Ch 3 – Role of GC and autonomic computing in CC

Contents

- Grid Computing
- Interaction of Models of Grid and CC
- Distributed Computing in Grid and Cloud
- Layered Models and Usages Patterns in Grid and Cloud
- Interoperability in Grids and Clouds
- Autonomic Computing
- System Models of Autonomic Computing
- Roles and Importance of Autonomic Computing in Cloud
- Autonomic Cloud Computing.

Grid Computing

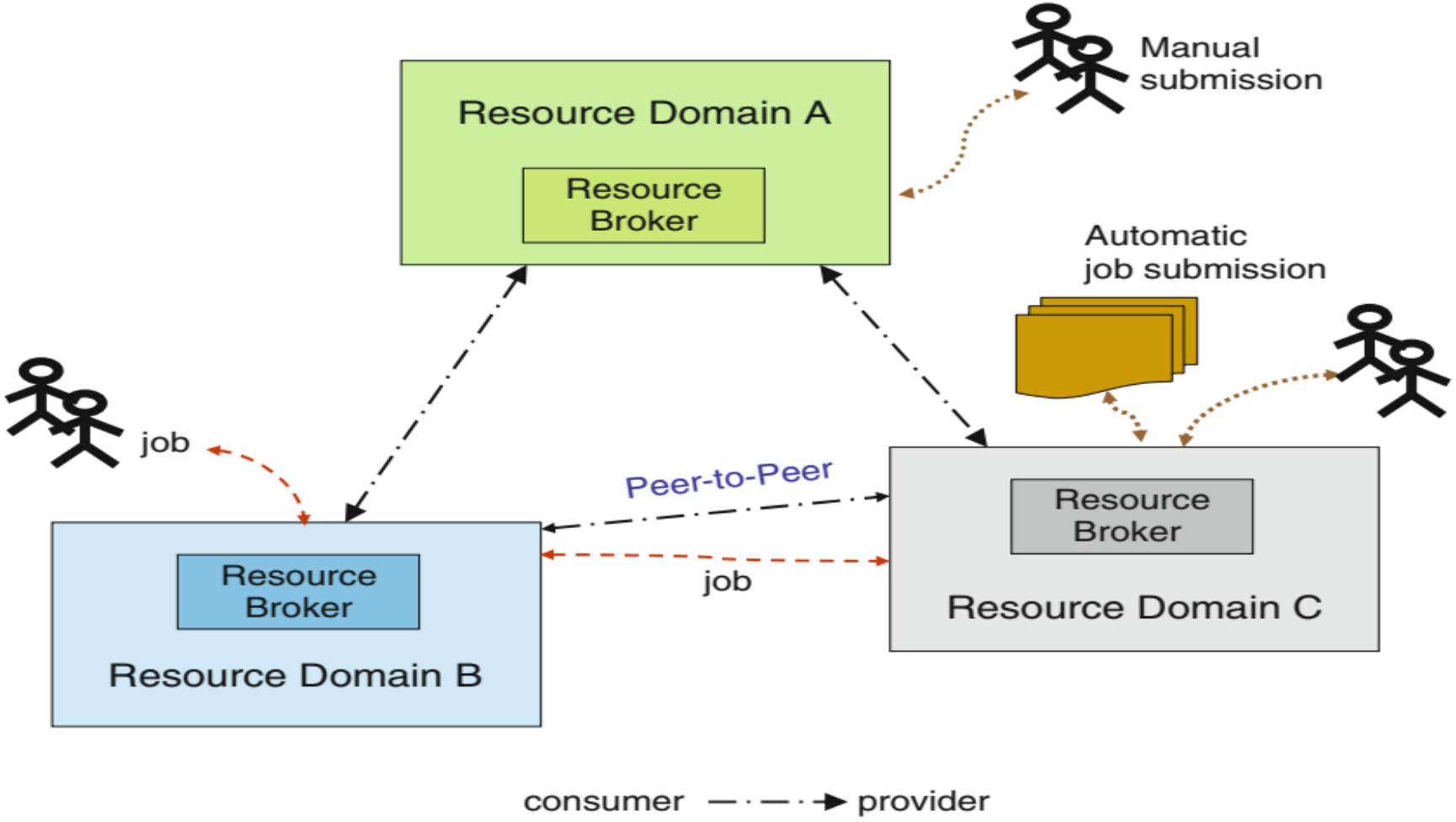
- Grid Computing get benefit distributed resources from various institutions called resource providers on the demands of clients consuming them. Heterogeneous resources from different providers: Software, Storage units, hardware architectures, etc.

- Where Cloud Services are “consumer and business products, services and solutions that are delivered and consumed in real-time over the Internet” while Cloud Computing is “an emerging IT development, deployment and delivery model, enabling real-time.

Grid Computing

- Grid Computing binds distributed resources from various institutions (resource providers), to meet the demands of clients consuming them.
- Resources from different providers are likely to be diverse and heterogeneous in their functions (computing, storage, software, etc.).
- Any hardware architectures (Intel x86, IBM P series, etc.) and usage policies set by owning institutions.
- Developed under the umbrella of Grid Computing and responsible for the aggregation of resource information and availability, selection of resources to meet the clients' specific requirements.

Grid Computing



. 8.1 Grid collaborating domains

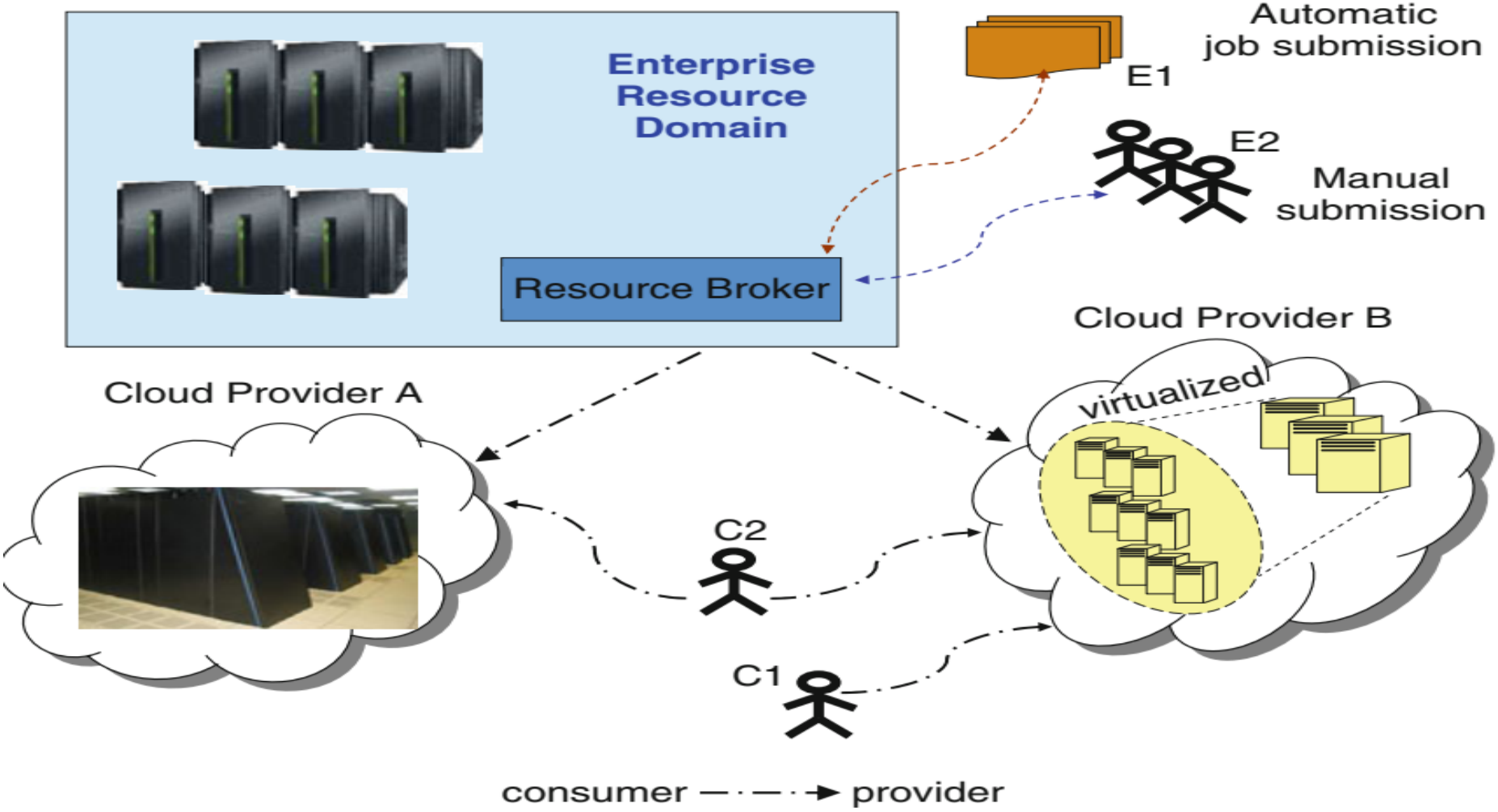
Grid Computing

- Above figure shows an typical relationship of resource providers and consumers for a collaborative Grid computing scenario.
- Clients or users submit their requests for application execution along with resource requirements from their home domains.
- A Resource broker selects a domain with appropriate resources to acquire from and to execute the application or route the application to domain for execution with results and status returning to the home domain.

Interaction of Models of Grid and CC

- Most scalable interaction models of Grid domains are P2P where most of the Grid participating organizations are both consumers and providers.
- In GC, there are usually resource sharing among the peers and clients of consumer organizations in Grids use heterogeneous resources from more than one resource provider belonging to the same VO to execute their applications.
- The organization of Open Grid Forum (OGF) has the goal of establishing necessary standards for Grid computing.
- Many vendor proposed standards include Job Submission Description Language (JSDL), Basic Execution Service (BES) and many others.

Interaction of Models of Grid and CC



- Cloud usage models

Interaction of Models of Grid and CC

- In Cloud, service providers offer their own proprietary service protocols and information formats. Cloud consumers will likely demand common protocols and standardized information formats for ease of federated usage and interoperability.
- The Open Virtualization format (OVF) of the Distributed Management Task Force (DMTF) is an model proposal in this direction. Modeled after similar formations in the Grid community, OGF officially launched a workgroup, named the Open Cloud Computing Interface Working Group (OCCI-WG) to develop the necessary common APIs for the lifecycle management of Cloud infrastructure services.

Distributed Computing in Grid and Cloud

- The Grid encompasses two areas of distributed system activity.
- First one is operational with an objective of administrating and managing an interoperable collection of distributed compute resource clusters on which to execute client jobs, typically scientific/HPC applications.
- The procedures and protocols required to support clients from complex services built on distributed components that handle job submission, security, machine provisioning, and data staging.

Distributed Computing in Grid and Cloud

- The Cloud has similar operational requirements for supporting complex services to provide clients with services on different levels of support such application, platform and infrastructure.
- The Grid also represents as a coherent entity a collection of compute resources that may be under different administrative domains, such as universities, but inter-operate transparently to form virtual organizations.
- Interoperability is not a near term priority in GC, there is a precedent for commercial Clouds to move in this direction similarly to how utilities such as power or communication contract with their competitors to provide overflow capacity.

Distributed Computing in Grid and Cloud

- Second aspect of distributed computing in the Grid is job themselves are distributed.
- Jobs running on tightly coupled nodes within a cluster and leveraging middleware services such as MPICH (MPI-Chameleon).
- Jobs running in the Grid are not typically interactive, and some may be part of more complex services such as e-science workflows.

Distributed Computing in Grid and Cloud

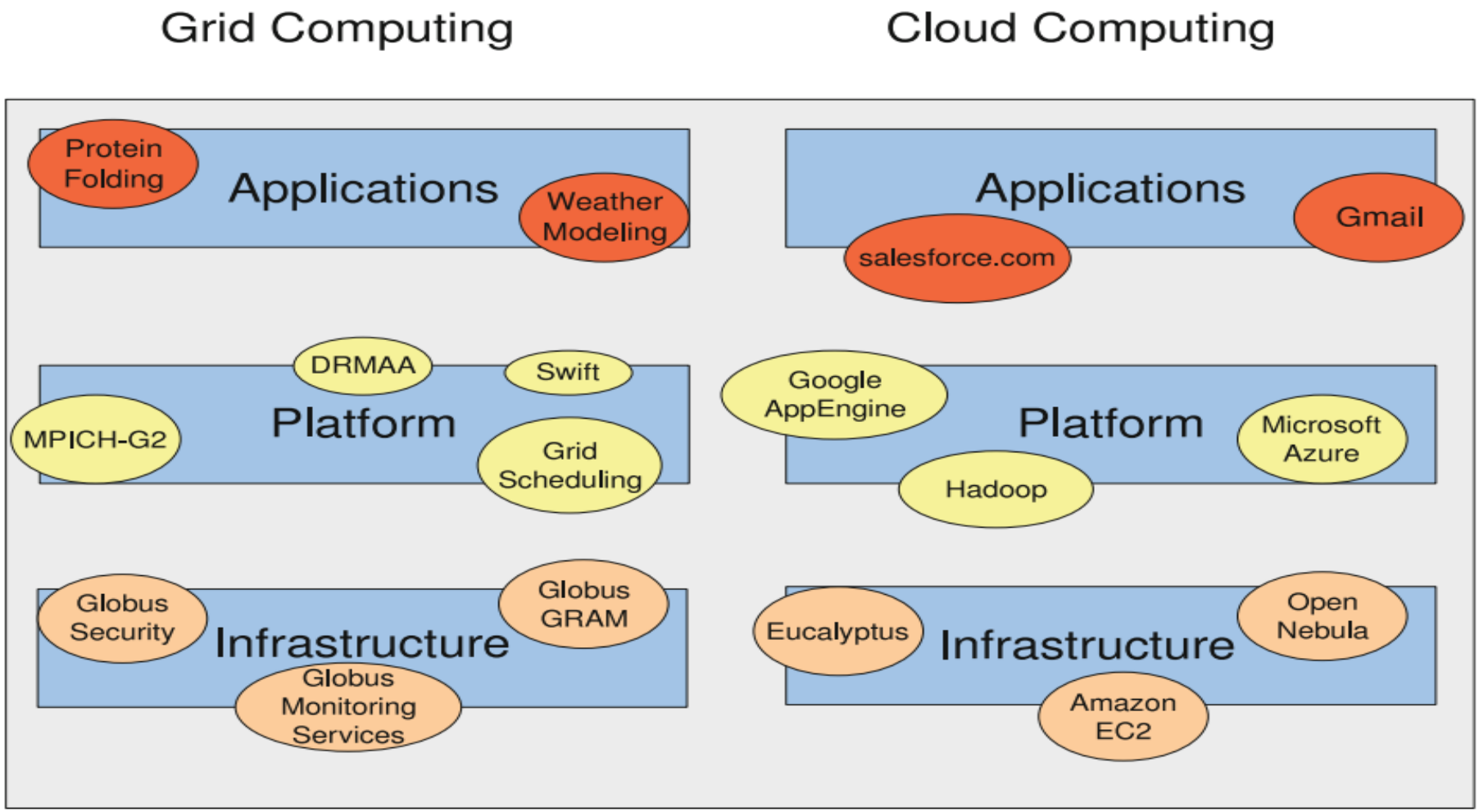
- Workloads in Clouds usually consist of more loosely coupled distributed jobs such as map/reduce, and HPC jobs written to minimize internode communication and leverage concurrency provided by large multi-core nodes.

- The larger business process workload aspects of jobs running in the Cloud or Grid have implications for structuring the services that administer and manage the quality of their execution.

Layered Models and Usages Patterns in Grid and Cloud

- There are many similarities in Grid and Cloud computing systems.
- We compare the approaches by differentiating three layers of abstraction in Grid: Infrastructure, Platform and Application.
- Then we map these three layers to the Cloud services of IaaS, PaaS, and SaaS in next slide.

Layered Models and Usages Patterns in Grid and Cloud



.3 Grid and cloud layers

Grid and Cloud: ***Infrastructure***

- In infrastructure layer, Clouds share most characteristics with the original purpose of Grid middleware like Eucalyptus, OpenNebula, or Amazon EC2.

- In these systems users can provision execution environments in the form of virtual machines through interfaces such as APIs or command line tools.

- The act of defining an execution environment and sending a request to the final resource has many similarities with scheduling a job in the Grid.

Grid and Cloud: ***Infrastructure***

- To be authorized to use the system in grid and cloud.
- In Grid systems this is managed through the Community Authorization System (CAS) or by contacting a Certificate Authority that is trusted by the target institution, which issues a valid certificate.
- In Clouds system usually offer web forms to allow the registration of new users, and have additional web applications to maintain databases of customers and generate credentials, such as the case of Eucalyptus or Amazon.
- Eg: Amazon API tools for Amazon EC2, the euca2ools for Eucalyptus or the OpenNebula command line interface.

Grid and Cloud: ***Infrastructure***

- For resource usage in Grid and cloud,
- Grid supports a Resource Specification Language (RSL) and a Job submission Description Language (JSDL) that can define what process is to be run on the target machine based on machine architecture, processor speed, amount of memory, etc.
- Where, Clouds require different attributes such as the size of the execution environment or the virtual machine image to be used.

Grid and Cloud: ***Infrastructure***

- After the job execution in Grid and cloud, there is a match-making and scheduling phase involved.
- In Grid, The GRAM component from Globus is specially flexible , the job manager just performs a fork call to spawn a new process on the target machine. Eg: LoadLeveler or Sun Grid Engine systems are able of multiplexing jobs that are sent to a site into multiple resources.
- Cloud systems have simpler job management strategies, since the type of jobs are homogeneous. For eg, Eucalyptus uses a Round Robin scheduling technique to alternate among machines and OpenNebula implements a Rank Scheduling Policy to choose the most adequate resource for a request.

Grid and Cloud: ***Infrastructure***

- For job submission, transferring the data to and from the execution machine involves retrieving the input data for the process from a remote destination, such a GridFTP server (**stage-in**).
- The second part of the process, **stage-out**, consists in either transferring the output data to the user's machine or to place it in a repository, possibly using the RLS.

- In the case of Cloud computing, the most important data that has to be transferred is the definition of an execution environment, usually in terms of Virtual Machine images.

Grid and Cloud: ***Infrastructure***

- Users upload the data describing the operating system and packages needed to instantiate the VM and later reference it to perform operations such as booting a new machine. There is no standard method for transferring data in Cloud systems, but it is worth noting S3, which allows users to move entities from 1 byte to 5 GB in size.
- In the case of monitoring, Grid and Cloud systems need to offer users a method to monitor their jobs, as well as their resource usage. This facility can also be used by site administrators to implement usage accounting in order to track resource utilization and enforce user quotas. Eg. Monitoring and discovery service (MDS) and CloudWatch.

Grid and Cloud: ***Platform***

- Platform is built on top of the physical infrastructure and offers a higher level of abstraction to users.
- The interface provided by a PaaS solution allows developers to build additional services without being exposed to the underlying physical or virtual resources.
- These facts enable additional features to be implemented as part of the model, such as presenting seemingly infinite resources to the user or allowing elastic behavior on demand.
- Examples of Cloud solutions that present these features are Google App Engine, Salesforce's force.com or Microsoft Azure.

Grid and Cloud: ***Platform***

- Abstraction from Physical Resources.
- Infrastructure layer provides users with direct access to the underlying infrastructure for the lower levels of resource interaction, in the Platform level a user should be isolated from them and allows developers to create new software.
- Programming API to Support New Services - API directly influences the programs that can be built on the Cloud, therefore each PaaS solution is usually designed.
- Grid systems allow developers to produce new software that take advantage of the shared resources in order to compare them with PaaS solutions.

Grid and Cloud: ***Platform***

- Many libraries are provided by Grid **middleware** to access resources programmatically like Commodity Grid (CoG) Kit is an example.
- GridSuperscalar is a programming paradigm to enable applications to run on the Grid.
- Another programming paradigm on top of the Grid is SWIFT which run very large numbers of jobs in the order of execution.
- So, Cloud PaaS paradigms compared to the Grid models need to use the lowest common denominator when implementing new services because the degree of compatibility with the middleware is directly related to the number of resources available.

Grid and Cloud: ***Applications***

- There is no clear distinctions between applications developed on Grids and those that use Clouds to perform execution and storage.
- But, it is undeniable that the vast majority of Grid applications fall in the realm of scientific software, while software running in Clouds has leaned towards commercial workloads.
- Some possible causes for the different levels of adoption of these technologies for the development of applications:

Grid and Cloud: ***Applications***

- Lack of business opportunities in Grids- Usually Grid middleware is installed only in hardware intended for scientific usage .This phenomenon has not successfully produced business opportunities.
- Complexity of Grid tools - Grid middleware is perceived by many as complex and difficult to install and manage while Cloud infrastructures have usually been developed by providers to fit their organization's needs and with a concrete purpose in mind, making them easier to use and solution oriented.
- Affinity with target software - Most Grid software is developed with scientific applications in mind, which is not true for the majority of Cloud systems. For eg virtualization, Clouds are more targeted to web applications.

Interoperability in Grids and Clouds

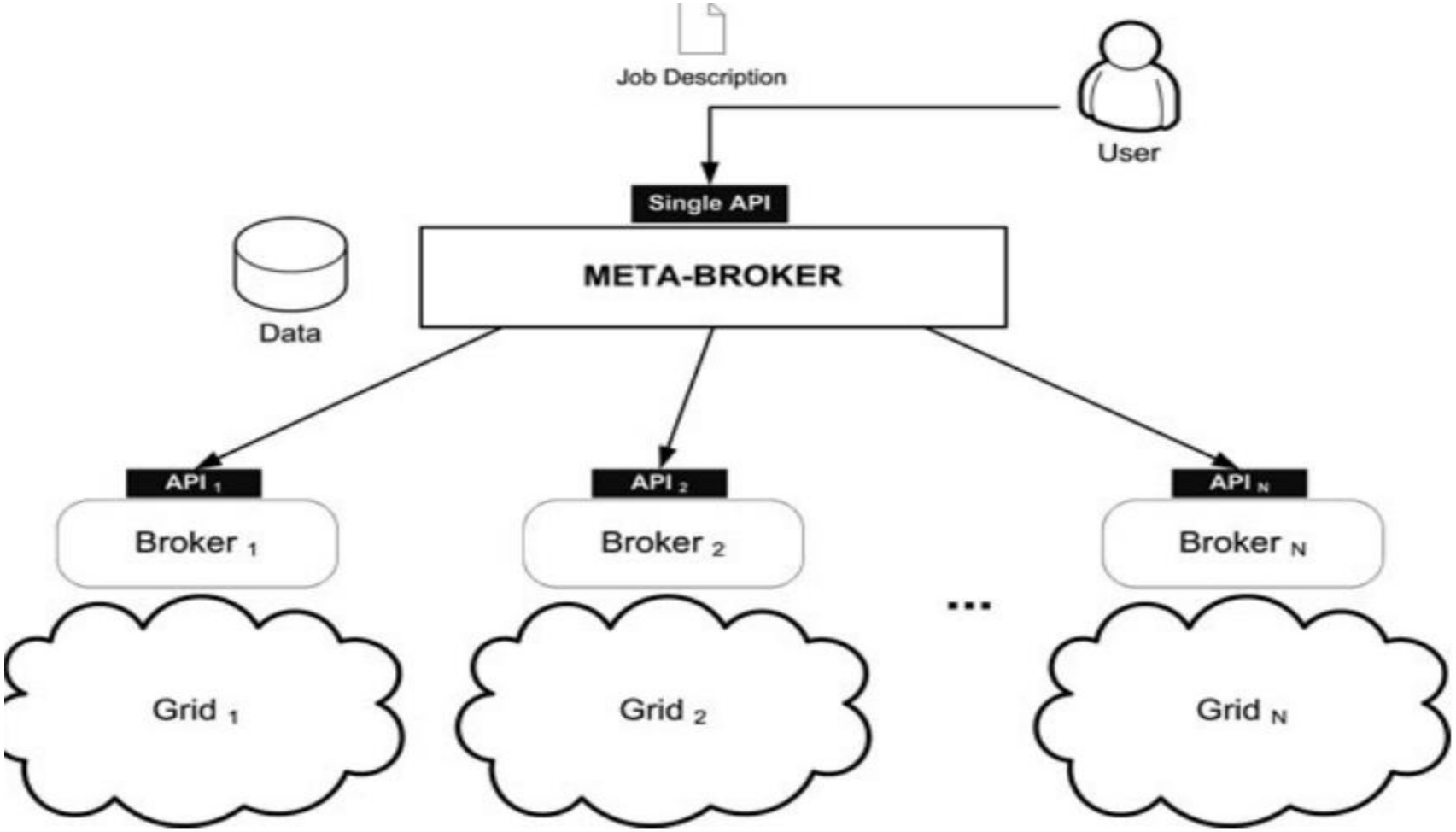
- The goal of Grid computing is to provide uniform and consistent access to resources distributed in different data centers and institutions.
- Because the majority of Grids are formed based on regional as opposed to local initiatives so interoperation is a key objective.
- For examples, TeraGrid in US, GridX1 in Canada, Naregi in Japan and EGEE in Europe.
- Interoperation is addressed at various architectural points such as the access portal, resource brokering function, and infrastructure standardization.

Interoperability in Grids and Clouds

- Some production Grid environments, such as HPC-Europa, DEISA and PRACE, approach interoperability using a uniform access interface to application users.
- Software layers beneath the user interface then abstract the complexity of the underlying heterogeneous supercomputing infrastructures.

- One tool that takes this approach for Grid interoperation is meta-brokering which supports the Grid interoperability from the viewpoint of the resource management and scheduling present in next slide.

Interoperability in Grids and Clouds



Meta-brokering architecture

Interoperability in Grids and Clouds

- I**nfrastructure interoperability**
- G**ridWay** is mainly on Globus, supports multiple Grids using Grid gateways to access resources belonging to different domains.
- **GridWay** forwards local user requests to another domain when the current one is overloaded.

- R**esource Optimization in interoperated Grids**
  K**oala Grid Scheduler** is focused on data and processor co-allocation. To inter-connect different Grid domains as different Koala instances. Their policy is to use resources from a remote domain only if the local one is saturated.

Interoperability in Grids and Clouds

- K**oala Grid Scheduler** use delegated matchmaking to obtain the matched resources from one of the peer Koala instances without routing the jobs to the peer domains.
- I**nterGrid** promotes interlinking different Grid systems through peering agreements based on economic (communication cost) approaches to enable inter-grid resource sharing.
- Many other projects explore the interoperability of Grid systems through the use of standard mechanisms, protocols and interfaces.

Interoperability in Grids and Clouds

- Today, Cloud standardization groups have started working on defining common interfaces for interoperation.
- Like, the Open Grid Forum Open Cloud Computing Interface (OCCI) working group of OGF is working on defining an API specification for remote management of Cloud computing infrastructure.
- Allowing for the development of interoperable tools for common tasks including deployment, autonomic scaling and monitoring.
- Project OpenNebula and RESERVOIR projects have provided OCCI-compiant implementations.

Interoperability in Grids and Clouds

- The Cloud Computing Interoperability Forum (CCIF) is a vendor neutral, not for profit community of technology advocates, and consumers dedicated to driving the rapid adoption of global Cloud computing services.

- While encouraging activities in the area of interoperable Clouds are occurring, Grid technologies are more mature. Thus, it is promising to extend these to the Cloud, particularly in the research and evaluation of scheduling and resource selection strategies.

Autonomic Computing

- Inspired by the autonomic nervous system, autonomic computing aims at designing and building self-managing systems and has emerged as a promising approach for addressing the challenges due to software complexity.
- An autonomic system is able to make decisions to respond to changes in operating condition at runtime using high-level policies that are typically provided by an expert.
- Autonomic system constantly monitors and optimizes its operation and automatically adapts itself to changing conditions so that it continues to achieve its objectives.

- There are several important and valuable milestones to reach fully autonomic computing:

Autonomic Computing

- First, automated functions will merely collect and aggregate information to support decisions by human users.
- Later, they will serve as advisors, suggesting possible courses of action for humans to consider.
- Self-management is the essence of autonomic computing and has been defined in terms of the following four aspects of self-management.
  - *Self configuration*:
  - *Self optimization*
  - *Self healing*
  - *Self protection*

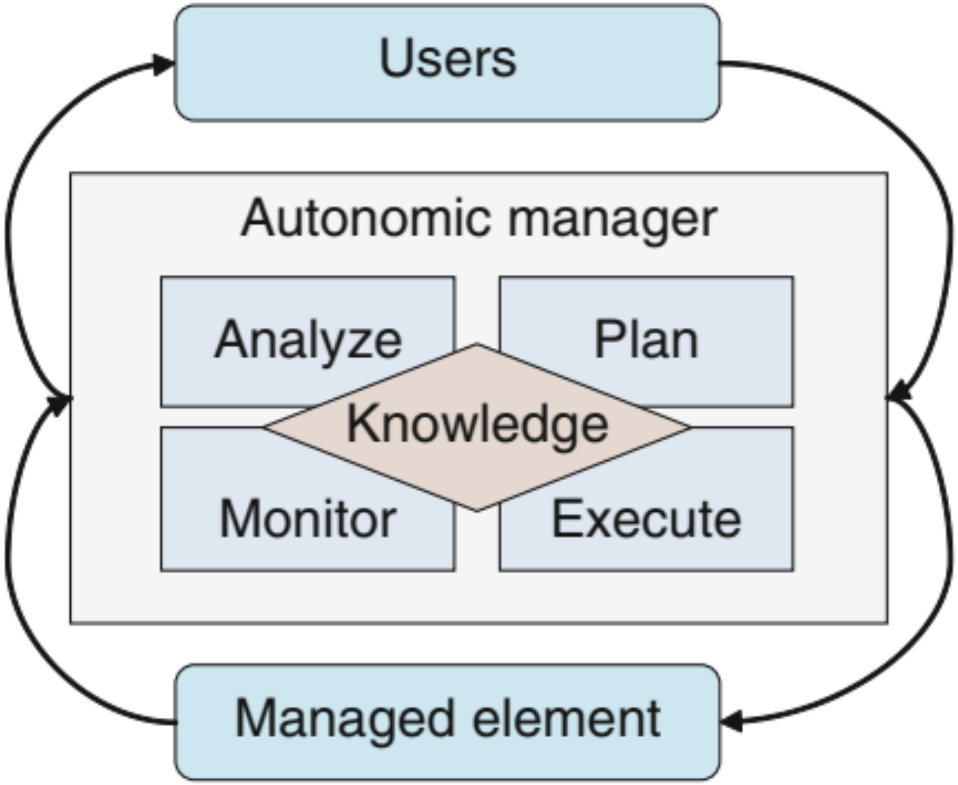
Autonomic Computing

- *Self configuration*: Autonomic systems will configure themselves automatically in accordance with high-level policies representing business-level objectives.
- *Self optimization*: Autonomic systems will continually seek ways to improve their operation, identifying and seizing opportunities to make themselves more efficient in performance and/or cost.
- *Self healing*: Autonomic computing systems will detect, diagnose, and repair localized problems resulting from bugs or failures in software and hardware.
- *Self protection*: Autonomic systems will be self-protecting in two senses. One when correlated problems arising from malicious attacks or cascading failures that remain uncorrected by self healing measures and second, anticipate problems based on early reports from sensors and take steps to avoid or mitigate them.

Autonomic Computing

- The basic structure of an autonomic element as proposed by IBM consists of autonomic manager which monitors, analyzes, plans and executes based on collected knowledge, and external environments including human users and managed elements.
- The managed element could be CPU, memory and storage, software resources such as a database, a directory service or a system, or an application.
- The autonomic manager monitors the managed elements and its external environment including changing users requirements, and analyzes them, computes a new plan reflecting changing conditions and executes this plan.

Autonomic Computing

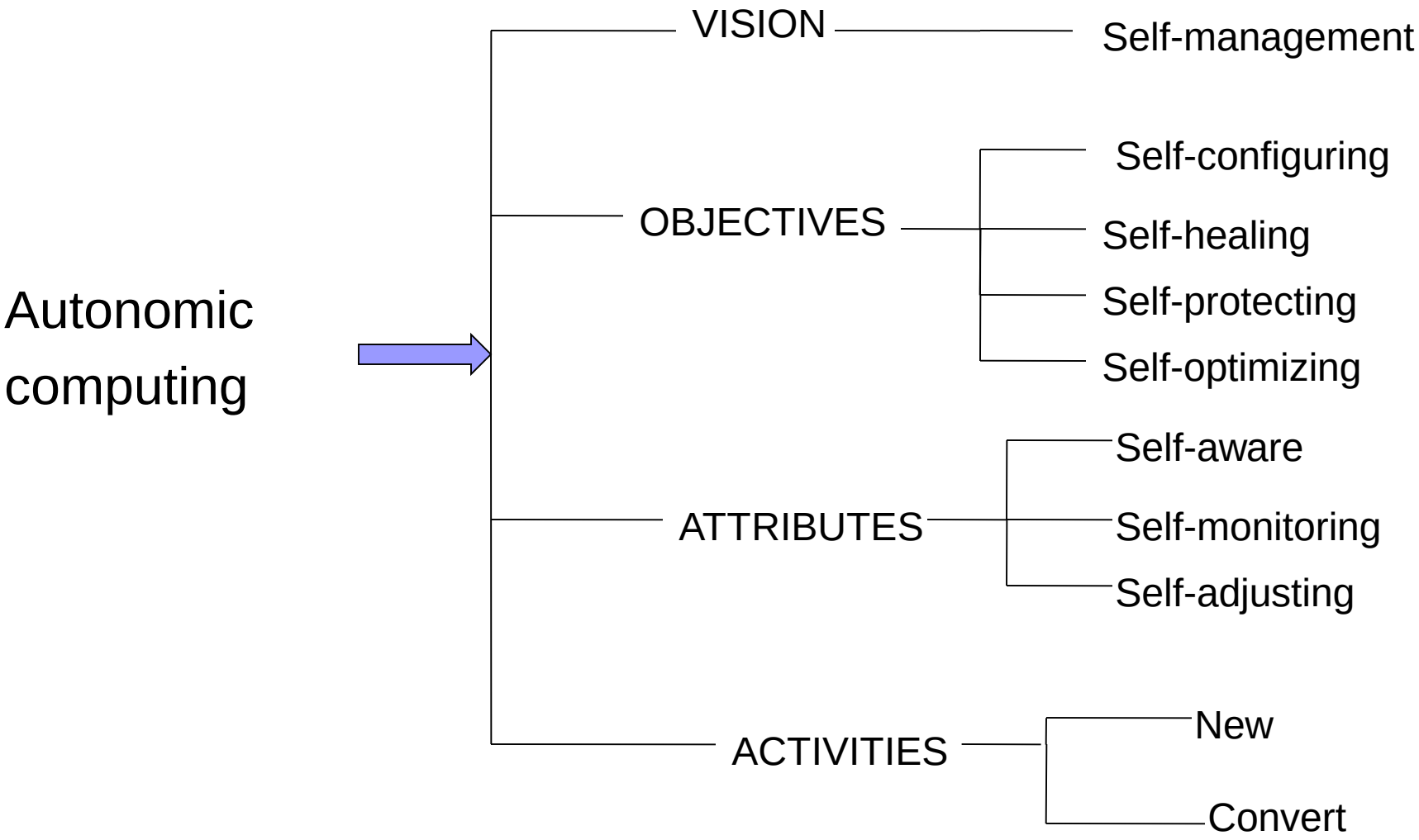


System Models of Autonomic Computing

Autonomic Computing: Architecture details

- Autonomic manager is a component that implements the control loop
  - **Monitor Function**
    - the function that collects, aggregates, filters and reports details (e.g. metrics, topologies)
  - **Analyze Function**
    - the function that models complex situations to understand current system state.
  - **Plan Function**
    - the function that structures the actions needed to achieve goals and objectives.
  - **Execute Function**
    - the function that changes the behavior of the managed resource using effectors.

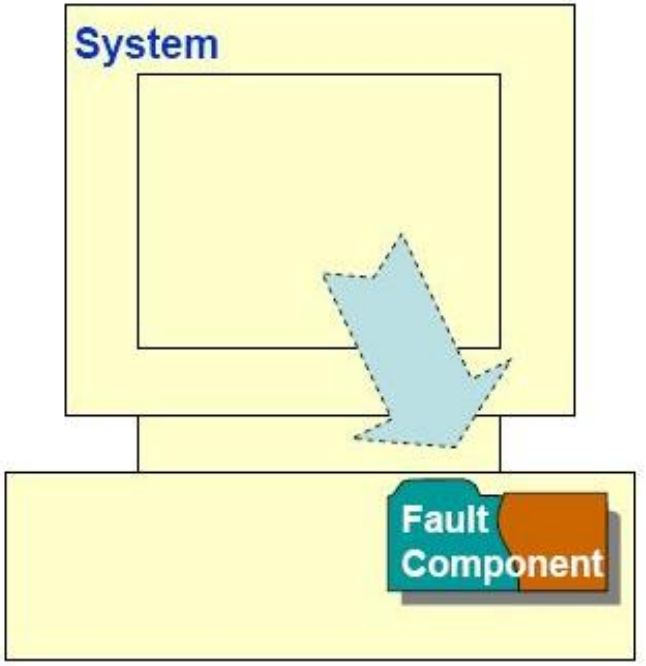
Autonomic Computing: Tree



Self Healing

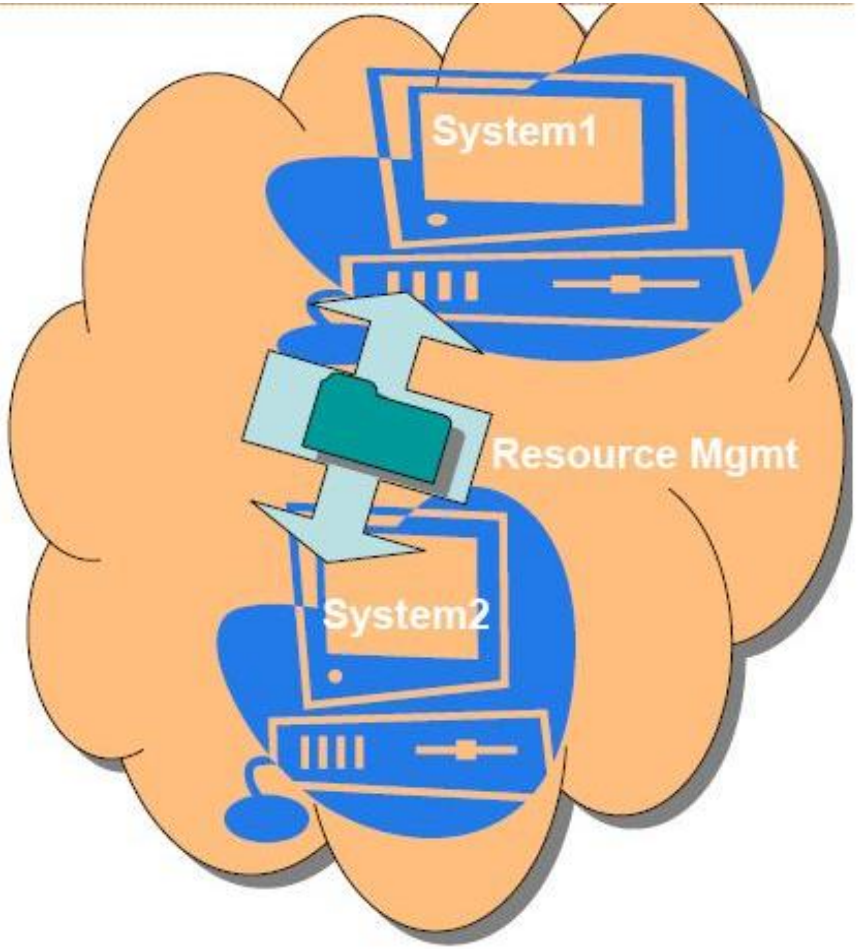
- Discover, diagnose and react to disruptions without disrupting the service environment

- Fault components should be
  - *detected*
  - *Isolated*
  - *Fixed*
  - *reintegrated*



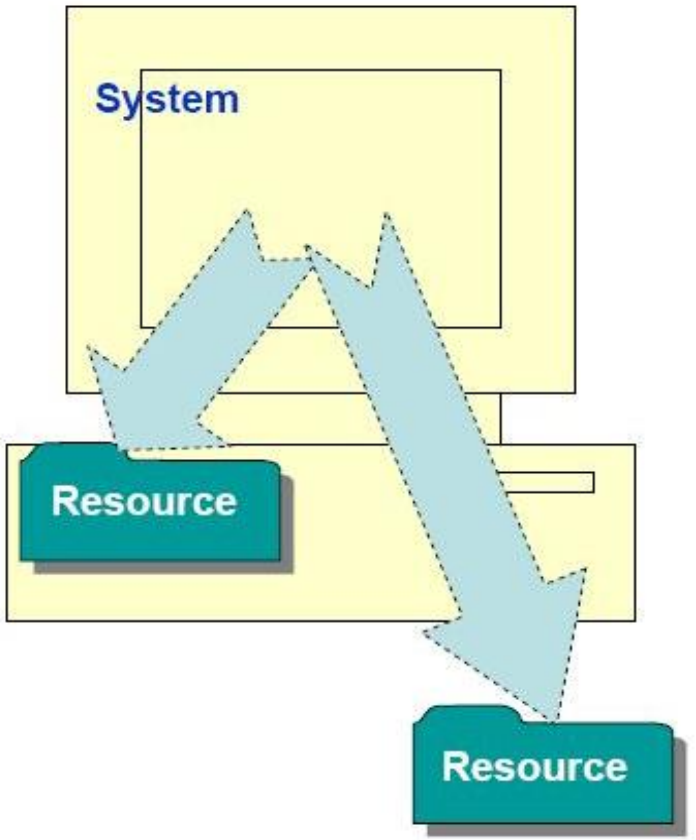
Self Optimization

- Monitor and tune resources automatically
  - *Support operating in unpredictable environment*
  - *Efficiently maximization of resource utilization without human intervention*
- Dynamic resource allocation and workload management.
  - *Resource: Storage, databases, networks*
  - *For example, Dynamic server clustering*



Self protection

- Anticipate, detect, identify and protect against attacks from anywhere
  - *Defining and managing user access to all computing resources*
  - *Protecting against unauthorized resource access, e.g. SSL*
  - *Detecting intrusions and reporting as they occur*

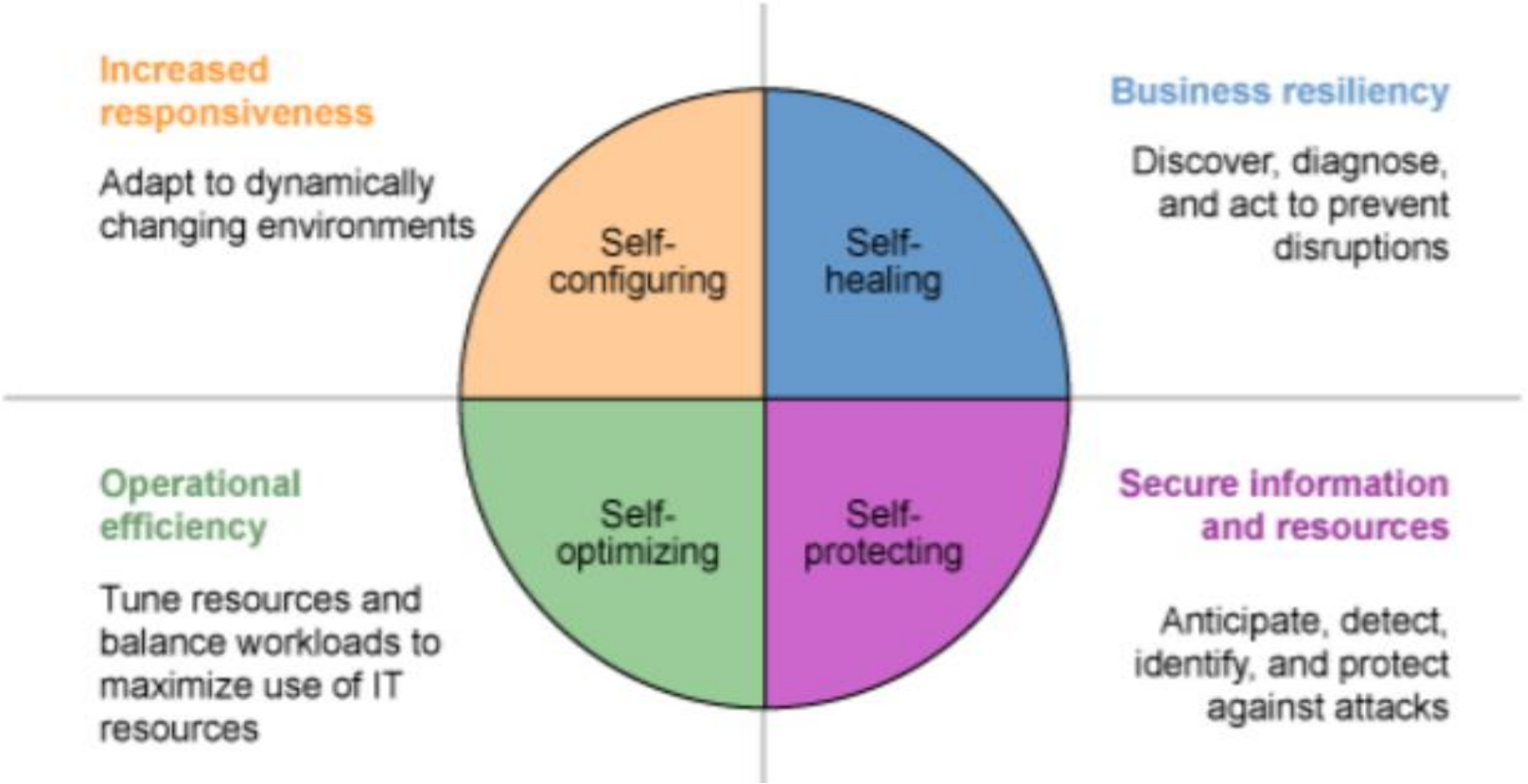


Self configuring

Autonomic Computing: Example

- **Intel's Itanium 2 Processor (self healing)**
- Intel Itanium 2 processor has built-in Autonomic Features.
- It allows the system to continue executing transactions as it recovers from several error conditions.
- DB2: Self-Optimization
- Standard query optimisers would not be considered as providing autonomicity. However if while a query was running and the DBMS was monitoring the query's execution and deciding on a different query plan, then we would consider that autonomic.

Roles and Importance of AC in Cloud



Autonomic and Cloud Computing

Comparative view	Cloud computing	Autonomic computing
Characteristics	1\ Dynamic computing infrastructure 2\ Minimally or self managed platform 3\ Consumption based billing	1\ Self-Configuration 2\ Self-Healing 3\ Self-Optimization 4\ Self-Protection
Location	Computers do not have to be in the same physical location	Computers do not have to be in the same physical location
Operating system	The memory, storage devices and net work communications are managed by the OS of the basic physical cloud units	All Computers controlled by individual OS Composed Error Correction And Detection
Node indecency	Every node acts as an independent entity	Every node acts as an independent entity

Autonomic and Cloud Computing

Comparative view	Cloud computing	Autonomic computing
Net work	Clouds are mainly distributed over MAN	Autonomics are mainly distributed over WAN
processing system	Allows Multiple Smaller Applications To Run At the same Times	Problems Solve According Hi Speed Micro Processor By Scheduling Algorithm
Computing area	1/ Banking 2/ Insurance 3/ Weather for Casting 4/ Space Exploration 5/ SAAS 6/ PAAS 7/ IAAS	1/ Composite resources tied to business decision-making 2/ Composite resources decision-making, e.g., cluster servers 3/ Resource elements managing themselves

Autonomic cloud Computing

- Autonomic cloud computing helps address challenges related to QoS by ensuring SLA are met.
- QoS is maintained by mostly scaling up or down resources automatically depending on demand by client's business.
- Autonomic cloud computing helps reduce the carbon footprint of data centers and cloud consumers by automatically scaling up or down energy usage base on cloud activity.
- Autonomic monitoring are mostly implemented on specific layers of the cloud computing architecture.

Autonomic cloud Computing

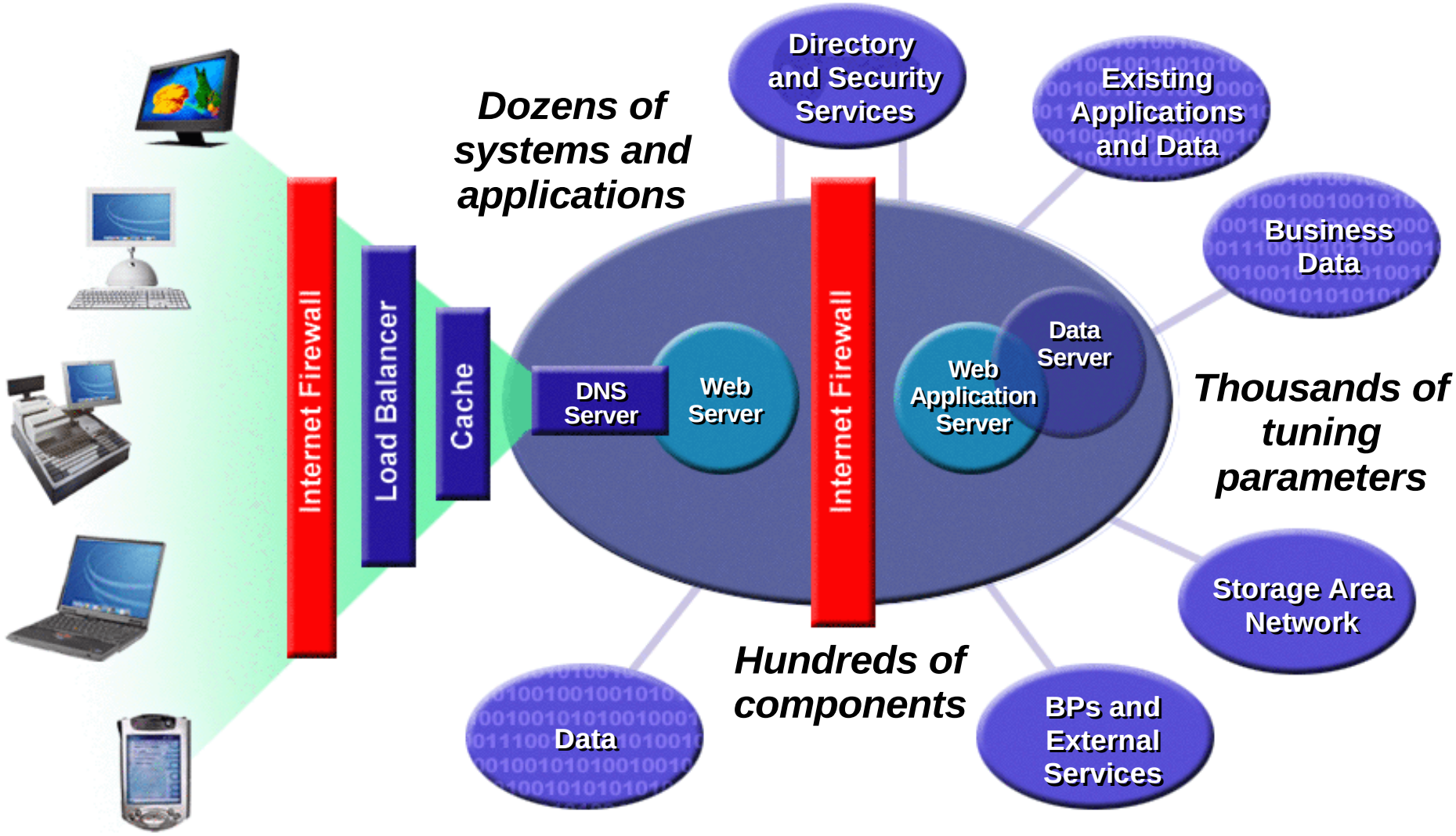
- Implementation of an autonomic management system on the PaaS to ensure SaaS layer meets SLA, energy efficiency and maintains security.

- For knowledge base, fuzzy Q-Learning for knowledge evolution. It is a self-learning. Self-adapting cloud controller that auto scales (down or up) the number of virtual machines that support the cloud.
- It uses data collected at run-time and automatically continues to tune the data in order to achieve desired goals. This is particularly favorable, when there is not enough knowledge at design time.

Autonomic cloud Computing

- Implementation of QoS autonomic information delivering system for delivering agricultural information systems to farmers. This was achieved at IaaS layer using Cuckoo optimization algorithm and fuzzy logic to attain autonomic resource allocation.
- Decentralized autonomic architecture for managing wireless sensor networks. They identified automatic operation, aware operation and adaptive operation as properties that each autonomic system must have.
- In order to address the lack of standard method of developing autonomic solution for the cloud, British Telecoms and Bournemouth University offered a studentship, titled Best Practice Design for Autonomic Applications in the Cloud.

Need for Autonomic Computing



Challenges in IT industry!

- It's complexity of applications!!!!
- As computing power has increased, we've got the ability to create much larger kinds of applications.
- With millions or tens of millions of computer systems all cooperating, this complexity comes at a cost because humans are sitting behind the scenes, making all these machines work together.

Need new Computing Strategies, Why?

- It's complexity of applications!!!!
- As computing power has increased, we've got the ability to create much larger kinds of applications.
- With millions or tens of millions of computer systems all cooperating, this complexity comes at a cost because humans are sitting behind the scenes, making all these machines work together.

Need new Computing Strategies, Why?

- In this present rapidly growing complex world, the odds to a complex computing system are very high.
- To overcome the rapid growth of complex computing systems and to reduce the barrier that complexity poses to further growth.
- IBM has initiated a vision to create self managed systems to address today's concern of complexity.
- The self-managed and self-regulated systems which are capable of making decisions on its own are known as

  "AUTONOMIC SYSTEMS"

Aspects of Self-management without and with Autonomic computing

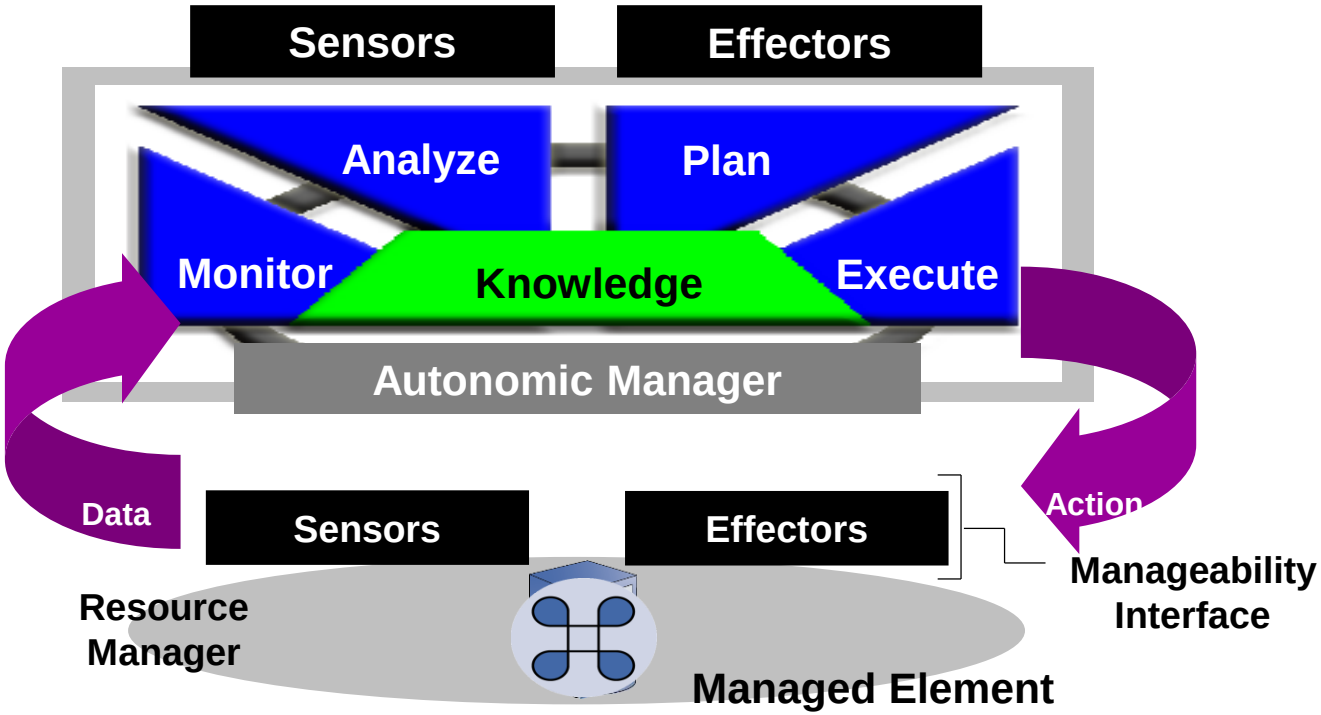
Properties of Autonomic computing	Current Computing without autonomic concept	Future computing with Autonomic Concept
Self-Configuration	Due to multiple platforms and vendors, installing configuring and maintaining systems are time consuming and error prone tasks	Automated configuration and system follows high-level policies. Rest of system adjusts automatically and seamlessly
Self-optimization	Systems have hundreds of manually set, nonlinear tuning parameters	Components and system continually seek opportunities to improve their own performance and efficiency
Self-healing	Problem determination in large complex systems can take a team of programmer weeks	System automatically detects , diagnoses and repairs localized software and hardware problems
Self-protection	Detection of recovery from attacks and cascading failure is manual	System automatically defends against malicious attacks or cascading failures. It uses early warning to anticipate and prevent system wide failure

Research Issues in Autonomic Software Development

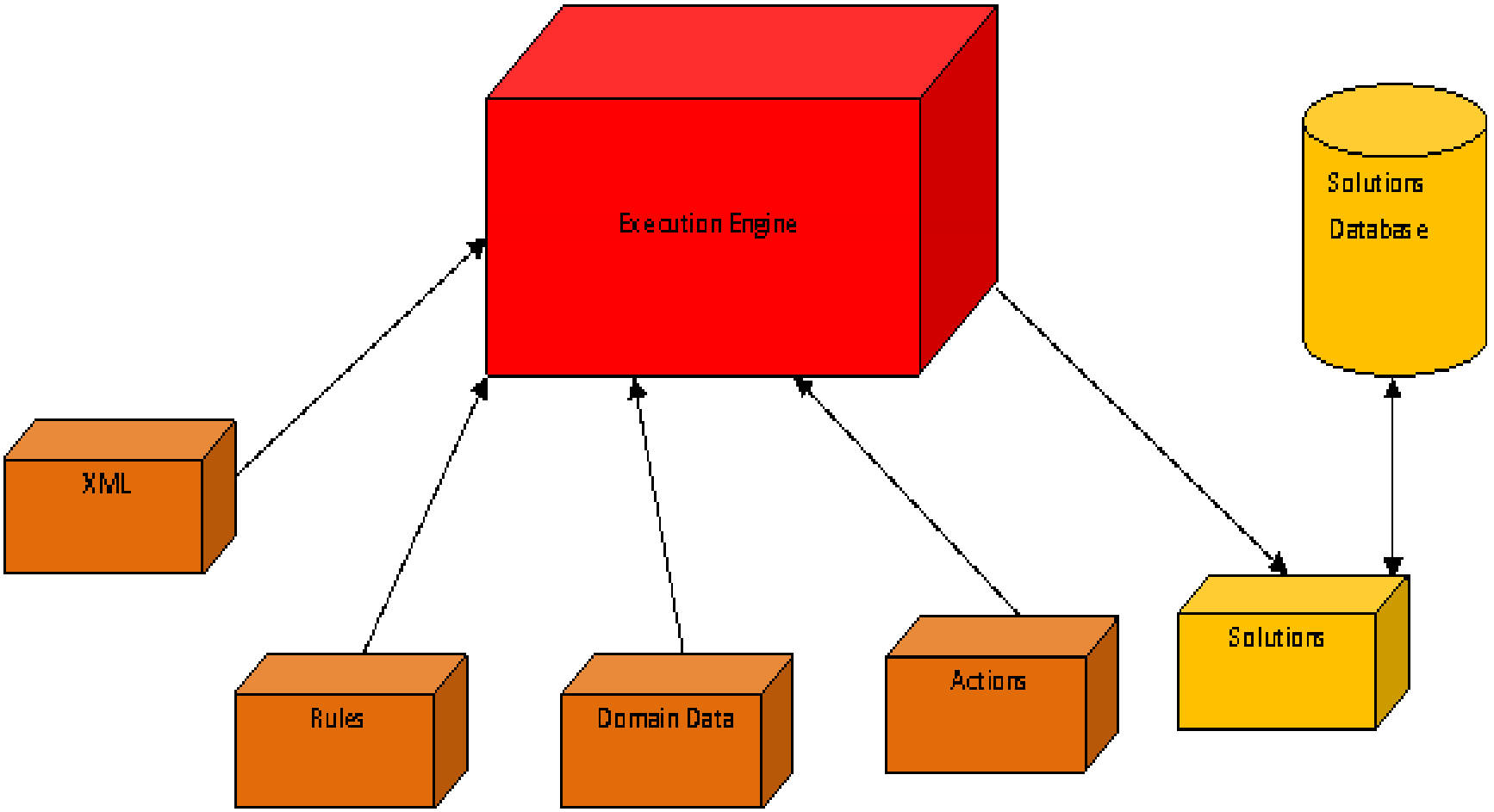
- Decision Making
- Agility – quickness (meet SLA)
- Cloud

Runtime Decision Making

- Introduce a runtime decision making (RDM)
- RDM will be based on Artificial Intelligence
- RDM will help IT systems to recover from unexpected errors



Runtime Decision Making



Runtime Decision Making

- Fuzzy Logic implementation for processing the actions
- Implementation of Intelligent Agents for learning
- Data Mining and Knowledge Discovery for getting historical data
- Incorporation of template decision trees to be used as base for creating new decision trees
- Data ware house for storing and retrieving for knowledge queries
- Artificial Neural Network (ANN) Based implementation

Development of Self-Managing Systems and Agile Methodology

- Self Managing requirements may not be clear in initial phases
- Adding extra Self Management behaviour results into higher cost in traditional SDLC models
- Customers can get early view of the benefits of Self Management features

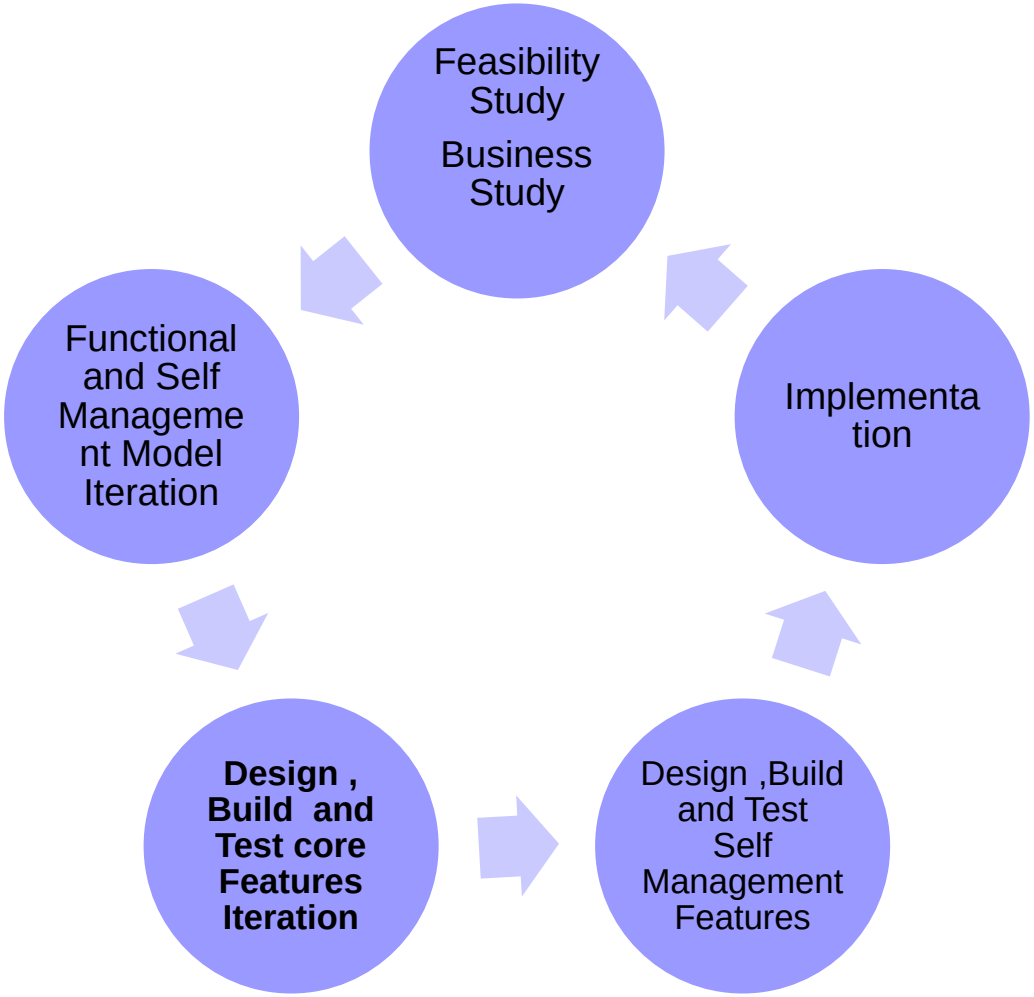
Agile SDLC’s

- Speed up or bypass one or more life cycle phases
- Usually less formal and reduced scope
- Used for time-critical applications
- Used in organizations that employ disciplined methods

Agile SDLC’s

- Speed up or bypass one or more life cycle phases
- Usually less formal and reduced scope
- Used for time-critical applications
- Used in organizations that employ disciplined methods

Agile Modeling for self-managing Systems



Autonomic through agile

- The Agile methodology is the best to create the different software components that support the change in requirements.
- Agile methodology may be the best solution for providing the Self-Managing capabilities in the system.

Autonomic Computing in Cloud

Cloud Computing

Both software applications and computing infrastructure are moved from private environments to third party data centres, and made accessible through the Internet. Cloud computing delivers infrastructure, platform, and software (applications) as subscription-based services in a pay-as-you-go model.

- Cloud computing is a style of computing paradigm in which typically real-time scalable resources such as files, data, software, hardware, and third party services can be accessible from a Web browser via the Internet to users.

Cloud Computing

- *"refers to both the applications delivered as services over the Internet, and the hardware and system software in the data centres that provide those services".*
- *"is a utility-oriented distributed computing system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resource(s) based on service-level agreements established through negotiation between the service provider and consumers"*

Cloud Computing - Some terms

Term cloud is used as a metaphor for internet.

Concept generally incorporates combinations of the following

- Infrastructure as a service (IaaS)
- Platform as a service (PaaS)
- Software as a service(SaaS)

Autonomic Computing and Cloud

Clouds are complex, large-scale, and heterogeneous distributed systems (e.g., consisting of multiple Data Centres, each containing 1000s of servers and peta-bytes of storage capacity), management is a crucial feature.

To manage it manually is very difficult.

Autonomic Computing and Cloud

Effective management of services becomes fundamental in software platforms that constitute the fabric of computing Clouds.

It needs to be automated and integrated with intelligent strategies for dynamic provisioning of resources in an autonomic manner with the services that are self managed, secure, reliable, and cost-efficient.

Autonomic Computing Benifit

- The Autonomic computing aims to provide a zero cost maintenance and highly reliable system to end user.
- Self-Management provides the monitoring, diagnosis and repair capabilities to maintain the systems' behaviour and grants the expected service. It may be a very cost effective and efficient method for cloud computing also.