ILLINOIS INSTITUTE OF TECHNOLOGY

Transforming Lives. Inventing the Future.

www.iit.edu

SOFTWARE QUALITY MANAGEMENT CSP587

Prof. Dennis Hood

Computer Science

Lesson Overview

- Software Quality Requirements
- Reading:
  - Ch. 3 – Software Quality Requirements
- Objectives
  - Examine the basis of software requirements and the perspective of the customer
  - Analyze the process of capturing and documenting effective requirements
  - Discuss the classification of requirements

Topics for Discussion

- Customer vs. User vs. Engineer
  - Perspectives and priorities
  - Language barriers
  - Common goals and common challenges
- The process of capturing requirements and the process of documenting them
- Gaining efficiency and effectiveness by relying on past efforts
- Preparing to prove success

Week 4 Software Quality Requirements

Software Quality Assurance

- Quality Model
  - A defined set of characteristics, and of relationships between them, which provides a framework for specifying quality requirements and evaluating quality.
  - Implication - Quality can be measured
- Evaluation
  - A systematic examination of the extent to which an entity is capable of fulfilling specified requirements.

Perspectives of Quality

- Transcendental approach – I know it when I see it (or at least I know lack of quality when I see it – because that's much easier to see anyway)
- User-based approach – fit for purpose
- Manufacturing-based approach – compliance with processes leads to quality
- Product-based approach – the software engineer focuses on the internal view of the product (e.g., architecture, source code components, etc.)
- Value-based approach – focuses on the elimination of activities that do not add value

Initial Quality Factors and Criteria

- Correctness
  - Traceability, completeness, consistency
- Reliability
  - Consistency, accuracy, error tolerance
- Efficiency
  - Execution efficiency, storage efficiency
- Integrity
  - Access control, access audit
- Usability
  - Operability, training, communicativeness

Current Quality Factors

- Performance Efficiency
- Functional Suitability
- Compatibility
- Usability
- Reliability
- Security
- Maintainability
- Portability

Requirements

- Classification
  - Functional
  - Non-functional
  - Performance
- Good software requirements are:
  - Necessary
  - Unambiguous
  - Concise
  - Coherent
  - Complete
  - Accessible
  - Verifiable

User Requirement Challenges

- Ambiguity
  - Clarity is difficult to achieve
  - Especially since brevity is also desirable
  - Human language is different than user language is different than system language
- Confusion
  - Functional vs. non-functional vs. system goals vs. design information
  - Confusion over how/where to capture requirements can lead to documentation issues
- Amalgamation
  - A single stated requirement may actually contain several requirements

User <-> Engineer Interaction

- Establish a standard format and adhere to it
- Use language consistently
  - Mandatory requirements use “shall”
  - Desirable requirements use “should”
- Highlight to distinguish key elements
  - Bold, italic, etc.
- Resist the use of technical jargon

System Requirements Challenges

- Although undesirable, some design / implementation language may be necessary
  - For example, architecture, interoperability, etc.
- Natural language is ambiguous
- Natural language allows for saying the same thing in multiple distinct ways
- Relating related requirements is difficult using natural language

Specification Notations

- Structure natural language
  - Human language with standard forms / templates
- Design description languages
  - Similar to psuedo-code
- Graphical notations
  - E.g., use-case and sequence diagrams
- Mathematical specifications
  - Based on mathematical concepts such as finite-state machines or sets

The Requirements Document

- Preface
- Introduction
- Glossary
- User requirements
- System architecture
- System requirements
- System models
- System evolution
- Appendices
- Index

Elicitation and Analysis

- Working with stakeholders to “discover” requirements
- Challenges
  - Stakeholders don’t always know exactly what they want
  - The terminology gap may be huge
  - Different stakeholders have different needs
  - Lack of “ownership” may lead to politically-swayed requirements
  - Change happens (a lot)
- The process
  - Discovery, classification, prioritization, and documentation

Requirements Validation

- A checkpoint for ensuring that the requirements as specified truly define the system the customer wants
- A gate that should not be passed without a fight
- Things to look for
  - Validity (necessary and sufficient)
  - Consistency (no conflicts)
  - Completeness
  - Realism (feasible relative to existing technologies
  - Verifiability (how will it be tested?)