An Introduction to Autonomic Computing

Daniel A. Menasce
Department of Computer Science
George Mason University

Based on the papers:

“The Vision of Autonomic Computing,” Jeff Kephart and D. Chess, IEEE Computer, January 2003.

and other papers by D.A. Menasce, his students, and colleagues.

Motivation for AC

- "…main obstacle to further progress in IT is a looming software complexity crisis." (from an IBM manifesto, Oct. 2001).
  - Tens of millions of lines of code
  - Skilled IT professionals required to install, configure, tune, and maintain.
  - Need to integrate many heterogeneous systems
  - Limit of human capacity being achieved

Motivation for AC (cont’d)

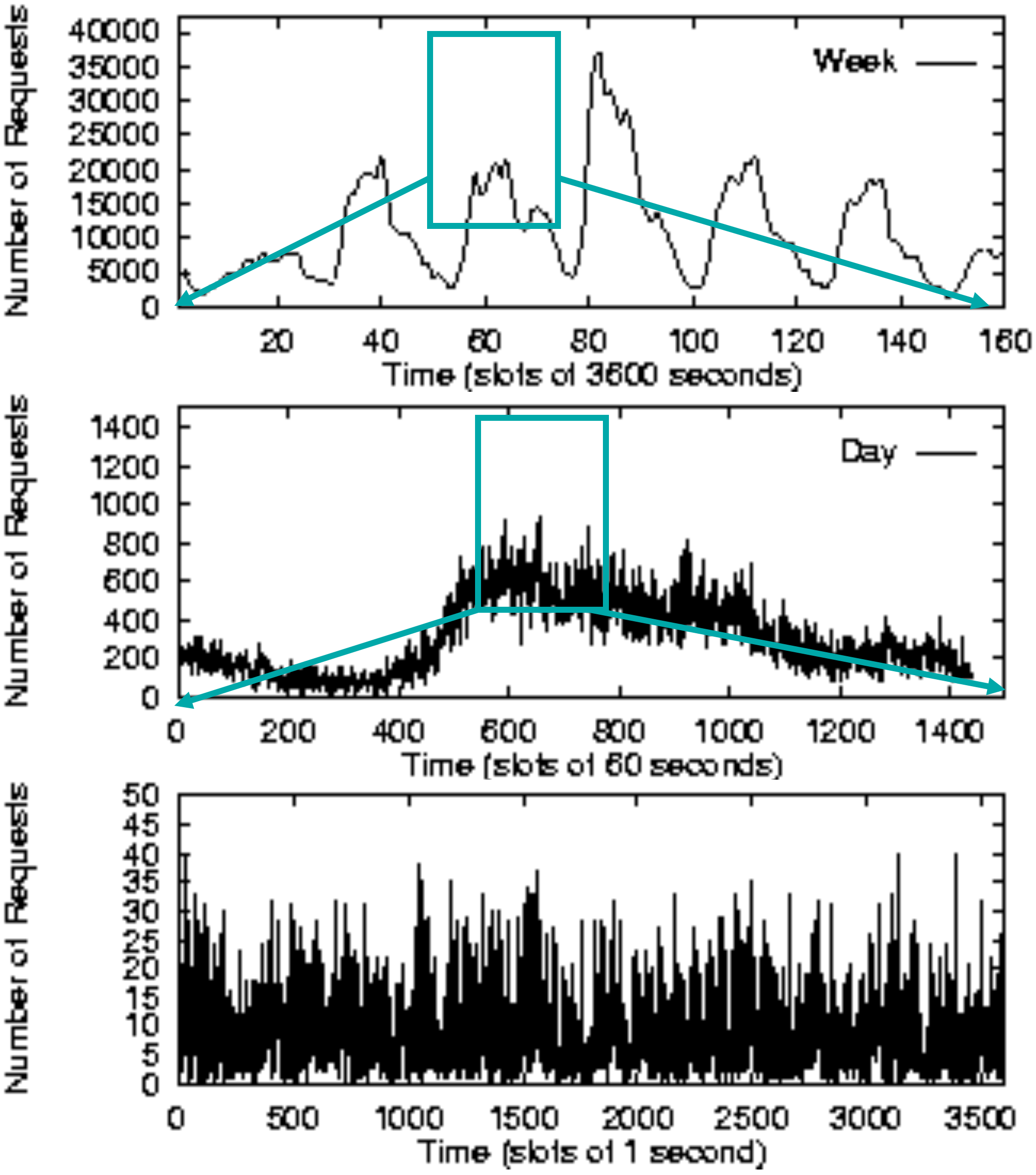
- Harder to anticipate interactions between components at design time:
  - Need to defer decisions to run time
- Computer systems are becoming too massive, complex, to be managed even by the most skilled IT professionals
- The workload and environment conditions tend to change very rapidly with time

Multi-scale time workload variation of a Web Server

3600 sec

60 sec

1 sec



Autonomic Computing

- System that can manage themselves given high-level objectives.
  - High-level objectives can be expressed in term of service-level objectives or utility functions.
- Autonomic computing is inspired in the human autonomic nervous system:
  - “The autonomic nervous system consists of sensory neurons and motor neurons that run between the central nervous system and various internal organs such as the: heart, lungs, viscera, glands. It is responsible for monitoring conditions in the internal environment and bringing about appropriate changes in them. The contraction of both smooth muscle and cardiac muscle is controlled by motor neurons of the autonomic system.” http://users.rcn.com/jkimball.ma.ultranet/BiologyPages/P/PNS.html
  - The autonomic nervous system functions in an involuntary, reflexive manner.

Evolution of AC Systems

- Automated data collection and aggregation in support of decisions by human administrators
- Provide advise to humans suggesting possible courses of action
- Take lower level actions automatically
- Increase the scope and impact of actions taken automatically by systems in support of their AC behavior
- Self-managing systems and devices will be completely natural

Autonomic Computing

- Inspiration on self-governing systems such as social and economic systems in addition to purely biological ones.
- Dimension of Self-Management:
  - Self-configuration
  - Self-optimization
  - Self-healing
  - Self-protection

Self-Configuring Systems

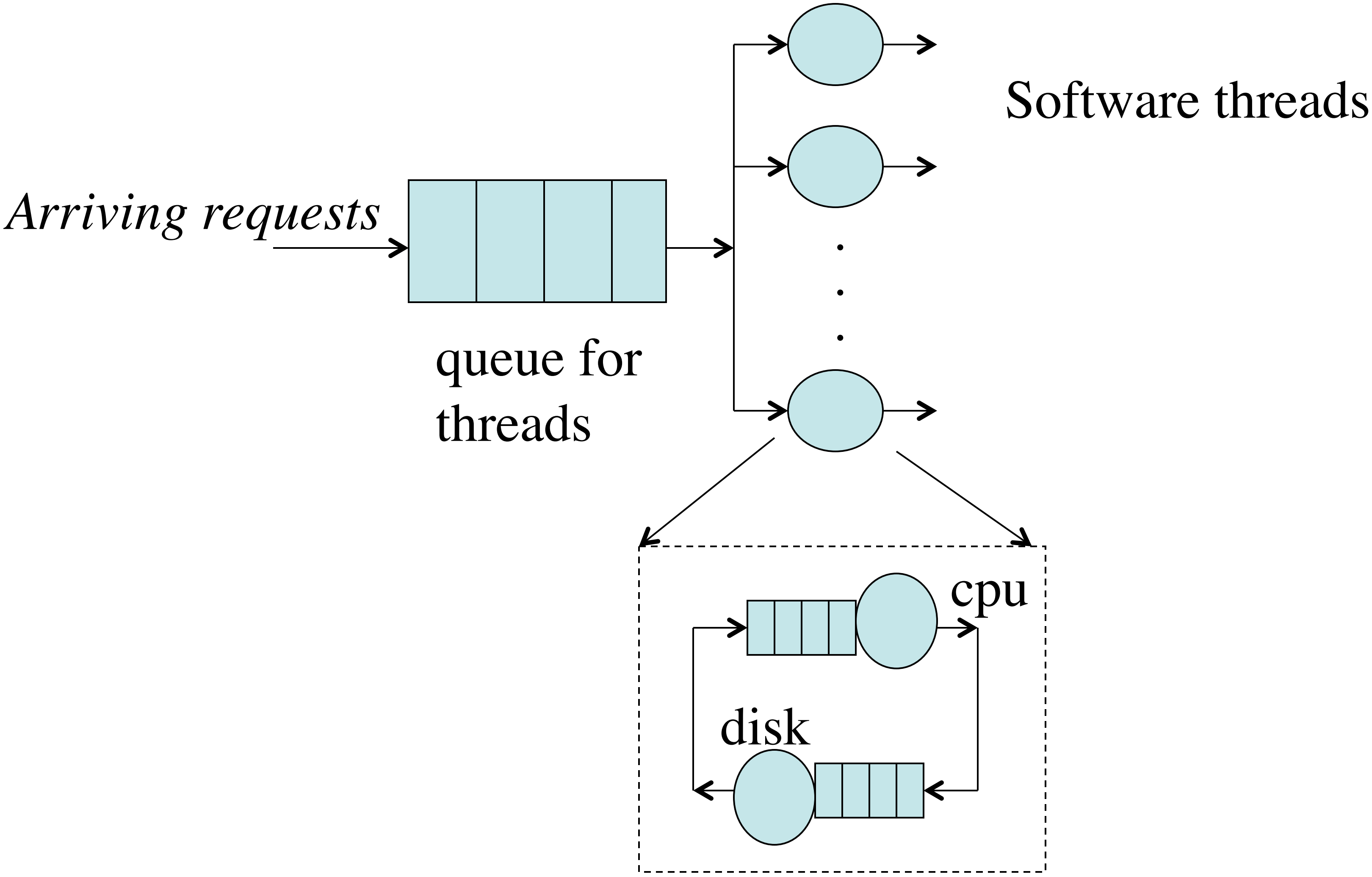
- Automatic configuration of components within larger systems
- Component registration
  - Need to advertise behavior
  - Need to advertise configuration options and mechanisms
- Automatic component discovery and integration

Self-Optimizing Systems

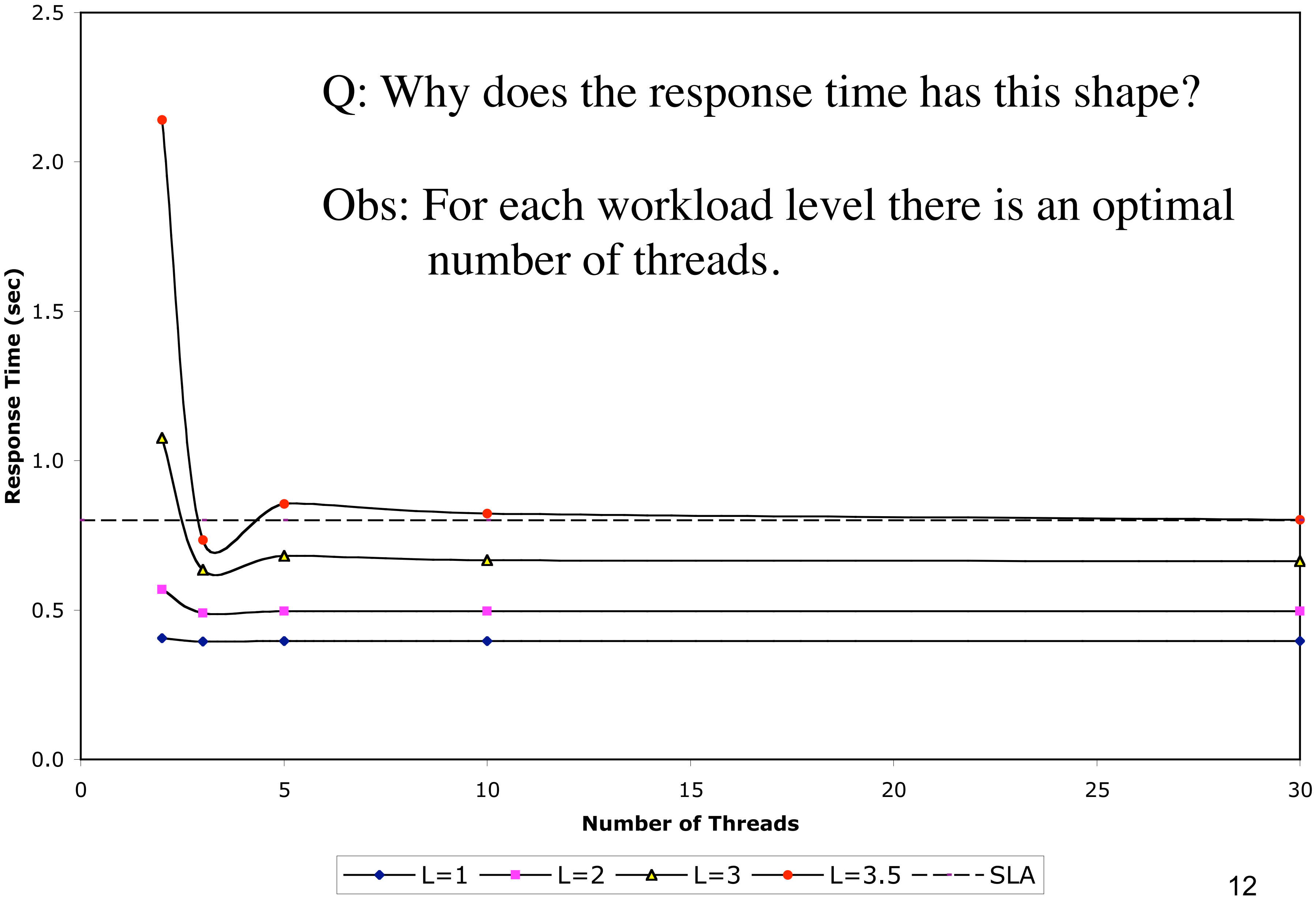
- Complex middleware and database systems have a very large number of configurable parameters.

Web Server (IIS 5.0)	**Application Server (Tomcat 4.1)**	**Database Server (SQL Server 7.0)**
HTTP KeepAlive	acceptCount	Cursor Threshold
Application Protection Level	minProcessors	Fill Factor
Connection Timeout	maxProcessors	Locks
Number of Connections		Max Worker Threads
Logging Location		Min Memory Per Query
Resource Indexing		Network Packet Size
Performance Tuning Level		Priority Boost
Application Optimization		Recovery Interval
MemCacheSize		Set Working Set Size
MaxCachedFileSize		Max Server Memory
ListenBacklog		Min Server Memory
MaxPoolThreads		User Connections
worker.ajp13.cachesize		

Self-optimizing systems: motivation



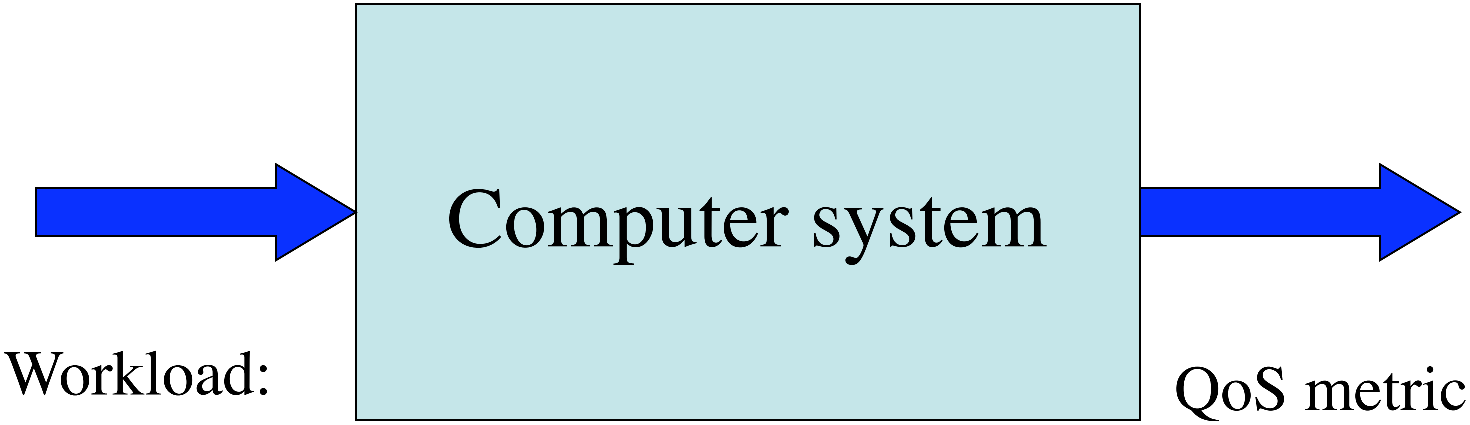
Q: How does the response time vary with the number of software threads?



Q: Why does the response time has this shape?

Obs: For each workload level there is an optimal number of threads.

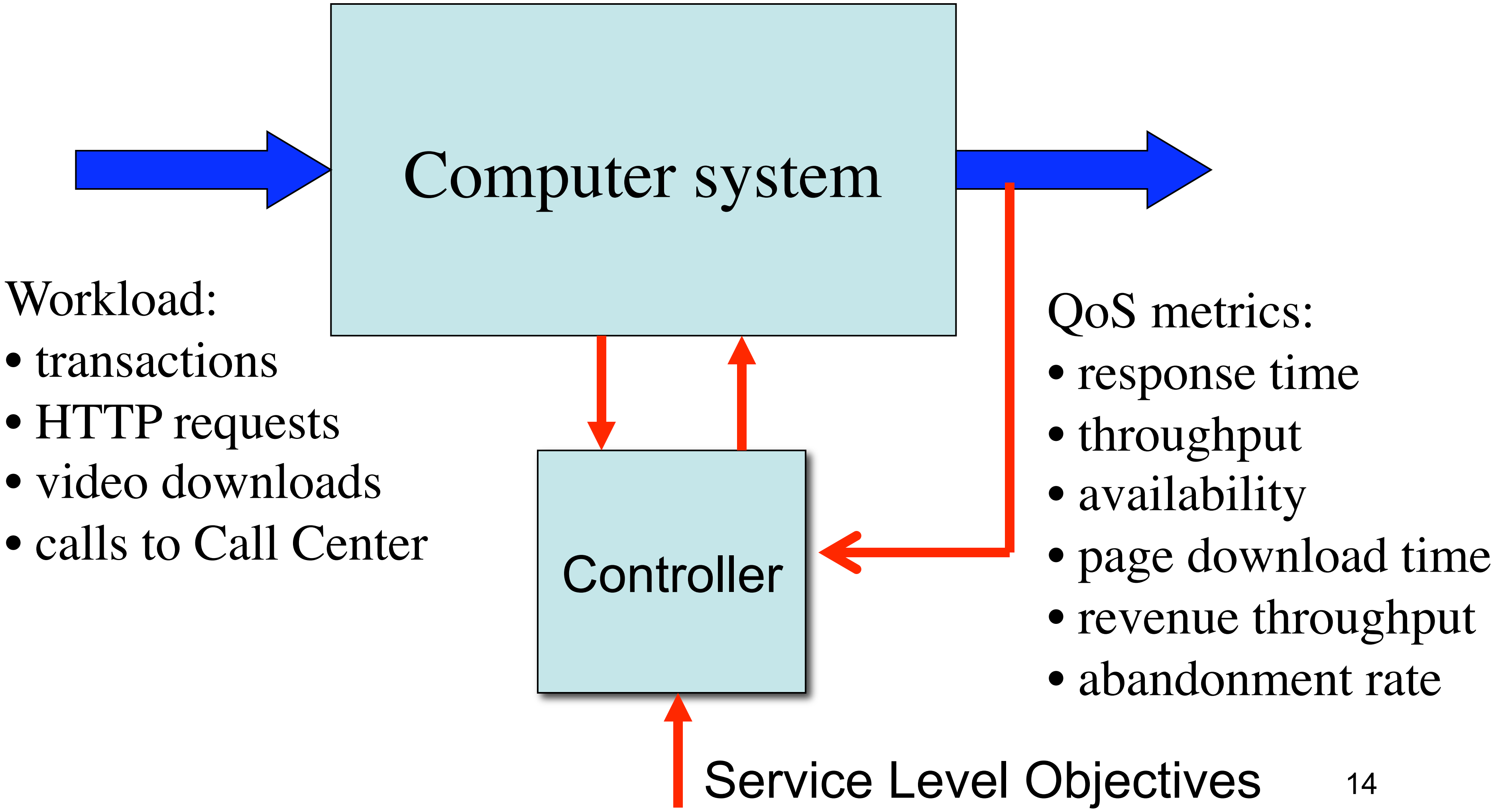
Workload and QoS metrics



Workload:
- transactions
- HTTP requests
- video downloads
- calls to Call Center

QoS metrics:
- response time
- throughput
- availability
- page download time
- revenue throughput
- abandonment rate

Self-optimizing System



Computer system

Controller

Workload:

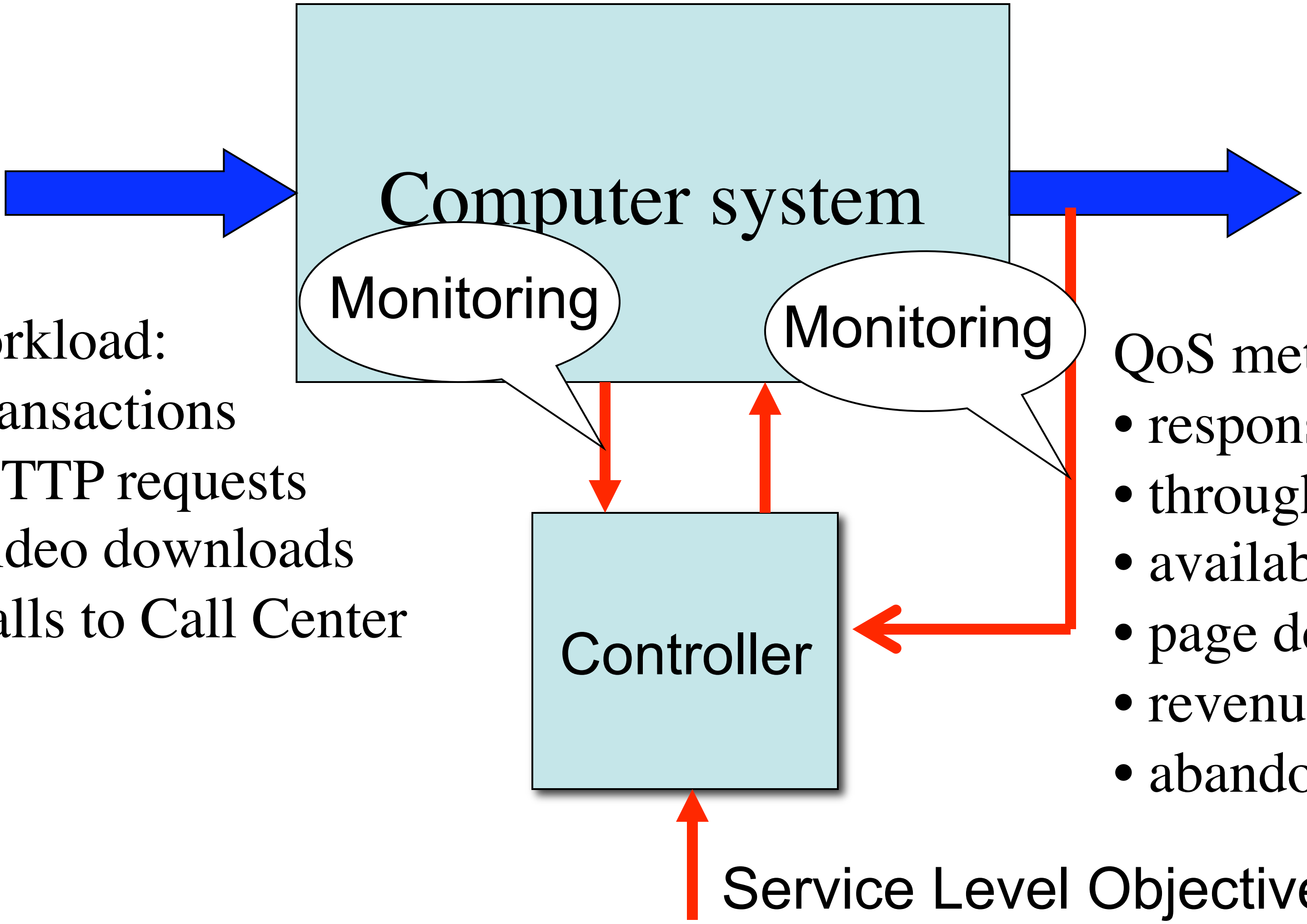
- transactions
- HTTP requests
- video downloads
- calls to Call Center

QoS metrics:

- response time
- throughput
- availability
- page download time
- revenue throughput
- abandonment rate

Service Level Objectives

Self-optimizing System



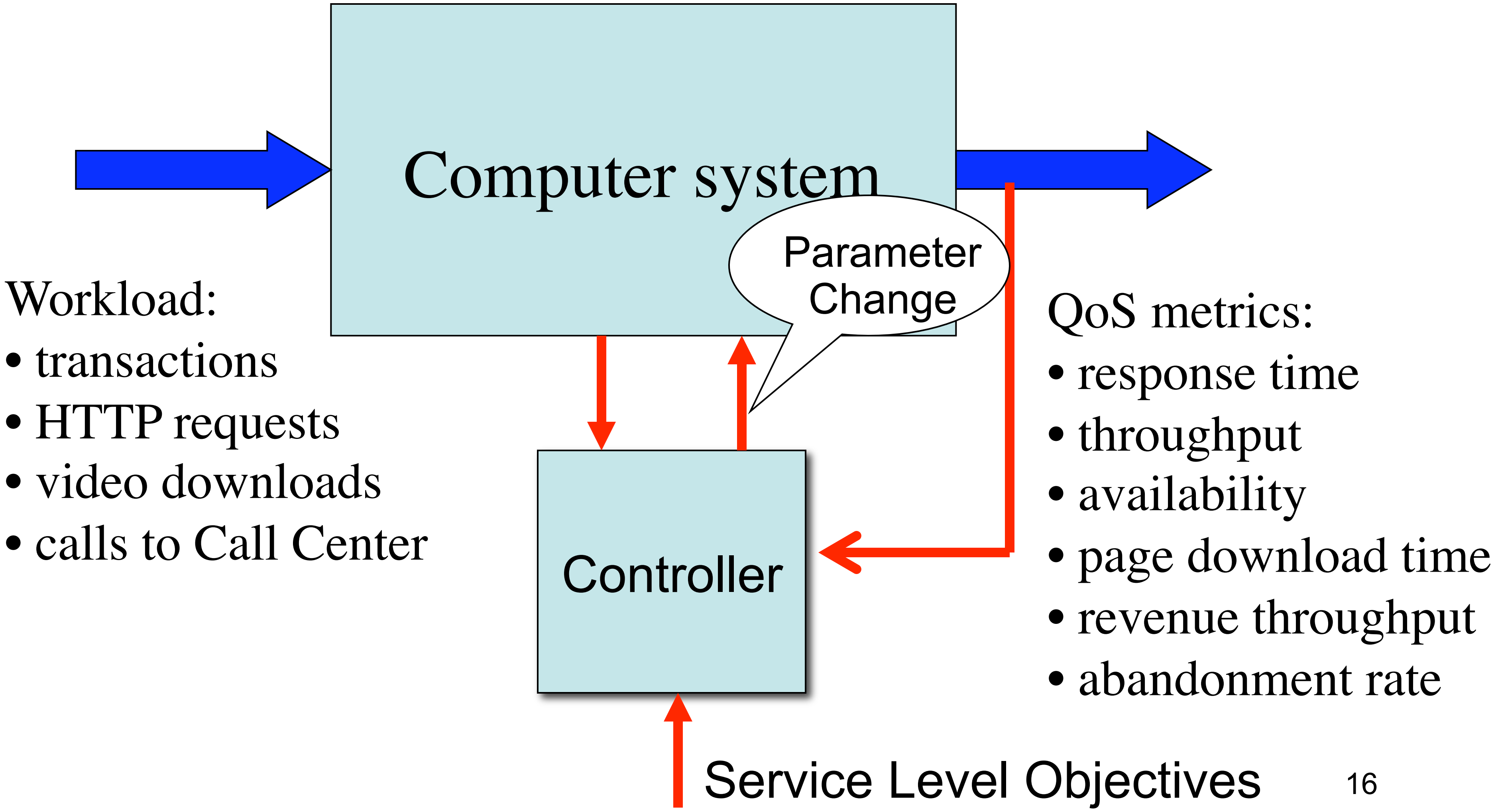
Workload:

- transactions
- HTTP requests
- video downloads
- calls to Call Center

QoS metrics:

- response time
- throughput
- availability
- page download time
- revenue throughput
- abandonment rate

Self-optimizing System



Workload:
- transactions
- HTTP requests
- video downloads
- calls to Call Center

QoS metrics:
- response time
- throughput
- availability
- page download time
- revenue throughput
- abandonment rate

Engineering with QoS in Mind

- Poor QoS may lead to loss of human life (e.g., homeland security and disaster relief support systems) and financial losses due to customer dissatisfaction.

Engineering with QoS in Mind

- Poor QoS may lead to loss of human life (e.g., homeland security and disaster relief support systems) and financial losses due to customer dissatisfaction.
- QoS has to be an integral part of the design of any computer system.

Engineering with QoS in Mind

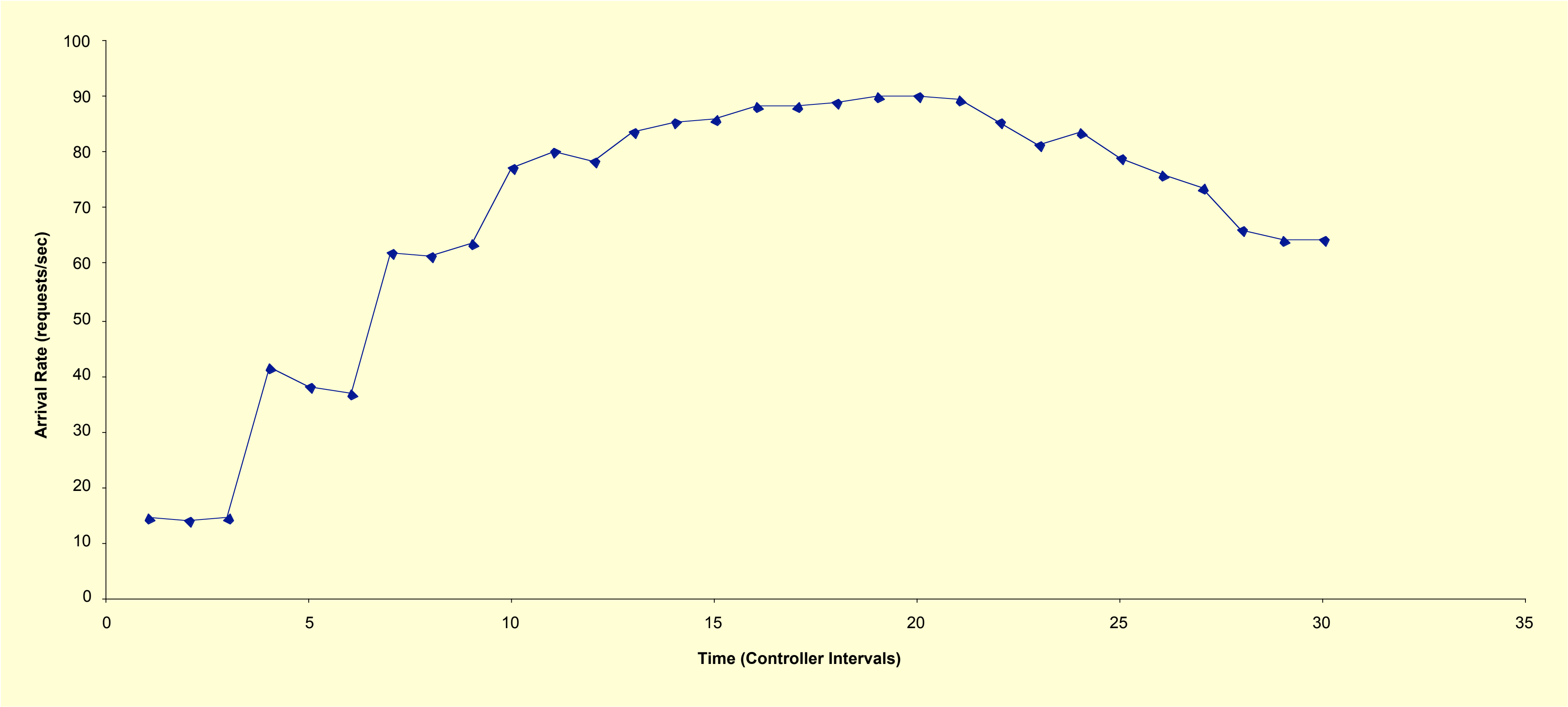
- Poor QoS may lead to loss of human life (e.g., homeland security and disaster relief support systems) and financial losses due to customer dissatisfaction.
- QoS has to be an integral part of the design of any computer system.
- Online computer systems (e.g., Web sites, e-commerce sites, call centers) have workloads that can be widely varying.

Engineering with QoS in Mind

- Poor QoS may lead to loss of human life (e.g., homeland security and disaster relief support systems) and financial losses due to customer dissatisfaction.
- QoS has to be an integral part of the design of any computer system.
- Online computer systems (e.g., Web sites, e-commerce sites, call centers) have workloads that can be widely varying.
- QoS autonomic control should be part of the design of complex online computer systems.

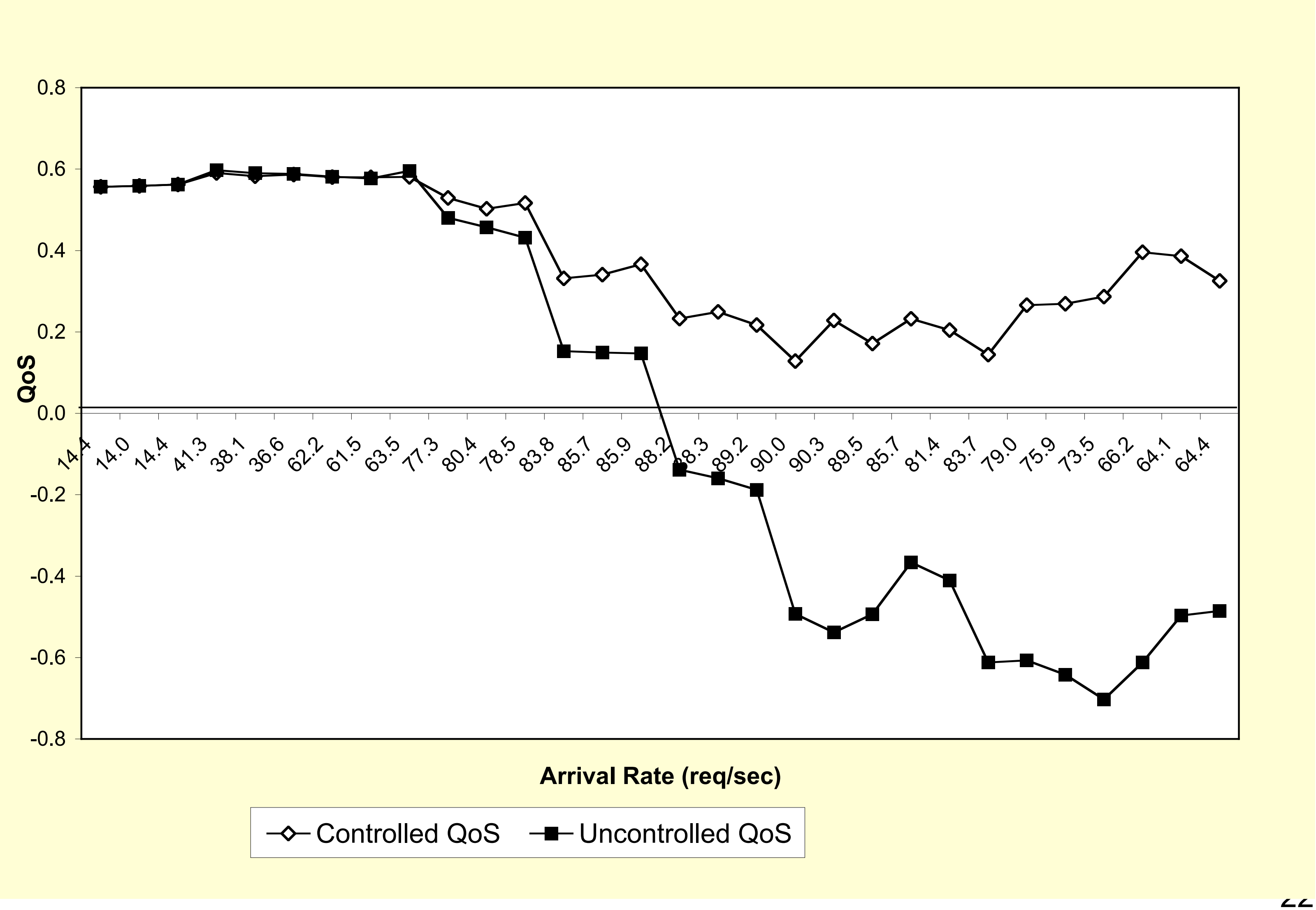
Results of a Self-Optimizing E-commerce Site

Arrival rate



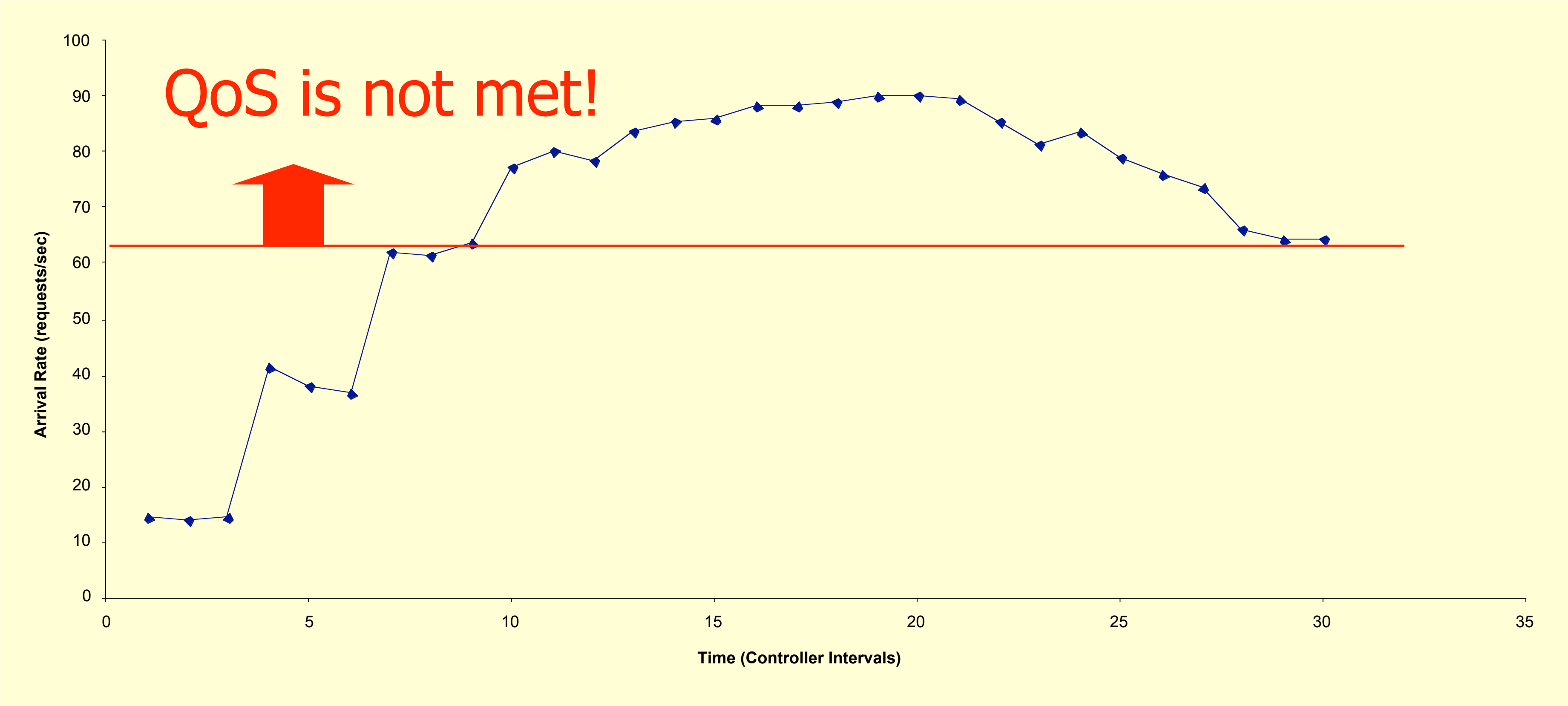
"Preserving QoS of E-commerce Sites Through Self-Tuning: A Performance Model Approach," Menasce, Dodge and Barbara, Proc. 2001 ACM Conference on E-commerce, Tampa, FL, October 14-17, 2001.

Results of QoS Controller

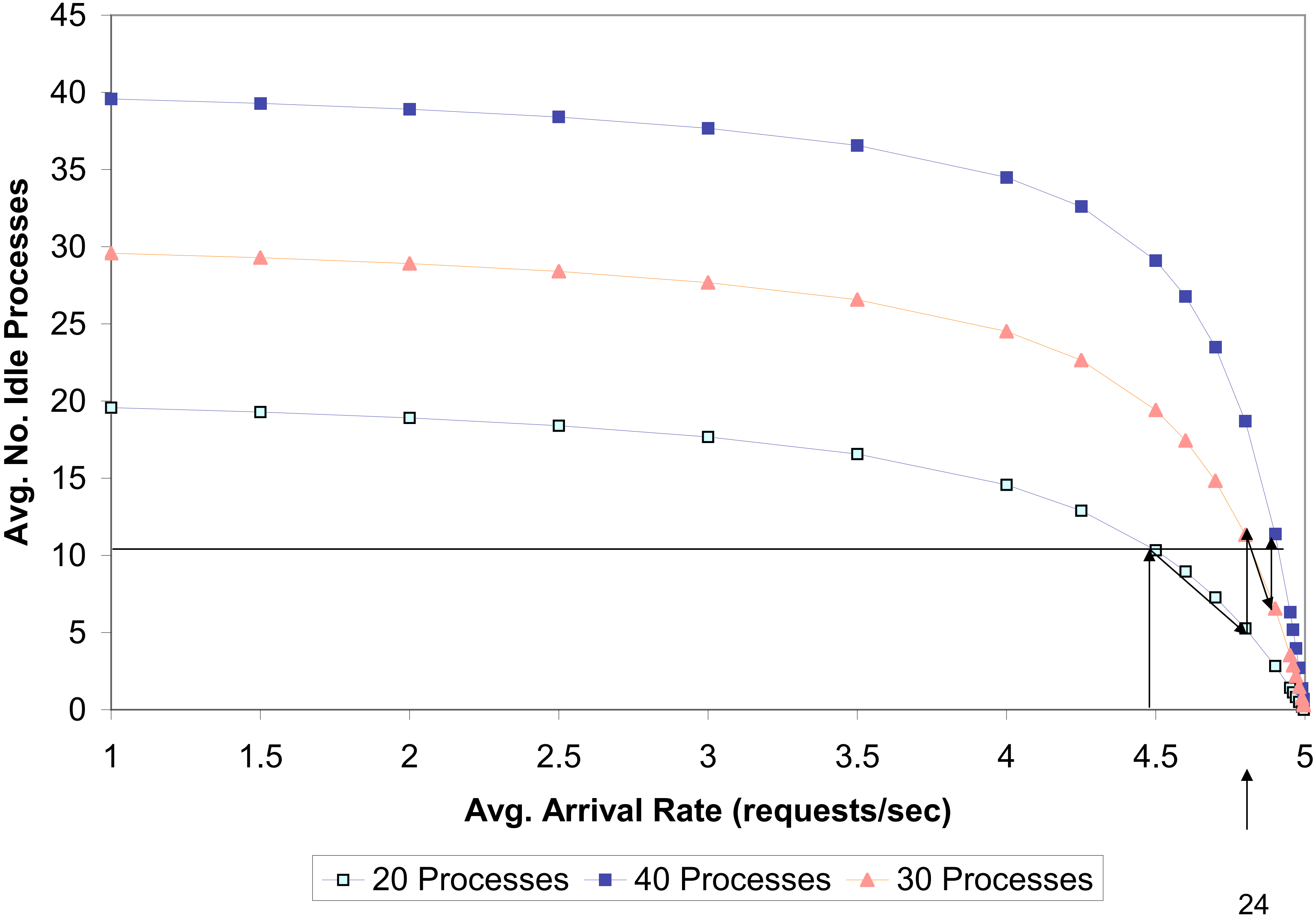


Experiment Results

Arrival rate



Dynamic Variation of Process Pool Size in Web Servers



Dynamic Variation of Pool Size

Scenario	Number of processes	Arrival Rate (req/sec)	Avg. No. Idle Processes
1	20	4.5	10.3
2	20	4.8	5.3
3	30	4.8	11.3
4	30	4.9	6.6
5	40	4.9	11.4

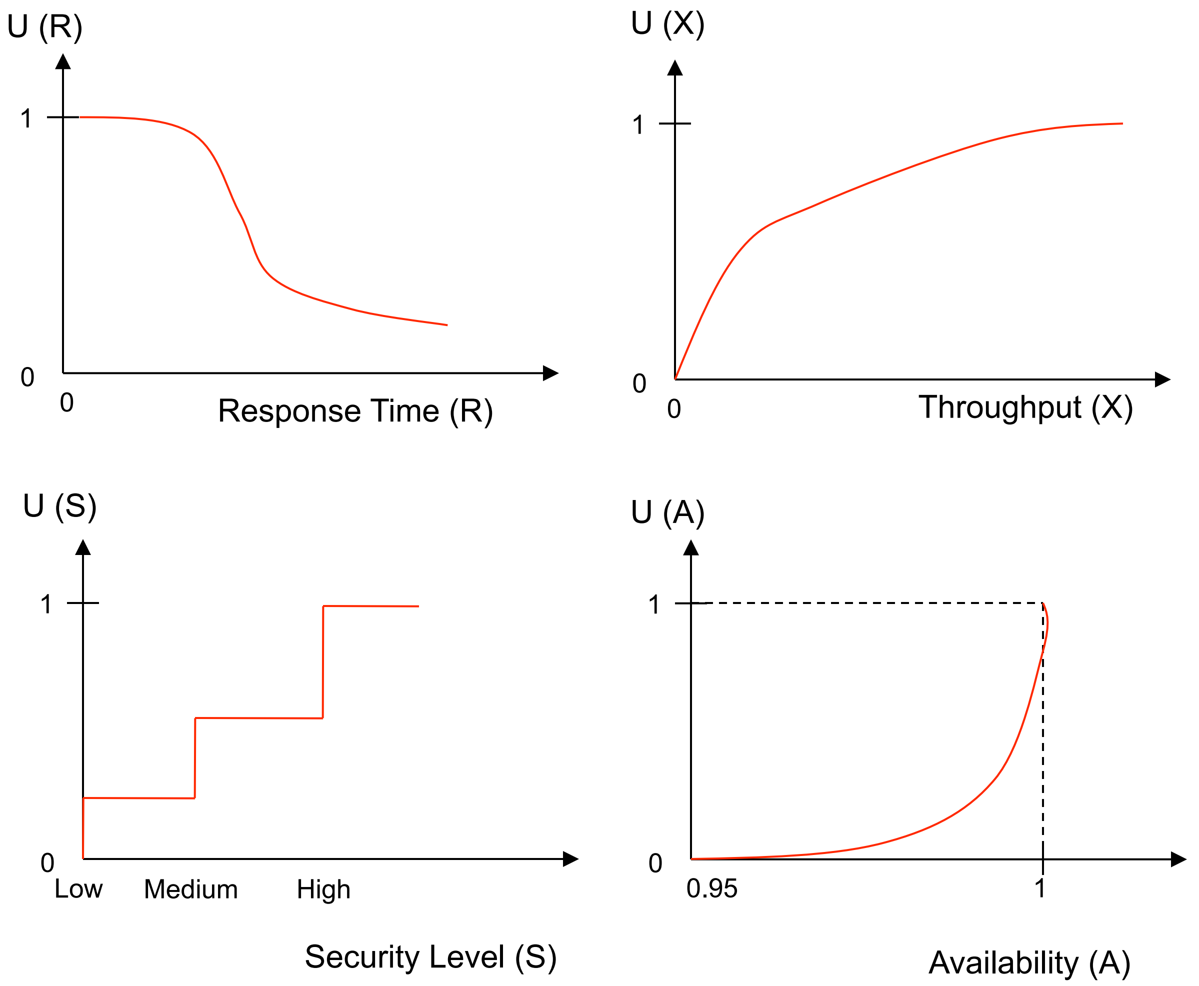
Self-Healing Systems

- Automatic mechanisms to:
  - Identify failures
  - Identify their root causes
  - Determine how to repair the system
  - Take into account complex dependencies between hardware and software failures
- Tools include:
  - Logs from various types of monitors
  - Data aggregation mechanisms
  - Statistical techniques to determine correlation between events and diagnose faults

Self-Protecting Systems

- Automatic mechanisms to:
  - Defend the system against malicious security attacks
  - Prevent security compromises to occur due to component failures
  - Predict the onset of security attacks by analyzing events recorded in various types of logs and analyzing correlations that may lead to attacks.

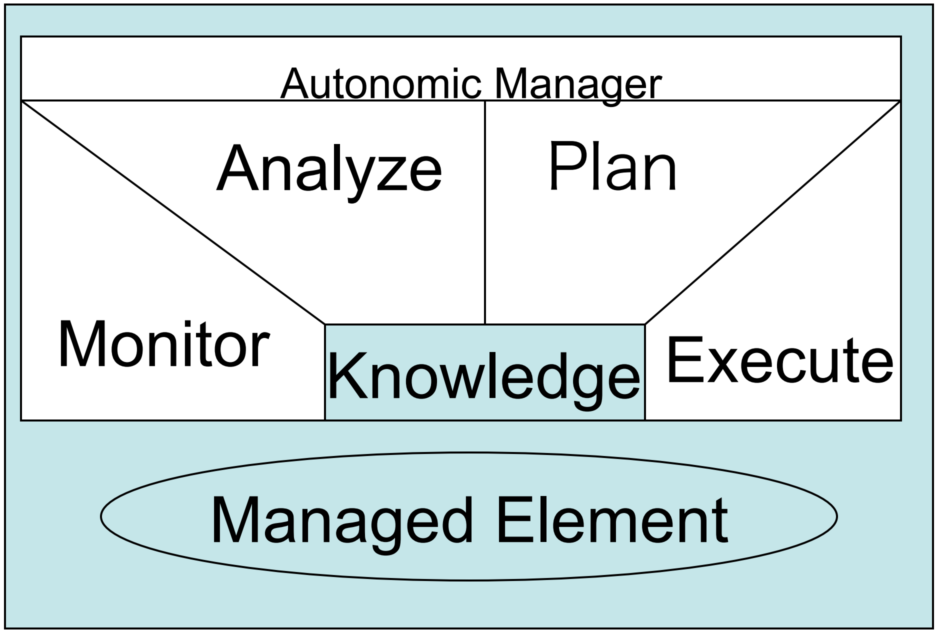
Utility Functions



$$U_g(R,X,A,S) = f(U(R),U(X),U(A),U(S))$$

Architectural Considerations

M A P E-K cycle



Autonomic Element

Architectural Considerations

- Manage internal behavior and relationships with other autonomic elements guided by human-specified policies
- Distributed service-oriented architecture is useful to support AC
  - SOA
  - Web Services
  - Grid Computing
- Autonomic Elements (AEs) may need resources from other autonomic elements
  - Need for robust and secure negotiation protocols for obtaining and releasing resources from other AEs
  - Need to monitor consumers for not overusing the AE’s resources
  - Behavior of one AE may depend on behavior of other loosely-coupled AEs
  - Need formal languages (machine and human-readable) to express service contracts and SLAs with other AEs

Engineering Challenges

- How to program AEs?
  - Need tools to acquire and represent policies
  - Need tools to translate from high-level to low-level goals
- How to test AEs?
  - Repeatability issues
  - Interconnected AEs
  - AE interconnection and binding determined at run-time
  - Difficult to put together controlled test environments
- Installation and configuration issues
  - Need directory services and brokers that locate AEs based on policies and SLAs
- Monitoring and problem determination
  - Continuously monitor suppliers to ensure compliance with SLAs and policies
  - Monitoring of consumers
  - Need statistical and aggregation techniques to be able to cope with huge amounts of data

Engineering Challenges

- Relationship with other AEs
  - Interoperability can be achived through ontologies
  - Need to assess reliability and trustworthiness of other AEs
  - Negotiation with other AEs:
    - Demand-for-service
    - FCFS
    - Posted-price
    - Bi-lateral or multi-lateral negotiations over multiple attributes
    - Third-party arbiter running auctions
- Provisioning of resources after agreements are achieved
- Monitoring to check for compliance
- Security and privacy issues
- Robustness with respect to erroneous/not feasible policies.
- Mapping from high-level objectives to low-level ones.