Avoiding Overruns in the Specification of Non-Functional Requirements

Barry Boehm, USC

NDIA SE Conference 2015
October 29, 2015
Critical nature of non-functional requirements (NFRs)

- Also called System Qualities, ilities
- Major source of project overruns, failures
- Poorly defined, understood
- Underemphasized in project management

Example sources of project overruns

- The conspiracy of optimism and its effect on SE
- Inflexible requirements
- Overagile and Underagile Methods
- Optimizing some NFRs at the expense of others: Security
- Chaotic nature of NFR definition, understanding
  - DoD-SERC NFR definition, practices efforts

Principles for avoiding the sources of project overruns
Critical Nature of NFRs
Major source of DoD, other system overruns

- NFRs have systemwide impact
  - System elements generally just have local impact
- NFRs often exhibit asymptotic behavior
  - Watch out for the knee of the curve
- Best architecture is a discontinuous function of NFR level
  - "Build it quickly, tune or fix it later" highly risky
  - Large system example below

![Graph showing cost versus response time]

Original Spec

Original Architecture: Custom; many cache processors

Required Architecture: Modified Client-Server

After Prototyping

Original Cost

$50M

$100M

1

2

3

4

5

Response Time (sec)
Example of Current Practice

• “The system shall have a Mean Time Between Failures of 10,000 hours”

• What is a “failure?”
  – 10,000 hours on liveness
  – But several dropped or garbled messages per hour?

• What is the operational context?
  – Base operations? Field operations? Conflict operations?

• Most management practices focused on functions
  – Requirements, design reviews; traceability matrices; work breakdown structures; data item descriptions; earned value management

• What are the effects of or on other SQs?
  – Cost, schedule, performance, maintainability?
Proliferation of Definitions: Resilience

• Wikipedia Resilience variants: Climate, Ecology, Energy Development, Engineering and Construction, Network, Organizational, Psychological, Soil

• Ecology and Society Organization Resilience variants: Original-ecological, Extended-ecological, Walker et al. list, Folke et al. list; Systemic-heuristic, Operational, Sociological, Ecological-economic, Social-ecological system, Metaphoric, Sustainabilty-related

• Variants in resilience outcomes
  – Returning to original state; Restoring or improving original state; Maintaining same relationships among state variables; Maintaining desired services; Maintaining an acceptable level of service; Retaining essentially the same function, structure, and feedbacks; Absorbing disturbances; Coping with disturbances; Self-organizing; Learning and adaptation; Creating lasting value
  – Source of serious cross-discipline collaboration problems
Outline

• Critical nature of non-functional requirements (NFRs)
  – Also called System Qualities, ilities
  – Major source of project overruns, failures
  – Poorly defined, understood
  – Underemphasized in project management

Example sources of project overruns
  – The conspiracy of optimism and its effect on SE
  – Inflexible requirements
  – Overagile and Underagile Methods
  – Optimizing some NFRs at the expense of others: Security
  – Chaotic nature of NFR definition, understanding
    • DoD-SERC NFR definition, practices efforts

• Principles for avoiding the sources of project overruns
The Conspiracy of Optimism

Take the lower branch of the Cone of Uncertainty
Added Cost of Minimal Software SysE
Based on COCOMO II calibration data

% Added Cost, Very Low vs. Very High RESL Rating

Software Product Size (KSLOC)

10/29/2015
How Much Architecting is Enough?

Percent of Project Schedule Devoted to Initial Architecture and Risk Resolution

Added Schedule Devoted to Rework (COCOMO II RESL factor)

Total % Added Schedule

Sweet Spot Drivers:
- Rapid Change: leftward
- High Assurance: rightward
Inflexible Requirements: Dual Cones of Uncertainty
– Obsolete large command and control system

Uncertainties, changes in competition, technology, organizations, mission priorities
**Overagile and Underagile Methods**

- **Overagile Methods: Easiest First**
  - Treat security, safety, scalability as user stories
  - Defer their development to late releases
  - Doing the easy parts will make the hard parts easier
    - Maybe for puzzles, but not for complex software-intensive systems

- **Underagile Methods: Apply rigorous methods to all system parts**
  - May need for some parts: security-critical, safety-critical
  - But not for others: user, evolving external-system interfaces
    - Particularly hard to change if included in contracts
      - Two systems of systems: 141 average workdays vs. 48
    - Important to modularize around sources of change
      - Avoids ripple effects on other system parts
Example of NFR Value Conflicts: Security IPT

- Single-agent key distribution; single data copy
  - Reliability: single points of failure

- Elaborate multilayer defense
  - Performance: 50% overhead; real-time deadline problems

- Elaborate authentication
  - Usability: delays, delegation problems; GUI complexity

- Everything at highest level
  - Modifiability: overly complex changes, recertification
• Critical nature of non-functional requirements (NFRs)
  – Also called System Qualities, ilities
  – Major source of project overruns, failures
  – Poorly defined, understood
  – Underemphasized in project management

• Example sources of project overruns
  – The conspiracy of optimism and its effect on SE
  – Inflexible requirements
  – Overagile and Underagile Methods
  – Optimizing some NFRs at the expense of others: Security

  Chaotic nature of NFR definition, understanding
  • DoD-SERC NFR definition, practices efforts

• Principles for avoiding the sources of project overruns
DoD-SERC NFR Definition, Practices Efforts: NFR Ontology, Set-based requirements and design

• Modified version of IDEF5 ontology framework
  – Classes, Subclasses, and Individuals
  – Referents, States, Processes, and Relations

• Top classes cover stakeholder value propositions
  – Mission Effectiveness, Resource Utilization, Dependability, Changeability

• Subclasses identify means for achieving higher-class ends
  – Means-ends one-to-many for top classes
  – Ideally mutually exclusive and exhaustive, but some exceptions
  – Many-to-many for lower-level subclasses

• Referents, States, Processes, Relations cover NFR variation
  • Referents: Sources of variation by stakeholder value context:
  • States: Internal (beta-test); External (rural, temperate, sunny)
  • Processes: Operational scenarios (normal vs. crisis; experts vs. novices)
  • Relations: Impact of other SQs (security as above, synergies & conflicts)
Set-Based NFRs Definition Convergence

RPV Surveillance Example

Phase 1. Rough ConOps, Rqts, Solution Understanding
Phase 2. Improved ConOps, Rqts, Solution Understanding
Phase 3. Good ConOps, Rqts, Solution Understanding
• Critical nature of non-functional requirements (NFRs)
  – Also called System Qualities, ilities
  – Major source of project overruns, failures
  – Poorly defined, understood
  – Underemphasized in project management

• Example sources of project overruns
  – The conspiracy of optimism and its effect on SE
  – Inflexible requirements
  – Overagile and Underagile Methods
  – Optimizing some NFRs at the expense of others: Security
  – Chaotic nature of NFR definition, understanding
    • DoD-SERC NFR definition, practices efforts

Principles for avoiding the sources of project overruns
Principles for Avoiding Project Overrun Sources

- **Stakeholder Value-Based Guidance**
  - Identify, involve missing success-critical stakeholders
  - Bank of America Master Net example

- **Incremental Commitment and Accountability**
  - Set-Based Requirements and Design: Build in Tradespace

- **Concurrent Multi-Discipline Engineering**
  - Complement NFR IPTs with NFRs IPT

- **Evidence and Risk-Based Decisions**
ICSM Principles Counterexample: Bank of America Master Net
Types of Decision Reviews

• **Schedule-based commitment reviews (plan-driven)**
  – We’ll release the RFP on April 1 based on the schedule in the plan
  – $70M overrun to produce overperforming system

• **Event-based commitment reviews (artifact-driven)**
  – The design will be done in 15 months, so we’ll have the review then
  – Responsive design found to be unaffordable 15 months later

• **Evidence-based commitment reviews (risk-driven)**
  – Evidence: affordable COTS-based system can’t satisfy 1-second requirement
    • Custom solution roughly 3x more expensive
  – Need to reconsider 1-second requirement
• Attempt to validate 1-second response time
  • Commercial system benchmarking and architecture analysis: needs expensive custom solution
  • Prototype: 4-second response time OK 90% of the time

• Negotiate response time ranges
  • 2 seconds desirable
  • 4 seconds acceptable with some 2-second special cases

• Benchmark commercial system add-ons to validate their feasibility

• Present solution and feasibility evidence at evidence-based decision review
  • Result: Acceptable solution with minimal delay
Backup charts
<table>
<thead>
<tr>
<th>Flexibility</th>
<th>Dependability</th>
<th>Mission Effectiveness</th>
<th>Resource Utilization</th>
<th>Physical Capability</th>
<th>Cyber Capability</th>
<th>Interoperability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain architecting within domain</td>
<td>Adaptability</td>
<td>Agile methods</td>
<td>Automated I/O validation</td>
<td>Adaptability</td>
<td>Adaptable</td>
<td>Adaptability</td>
</tr>
<tr>
<td>Modularity</td>
<td>Many options</td>
<td>Service oriented</td>
<td>Loose coupling</td>
<td>Spare capacity</td>
<td>Spare capacity</td>
<td>Loose coupling</td>
</tr>
<tr>
<td>Self Adaptive</td>
<td>Spare capacity</td>
<td>For sustainability</td>
<td>Product line architectures</td>
<td>Versatility</td>
<td>Staffing, Empowering</td>
<td>Product line architectures</td>
</tr>
<tr>
<td>Smart monitoring</td>
<td>User programmability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spare Capacity</td>
<td>Use software vs. hardware</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Dependability        |                        |                       |                      |                    |                  |                  |
| Accreditation        |                        |                       |                      |                    |                  |                  |
| Agile methods assurance |                   |                       |                      |                    |                  |                  |
| Encryption           | Many options           |                       |                      |                    |                  |                  |
| Multi options        |                       |                       |                      |                    |                  |                  |
| Multi-domain modifiability |               |                       |                      |                    |                  |                  |
| Multi-level security |                       |                       |                      |                    |                  |                  |
| Survivability        |                       |                       |                      |                    |                  |                  |
| Scalability          |                       |                       |                      |                    |                  |                  |
| Redundancy           |                       |                       |                      |                    |                  |                  |
| Total Ownership Cost | Value prioritizing    |                       |                      |                    |                  |                  |

<table>
<thead>
<tr>
<th>Mission Effectiveness</th>
<th>Resource Utilization</th>
<th>Physical Capability</th>
<th>Cyber Capability</th>
<th>Interoperability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autonomy vs. Usability</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anti-tamper</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modularity slowdowns</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Armor vs. Weight</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-domain architecture interoperability conflicts</td>
<td>Multi-domain architecture interoperability conflicts</td>
<td>Multi-domain architecture interoperability conflicts</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Versatility vs. Usability</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Redundancy</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scalability</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spare Capacity</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tight coupling</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use software vs. hardware</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Resource Utilization</th>
<th>Physical Capability</th>
<th>Cyber Capability</th>
<th>Interoperability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agile methods scalability</td>
<td>Acquisition Cost</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Certification</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-domain architecture interoperability conflicts</td>
<td>Multi-domain architecture interoperability conflicts</td>
<td>Multi-domain architecture interoperability conflicts</td>
<td></td>
</tr>
<tr>
<td>Fallbacks</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-domain architecture interoperability conflicts</td>
<td>Multi-domain architecture interoperability conflicts</td>
<td>Multi-domain architecture interoperability conflicts</td>
<td></td>
</tr>
<tr>
<td>Over-optimizing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use software vs. hardware</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Physical Capability</th>
<th>Cyber Capability</th>
<th>Interoperability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lightweight agility</td>
<td>Physical architecture or cyber architecture</td>
<td>Reduced speed of Assertion checking</td>
</tr>
<tr>
<td>Multi-domain architecture interoperability conflicts</td>
<td>Over-optimizing</td>
<td>Reduced speed of connectors, standards compliance</td>
</tr>
<tr>
<td>Tight coupling</td>
<td></td>
<td>Tight vs. Loose coupling</td>
</tr>
<tr>
<td>Use software vs. hardware</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cyber Capability</th>
<th>Interoperability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agile Methods scalability</td>
<td>Multi-domain architecture interoperability conflicts</td>
</tr>
<tr>
<td>Multi-domain architecture interoperability conflicts</td>
<td>Multi-domain architecture interoperability conflicts</td>
</tr>
<tr>
<td>Over-optimizing</td>
<td></td>
</tr>
<tr>
<td>Tight coupling</td>
<td></td>
</tr>
<tr>
<td>Use software vs. hardware</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Security</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>Security</td>
<td>Confidentiality, Integrity, Availability</td>
</tr>
<tr>
<td></td>
<td>Assurance Cases</td>
</tr>
<tr>
<td></td>
<td>Certification</td>
</tr>
<tr>
<td></td>
<td>Failure Modes and Effects Analysis</td>
</tr>
<tr>
<td></td>
<td>Fault Tree Analysis</td>
</tr>
<tr>
<td></td>
<td>Recertification</td>
</tr>
<tr>
<td>Reliability</td>
<td>Non-redundancy (For Security)</td>
</tr>
<tr>
<td></td>
<td>Redundancy (For Reliability)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Maintainability</td>
<td>Accessibility</td>
</tr>
<tr>
<td></td>
<td>Compartmentalization</td>
</tr>
<tr>
<td></td>
<td>Encryption</td>
</tr>
<tr>
<td></td>
<td>Recertification</td>
</tr>
</tbody>
</table>