W4156

Requirements Engineering II

Agenda

- ❏ Woops: We missed a whole category of requirements!
- ❏ Non-Functionals
- ❏ Complete View of Requirements

Functional

In last lecture we almost exclusively focussed on *functional requirements*:

Functional Requirement: the behavior of a system under defined scenarios
(think function of system)

Be careful what you optimize for ....

- Many years ago I was at Lehman Brothers (that is a whole other story*)
- The bank had been experimenting with outsourcing
  - Very detailed functional specifications to process trades was written
  - The outsourcer built it
  - It met the specification provided and failed!

* though one of my favorite stories is the interview process consisted of “declare then defend your favorite ice-cream for 45 seconds”

What did we not specify?

- The specification focussed on what the system needed to ‘do’
  - “Take in these trades and add them in this way and produce this result”
  - The system did this perfectly

However,

- We needed >2000 trades **per second**. It managed 200 a **minute.**
  - It failed to meet ***performance*** **needs**

- Code was unreadable. 10k LOC across 4 ‘God Classes’.
  - It was not ***maintainable***.

What Happened?

The ***non-functional*** requirements were not defined or satisfied

Non-Functionals

Nonfunctional Requirement(NFR): A description of a property or characteristic that a system must exhibit or a constraint that it must respect.
(think “its not about it’s function - its about something else”)

Note:

A) Non-Functionals are *often fatally overlooked*

Non-Functionals

Quality attribute: One kind of nonfunctional requirement that describes a service or performance characteristic of a product. (usability, security, availability, capacity, performance, etc etc). Often described as the ‘ilities’

Note:

A) Industry usage often imprecise (NFR equated to be Quality Attribute)

Quality Attributes

External Quality	Internal Quality
Observable from the ‘outside’ view of the software product / watching execution	Describes the internal quality of the software / exposed to the development team

Quality Attributes

External Quality	Internal Quality
Availability	Efficiency
(Data) Integrity	Modifiability
Interoperability	Portability
Performance	Reusability
Reliability	Scalability
Robustness	Verifiability
Safety	
Security	
Usability	
Cost	

Each of these is a field of study .....

1. *defining* the requirement
2. engineering techniques to meet these goals .....

BagelRush QAs in Imprecise Language

NFR	Internal Quality
Availability	Obviously if our site goes down that is bad. 100% uptime is impossible/expensive. Our business wants 99.<how many 9s?> and are we willing to pay for it?
Performance	Can be defined for multiple paths/points. Often discuss user perspective and how long certain processes/clicks/responses take? Are there other flows that need performance to be defined?
Security	Depending on how we design it we may have credit card data? We will also store people routes? That would be embarrassing if hacked. So we would want to protect?
Usability	We may think of this for our users but there are also legal requirements for users with disabilities.
Scalability	How many concurrent users/order during peak time?
….	…...

Performance

In safety critical systems the value of ‘the answer’ decays over time (late is useless)

In consumer software latency cost money

Amazon found each additional **100ms** of latency reduced sales by **1%**

Note: consider all business processes (not just front end)

Note 2: Latency != Capacity

Availability

Is the system available to use when required

Uptime is one measure:

Uptime Percentage	Mins Down / Yr
99.999	5m
99.9	8 hr 45 min
99.0	3d 15hr

System with 99% can fail once a year for 3d or every day for 11 mins. This leads to many other measures: MTTF, MTBF, MTTR, RTO, RPO (not in exam). We need to consider the physical world (fire, flood, hard drive failure), etc.

(We don't have time to go into each quality attribute. But I want to give a flavor that they each are a 'sub discipline' of *defining* and *designing)*

Security

The security climate is well documented

Security: The requirement for the system to ensure only authorized access to functionality and data.

Note: includes hacking but related to business rules on entitlement (who can see what)

Scalability

What does Kim Kardashian have to do with Scalability of software systems?

Architecture Article

Cost

Is cost a non-functional requirement?

Quality Attributes

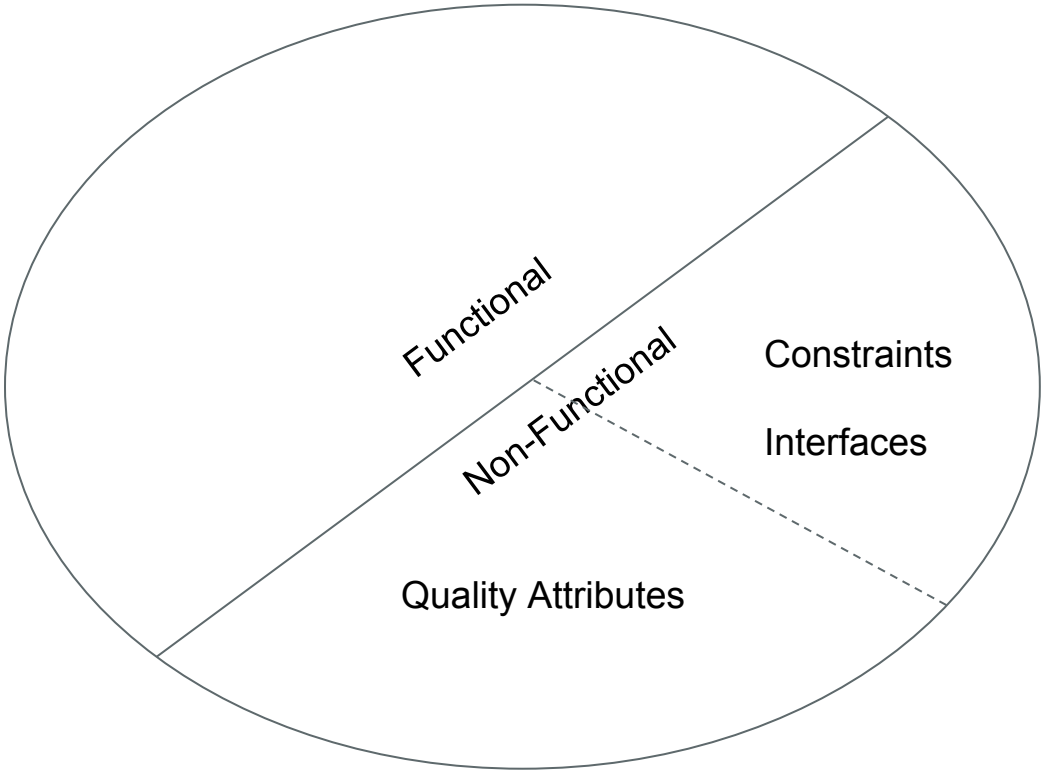
Question:

1. Do you think that different projects have different non-functionals?

2. Is there an *engineering cost* to achieving quality attributes?
   Can you give an example?

3. Do any quality attributes impact each other?
   (increasing one makes it easier/harder to achieve other?)

Venn Diagram of Requirements



Recap of Key Learning Outcome

1. There are both functional and non-functional requirements
2. Non-functional decomposes into quality attributes, interfaces and constraints
3. Both can cause projects to fail if not met
4. Requirements can interact with each other (conflict)
5. Requirements ‘cost’ design time or solution cost

Pop Quiz

Question	Our answer
Can a project can fail due to not meeting <F/NFR/Both>?	
Do requirements cost design/development time?	
Should we design for infinite <quality X>?	
“The software should cope with all future requirements” is a <genius/idiotic> requirement?	

Reading

Material	Optionality
Finish Beginning S.E. Chapter 4 (across this and next lecture)	Required
Paper Engineering (but look for/identify the NFRs they were engineering for)	Required