Banking Grid Access Control Frameworks

A project report submitted in partial fulfillment
of the requirements for the Award of the Degree of

Master of Technology
in
Information Technology
(With specialization in Banking Technology and Information Security)

by

D.ROHITH KUMAR
(08MCMB06)

Under The Guidance of
Dr.V. N. Sastry

Institute for Development and Research in Banking Technology
Road No. 1, Castle Hills, Masab Tank,
Hyderabad – 500057

Submitted to
Department of Computer and Information Sciences
School of Mathematics and Computer/Information Sciences
University of Hyderabad
Hyderabad – 500046
A.P., India.
April 2010

CERTIFICATE

This is to certify that the project work entitled **"*Banking Grid Access Control Frameworks",*** submitted for partial fulfillment of the requirements for the award of the Degree of **Master of Technology in Information Technology** (M.Tech (I.T.)) (**with specialization in Banking Technology and Information Security**) to the **University of Hyderabad** is a record of bonafide project work carried out by **Mr. D.ROHITH KUMAR (Reg. No. 08MCMB06)** at IDRBT (Institute for Development and Research in Banking Technology), Hyderabad, during July 16, 2009 to April 29, 2010 under my guidance.

The subject matter embodied in the project report has not been submitted for the award of any other degree or diploma.

Dr. V. N. Sastry,
Associate Professor,
IDRBT, Castle Hills,
Masab Tank,
Hyderabad–500057
Email ID: vnsastry@idrbt.ac.in

CERTIFICATE

This is to certify that the project work entitled "*Banking Grid Access Control Frameworks",* submitted for partial fulfillment of the requirements for the award of Degree of **Master of Technology in Information Technology** (M.Tech (I.T.)) (**with specialization in Banking Technology and Information Security**) to the University of Hyderabad is a record of bonafide project work carried out by **Mr. D.ROHITH KUMAR** (Reg.No**.08MCMB06**) at IDRBT (Institute for Development and Research in Banking Technology), Hyderabad, during July 16, 2009 to April 29, 2010 under the guidance of **Dr. V. N. Sastry**, Associate Professor (IDRBT).

Dr. Mahil Carr
Associate Professor,
Coordinator – M. Tech (I.T.)
IDRBT,
Hyderabad.

Prof. Arun Agarwal
Head of the Department, DCIS,
University of Hyderabad,
Gachibowli,
Hyderabad.

Prof. T. Amaranath
Dean (School of MCIS),
University of Hyderabad,
Gachibowli,
Hyderabad.

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to Dr. V. N. Sastry, Associate Professor, IDRBT, Hyderabad, who generously supported and guided me throughout my project, IDRBT for providing me with the infrastructure and technical support that I needed for this project. The project would not have been possible without his assistance.

I also thank Mr. B. Sambamurthy, Director, IDRBT, Prof. T. Amaranath, Dean, School of MCIS, Prof. Arun Agarwal, Head of the Department (DCIS), University of Hyderabad for extending their cooperation.

The guidance of all the faculty members of IDRBT and the Department of Computer and Information Sciences, University of Hyderabad has been precious and timely. I wish to thank them for being very patient, understanding and helpful.

D.Rohith Kumar,
University of Hyderabad & IDRBT,
Hyderabad.
E-mail ID: rohithdhatric.hcu@gmail.com

LIST OF RESEARCH PAPERS

[1] Rohith Kumar, V.N. Sastry, "**Banking Grid Portal Based on SSH Frameworks**". (To be communicated)

[2] Rohith Kumar, V.N. Sastry, "**RBAC XACML in Grid Environments**". (To be communicated)

Abstract

GRID has evolved as an integration infrastructure for sharing and coordinated use of diverse resources in dynamic, distributed virtual organizations (VOs). Banking Grid is an integrated infrastructure that enables collaborative use of computational power of high end computing devices such as servers and payment switches, heterogeneous applications such as file system, databases, core banking software, security services and information storage platforms etc. owned and managed by multiple banks or service providers. To build enterprise information grid, the following must be addressed:

1. All enterprise information resources are distributed in a number of heterogeneous computer nodes. Different organizations, departments demand for access control as a level of security and use of the access control based strategy may be different from one another. Therefore, the need to use portable, platform independent, unified approach and more extensive vocabulary to describe the content of these access controls to resource owners can be fine-grained access control.
2. A large number of dynamic grid users and resources will make the overall access control Inefficient, cost surge. Therefore, a need to reduce the complexity of authorization management, reduce administrative overhead. For these reasons to improve grid security we use XACML and Role Based Access Control (RBAC) mechanisms for enterprise information grid.
3. To develop a banking grid portal based on J2EE technology through which grid users can securely access and share grid resources and make used of various grid services.

The project focuses on the study of Role Based Access Control (RBAC) for building security and trust among banks to share common infrastructure and costly resources. Struts, Spring and Hibernate which are based on J2EE technology are open source frameworks, which are platform independent and enhances inter-operability. These frameworks are mostly used for building web applications. This project analyses the integration of the above mentioned three frameworks according to their advantages and apply them to develop a secure Banking Grid portal.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 Grid Computing:

The concept of coupling geographically distributed (high-end) resources for solving large-scale problems is becoming increasingly popular; forming what is popularly called grid computing. Grid computing is becoming more and more important in the current days as a possible solution for the increasing needs of resources in terms of computing power and storage capacity for the current research challenges in the world.

This new paradigm offers very amazing views for the ways in which the new generation information networks will be established and managed as well as the new highest performances ever seen. However, its difficulties are also higher than before because this new perspective needs very abstract and complex tasks to perform high-end, general and extensible results. This new environment is mainly based on the wide distribution of services. Services that provide high-end features to a global infrastructure and delivered through different partners and different sets of non-homogeneous resources.

In recent years, the amount of data being stored, accessed and analyzed has increased tremendously. The advancements in hardware and software have further facilitated the storage and access of this enormous data. In addition, the datasets are also stored in a distributed manner and are shared by many people across several different domains. Grid technologies and infrastructure are developed to support the sharing and coordinated use of diverse resources in dynamic, distributed virtual organizations (VOs) [1]. A virtual organization is termed as a set of individuals and resources across multiple organizations (for example, a collaboration formed by two independent physical organizations). These individuals and resources can be dynamic in nature. There are a number of projects worldwide [2], which are actively exploring the development of various grid computing system components, services, and applications. They include Globus [3], Legion [4],NetSolve [5], Ninf [6], AppLes [7], Nimrod/G [8], and JaWS [9].

According to Ian Foster, the essence of grid computing can be captured in a simple checklist [10], according to which a grid is a system that:

- ***Coordinates resources that are not subject to centralized control***

A grid integrates and coordinates resources and users that live within different control domains, for example, the user's desktop vs. central computing; different administrative units of the same company; or different companies; and addresses the issues of security, policy, payment, membership, and so forth that arise in these settings

- ***using standard, open, general-purpose protocols and interfaces***

A grid is built from multi-purpose protocols and interfaces that address such fundamental issues as authentication, authorization, resource discovery, and resource access. These protocols and interfaces need to be standard and open so that system is able to execute generic applications.

- ***to deliver nontrivial qualities of service***

A grid allows its constituent resources to be used in a coordinated fashion to deliver various qualities of service, relating - for example - to response time, throughput, availability, and security, and/or co-allocation of multiple resource types to meet complex user demands, so that the utility of the combined system is significantly greater than that of the sum of its parts. Grids are becoming mission-critical components in research and industry, offering sophisticated solutions in leveraging large scale computing and storage resources. Grid resources are usually shared among multiple organizations in an opportunistic manner. However, an opportunistic or "best effort" quality-of-service scheme may be inadequate in situations where a large number of resources need to be allocated and applications which rely on static, stable execution environments.

1.1.1 Elements of Grid Computing

Grid has emerged recently as an integration infrastructure for the sharing and coordinated use of diverse resources in dynamic, distributed virtual organizations (VOs).A Data Grid is an architecture for the access, exchange, and sharing of data in the Grid environment. It provides a distributed system middleware that allows different communities to access and share data, networks, and other resources in a controlled and secure manner [11].

Present-day grids encompass the following types:

1. Computational grids, in which machines will set aside resources to "number crunch" data or provide coverage for other intensive workloads.

2. Scavenging grids, commonly used to find and harvest machine cycles from idle servers and desktop computers for use in resource-intensive tasks (scavenging is usually implemented in a way that is unobtrusive to the owner/user of the processor)
3. Data grids [12], which provide a unified interface for all data repositories in an organization, and through which data can be queried, managed and secured.
4. Market-oriented grids [13], which deal with price setting and negotiation, grid economy management and utility driven scheduling and resource allocation.

Scientific and Business communities are increasingly collaborating, and this is giving rise to the need for more sophisticated technologies for data and resource sharing. Data Grids reduce hardware and software costs by enabling the secure exchange of programs and data between collaborating organizations. Without a Data Grid, a separate set of resources are purchased (and managed) in a demilitarized zone (DMZ) behind a completely separate firewall. By using Data Grid technology, there is no need to build a separate DMZ [14].Data Grids facilitate the management of distributed heterogeneous data. The burden of managing the operations is removed from the user. The collective operations required are all managed by the system via a single sign-on and uniform querying mechanism for the user. The motivation behind such a system is to address the following considerations [15]:

1. Large data set size, geographic distribution of users and resources, and computationally intensive analysis results in complex and stringent performance demands that are not satisfied by any existing data management infrastructure;
2. No integrating architecture exists that allows us to identify requirements and components common to different systems and hence apply different technologies in a coordinated fashion to a range of data-intensive large-scale application domains. Current technology cannot easily handle these scenarios which require the coordinated sharing of data and resources across multiple organizations. It either does not accommodate the range of resource types or does not provide the flexibility and control on sharing relationships [1].

The key components of grid computing include the following:

1. Resource management: a grid must be aware of what resources are available for different tasks
2. Security management: the grid needs to take care that only authorized user can access and use the available resources

3. Data management: data must be transported, cleansed and processed.
4. Services management: users and applications must be able to query the grid in an effective and efficient manner [16]

More specifically, grid computing environment can be viewed as a computing setup constituted by a number of logical hierarchical layers. Figure 1 represents these layers. They include grid fabric resources, grid security infrastructure, core grid middleware, user level middleware and resource aggregators, grid programming environment and tools and grid applications.

The major constituents of a grid computing system [17] can be identified into various categories from different perspectives as follows:

- Functional view
- Physical view
- Service view

Basic constituents of a grid from a *functional* view are decided depending on the grid design and its expected use. Some of the functional constituents of a grid are

1. Security (in the form of grid security infrastructure)
2. Resource Broker
3. Scheduler
4. Data Management
5. Job and resource management
6. Resources

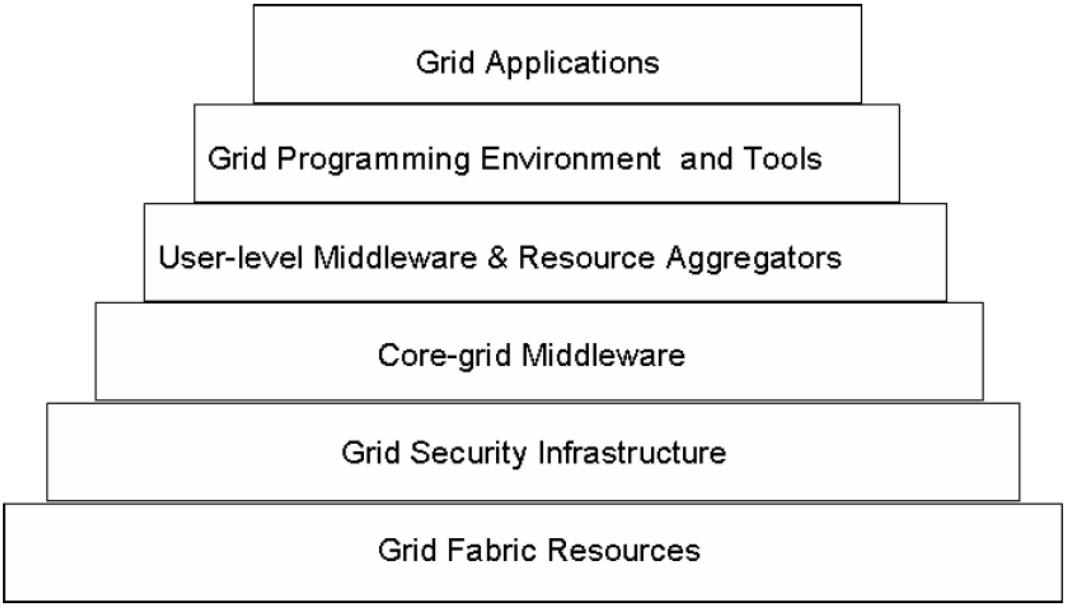


Figure 1: Grid Computing Layers

A resource is an entity that is to be shared [18]; this includes computers, storage, data and software. A resource need not be a physical entity. Normally, grid portal acts as a user interaction mechanism which is application specific and can take many forms. A user-security functional block usually exists in the grid environment and is a key requirement for grid computing. In a grid environment, there is a need for mechanisms to provide authentication, authorization, data confidentiality, data integrity and availability, particularly from a user's point of view. In the case of inter-domain grids, there is also a requirement to support security across organizational boundaries. This makes a centrally managed security system impractical. The grid security infrastructure (GSI) provides a "single sign-on", run-anywhere authentication service with support for local control over access rights and mapping from global to local identities.

Different types of resources:

- *Hardware resource* - A hardware resource represents a particular host on the Grid. Hardware resources contain service resources, software resources and hardware accounts, all of which are described below.
- *Service resource* - A service resource is a resource that represents a "service" that is accessible over a network. All service resources have at least one "port" associated with them, through which they can be invoked by clients, and a "protocol" for communicating with the service.

- *Software resource* - A software resource is a resource that represents“software”. At minimum, a software resource has a “path” on a given hardware resource.
- *Resource account* - A resource account represents an “account” on a resource. For example, a *hardware account* represents an account on a particular hardware resource.

1.1.2 Grid Architecture

Grid architecture definition starts from the Virtual Organizations point of view because is the main entity for providing the desired virtualization and provisioning characteristics. From this view, the main task is automation and management of relationships from any potential user. So interoperability appears as the main goal to address and therefore the protocol stack.

Interoperability is the key point in order to guarantee the proper robustness and flexibility which allow relationships to be established among arbitrary entities. Otherwise, users or applications (on behalf of users) are forced to enter into bilateral arrangements, what does not ensure extensibility to third parties. The way of achieving interoperability is across definition of standard protocols. These ones specify how distributed system elements interact with others and the information exchanged during this interaction, but focusing on external requirements and giving internal implementation details up. This also preserves control over local resources so individual organizations control still their own resources and user policies.

Consequently, the open architecture for Grid computing to address those issues has been defined in [1] [19]. It is based on a conical model whose vertex is composed of the core protocols (a small number itself), different underlying technologies are under it and the high-end applications are mapped onto it.

It is important to denote the analogy with current Internet model shown in figure 2, where Internet Protocol is in the core, over it a broad range of different protocols for different purposes and under it the set of protocols that enable access to different physical resources. As this model has brought Internet expansion successfully, the same Grid approach should get the same success.

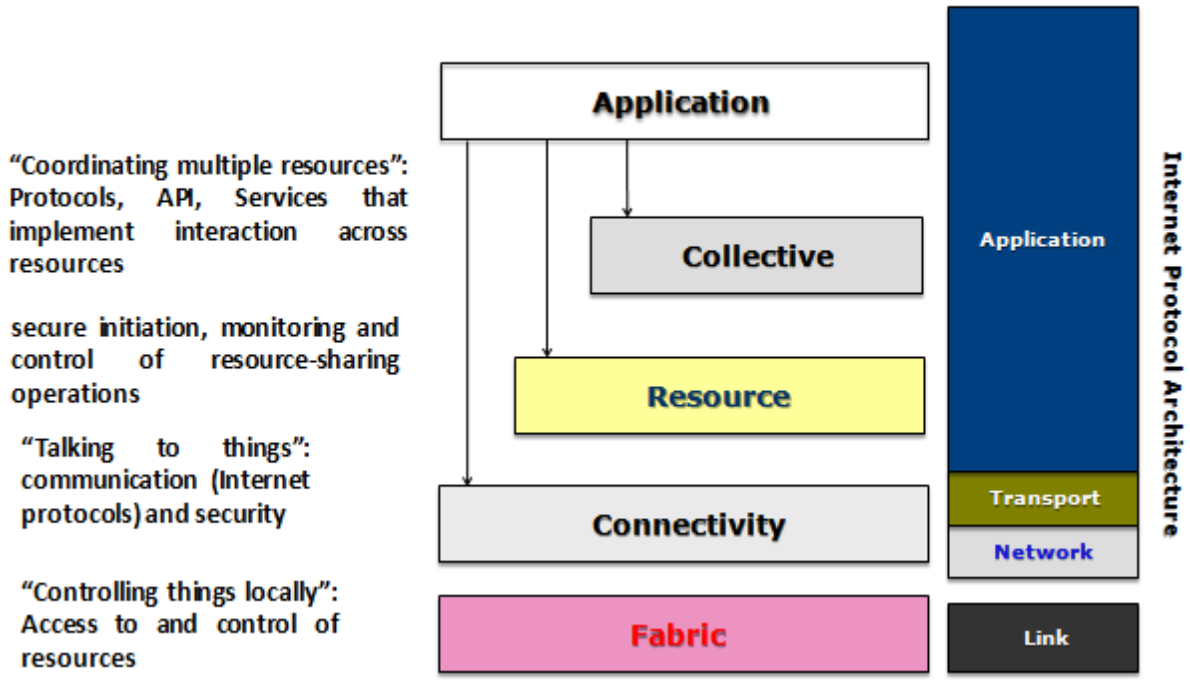


Figure 2: Grid Architecture **Vs** IP Architecture

Application layer

Application layer is the last level in the Grid protocols stack and comprises all the end-to-end user applications which interact with collective layer (also with lower layers) in order to provide all desired features. Applications are based on high level languages and frameworks that implement the mentioned services within every layer. So that is the mean for achieving interoperability and portability at user level.

Collective layer

The collective layer is built upon the resource layer to extend its individual capabilities and provide global aware interactions across collection of resources hosted in different locations. Again, like in the TCP/IP stack equivalent (transport level), this layer can be considered the first end-to-end layer. As a result of this design, provisioning properties mentioned before are supplied by this layer.

Some of these services are the following ones:

1 Directory services: discovering and enquiring of VOs properties or statuses.

2 Co-allocation, scheduling and brokering services: to request and establish distributed allocation of jobs.

3 Monitoring and diagnostic services: for supporting failure tolerance and improve delivered QoS.
4 Data replication services: for maximizing performance data transfers in term of latency, reliability and cost.
5 Grid-enabled programming services: for enabling high abstract programming languages to use global Grid services seamlessly.
6 Software discovery services: to discover best software implementations in term of platform requirements and user requirements.
7 Authorization services: acting as the policy definition point inside each VO (implementing RBAC security model).
8 Collaborative services: for coordinated exchange of information between large user communities.

Resource layer:

Resource layer protocols enable secure initiation, monitoring and control of operations over individual resources. Like in the analogy with IP stack, this layer establishes the constraints upon which upper layer protocols will interact with resources. So, this level performs its actions entirely over individual resources. This distinction is the key oppositeness to collective layer in terms of, for example, atomic issues (like the reliability property provided only by TCP in TCP/IP model).

Two classes of protocols can be considered:

1 *Information protocols:* for enquiring about structure details or statuses like load or policy issues.
2 *Management protocols:* for accessing and processes creation, monitoring and control. The important aspect is this layer serve as policy application point (so called policy enforcing point) in order to ensure consistency with the policy definition under which the resource is shared.

Connectivity layer

The connectivity layer establishes the communication and authentication protocols which enable data exchange among different fabric resources. It is built upon fabric layer in order to provide the abstraction to resource layer which enable seamlessly accesses to fabric resources.

Communication requirements comprise transport, routing and naming capabilities. In most of the cases, this will be mapped to equivalent protocols in IP stack, but general use of resources, specially networks, involves the development of some mechanisms for adding extensibility and abstraction to that approach.

Regarding authentication requirements, this layer should provide the next features:

1 Single sign on (SSO): ability to get authenticated only once (just entering into Grid services).
2 Delegation: ability to endow a program to access services under user's behalf. This one together with SSO enable the integral security model required in Grid environments.
3 Integration with local security: seamless integration with local security solutions already deployed into some organization (such us Kerberos, Active Directory, Java Enterprise System, etc.)
4 User-based trust relationships: mechanisms for centralizing usage granting when services are provided among different providers. This avoids interactions between them as a prerequisite of use.

Fabric layer

The fabric layer is the lowest one in the Grid protocols stack so this one is the main responsible to provide access to the capabilities offered by resources. In this case, resources comprise physical ones (computational, storage, network, etc.) or logical sets of these ones (clusters, distributed file systems, etc.).

Of the two main characteristics of Grid, this layer mainly provides the virtualization property for higher level protocols so it must deal with the details of implementations of every resource. Thus, there is a subtle dependency between sharing operations that can happen at higher levels and the functions implemented at fabric one. Features provided by resources (i.e. the fabric layer) should be prepared for enquiry and management mechanisms. The first ones allow browsing the resources structure and eventually knowing their status and the second ones use of their capabilities under some controlled way to provide quality of service over demand.

Finally, some capabilities that should be provided at this layer are:

1 Computational resources: monitor and control of processes, allocation, reservation, status, characteristics browsing, etc.
2 Storage resources: global and performance transferring, capabilities management or enquiry mechanisms.
3 Network resources: control over resources for enabling QoS, enquiry mechanisms, etc.
4 Code repositories: software configuration management capabilities for source and object code.
5 Catalogs: datasets for browsing and discovering of resources, configuration information for operations and management, etc.

1.1.3 Main characteristics of Grids

The main characteristics of a grid computing environment can be listed as follows:

1 **Large scale:** A grid must be able to deal with a number of resources ranging from just a few to millions.
2 **Geographical distribution:** Grid resources may be spread geographically.
3 **Heterogeneity:** A grid hosts both software and hardware resources that can be ranging from data, files, software components or programs to sensors, scientific instruments, display devices, personal digital organizers, computers, super-computers and networks.
4 **Resource sharing and coordination:** Resources in a grid belong to different organizations that allow other organizations (i.e. users) to access them. The resources must be coordinated in order to provide aggregated computing capabilities.
5 **Multiple administrations:** Each organization may establish different security and administrative policies under which resources can be accessed and used.
6 **Accessibility attributes:** Transparency, dependability, consistency, and pervasiveness are attributes typical to grid resource access. A grid should be seen as a single virtual computing environment and must assure the delivery of services under established Quality of Service requirements. A grid must be built with standard services, protocols and interfaces thus hiding the heterogeneity of the resources [195] while allowing its scalability. A grid must grant access to available resources by adapting to dynamic environments where resource failure is common.

1.2 Access Control Models

Research on security has developed so far a number of models to deal with access control solutions. The authorization process is used to protect the system resources, allowing those resources to be used by consumers that have been granted authority to use them.

Access means entry, approach, or privilege to a resource. Access control is a mechanism to secure resources from unauthorized use [20]. It constrains what a user can do directly as well as what the programs executing on the user's behalf are allowed to do. Access control models use a set of rules, which permit or deny access for a subject to an object [21]. This ensures that information does not fall into wrong hands. The process involves a subject requesting for an object. The permission or denial of access to the object depends upon the 'right' that the subject possesses.

The basic way to model access control is a four tuple: (S; O; A; M), where S is the set of subjects, O is the set of objects, A is the set of actions (access rights), and M is a function that maps a $tuple(s; o; a) \in S \times O \times A$ to a decision $\in \{T; F\}$. The mapping M can be stored in an access matrix, with rows corresponding to subjects, columns corresponding to objects, and matrix entries indicating allowed access rights. In practice, a typical access matrix is large and sparse, and it is difficult to store, manage, and understand such a matrix directly. Therefore, various access control policies have been developed.

1.2.1 Access Control Policies

1 ***Discretionary Access Control (DAC):*** It is a means of restricting access to objects based on the identity of subjects and/or groups to which they belong. The controls are discretionary in the sense that a subject with certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject (unless restrained by mandatory access control). DAC permits the granting and revoking of access privileges to be left to the discretion of individual users. This is based on the notion that individual users are \owners" of objects. Ownership is usually acquired as a consequence of creating the object. The DAC model had its own drawbacks. It does not provide real assurance on the flow of information in a system. Also it does not impose any restriction on the usage of information by a user once the user has received it.

2 ***Mandatory Access Control (MAC):*** MAC is defined as a means of restricting access to objects based on sensitivity (as represented by a label) of the information

contained in the objects and the formal authorization (i.e., clearance) of subjects to access information of such sensitivity. The different security levels in a system form a lattice. MAC is typically used to enforce one directional information flow in such a lattice. The rule for read access requires that a user with a given clearance level can only read information with the same or lower classification level. The rule for write access requires that a user with a given clearance level can only write information to a target with the same or higher classification level. This prevents a user from declassifying information without authorization.

3 ***Originator Controlled Access Control (ORCON):*** It is advancement over Mandatory and Discretionary Access Controls models, as both these models cannot handle environments in which the originators of documents retain control over them even after those documents are disseminated. In ORCON, a subject can give another subject rights to an object only with the approval of the creator of that object. Organizations that use categories grant access to individuals on a "need to know" basis. There is a formal, written policy determining who needs the access based on common characteristics and restrictions.

The main features of ORCON are as follows.

- The owner of an object cannot change the access controls of the object
- When an object is copied, the access control restrictions of that source are copied and bound to the target of the copy
- The creator (originator) can alter the access control restrictions on a per-subject and per-object basis

In ORCON, the access control associated with the object is under the control of the originator and not the owner of the object. Possession equates to only some control. The owner of the object may determine to whom he or she gives access, but only if the originator allows the access. The owner may not override the originator.

4 ***Role Based Access Control:*** The standard discretionary access control and mandatory access control approaches cannot cater to an enterprise's access control requirements as the former is a user discretion mechanism and the later an approach more suitable for Operating System security. The need to evolve a new model for enterprise-wide security lead to the development of Role Based Access Control Model or the RBAC.

The Role Based Access Control Model is a technical means for controlling access to computer resources [22]. With role-based access control, access decisions are based on the *roles* that individual users have as part of an organization [23]. A *role* is the standard unit of access control in RBAC and reflects the responsibilities of a user in an organization. Users take on assigned roles (such as doctor, nurse, teller, manager). The process of defining roles should be based on a thorough analysis of how an organization operates and should include input from a wide spectrum of users in an organization. Access rights are grouped by role name and the use of resources is restricted to individuals authorized to assume the associated role [24].For example, within a hospital system the role of doctor can include operations to perform diagnosis, prescribe medication, and order laboratory tests; and the role of researcher can be limited to gathering anonymous clinical information for study.

The National Institute of Standards and Technology proposed the following standards for RBAC models [25]

- Core RBAC
- Hierarchical RBAC
- General Hierarchical RBAC
- Restricted Hierarchical RBAC
- Constrained RBAC
- Symmetric RBAC

Core RBAC sets guidelines for the basic RBAC functionalities. Figure 3 represents the set of users, roles and operations and objects (together known as privileges) as per the core-RBAC standard. It also depicts the user-role assignment and role-permission assignment. Figure 4 shows the hierarchical relationship between various components of the RBAC model. This category of RBAC model is called hierarchical RBAC model.

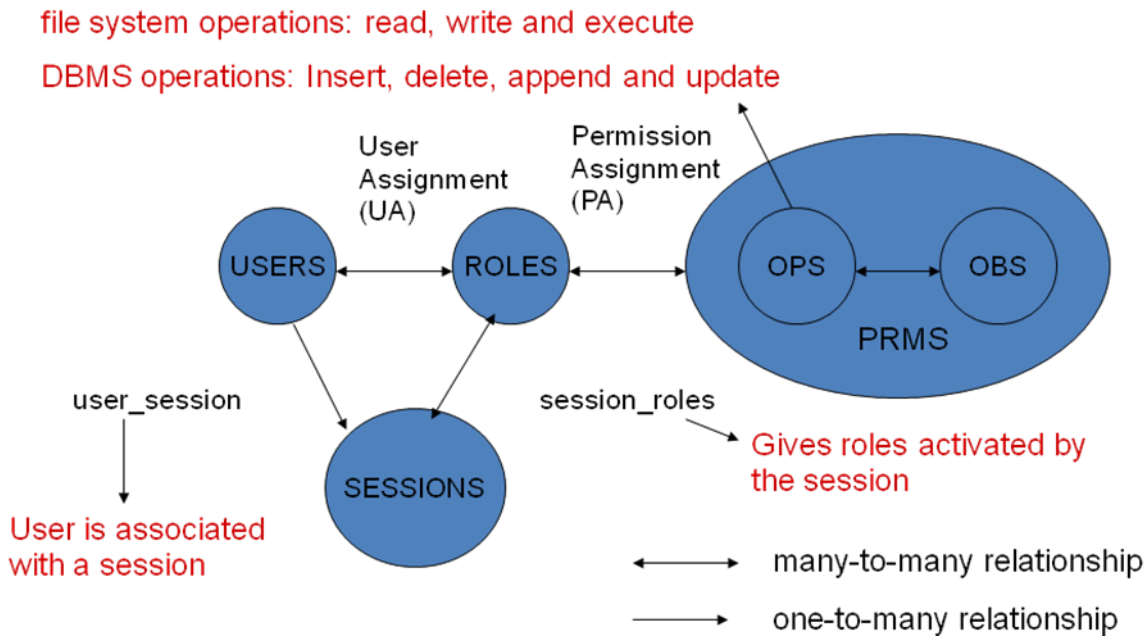


Figure 3: Core RBAC Model

Roles can be an effective means for developing and enforcing enterprise-specific security policies [26, 27] and for streamlining the security management process [28]. Under the RBAC framework, users are granted membership into roles based on their competencies and responsibilities in the organization. The operations that a user is permitted to perform are based on the user's role. User membership into roles can be revoked easily and new memberships established as the job assignments dictate. Role associations can be established when new operations are instituted, and old operations can be deleted as organizational functions change and evolve [29]. This simplifies the administration and management of privileges [30]; roles can be updated without updating the privileges for every user on an individual basis.

When a user is associated with a role, the user can be given no more privilege than is necessary to perform the job. This concept of least privilege requires identifying the user's job functions, determining the minimum set of privileges required to perform that function and restricting the user to a domain with those privileges and nothing more. In less precisely controlled systems, this is often difficult or costly to achieve. Someone assigned to a job category may be allowed more privileges than needed because it is difficult to tailor access based on various attributes or constraints [31]. Since many of the responsibilities overlap between job categories, maximum privilege for each job category could cause unlawful access.

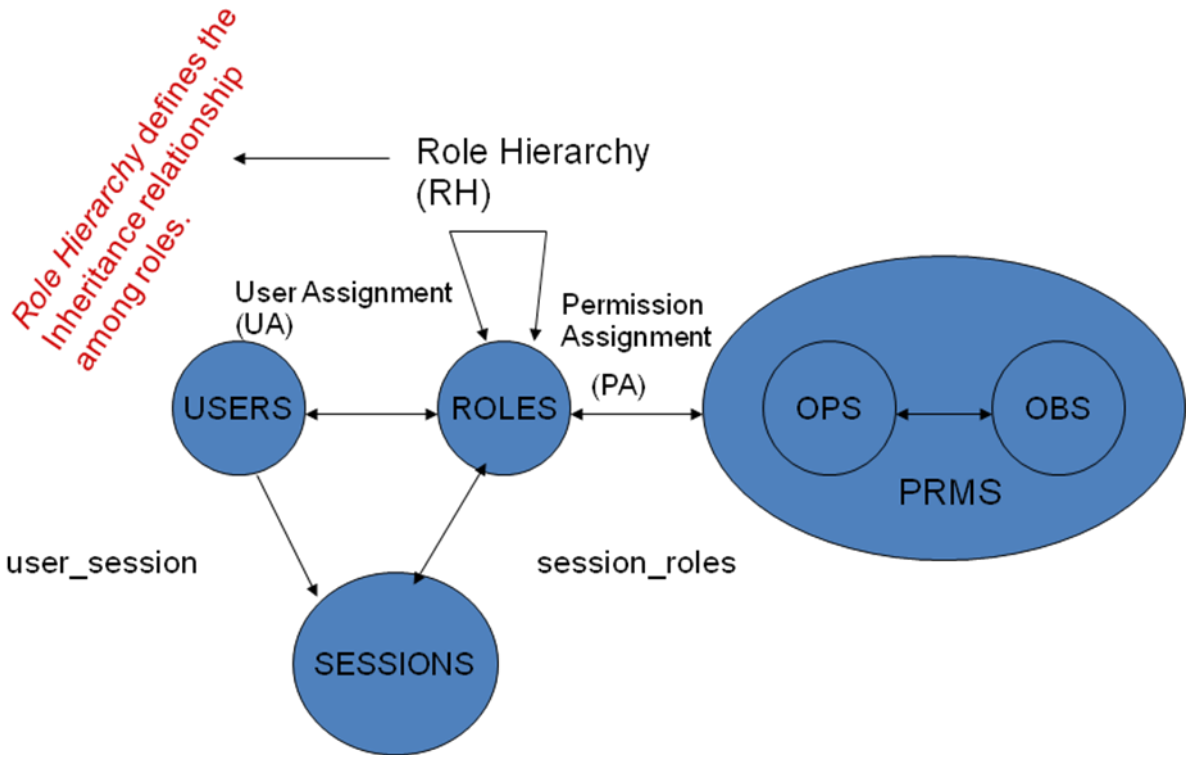


Figure 4: Hierarchical RBAC

The advantages of RBAC are as follows. A properly-administered RBAC system enables users to carry out a broad range of authorized operations, and provides great flexibility and breadth of application. System administrators can control access at an abstract level that is natural to the way that enterprises typically conduct business. This is achieved by statically and dynamically regulating users' actions through the establishment and definition of roles, role hierarchies, relationships, and constraints [32]. Thus, once an RBAC framework is established for an organization, the principal administrative actions are the granting and revoking of users into and out of roles. This is in contrast to the more conventional and less intuitive process of attempting to administer lower-level access control mechanisms directly (e.g., access control lists [ACLs], capabilities, or type enforcement entities) on an object-by-object basis.

Table 1 shows the evolution of various access control models.

Name of Access Control Model	Year of Evolution
MAC and DAC	1975-1985
Alternative Models (CW, TBA, RBAC)	1985-1995
After RBAC (GTRBAC, Partial Outsourcing	1995-2003
Recent Models (Domain Specific)	2003-Till Date

1.2.2 Comparison of Access Control Models

Access control models can be compared based on various factors [33] like:

- Expressive power of the models
- Inheritance of privileges
- Type of access control structures used
- Delegation of roles
- Constraints in permissions
- Existence of negative permissions
- Security analysis and evaluation
- Propagation of rights
- Facilities for Role Engineering
- Applicability in static and dynamic security environments
- Scalability
- Flexibility and ease of configuration
- Implementation mechanisms
- Security assurance

Table 2 shows the comparison of RBAC model with non-RBAC models with regard to user privileges and cost comparison [34]. The table reflects the cost benefits for RBAC-oriented security mechanisms over non-RBAC mechanisms. The values given in the table are the estimated time (in minutes) required for performing access administration. We can observe a clear advantage for RBAC-oriented administrative tasks over the non-RBAC ones with regard to the time factor

Table 2: Comparison of RBAC and Non-RBAC

Task	RBAC	Non-RBAC	Difference
Assign existing privileges to new users	6.14	11.39	5.25
Change existing users privileges	9.29	10.24	0.95
Establish new privileges for existing users	8.86	9.26	0.40
Termination of privileges	0.81	1.32	0.51

1.2.3 Access Control Mechanisms

Techniques to implement the above access control policies include the following:

Access Control Lists (ACLs): An access control list is an attribute of a target object, stating which users can invoke which actions on it. An access control list specifies the contents of the column related to the target object in the access control matrix. Access Control Matrix is shown in figure 5

Capabilities: Capabilities are results of decomposing the access matrix by rows. In this scheme, associated with each subject is a list that defines the objects to which the subject has access rights and what are those authorized rights. A capability is effectively a ticket, possessed by a requester that authorizes the holder to access a specified object in specified ways. Some capabilities can only be used by a specified principal, while others may be transferred to other principals.

Security Labels: A security label is a set of security attribute information bound to a user, a target, or a piece of information in transmission. The label indicates the sensitivity level of the data. This mechanism is used to implement MAC.

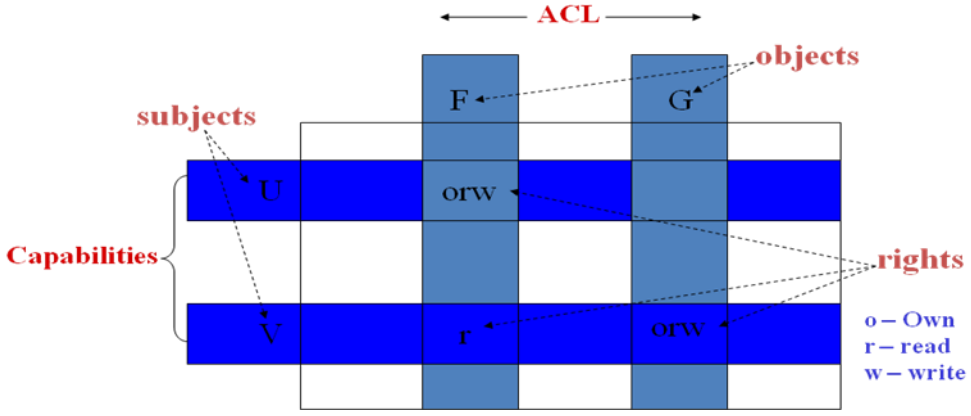


Figure 5: Access Control Matrix

1.3 Motivation & Problem Statement:

User authentication, authorization is one of the most challenging issues in Grid Environment. Current authorization mechanisms cannot address all the issues that arise in dynamic Grid environments which often encompass multiple organizations, each with its own security policy [35].In a grid environment users, resources are geographically distributed so there is a need to provide access to these resources based on user role and permissions needed to access a particular resource .The typical identity-based authorization used in Grids today is not scalable because authorization information should be maintained for each user. In role-based access control (RBAC) [36] permissions are associated with roles, and users are assigned appropriate roles, there by acquiring the roles permissions [37].

Hence, RBAC is quite scalable since authorization information is associated with roles, not with individual users. RBAC shows clear advantages over traditional access control models in Grid environments, because it allows a uniform representation of diverse security policies and ensures that no security violations occur during inter-domain accesses [38]. None of the current access control systems in Grids provide comprehensive support for RBAC.

To address this issue my project focuses on different access control frameworks for sharing and accessing resources in a grid environment, users can access various grid resources any time. XACML is a standard of the Organization for the Advancement of Structured Information Standards (OASIS) for describing access control policies uniformly across different security domains [39]. The Core and Hierarchical RBAC profile of XACML defines how the ANSI core and hierarchical RBAC standard can be specified in XACML.SUN XACML [40] is used which is an open source implementation of the OASIS XACML standard, written in the Java programming language. It provides full support for parsing both policy and request/response documents, determining applicability of policies, and evaluating requests against policies. The entire standard attributes types, functions, and combining algorithms are supported, and there are APIs for adding new functionality as needed. There are also APIs for writing new retrieval mechanisms used for finding things like policies and attributes.

Grid portals build upon the familiar Web portal model to offer virtual communities of users a single point of sharing/access to computational resources. The banking gird portal which based on portlet technology can make full use of banking resources. Now a day's web portals are constructed based on J2EE, PHP and ASP.NET out of which J2EE and ASP.NET are

build on three tier architecture. Since J2EE is integrated and platform independent more companies use this technology in constructing business portals. There are many Open Source frameworks which are based on J2EE technology such as Struts, webwork, jsf, echo, Spring, Doff keel, Hibernate, Seam, jpa, ibatis, GlassFish, AppFuse and so on. [41]

In this project we have developed a banking grid portal based on struts, spring and hibernate frameworks which offers various grid services along with secure sharing/accessing of resources in a dynamic grid environment. Spring Security helped us in providing security and enabling single-on concept while accessing resources of other domain.

1.4 Contributions

We summarize the main contributions of the work below:

Banking Grid Monitoring Architecture: We have proposed an architecture which monitors all the activities in grid environment such as resource discovery, resource scheduling, resource allocation, resource sharing, resource costing and job scheduling etc

Banking Grid Portal Based on SSH Frameworks: We have developed a banking grid portal in which grid services are monitored, managed and resources are accessed online based on user role and action on a particular resource. Struts, Spring & Hibernation which are built on J2EE technology is used in developing this application which offers three level security to access/share resources of other domains.

RBAC-XACML Grid: We have written various accesses control policies based on XACML-RBAC profile 1.0, also maintained a policy database for various users, roles and explained whole scenario with an example. Also proposed XACML RBAC architecture for single domain banking grid enterprise where authorization decision is taken to give access/deny a resource.

1.5 Organization

Organization of thesis is as follows:

Chapter 1: Introduction to Grid Computing & Access Control Models

Chapter 2: Literature Survey

Chapter 3: Banking Grid Monitoring Architecture

Chapter 4: Banking Grid Portal Based on SSH Frameworks

Chapter 5: RBAC-XACML in Grid Environment

Chapter 6: Conclusion & Future Work followed by references and appendix

1.6 Conclusion

This chapter provides an introduction to grid computing, an emerging technology of enormous promise. We discuss the elements which constitute the grid and also its economic aspects. We also discuss in great detail about grid architecture .Next, we discuss on various access control policies and how they can be implemented and also comparison between them. The problem definition and the contribution of the thesis are presented along with the organization of the thesis.

CHAPTER 2
LITERATURE SURVEY

In the previous chapter we have seen various aspects of grid computing, grid architecture, access controls mechanisms etc...In this section, we present a detail literature survey in grid security and access control models. The aim of this survey work is to present current research trends from the point of view of the security models, grid monitoring models and authorization mechanisms available in the grid computing environment.

Ian Foster and Carl Kesselman make the initial definition of grid in their book in 1998. The name of the book is the Grid: Blueprint for a New Computing Infrastructure. The definition is computational grid is a reliable, consistent, ubiquitous, inexpensive hardware and software infrastructure, used for high-end computing as in [42]. Grid wants to make the Local Area Network, Metropolitan Area Network and even Internet into a huge supercomputer, achieve the overall share of knowledge resources, storage resources, computing resources, information resources and expert resources. As the same as internet, the concept of grid computing comes from the needs to achieve sharing resources among banking domains. The main feature of the grid is the share sharing of resources as in [43].

There have been some attempts at providing security to the resources of a grid environment through access control. Here we discuss some of such attempts. Kevin Kane and James C. Browne, in their research paper [44], presented a classification of implementations of access control systems based on lattice taxonomy. Pietro Mazzoleni et.al [45] discussed ways to provide fine-grained access control in large scale grid services. They developed a novel resource broker service for grids that integrates access control with resource scheduling. Weizhong Qiang et.al [46], proposed a general authorization and access control architecture, RBGACA, for grid environment. It is based on the classical access control mechanism - Role Based Access Control (RBAC). The architecture provides convenient policy evaluation and decision making approach. They provided a framework for access control management that treats the whole grid as a series of independent, interrelated, dynamic domains. Though this approach introduced RBAC in grids, it did not address many issues like delegation, scalability and trust relationships. Also, no provision for cross-domain authorization was made.

“XACML Policy Integration Algorithms”, a work done by P. Mazzoleni et.al [47], suggested the use of XACML, an OASIS [39] standard language for the specification of authorization and entitlement policies.

James B.D.Joshi et.al proposed an access control language for multi-domain environments in [48].

“Architectural Models for Resource Management in the Grid”,work done by Rajkumar Buyya, Steve Chapin , and David DiNucci [49] discussed three different models for grid resource management architecture namely Hierarchical Model,Abstract Owner Model, Computational Market/Economy Model.

“Administration of an RBAC System”,work done by Fredj Dridi, Bjorn Muschall and G¨unther Pernul[50] have CSAP, an autonomous software module offering programming interfaces to core security services such as authentication, access control, auditing and security management.They have implemented using Core RBAC concept

“XACML Policies for Exclusive Resource Usage”, work done by Vijayant Dhankhar, Saket Kaushik, and Duminda Wijesekera[51], proposed a locking mechanism which addresses issues such as XACML does not currently support access to globally available resources, preventing access to a resource given a concurrent conflicting use of another resource (DSoD constraints), and preventing access to a resource given a history of conflicting access.

“Security for Grids”, research done by MARTY HUMPHREY, MEMBER, IEEE, MARY R. THOMPSON, MEMBER, IEEE, AND KEITH R. JACKSON [52] discusses 4 important aspects of security i.e. naming and authentication; secure communication; trust, policy, and authorization; and enforcement of access control. It examined the current state of the art in securing these activities and introduced new technologies that promise to meet the security requirements of Grids more completely.

“A CROSS - DOMAIN ROLE MAPPING AND AUTHORIZATION FRAMEWORK FOR RBAC IN GRID SYSTEMS”, research done by G GEETHAKUMARI, DR V N SASTRY, DR ATUL NEGI [53] proposed an architecture based on RBAC, which can establish role equivalence among the domains by mapping a local domain role to its equivalent global role

and used weighted rank concept for resources, also addressed solutions for delegation and revocation in a single domain grid enterprise.

Also a comprehensive set of grid usage scenarios were presented and analyzed by M Humphrey and M R Thompson in the year 2001 [54], with regard to security requirements such as authentication, authorization, integrity, and confidentiality. There were also works which made suggestions for implementing security without much performance degradation in grids.

Jeffrey Dwoskin et.al [55], in their research work, defined three generic grid security scenarios: mutual trust, partial trust (distrusted user) and mutual distrust. According to them, decreasing levels of trust enable one to expose new vulnerabilities and show the increasing levels of security support required.

"A Performance Oriented Grid Monitoring Architecture", work done by S. De Smet, P. Thysebaert_, B. Volckaert, M. De Leenheer, D. De Winter, F. De Turcky, B. Dhoedt, P. Demeester[56] presented a performing and portable implementation of the GGF Grid Monitoring Architecture. Performance was obtained through the use of C++ as base implementation language;also described a scalable, portable and nonintrusive Grid Monitoring Architecture.

"GridRM: A Resource Monitoring Architecture for the Grid", work done by Mark Baker and Garry Smith [57] described a generic open-source resource monitoring architecture that has been specifically designed for the Grid. GridRM is an open source, generic, distributed, resource-monitoring system for Grid environments.It is composed of a global GridRM layer, based on the Global Grid Forum's, Grid Monitoring Architecture, and a local GridRM layer.

"Defining methodologies for developing J2EE web-based information systems", work done by Askar S. Boranbayev [58], describes the concepts behind the developed framework for Java-based projects and describes how it can be used for IT projects. The developed framework was created because many common design and development tasks are being repeated in different ways, and are not always consistent with best practices and also discussed on J2EE framework and also has addressed important architecture topics, technologies and development steps that one should consider in a J2EE project.

"Research of Structure Integration Based on Struts and Hibernate" work done by Juanjuan Yan Bo Chen Xiu-e Gao Le Wang[59], proposed the integration of Struts and Hibernate and showed a simple system to explain the advantages of the integration.

"Using web technologies based on college - graduate web contact management system. Form system" work done by Piatek Piatek, J. Zabierowski,W. Napieralski, A. Katedra Mikroelektroniki i Tech. Informatycznych, Politech. Lodzka, Lodz [60], discussed about J2EE technology and hibernate, spring, struts and tiles frameworks.

G.Geeta kumari, V.N.Sastry, Atul Nehi, "Grid Computing Security through Access Control Modelling", PHDThesis, March 2010. [74]

CHAPTER 3
BANKING GRID MONITORING ARCHITECTURE

3.1 Introduction:

Banking Grid is an integrated infrastructure that enables collaborative use of computational power of high end computing devices such as servers and payment switches, heterogeneous applications such as file system, databases, core banking software, security services and information storage platforms etc. owned and managed by multiple banks or service providers.

A wide-area distributed system such as a Grid requires that a broad range of data be monitored, scheduled and collected for a variety of tasks such as fault detection and performance monitoring, analysis, prediction and tuning. Gathering information about the state and performance of a large scale distributed system is essential for system management, failure detection, accounting, auditing, performance tuning and intelligent steering. Providing such information however is not easy. There were several monitoring systems created in the past that cover more or less aspects of grid monitoring but there is still no generally accepted architecture for unifying the existing approaches.

In this Chapter, we present a banking grid monitoring architecture/system in which resources are registered, classified, clustered, scheduled, allocation, discovered, monitored, and rated both within a single domain as well as multiple domains. This architecture also proposes how files, messages are transferred securely in grid environment. The grid middleware offers services that help in coupling a grid user and (remote) resources through a resource broker or grid enabled application. It offers core services such as remote process management, co-allocation of resources, storage access, information (directory), security, authentication, and Quality of Service (QoS) such as resource reservation for guaranteed availability and trading for minimising computational cost. Globus Toolkit (GT4) [3] which contains Globus Authorization Framework is used as a grid middleware between multiple domains for authentication, authorization of users and resources. Job allocation, monitoring, scheduling, execution is also discussed. This proposed architecture is build upon existing Grid Architecture which is discussed in section 1.1.2.

3.2 Requirements of Banking Grid Monitoring System

This architecture is presented to discuss the main issues in a grid environment that are scalability, interoperability, standardization, co-allocation of resources dynamically, fault tolerance, maximum resource usage, concurrent execution of jobs on a resource.

Scalability: Grids consist of large and dynamic users. So, the system mechanism must be able to deal with large number of users. Traditional SOA architecture should be used for building grid applications. The system must be scalable in terms of computing resources, data storage and supported users. Applications well-suited for exploiting a Data Grid include ASP.NET, PHP and Java JSP programs, as well as Web services developed in .NET or another technology. These are all highly scalable application architectures.

High data rate*:* If the applications to be monitored produce a lot of data in a short time this data must not lead to damage in the monitor components as well as on the compute node itself.

Interoperability: Irrespective of platform, OS; the system should run applications and access/share computing, data resources.

Standardization: In order to achieve interoperability the implementation and design should be built on latest technologies, standards which are widely accepted.

Maximum Resource Usage: Every Resource in grid environment should be utilized at maximum based on its availability. Resource idle time should be negligible .Tasks should be assigned continuously to those resources which have high configuration along with performance.

Co-allocation of resources dynamically: Complex systems require simultaneous execution of code on very high numbers of CPUs. In this context, co-allocation also means that resources for a certain task are allocated in advance. Those resources must be negotiated in advance and guaranteed to be available when the task's time slot arrives. This implies the need for a sophisticated distributed negotiation protocol which is supported by advance reservation mechanisms.

Concurrent execution of jobs: Based on the status, performance, configuration of high end computational resources jobs should be submitted and executed concurrently on various nodes in grid environment.

Fault tolerance: In grids, nodes and network failures will inevitably occur. However, to assure that an entire application will not be aborted after a single failure, distributed checkpoints and restart protocols must be used to stop and migrate the whole application or part of it.

3.3 Banking Grid Monitoring Architecture

Figure 6 below shows a complete architecture of banking grid monitoring system of single domain as well as multiple domains. Bank1 and Bank2 are two different domains connected through a grid middleware. In Bank1 domain all the resources are shared, managed and monitored locally i.e. Bank1 domain users do not require a grid to access the resources available locally, the same is the case with Bank2 domain. But when user from Bank1 domain intends to access the resource of Bank2 domain, it can be possible by a grid middleware only and vice versa.

When Bank1 users want to access/share resources of Bank2, lot of things should be kept in mind. Both domains should stick to a unique policy which contains some important information, rules, and restrictions on particular user or resource. After defining policy then they should look at what are the services they are going to share and how effectively they are sharing the services. Service Level Agreement (SLA) defines a dynamically established and managed relationship between the resource providers and resource consumers.

Monitoring/Allocation of resources locally within a single domain is not a important task. When multiple domains are involved real challenge is share /access resources globally. Some organizations may charge some price for accessing /sharing resources, for that purpose a pricing algorithm should be defined which clusters (groups) resources based on rating, category, price, utilization time and no. of times accessed etc

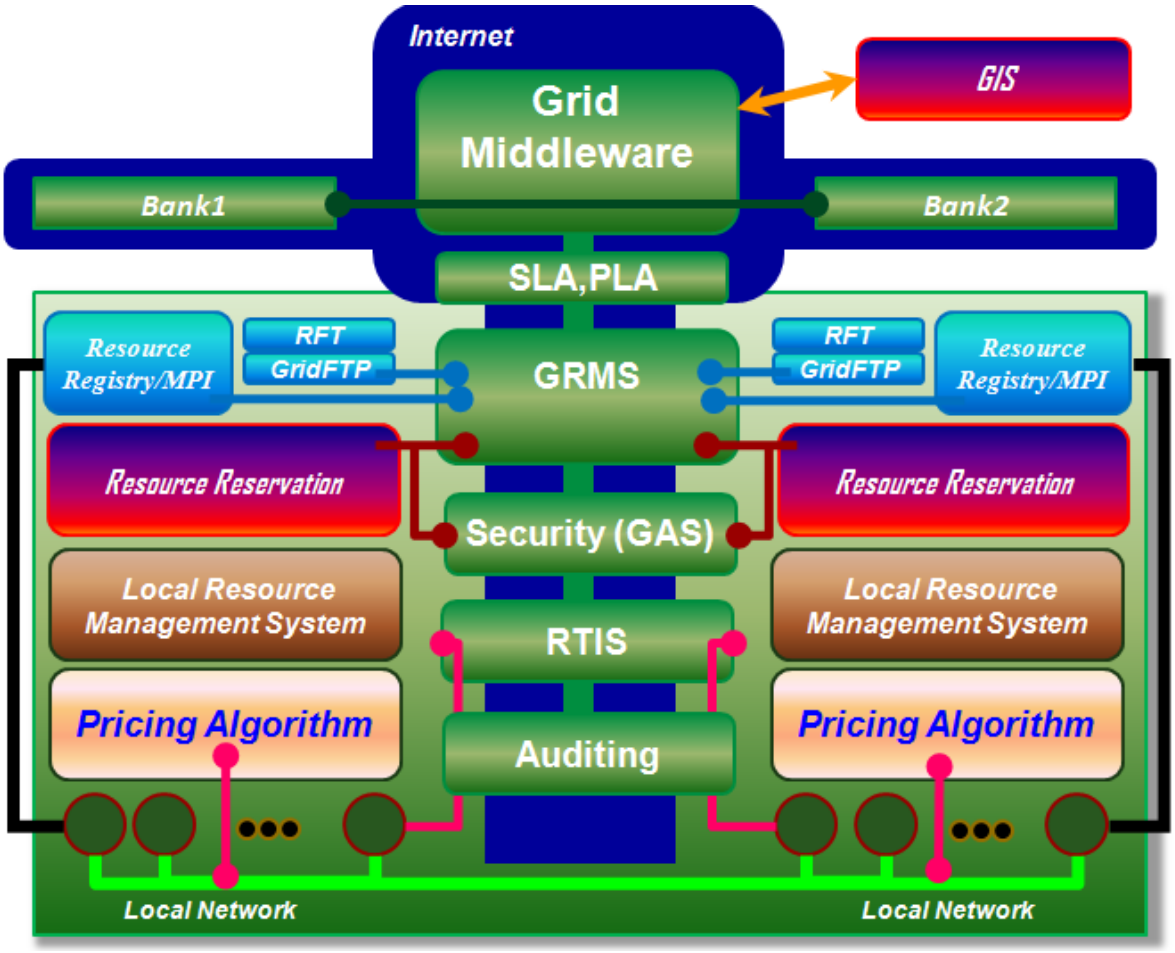


Figure 6 Banking Grid Monitoring Architecture

3.4 Overview of Each Components:

Resource Registry: All the resources are registered here with attributes such as resource id, resource name, resource type, resource cost, resource owner, resource location and resource state.

Message Passing Interface (MPI) & ProActive: It is a specification for an API that allows many computers to communicate with one another. MPI's goals are high performance, scalability, and portability. MPI remains the dominant model used in high-performance computing today. MPI is not sanctioned by any major standards body; nevertheless, it has become a *de facto* standard for communication among processes that model a parallel program running on a distributed memory system. Actual distributed memory supercomputers such as computer clusters often run these programs.

ProActive is a Java-based grid middleware for parallel, distributed and multi-threaded computing integrated with OpenDSP. It is based on ProActive, which provides a

comprehensive framework and programming model to simplify the programming and execution of parallel applications. ProActive uses by default the RMI Java standard library as a portable communication layer, supporting the following communication protocols: RMI, HTTP, Jini, RMI/SSH, and Ibis

Resource Reservation: Many algorithms, protocols have been proposed for reserving of resources in a distributed environment.

Local Resource Management System: All the resources are managed locally with in a banking domain. Resources are clustered based on category. The current load of CPUs, network connections, and other monitoring statistics are collected by the cluster and networking monitoring system, which is tightly integrated with LRMS and external middleware monitoring services.LRMS offers job submission, monitoring and reservation.

Pricing Algorithm: Some organizations may charge some price for accessing /sharing resources, for that purpose a pricing algorithm should be defined which clusters (groups) resources based on rating, category, price, utilization time and no. of times accessed etc

Grid FTP/RFT: Transferring files securely across multiple domains.

Policy Level Agreement: In collaborative applications, participants agree on certain level of secure communication based on communication policy specifications.

Service Level Agreement: A SLA defines a dynamically established and managed relationship between the resource providers and resource consumers. Both parties are committed to the negotiated terms. These commitments are backed up by organization-wide policies, incentives, and penalties to encourage each party to fulfil its obligations. For each scheduled task, a set of SLAs is signed by the administrative domain of the task owner, and by each of the provider administrative domains. The SLA describes the service time interval, and the provided QoS – resources, topology, communication, and mapping of user processes to provider's resources. SLAs are represented using the RTG model, and are stored in RTIS.

Resource Topology Information Service (RTIS): It provides information on the resource topology and availability. Information is provided by means of the Resource Topology Graph (RTG) schema, instances of which depict the properties of the resources and their interconnections. For a simpler description process, the RTG does not contain a "point-to-point" representation of the desired connections but is based instead on the communication

group's concept, which is quite similar to the MPI communicator definition. The main goals of the RTIS are to facilitate topology-aware services to discover the grid resources picture as well as to disclose information about those resources, on a "need-to-know" basis.

Auditing/Economic Services:

- A log of all actions/transactions is stored
- Total cost of resources are calculated based on pricing algorithm
- Capturing resource usage records across the administrative domains
- Assigning a cost to operations and charging the user taking into account the quality of service actually received by the user
- The ability to transfer credits from one administrative domain to another as means of payment for received services and resources
- Management of user groups' credit accounts, tracking budget, economical obligations etc.

Global/Grid Resource Management System: It is a grid meta-scheduling framework which allows developers to build and deploy resource management systems for large-scale distributed computing infrastructures at both administrative domain and grid levels.GMRS looks after dynamic resource allocation, reservation, scheduling across multiple domains. At the administrative level, the GRMS communicates with resource reservation services to expose the remote access to underlying computing resources controlled by LRMS. Administrative domain-level LRMS is synchronized with the Grid-level GRMS during the job submission, job scheduling and execution processes. At the grid level, the GRMS offers much more advanced co-allocation and topology-aware scheduling mechanisms.

Grid Information Server (GIS): Grid Information Service provides information about entities in a grid. An entity is something of value to a computational grid. Entities can be servers, such as software; server as a resource, such as a computer cluster, or exist as a person, group, or organization. It is used in several ways:

First, event producers use a GIS to advertise their existence and the types of events that they provide to event consumers. Consumers can then search the GIS to find producers that supply the information they are interested in.

Second, a location is needed to store what we are calling an *event dictionary*: a database of commonly used event types and event elements. This dictionary gives producers and consumers of events from different organizations a common "language" to use when communicating with each other. It identifies the list of authorized domains, machines, and keeps track of resource status information.

Grid Authorization System: Currently, the most common solution for mutual authentication and authorization of grid users and services is the Grid Security Infrastructure (GSI). The GSI is a part of the Globus Toolkit and provides fundamental security services needed to support grids. In many GSI-based grid environments, the user's identity is mapped to a corresponding local user identity, and further authorization depends on the internal LRMS mechanisms. The authorization process is relatively simple and static.

Moreover, it requires that the administrator manually modify appropriate user mappings in the gridmap file every time a new user appears or has to be removed. If there are many users in many administrative domains whose access must be controlled dynamically, the maintenance and synchronization of various gridmap files becomes an important administrative issue. We believe that more advanced mechanisms for authorization control are required to support dynamic changes in security policy definition and enforcement over a large number of middleware services. Grid Authorization Framework offers dynamic fine-grained access control and enforcement for shared computing services and resources. GAS can also be treated as a trusted single logical point for defining security policies.

3.5 Conclusion

In this chapter, a Banking Grid monitoring architecture is proposed which monitors all the activities in grid environment from resource sharing to job scheduling and also discussed about various components which may be useful for organizing the things in a multi domain grid environment.

CHAPTER 4
BANKING GRID PORTAL BASED ON SSH FRAMEWORKS

Abstract:
J2EE is Java, optimized for enterprise computing. J2EE is used in many business portals for its characteristics such as high stability, construct rapidly, easy to replant, maintenance and so on. Struts, Spring and Hibernate, which are based on J2EE technology, are Open Source frameworks. This chapter analyses how these three frameworks can be integrated together in building a secure banking grid portal which offers wide range of services to its users based on their roles in the organization. Users are permitted to access specified range of resources based on their profiles. GridGain is used for job execution on different nodes in grid environment.

4.1 Introduction:

The fast development of information technology and communication technology has attracted companies towards construction of E-business banking portal. E-business banking portal offers resource sharing across multiple domains and a wide range of services to its employees [61]. It changes the way of businesses which are operated from traditional approaches to an advanced and more efficient operations [62]. The major advantages of adapting E-Business banking portal are: sharing of the resources across multiple domains, single sign-on access, reduction of time and money spent and offering various services to employees based on their role and permissions granted to them [63].

Resources that can be shared between banks are file documents, payment switches, servers (database, web, application), Core Banking Software, Applications, Storage, Bandwidth, Security Services, ATM service etc.

Grid portals build upon the familiar Web portal model to offer virtual communities of users a single point of access to computational high end resources. The Web has proven to be an effective means for delivering information, business services and commercial applications to virtual communities of users. Web portals [64] like Yahoo, Microsoft or MSN, offer users a single point of access to a wide variety of resources, services and technologies. Now a day's

web portals are constructed based on J2EE, PHP and ASP.NET out of which J2EE and ASP.NET are build on three tier architecture. Since J2EE is integrated and platform independent more companies use this technology in constructing business portals. There are many Open Source frameworks which are based on J2EE technology such as Struts, webwork, jsf, echo, Spring, Doff keel, Hibernate, Seam, jpa, ibatis, GlassFish, AppFuse and so on [65] Each framework has its own importance, in this chapter we integrated framework by Struts, Spring and Hibernate. In presentation layer Struts is used, in the business logic layer web have created business objects, business services and now we are going to integrate them using Spring framework, and Hibernate is used at data layer which is an object/relational mapping tool for java environments.

4.2 Overview of SSH

4.2.1 Struts-MVC Framework:

The Apache Struts project is an open-source framework [66] sponsored by the Apache Software Foundation. It was designed for developing J2EE web applications following the Model-View-Controller (MVC) [67] design pattern. It was originally created by Craig McClanahan and in May 2000 donated to be taken over by the open-source community.Apache Struts framework is based on the MVC pattern which is a software architecture that divides the data model, user interface, and control logic from the application into three different parts. One of the advantages of using MVC is that it is extensible in different ways. For example, if we want to export this project to other platforms such as mobile devices, we only have to adapt the presentation layer according to the mobile standard.

The ***Model*** represents the data objects of the Struts application. These objects are what are going to be manipulated and presented to the user. They can be implemented as objects representations of data stored in a relational database. At this point is when Hibernate comes into action. Hibernate is an object-relational mapping (ORM) solution which permits to manipulate a database tables as Java classes.

The ***View*** presents to the user the information of the Model. The View components are mapped to a JSP file that contains Struts custom tags, HTML and JSP. All the JSP files are controlled by the Tile framework. Using tiles allows us to develop reusable presentation

components and we can classify our presentation tier to perform greater reuse of layouts, HTML, and other visual components like images and multimedia files.

The ***Controller*** defines the way that the UI responds to the user input. This component is the one in charge of manipulate the Model.

Benefits of using MVC pattern are [67]:

Reliability: We are able to change the appearance of the application without recompiling the code of the Model or the Controller components due to they are visibly separated from the presentation and transaction layers.

High reuse and adaptability: The MVC pattern allows us to present our application in different formats, all of them connecting to the same Model and Controller parts. I.e. Web browsers through HTTP protocol or mobile devices through WAP protocol.

Very low development and lifecycle costs: We can distribute development effort, with the intention that implementation changes in one module of the application do not involve changes to another.

Rapid deployment: Deployment time can be significantly reduced, because programmers focused on the Controller module can work independently of the developers responsible for the View or Model components.

Maintainability: The clearly separation of presentation and business logic become easier to maintain and modify the application.

The Model represents the data objects of the Struts application. These objects are what are going to be manipulated and presented to the user. They can be implemented as objects representations of data stored in a relational database. The View is take account for Interact with the User; Control receive user's input, then invoke Model and View to perform the user's request. The View sends updates to the Controller, the Controller updates the Model, and the View gets updated directly from the Model [67]. All the user requests to the application go through the controller. The controller intercepts the requests from view and passes it to the model for appropriate action. Based on the result of the action on data, the controller directs the user to the subsequent view, it works as following figure 7 [66]

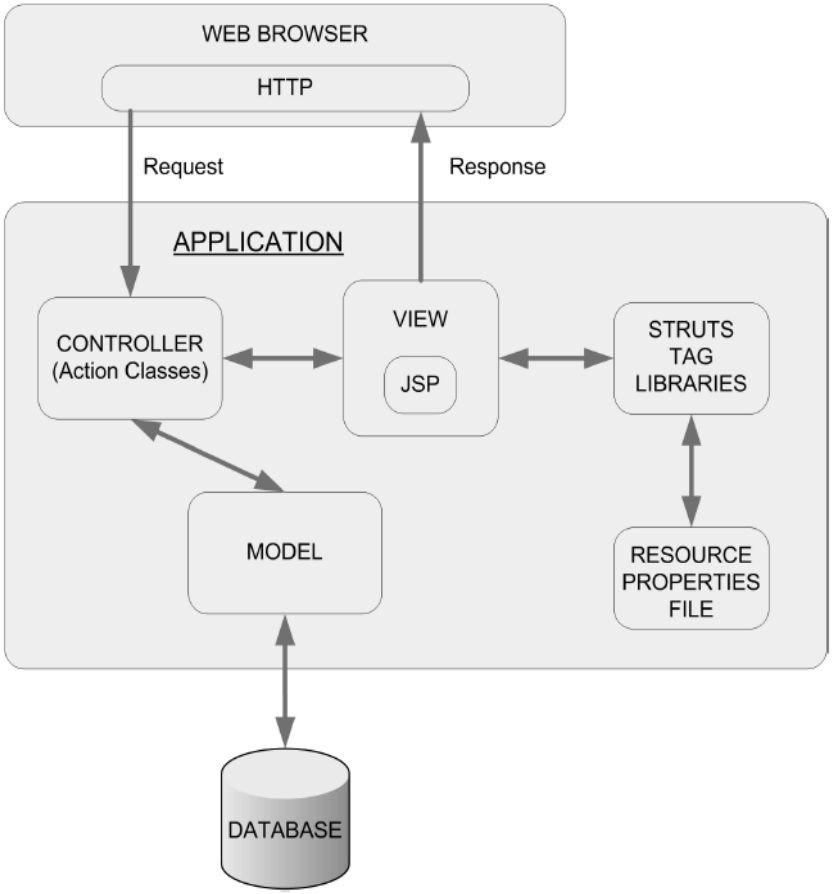


Figure7. Struts Implementation

Struts include a set of Tag libraries that allow us to develop the View component without inserting Java code into our JSP pages. These Struts libraries are: Bean Tags, HTML Tags, Logic Tags, Nested Tags, Template Tags and Tiles Tags. One of the main rules for using Struts is that the Model does not have to contain any View code as well as all JSP files do not have to contain any forward code since the flow of the application is controlled by the Struts-config.xml file. Here in below there is a snippet of the Struts-config.xml file used in application:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
    "http://struts.apache.org/dtds/struts-2.0.dtd">
    <struts>
        <package name="login" extends="struts-default">
                <action name="home">
                <result>/WEB-INF/jsp/home.jsp</result>
                </action>
                <action name="loans">
```

```
                <result>/WEB-INF/jsp/loans.jsp</result>
                </action>
                <action name="accounts">
                <result>/WEB-INF/jsp/accounts.jsp</result>
                </action>
                <action name="atm">
                <result>/WEB-INF/jsp/ATM.jsp</result>
                </action>
                <action name="upload">
                <result>/WEB-INF/jsp/upload1.jsp</result>
                </action>
        <action name="resource request">
                <result>/WEB-INF/jsp/requestresource.jsp</result>
                </action>

                <action name="login" class="loginAction" method="login">
                <result name="input">/WEB-INF/login.jsp</result>
                <result name="error">/WEB-INF/login.jsp</result>
                <result>/WEB-INF/jsp/home.jsp</result>
                </action>
        <action name= "employee" class="employeeAction">
         <result>/WEB-INF/jsp/employees.jsp</result>
                </action>
                <action name="saveEmployee" class="employeeAction"
method="saveEmployee">
                <result name="input">/WEB-INF/jsp/employees.jsp</result>
                <result name="error">/WEB-INF/jsp/employees.jsp</result>
                <result>/WEB-INF/jsp/employees.jsp</result>
                </action>
                <action name="rolesAction" class="rolesAction">
                <result name="error">/WEB-INF/jsp/roles.jsp</result>
                <result>/WEB-INF/jsp/roles.jsp</result>
                </action>
              <action name="managerUpload">
              <result>/WEB-INF/jsp/managerUpload.jsp</result>
              </action>

              <action name="employeeUpload">
             <result>/WEB-INF/jsp/employeeUpload.jsp</result>
      </action>
               <action name="saveRoles" class="rolesAction" method="saveRoles">
               <result name="input">/WEB-INF/jsp/roles.jsp</result>
              <result name="error">/WEB-INF/jsp/roles.jsp</result>
              <result>/WEB-INF/jsp/roles.jsp</result>
              </action>
  </package>
  </struts>
```

4.2.2 Spring Framework

The Spring framework [68] is a layered architecture consisting of seven well-defined modules. Each module can exist individually or unite each other. The Spring modules are built on top of the core container, which defines how beans are created, configured, and managed, as shown in Figure 8. Spring is an application framework in which spring MVC is one of the modules of spring framework.

Spring is a light containers (light-weight container), the core of which is factory Bean (Bean Factory), we need to structure M (Model). Spring uses the concept of Inversion of control (IOC) where the objects are not hard coded in java but they are injected using constructor/setter injection. On this basis, Spring provides AOP (Aspect-Oriented Programming, and the level of programming) the realization use it to provide non-regulatory environment, affirms that way affairs, security and other services for the expansion of the factory Bean more ApplicationContext help us achieve J2EE applications; DAO / ORM facilitate the realization of our database development; Web MVC Spring Web and Java Web Application provides the framework or with other popular web framework for integration.

Spring Security 2.0.4 bases its authorization decisions on user roles. In some situations, it is preferable to base authorizations on fine-grained permissions, and to treat roles as "groups of permissions". This allows more flexibility in managing the authorities, for example, allowing an individual permission to be granted to, or taken away from one or more roles, administratively, without the need for any code changes, and without the need to update individual user credentials.

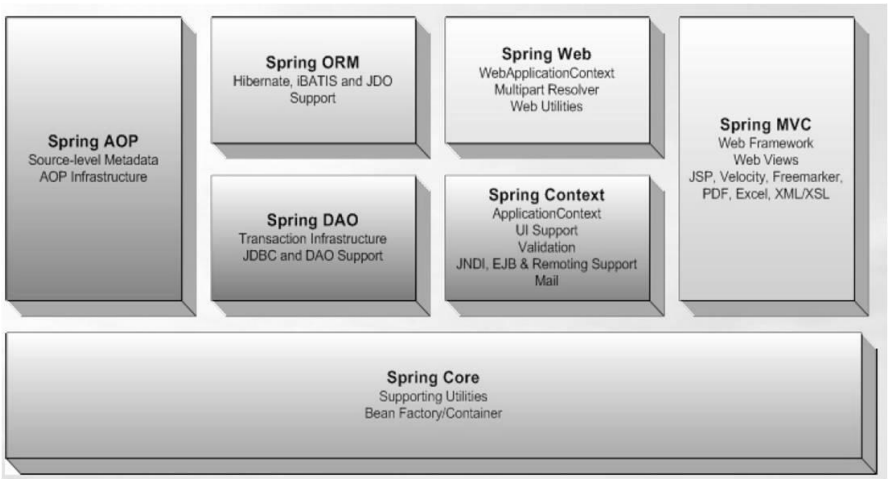


Figure 8. Implementation of Spring

4.2.3 Hibernate Framework:

Hibernate [69] is an object/relational mapping tool for Java environments. The term object/relational mapping (ORM) refers to the technique of mapping a data representation from an object model to a relational data model with a SQL-based schema. Hibernate uses three basic files to provide its services [70]. The configuration file, the mapping definition file and the primary Hibernate class used to retrieve and persist java classes.

Hibernate not only takes care of the mapping from Java classes to database tables, but also provides data query and retrieval facilities and can significantly reduce development time otherwise spent with manual data handling in SQL and JDBC, it works as following Figure 9

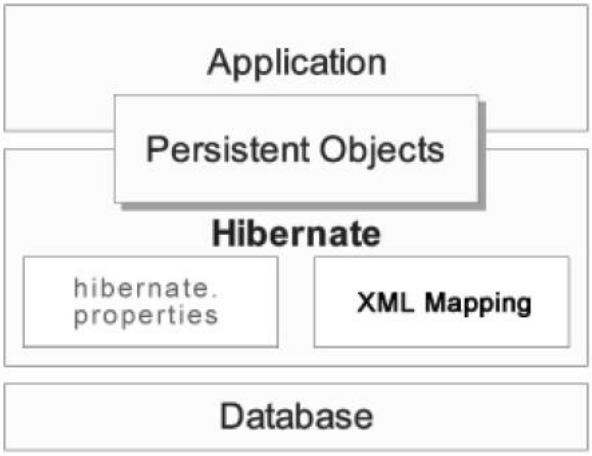


Figure 9. Structure of Hibernate

A foremost element in the project is the database. All the information is stored in this component and must be available in an effective way. User's data, roles data, resources data, actions, payment data etc are stored in database. The database is designed in accordance with the relational database model using a normalization process which eliminates duplicated data in the relational database.

Comparison between JDBC and hibernate is shown in below table:

JDBC	Hibernate
Most of well-known developers know it.	Easy and fast to learn.
Designed for small systems.	Good performance.
Hard coding effort.	Safe up to 95% of common data persistence related programming tasks.
Stored procedure usage.	Reduce common data persistence related programming tasks.

4.3 Integration of SSH Frameworks:

Struts Framework is a standard for developing well-architected Web applications and is based on MVC architecture. Stores application routing information and request mapping in a single core file, struts-config.xml. Hibernate is a powerful technology for persisting data, and it enables Application to access data from any database in a platform-independent manner. Spring is a dependency injection framework that supports IOC. The beauty of Spring is that it can integrate very well with most of the present popular technologies. Spring security is used for authentication, authorization of user along with single sign-on facility thus integration Struts, Spring and Hibernate is a very perfect pattern for developing a banking grid portal.

In this integration process, user sends request from browser to web server, the request will be then forwarded to ActionServlet of struts, and then this ActionServlet is send to bean of spring ,it checks for business logic, business service with which that action is associated and is configurable with help of IOC module. Spring allows you to define resources like a JDBC DataSource or a Hibernate SessionFactory as beans in an application context. Set up a JDBC DataSource and a Hibernate Session Factory on top of it. Spring Beans deal with the data between application and database, thus there is no need for the Application to depend on the JDBC details like opening connection, closing connection managing connection, keeping connection active and dealing with statements. By Using Hibernate, the access of database becomes object-oriented and very easy.

4.4 Construction of Banking Grid Portal with SSH Frameworks:

Banking Grid Portal is developed using struts, spring and hibernate frameworks which offers various services, Services offered to them depend on their role in the organization and to which department the user belongs to. The functionality of portal is: employee registration; online access to resources within and outside the domain; submit a job to resource; online access to database, web and application servers of other domains; editing, deleting, adding, saving resources such as files, html pages; upload/download files from web server, administrator management; employee management; request to access a resource; Loans & Deposits monitoring; track resource usage; online help; uniform access to resources globally.

Globus Toolkit (GT4) [3] is used in implementation process which creates a grid environment. Apache ant, Apache tomcat server in globus tool kit is used for deploying this web application and for submitting jobs to various nodes in organization which in turn are connect to various resources.

GridGain [71] which support spring framework is used for executing jobs concurrently on different nodes/computers in grid environment.

In this paper, Administrator management will be used as an example to describe how to realize by SSH. This module is realized by three-tiered: presentation layer, middleware layer (GT4, Business Logic/Business Service) and Data layer as Figure 10

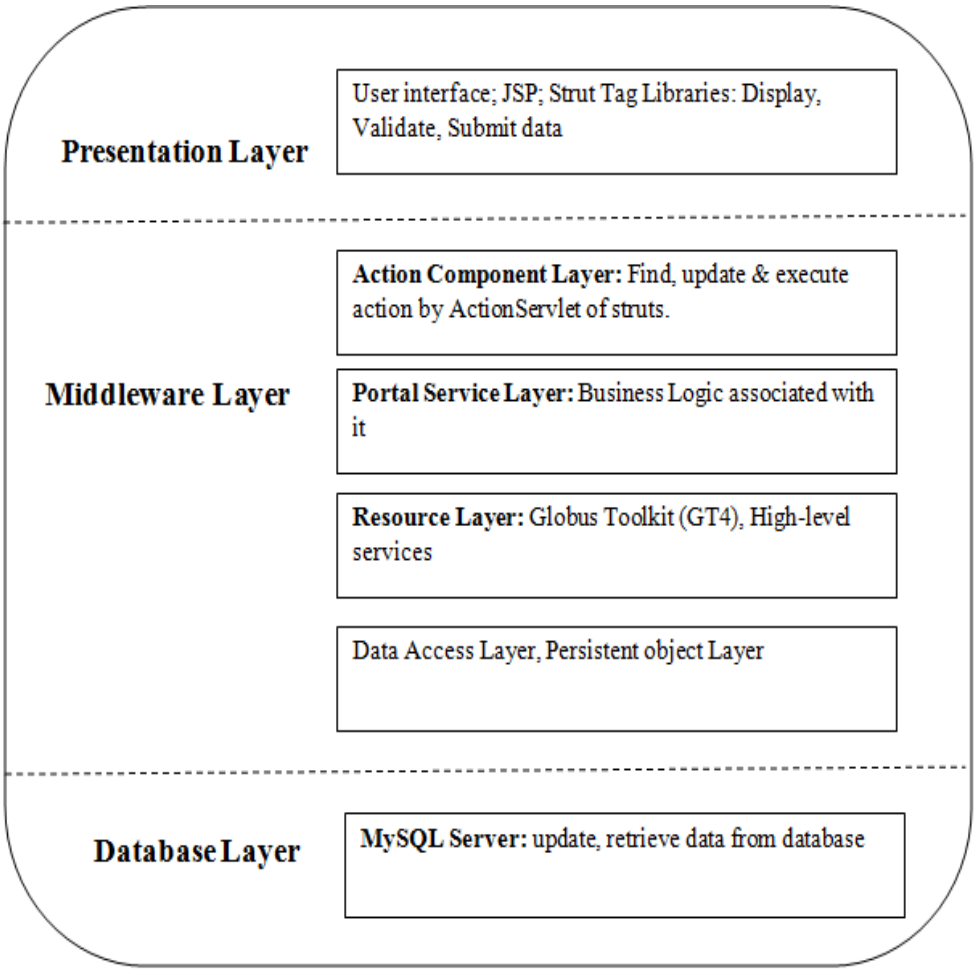


Figure 10 . Banking portal layered architecture

4.4.1 Presentation Layer:

Administrator can enter details of employee, modify details, send mail to users, access resources, download/upload files to server, edit files online and save, track resource usage etc through browser. This layer is built upon reusable Java Server Pages (JSP) [72] based user interface components, called action components. JSP collects all the information submitted

through browser and based on corresponding control, action in middleware layer service is provided for that action and reply is sent back to browser by using strut tag libraries.

Here is what Presentation layer can do:

- Managing requests and responses for a user.
- Providing a controller to delegate calls to business logic and other upstream processes.
- Handling exceptions from other tiers that throw exceptions to a Struts Action.
- Assembling a model that can be presented in a view.
- Performing UI validation.

4.4.2 Middleware Layer:

This layer is divided into 4 sub layers: action component layer, service layer, resource layer, data access & persistent object layer.

a. Action Component Layer:

This layer deals with all the actions to access a particular service.

b. Portal Service/Business Layer:

Based on business logic service is provided to users, and handles the information exchange between DAO component and Presentation layer. Business layer adds flexibility between the presentation and persistence layer so that they do not directly communicate with each other. Business layer allows interfaces for interaction with other layers and also manage dependencies between business level objects.

c. Resource Layer:

Provides support for persisting information about resources and tasks performed by users on the Grid. This service is deployed in grid node using apache ant which is present in Globus Toolkit (GT4).

d. Data Access Object (DOA) & Persistent Object Layer (PO):

i. DAO Layer: It encapsulates the functions of employee data query, modification, add and deletion. And it handles the information exchange between Service tier and PO tier.

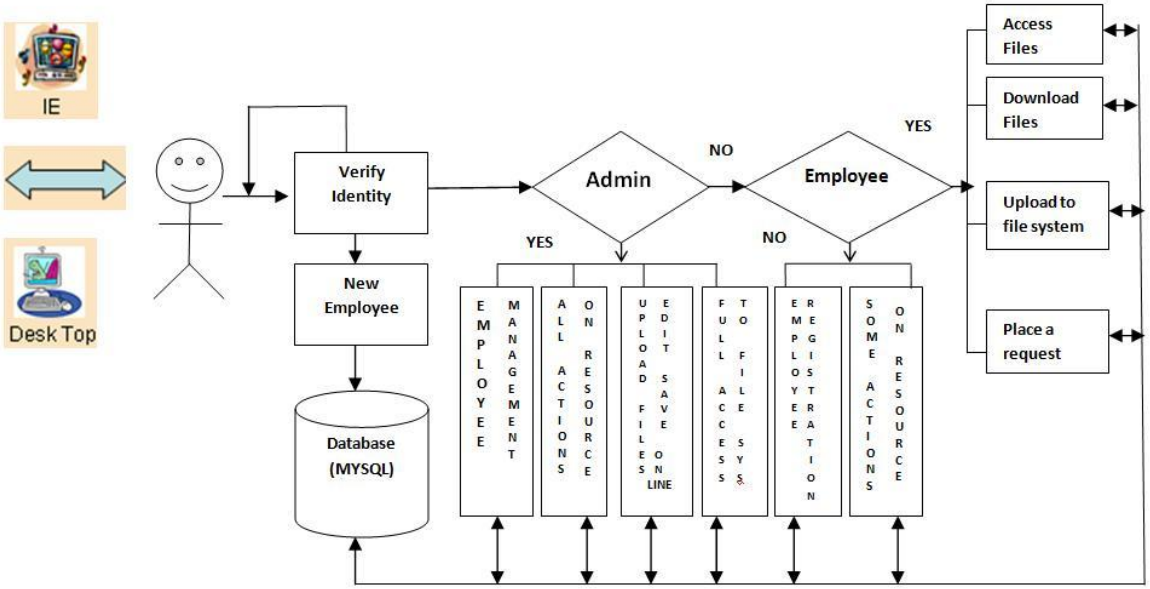
ii. PO Layer: The relational database table which stores, updates and deletes user information are mapped to data objects by the object/relational mapping tool of the Hibernate. Hibernate persistent objects are based on plain-old Java objects and Java collections.

4.4.3 Database Layer:

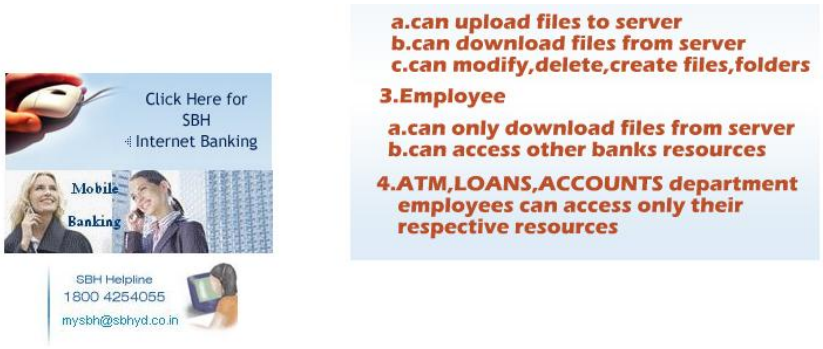
In the implementation part, MySQL 5.0 database server is used to store, update and retrieve information. This layer keeps data neutral and independent from application servers or business logic. MySQL is a multithreaded, multi-user, SQL Database Management System. One of the advantages is that MySQL is available as free software under the GNU General Public License (GPL), decreasing the cost of the final product. MySQL supports transactions, procedures and SSL which provides secure communications, an essential feature in e-Commerce.

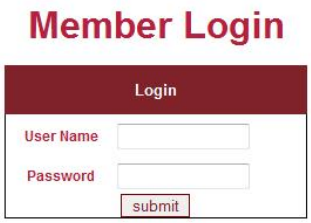
4.5 Screenshots

Portal Functionality:



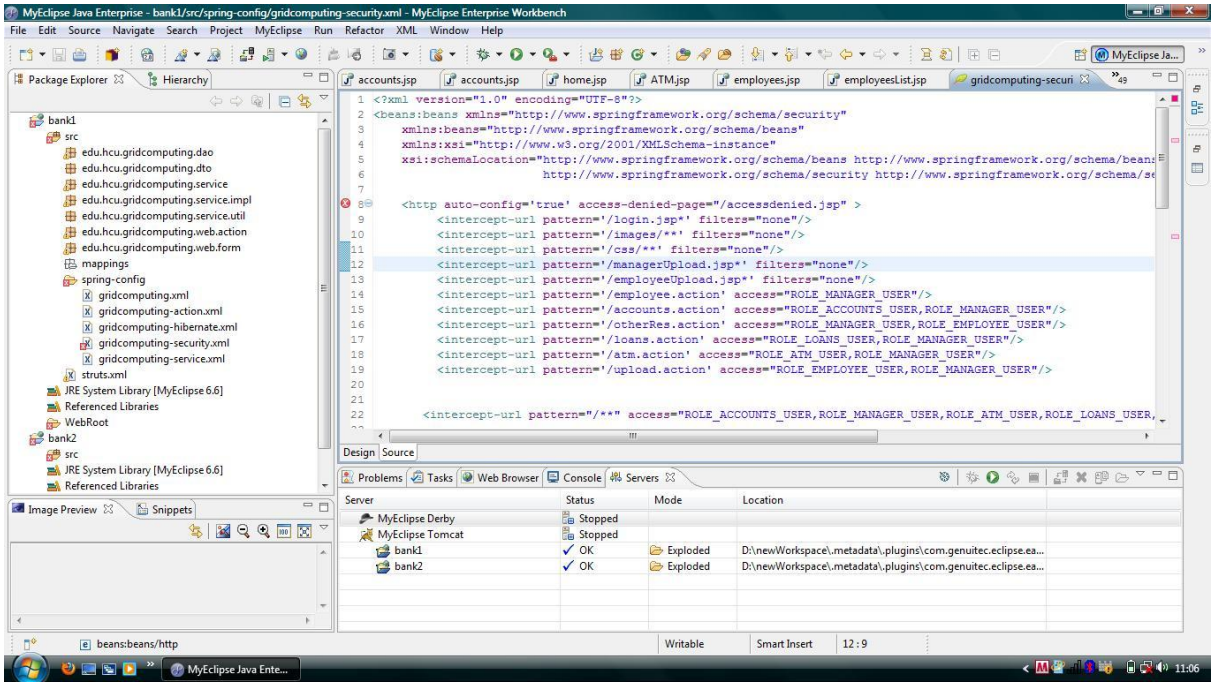
User Login Page:



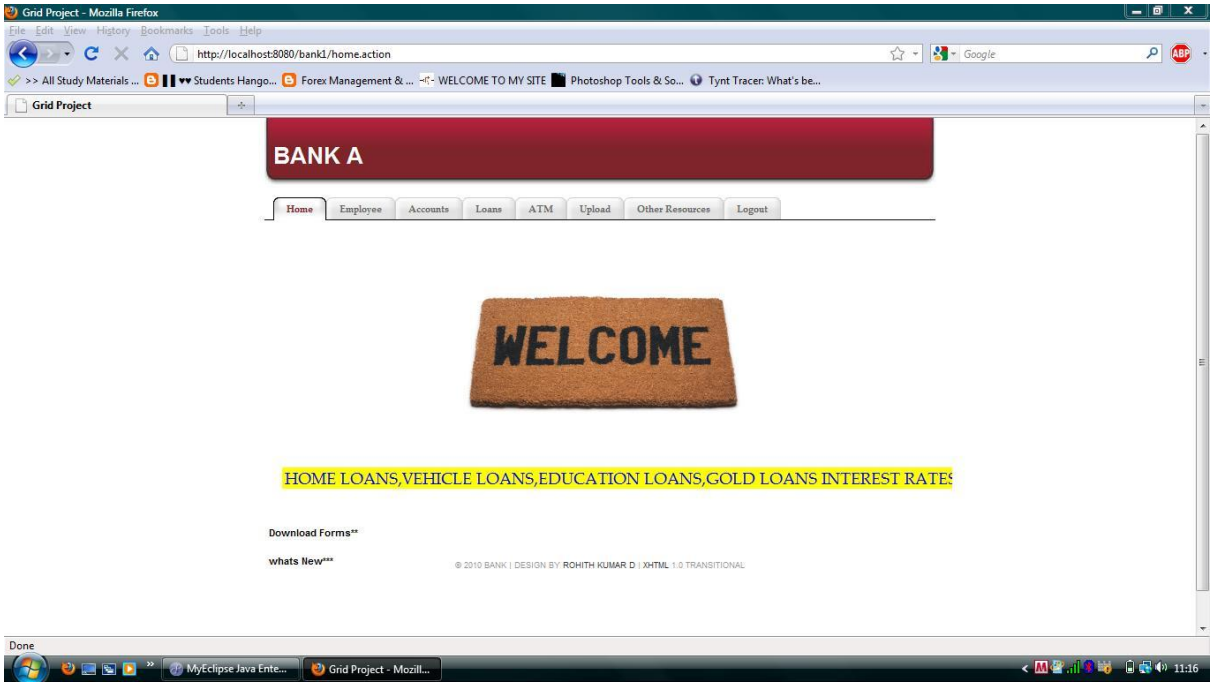


IOME LOANS,VEHICLE LOANS,EDUCATION LOANS,GOLD LOANS INTEREST RATE

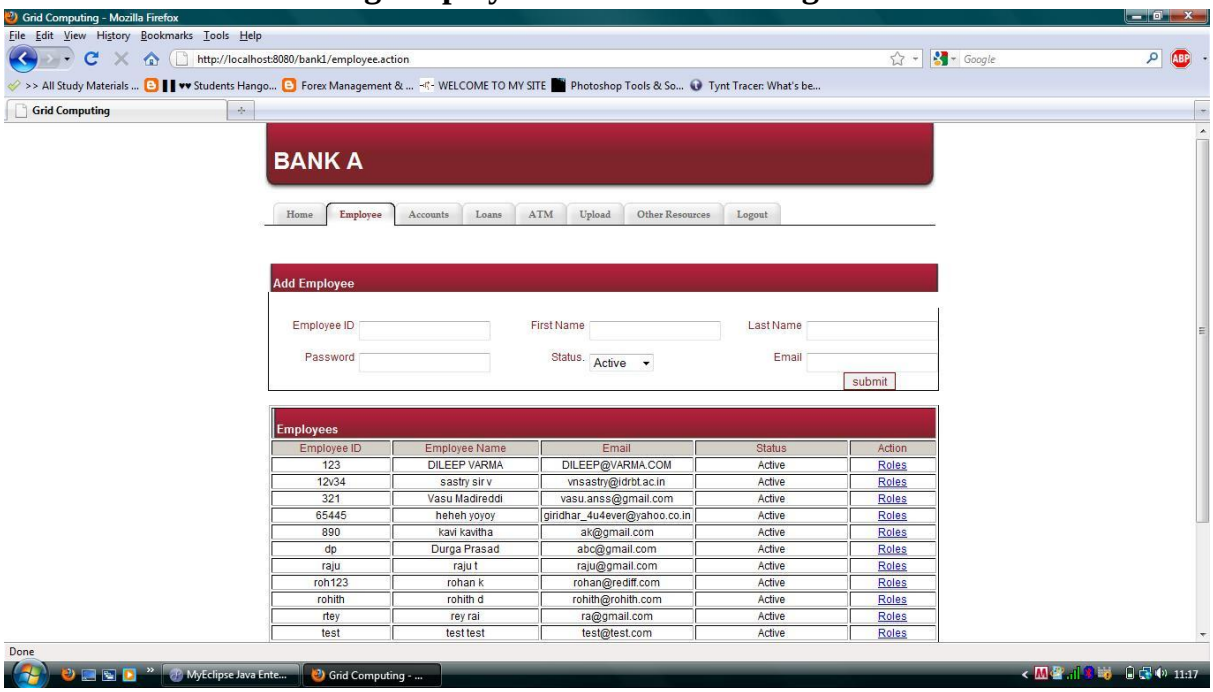
Working Environment: MyEclipse



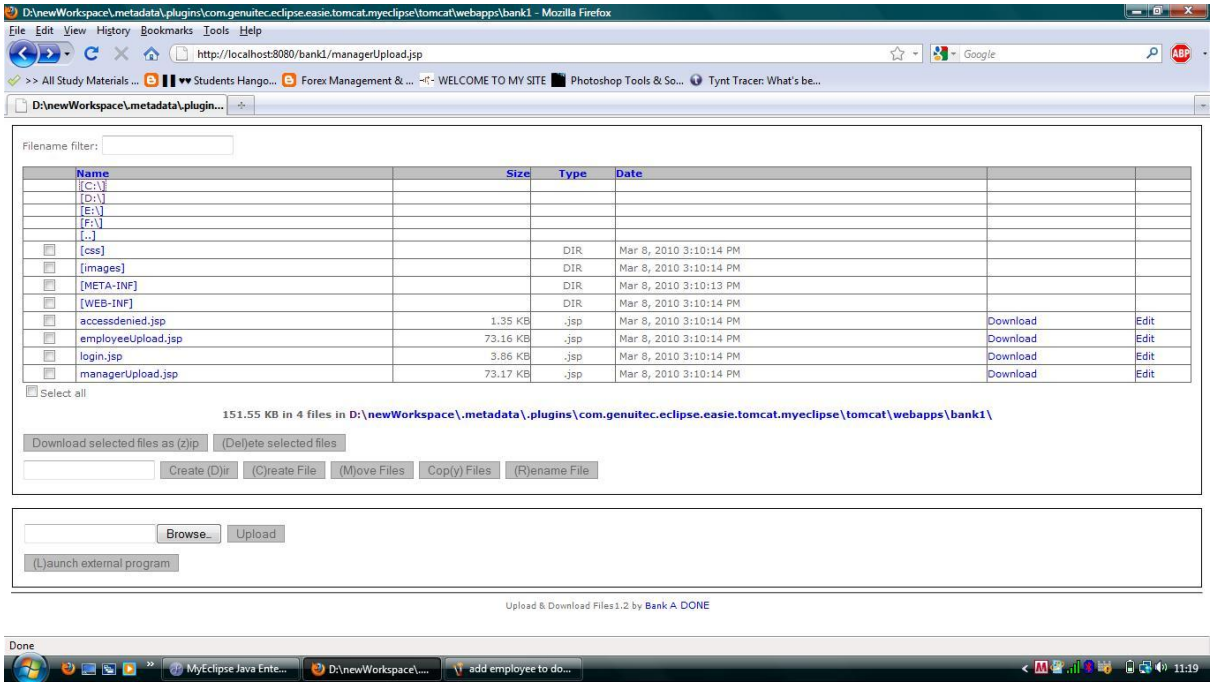
User Home Page:



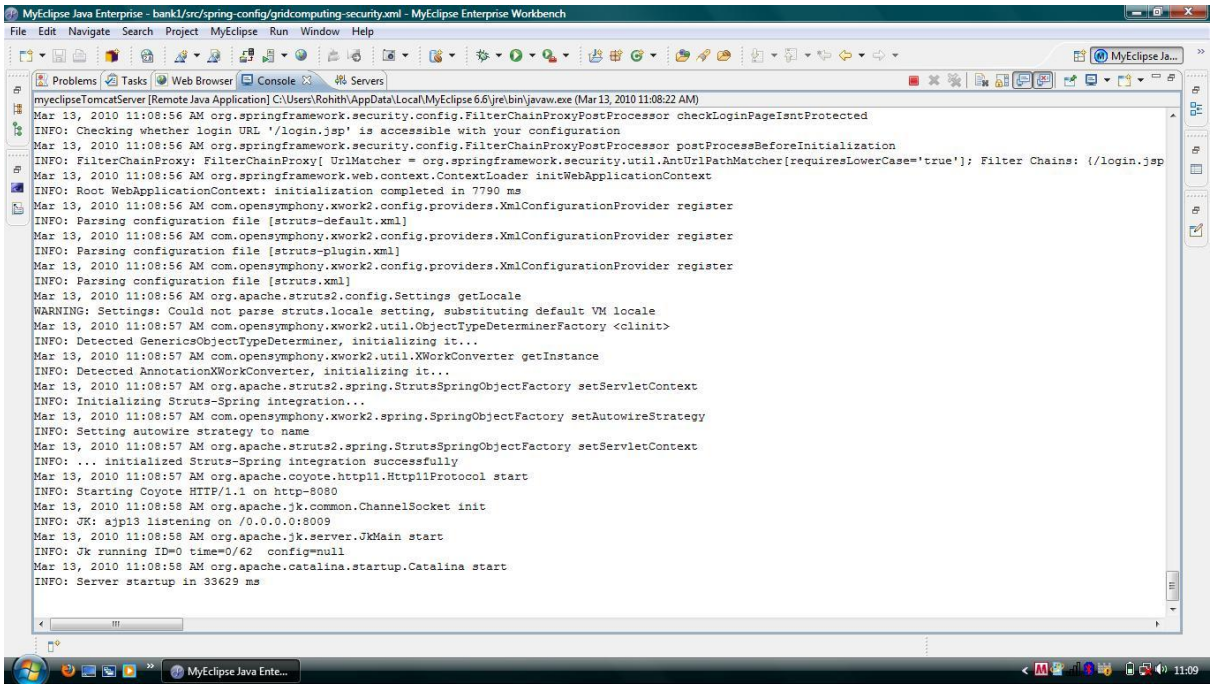
Adding Employee to domain & Assign Roles:



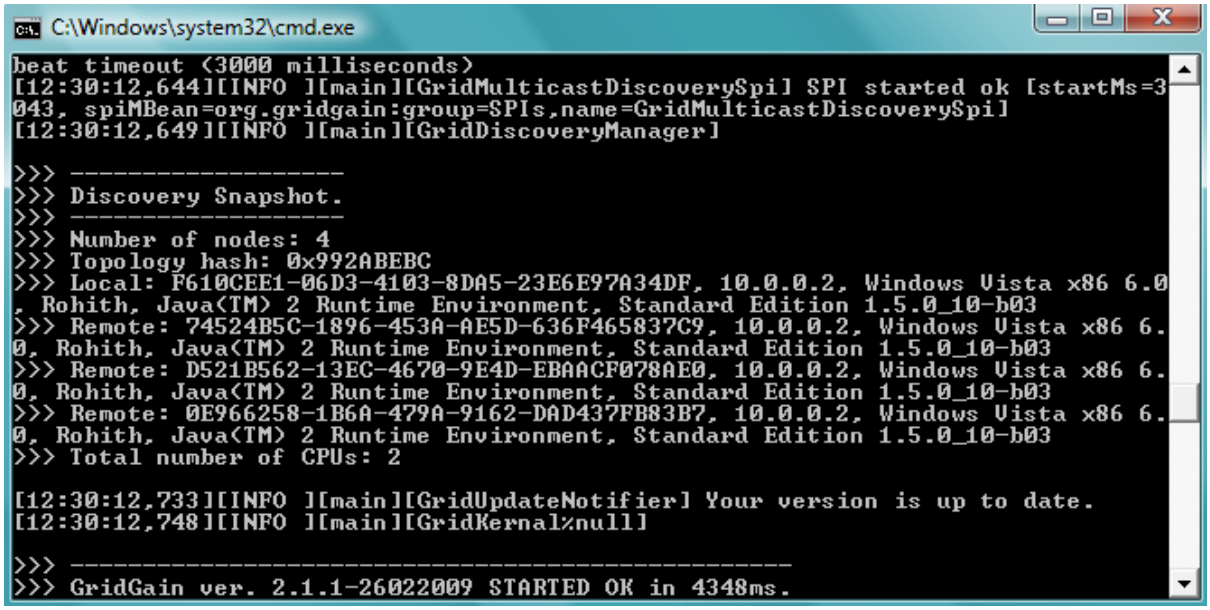
Upload & Resource Page:



Server Status:



GridGain:



4.6 Conclusion

We developed a banking grid portal using SSH framework which is based on J2EE three-tier architecture. In this portal banking grid services are monitored, managed and resources are accessed online based on user role and action on a particular resource. In the portal system, the presentation, the application processing and the data management are logically separate processes. Thus it satisfies the demands of the business and achieves the goal of expansibility and maintenance by designing the clear, rational structure of system architecture. Apache tomcat and Globus toolkit which is open source is used in developing this portal. Three level security is provided through portal to access/share resources of other domains.

CHAPTER 5
RBAC XACML IN GRID ENVIRONMENT

Abstract:

Once the authorization mechanism is defined, the next step is to write policies for different set of users based on their roles and persmissions.In this chapter we will explain how a XACML RBAC policy can be written with the help of an example which contains 5 roles. The XACML RBAC is an extension to XACML specification 2.0 and defines a set of policies for creating roles, role permissions, role assignments and separation of duty policies. SUNXACML API is used for implementation.

5.1 Introduction:

A large number of dynamic grid users and resources make the overall access control Inefficient and cost surge. Therefore, there is a need to reduce the complexity of authorization management, reduce administrative overhead. For these reasons to improve grid security we use XACML and Role Based Access Control (RBAC) mechanisms for enterprise information grid.

To address some of the problems associated with role-based access control in XACML there is a specific XACML-RBAC profile [73]. The profile has been defined in the XACML-RBAC specification as an OASIS approved standard. The profile allows one to define Roles, Permissions, References from roles to permissions, and roles enabling and assigning policies.

The policies specified in this profile can answer three types of questions:

1. If subject X has roles R1, R2….Rn enabled, can subject X access carry an action ‘a’ to a given resource?
2. Is subject X allowed to have a specific role R1 enabled?
3. If subject *X* has multiple roles *R1, R2…. Rn* enabled, does that mean the subject *X* will also have permissions associated with role *Ri (where Ri is* junior to any of the roles R1, R2,…Rn)

The policies specified in this profile assume all the roles for a given subject have already been enabled at the time an authorization decision is requested. The set of roles a subject may

have is handled by a Role Enablement Authority, but not by the Policy Decision Point (PDP) which makes the actual authorization decision. So in terms of architecture this requires an additional component that carries out an authentication process first, in order to determine whether a user is allowed to carry a role.

Assuming a subject has a valid role, the specification suggests two special types of PolicySets to describe the permissions of various roles. They are called Role Policy Set (RPS) and Permission Policy Set (PPS). A Role Policy Set is only applicable to a specific role, i.e. each role should have its own RPS. A Permission Policy Set specifies the actual permissions for a role and is referenced by the RPS.

A reference is used to allow roles to inherit its other's permissions. So in order to allow one role to possess all the permissions of another role, a reference is included within the junior role's PPS, pointing to the senior role's PPS. This is simply an inclusion mechanism to support role inheritance. The main benefit of using such mechanism is the decoupling or the role and the permissions policy set. This also provides some flexibility for carrying out changes of association between a role and its permission sets. A change of reference will suffice without requiring change to either the role or the permission sets.

This profile also addresses the issues of assignment of various roles attributes to users although the enabling of those attributes are outside the scope of the XACML PDP. Role Enablement Authorities are implemented to perform these functions, which answers the question "Is subject X allowed to have role Ri enabled?" .It however doesn't answer "Which set of roles is subject X allowed to have enabled?". The profile assumes that the presence of a role attribute in the XACML request for a given user is a valid assignment at the time the access decision is requested. Therefore to verify the link between a user and the user's role requires some additional authentication.

5.2 eXtensible Access Control Mark-up Language (XACML)

XACML is a widely used standard for representing attribute-based access control policies. The XACML specification [39] defines both a language model and a data-flow model. The XACML language model defines the following main components to represent policies:

(1) A <PolicySet> contains a set of access control policies or other policy sets.

(2) A <Policy> represents an access control policy described through a set of rules.

(3) A <Rule> represents an access rule or permission. The effect of a rule is either permit or deny.

An XACML <PolicySet>, <Policy> or <Rule> may contain a <Target> element. A <Target> element specifies a set of subjects, resources, actions and environments to which the <PolicySet>, <Policy> or <Rule> is applied shown in figure 11

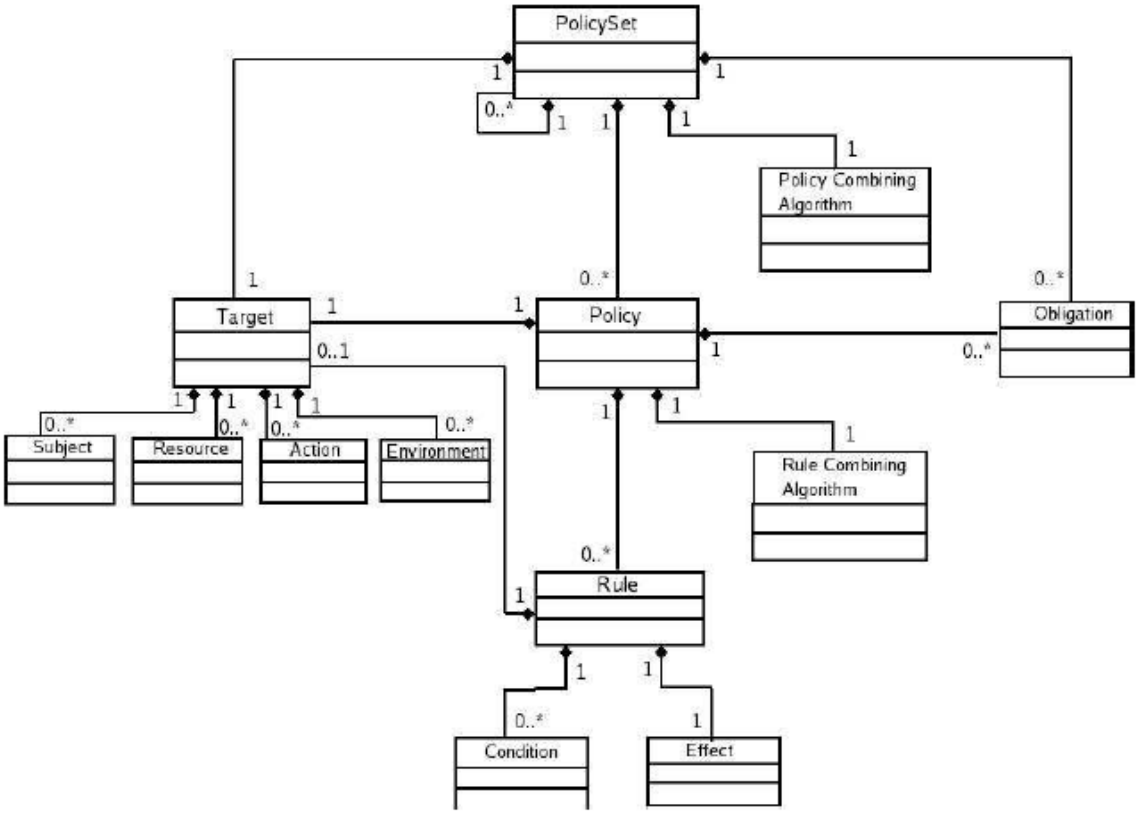


Figure 11. XACML Policy language Model

5.3 RBAC

Role-based access control has gained in popularity in recent years, with one of the principal reasons for this being that the approach offers the benefits of greatly simplified policies by exploiting the natural hierarchies of the workplace. In RBAC, permissions are associated with roles, and users are assigned appropriate roles, thereby acquiring the roles' permissions .In a virtual organization which includes large number of users who are divided into several groups, each with different level of access. A role has certain privileges associated with it. When a VO role is mapped to a local role, it will specify the privileges a user can have; for example, access to a specific table of a database or a file.RBAC Structure is shown in fig 12

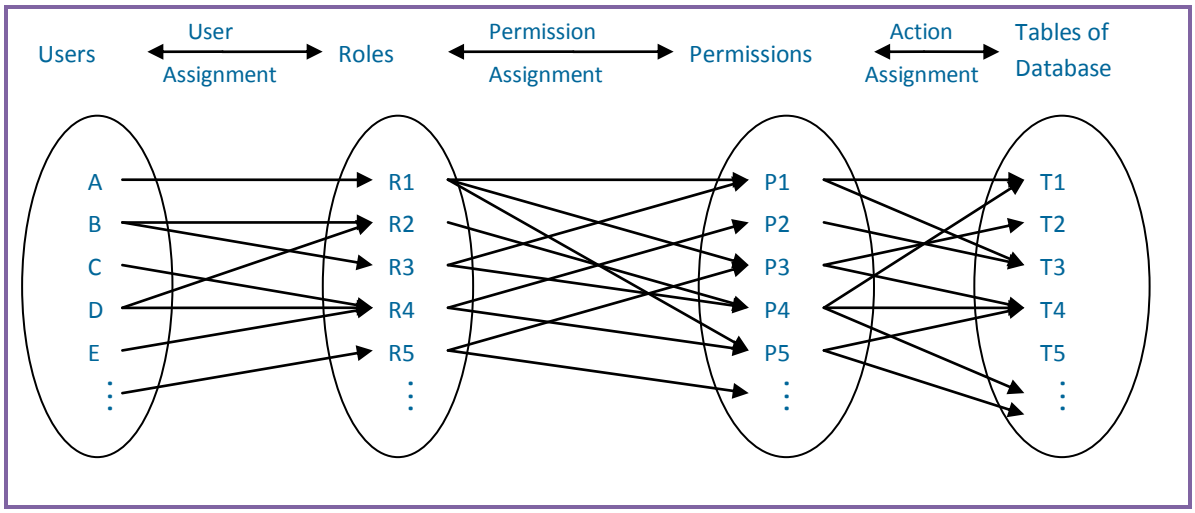


Figure 12. RBAC Structure

In VOs, users may be assigned specific tasks, and there may be constraints related to the execution of those tasks. For example, a user may have access to data only during certain days of the week, or certain tasks may be considered mutually exclusive for a user; i.e., any two or more tasks cannot be executed at the same time.

The typical identity-based authorization used today is not scalable because authorization information should be maintained for each user. In RBAC, authorization information is associated with roles, not with individual users. It has been shown that the cost of administering RBAC is proportional to $U+P$ per role, where U is the number of individuals in a role and P is the number of permissions required by the role, while the cost of associating users directly with permissions is proportional to $U \times P$.

In certain instances, a user may wish to delegate only a subset of its rights to an application to act on its behalf. This requirement can usually arise in systems where a limited trust relationship is established between entities. More information about RBAC is discussed in section 1.2.1

5.4 XACML RBAC Example:

Consider a banking domain with 5 users 4 roles; with the roles being Manager, Clerk, Officer, Financial Analyst and Some users have multiple roles as indicated in the sets below.

User 1 -> Manager

User 2 -> Officer

User 3 -> Clerk

User 4 -> Manager, Financial Analyst

User 5 -> Clerk, Officer

Relationship between Roles:

1. Officers and Clerks are more juniors than Manager;
2. Officers and Clerks are more juniors than Financial Analyst
3. Financial Analysts are junior than Managers
4. Clerks are junior than Officers
5. Officers are junior than Financial Analysts

Permissions for each role:

1. Manager can:

   a) Add, Delete, Edit, Modify employee, clerk, officer details.
   b) Have access to all the resources(files, Servers)
   c) Upload files to server ,Download files from server, Edit Files online and save them

2. Officers, Clerks can:

   a) Read Files
   b) Download Files from server

3. Financial Analyst can:

   a) Write ,Read Files
   b) Upload Files to server.
   c) Download Files in between 5Am – 7Pm

4. You need to be both Manager and a Financial Analyst to add, modify, and delete employee, clerk, and officer details

The 1st step is to determine the list of resources as well as the set of actions that can be applied on those resources. This will result in a set of pairs (action, resource) as shown in the fig 12. Each pair is given a reference identity. In addition they can also be grouped to form sets of permissions. Again a reference is also used to identify that set. The process results in a list of *references* that are used to identify either individual action of groups of such actions.

The second step is to determine the roles. In XACML RBAC these are expressed in the documents we call role policy sets. Third step is to define permissions with respect to a role this contains set of rules, describing what a role can or cannot do. Fourth and last step is that of agreeing on separation of duty which is achieved by defining another policy Set. The purpose of this policy is simply to prevent an entity that possesses conflicting role attributes from gaining any access to resources. For example we may want to prevent a user to access a resource if he also carries another role.

5.5 XACML-RBAC Implementation in Grid Environment (GT4):

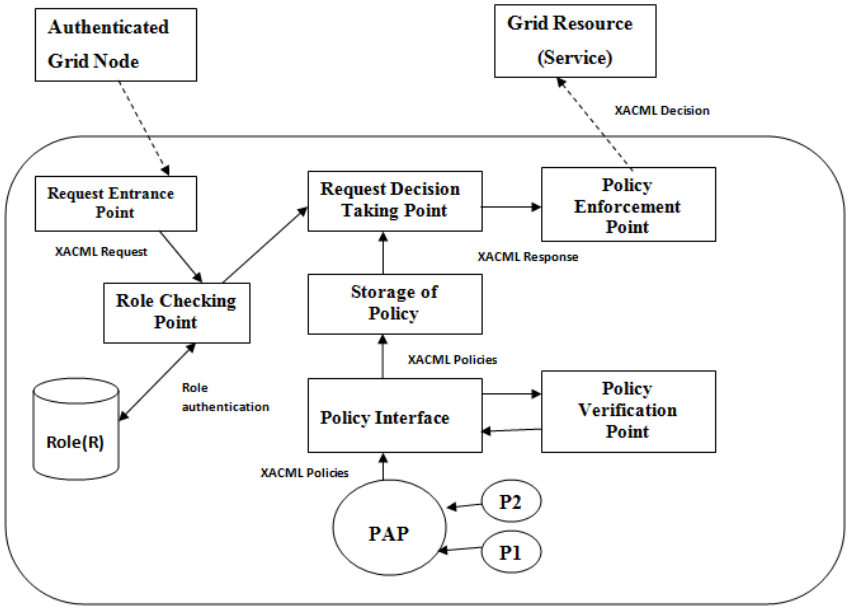


Figure13.RBAC XACML Architecture

XACML Policy Framework: It is designed as a separate service running on remote system. It accepts the requests from multiple authorization modules of the middleware and makes access decisions. Policies are specified using XACML. These policies have been placed on the same system where the XACML framework resides.

Globus container: This component contains all the services and accepts requests from the grid clients and runs on a separate system

Database: It contains information about user-role, role-delegation etc. The Policy Enforcement Point (PEP) of the policy framework queries the database for user, role and permission details

LDAP (Lightweight Directory Access Protocol) server: It runs on a separate system and accepts request from the LDAP clients. It contains the complete details of users and resources in the organization. It also responds to the queries from the Policy Enforcement Point (PEP) for additional user attributes

Policy Enforcement Point (PEP):

- ✓ XACML Request is generated
- ✓ Protect the resources
- ✓ Queries the LDAP server for additional user attributes
- ✓ Forms request based on subject, action and resource attributes

Policy Development Point (PDP):

- ✓ Receives and examines the request
- ✓ Determines whether access should be granted or not
- ✓ Retrieves applicable policies
- ✓ Returns access decision to PEP

SUNXACML API:

It contains all the interfaces for PDP, PEP, Policy Finder, Response Context, rule/policy combining algorithm etc. [40]

MySql*:* Database for storing user-role, role-permissions

Role checking point serves as an authentication point, its purpose is to check whether the user with a specified role is allowed to carry that role or not.

Grid Node receives the request from user and that request is forwarded to Request entry point (REP). Policy Administration Point defines roles, persmission policies along with role assignment and separation of duties among users, he is owner of all policies and policy sets

A role R requests access to a resource (resources are expressed as URIs) several services coordinate the process of expressing a request and providing a response. Here the request is in the form of Role (action, resource) instead of user (action, resource)

The Request Entrance Point service provides an interface that accepts requests in the form of Role (resource-URI, action) and translates these to XML documents which complies with the XACML schema for requests. As it was stated earlier the initial request is send to the role enabling point to be authenticated before it is directed to the PDP.

The Request Decision Point is the service that actually makes the decision whether a request should be granted. Upon receipt of an XACML request it retrieves the relevant policies documents from the database. At this point the evaluation takes place. This is done through a collection of Combining Algorithms. There are Policy Combining Algorithms and Rule Combining Algorithms. When a decision is reached a XACML response document is created which contains the decision and is forwarded to the Policy Enforcement Point. This is the service that parses the XACML response, retrieves the response and forces the decision on role R.

Policies & Request Formats with Employee and Manager Roles:

Manager Policy:

Table below shows a sample policy in XACML v3.0 which specifies that users with the Manager role have read and write permissions on the File resource.

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy xmlns="urn:oasis:names:tc:xacml:3.0:schema:os"
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:policy1"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-
overrides">
<Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:rule1" Effect="Permit">
<Target>
<AnyOf>
<AllOf>
<Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
<AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">Manager</AttributeValue>
<AttributeDesignator Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-
subject"
AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:role"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
</Match>
```

```
</AllOf>
</AnyOf>
<AnyOf>
<AllOf>
<Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
<AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">File</AttributeValue>
<AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:target-namespace"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
</Match>
</AllOf>
</AnyOf>
<AnyOf>
<AllOf>
<Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
<AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">read</AttributeValue>
<AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
</Match>
</AllOf>
<AllOf>
<Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
<AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">write</AttributeValue>
<AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
</Match>
</AllOf>
</AnyOf>
</Target>
<ObligationExpressions>
<ObligationExpression
ObligationId="urn:oasis:names:tc:xacml:example:obligation:email"
FulfillOn="Permit">
<AttributeAssignmentExpression
AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:mailto">
<AttributeSelector Category="urn:oasis:names:tc:xacml:3.0:attribute-
category:resource"
RequestContextPath="md:record/md:employee/md:employeeContact/md:email"
    DataType="http://www.w3.org/2001/XMLSchema#string"/>
</AttributeAssignmentExpression>
<AttributeAssignmentExpression
AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:text">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string" >
Your File has been accessed
</AttributeValue>
```

```
</AttributeAssignmentExpression>
<AttributeAssignmentExpression
AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:text">
<AttributeDesignator
 Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
</AttributeAssignmentExpression>
</ObligationExpression>
</Rule>
</Policy>
```

Employee Policy:

Table below shows a sample policy in XACML v3.0 which specifies that users with the

Employee role have read permission on the File resource only from 9:00 AM to 7:00 PM

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy xmlns="urn:oasis:names:tc:xacml:3.0:schema:os"
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:policy1"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-
overrides">
<Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:rule1" Effect="Permit">
<Target>
<AnyOf>
<AllOf>
<Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
<AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">Employee</AttributeValue>
<AttributeDesignator Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-
subject"
AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:role"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
</Match>
</AllOf>
</AnyOf>
<AnyOf>
<AllOf>
<Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
<AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">File</AttributeValue>
<AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:target-namespace"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
</Match>
</AllOf>
</AnyOf>
<AnyOf>
```

```
<AllOf>
<Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
<AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">read</AttributeValue>
<AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
</Match>
</AllOf>
<AllOf>
</Target>
<Condition>
<Apply FunctionId="&function;and">
<Apply FunctionId="&function;time-greater-than-or-equal">
<Apply FunctionId="&function;time-one-and-only">
<EnvironmentAttributeDesignator AttributeId="&environment;current-time"
DataType="&xml;time"/>
</Apply>
<AttributeValue DataType="&xml;time">9h</AttributeValue>
</Apply>
<Apply FunctionId="&function;time-less-than-or-equal">
<Apply FunctionId="&function;time-one-and-only">
<EnvironmentAttributeDesignator AttributeId="&environment;current-time"
DataType="&xml;time"/>
</Apply>
<AttributeValue DataType="&xml;time">19h</AttributeValue>
</Apply>
</Apply>
</Condition>
</Rule>
</Policy>
```

Request Structure:

Manager Request:

```
<Request xmlns="urn:XACML3.hcu.ert.in">
<Attributes Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
<Attribute AttributeId="urn:oasis:names:tc:xacml:3.0:attribute:role">
<AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">Manager</AttributeValue>
</Attribute>
</Attributes>
<Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
<Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
includeInResult="true">
<AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">FILE</AttributeValue>
</Attribute>
```

```
</Attributes>
<Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
<Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id">
<AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">read</AttributeValue>
</Attribute>
</Attributes>
</Request>
```

Employee Request:

```
<Request xmlns="urn:XACML3.hcu.ert.in">
<Attributes Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
<Attribute AttributeId="urn:oasis:names:tc:xacml:3.0:attribute:role">
<AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">Employee</AttributeValue>
</Attribute>
</Attributes>
<Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
<Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
includeInResult="true">
<AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">FILE</AttributeValue>
</Attribute>
</Attributes>
<Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
<Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id">
<AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">Write</AttributeValue>
</Attribute>
</Attributes>
</Request>
```

5.6 Conclusion:

In this chapter, we have written policies based on XACML-RBAC profile 1.0 and explained with an example. Also proposed XACML RBAC architecture for single domain banking grid enterprise where authorization decision is taken to give access/deny a resource. Policy database for various roles is also maintained and also investigated the extent to which obligations, XACML policies and role-based administrative models can be used to manage XACML RBAC policies.

CHAPTER 6
CONCLUSIONS & FUTURE WORK

Conclusions

Banking Grid is an integrated infrastructure that enables collaborative use of computational power of high end computing devices such as servers and payment switches, heterogeneous applications such as file system, databases, core banking software, security services and information storage platforms etc. owned and managed by multiple banks or service providers. When the resources are shared among multiple domains, monitoring the resources become the most important issue for efficiently using and managing recourses across domains. Hence, in this project I have proposed a Banking Grid monitoring architecture which monitors all the activities in grid environment from resource sharing to job execution/scheduling and also stated the requirements for such architecture.

Also a Banking Grid Portal is developed through which all the authenticated grid users based on their roles; permissions can share/access resources online and can also make use of various grid services. Three level security is provided to access resources.SSL encryption may be used for certain modules, it is left to future work

Also proposed XACML RBAC architecture for single domain banking grid enterprise where authorization decision is taken to give access/deny a resource. Policy database for various roles is also maintained and also investigated the extent to which obligations, XACML policies and role-based administrative models can be used to manage XACML RBAC policies.

Future Work:

In role management, we did not consider hierarchical relationship among roles and policies. Also separation of duty including both static and dynamic and special constraints like cardinality constraints implementation needs to be addressed. And also investigate to what extent SAML and XACML can inter-operate to support Attribute-based role assignment. As Grid involves several organizations, the distributed storage and administration of the XACML policies has to be supported. As users change quite frequently in dynamic grid environments, storing the XACML policy in xml file is not a correct method to deal with policies. In future, as more organizations will follow XACML standard it is important to address these issues and design a new system that can create, store, retrieve and manage the XACML policies efficiently.

BIBLIOGRAPHY

[1] I. Foster, C. Kesselman and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," Int'l Journal of Supercomputer Applications and High-Performance Computing, vol. 15, no. 3, pp. 200–222, 2001.

[2] Rajkumar Buyya, Grid Computing Info Centre: http://www.gridcomputing.com

[3] Ian Foster and Carl Kesselman, Globus: A Metacomputing Infrastructure Toolkit, *International Journal of Supercomputer Applications*, 11(2): 115-128, 1997.

[4] Steve Chapin, John Karpovich, Andrew Grimshaw, The Legion Resource Management System, *5th Workshop on Job Scheduling Strategies for Parallel Processing*, April 1999.

[5] Henri Casanova and Jack Dongarra, *NetSolve:* A Network Server for Solving Computational Science Problems, *Intl. Journal of Supercomputing Applications and High Performance Computing*, Vol. 11, No. 3, 1997.

[6] Hidemoto Nakada, Mitsuhisa Sato, Satoshi Sekiguchi, Design and Implementations of Ninf: towards a Global Computing Infrastructure, *FGCS Journal*, October 1999.

[7] Fran Berman and Rich Wolski, The AppLeS Project: A Status Report, *8th NEC Research Symposium*, Berlin, Germany, May 1997. http://apples.ucsd.edu

[8] Rajkumar Buyya, David Abramson and Jon Giddy, Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid, *4th Intl. Conf. on High Performance Computing in Asia-Pacific Region (HPC Asia 2000)*, China.

[9] Spyros Lalis and Alexandros Karipidis, JaWS: An Open Market-Based Framework for Distributed Computing over the Internet, *IEEE/ACM International Workshop on Grid Computing (GRID 2000)*, Dec. 2000. http://roadrunner.ics.forth.gr:8080/

[10] Ian Foster. What is the Grid? A Three Point Checklist. Grid Today, 2002.

[11] I. Foster and R. L. Grossman, "Data Integration in a Bandwidth-Rich World,"Communications of the ACM, vol. 46, no. 11, pp. 50–57, 2003.

[12] Chervenak et.al. The Data Grid:Towards an Architecture for the Distributed Management and Analysis of Large Scientific Data Sets. *Journal of Network and Computer Applications:Special Issue on Network-Based Storage Services*, (3):187–200, 2000.

[13] James Broberg, Srikumar Venugopal and Rajkumar Buyya. Market-oriented Grids and Utility Computing: The State-of-the-art and Future Directions. *Journal of Grid Computing*, (3):255–276, 2008.

[14] A. Grimshaw, The ROI Case for Grids, Grid Today, vol. 1, no, 27, 2002.

[15] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke, The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets,'' Journal of Network and Computer Applications, vol. 23, no. 3, pp. 187.200, 2001.

[16] Tuecke, S. et.al. Grid Service Specification. *Global Grid Forum*, 2002

[17] V. Bertstis. Fundamentals of Grid Computing. *Red Books, IBM Corporation*, 2002.

[18] Czajkoski et.al. Grid Information Services for Distributed Resource Sharing. In *10th IEEE International Symposium on High-Performance Distributed Computing (HPDC-10)*, 2001.

[19] J. Nick I. Foster, K. Kesselman and S. Tuecke. The physiology of the Grid. Technical report, Argonne National Laboratoty, University of Chicago, University of Southern California, IBM Corporation, 1999.

[20] Messaoud Benantar. *Access Control Systems*. Springer Publications, 2006.

[21] Sabrina De Capitani et.al. Policies, Models and Languages for Access Control. In *DNIS 2005*, pages 225–237, 2005.

[22] Ravi S.Sandhu, Edward J.Coyne, Hal L.Feinstein and Charles E.Youman. Role-Based Access Control Models. *IEEE Computer*, 29(2):38–47, 1996

[23] S Osborn and Y. Gou. Modeling Users in Role-Based Access Control. In *The 5th ACMWorkshop on Role-Based Access Control*, pages 31–38, Berlin,Germany, 2000.

[24] J.F Barkley and A.V Cincotta. Implementation of Role/Group Permission Association Using Object Access Type. *US Patent 6,202,066*, 2002.

[25] Ferraiolo, Sandhu et.al. NIST Standard for Role-Based Access Control. *ACM Transactions on Information and System Security*, 4(3), 2001.

[26] A. Kern. Advanced Features for Enterprisewide Role-Based Access Control.In *Proceedings of the 18th Annual Computer Security Applications Conference*, Las Vegas,NV, 2002.

[27] A. Kern and M. Kuhlmann. Observations on th Role Life-Cycle in the Context of Enterprise Security Management. In *The 7th ACM Symposium on Access Control Models and Technologies SACMAT 2002*, Monterey,California,USA, 2002.

[28] E. Messmer. Role-Based Access Control on a Roll. *Network World*, 2001

[29] Sejong Oh et.al. An Effective Role Administration Model Using Organization Structure. *ACM Transactions on Information and System Security*, (2):113–137, 2006.

[30] R. McRae. The Stanford Model for Access Control Administration. *Stanford University Journal*, 2000.

[31] J.E. Tidswell and T. Jaeger. Integrated Constraints and Inheritance in DTAC. In *The Fifth ACMWorkshop on Role-Based Access Control*, pages 93–102, Berlin, Germany, 2000.

[32] W.B. Shim and S. Park. Implementing Web Access Control System for Multiple Web Servers in the Same Domain Using RBAC Concept. In *8th International Conference on Parallel and Distributed Systems (ICAPDS)*, pages 768–773, 2001.

[33] Sabrina De Capitani et.al. Policies, Models and Languages for Access Control. In *DNIS 2005*, pages 225–237, 2005.

[34] Gallaher,M.P. et.al. The Economic Impact of Role-Based Access Control.

Planning Report01-2, National Institute of Standards and Technology, 2001.

[35] J. B. D. Joshi, R. Bhatti, E. Bertino, and A. Ghafoor, .Access-Control Language for

Multidomain Environments,. *IEEE Internet Computing*, vol. 8, no. 6, pp. 40.50, 2004.

[36] D. Ferraiolo and R. Kuhn, Role-based Access Control, *Proc. of the 15th National*

Computer Security Conference, 1992.

[37] C. Ramaswamy and R. S. Sandhu, Role-based Access Control Features in Commercial Database Management Systems, *Proc. of the 21st National Information Systems Security Conference*, 1998.

[38] J. B. D. Joshi, R. Bhatti, E. Bertino, and A. Ghafoor, Access-Control Language for Multidomain Environments, *IEEE Internet Computing*, vol. 8, no. 6, pp. 40.50, 2004.

[39] Organization for the Advancement of Structured Information Standards (OASIS), eXtensible Access Control Markup Language (XACML) Version 2.0, available at http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf, 2005.

[40] "Sun's XACML Implementation,"available at http://sunxacml.sourceforge.net/guide.html, 2004.

[41] Open Source J2EE Frameworks http://java-source.net/open-source/j2ee-frameworks

[42] Foster, I., Kesselman, C. (eds.): The grid: Blueprint for a New Computing Infrastructure, 1st edn. Morgan Kaufmann, San Francisco (1998)

[43] Zhang, G.: Study on the Application of Service of Grid Computing Environment. Southeast University, Nanjing (2004)

[44] Kevin Kane et.al. On Classifying Access Control Implementations for Distributed Systems. In *SACMAT 2006*, pages 29–38, 2006.

[45] Pietro Mazzoleni, Bruno Crispo, Swaminathan Sivasubramanian and Elisa Bertino. Efficient Integration of Fine-grained Access Control in Large scale Grid Services. In *IEEE International Conference on Services Computing*, pages 77–86, Orlando, FL, USA, 2005.

[46] Weizhong Qiang, Hai Jin, Xuanhua Shi, Deqing Zou and Hao Zhang. RBGACA: A RBAC Based Grid Access Control Architecture. *International Journal of Grid and Utility Computing*, pages 61–70, 2005.

[47] Mazzoleni,P. et.al. XACML Policy Integration Algorithms. In *SACMAT 2006*, Lake Tahoe, California, USA, 2006.

[48] James B. D. Joshi et. al. Access-Control Language for Multidomain Environments. *IEEE Internet Computing*, 2004.

[49] Rajkumar Buyya, Steve Chapin, David DiNucci, Architectural Models for Resource Management in the Grid, *The First IEEE/ACM International Workshop on Grid Computing (GRID 2000)*, Springer Verlag LNCS Series, Germany, Dec. 17, 2000, Bangalore, India.

[50] Fredj Dridi , Björn Muschall , Günther Pernul, Administration of an RBAC System, *Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04)* - Track 7, p.70187.2, January 05-08, 2004

[51] V. Dhankhar, S. Kaushik, and D. Wijesekera, "Xacml policies for exclusive resource usage", *21st Annual IFIP WG 11.3 Working Conference on Data and Applications Security (DBSec 07)*, 2007.

[52] Marty Humphrey, Member, IEEE, Mary R. Thompson, Member, IEEE, And Keith R. Jackson, Security for Grids, Proceedings Of The IEEE, Vol. 93, No. 3, March 2005.

[53] G. Geethakumari, Atul Negi, V. N. Sastry: A Cross - Domain Role Mapping and Authorization Framework for RBAC in Grid Systems. IJCSA 6(1): 1-12 (2009)

[54] M Humphrey and M R Thompson. Security Implications of Typical Grid Computing Usage Scenarios. In *International Conference on High Performance Distributed Computing(HPDC)*, California, USA, 2001

[55] Jeffrey Dwoskin et.al. Scoping Security Issues for Interactive Grids. In *37th Annual Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, California,USA, 2003.

[56] S. De Smet P. Thysebaert B. Volckaert M. De Leenheer D. De Winter F. De Turck B. Dhoedt P. Demeester, "A performance oriented grid monitoring architecture", published in Proceedings of the 2nd IEEE Workshop on End-to-End Monitoring Technics and Sercives (E2EMON), Monitoring Emerging Network Services, California, 03 October 2004

[57]Mark Baker, Garry Smith, "GridRM: A Resource Monitoring Architecture for the Grid", *Proceedings of the Third International Workshop on Grid Computing*, p.268-273, November 18, 2002

[58] Askar S. Boranbayev, "Defining methodologies for developing J2EE web-based information systems", Nonlinear Analaysis: Theory, Methods & Applications, Vol. 71, No. 12, pp. e1633-e1637, December 2009

[59] *Juanjuan Yan, Bo Chen, Xiu-e Gao, Le Wang*, "*Research of Structure Integration Based on Struts and Hibernate*," *csie*, vol. *7*, pp.*530-534, 2009*.

[60] Piatek, J. Zabierowski, W. Napieralski, A. Katedra Mikroelektroniki iTech. Informatycznych, Politech. Lodzka, Lodz,"Using web technologies based on college - graduate web contact management system. Form system",CAD Systems in Microelectronics, 2009. CADSM 2009. 10th International Conference, pp 251-254, Lviv-Polyana, Feb 2009

[61] Counsell A. (2004). Opening the Door to Collaboration. *Financial Times IT Review*, March 31st.

[62] Gregotio, D. D. & Kassicieh, K. S. [2005], "Drivers of E-Business Activity in Developed an Emerging Market", *IEEE Transaction on Engineering Management*, Vol .52, No. 2, May 2005

[63] Hendershot, R. (2007), "E-business Benefits: Learn about the advantages of adopting an eBusiness solution for your small business", www.enetsc.com/EBusinessArticles.htm

[64] Web Portals. A Word Definition from Webopedia. http://www.webopedia.com/TERM/W/Webportal.html

[65] Open Source J2EE Frameworks http://java-source.net/open-source/j2ee-frameworks

[66] Apache Struts http://struts.apache.org/1.x/userGuide/index.html

[67] JInt Demo: Implementing MVC Architecture using struts http://www.oracle.com/technology/sample_code/tech/java/j2ee/jintdemo/tutorials/Struts.html

[68] Introduction to the Spring Framework

http://www.theserverside.com/news/1364527/Introduction-to-the-Spring-Framework

[69] HIBERNATE - Relational Persistence for Idiomatic Java , Accessed from: http://www.Hibernate.org/hib_docs/v3/reference/en/html/preface.html

[70] P. Peak and N. Heudecker, Hibernate Quickly. Manning Publications, 2006, vol. ISBN 1932394419. 38

[71] GridGain, http://www.gridgain.com/

[72] Java Community Process: *JSR 152 Java Server Pages 2.0 Specification*. http://www.jcp.org/en/jsr/detail?id=152

[73] XACML RBAC Profile 1.0 http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-rbac-profile1-spec-os.pdf

[74] G.Geeta kumari, V.N.Sastry, Atul Nehi, "Grid Computing Security through Access Control Modelling", PHDThesis, March 2010

APPENDIX

GLOBUS TOOLKIT

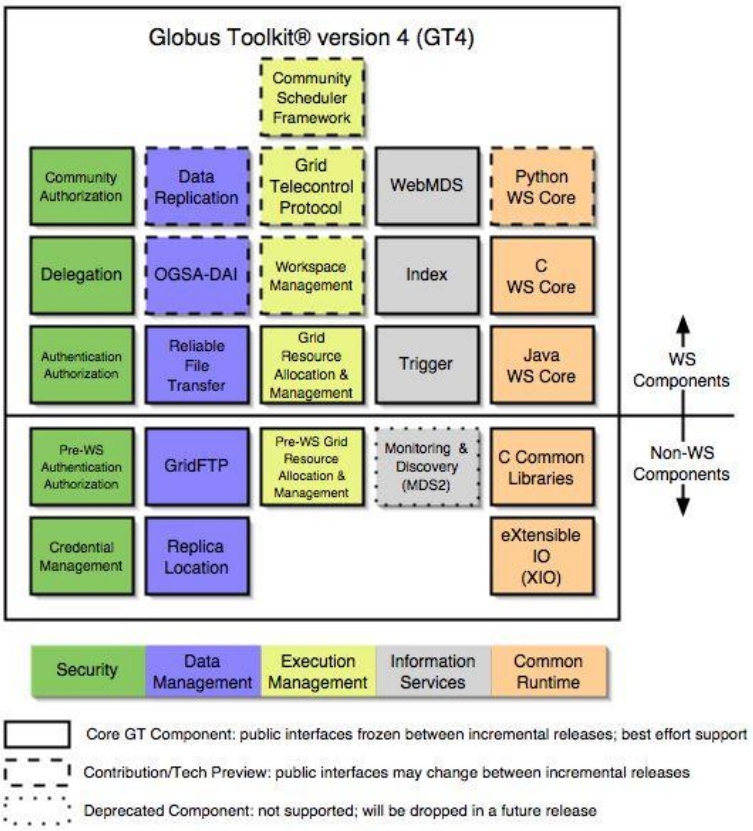
Installation of Globus Toolkit

This helps as a guide to Globus Toolkit 4.0.1 (GTK4.0.1) installation. It deals with basic installation that installs core services and a few base services namely: a security infrastructure (GSI), GridFTP and RFT. After the installation and testing, we explore the possibility of using the CA of one machine on another machine.

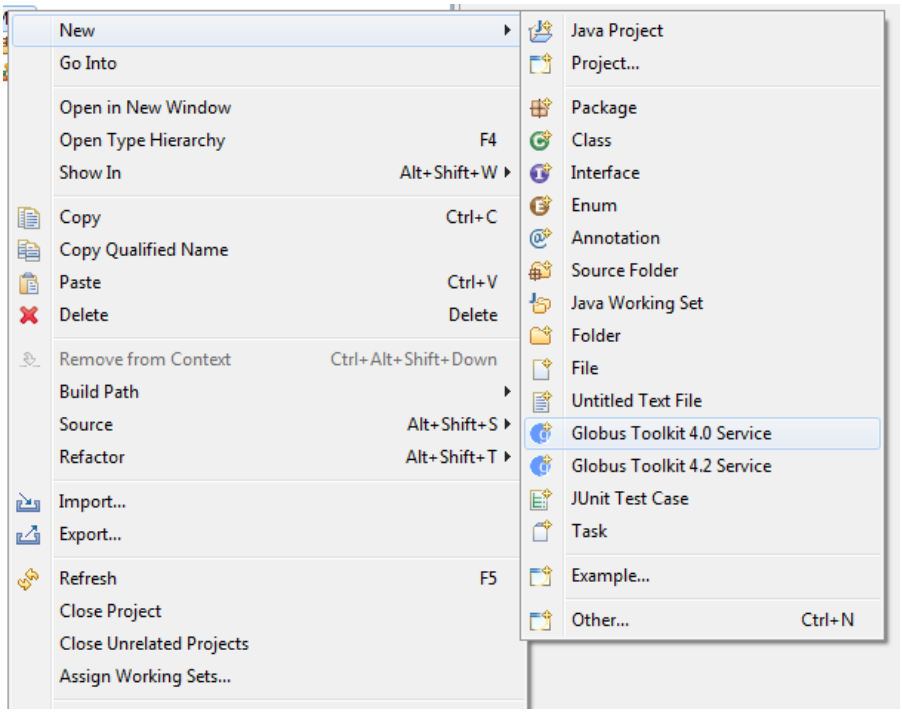
1. Pre-requisites (Softwares/Databases)
   - Java
   - Apache Ant
   - Apache tomcat
   - Postgresql database
2. Optional Software
   - IODBC

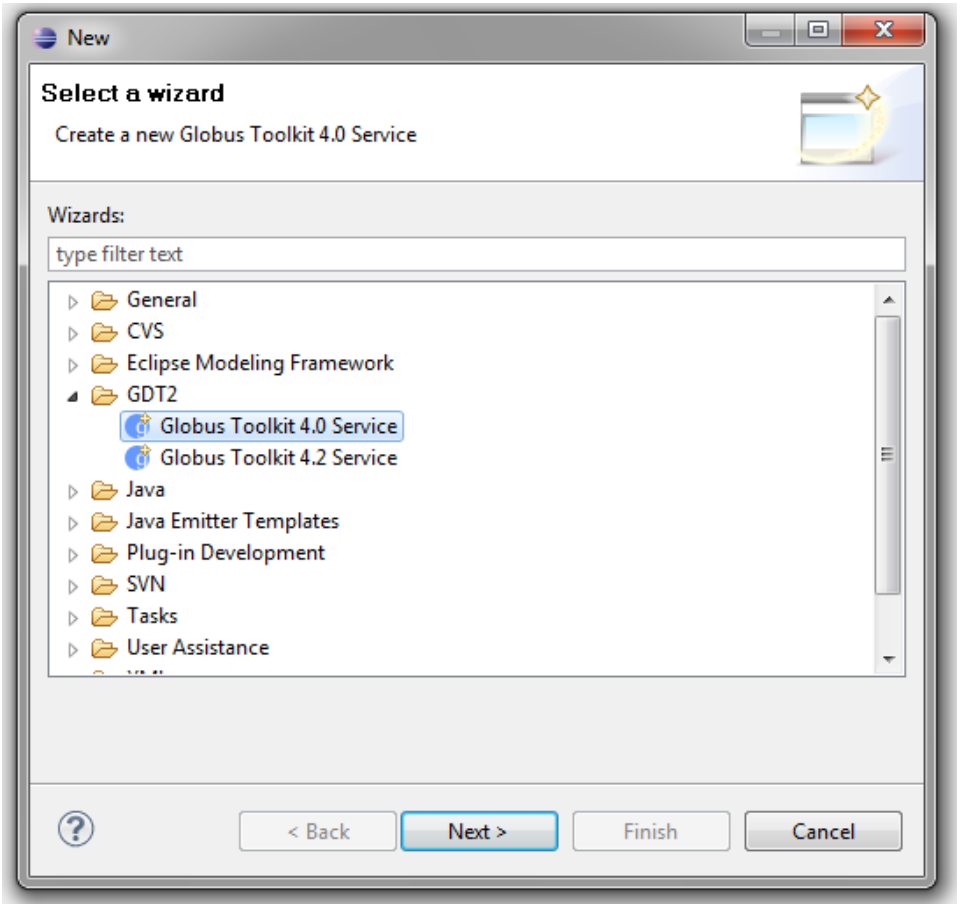
   Before starting the installation, first we need to create a globus user.
3. Building the Toolkit
4. Setting up the simpleCA
5. Get the host and container certificates signed by simpleCA
6. Get valid credentials from the CA
7. Create grid-map file and add entries to it
8. Configure GridFTP
9. Install postgresql and configure RFT
10. Start the Globus container and test using the globus client

Globus Toolkit (GT4) Architecture:



Globus Toolkit in MyEclipse:




New
Select a wizard
Create a new Globus Toolkit 4.0 Service
Wizards:
type filter text
General
CVS
Eclipse Modeling Framework
GDT2
Globus Toolkit 4.0 Service
Globus Toolkit 4.2 Service
Java
Java Emitter Templates
Plug-in Development
SVN
Tasks
User Assistance
< Back
Next >
Finish
Cancel