Trường Đại học Khoa Học Tự Nhiên
Khoa Công Nghệ Thông Tin
Bộ môn Công Nghệ Phần Mềm

CTT526 - Kiến trúc phần mềm

Các tiêu chí và yêu cầu về Kiến trúc phần mềm

PGS.TS. Trần Minh Triết
tmtriet@fit.hcmus.edu.vn

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

- Nội dung của bài giảng sử dụng:

  Session 3: Quality Attributes

  trong bộ slide Software Architecture Essential

  của GS. Ian Gorton

  Software Engineering Institute

  Carnegie Mellon University

What are Quality Attributes

- Often know as –ilities
  - Reliability
  - Availability
  - Portability
  - Scalability
  - Performance (!)
- Part of a system's NFRs
  - "how" the system achieves its functional requirements

Quality Attribute Specification

- Architects are often told:
  - "My application must be fast/secure/scale"
- Far too imprecise to be any use at all
- Quality attributes (QAs) must be made precise/measurable for a given system design, e.g.
  - *"It must be possible to scale the deployment from an initial 100 geographically dispersed user desktops to 10,000 without an increase in effort/cost for installation and configuration."*

Quality Attribute Specification

- QA's must be concrete
- But what about testable?
  - Test scalability by installing system on 10K desktops?
- Often careful analysis of a proposed solution is all that is possible
- "It's all talk until the code runs"

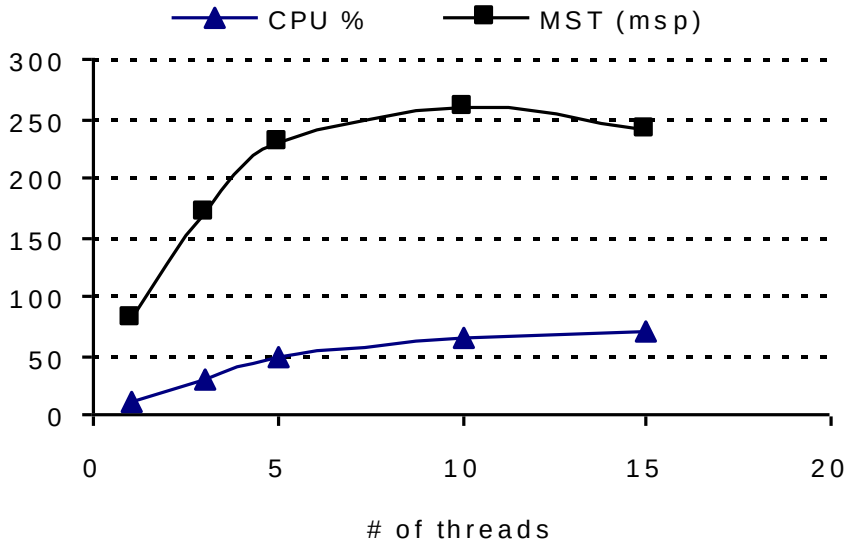
Performance

- Many examples of poor performance in enterprise applications
- Performance requires a:
  - Metric of amount of work performed in unit time
  - Deadline that must be met
- Enterprise applications often have strict performance requirements, e.g.
  - 1000 transactions per second
  - 3 second average latency for a request

Performance - Throughput

- Measure of the amount of work an application must perform in unit time
  - Transactions per second
  - Messages per minute
- Is required throughput:
  - Average?
  - Peak?
- Many system have low average but high peak throughput requirements

Throughput Example



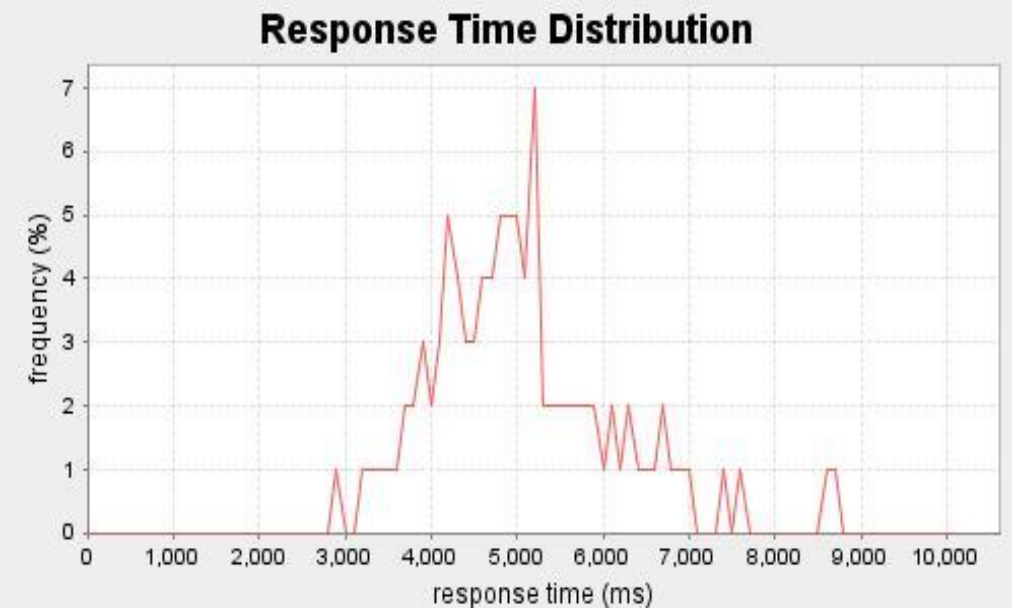
- Throughput of a message queuing system
  - Messages per second (msp)
  - Maximum sustainable throughput (MST)
- Note throughput changes as number of receiving threads increases

Performance - Response Time

- measure of the latency an application exhibits in processing a request
- Usually measured in (milli)seconds
- Often an important metric for users
- Is required response time:
  - Guaranteed?
  - Average?
- E.g. 95% of responses in sub-4 seconds, and all within 10 seconds

Response Time

- Example shows response time distribution for a J2EE application



Performance - Deadlines

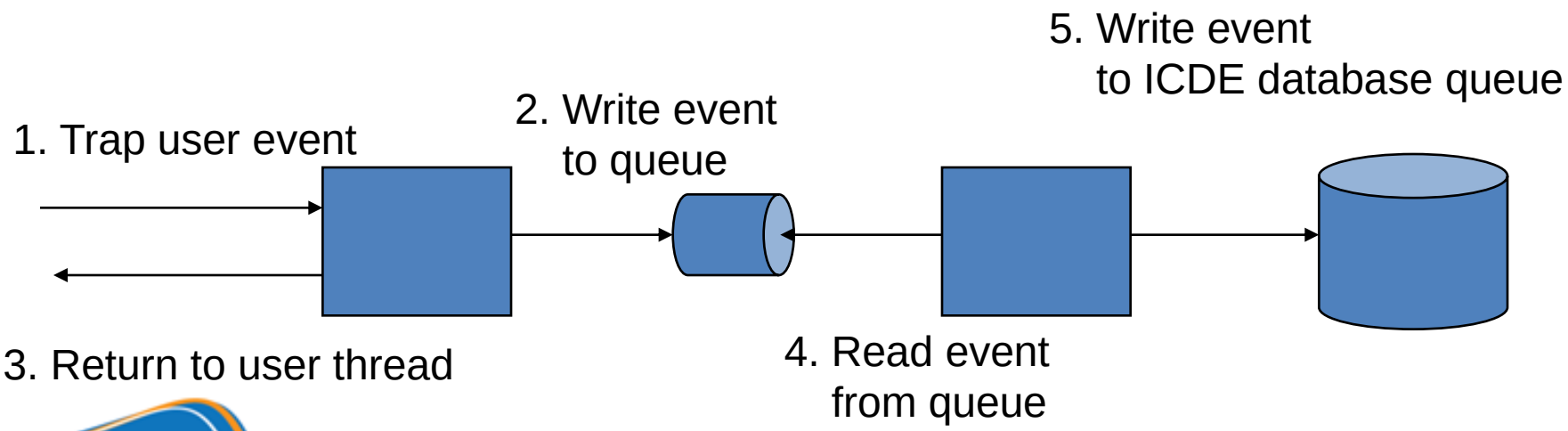
- ‘something must be completed before some specified time’
  - Payroll system must complete by 2am so that electronic transfers can be sent to bank
  - Weekly accounting run must complete by 6am Monday so that figures are available to management
- Deadlines often associated with batch jobs in IT systems.

Something to watch for …

- What is a
  - Transaction?
  - Message?
  - Request?
- All are application specific measures.
- System must achieve 100 mps throughput
  - BAD!!
- System must achieve 100 mps peak throughput for *PaymentReceived* messages
  - GOOD!!!

ICDE Performance Issues

- Response time:
  - Overheads of trapping user events must be imperceptible to ICDE users
- Solution for ICDE client:
  - Decouple user event capture from storage using a queue



Scalability

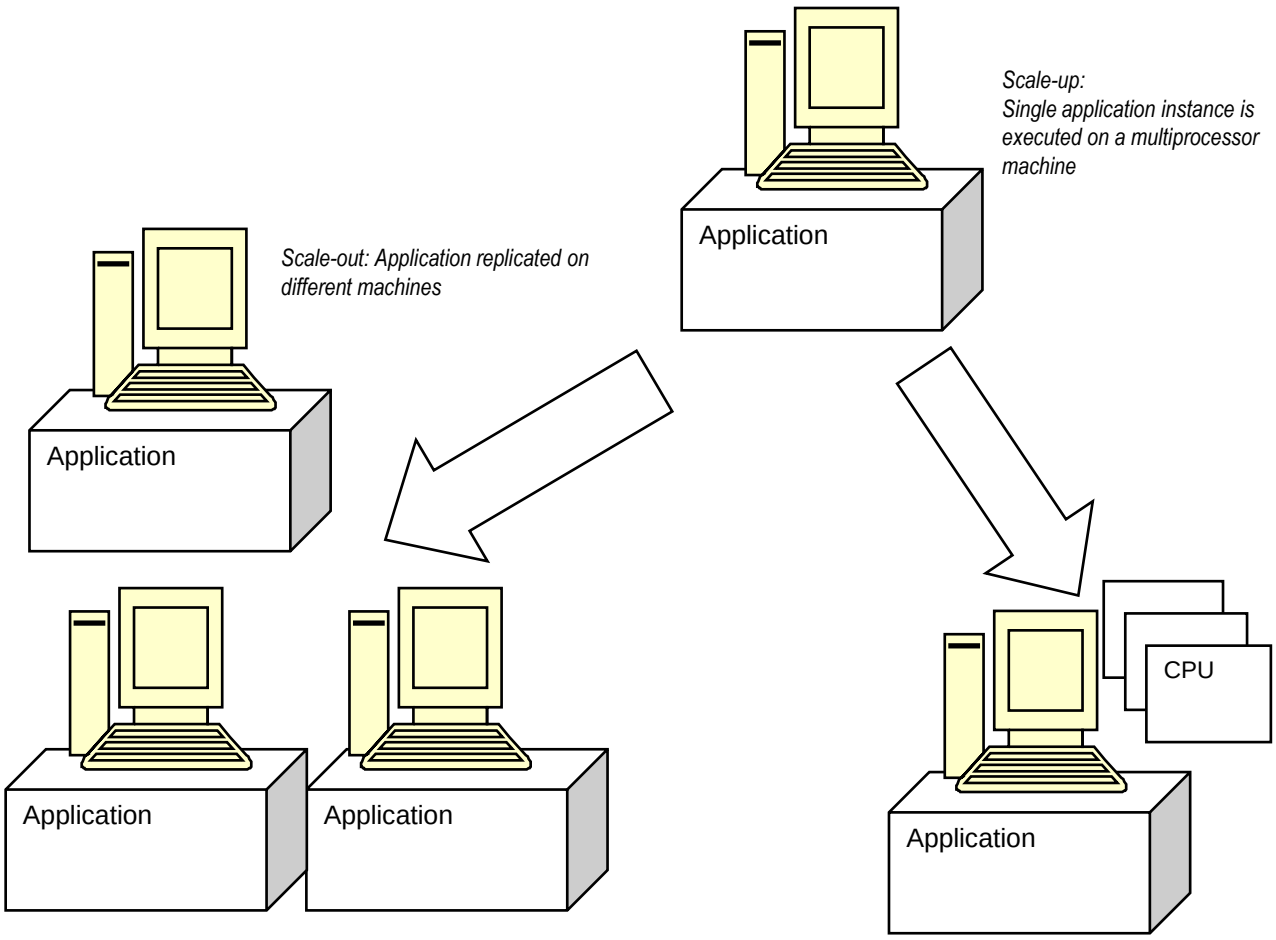
- *“How well a solution to some problem will work when the size of the problem increases.”*
- 4 common scalability issues in IT systems:
  - Request load
  - Connections
  - Data size
  - Deployments

Scalability – Request Load

- How does an 100 tps application behave when simultaneous request load grows? E.g.
  - From 100 to 1000 requests per second?
- Ideal solution, without additional hardware capacity:
  - as the load increases, throughput remains constant (i.e. 100 tps), and response time per request increases only linearly (i.e. 10 seconds).

Scalability – Add more hardware

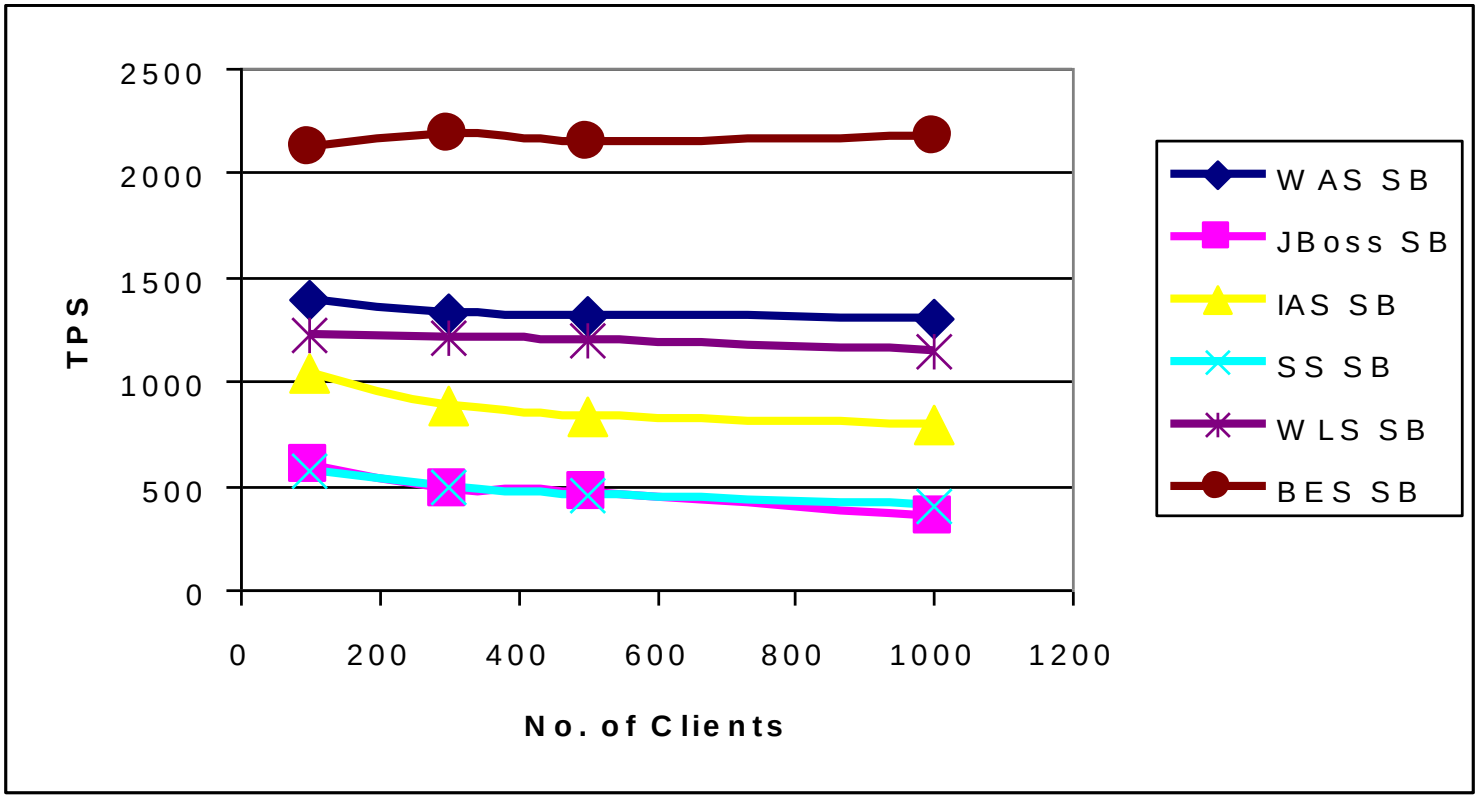
...



Scalability - reality

- Adding more hard ware should improve performance:
  - scalability must be achieved without modifications to application architecture
- Reality as always is different!
- Applications will exhibit a decrease in throughput and a subsequent exponential increase in response time.
  - increased load causes increased contention for resources such as CPU, network and memory
  - each request consumes some additional resource (buffer space, locks, and so on) in the application, and eventually these are exhausted

Scalability – J2EE example



I.Gorton, A Liu, *Performance Evaluation of Alternative Component Architectures for Enterprise JavaBean Applications, in IEEE Internet Computing, vol.7, no. 3, pages 18-23, 2003.*

Scalability - connections

- What happens if number of simultaneous connections to an application increases
  - If each connection consumes a resource?
  - Exceed maximum number of connections?
- ISP example:
  - Each user connection spawned a new process
  - Virtual memory on each server exceeded at 2000 users
  - Needed to support 100Ks of users
  - Tech crash ….

Scalability – Data Size

- How does an application behave as the data it processes increases in size?
  - Chat application sees average message size double?
  - Database table size grows from 1 million to 20 million rows?
  - Image analysis algorithm processes images of 100MB instead of 1MB?
- Can application/algorithms scale to handle increased data requirements?

Scalability - Deployment

- How does effort to install/deploy an application increase as installation base grows?
  - Install new users?
  - Install new servers?
- Solutions typically revolve around automatic download/installation
  - E.g. downloading applications from the Internet

Scalability thoughts and ICDE

- Scalability often overlooked.
  - Major cause of application failure
  - Hard to predict
  - Hard to test/validate
  - Reliance on proven designs and technologies is essential
- For ICDE - application should be capable of handling a peak load of 150 concurrent requests from ICDE clients.
  - Relatively easy to simulate user load to validate this

Modifiability

- Modifications to a software system during its lifetime are a fact of life.
- Modifiable systems are easier to change/evolve
- Modifiability should be assessed in context of how a system is likely to change
  - No need to facilitate changes that are highly unlikely to occur
  - Over-engineering!

Modifiability

- Modifiability measures how easy it **may** be to change an application to cater for new (non-) functional requirements.
  - **'may'** – nearly always impossible to be certain
  - Must estimate cost/effort
- Modifiability measures are only relevant in the context of a given architectural solution.
  - Components
  - Relationships
  - Responsibilities

Modifiability Scenarios

- Provide access to the application through firewalls in addition to existing “behind the firewall” access.
- Incorporate new features for self-service check-out kiosks.
- The COTS speech recognition software vendor goes out of business and we need to replace this component.
- The application needs to be ported from Linux to the Microsoft Windows platform.

Modifiability Analysis

- Impact is rarely easy to quantify
- The best possible is a:
  - Convincing impact analysis of changes needed
  - A demonstration of how the solution can accommodate the modification without change.
- Minimizing dependencies increases modifiability
  - Changes isolated to single components likely to be less expensive than those that cause ripple effects across the architecture.

Modifiability for ICDE

- The range of events trapped and stored by the ICDE client to be expanded.
- Third party tools to communicate new message types.
- Change database technology used
- Change server technology used

Security

- Difficult, specialized quality attribute:
  - Lots of technology available
  - Requires deep knowledge of approaches and solutions
- Security is a multi-faceted quality …

Security

- **Authentication:** Applications can verify the identity of their users and other applications with which they communicate.
- **Authorization:** Authenticated users and applications have defined access rights to the resources of the system.
- **Encryption:** The messages sent to/from the application are encrypted.
- **Integrity:** This ensures the contents of a message are not altered in transit.
- **Non-repudiation:** The sender of a message has proof of delivery and the receiver is assured of the sender's identity. This means neither can subsequently refute their participation in the message exchange.

Security Approaches

- SSL
- PKI
- Web Services security
- JAAS
- Operating system security
- Database security
- Etc etc

ICDE Security Requirements

- Authentication of ICDE users and third party ICDE tools to ICDE server
- Encryption of data to ICDE server from 3rd party tools/users executing remotely over an insecure network

Availability

- Key requirement for most IT applications
- Measured by the proportion of the required time it is useable. E.g.
  - 100% available during business hours
  - No more than 2 hours scheduled downtime per week
  - 24x7x52 (100% availability)
- Related to an application's reliability
  - Unreliable applications suffer poor availability

Availability

- Period of loss of availability determined by:
  - Time to detect failure
  - Time to correct failure
  - Time to restart application
- Strategies for high availability:
  - Eliminate single points of failure
  - Replication and failover
  - Automatic detection and restart
- Recoverability (e.g. a database)
  - the capability to reestablish performance levels and recover affected data after an application or system failure

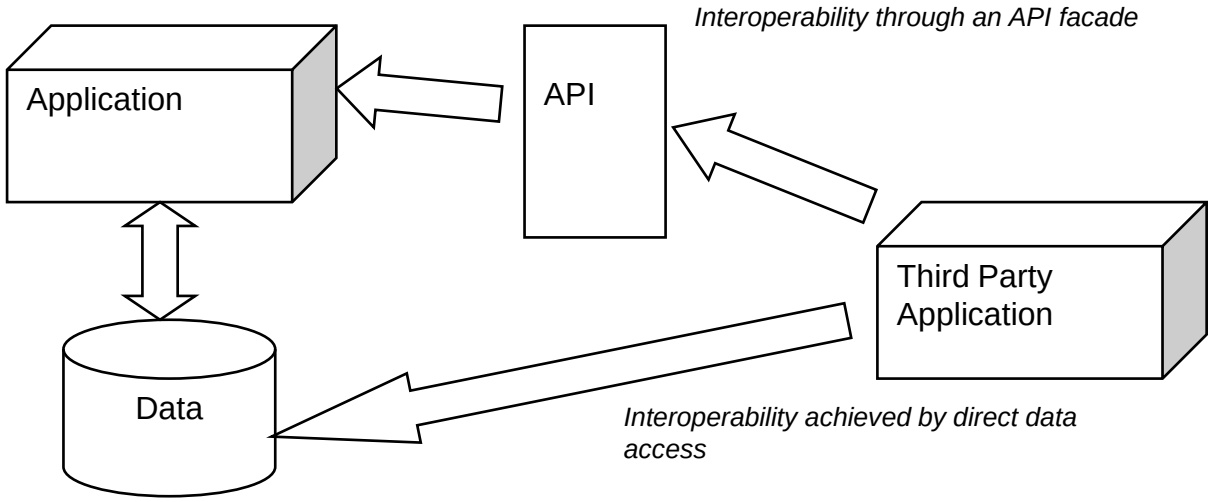
Availability for ICDE

- Achieve 100% availability during business hours
- Plenty of scope for downtime for system upgrade, backup and maintenance.
- Include mechanisms for component replication and failover

Integration

- ease with which an application can be incorporated into a broader application context
  - Use component in ways that the designer did not originally anticipate
- Typically achieved by:
  - Programmatic APIs
  - Data integration

Integration Strategies



- Data – expose application data for access by other components
- API – offers services to read/write application data through an abstracted interface
- Each has strengths and weaknesses …

ICDE Integration Needs

- Revolve around the need to support third party analysis tools.
- Well-defined and understood mechanism for third party tools to access data in the ICDE data store.

Misc. Quality Attributes

- Portability
  - Can an application be easily executed on a different software/hardware platform to the one it has been developed for?
- Testability
  - How easy or difficult is an application to test?
- Supportability
  - How easy an application is to support once it is deployed?

Design Trade-offs

- QAs are rarely orthogonal
  - They interact, affect each other
  - highly secure system may be difficult to integrate
  - highly available application may trade-off lower performance for greater availability
  - high performance application may be tied to a given platform, and hence not be easily portable
- Architects must create solutions that makes sensible design compromises
  - not possible to fully satisfy all competing requirements
  - Must satisfy all stakeholder needs
  - This is the difficult bit!

Summary

- QAs are part of an application's non-functional requirements
- Many QAs
- Architect must decide which are important for a given application
  - Understand implications for application
  - Understand competing requirements and trade-offs

Selected Further Reading

- L. Chung, B. Nixon, E. Yu, J. Mylopoulos, (Editors). Non-Functional Requirements in Software Engineering Series: The Kluwer International Series in Software Engineering. Vol. 5, Kluwer Academic Publishers. 1999.
- J. Ramachandran. Designing Security Architecture Solutions. Wiley & Sons, 2002.
- I.Gorton, L. Zhu. *Tool Support for Just-in-Time Architecture Reconstruction and Evaluation: An Experience Report*. International Conference on Software Engineering (ICSE) 2005, St Loius, USA, ACM Press