ILLINOIS INSTITUTE OF TECHNOLOGY

Transforming Lives. Inventing the Future.

www.iit.edu

Software Engineering CS 487

Prof. Dennis Hood

Computer Science

Lecture 3 Requirements Engineering

Lesson Overview

- Requirements Engineering
- Ch. 4 – Requirements Engineering
- Objectives
  - Discuss the concept of requirements in the context of software engineering
  - Distinguish between functional and non-functional
  - Describe the relationship between requirements and testing
  - Analyze the communication processes at the core of requirements gathering and validation
  - Discuss opportunities for reuse

Participation – P2

- Describe the role of test case development in improving requirements. Show an example to illustrate this improvement.
- Automation is generally beneficial since it promises to save time, effort, and money. Discuss the feasibility of automating each of the following software engineering activities:
  - Requirements engineering
  - Coding
  - Testing
  - Discovering the root cause of a production incident
- Choose any "real-world" example to explain the use of ethnography in the analysis and design of software systems.

Requirements Defined

Requirements Defined

- Purpose
  - Establish the needs of the user and the constraints of the environment
- Formality and detail
  - Initially left open to interpretation (contract bids)
  - Progress to specify what the system must do and how it must do it (binding contract)
- Perspectives
  - User – the functionality and performance expected by the user
  - System – precise specification of what is to be implemented

Types of Requirements

- Functional
  - Describe what the system should do
  - How the system should react to certain inputs
  - How the system should behave in certain situations
- Non-functional
  - Constraints on the services or functions offered by the system
  - Often apply to the system as a whole, not directly concerned with specific functions
  - System performance, security, availability, etc.
- Domain
  - Derived from the system's domain

Types of Non-functional Requirements

- Product
  - Execution speed, memory requirements, acceptable failure rate, portability, usability, etc.
- Organizational
  - Policies and procedures
  - Process standards, programming languages, methodologies, tools, delivery timeframe, etc.
- External
  - Interoperability with other systems, legal requirements, ethical requirements, etc.

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

Capturing Requirements

Helpful Hints

- 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
  - Including pseudo-code
- Graphical notations
  - Flowcharts, use-case and sequence diagrams, etc.
- Mathematical specifications
  - Based on mathematical concepts such as finite-state machines or sets

Interface Specification

- Clearly define the boundaries and manner in which information / commands / requests will pass through
- Procedural (APIs)
- Data structures
- Representations of data that have been established for an existing subsystem

The Requirements Document

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

Requirements Engineering

Requirements Engineering

- Process goal
  - To create and maintain a system requirements document
- Process steps
  - Feasibility study
  - Requirements elicitation and analysis
  - Requirements specification
  - Requirements validation
- Corresponding outputs
  - Feasibility report
  - System models
  - User and system requirements
  - Requirements document

Feasibility Studies

- Questions to be addressed
  - Should we build it?
  - Are we currently capable of building it?
  - Can we become capable of building it?
  - Will the resultant system integrate with existing systems?
  - Will we be able to maintain it?

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 Discovery

- Understanding multiple viewpoints
  - Interactor viewpoints
  - Indirect viewpoints (influencers)
  - Domain viewpoints
- Interviewing stakeholders
  - to learn their job, constraints, needs, etc.
- Capturing scenarios
  - Real-life stories
- Use-cases
  - Describing typical user interactions

Ethnography

- An observational technique used to understand social and organizational requirements
  - Discovering user perspectives
- Effective at discovering
  - Requirements that are derived from the way in which people *actually* work rather than how they are *supposed* to work
  - Requirements that are derived from cooperation and awareness of stakeholder’s activities

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
  - Clarity (freedom from ambiguity)
  - Consistency (no conflicts)
  - Completeness (is anything missing?)
  - Feasibility / Validity (necessary and sufficient)
  - Verifiability (how will it be tested?)