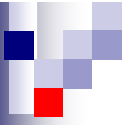




Ignore non-functional requirements at your peril!

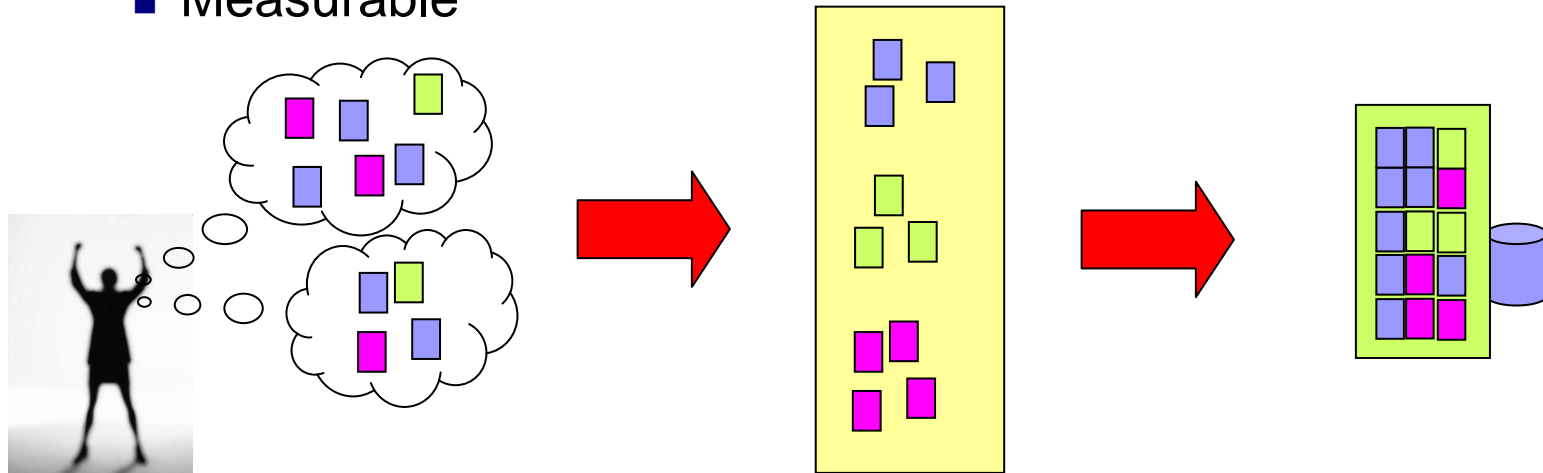
Perspectives on Software Quality
Presented by Steve Erlank

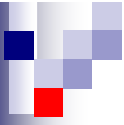
© 2007 Steve Erlank
Steve@fti.co.za



Overview of requirements

- Key concepts
 - What is a “requirement”?
 - What makes a quality requirement?
 - Discrete
 - Clearly and unambiguously stated
 - At the appropriate level of granularity
 - Traceable
 - Measurable





Basic requirements taxonomy

- Functional

- What the system must do



- Informational

- What the information stakeholders require



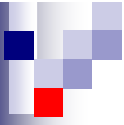
- Non-functional

- The qualitative characteristics of the system, sometimes known as 'constraints'



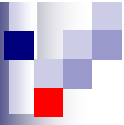
Performance
Control
Compliance
Operating Environment
etc

Non functional requirements can apply to entire solution, or to other requirements



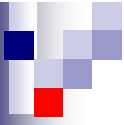
Example: Performance-related

- Response times and turnaround times
 - Rq 75..... web forms to be displayed within 1 sec...
 - Rq 76....complete a booking in less than 120 seconds...
 - Rq 74.....update 50000 transactions per hour...
- Scalability
 - Rq 77... scale to 1.5 m transactions per year
- Reliability
 - Rq 78.... system availability over 6 months to be 99.9%
 - Rq 72....MTBF to be.....MTTR less than.....
 - Rq 63..... system must be available 24x7...
- Flexibility/customisability
 - Rq 79.... reports must be available in PDF or printed format
 - Rq 80.drop-down lists to be user-updatable....
 - Rq 81.allow for multiple company letterheads...



Example: Control-related

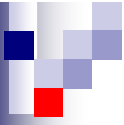
- Security
 - Rq 82..... users required to log on
 - Rq 83..... passwords change every 30 days...
- Auditability
 - Rq 84.... cash transactions to have date, time and user-id ...
 - Rq 85. .. override transactions to be written to audit log file...
- Integrity
 - Rq 86. ...is to print control totals at day end...
 - Rq 87.... orders may not exist without customers...
- Archiving & Backup Strategy
 - Rq 88..... to have rollback/roll forward capability...
 - Rq 89.to make automatic backups nightly...



Example: Conformance & Compliance

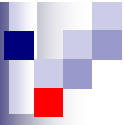
- Legislation that requires conformance
 - Rq 91. ... conform to following requirements of Disclosure of Information Act
- Interoperability
 - Rq 92....solution must integrate to
 - Rq 93.....solution must operate under following O/S
- Maintainability
 - Rq 94. ... source code to be supplied.....
- Technical or organisational standards compliance
 - Rq 96. ...solution must be .NET compliant...
 - Rq 97. ...solution must comply with following SOA standards...

NB: Technology-related 'requirements' are the LEAST IMPORTANT OF ALL



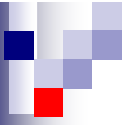
Example: Environmental

- Global/Geographic reach
 - Rq 98.... system must support following languages....
 - Rq 99.system must support following currencies....
- Operating environment
 - Rq 65. ...hardware must operate in (outdoor/wet/dry/cold/hot/dusty) conditions
- Usability (human factors engineering)
 - Rq 66.... User interface should have following...
 - Rq 68.....accessible by blind customers....



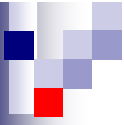
Tools & techniques

- There is a shortage of specific techniques for NFR analysis and specification
- Some options
 - Plain structured text (Imperative statements)
 - Matrices (performance matrix, security matrix etc)
 - Goal-oriented Requirements Language (GRL)
 - Quality Attribute Scenarios
 - etc



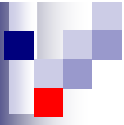
Implications of non-functional requirements

- When you consider non-functional requirements, you..
 - Increase stakeholder base
 - Increase project cost and time
 - Increase complexity of required solution
 - Affect architecture decisions
 - Require more skilled people, with more vision, and more commitment from the business
- Benefits
 - Better buy-in
 - Improved system longevity
 - Less rework
 - Better architectures
 - Better re-usability



So why do we NOT consider NFR enough...?

- They increase costs and take longer; and project managers always focus on time and cost, rather than risk and quality
- Too many of us are 'code writers' (or spec writers), and not enough are architects, designers, engineers and developers
- We don't employ people who understand quality and business value and how to deliver them
- If there are time constraints, we will always focus on functionality at the expense of quality
- Our instinctive problem-solving style is to reduce scope and ignore complexity; we satisfice, not optimise



So what do we do?

- Hire smarter people
- Develop a culture/methodology that includes constant critique and review at the design level
- Develop an NFR taxonomy and use quality checklists (http://www.itchecklists.com/template_nonfunctional_requirements_deliverables.html)
- Include more stakeholders in our requirements analysis
- Understand RISK
- Write better requirements, and deliver against these
- Reward quality and penalise poor delivery
- Design for flexibility and reusability by default;
- Develop a personal quality credo and create every artifact as though it was your last

