Are All Quality Goals Created Equal?

How to Augment your in-place Process

John Steven
Technical Director
Office of the CTO
Cigital Inc.

Functional vs. Non-functional

Functional goals
- Relatively straightforward
- Hierarchical map from goals to requirements

Non-functional goals
- Difficult to quantify
- Subjective, at times
- Emergent or resultant behaviors
- Interrelated and conflicting
Conflicting Goals

Security vs. Functionality
- Remote access
- Scripting, macros, and interop.

Performance vs. Security
- Access checks and input filtering

Performance vs. Maintainability

Reliability vs. Performance
- Error handling

What about Compliance... reporting?
- Log all XXX transactions
- Do not store XXX transactions in logs
- Logging all XXX transactions is slow--we can’t meet our performance goals

Conflicting Goals Explored #1

Leveraging Keystores and Ciphers in Java
Goals: Extensibility vs. Information Leakage

- Keystore getKey()
- Applet possesses:
  - Cipher
  - Keystore
  - Key
  - Simple, straightforward
  - Keystore, ciphers are easily extended
  - Leaks data
- Introduce intermediate class: cipherOp
  - Accepts plain-text
  - Returns cipher-text
  - Privileged, signed
  - Evaluates caller
  - Does not leak data to Applet
Conflicting Goals Explored #2

- Encrypt credit card data on merchant’s systems
- Must resist attack AND work on mainframe

- Data size constraint
- Most architectures ‘key’ data on the CC
  - If I use my CC at two merchant sites, will it be the ‘same’?
- Data must be protected in transport and at rest

- CC companies have created a “disposable” card paradigm
  - Does this meet the constraints?
  - How will this affect the merchant’s capabilities for logging, intrusion detection?
- What about SSL/TLS?

ISO Quality Goals

- How does taxonomy help us with interaction?
- How does priority affect hierarchy and relationships?

- Leverage risk management philosophy rather than completeness
- Use models as completeness questionnaires for Stakeholder interrogation

Internal and External Software Quality Model (ISO/IEC 9126)

<table>
<thead>
<tr>
<th>Quality Characteristics</th>
<th>Subcharacteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functionality</td>
<td>Suitability, Accuracy, Interoperability, Security, Compliance</td>
</tr>
<tr>
<td>Reliability</td>
<td>Maturity, Fault Tolerance, Recoverability, Compliance</td>
</tr>
<tr>
<td>Usability</td>
<td>Understandability, Learnability, Operability, Comp, Attractiveness</td>
</tr>
<tr>
<td>Efficiency</td>
<td>Time behavior, Resource utilization, Compliance</td>
</tr>
<tr>
<td>Maintainability</td>
<td>Analyzability, Changeability, Stability, Testability, Compliance</td>
</tr>
<tr>
<td>Portability</td>
<td>Adaptability, Installability, Co-existence, Replaceability, Comp</td>
</tr>
</tbody>
</table>

© 2000 by Carnegie Mellon University
Architectural Frameworks

Processes and the Organizations They Serve
Organization Type and Quality

- What do we know?
  - Different Drivers
  - Different Target
  - Different Language
  - Different Problems
  - Different Solutions

Users of Software

Manufacturer of Software

Level of Business Maturity and Complexity

'Simple' Organizations

CIO Office

Application Dev.

IT

Policy

Process

Quality Assurance

Net/Ops

Systems
Augmenting Software Development

- Three-pronged approach
- As you iteratively improve ask:
  - “Does the stool stand up?”

- Observe
  - Constructive “in-line” activities
  - Assurance activities

- Ask
  - What extra activities have been added?
  - What activities have been augmented?
  - What has fundamentally changed?
Augmenting Enterprise Frameworks

- How many boxes can you “light up”?
- There are still cross-cutting concerns:
  - Communication
  - Policy
  - Portfolio management
  - Vendor management
    - Outsourcing
    - Consulting
    - Partner Management
- Have you just created another group?

Security & Process

- Software producer’s process
- Note the commitment to training
- Use of a growing list of internal tools
- “Go-no-go” decision
- No explicit iterative project-to-project knowledge reuse
- Doesn’t help adopting organizations think through policy, risk management, and so on.
Software security in the SDLC (touchpoints)

- Generic, iterative process that can be adopted piecemeal
- Solid assurance framework

- No prescription for tools
- Threat modeling?
- What constructive activities do you add in order to not fail assurance

CLASP

- Institute security awareness program
- Monitor security metrics
- Manage certification process
- Specify operational environment
- Identify global security policy
- Identify user roles and requirements
- Detail misuse cases
- Perform security analysis of requirements
- Document security design assumptions
- Specify resource-based security properties
- Apply security principles to design
- Research and assess security solutions
- Build information labeling scheme
- Design UI for security functionality
- Annotate class designs with security properties
- Perform security functionality usability testing

- Manage System Security Authorization Agreement
- Specify database security configuration
- Perform security analysis of system design
- Integrate security analysis into build process
- Implement and elaborate resource policies
- Implement interface contracts
- Perform software security fault injection testing
- Address reported security issues
- Perform source-level security review
- Identify, implement and perform security tests
- Verify security attributes of resources
- Perform code signing
- Build operational security guide
- Manage security issue disclosure process
Java Performance Process
- Simple & Iterative
- Recognizes responsibility in analysis and design
- Explicitly mentions measurement

Their key points:
- “Writing high-performance software requires action during all phases of the software development lifecycle.”
- “Creating clear system and performance requirements is the key to evaluating the success of your project.”
- “Scalability is more dependent on good design decisions than optimal coding techniques.”
- “Performance tuning is an iterative process. Data gathered during profiling needs to be fed back into the development process.”
General Problems with Quality Goals

- Elicit non-functional quality goals
- Gain stakeholder consensus
- Framework for discussion regarding the tradeoffs between non-functional attributes
Common Pitfalls

Pitfall #1: Security = Feature

- Each vertical tends to hammer certain features
  - Authentication, Authorization, and Access control
  - Cryptography, PKI
  - “SSL”
  - And so forth...
- Can you name elements of security that aren’t features?
- How does security intersect the lifecycle?
The classic security tradeoff

Windows Complexity

<table>
<thead>
<tr>
<th>Windows Version</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win 3.1</td>
<td>(1990)</td>
</tr>
<tr>
<td>Win NT</td>
<td>(1995)</td>
</tr>
<tr>
<td>Win 95</td>
<td>(1997)</td>
</tr>
<tr>
<td>Win NT 4.0</td>
<td>(1998)</td>
</tr>
<tr>
<td>Win 98</td>
<td>(1999)</td>
</tr>
<tr>
<td>Win NT 5.0</td>
<td>(2000)</td>
</tr>
<tr>
<td>Win 2K</td>
<td>(2001)</td>
</tr>
<tr>
<td>XP</td>
<td>(2002)</td>
</tr>
</tbody>
</table>

Millions of Lines

Pitfall #2: Tools are the answer.

- Security
  - SSO
  - Application Servers
  - Firewalls
  - IDSes
  - Crypto kits
- Performance
  - Load balancers
  - Profiling tools

Cybersecurity vendors and solutions:
- symantec
- Computer Associates
- MERCURY
- Cisco Systems
- VERITAS
- QUNCE LABS
- aspect security
- Optimizeit™ Suite
- Check Point
- Entrust
Pitfall #3: You can ‘bolt on’ <a quality goal>!

- Make the features work, then add SSL, Authentication, crypto
- “We’ll get <the two> talking first, then optimize it…”
- Enables turning to vendors and tools
- Disregards essential analysis, and other “early lifecycle” activities
- Fails to understand nature of emergent properties
  - Think of your most pesky performance bottleneck
  - Secure design, vulnerable to Buffer Overflow

Pitfall #4: Over Verticalization

- Motivators towards a single quality goal are often vertical specific:
  - Non-software producers tend to manage only to schedule
  - Quality goals are driven by a single stakeholder/motivator
    - Security =
      - Authentication/Authorization
      - Crypto
    - Performance =
      - Parallelization
      - Response time
    - Drive to regulatory compliance only
Pitfall #5: …Create a Group for that

- <Quality goal XXX> doesn’t fit into my process so:
  - Information Security Wonks
    - These guys don’t understand how to write code
    - They don’t understand how things work
    - They just write policy
    - Their job is to say, “No”
  - Performance group
    - Heroes who re-write everything
    - Break everything they touch
    - Destroy maintainability

Making the Best of “Security Controls”
Enterprise Security Initiatives

- Enterprise directory
- Single Sign on
- Role-based Access Control
- Enterprise logging
- PKI
- ...

Challenges:
- Supporting legacy platforms
- Defining roles
- Defining access policy
- Building a coherent enterprise picture that works for all applications
- Roll out
- Operational maintenance

RBAC: Guidelines

- Start with applications built on a supportive platform
- Start with an application with well-defined roles

- Avoid tackling delegation & entitlement in your early efforts
- Avoid applications that interact by host-to-host auth only
- Avoid artificial or overly specific roles to support single workflows

- Force definition of:
  - Use cases, workflows
  - Data sensitivity classification
  - Roles, groups, access lists
SSO: Guidelines

- Incorporate Applications used as part of a single workflow
- Separate classes of users
- Compartmentalize applications

- Avoid a highly privileged app allowing a less trusted app to vouch for user identity
- Avoid sign on once and forget it
- Think deeply about password ‘fitness’

- Valuable in that it forces:
  - Consideration of simultaneous access, timeout, and password fitness
  - Consideration of application-to-application trust

Audit Logging: Guidelines

- Follow workflow semantics
- Log misuse and use at a semantic level
- Incorporate into application logic, error-handling
- Codify usage trends

- Avoid capturing data that is too fine grain
- Avoid capturing too much data to sift through

- Valuable in that it forces:
  - Misuse case definition
  - Developers to think about misuse, error handling
  - Helps introduce developers to operations
When wrestling your own Enterprise Initiative…

- What are the challenges?
- What pitfalls are to be avoided?
- What guides help you resist attack?
- Where are opportunities to inject security activities?