CIS 422/522

NSF Project Assessment
Documenting Development Decisions
Documenting Requirements

Assessment Questionnaire

For each question chose only one answer!

1. Each foo of a system should be
a. A reusable component of a system.
b. An algorithm for producing certain required behavior in a system.
c. A work assignment for a programmer or small team of programmers.
d. The best possible foo of all foos.

Documenting Development Decisions

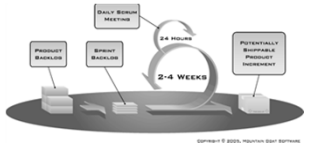
Choosing a Process (Review)

- Goal: proceed as rationally and systematically as possible from a statement of goals to a design that meets those goals with development constraints
- Choose a process to provide an appropriate level of control for the given product and context
  - Sufficient control to achieve results
  - No more than necessary to contain cost and effort
- Development goals: want to choose a process that supports project development and addresses risks
  - Schedule
  - Failure to deliver working software
- Instructional goals: process must also support learning software engineering
  - Provide experience with a range of artifacts for all team members
  - Support teacher evaluation

Which process for projects?

- Process viewed as a sequence of iterations, each iteration produces an increment of the working software
  - Build minimal useful subset, test, validate
  - After first iteration, always have working software
  - Document requirements, design, etc.

- Process viewed as nested sequence of builds (sprints)
  - Each build adds small feature set
  - Customer in loop, code centered (little or no documentation)
  - Problem detection and correction though daily team meetings (scrum)



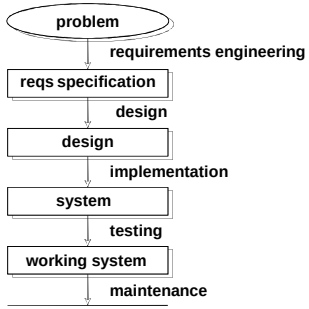
Course Approach

- Will learn a document-driven approach
- Provides broader experience with development roles, activities, and artifacts
- Supports external tracking and review
- Appropriate for a broader range of development situations
- Nothing additional is needed to switch to agile

Focus on Disciplined Process

- Focus on a disciplined development process
  - "Disciplined" means as systematic and rigorous as is **practical**
  - Basis for maintaining intellectual and managerial control
- A complete approach defines both *process* and *products*
  - Process: The (partially ordered) sequence of activities, entrance and exit criteria for each activity, which work products are produced by the activity, and what kinds of people should do the work.
  - Products: The work products to be produced and, for each, the resources needed to produce it, the information it contains, the expected audience, and the acceptance criteria it must satisfy.

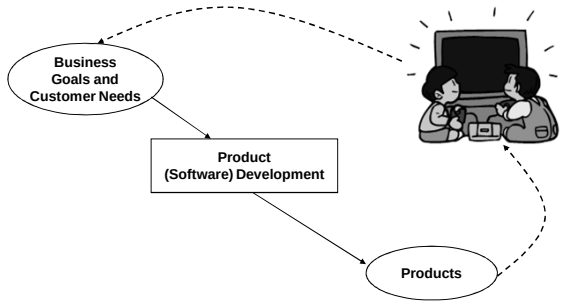
Simple Process & Products



From van Vliet

Why document?

10,000 ft. View

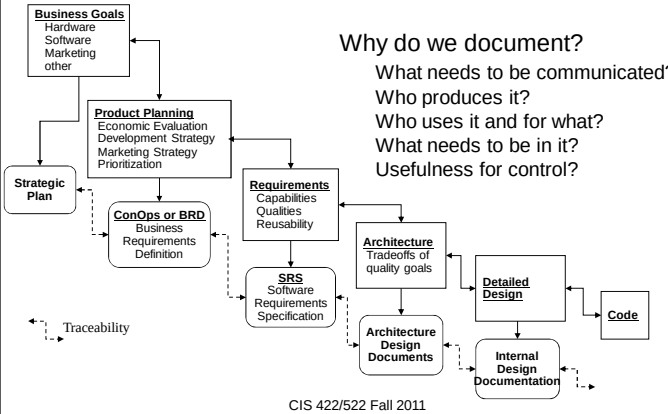


What should the development process accomplish?

Role of Documentation

- To understand what kind of documentation is useful, helps to understand the "why"
- Consider
  1. Goal is to turn an idea into a product
  2. Software engineering is a decision making process
  3. We decompose a complex development process into distinct concerns (requirements, design, code, test, deployment, maintenance, etc.)
- Why document?
  - What purpose does it play?
  - What kinds of things should be documented?

Product Development Cycle and Documentation



Why do we document?

- What needs to be communicated?
- Who produces it?
- Who uses it and for what?
- What needs to be in it?
- Usefulness for control?

Document Types and Purposes

- Management documents
  - Basis for project management (managerial control of resources)
    - Calendar time, skilled man-hours budget
    - Other organizational resources
  - Project plan, WBS, Development schedule
  - Use: allows managers to track actual against expected consumption of resources
- Development documents
  - Basis for product development (intellectual control)
  - ConOps, Requirements (SRS), Architecture, Detail design, etc.
  - Uses:
    - Making and recording development decisions
    - Allows developers to track decisions from stakeholder needs to implementation

Meeting Developmental Goals Means…

- We have a clear understanding of customer needs and product goals
- External view: We develop products the customer's wants, on time and within budget
- Internal view: We create process and product infrastructures supporting our business goals

Introduction to Software Requirements

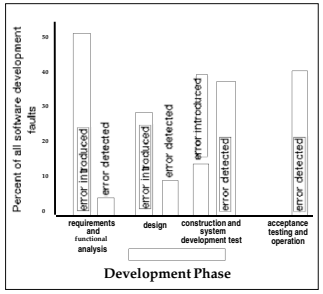
What is a "software requirement?"

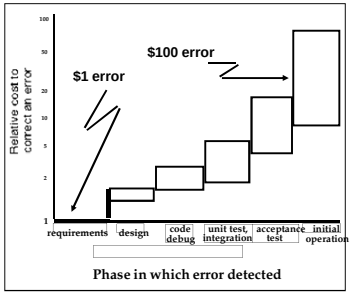
- A description of something the software must do or property it must have
- The set of system requirements denote the problem to be solved and any constraints on the solution
  - Ideally, requirements specify precisely what the software must do without describing how to do it
  - Any system that meets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

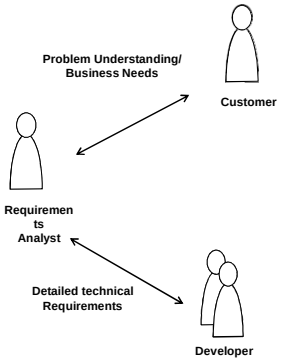
- Comprehension (understanding)
  - People don't (really) know what they want (…until they see it)
  - Superficial grasp is insufficient to build correct software
- Communication
  - People work best with regular structures, conceptual coherence, and visualization
  - Software's conceptual structures 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

Purposes and Stakeholders

- Many potential stakeholders using requirements for different purposes
  - Customers: the requirements typically document what should be delivered and 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 Software Requirements

- Communicate with customers: i.e., stakeholders who understand the problem domain but not necessarily programming (solution domain)
  - Do not understand computer languages but may understand technical domain-specific languages
  - Must develop understanding in common languages
- 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
- For businesses, these may be two separate documents

Documentation Approaches

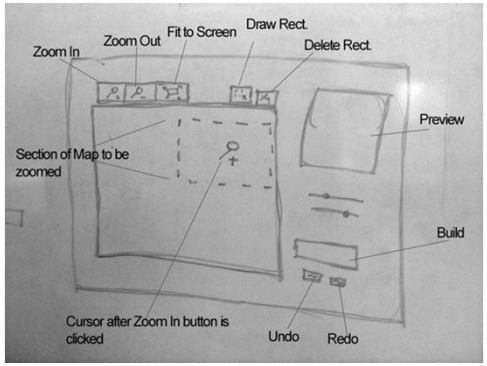
- Informal requirements to describe the system's capabilities from the customer/user point of view
  - Purpose is to answer the questions, "What is the system for?" and "How will the user use it?"
  - Tells a story: "What does this system do for me?"
  - Helps to use a standard template
- More formal, technical requirements for development team (architect, coders, testers, etc.)
  - Purpose is to answer specific technical questions about the requirements quickly and precisely
    - Answers, "What should the system output in this circumstance?"
    - Reference, not a narrative, does not "tell a story"
  - Goal is to develop requirements that are precise, unambiguous, complete, and consistent
  - What are the problems with use cases for this purpose?

Informal 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



Use Cases

~~1 Use Case: Manage Reports~~

1.1 Description
This Use Case describes operation for Creating, Saving, Deleting, Printing, Exiting and Displaying reports.

1.2 Actors
User
Project database

1.3 Triggers
Program Manager selects operations from menu.

1.4 Flow of events

1.4.1 Basic Flow
1. User chooses desired report by selecting "Report" -> "Open" from the menu bar
2. System displays report to screen
3. User selects desired report layout using Use Case Specify Report
4. Steps 2 and 3 are repeated until user is satisfied
5. User can Save or Print report using use case Save Report or Print Report
6. User Exits report by selecting "Exit" from the "File" menu

1.4.2 Alternative Flows

1.4.2.1 Create New Report
1. User selects "Create New Report" from file menu
2. ...

1.4.2.2 Delete Report
...

1.4.3 Preconditions
etc

A systematic approach to use cases

- **Uses a standard template**
- **Easier to check, read**
- **Still informal**

Technical Specification

The SRS

The role of rigorous specification

Requirements Documentation

- Is a detailed requirements specification necessary?
- How do we know what "correct" means?
  - How do we decide exactly what capabilities the modules should provide?
  - How do we know which test cases to write and how to interpret the results?
  - How do we know when we are done implementing?
  - How do we know if we've built what the customer asked for (may be distinct from "want" or "need")?
  - Etc…
- Correctness is a *relation* between a spec and an implementation (M. Young)
- Implication: until you have a spec, you have no standard for "correctness"

From Example SRS/SDS/Plan

SRS or Technical Reference

Informal Operational Requirements

Technical Requirements

- Focus on developing a technical specification
  - Should be straight-forward to determine acceptable inputs and outputs
  - Preferably, can systematically check completeness consistency
- A little rigor in the right places can help a lot
  - Adding formality is not an all-or-none decision
  - Use it where it matters most to start (critical parts, potentially ambiguous parts)
  - Often easier, less time consuming than trying to say the same thing in prose
- E.g. in describing conditions or cases
  - Use predicates (i.e., basic Boolean expressions)
  - Use mathematical expressions
  - Use tables where possible

Formal Specification Example

Type Dictionary

Name	Base Type	Units	Legal Values	Comment
Speed	Integer	Knots	[0, 250]	Speed measured in nautical miles per hour.
Weight	Integer	percent	[0,100]	Weighting for weighted average
time	Integer	seconds	time > 0	Time in seconds.

Monitored Variable Dictionary

Name	Type	Initial Value	Accuracy	Comment
LowResWS1	Speed	0	1	Wind speed reported by first low resolution sensor
LowResWS2	Speed	0	1	Wind speed reported by second low resolution sensor
HighResWS1	Speed	0	2.5	Wind speed reported by first high resolution sensor
HighResWS2	Speed	0	2.5	Wind speed reported by second high resolution sensor

Controlled Variable Dictionary

Name	Type	Initial Value	Accuracy	Comment
TransmWindSpeed	MsgType	ShortMsg	N/A	Transmitted value of wind speed

- SCR formal model
  - Define explicit types
  - Variables monitored or controlled

Formal Specification Example

TransmWindSpeed Event Function

The transmitted wind speed is a moving, weighted average over the length of the history of sensor readings defined as follows:

LW= LowResWeight, HW=HighResWeight, H = History

For *i* be the current count of all sensor readings and v[*i*] the ith sequential value of variable *v* (hence LWS1[i] is the most recent value of LWS1).

TransmWindSpeed Event Function

	Events
	@(TransmitPeriod)
TransmWindSpeed =	(LW* (LWS1[i] + LWS21[i] +… + LWS[i-H]+ LWS[i-H]) + HW *(HWS1[i] + HWS21[i] +… + HWS[i-H]+ HWS[i-H]) / H * (LW + HW)

Control Production

- Control production to ensure the final system satisfies the requirements: this means:
- Team members use the requirements specification as the basis of development
  - Designing the architecture: requirements define all external *design goals*, e.g.: expected changes, subsets, extensions, etc.
  - Designing modules to provide required capabilities
- Team members use the requirements spec as a basis for verification
  - Verify designs against requirements
  - Basis of system test planning (as opposed to module)
- If the spec is not adequate for these purposes, then fix the spec!

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

Assignment for next Tue.

- Reading
  - Requirements: Ch. 9
  - 522: Faulk paper, Brooks
- Project
  - First cut at assembla pages due
    - Team page (add a picture with names)
    - Current project plan with risks, mitigations, shcedule
    - Modified ConOps with any decisions and open issues
  - Schedule group meetings with instructor (may do some in class)