zühlke
empowering ideas

Solutioning Architectures

Software Quality Attributes

9. August 2011

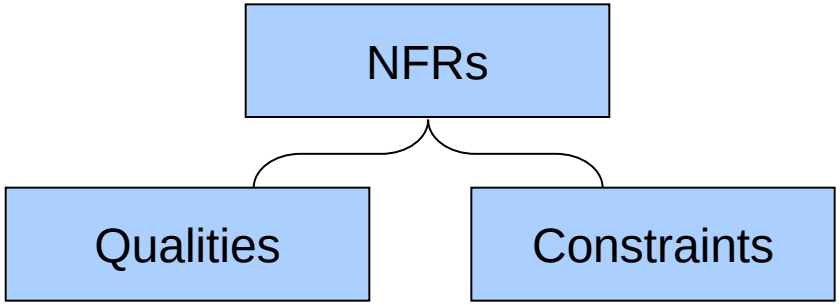
Kai Schwidder

Qualities and Constraints are often referred to as '*Non-Functional Requirements*'

zühlke
empowering ideas

Non-functional requirements (or NFRs) define the desirable qualities of a system *and* the constraints within which the system must be built

- *Qualities* define the properties and characteristics which the delivered system should demonstrate
- *Constraints* are the limitations, standards and environmental factors which must be taken into account in the solution



Good vs. Bad Software Qualities

zühlke
empowering ideas

Well specified are:

- *Correct*
- *Unambiguous*
- *Complete*
- *Consistent*
- *Measurable (verifiable)*
- *Traceable*
- *Actionable*
- *Design independent*

Note on feasibility:

- *It may not be possible to meet a particular NFR given other constraints – if so, this is a design/business issue*

They are *not* well specified if they are:

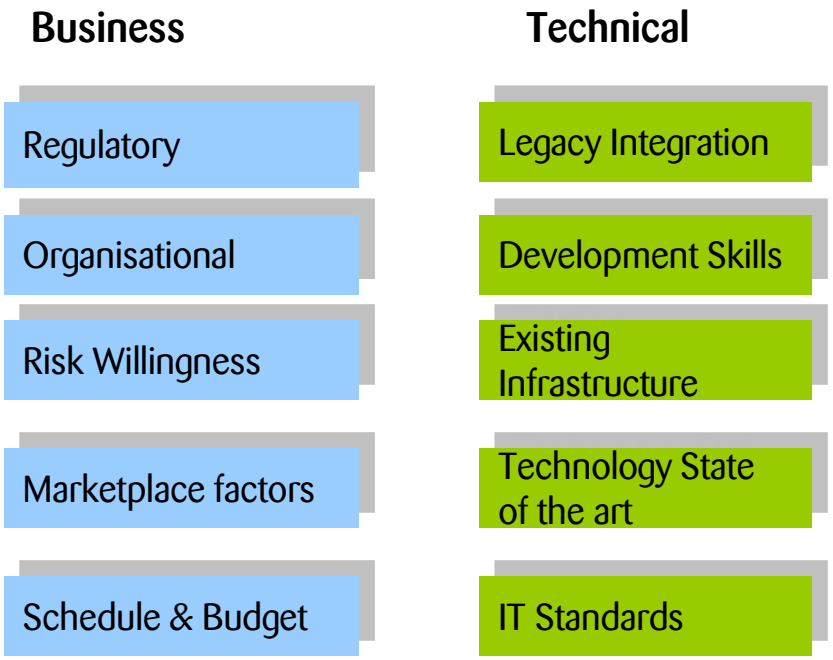
- *Misrepresentative of the true business need*
- *Open to interpretation*
- *High-level "principles" or "guidelines"*
- *Conflicting*
- *Not possible to test*
- *Implying a specific solution or technology*
- *Missing !*

Constraints

zühlke
empowering ideas

The business aspects of the project, customer's business environment or IT organization that influence the architecture

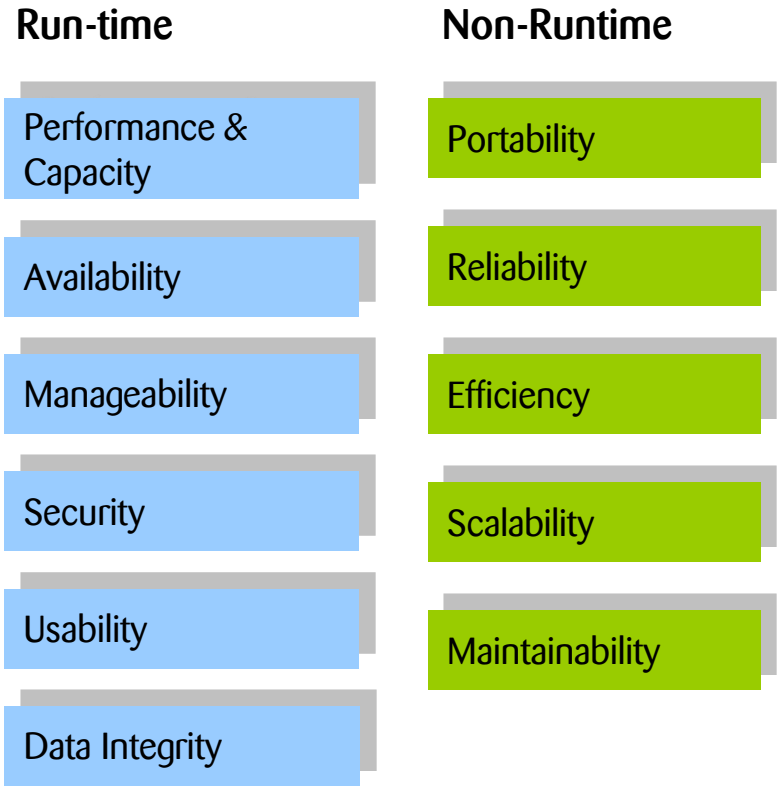
The technical environment and prevailing standards that the system, and the project, need to operate within



Qualities

Runtime qualities are ‘measurable’ properties, often expressed as “Service Level Requirements”.

Qualities might also be related to the development, maintenance, or operational concerns that are not expressed at runtime.

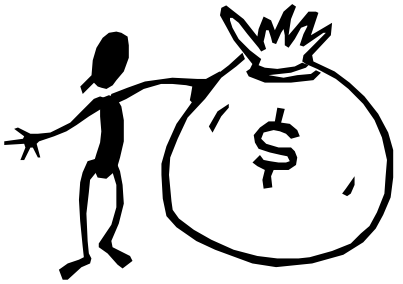


The best technique for reducing the risk of poor quality of service is to consider the qualities from the start

zühlke
empowering ideas

Build 'quality' into the solution starting with early design

- Understand the risks to the project
- Conduct quality of service engineering from the first elaboration of the architecture model
- Set guidelines for the developers (software & infrastructure)
- Test the application/system at each major stage of development
- Make sure that the live support teams will be able to manage quality



Fix it early, and save money and problems later …

Common problems with Non-Functional "requirements"

zühlke
empowering ideas

Requirements are often vague or unactionable

- They need further elaboration, clarification, investigation (and possibly rejection)
- It may be possible to derive clear, actionable intentions from them

Requirements can be statements of principle or good intention but come with little enforcement

- The organisation's governance models are central

Once captured, requirements are often treated as "musts" or "givens" whereas in fact they are "tradable" and may need to be challenged

- Classic example is "given" technology standards (e.g. "all applications in .NET") or infrastructure constraints ("64kbps links to offices")

Common problems with Non-Functional "requirements" (cont.)

zühlke
empowering ideas

Requirements are often of poor quality

- Watch out for these issues:
  Unrepresentative, unclear, inaccurate, inconsistent, incomplete or unnecessarily constraining

NFRs documents often become "dumping grounds" for things which don't have another home

- (regardless of quality or suitability)

In reality, "requirements" are actually "influences"..whose characteristics we have to be clear about

zühlke
empowering ideas

A "requirement" in the widest sense …

stems from many sources … [**Context**]

is either an aspiration or a constraint …[**Polarity**]

may be negotiable (i.e. varying in importance) … [**Strength**]

may be generic or specific … [**Level of generality**]

may be directly actionable or difficult to interpret … [**Actionability**]

may affect many components of the Solution … [**Affected objects**]

may be helpful (good quality) or unhelpful (poor quality) [**Quality**]

Unless we understand the real context and importance of each requirement, we risk producing the wrong solution

The reality of Availability is that customers directly relate it to the End User experience

zühlke
empowering ideas

The Availability of a system is a measure of its readiness for usage

There are certain key terms that are used to define Availability-related concepts

zühlke
empowering ideas

- **High Availability**
  is taken to mean a requirement for a system or service to be over 99% available – typically implies thorough design and may require redundant components

- **Disaster Recovery**
  means the recovery of essential services in the event of a major business disruption that has resulted from the occurrence of a disaster

- **Business Continuity**
  means the continued operation of business processes to a predetermined acceptable level in the event of a major business disruption

- **Unscheduled Outage**
  is a time period when the system is not ready for use and the users expect it to be. These are unplanned outages caused by 'Random Events'

- **Scheduled Outage**
  is a time period when the system is not ready for use and the users do not expect it to be. These are planned outages driven by predefined events

There are certain key terms that are used to define Availability-related concepts

zühlke
empowering ideas

- **Continuous Operations**
  is the requirement for perpetual operations 365 days per year 24 hours per day with perhaps very rare scheduled outages

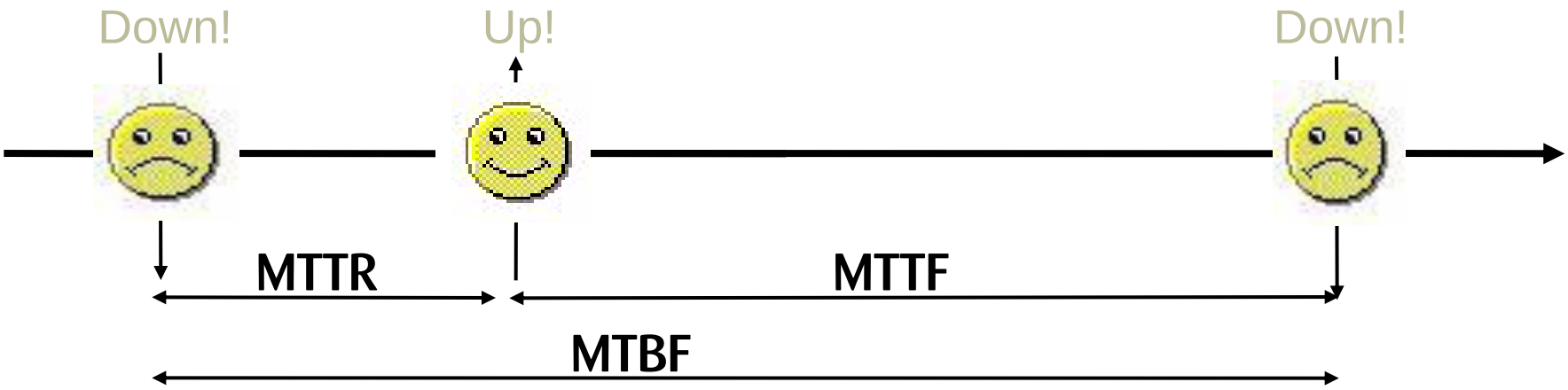
- **Fault Tolerance**
  is that property of a component, sub-system or system that means that normal service continues even though a fault has occurred within the system

- **Reliability**
  is the probability that an item will perform its intended function for a specified interval under stated conditions

- **Maintainability (or Recoverability)**
  is the probability that using prescribed procedures and resources, an item can be retained in, or restored to, a specific condition within a given period

Key Availability terms – Mean Times

...



- **Mean Time to Recover (MTTR)**
  is the typical time that it takes to recover (includes repair) a component, sub-system or a system.
- **Mean Time to Failure (MTTF)**
  is the mean time between successive failures of a given component, sub-system or system.
- **Mean Time between Failure (MTBF)**
  is the average time between successive failures of a given component, sub-system or system

Availability – a final word

zühlke
empowering ideas

It is estimated that

- ~20% of your total availability is a function of your use of technology
- ~80% is a function of your people and processes

E.g. someone says the:

- Root cause was that firewall logs were full
- The real reason was there was insufficient process in place to monitor the logs and clear them down

Technology and design is important, however don't assume that is your only challenge

What is Performance?

zühlke
empowering ideas

Definition

- “Performance. The degree to which a system or component accomplishes its designated functions within given constraints, such as speed, accuracy, or memory usage.” [IEEE-610.12]

In general

- Timeliness of response, and predictability, are the two main goals
- “Faster” is not always enough, as in for example, a real time system requires extremely consistent performance

An (old) quote (from ICCM):

- “A manager's goal should always be to strike the right balance between system function, processing costs, people costs, and performance. This is why the technical aspects of performance can never be entirely divorced from organizational politics”

Performance Aspects

zühlke
empowering ideas

There are three main, heavily inter-related aspects of Performance to be considered

- Response Times
  - On-line response times
  - Batch run times
- Throughput
  - Transactions per second
  - Records processed per hour
- Capacity
  - Component sizing to handle load
  - Contingency and Scalability

Must have adequate throughput to avoid poor response times

Sufficient capacity is required to meet throughput requirements

The Importance of Numbers

zühlke
empowering ideas

Performance Architects rely on **VOLUMETRIC DATA** and **ASSUMPTIONS** in order to ….

What do you do when these are vague or difficult to get?

Feed **performance and capacity models**, in order to ….

Or difficult to map down to the technical level?

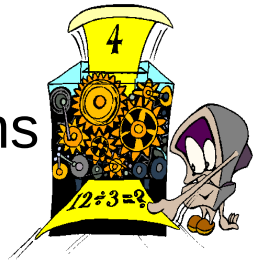
Predict system performance
- *online and batch*

Size systems

Evaluate & improve designs

Plan capacity

Plan testing



Security is an example of a run-time quality and covers these topics

zühlke
empowering ideas

Safety

- To reduce or eliminate danger
- To reduce or eliminate anxiety
- To reduce or eliminate risk or liability

Protection

- To defend against attacks (insider and outsider)
- To defend against fraud (misuse of assets or misrepresentation of identity)
- To defend tangible assets (IT systems and applications or stored information or information in transit)
- To defend intangible assets (reputation)

Assurance

- To ensure correct and reliable operation
- To enforce identity and ownership
- To promote trust

Security & Safety

zühlke
empowering ideas

Security is a wide and fascinating topic encompassing a vast range of issues, arenas and disciplines

- from deep mathematics to international espionage

In IT systems, "security" can be associated with the following qualities:

- Not open to intentional misuse
- Not open to accidental misuse
- Protects the truth – maintains integrity
- Protects service in the face of attack (overlap with Availability)

Secure means SAFE:

- Your data, your assets, your reputation

A good general approach to tackling IT security is to take a 'threat-based' approach

zühlke
empowering ideas

- **Document assets**
  Identify and decide what you need to protect. This could be data, intellectual capital, processes, physical resources, or any other thing of value in the organisation
- **Understand threats**
  Know your enemy. Determine from whom or what are you protecting your system and/or network
- **Define policy**
  Create a comprehensive security policy and implementation plan which is appropriate to the level of threat
- **Implement policies**
  Apply the security policies to your organisation and systems. Update or include security elements and configurations in IT solutions
- **Monitor policy**
  Continually monitor to detect any deviation from your policies and take actions if needed

Key objectives of Security Engineering (1/2)

zühlke
empowering ideas

Authentication – knowing who

- The process of determining who users (human or otherwise) are and that they are who they claim to be. The most common technique for authenticating is by user ID and password. Others include certificate-based methods or biometrics

Authorisation – knowing what can they do

- The process of establishing the 'rights' that a user has to access and to perform actions on resources. (Simple example – the permissions to read and/or write a file)

Confidentiality – protecting confidential data

- Ensuring that data classed as confidential is only seen by appropriately authorised parties. Often achieved through cryptography – i.e. encrypting data

Key objectives of Security Engineering (2/2)

zühlke
empowering ideas

Integrity – protecting the "truth"

- The quality of a system whereby data and processing always conforms to the specified rules and constraints within the system

Auditable – what did they do?

- The trail of evidence proving the activities that have been performed on an internal asset – and attributing this to a known identity. This must be stored in a non-repudiable (tamper proof) format.

Non-Repudiation – proving what happened happened

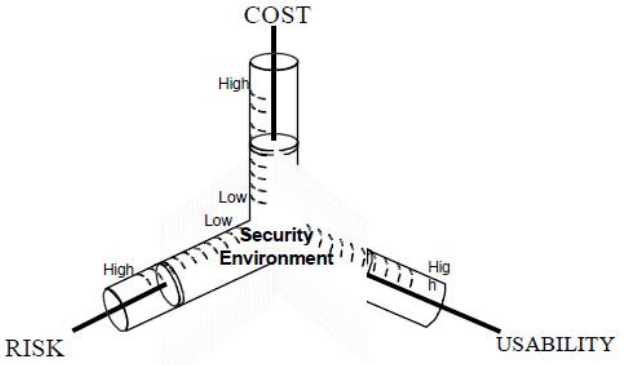
- The ability to prove without contradiction that a transaction or event which is recorded as having taking place did take place May need to be able to prove events in a court of law

Security architecture is about answering the question “how much security is enough security”

zühlke
empowering ideas

From a security perspective, all IT solutions must balance three conflicting factors:

- **The risk** – to the organisation of operating the IT solution
- **The cost** – of implementing and operating the security controls in general, the tighter the controls the lower the risk
- **The usability** – of the solution in general, the tighter the controls, the greater the impact on the users of the system



The resulting set of controls must be, as far as possible “**necessary** and **sufficient**”.