Quality Attributes

March 2014

Ying SHEN

SSE, Tongji University

Lecture objectives

This lecture will enable students to

- be familiar with different system qualities and examples through a real project (ICDE system).

Architecture and requirements

All requirements encompass the following categories:

- Functional requirements
- Quality attribute requirements
- Constraints

Architecture and functionality

Functionality is the ability of the system to do the work for which it was intended.

Functionality does not determine architecture.

- If functionality is the only requirement, a single monolithic module works!

Although functionality is independent of structure, it is achieved by assigning responsibilities to architectural elements.

Quality attributes

A **quality attribute** (QA) is a measurable or testable property of a system that is used to indicate how well the system satisfies the needs of its stakeholders.

A quality attribute can be regarded as measuring the "goodness" of a product along some dimension of interest to a stakeholder.

Quality attributes

QAs specify how well the system performs its functions:

- How fast must it respond?
- How easy must it be to use?
- How secure does it have to be against attacks?
- How easy should it be to maintain?

Software qualities are not orthogonal.

- A change in structure that improves one quality often affects the other qualities.

A Case Study in Quality Attribute

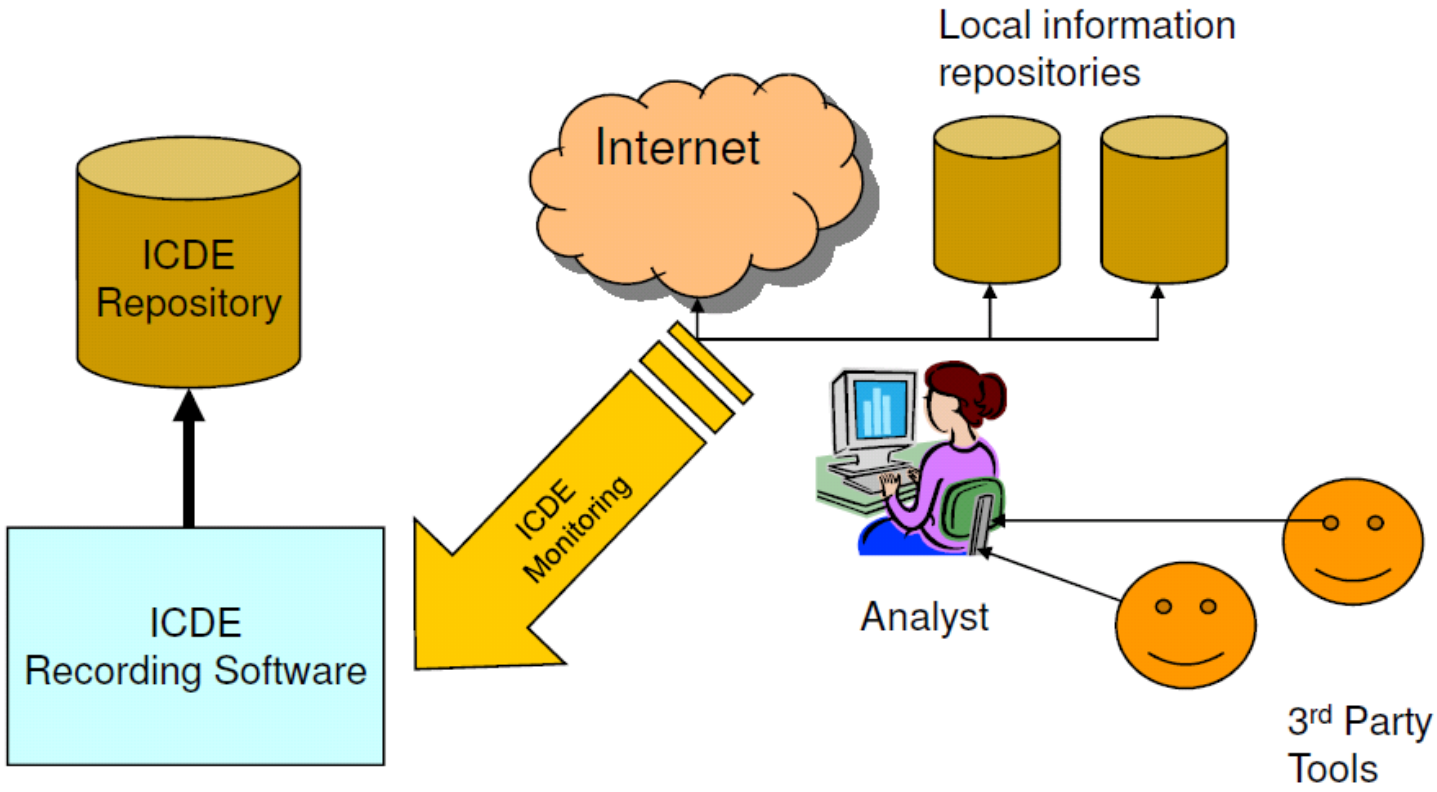
From *Essential Software Architecture*

ICDE system

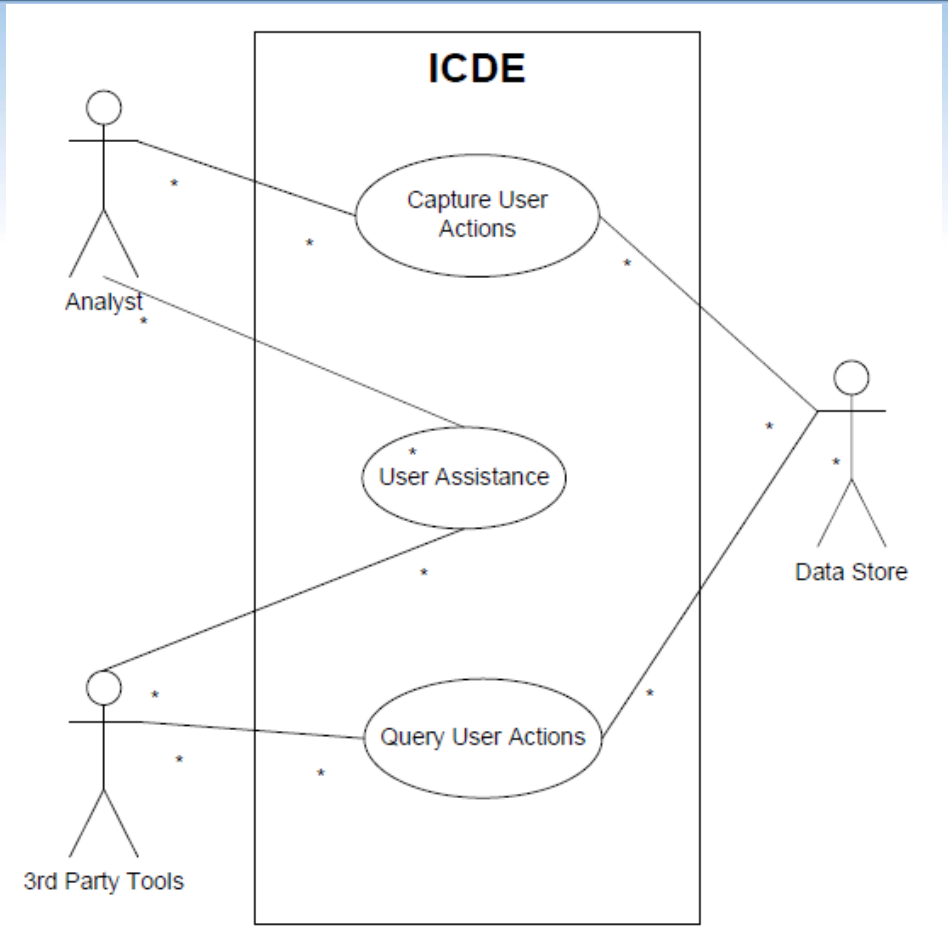
Information Capture and Dissemination Environment (ICDE) is a software system for providing intelligent assistance to

- financial analysts
- scientific researchers
- intelligence analysts
- analysts in other domains

ICDE schematic



ICDE use cases



Case study context

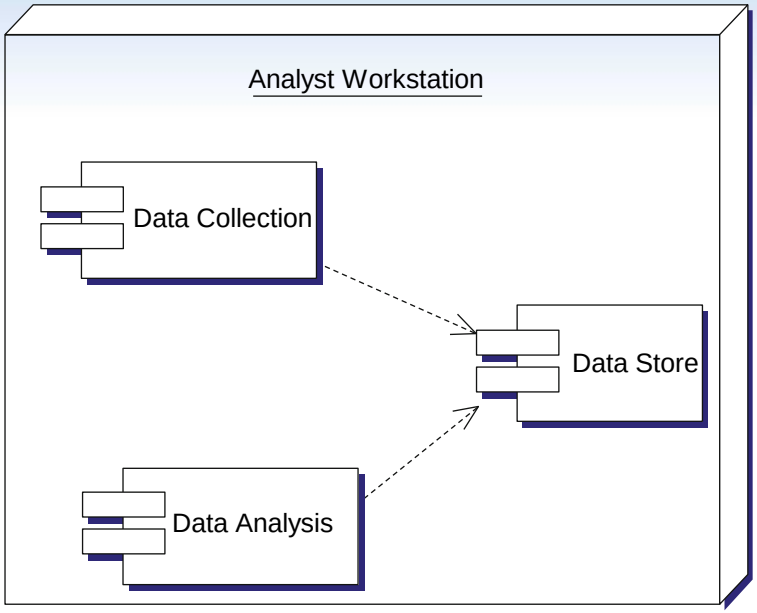
ICDE version 1.0 in production

Basically a complex, *raw* information capture tool, GUI for looking at captured data.

2 tier client-server, single machine deployment.

- Java, Perl, SQL,
- Programmatic access to data through very complex SQL (38 tables, 46 views)
- Windows XP platform

Case study context



ICDE Version 1.0 application architecture

ICDE version 2.0

Business goals:

Business goal	Supporting technical objective
Encourage third party tool developers	Simple and reliable programmatic access to data store for third party tools
	Heterogeneous (i.e., non-Windows) platform support for running third party tools
	Allow third party tools to communicate with ICDE users from a remote machine
Promote the ICDE concept to users	Scale the data collection and data store components to support up to 150 users at a single site
	Low-cost deployment for each ICDE user workstation

ICDE version 2.0

ICDE v2.0 scheduled for development in 12 month timeframe

- Fixed schedule, budget

Major changes to:

- Enhance data capture tools (GUI)
- Support 3rd party tool integration, testing, data access and large production scale deployments (150's of users)

Architecturally significant requirements for ICDE v2.0

ICDE project requirements:

- ✓ *Heterogeneous platform support for access to ICDE data*
- ✓ *Instantaneous event notification (local/distributed)*
- ✓ *Over the Internet, secure ICDE data access*
- ✓ *Ease of programmatic data access*

ICDE project team requirements:

- ✓ *Insulate 3rd party projects and ICDE tools from database evolution*
- ✓ *Scalable infrastructure to support large, shared deployments*
- ✓ *Minimize license costs for a deployment*

Unknowns

- ✓ *Minimize dependencies, making unanticipated changes potentially easier*

Summary

ICDE is a reasonably complex system

Will be used to illustrate concepts during the remainder of this course

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."

QAs of ICDE system

Performance

Scalability

Modifiability

Security

Availability

Integration

Any other quality attributes?

Performance

A performance quality requirement defines 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 is fundamental for software system.

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

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

Performance - Deadlines

‘Something must be completed before some specified time’

- Payroll system must complete by 2 am so that electronic transfers can be sent to bank
- Weekly accounting run must complete by 6 am Monday so that figures are available to management

Deadlines often associated with batch jobs in IT systems.

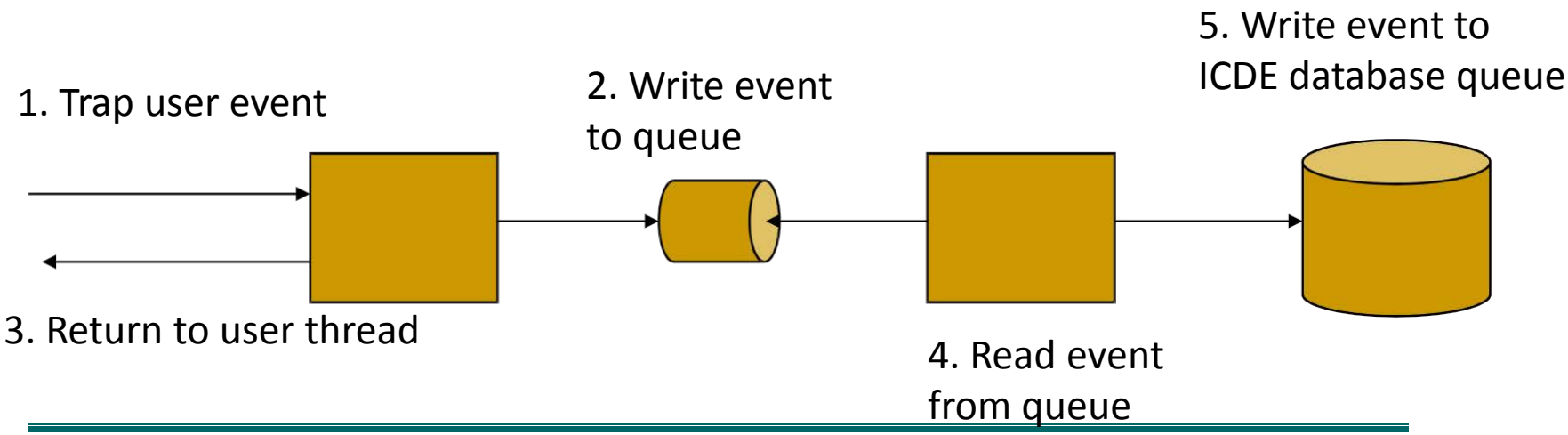
ICDE performance issues

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

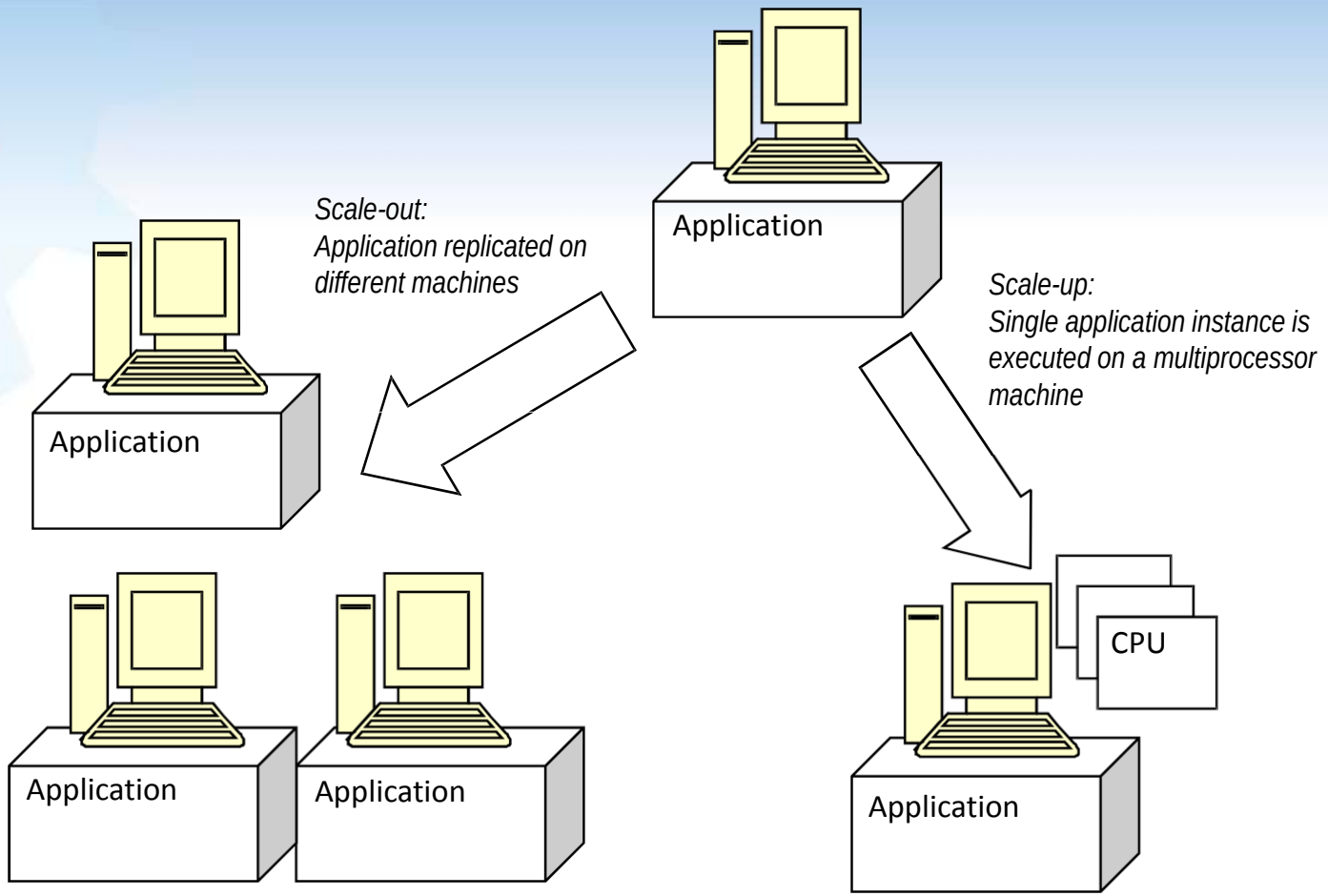
How does an 100 tps application behave when simultaneous request load grows? E.g.

- 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 hardware should improve performance:

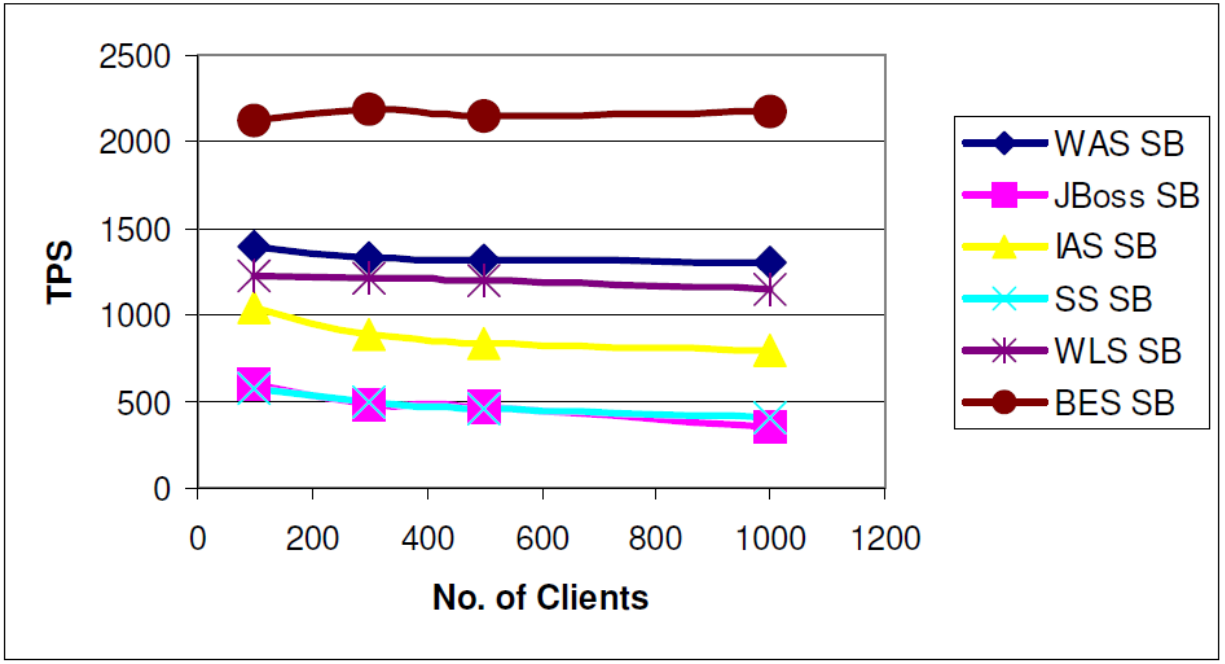
- Scalability must be achieved without modifications to application architecture

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
- 24*7*52 (100% availability)

Related to an application's reliability

- Unreliable applications suffer poor availability

Related to recoverability

Availability

Period of loss of availability determined by:

- Time to detect failure
- Time to correct failure
- Time to restart application

$$\frac{MTBF}{MTBF + MTTR}$$

Availability

High availability typically refers to designs targeting availability of 99.999 percent (“5 nines”) or greater.

Strategies for high availability:

- Eliminate single points of failure
- Replication and failover
- Automatic detection and restart

Availability

System Availability Requirements

Availability	Downtime/90 Days	Downtime/Year
99.0%	21 hours, 36 minutes	3 days, 15.6 hours
99.9%	2 hours, 10 minutes	8 hours, 0 minutes, 46 seconds
99.99%	12 minutes, 58 seconds	52 minutes, 34 seconds
99.999%	1 minute, 18 seconds	5 minutes, 15 seconds
99.999%	8 seconds	32 seconds

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

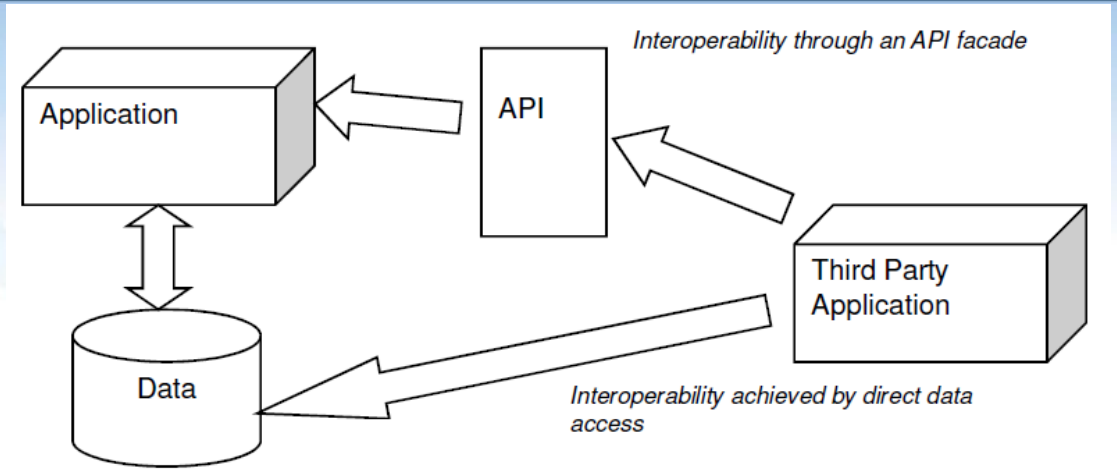
Ease with which an application can be incorporated into a broader application context

- 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.

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 nonfunctional requirements

Many QAs

Architect must decide which are important for a given application

- Understand implications for application
- Understand competing requirements and tradeoffs

Discussion question

1. How many other qualities of software can you name that were not covered in this lecture? With which other qualities does it most often interact?

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?

Usability

The End