Lecture 05:
Non Functional Requirements (NFR) – Quality Attributes (2)

Emad Shihab
Adapted from Ahmed E. Hassan and Ian Gorton
Last Class - Recap

- Use quality attributes to make NFRs clearer and more precise

- Performance
  - Throughput
  - Response Time
  - Deadlines

- Scalability
  - Request Load
  - Connections
  - Data size
  - Deployment
Today

- Modifiability
- Security
- Availability
- Integration
What is Modifiability?

- Modifiability measures how easy it may be to change an application.
Why Consider Modifiability?

- Software systems are (almost) guaranteed to change
  - New (non-) functional requirements

- Modifiable systems are easier to change/evolve
  - Estimate cost/effort
How to Measure Modifiability?

- Evaluate based on context
  - Research projects vs. Industrial tools
  - Avoid over-engineering!

- Architect asserts likely change scenarios
Modifiability Scenarios

- How hard is it to.....
  - Incorporate new features for self-service check-out kiosks.
  - Replace COTS component since vendor goes out of business
  - Port application from Linux to the Microsoft Windows platform.
Modifiability Analysis

- Difficult to quantify impact!

- The best possible is...
  - Convincing impact analysis
  - Solution can accommodate modification without much change
Modifiability General Rules

Some general rules ....

- Minimizing dependencies increases modifiability
- Avoid ripple effects!
Modifiability for ICDE

- The range of events trapped and stored by the ICDE client to be expanded
  - e.g. Different types of search inputs

- Third party tools to communicate new message types
Security

- Specialized quality attribute:
  - Lots of technology available
  - Depends on the application and the context
Security is…

- **Authentication**
  - Verify the identity of users

- **Authorization**
  - Access rights

- **Encryption**
  - Messages sent to/from application are encrypted

- **Integrity**
  - Contents are not altered in transit

- **Many others…**
Security Approaches

- Internet application security (SSL, PKI)
- Authentication and Authorization in Java (JAAS)
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

■ The proportion of the required time it is useable
  – Example availability requirements
    • 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
Measuring Availability

Period of loss of availability determined by:

- Time to detect failure +
- Time to correct failure +
- Time to restart application
Availability General Rules

- Eliminate single points of failure
- Replication and failover
- Automatic detection and restart
- Recoverability (e.g. Microsoft Word)
  - 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

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
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**
  - Move to new HW/SW platform

- **Testability**
  - How easy/difficult to test?
  - Consider program complexity

- **Supportability**
  - How easy to support once deployed?
  - Consider modularity
Design Trade-offs

- QAs are rarely orthogonal
  - highly secure system, difficult to integrate in open environment
  - highly availability, may lead to lower performance
  - high performance, may require being tied to a given platform
NFR – Final Remarks
Importance of NFR

■ Functional reqs must be met (ie. mandatory)

■ NFRs could be:
  – Mandatory: eg. response time a valve to close
    • The system is unusable
  – Not mandatory: eg. response time for a UI
    • The system is usable but provides a non-optimal experience

■ NFRs are very important: 20% of the requirements, hardest to elicit and specify

■ NFR: importance increases as market matures
Expressing NFRs

- Functional are usually expressed in Use-Case form

- NFR cannot be expressed in Use-Case form
  - usually do not exhibit externally visible functional behaviour

- Not enough to list NFRs,
  - should be clear, concise, and measurable

- Defining good NFRs requires not only the involvement of the customer but the developers too
  - Ease of maintenance (lower cost) vs. ease of adaptability
The effects of NFRs on high level design and code

- Their implementation does not map usually to a particular subsystem

- Very hard to modify a NFR once you pass the architecture phase:
  - Consider making an already implemented system more secure, more reliable, etc.
Next Class

- Tuesday, Sep. 20
- Architectural Styles
Selected Further Reading