



CSE 403, Software Engineering Lecture 6

Non-functional specification



UW CSE Colloquium 10/10/02

- Agnes Kwan (UW CSE '82)

"The one thing I didn't learn in school was how to gather requirements from users"



Today

- Complete discussion of requirement representation
- Non-functional requirements



Recap from Wednesday

- Multiple customers for requirements
- Multiple representations of requirements

- Process for managing requirements



Redundancy: Good or bad?

- Multiple forms of documentation
 - Used for different audiences or views
 - But risk of inconsistency
- Functional spec and user spec
 - Should describe the same thing!
 - But what if they differ
 - Isn't that Test's job?
- Development principle
 - Avoid computing the same thing in multiple places

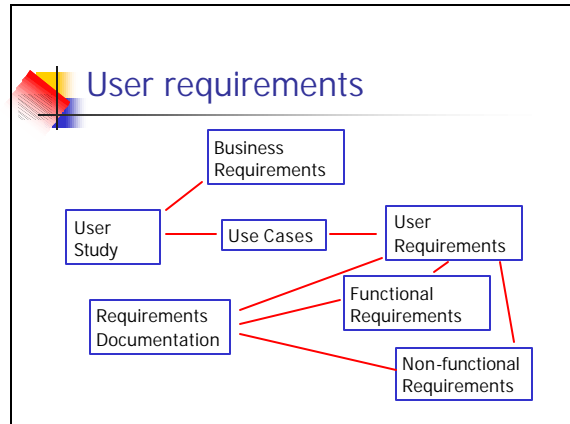


Brooks on flow charts

- The flow chart is the most thoroughly oversold piece of program documentation
- I have never seen an experienced programmer who routinely made detailed flow charts before beginning to write programs. Where organization standards require flow charts, these are invariably done after the fact.

Flow charts

- "The emperor has no clothes"
- Formal processes
 - Many processes are onerous and unpleasant to follow – but enhance overall product quality
 - Some are without value, and should be dropped
 - Process is not the end in itself



Non-functional requirements

- Requirements beyond user interaction with the system
- Kulak and Guiney
 - Availability, cost of ownership, maintainability, data integrity, extensibility, functionality (?), installability, reuse, operability, performance, portability, quality, robustness, scalability

Non-functionality requirements


- Wiegers
 - Performance requirements
 - Safety requirements
 - Security requirements
 - Software quality attributes

Safety requirements

- Safety critical applications
 - Where bugs can kill
- Famous cases
 - Therac-25 radiation therapy machine
 - US Air traffic control which failed in UK
 - Reflected map on Greenwich Median
 - US Aviation software failed in Israel
 - Encountered negative altitudes over Dead Sea


Safety critical systems

- Very high cost of failure
- Software component of a large system
 - e.g. nuclear reactor
- Characteristics of software lead to failures
- Safety requirements
 - Low probability of failure (risk analysis)
 - Understood failure modes



Software Safety

- Safety vs. Reliability
 - Safety
 - Reliability
- System hazard analysis
 - High risk tasks
 - Safety critical operator errors
- Design of Human-Machine Interface




Security requirements

- Applications are run in a hostile world
- Application compromise vs. system compromise
- Example requirements
 - Only authenticated users can change data
 - Application can change security permissions or execute programs
 - Malicious user cannot crash system with bad data
- Threat analysis




Security requirements for multiplayer games

- Cheating ruins game play (and consequently market)
- Threats
 - Players introducing counterfeit weapons
 - Sending packet of death across network
 - Using profiling tools to detect areas of activity in dungeons



Threat modeling

- The STRIDE Threat Model
 - Spoofing identity
 - Tampering with data
 - Repudiation
 - Allow users to deny having performed actions
 - Information disclosure
 - Denial of service
 - Elevation of privilege



Useful references

- Writing Secure Code, Michael Howard and David LeBlanc
 - Good book, but strongly oriented towards Windows
- Safeware: System Safety and Computers, Nancy Leveson
 - Defines the field of software safety