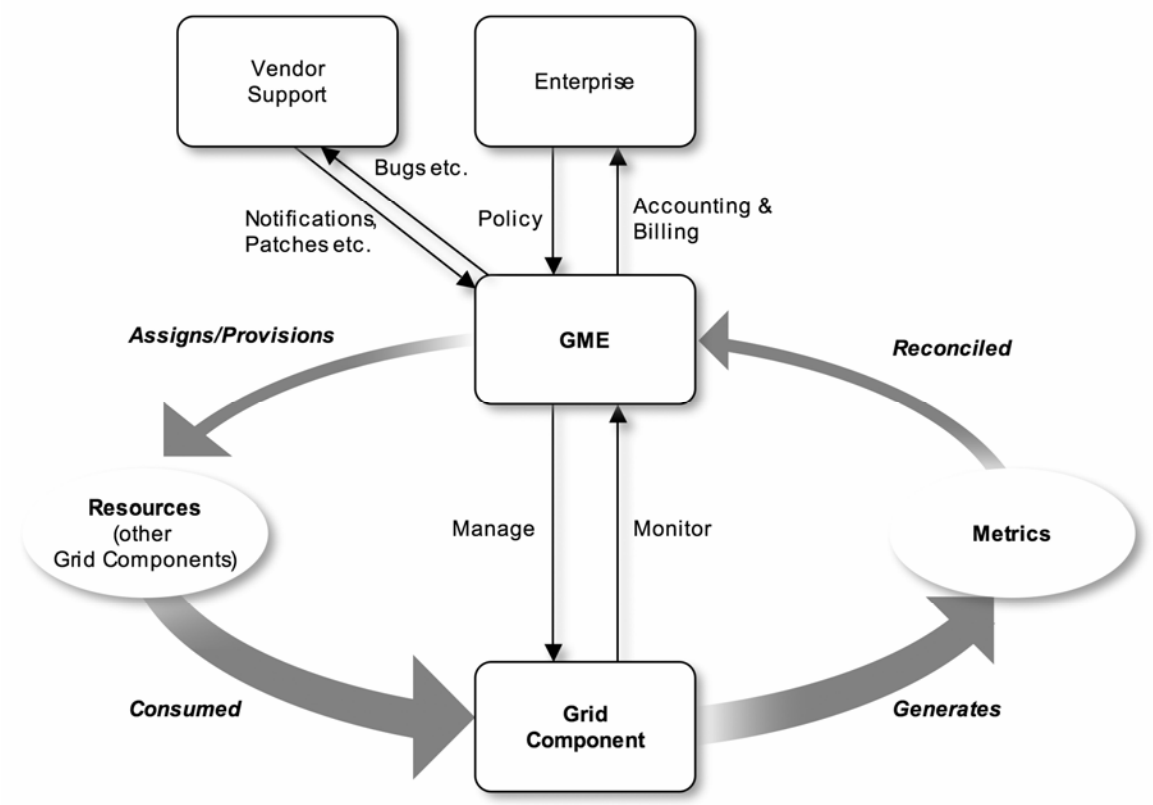

Figure xiii - The GME And Service Level Management

As discussed elsewhere, a grid component may be decomposed or viewed as a DAG. Thus the SLM cycle can be applied to every lower level grid component based on the composition of the more abstract grid components, such as an application or service, which it contributes to. In this case the policy input and billing/accounting output could actually be from/to the GME component that manages the next higher level of abstraction. At the highest level of abstraction, the provider of policy and consumer of billing and accounting information is the enterprise itself. This is illustrated in Figure xiv - Hierarchical SLM that shows some of the grid components in the top two levels of the DAG for the electronic bookstore. Note that the SLM component for the more abstract grid component may drive the provisioning of a less abstract component, i.e. one that is below it in the DAG. This diagram does not imply that the SLM functional component implements this provisioning. Merely that it controls or initiates it. It may be more appropriate to consider the provisioning to be undertaken by the GME component associated with the grid component to be provisioned.

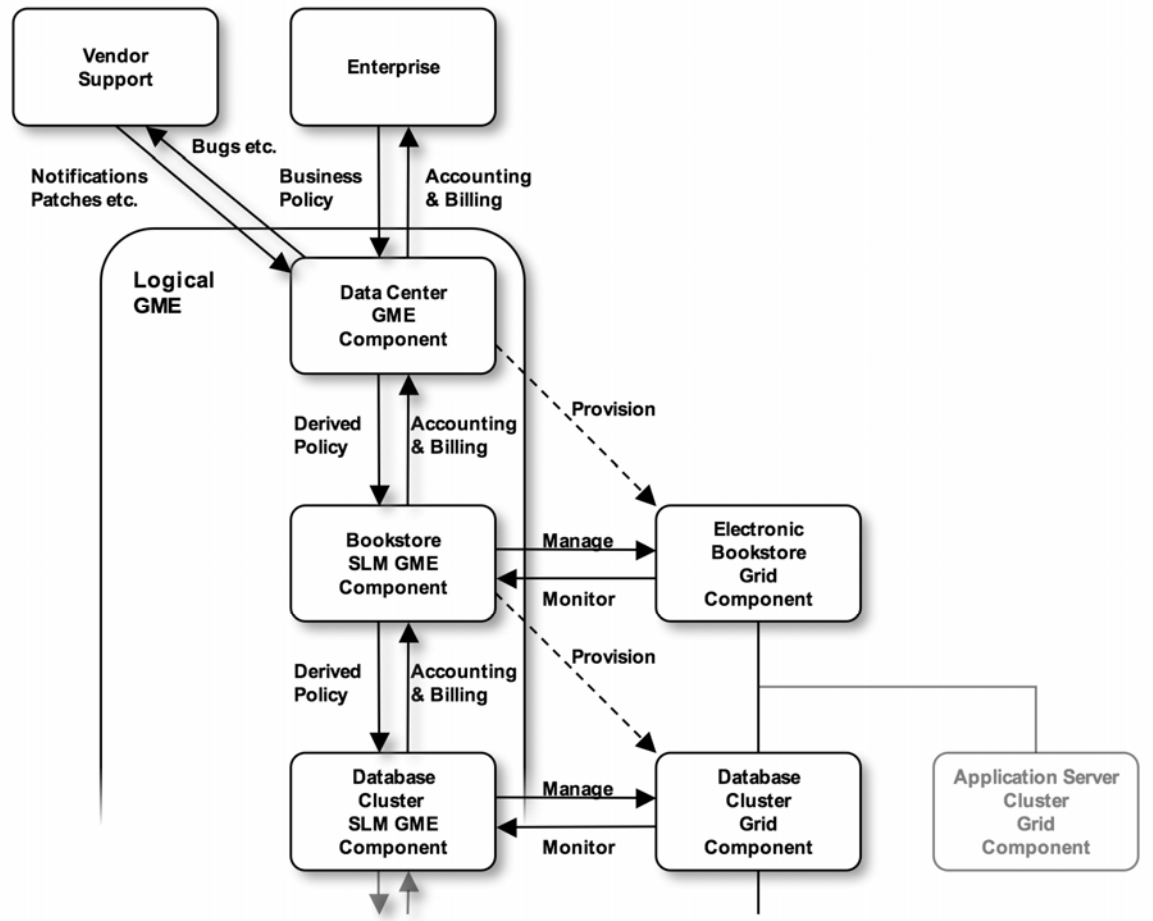

Figure xiv - Hierarchical SLM

As an illustration of the fact that whilst the GME functionality may be logically distinct, yet be realized in the implementation of a grid component, Figure xv below shows how a database cluster could incorporate both GME and non-GME functionality within a product.

The SLM process may be broken down into two phases –

1) Initial Deployment – where some initial policies and configuration parameters define the starting position.

2) Ongoing closed loop workload management and optimization.

These are discussed in the following sections.

Note that no assumptions are made about the characteristics of a given workload, nor what a unit of work is, with respect to the EGA reference model.

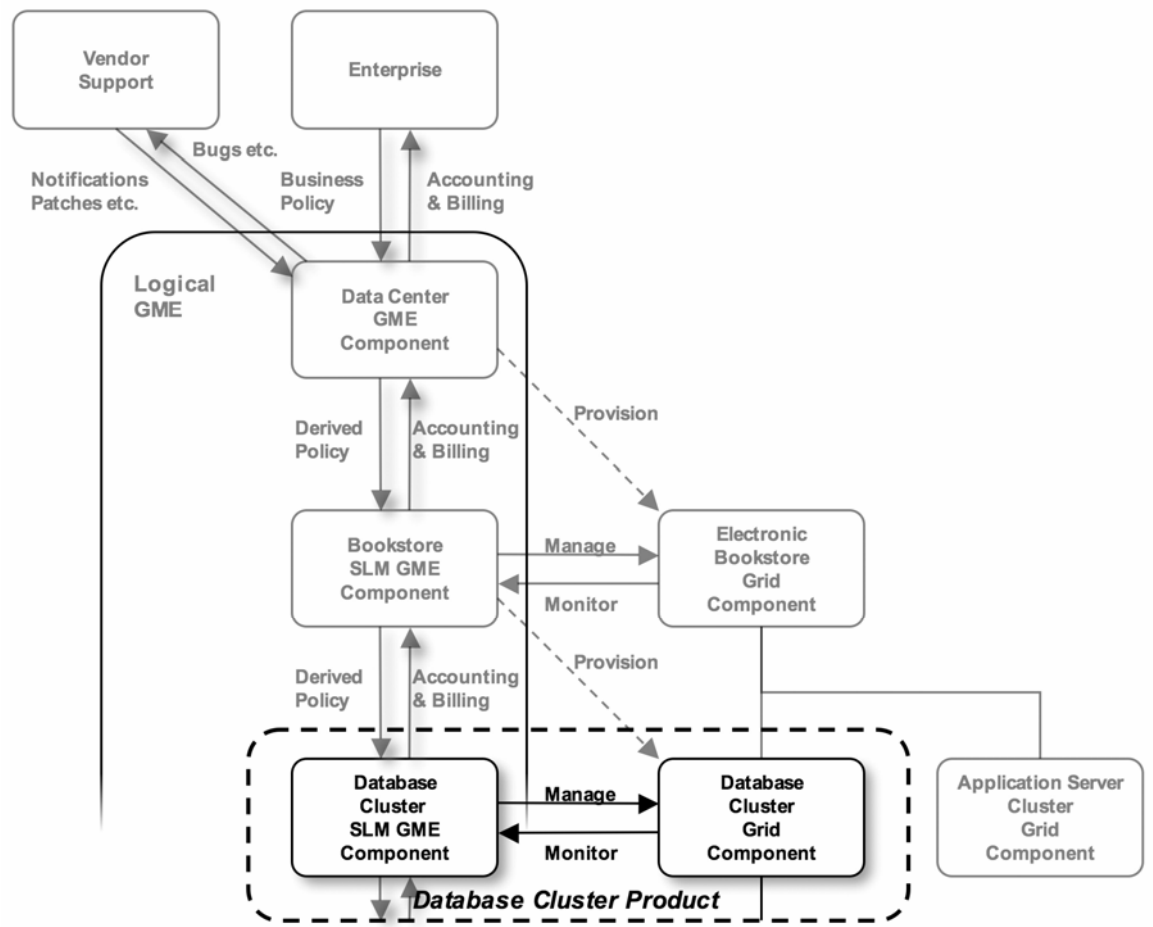

Figure xv - Hierarchical SLM showing GME functionality within a realized grid component - i.e. a database cluster

Workloads may range from the completely non-interactive batch type of workload (traditionally called a "job") to high frequency, short duration transactional workloads which are undertaken by a network of distributed components. Each is undertaken in the same way – a set of related grid components that undertake the work is captured within a DAG, deployed and then managed, prior to decommissioning. Obviously the DAG used to represent performance and scaling dependencies will vary greatly based on the type of workload, as will the rate and number of cycles of the SLM feedback loop when managing that workload. Service characterization and sizing (see below), as well as monitoring and management during the active phase of the life cycle of the workload may also vary greatly. Obviously the less cyclic or transactional and less interactive, the more important the initial characterization, as adjustments will be harder once the work is initiated. Nonetheless the process is common.

What may be called a “unit of work” will vary by grid component and use of that grid component. In many ways this will tie back to the SLOs, or configuration, of a given grid component, and ultimately to the high level SLOs of the service or application and value it delivers. For example at a high level, an electronic bookstore unit of work could be a particular transaction, perhaps a sale. That unit of work is undertaken by a collection of interacting, linked components. The derived unit of work for the database could be a stock lookup. The derived unit of work for the operating environment could be seconds of accrued processor time and megabyte seconds of storage and network bandwidth. Alternatively the unit of work of a weather simulation could range from the whole simulation to the execution of one of the non-interactive simulation elements. The point is that no standard unit of work for the general grid component is easily definable, even though in a grid, the ability to reconcile value delivered by a grid component with the cost and/or resources consumed in delivering it are key to both managing QoS and to enabling effective utility billing and accounting.

2.3.6.2 Initial Deployment

Before initial deployment of any grid component the GME must first derive it’s SLOs and configuration parameters. This may be achieved through -

- *Service Characterization* – the process by which the composition or pattern of a service and the relationships with its dependencies is derived, i.e. creating the DAG(s). Note that a number of different DAGs may be used in characterizing a service. Whilst all of the grid components may be the same, the vertices which join them may differ based on whether the DAG is being used to represent instantiation, performance, scaling, availability or security dependencies.
- *Service Sizing* – the process by which the number, performance, scaling, availability and security attributes of grid components is derived. This may include
    - Application specific benchmarks.
    - Capacity planning exercises
    - Data gathered from other similar installations

Today there is no satisfactory standard solution for automatically deriving the desired component level SLOs or configuration parameters for grid components. It remains an area where expert manual intervention or execution is required. Nonetheless guidance may be provided by vendors of applications or through the use of standard benchmarks that bear some relation to the target business application. Note that standard benchmarks, whilst providing a ready means of comparison for a very specific workload are often less than representative of actual usage patterns or scaling for many real deployments. Interestingly, the inherent flexibility of enterprise grids means that errors in initial sizing, especially for transactional workloads, can be corrected more rapidly – see section 2.3.6 on service level management below.

Once a business application or service has been characterized and sized with respect to performance, scaling, availability, security and so forth, and this has been input to the GME, then the GME may derive the policies to be applied to all of the grid components that comprise that service, provision them and manage them through their life until the service is decommissioned. This is facilitated by the internal model of the composition of the application as a set of grid components, for example in a DAG (as per Figure v - A Grid Component Dependency Graph).

This activity may include the decommissioning or re-provisioning of other grid components. For example if the set of available server grid components is limited and the new business application takes precedence over one already using some of those components, then the GME may elect to place some grid components (say a web server instance or two) into the inactive state so that the server and/or operating system instance grid component on which they were executing may be repurposed.

Once all of the necessary grid components are available, the business application grid component may be provisioned. This consists of the GME resolving all of the dependencies in the DAG and provisioning each grid component in turn, building on those "below" it until the whole service is provisioned. This is described in more detail in section 2.2.4.2.

2.3.6.3 Workload Management And Optimization

Once a grid component has been initially provisioned, then it is be managed by the GME so that it meets its SLOs, as illustrated in Figure xiii. Indeed the initial provisioning is equivalent to the resource allocation phase of the first cycle round the loop.

Monitoring

In its role as the service level manager, the GME monitors the provisioned grid component with respect to how it is progressing against its goals, in terms of performance, scaling, availability, security, efficiency and so forth.

Monitoring may be achieved through interacting and gathering monitoring statistics from the grid component itself, or through monitoring those that depend on it or upon which it depends. For example transactional response time may be deduced by querying the database grid component, or by querying the application server that sends queries to the database, or by sending simulated null transactions to the database. Monitoring availability may range from simply checking that the grid component exists to actively querying its state and responsiveness. Monitoring includes both polling and notification mechanisms. The EGA reference model currently makes no assumption about which methods are used, although it does assume both synchronous and asynchronous mechanisms.

It should also be noted that monitoring techniques will likewise vary by workload type, but that again the EGA reference model makes no assumptions with regards to this. For example a long running compute intensive workload or perhaps a batch component of an enterprise application may provide no obvious means of monitoring its progress, as progress is not defined through externally visible repetition, as it is with transactional workload, but by internal state changes and so forth.

Assumptions may be made, and the model modified as and when EGA working groups research these areas and choose to address them.

Reconciliation, Billing & Accounting

At regular intervals, which may vary by grid component and workload type, the GME will –

- Produce billing and accounting information to be fed back to the GME SLM components for the more abstract grid components that depend on it.
- Modify its policies and goals based on new business input, or requirements derived by the GME, for example if another, higher priority grid component is underperforming and requires additional resources. They may also be modified based on other external input, for example if a vendor notifies the GME that certain patches are required to improve security or performance perhaps.
- Reconcile the information gained from monitoring the active grid component, with its policies and goals and adjust accordingly. This is described more fully in the following section.

Resource Optimization And Balancing

The monitoring information gathered by the GME allows it to determine how well the grid component is performing against its various SLOs. If the grid component is failing to meet it's SLOs, the GME will need to determine whether to adjust the resources assigned to it. It may

choose to rebalance resources already at it's disposal, or it may request additional resources from other GME SLM components, for example above it in the DAG.

There are many possible methods for doing the rebalancing. All involve the donation of resources from grid components that are over performing, or are of lower priority, to underperforming grid components. Some simply adjust known parameters, one at a time, without seeking to determine the bottleneck, until improvement in the desired dimension(s) is observed. Others may allow the bottleneck to be automatically determined. Some may use trend analysis or other predictive techniques to determine how much resource needs to be reassigned. Trend analysis may be used to drive rebalancing on the fly, or it may be used to track long term cycles, such as quarterly billing or reporting for example. Some simply reassign small fixed quantities at regular intervals until the desired outcome is reached. The appropriate method will depend on many things, including the workload type. The EGA reference model does not assume any specific algorithm.

Reallocating

Once a decision has been made, resources may need rebalancing. This usually means that one or more grid components require reconfiguring or repurposing. Examples include, but are not limited to –

- Moving (decommission and provision) an existing web server grid component from one application to another.
- Provisioning (provision) a new application server instance.
- Changing the number of processors available to a database running on a shared operating system instance or server.
- Changing the number of processors available to a job running on a shared OS instance or server.
- Checkpointing a running job on one server and starting it on another.
- Changing the load-balancing algorithm in a load balancer.

All of these essentially involve the provisioning, decommissioning and reconfiguration of one or more grid components.

As an aside, reallocation is usually only carried out on a small number of grid components at any given time, s as to avoid potentially systemic instability.

Once reallocation is complete the cycle repeats.

2.3.7 GME Functional Breakdown

2.3.7.1 General

The previous sections have described, in depth, a number of the core areas of the functionality of the GME and the form of the GME. This section summarizes the functional breakdown of the GME.

Firstly the GME may be broken down into logical functional elements (GME components), each of which may, as per the previous section, be distributed within the enterprise grid and each of which is associated with the management of a grid component. This is illustrated in Figure xvi below.

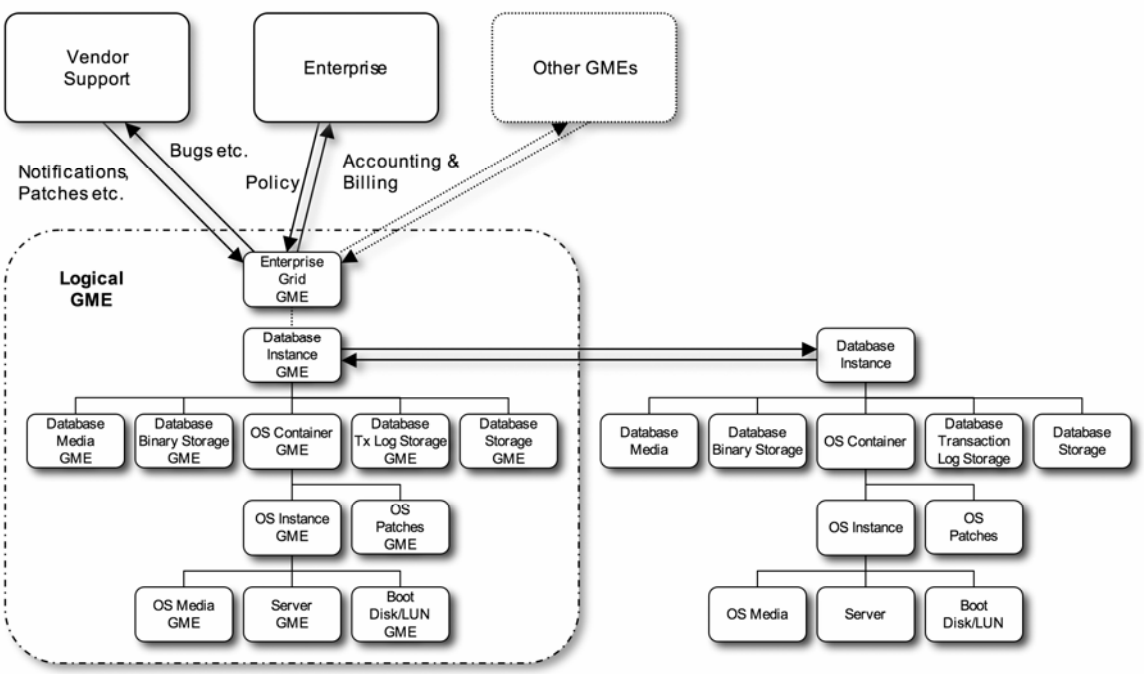

Figure xvi - Mapping of logical GME components within the GME to managed grid components

Note that there is a GME component for the enterprise grid as a whole.

Each GME component may be broken down into the following functional areas, each of which has already been described –

- Provisioning/Decommissioning – the act of putting a grid component into the active state. This may include –
    - Creation/Discovery
    - Configuration
    - Starting
    - Stopping
    - Unconfiguring
    - Descruction
- Management/Control – controlling an active grid component. This may include reconfiguring it, changing its state and so forth.
- Monitoring – this may include getting status through polling or receiving notification.
- Service Level Management - this may include management of
    - Performance
    - Scaling
    - Availability
    - Security

- Policy – including
    - Enforcement
    - Derivation
    - Billing And Accounting
    - Orchestration

This is illustrated in Figure xvii - GME Functional Breakdown. The break down is a logical one that can be applied to the logical GME as a whole or to the GME components. Realized GME software products may implement one or more of these functional areas and be associated with one or more realized grid components.

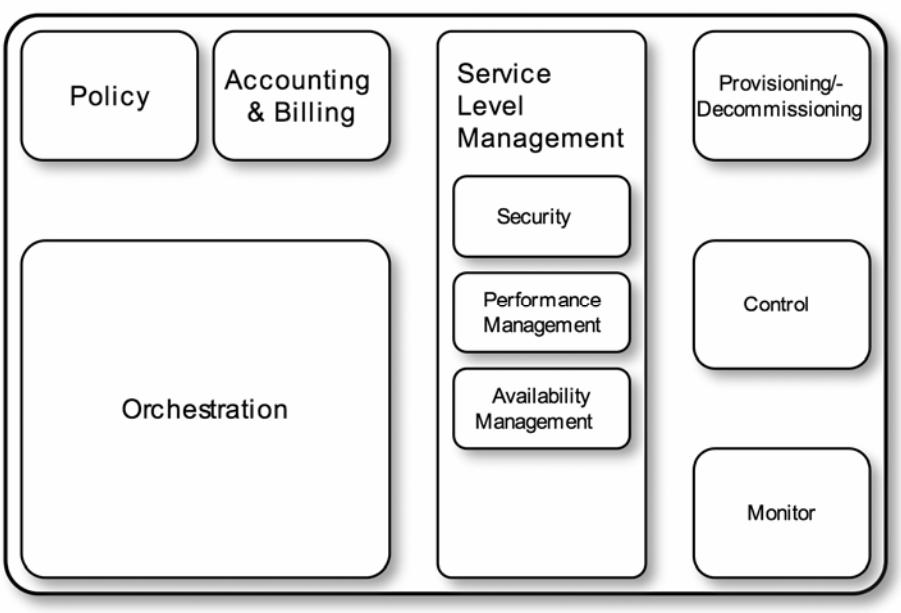

Figure xvii - GME Functional Breakdown

[NOTE add GME Repository to the diagram above.]

Figure xvii - GME Functional Breakdown, above, captures all of the components of the GME that undertake specific actions. In addition to these the GME also includes the notion of a repository. The GME Repository is the logical repository within which all information about grid components, including their location, connectivity and state, their relationships to each other (as expressed in DAGs), the sets of actions that may be applied to them and log of actions that have already been applied to them, exist. Like the GME itself, the GME repository may be implemented as a monolithic entity (data base) or it may be implemented as a federation of loosely coupled components (databases), distributed amongst any number of products.

NOTE – The GME Repository is analogous to the Configuration Management Database (CMDB) concept within OGC's ITIL.

2.3.8 GME Summary

The Grid Management Entity (GME) is a logical entity with which all management functionality in the enterprise grid is associated, whether implemented by people or software. That functionality may be realized in discrete components that only serve a management function, or may be embedded within the implementations of certain managed grid components. Thus, the GME is

essentially distributed and hierarchical in nature. This also implies that the GME is itself composed of grid components, i.e. GME components whose function is to manage the enterprise grid.

2.4 EGA Reference Model - Known Issues

This section is for known issues with the EGA reference model. A table will be added in future revisions as issues are raised and listed. Their resolution will be captured in both the known issues table and in the revision history table at the front of this document.

2.5 The EGA Reference Model And Other Standards/Models

2.5.1 General

The EGA reference model is intended to provide a high-level model that enables the EGA, its working groups and others to describe enterprise grids. The model is intended to be complementary to other extant models, which are typically either broader in scope, for example the Open Grid Services Architecture (OGSA)[3] from the Global Grid Forum (GGF)[4], or more component specific, for example some of the CIM[2] modeling work within the Distributed Management Task Force (DMTF)[1] or within the Storage Networking Industry Association (SNIA)[5].

The EGA reference model does not assume any particular implementation. Thus, there is no reason why the EGA model may not be realized by using essentially peer-to-peer web services as prescribed by OGSA or by more traditional means. The EGA model is as implementation neutral as possible so that it may be used to describe both existing and future implementations.

3 Generic Use Cases

3.1 General

This section provides a basic set of high-level use cases that provide the potential starting points and context for the work undertaken by the various EGA working groups.

The use cases are broken down into generic use cases where the focus is on managing the lifecycle of a non-specific grid component. The use cases associated with the provisioning and decommissioning of components is to be found in the EGA Reference Model Provisioning Use Cases document [1].

3.2 Generic Use Cases

The purpose of the generic use cases is to provide a basic context for all of the more specific use cases in various other EGA Reference Model use case documents. These are meant to cover the sets of shared operations and interactions common to all grid components.

Figure xviii - Manage Grid Component Life Cycle Use Case – below captures the highest level interaction between the actor, in this case the GME, and the grid component that exists within the enterprise grid. Note that that component could be any grid component, a server, a disk, an operating system instance, a database server or perhaps an application or service such as ERP.

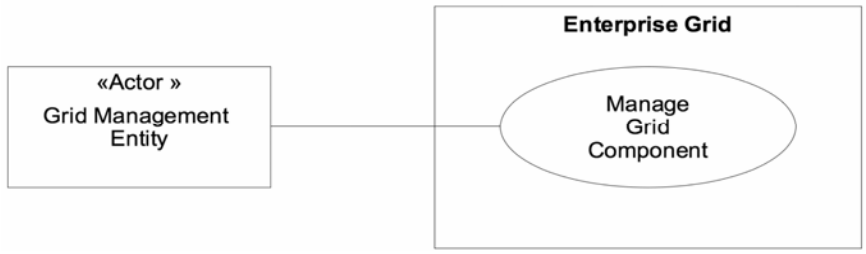

Figure xviii - Manage Grid Component Life Cycle Use Case

Indeed managing the life cycle of the enterprise grid is an aggregation of “manage grid component life cycle” use cases. This high-level use case may also be broken down into a set of sub use cases that map to the life cycle and state transitions of an enterprise grid component. This is illustrated in Figure xix - Manage Grid Component Sub Use Cases.

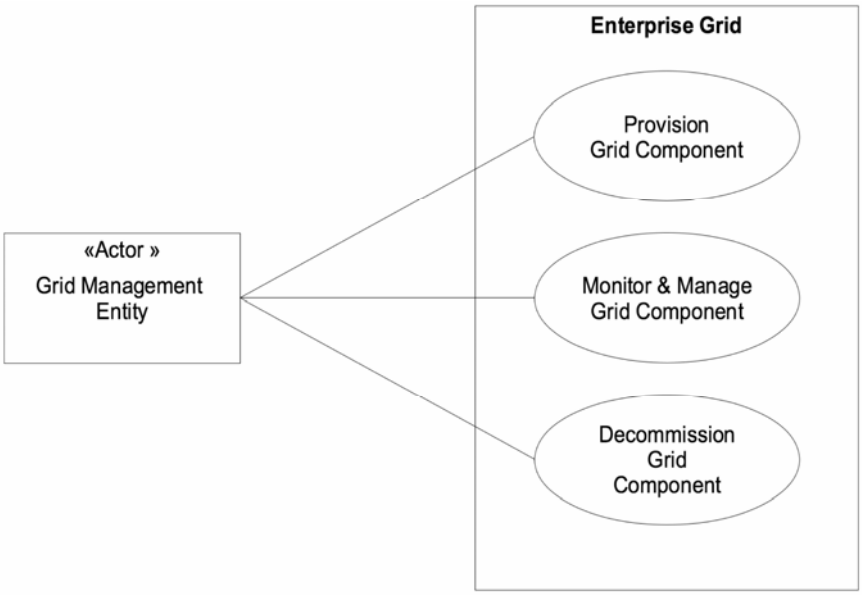

Figure xix - Manage Grid Component Sub Use Cases

3.3 Generic Use Case - Provision A Grid Component

3.3.1 General

This use captures the general process associated with the provisioning of a grid component.

3.3.2 Business Requirement(s)

To realize the benefits of grid computing, it is important to be able to provision and decommission grid components, as business needs change. This use case deals with provisioning a grid component within an enterprise grid.

3.3.3 Actor(s)/Stakeholder(s)

- GME – including people, process and technology.

3.3.4 Reference(s)

None.

3.3.5 Brief Description

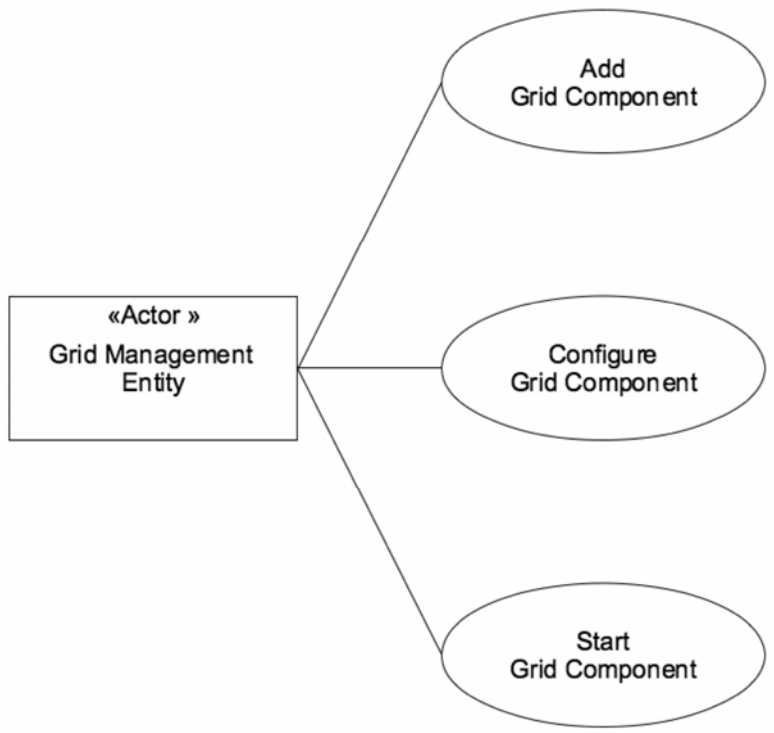

Figure xx - Provision A Grid Component Sub Use Cases

As shown in Figure xx - Provision A Grid Component Sub Use Case - the provisioning process may be broken down into 3 phases, which could be considered sub use cases –

- Add/Create Grid Component – which captures the creation of a new grid component. This activity could be one of –
  - Discovering a grid component and making it manageable by the GME. Examples include but are not limited to putting a server in a data center, thus making it physically manageable, or switching on a device which advertises its presence to the network, for example via Jini.
  - Creating a new instance of a grid component from an existing logically manageable grid component – for example creating an operating system instance grid component (not necessarily bootable, just a separate and distinct managed logical entity) from a copy of the distribution media for that operating system (which is itself a grid component).

  The act of creation/discovery also includes the GME being made aware of the nature of the new grid component including –
  - The set of supported interactions between the GME and the grid component.
  - Its attributes and properties.
  - Inherent constraints and dependencies – i.e. those that are not instance specific. For example an OS version created from existing media, and which was compiled to run on a specific platform such as x86.
- Configure Grid Component – which captures the act of configuring a given component so that it may be activated. Configuration may include, but need not be limited to
  - Defining dependencies or constraints on the grid component.

  - Setting internal configuration parameters that enable the component to be bound to other components to form a service. An example of this could be specifying an instance ID for a database instance.
  - Setting internal configuration parameters that statically bind the component to other grid components on which it specifically depends. Examples of this could be specifying the node(s) or node types on which an application server instance executes.
  - Specifying configuration or policy parameters which define the SLOs, and thus the desired performance characteristics of the grid component, for example desired latency for file operations, or desired number of concurrent connections.
- Activate Grid Component – which describes the act of putting the component in the active state so that it can be used.

Notes –

- Any given implementation will include these phases, though they may not be distinct in some, or they may be decomposed into distinct sub phases in others. For example some parts of the configure phase may actually be carried out during activation in the form of late binding.
- This use case does not deal with the phases as separate use cases at present.
- Provisioning is the act of putting the grid component into the active state. It does not assume anything about the existence or current state of a grid component, i.e. the grid component to be provisioned may already exist and may be being repurposed.

3.3.6 Pre-Conditions

- The grid component is manageable by the GME. For a physical component (e.g. server, drive array, network switch/router, software distribution media), this means that it must be at the enterprise grid's physical location. For all other grid components, it means that they must be logically manageable by the GME.
- All SLOs and constraints for the grid component have been defined. Depending on the particular type of grid component this may range from simple parameters such as the number of processors to be used by an application, to the high level business SLOs (e.g. average transaction response time) which map directly to the SLA for the grid component which is a complete business application. It is the role of the GME to derive component goals from those of aggregated sets of components or more abstract components such as business applications or services.
- Use of the grid component and its intended deployment has been appropriately licensed or will be appropriately licensed in parallel to the provisioning process.
  Note – it may be appropriate for future iterations of this document to more clearly capture the licensing process, as this is a key, long-term enabler for utility computing environments and may be closely bound to the utility accounting process.

3.3.7 Initiation Of Use Case

The GME determines that an additional grid component is required in order for one or more of the services hosted within the enterprise grid to meets its SLOs and thus satisfy its SLA.

3.3.8 Flow Of Events (Basic, Alternative And Exceptional)

Basic Flow

1) If the grid component depends on other grid components, i.e. it is not a leaf node in the provisioning dependencies DAG (see in Figure v - A Grid Component Dependency Graph above), then

   Provision each prerequisite grid component necessary to support the grid component to be provisioned. Determining the set of prerequisites may be achieved through the use of a provisioning dependencies DAG. Note that the grid components depended upon may need to be in a specific sub-state of active before the main grid component may be activated. Thus additional update configuration use cases may be called.

2) Create/discover the grid component

3) Configure the grid component

4) Activate the grid component

Alternative Flows

If the necessary grid component has already been created, then it may need configuration or reconfiguration, which may or may not require it to be stopped and reactivated. This may occur when a grid component is to be shared or repurposed.

Exceptional Flows

Exceptions would be generated if the appropriate grid components are not available, or any of the steps generate errors.

3.3.9 Success Criteria

The specified grid component is in the active state.

3.3.10 Post Conditions

The grid component is active.

3.4 Generic Use Case - Monitor And Manage A Grid Component

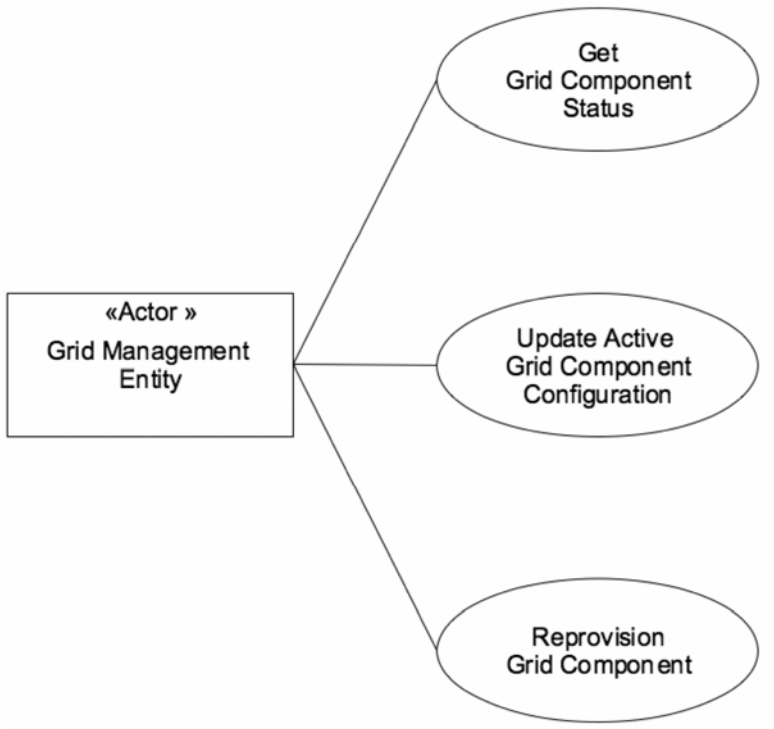

Figure xxi - Manage And Monitor A Grid Component Sub Use Cases

3.4.1 General

This set of use cases has yet to be developed.

3.5 Generic Use Case - Decommission A Grid Component

3.5.1 General

This use captures the general process associated with the decommissioning of a grid component.

3.5.2 Business Requirement(s)

To realize the benefits of grid computing, it is important to be able to provision and decommission grid components, as business needs change. This use case deals with decommissioning a grid component within an enterprise grid. Examples could include retiring a service and/or removing obsolete grid components.

3.5.3 Actor(s)/Stakeholder(s)

- The GME – including people, process and technology.

3.5.4 Brief Description

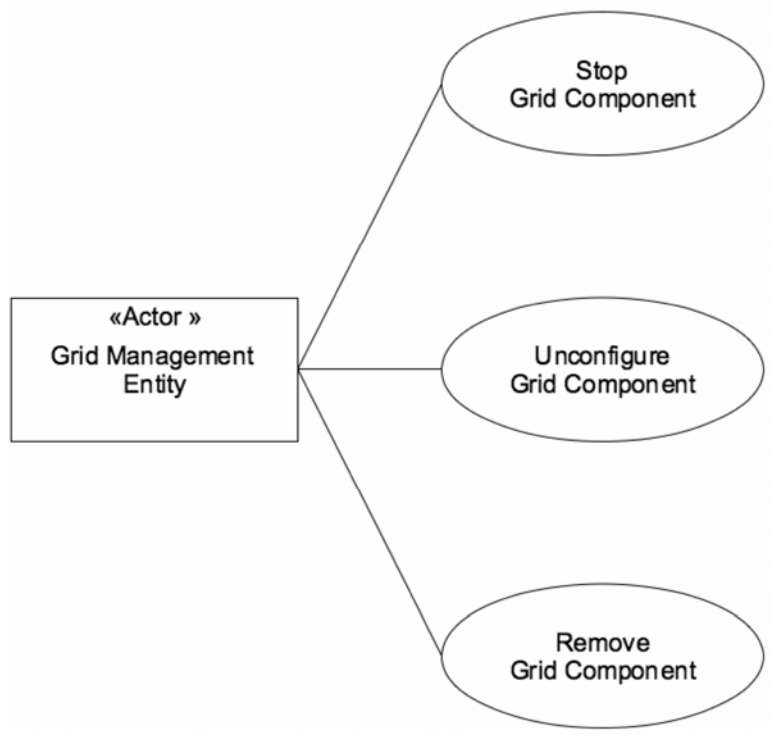

Figure xxii - Decommission A Grid Component Sub Use Cases

As shown in Figure xxii - Decommission A Grid Component Sub Use Cases - the decommissioning process may be broken down into 3 sub use cases –

- Deactivate Grid Component – which describes the act of removing the component from the active state so that it can be reconfigured or un-configured.
- Un-configure Grid Component – which captures the act of un-configuring a given component so that it may be reconfigured as part of the reconfigure grid component use case or in anticipation of being destroyed.
- Destroy Grid Component – which captures the destruction of a grid component

Note that any given implementation will include these phases, though they may not be distinct in some, or they may be decomposed into distinct sub phases in others. For example the un-configure phase may actually be carried out during destruction or reconfiguration.

3.5.5 Pre-Conditions

- The grid component is in one of the following states – Unconfigured, Inactive, Active.

3.5.6 Initiation Of Use Case

The GME determines that a grid component is no longer required in order for one or more of the services hosted within the enterprise grid to meets its SLOs and thus satisfy its SLA, and that that component should be decommissioned.

3.5.7 Flow Of Events (Basic, Alternative And Exceptional)

Basic Flow

1) Stop the grid component

   If the grid component depends on other grid components, i.e. it is not a leaf node in the provisioning dependencies DAG (see in Figure v - A Grid Component Dependency Graph above), then

   Decommission each prerequisite grid component that supports the grid component to be decommissioned. Determining the set of prerequisites may be achieved through the use of a provisioning dependencies DAG.

2) Un-configure the grid component

3) Destroy the grid component

Alternative Flows

The GME determines that a grid component upon which the target grid component depends cannot be decommissioned. For example, it may be shared by multiple services and/or dependent grid components. In this case the GME either ignores that grid component or reconfigures it.

Exceptional Flows

Exceptions would be generated if the appropriate grid components are not available, or any of the steps generate errors.

3.5.8 Success Criteria

All non-shared grid components are decommissioned.

3.5.9 Post Conditions

Unused grid components are available for use by other services.

4 Glossary Of Terms

Term	Definition
Abstraction	Abstraction involves changing the interface of an object and exposing it in a more useful or appropriate form. This may be done by providing a layer of software, or whatever, which makes an object or collection of objects easier to manage by exposing a different, abstracted object with a different interface. An example of this would be presenting the user with a tier of a service, for example a tier of web servers and their associated load balancer. The user would then manage the whole tier as a single object, including web tier attributes and load balancing policies, rather than managing each individual component. See also: *Virtualization*
Acyclic Digraph	See *Directed Acyclic Graph*
Commercial Enterprise Application	A class of *enterprise application* which usually exhibits some of the following attributes - • Multi-tier architecture • Interactive, as well as batch components. • Mostly packaged, often with some customization. • Centrally managed to enforced standards and policies. • Operational management to meet defined SLAs. Examples include ERP, CRM etc. With respect to *enterprise grids*, they are typically - • Hard to change due to architectural inertia and large installed bases. • Not yet deployed in enterprise grid contexts, i.e. they typically have dedicated silo-ed resources such as servers and storage. • Proprietary and not yet focused on standards support or interoperability. See also: *Enterprise Application, Enterprise Grid, Service Level Agreement(SLA)*
CIM	*Common Information Model*.
Compute Element	A physical *grid component* that is a general purpose computer, such as a server. See also: *Grid Component*
DAG	See *Directed Acyclic Graph*
Directed Acyclic Graph (DAG)	A directed graph containing no directed cycles, i.e. closed directed loops. See also: http://mathworld.wolfram.com/AcyclicDigraph.html

Term	Definition
Decommissioning	*Decommissioning* is the act of putting a grid component into the destroyed state so that it is no longer available to the GME. See also: *Decommissioning, Grid Component, Grid Management Entity*
DMTF	*Distributed Management Task Force.*
Enterprise Applications	The applications or *services* that are run within modern enterprise data centers. These often exhibit some or all of the following attributes: • Most run within a single datacenter, there being exceptions for availability, load balancing and co-operative processing. • Most run within an Enterprise's intranet, exceptions include message-passing applications, for example stock trading applications.
Enterprise Computing	Computing to support the operation of an enterprise.
Enterprise Grid	An *enterprise grid* is a collection of interconnected (networked) *grid components* under the control of a *grid management entity*. An enterprise grid is typically managed by a single enterprise - i.e. a business entity is responsible for managing a pool of resources, and a set of services, and for managing the assignment of resources to services in order to meets its business goals. The resources and services may or may not be owned by the business, for example in the case of managed services or a service provider/outsourcer. What defines the boundaries of the enterprise grid is management responsibility and control. The services that run on an enterprise grid may range from the traditional *commercial enterprise applications*, such as ERP or CRM packaged applications, to newer, distributed applications or services. Enterprise grids are typically differentiated from more traditional data centers by management practices and technology which – • Make management service or application centric, rather than component centric. • Enable the pooling and sharing of networked resources. • Enable agility through rapid and automated service provisioning. An enterprise grid may be confined to a single data center or it may extend across several. For more information see [NOTE - insert section with longer description] See also: *Grid Component, Grid Management Entity, Provisioning, Resource, Service*
Enterprise Grid Computing	A form of *grid computing*, specifically within an enterprise, resulting in an *enterprise grid* that includes *commercial enterprise applications* as a type of workload/service. See also: *Commercial Enterprise Applications, Enterprise Grid, Grid Computing*
GGF	*Global Grid Forum*.

Term	Definition
GME Component	A super-class of object that represents the management functionality within the logical GME that is associated with a grid component. See also: *Grid Component, Grid Management Entity*
GME Repository	The GME Repository is the logical component of the GME in which all of the configuration, state, audit and event logging information for an enterprise grid is stored. The GME repository is a logical entity and may be realized in number of ways, including but not limited to – ▪ A single, discrete database ▪ A federation of discrete databases ▪ A federation of databases and flat files, some discrete and some internal to specific GME component or grid component implementations. The GME repository is directly analogous to the CMDB in ITIL.

Term	Definition
Grid Component	The *grid component* is a super class of object that represents every managed entity, whether physical or logical, within an enterprise grid. Grid components are recursive, in that they may be aggregated to form other grid components, or decomposed into their constituent grid components. Examples of grid components include, but are not necessarily limited to – • Traditional resources such as servers, switches, disks, arrays and routers. These are considered to be elemental grid components. • Less obvious resources such as a software distribution or perhaps data. • Services, such as CRM, ERP etc. • Everything in between. Thus, for example, an ERP service may be decomposed into database, application server and web tiers. These may in turn be decomposed into instances that are bound to operating system instances and servers. Each of these is considered to be a grid component. Whether a grid component is viewed as a *service* or as a *resource* and depends on context. Each grid component has a life cycle which has to be managed and may include a number of phases, which may include – • Add • Configure • Start • Stop • Unconfigure • Remove See section [Insert here] for a fuller description of the grid component and its life cycle, as well as classifications of various types of grid component. See also: *Enterprise Grid, Resource, Service*

Term	Definition
Grid Computing	The use of pools of resources onto which applications or services may be dynamically provisioned and re-provisioned to meet the goals of one or more enterprises, whilst improving both efficiency and agility. Grid computing environments may be typified by • the use network distributed, shared pools of discrete resources to achieve greater performance, scaling, resilience and utilization • flexibility or mutability, as components are regularly repurposed or re-provisioned in line with the business goals for the services that run on them • a focus on services, rather than on components, as grids turn networks into arbitrarily rich/complex fabrics of resources. • application or service architectures which are disaggregated or distributed in nature, for example Service Oriented Architectures (SOAs) and which leverage the properties of the fabric of resources. • the consolidation of computing components into [typically] a smaller number of larger resource pools to enable easier provisioning and greater resource utilization. • the standardization of components and/or their interfaces, configurations, processes and applications. Virtualization is an example of this – see section 2. Grid computing is subtly different from Utility Computing, which includes most, possibly all of the attributes of grid computing, but implies a pricing model associated with the usage of a grid computing environment, together with enabling concepts and technologies such as standard units of work/value and metering. See also: *Utility Computing*
Grid Management Entity (GME)	The *grid management entity (GME)* manages all of the *grid components* within an *enterprise grid*, including – • their relationships with one another, for example mapping operating systems and applications onto servers and networks • their life cycle, for example discovering, configuring, starting and stopping components. The GME may be realized – • Through people and process • Through technology • A combination of the above. The GME provides the context for understanding the key problems associated with managing enterprise grids and grid components today. See also: *Enterprise Grid, Grid Component*
Horizontal Scaling	See Scale Out.

Term	Definition
ITIL	The Information Technology Infrastructure Library. ITIL is a cohesive set of best practices around service and infrastructure management developed and published by the OGC. ITIL has been widely adopted by both government and commercial organizations as the basis for their internal data center and IT management processes. See: *Section* ***Error! Reference source not found., Error! Reference source not found.*** See also: *OGC*
OGC	The Office of Government Commerce. The OGC is an independent office of the treasury of the UK government, and is the body responsible for the development and publication of the ITIL best practices. See also: *ITIL*
OGSA	*Open Grid Services Architecture*.
Network Element	A physical *grid component* that is a networking device, such as a switch or router. See also: *Grid Component*
Pool	A pool is a set of grid components. Membership of a pool is dictated by membership criteria, which may range from explicit assignment to the pool by an operator, to implied membership such as sharing a common state. The pool may be persistent or non persistent. For example a non-persistent pool could be one that exists momentarily during a decision making process when some set of components that are candidates for some operation needs to be determined. Membership of a pool could make a member a subject of an aggregate verb (one which when applied to the pool is then applied to each member) perhaps or a candidate for a member specific verb. Of course there may be pool specific properties and verbs too. See also: *Grid Component*
Provisioning	*Provisioning* is the act of putting a grid component into the active state so that it is available to a consumer of the service or resource it supplies. No assumptions are made as to the starting state of the grid component. See also: *Decommissioning, Grid Component*
Resource	*Resources,* in the most general sense, are things, usually physical or logical components, onto which services are provisioned. Resources are considered to be *grid components* within the context of the EGA reference model. Examples include server hardware, network switches, disc arrays, software media and so forth. See also: *Grid Component*
Scale Out	*Scale Out* is the term usually applied to scaling an application or service through the use of multiple service component instances, which typically resolves to additional operating system instances and/or servers too (plus clustering frameworks of various forms). This is synonymous with *Horizontal Scaling*. A typical example of a service that scales out is a web server tier of a multi-tier service. See also: *Horizontal Scaling*

Term	Definition
Scale Up	*Scale Up* is usually applied to scaling an application or service by increasing service performance and/or capacity through making more resources available to an instance of a service or service component, typically within a single instance of an operating environment and/or server. This is synonymous with Vertical Scaling. See also: *Vertical Scaling*
Service	A service in the most general sense is something, usually one or more software components, which responds to client requests. An electronic bookstore application is a service. The database component of the bookstore provides a data base service to the bookstore. Thus high level, abstract services may be recursively decomposed into lower level constituent services. In general, a service and all off its decomposable sub services are considered to be *grid components* within the context of the EGA reference model. In general the term should be qualified or its meaning made obvious through context. See also: *Grid Component*
Service Characterization	The process by which the composition or pattern of a service and the relationships with its dependencies is derived. See also: section 2.3.6.
Service Level Agreement (SLA)	An agreement between the provider and consumer of a service which stipulates a set of properties or attributes that the service must satisfy, possibly together with a definition of the payment and/or penalties associated with meeting or failing to meet the agreed criteria. See also: *Service Level Objective*
Service Level Objective (SLO)	The specific quantifiable and measurable attributes of a service that form the basis of a Service Level Agreement. See also: *Service Level Agreement*
Service Processor	A component of a physical server which runs independently of that server's operating system and main processors, memory and IO, and which allows the physical platform to be managed independently of whether the server is booted or not, or has an operating system installed or not.
Service Sizing	The process by which the number, performance, scaling, availability and security attributes of grid components is derived. See also: section 2.3.6.
Storage Element	A physical *grid component* that is a storage component, such as a disk or array. See also: *Grid Component*

Term	Definition
Technical Enterprise Applications	A class of *enterprise application* which usually exhibits some of the following attributes - • Often highly parallel • Heavily batch oriented. • Mostly heavily customized. Examples include Rendering, Portfolio Simulation, Finite Element Modeling etc. With respect to Enterprise Grids, they are typically - • Easier to change due to inherent customization. • Already deployed in some Enterprise Grid contexts. • Often driven by cooperative processing and some focus on standards support or interoperability. See also: *Enterprise Application*
Utility Computing	Utility Computing is a term often applied to IT infrastructure and technology which really captures the fact that that infrastructure may be paid for or may deliver services which may be paid for based on use or value rather than on component cost. The analogy being with a typical electrical grid. It is implied that there is a standard set of billable units and appropriate metering capabilities. Grids, by their nature, will be service centric and will require telemetry that will enable reconciliation of resources with the services that consume them, i.e. cost with value, and thus to some degree they enable utility computing. Nonetheless there is a subtle distinction between a compute utility and a grid, typically in terms of the use of metering, billing, rating and so forth. See also: *Enterprise Grid Computing, Grid Computing*
Vertical Scaling	See Scale Up.
Virtual Machine Monitor/Manager (VMM)	A layer of software or firmware/microcode that sits between one or more operating systems and a server. It presents each hosted operating environment with the impression that it has it's own dedicated server, although in reality they all share the same physical server. See also: *Virtualization*

Term	Definition
Virtualization	Adding a software abstraction layer onto some entity so that the new entity exhibits the interface properties of the original or a normalized interface broadly similar to that of the original. This layer hides the true implementation of the virtualized entity so that the original can be changed or replaced without fundamentally impacting the interaction of entities that have a dependency on it. The abstraction layer may allow a single entity to be partitioned and presented as a set of virtual entities, or it may allow a set of discrete entities to be treated as a single aggregate entity. An example is disk LUNs, which present the interface of a disk, yet may be implemented as a whole disk, a partition of a disk or perhaps an aggregation such as a RAID stripe. Note that virtualization is specific type of abstraction that seeks to preserve the interfaces and general properties of the abstracted entity. See also: *Abstraction*
VMM	See *Virtual Machine Monitor/Manager*

5 References

1. "EGA Reference Model and Use Cases v1.5" Part 2 of 2, Enterprise Grid Alliance, 10 March 2006.
2. Distributed Management Task Force (DMTF), http://www.dmtf.org
3. DMTF Common Information Model (CIM), http://www.dmtf.org/standards/cim
4. I. Foster et al., "The Open Services Grid Architecture, Version 1.0", Global Grid Forum, Lemont, Illinois, U.S.A., GFD-I.30, 2005, http://www.ggf.org/documents/GWD-I-E/GFD-I.030.pdf
5. Global Grid Forum (GGF), http://www.ggf.org
6. Storage Networking Industry Association, http://www.snia.org
7. ITIL - http://www.itsmfusa.org/mc/page.do?sitePageId=2995
8. eTOM - http://www.tmforum.com/browse.asp?catID=1647

6 Appendix A – Grid Component Life Cycle Examples

6.1 Example 1 – Server Life Cycle

Figure xxiii - Server (Compute Element) Grid Component Life Cycle – shows the life cycle of a compute element grid component, i.e. a server. The server is first made physically manageable, i.e. it is purchased and made physically manageable by the data center operators. It can then be configured so that it is logically manageable. This may involve physically connecting it to the management network (perhaps via a terminal server) and to remote power management devices. Once it is logically manageable, the server may be powered up, its BIOS or OBP configured and then it may be assigned for use by other grid components, such as an operating system. Note that the Active state is a macro state that may be comprised of many micro-states which can then capture whether or not an OS is booted or whatever.

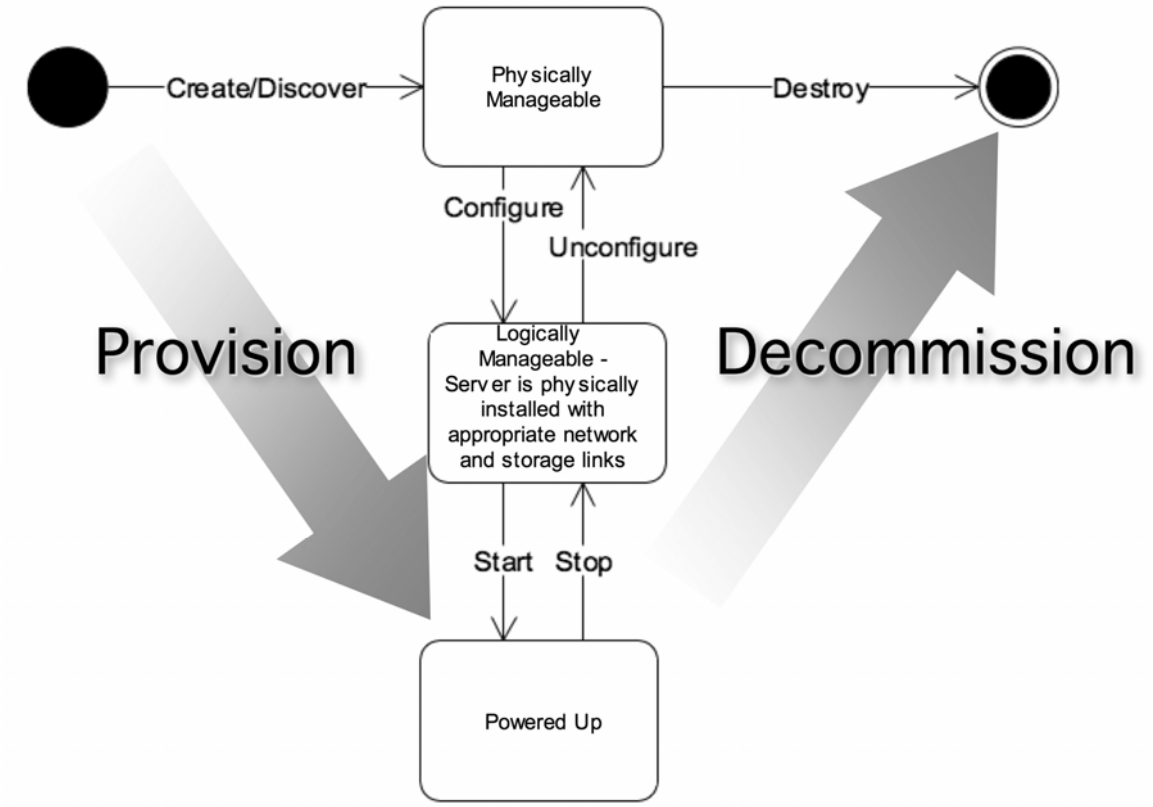

Figure xxiii - Server (Compute Element) Grid Component Life Cycle

6.2 Example 2 – Operating System Instance

Figure xxiv - OS Grid Component Life Cycle – illustrates the more complex life cycle of an Operating System grid component. Whilst the general life cycle is identical to that of the server grid component, there are some subtle differences. Specifically one discovered instance of the operating system, for example originally on a distribution CD, may result in multiple copies of the OS binaries. Each of these may result in multiple configured copies of the binaries, each of which could be bound (i.e. booted) on a number of servers (for example identical blades) concurrently.

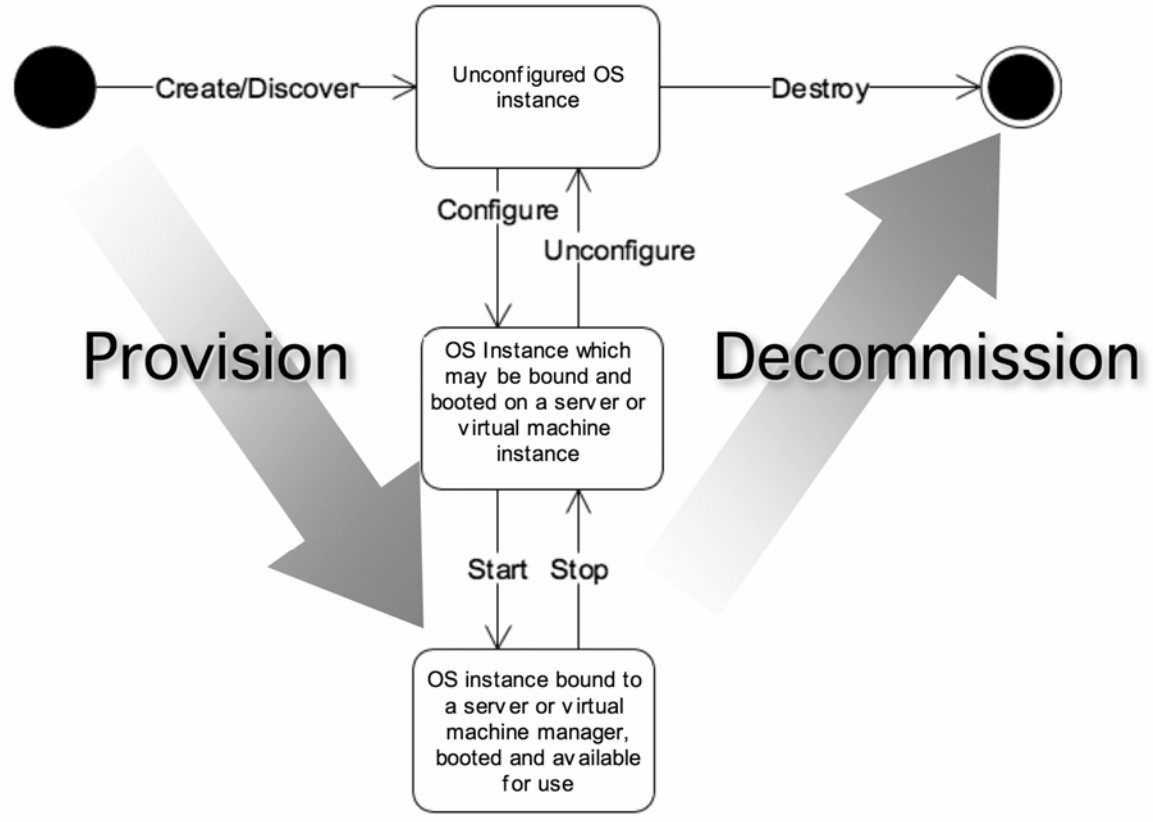

Figure xxiv - OS Grid Component Life Cycle

