Software Development Fundamentals

CSC490 - Spring 2006

Fundamentals

- ✓Management
- ✓Technical
- ✓Quality-Assurance

Management Fundamentals

- ✓Determining size of product (functionality, complexity, …)
- ✓Allocating resources appropriately
- ✓Creating a plan for allocating resources
- ✓Monitoring & directing resources

Management = Planning

- ✓Poor planning -- #1 source of project problems (Metzger 1981)
- ✓Good planning
  - Estimation & scheduling
  - Determining & organizing project team
  - Choosing lifecycle model
  - Managing risks
  - Strategic decision-making

Estimation & Scheduling

- ✓Estimate size of product
- ✓Estimate effort needed to build product of that size
- ✓Estimate schedule based on effort estimate

Tracking

- ✓ Management-level
  - Task lists
  - Status meetings/reports
  - Milestone reviews
  - Budget reports
  - Walking around
- ✓ Technical-level
  - Technical audits/reviews
  - Quality gates to review milestones completed

Technical Fundamentals

- ✓ "Modern programming practices"
  - No high productivity w/out them (Vosburgh '84)
- ✓ Requirements Management
- ✓ Design
- ✓ Construction
- ✓ Software Configuration Management

Requirements Management

- ✓ Gathering requirements
- ✓ Recording in document/email/…
- ✓ Tracking design & code against them
- ✓ Managing changes over course of project
- ✓ Too rigid?
  - Top 3 reasons for late projects (study of 8000): lack of user input, incomplete requirements, changing requirements (Standish Group 1994)

Architecture & Design

  - 10x longer to fix errors @ system testing than @ design stage (Dunn 1984)
- ✓ Fundamentals
  - Major styles: OO, structural, data-structure design
  - Foundational concepts: information hiding, modularity, abstraction, encapsulation, cohesion, coupling, hierarchy, inheritance, polymorphism, basic alg. + data structures
  - Approaches to challenging aspects: exception handling, internationalization, portability, I/O, data storage, f.p. arithmetic, database design, performance, reuse, …
  - Application domain considerations: financial, scientific, embedded, real-time, safety-critical, …

Construction

- ✓ Most of groundwork for success/failure already laid
  - Requirements & design offer greater leverage on development schedule than construction
- ✓ Poor construction practices => introduce subtle errors take days/weeks to find/fix
  - E.g. off-by-one ('+1') array declaration error

Construction Fundamentals

- ✓ Coding practices (naming, layout, docum.)
- ✓ Data-related concepts (scope, persistence)
- ✓ Guidelines for data types
- ✓ Control-related concepts (conditionals, loops, complexity, goto/return, recursion)
- ✓ Assertions/error-detection
- ✓ Rules for packaging code in routines/modules/classes/files
- ✓ Unit-testing & debugging
- ✓ Integration strategies
- ✓ Code-tuning
- ✓ Language particularities
- ✓ Use of construction tools (IDE, source code control, libraries, code generators)

Software Configuration Management

- ✓ Managing project so it stays consistent over time
- ✓ Practices for…
  - Evaluating proposed changes
  - Tracking changes
  - Handling multiple versions
  - Keeping copies of project artifacts from various times

Quality Assurance Fundamentals

✓ Key to avoiding/preventing software defects

✓ Error-prone modules
✓ Testing (most common practice)
- Unit tests (by developer)
- System tests (by independent tester)

✓ Technical reviews
- Vary in formality & effectiveness: walkthroughs, code reading, inspections

Introduction to Requirements

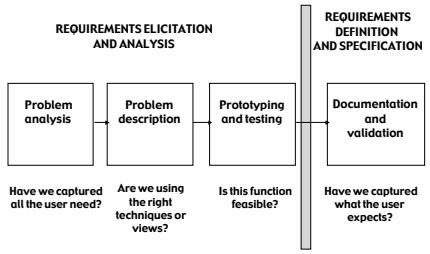
✓ Definition

> *"A feature of the system or a description of something the system is capable of doing in order to fulfill the system's purpose"*

✓ Types of Requirements
- Functional
- Non-functional

✓ Strengths
- Must/Shall
- Should
- May

Requirements Engineering



Requirements

- Goal
  - To understand the problem
- Necessary to Understand Requirements
  - Organization
  - Existing Systems
  - Processes
  - Improvements
- Once you have all this information, now what?

Requirements Elicitation

✓ Techniques
- Interview / Meeting
- Survey / Questionnaire
- Observation
- Ethnography / Temporary Assignment
- Business Plans
- Review Internal / External Documents
- Review Software

Requirements Analysis

- Goal
  - To bridge the gap between the problem domain and the technical domain
- Tasks
  - Problem Recognition
  - Evaluation and synthesis
  - Modeling
  - Specification
  - Review

Requirements Analysis Principles

- Information domain of a problem must be represented and understood
- Models that depict system information, function, and behavior should be developed
- Models must be partitioned in a manner that uncovers detail in a layered fashion
- Analysis process should move from essential information toward implementation detail

Requirements Review?

- Are the requirements complete?
- Are the requirements concise?
- Are the requirements correct?
- Are the requirements consistent?
- Are the requirements modular? Can they accommodate change?
- Are the requirements realistic?
- Is the requirement needed by the customer?
- Are the requirements traceable?

Class Work

✓Set up basic project home page

✓Research software requirements specification templates on Internet