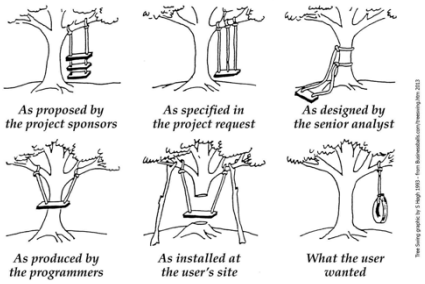
CIS 422/522

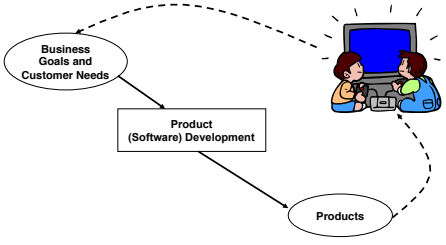
Software Requirements
and a little Quality Assurance

Understanding Software Requirements (and why we get it wrong so often)

"Problem solving is an art form not fully appreciated by some"

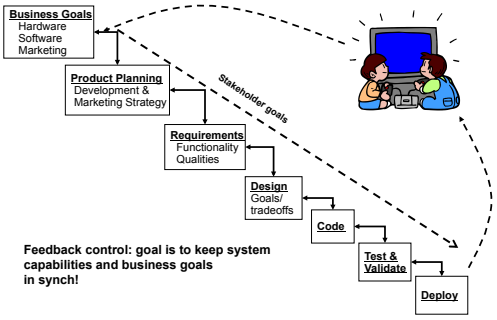


10,000 ft. View



What should the development process accomplish?

Product Development Cycle



Feedback control: goal is to keep system capabilities and business goals in synch!

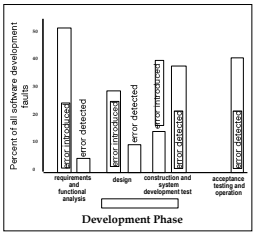
What is a "software requirement?"

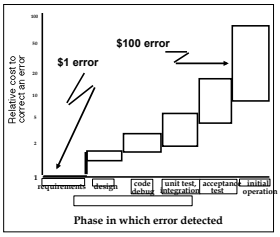
- *Definition*: A description of something the software must do or property it must have
- The set of system requirements denote the problem to be solved and any constraints on the solution
  - Ideally, requirements specify precisely what the software must do without describing how to do it
  - Any system that meets requirements should be an acceptable implementation

Importance of Getting Requirements Right

1. The majority of software errors are introduced early in software development

2. The later that software errors are detected, the more costly they are to correct





Requirements Phase Goals

- What does "getting the requirements right" mean in the systems development context?
- Only three goals
  1. Understand precisely what is required of the software
  2. Communicate that understanding to all of the parties involved in the development (stakeholders)
  3. Control production to ensure the final system satisfies the requirements
- Sounds easy but hard to do in practice
- Understanding what makes these goals difficult to accomplish helps us understand how to mitigate the risks

"The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements...No other part of the work so cripples the resulting system if done wrong. No other part is as difficult to rectify later."

F.P. Brooks, "No Silver Bullet: Essence and Accidents of Software Engineering"

What makes requirements difficult?

- Comprehension (understanding)
  - People don't (really) know what they want (…until they see it)
  - Superficial grasp is insufficient to build correct software
- Communication
  - People work best with regular structures, conceptual coherence, and visualization
  - Software's conceptual structures are complex, arbitrary, and difficult to visualize
- Control (predictability, manageability)
  - Difficult to predict which requirements will be hard to meet
  - Requirements change all the time
  - Together can make planning unreliable, cost and schedule unpredictable
- Inseparable Concerns
  - Many requirements issues cannot be cleanly separated (I.e., decisions about one necessarily impact another)
  - Difficult to apply "divide and conquer"
  - Must make tradeoffs where requirements conflict

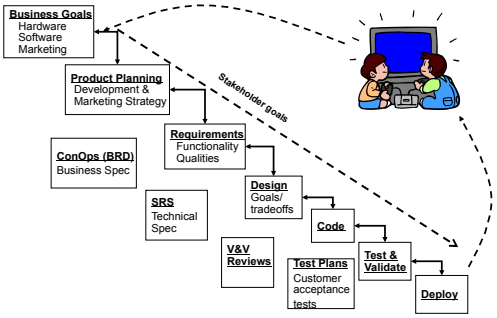
Requirements Process

Understand, Communicate & Control

A good process helps manage requirements difficulties requires having

1. Requirements Understanding (Understand)
   - Elicitation - How do we establish "what people want?"
   - Negotiation - How do we resolve stakeholder conflicts?
2. Requirements Specification (Communicate)
   - Concept of Operations (ConOps) - How do we communicate with non-programmer audiences?
   - Software Requirements Specification (SRS)- How do we specify precisely what the software must do?
3. Requirements V&V (Control)
   - Validation- How do we establish that we have the right requirements?
   - Verification - How do we establish that the implementation is consistent with the specification?

Related Products



1.1 Elicitation

- Goal: Understand precisely what is required of the software
  - Answer the question, "What do the stakeholders want?"
  - Stakeholder: define as anyone with a valid interest in the outcome of a software development
- Inherently open-ended, ambiguous question
- Addressed by a number of elicitation methods
  - Interview – traditional standard
  - Focus groups
  - Prototyping
  - Scenario analysis (next), etc.
- All have differing costs, strengths, and weaknesses. None is a complete solution
  - Use more than one approach
  - Check the results *early and often*

1.2 Requirements Negotiation

- or "Why the customer is not always right."
- Stakeholders' requirements often conflict
  - Needs of different customers/users may conflict
    - E.g., Salesmen want convenience and speed, management wants security and accountability
  - Developer's needs may conflict with customer's
    - E.g., development cost vs. customer desires
- Choosing which requirements should be addressed and their relative importance requires *negotiation* and *tradeoffs* among stakeholders

2. Requirements Specification

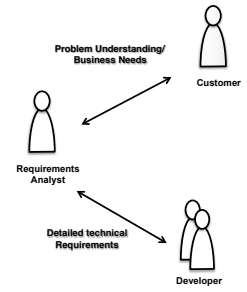
- Goal: Communicate requirements understanding to all system stakeholders
- Q: What kinds of information need to be communicated?
  - System context
    - System stakeholders
    - Business goals
    - System purpose
    - Interfacing systems (if any)
  - System requirements
    - Behavioral requirements
    - Quality requirements

Purposes and Stakeholders

- Many potential stakeholders using requirements for different purposes
  - Customers: document what should be delivered, may provide the contractual basis for the development
  - Managers: provides a basis for scheduling and a yardstick for measuring progress
  - Software Designers: provides the “design-to” specification
  - Coders: defines the range of acceptable implementations and is the final authority on the outputs that must be produced
  - Quality Assurance: basis for validation, test planning, and verification
  - Also: potentially Marketing, regulatory agencies, etc.

Needs of Different Audiences

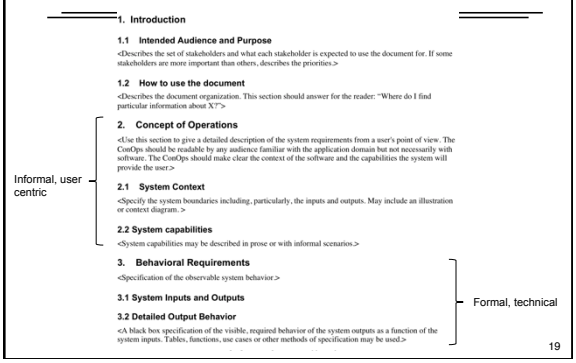
- Customer/User
  - Focus on problem understanding
  - Use language of problem domain
  - Technical if problem space is technical
- Development organization
  - Focus on system/software solutions
  - Use language of solution space (software)
  - Precise and detailed enough to write code, test cases, etc.



Two Kinds of Requirements Documentation

- Communicate with stakeholders who understand the problem domain but not necessarily programming :
  - e.g. customers, users, marketing
  - Do not understand computer languages but may understand technical domain-specific languages
  - Must develop understanding in common languages
  - Role of ConOps (Concept of Operations)
- Communicate with developers: sufficiently precise and detailed to code-to, test-to, etc.
  - Stated in the developer’s terminology
  - Addresses properties like completeness, consistency, precision, lack of ambiguity
  - Role of SRS (Software Requirements Specification)
- For businesses, these may be two separate documents

SRS Template



19

Documentation Approaches

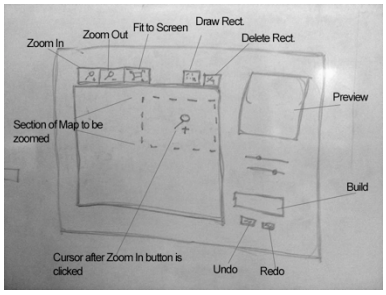
- Informal requirements to describe the system's capabilities from the customer/user point of view
  - Purpose is to answer the questions, "What is the system for?" and "How will the user use it?"
  - Tells a story: "What does this system do for me?"
  - Focus on communication over rigor
- More formal, technical requirements for development team (architect, coders, testers, etc.)
  - Purpose is to answer specific technical questions about the requirements quickly and precisely
    - "What should the system output for this set of inputs?"
    - Reference, not a narrative, does not "tell a story"
  - Goal is to develop requirements that are precise, unambiguous, complete, and consistent
  - Focus on precision and rigor

Informal Specification Techniques

- Most requirements specification methods are informal
  - Natural language specification
  - Use cases
  - Mock-ups (pictures)
  - Story boards
- Benefits
  - Requires little technical expertise to read/write
  - Useful for communicating with a broad audience
  - Useful for capturing intent (e.g., how does the planned system address customer needs, business goals?)
- Drawbacks
  - Inherently ambiguous, imprecise
  - Cannot effectively establish completeness, consistency
- However, can add rigor with standards, templates, etc.

Mock-up Example

Can use simple technology



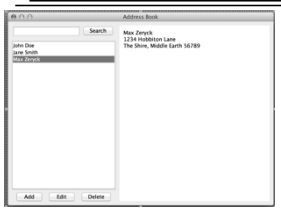
3. Validation and Verification

- Part of *Quality Assurance* – provides feedback in the feedback-control-loop
- *Validation*: activities to answer the question – "Are we building a system the customer wants?"
  - Familiar activity: customer review of prototype
- *Verification*: activities to answer the question – "Are we building the system consistent with all specifications?"
  - Most familiar verification activity is functional testing

Project V&V

- QA Goal: How can we establish whether the development is under control?
- Project sub-questions:
  - How will you establish that the system does what it should?
  - What is the role of testing?
  - What can testing establish about system quality (and what can't it)?
  - How will you write test cases?
    - E.g., for the class project

Customer Validation "Is this OK?"



- Usually get prototypes with questions of the form "Is this OK?"
- What answer are you likely to get?
  - What does the question actually mean?
- What do you actually want from the review?
  - What question do you actually want the customer/reviewer to answer?

Analysis and Informal Specification with Use Cases

Use Cases

- Use Case: a story describing how the system and a user interact to accomplish a user task
- A form of User Centered Analysis – capturing requirements from the user's point of view
  - Goal of helping identify user needs
  - Solve the right problem
  - Describe the "business logic" of the system
- Use cases specify a *subset of functional requirements*
  - Only system behavior observable to the user
  - Does not address non-functional constraints, qualities
- Use cases should not specify design or implementation (including UI design)

Scenario Analysis Process

Applying scenario analysis in the requirements process

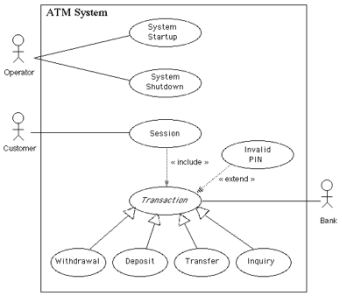
- Requirements Elicitation
  - Identify stakeholders who interact with the system
  - Collect "user stories" - how people would interact with the system to perform specific tasks
- Requirements Communication (ConOps)
  - Record as use-cases with standard format
  - Use templates to standardize, drive elicitation
- Requirements verification and validation
  - Review use-cases for consistency, completeness, user acceptance
  - Apply to support prototyping
  - Verify against code (e.g., use-case based testing)

Identifying Actors

- Actors – identifies the roles different users play with respect to the system
  - Roles represent different classes of users (users with different goals)
  - Actors carry out use cases
- Helps identify requirements for different kinds of users
  - "How would depositors use the system?"
  - "How would a library patron use the system?"
- Diverse classes of users may very different goals and require different interfaces
  - E.g., users vs. administrators vs. content providers

UML Graphic Example

http://www.math-cs.gordon.edu/local/courses/cs211/ATMExample/



Scenario Elicitation

- Each class of actor is interviewed and/or observed
  - How do you do task T?
  - How will the user interact with the system to do X?
- Collect in the form of "user stories"
  - Documented as scenarios (informal or standardized)
  - Identify relative priorities of tasks
  - Resolve conflicts, tradeoffs

Creating Use Cases

- Identify a key *actor* and *purpose*
  - The purpose informs the use case title and description
- Identify the main flow (ideal path) from the starting point to the result
  - Preconditions: anything that must be true to initiate the Use Case
  - Trigger: event, if any, initiating the Use Case
  - Basic Flow: sequence of interactions from the trigger event to the result
  - Alternative Flows: identify sequences branching off the Basic Flow

Guidelines for Good Use Cases

- Use Cases should express requirements, not design
  - Focus on import *results* that provide *value* to specific actors
    - I.e., if nobody really cares about the outcome, it is not a good use case
  - Focus on *what* the actor is doing, not the details of *how*
    - Not: "The user left-clicks on the radio button labeled *Balance* and presses the *Enter* button"
    - "The user elects the option to view the balance."
- Looking for a small number of use cases that capture the most important interactions
  - Read the IBM Use Case paper

Use-Case Specification – Register for Courses

Brief Description

This use case allows a Student to register for course offerings in the current semester. The Student can also modify or delete course selections if changes are made within the add/drop period at the beginning of the semester. The Course Catalog System provides a list of all the course offerings for the current semester.

Actors

1. ***Primary Actor – Student***
2. ***Secondary Actor - Course Catalog System***

Flow of Events

1. ***Basic Flow***

1.1. LOG ON.
This use case starts when a student accesses the Course Registration System. The student enters a student ID and password and the system validates the student.

1.2. CREATE SCHEDULE.
The system displays the functions available to the student. These functions are: Create A Schedule, Modify a Schedule and Delete a Schedule. The student selects 'Create a Schedule'.

1.3. SELECT COURSES
The system retrieves a list of available course offerings from the Course Catalog System and displays the list to the student. The Student selects up to 4 primary course offerings and 2 alternate course offerings from the list of available offerings. The student can add and delete courses as desired until choosing to submit the schedule.

1.4. SUBMIT SCHEDULE.
The student indicates that the schedule is complete. The system validates the courses selected and displays the schedule to the student. The system displays the confirmation number for the schedule. The systems saves the student's schedule information. The use case ends.

Address Book Example

- Who are the actors?
- What are the major tasks?
- What are the outcomes?
- What would be an alternative flow?

Summary

- Requirements characterize “correct” system behavior
- Being in control of development requires:
  - Getting the right requirements
  - Communicating them to the stakeholders
  - Using them to guide development
- Requirements activities must be incorporated in the project plan
  - Requirements baseline
  - Requirements change management

Questions?