CIS 422/522

Software Documentation Overview
Documentation Principles

Notes:

Assignment

- Project
  - Set up meeting times
    - 12:00 – 3:30 Wednesday, Apr 13th
    - Room 160 Deschutes (first floor, Northwest corner)
  - Prepare walkthrough of your projects
    - Make an outline of topics
    - Describe development approach
    - Describe progress against milestones
  - Meet on Wednesday 30 min. or so

Notes:

Teams

Team 1 (Time:) Anson Henner Smith Biersdorf Thayer	**Team 2** (Time:) Jenkins Voll Fukushima Jensen	**Team 3** (Time:) Hosler Loiselle Winter McClure Otto
Team 4 (Time:) Vaughan Collins Strandlien Bryant Choi	**Team 5** (Time:) Matloff Hojnacke Sato Barker Lee	

Notes:

Contents

- Overview of Life-Cycle Products
- What is the role of documentation in a disciplined development?
- Characterizing distinct development concerns
- Documentation principles

Notes:

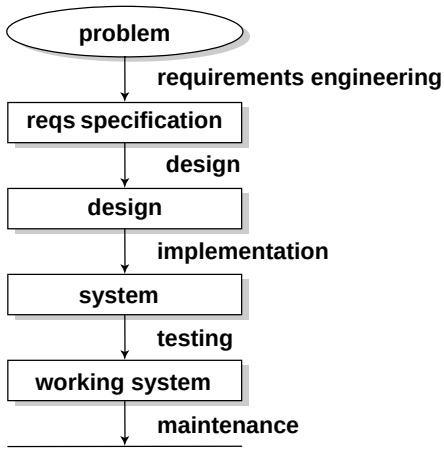
Focus on Disciplined Process

- 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.

Notes:

Review from last week: Focus of the course is on introducing discipline into the requirements understanding and requirements specification phases.

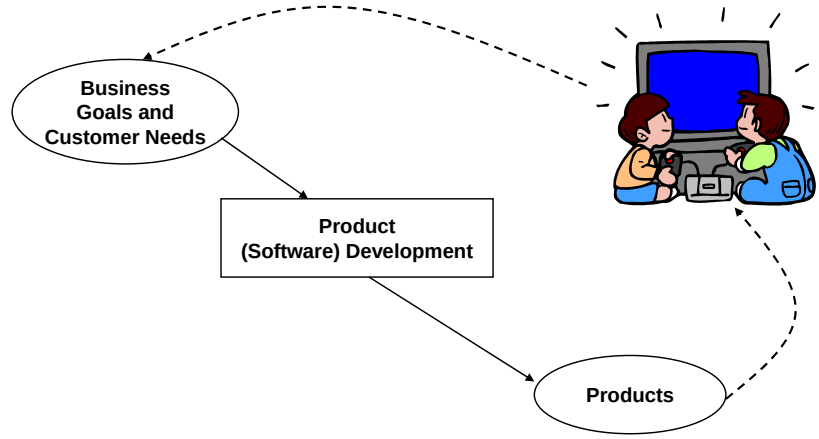
Simple Process & Products



From van Vliet

Notes:

10,000 ft. View

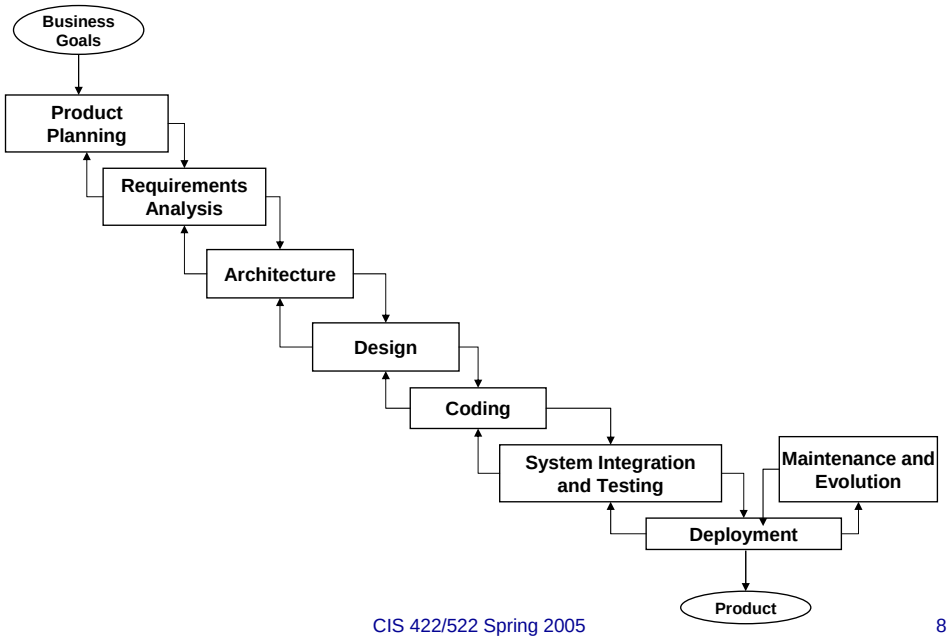


What should the development process accomplish?

Notes:

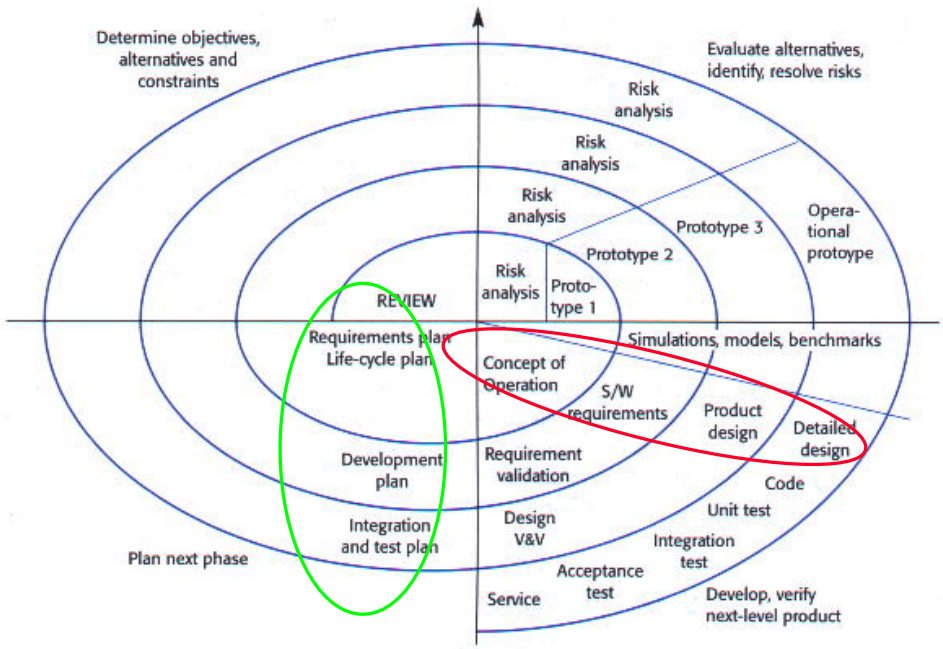
Everything else is scaffolding.

Iterative “Waterfall” Model



Notes:

Spiral Process Model (Boehm)



9

Notes:

Document Types

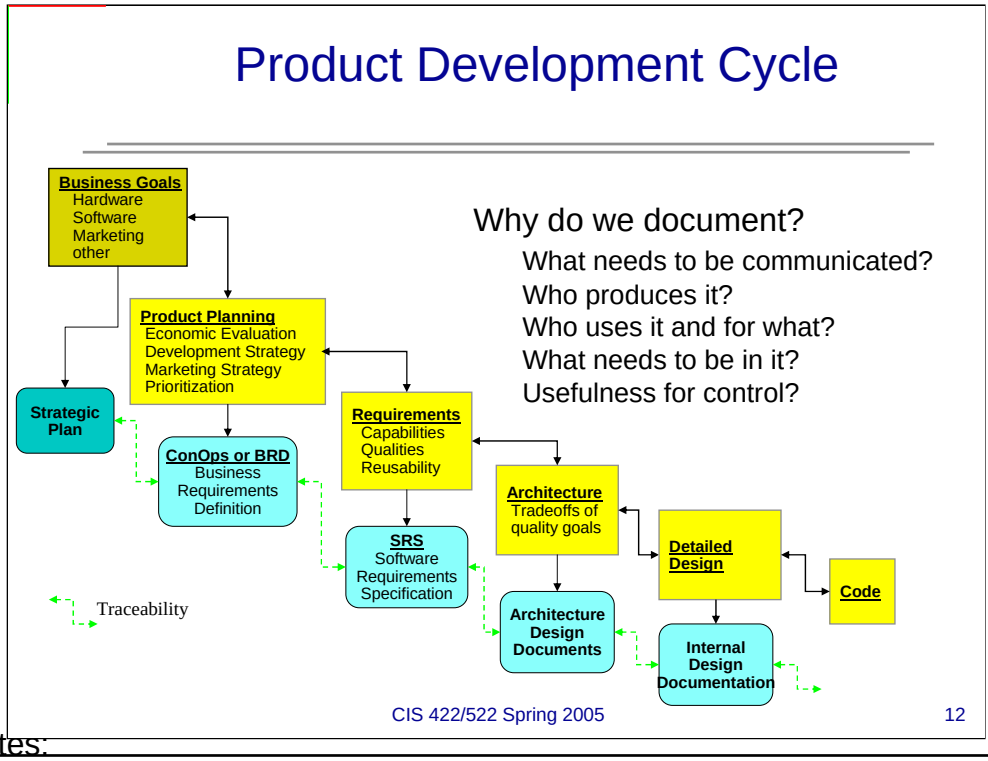
- Management documents
  - Basis for project management (control of resources)
  - Project plan, schedule, WBS, etc.
- Development documents
  - Basis for product development
  - ConOps, Requirements (SRS), Architecture, Detail design, etc.

Notes:

Questions

- What do we gain? (I.e., what is the benefit of having conceptually distinct phases and products?)
- What is the price? (What problems or issues are introduced?)
- What role do the documents play? How do they help?

Notes:



Notes:

This picture illustrates the correspondence between phases of the development process and the products produced. In particular, it shows the two kinds of requirements specifications, one owned by the business side (the BRD, MRD, or ConOps) and one owned by the technical side (the Software Requirements Specification). We will discuss the purpose and content of each of these in the following weeks.

Note, however, that these indicate distinct kinds of information. Not all organizations actually develop distinct documents.

Having Control Means…

- We maintain a clear understanding of customer needs and product goals
- The development process results in software that:
  - 1) Meets customer needs and product goals
  - 2) Is developed on time and within budget
- We maintain intellectual and managerial control throughout development ("plan the work then work the plan")
- Documentation records and communicates the results of understanding and planning
  - What do we need to do next?
  - Did we do the right thing?

Notes:

It makes no sense to talk about being in control if we do not know where we are going in the first place. Determining the destination (even if it may change over time) is the first, necessary, step.

Being in control has aspects that are driven by external concerns and aspects that are driven by internal (to the project or organization) concerns. External concerns are things customer needs. Internal concerns are things like reusability or plan of work.

Primary Documents

Notes:

Phases and Products

- Product Planning
  - Goal: Link organizational (e.g., business) goals (why) with product features (what)
  - Product: Concept of Operations (ConOps), Business Requirements Document (BRD), MRD, etc.
- Requirements
  - Goal: implementation-independent specification of what the software must do and any constraints on its development
  - Product: Software Requirements Specification (SRS)

Notes:

Phases and Products

- Architecture
  - Goal: decomposition of the problem into components that together satisfy the requirements and *quality goals* within the constraints
  - Products: Architectural specification of components, relations, interfaces(class structure, calls structure, task structure, etc.)
- Detail Design
  - Goal: internal design of components (e.g., objects) to identify appropriate algorithms and data structures supporting the interface
  - Products: design documentation, pseudo-code

Notes:

Phases and Products

- Implementation
  - Goal: realization of the design in a machine-executable language
  - Product: code, internal documentation
- Testing (not really a phase, continuous)
  - Goal: validation and verification of the implementation against requirements and design
  - Products: test plan, test cases
- Maintenance (also not a phase)
  - Goal: maintain deployed system
  - Products: bug fixes, patches, new versions

Notes:

Product Spec (ConOps or BRD)

- Why: Documents why this product is the right one to build for stakeholders (esp. business and customers)
- What: link what will be developed with why we are developing it
  - Rationale: “What are the overall objectives and rationale (Purpose, Business case, Mission) for creating/changing this system?”
  - Solution Requirements: “What characteristics should the system have and capabilities should it provide to address the objectives and rationale?
- Who:
  - Writes: ideally the customer, often a surrogate (marketing)
  - Uses: customers, management, marketing, tech. staff

Notes:

Requirements (SRS)

- Why: Documents what we are going to build before we build it
- What: Everything one needs to know to write an acceptable system – and no more
  - Every statement should be valid for all acceptable products
  - Any system that satisfies every statement should be an acceptable implementation
- Who:
  - Writes: technical staff (may include customers)
  - Uses: designers, testers, coders, managers, (sometimes customers if technically inclined)

Notes:

Architecture

- Why: Describes the system decomposition and how that decomposition satisfies quality goals (e.g., performance, security, safety, maintainability)
- What: Design structures that address key design goals in terms of their components, relations, and interfaces
  - E.g., calls structure, class structure, uses structure
  - Includes the rationale for design decisions
- Who:
  - Writes: software architect
  - Uses: implementers, testers, management (work breakdown)

Notes:

Detail Design

- Why: Describe the detailed code design decisions and their rationale (e.g., choice of algorithms and data structures)
- What: sufficient description to understand
  - How the code works without reading the code in detail
  - Rationale for key decisions
- Who:
  - Writes: programmer
  - Reads: programmer, reviewers, maintainers, testers

Notes:

Summary

- Documentation for project control
- Management Documents
  - Project plan
  - Test Plan
- Development documents
  - ConOps
  - SRS
  - Architecture
  - Detail Design
  - Test Suites

Notes:

Documentation Principles

Notes:

Documentation Problems

- Documentation regarded as a “necessary evil”
  - Written only because it’s required, as an afterthought
  - Do not expect it to be useful
  - A self-fulfilling prophecy
- Result of spending little effort
  - Poor organization (e.g., stream of consciousness) – cannot find needed information
  - Boring prose – difficult to read, maintain
  - Not user friendly – developer’s perspective
- Upshot: vicious cycle – difficult to use documents, don’t get used, not worth maintaining, go out of date, become useless

Notes:

Documentation Principles

- Write specifications for the document
  - Easiest way to ensure agreement on:
    - Intended audience(s)
    - Purpose(s) of the document
    - Document organization or design.
  - Keep design simple and reusable
    - Avoid starting another project in specification design
    - …but, do enough to capture consensus and resolve dissention
- Characterize and formulate questions to be answered in each section before starting to answer them
  - Avoids adding extraneous material
  - Avoids answering only the easy questions

Notes:

Key idea: The document will tend to wander from its design/purpose if there is no agreed upon basis for controlling form and content. What we are looking for is a relatively small up-front investment in deciding and documenting the requirements specification's purpose and content.

Documentation Principles (2)

- Design the document applying *Separation of Concerns*
- Principle of Separation of Concerns: distinct concerns (i.e., relatively independent issues) belong in separate parts of the document
- Goal is to divide the document into distinct and relatively independent sections
  - Readability: users can read and understand one section of the document without having to read or reference many others.
  - Modifiability: relative independence of the parts means that a changes tend to be confined to one part of the document
  - Reusability: parts that form a distinct whole (i.e. not dependent on other parts) are easier to reuse.

Notes:

Key Idea: Applying principle of separation of concerns yields a couple of key document properties.

Readability: avoid phenomenon that you can't understand anything until you understand everything.

Modifiability: same as modularization, encapsulation, design for ease of change.

Documentation Principles (3)

- Write once - read many times
  - Remember that the specification will be used many times but only written once (more or less)
  - Best ROI when effort is spent in writing to make it easy to read!
- Formal vs. informal notation
  - Informal (natural language): for introduction, motivation, gaining an overview, or review
  - Formal : for binding precise information, ease of reference, ease of checking correctness (e.g., mathematical functions, tables, formal notations)

Notes:

Key idea: it is the writers obligation to make the reader's job as easy as possible.

Example SRS Sections

Example applying separation of Concerns:
A) Computing resources (platform assumptions)
B) Inputs and Outputs
C) Time-independent Behavioral Requirements
D) Timing Constraints
E) Accuracy Constraints
F) Exceptions
G) Dictionaries
H) Required Subsets
I) Assumptions and Expected Changes
J) Sources and references (people, documents, etc.)
K) Indices

Notes:

A requirements specification where similar requirements are grouped together. Organization is pretty clear from the contents.

Example Requirements Spec Sections

- Inputs and Outputs
  - What inputs and outputs are available to the software?
  - What are their accuracy and timing characteristics?
  - What environmental values (abstract data items) do they correspond to?
  - How can they be retrieved by the software?
- Time Independent Output Functions
  - For each output data item: what relation is the output value required to have relative to the current state of the inputs and the aircraft operating conditions?
  - For each output data item: what is the range of acceptable values for every possible values of the related inputs?
  - What function determines the value in each state?

Notes:

For each section of an SCR-style specification, the slides give the questions that must be answered in that section. This is determined in advance then checked throughout the process to determine the quality and completeness of the specification.

Example Requirements Spec Sections

- Exceptions
  - What is the software required to do in response to unexpected input values?
  - Unexpected hardware states or failures?
- Required Subsets
  - What useful subsets of the software must be developed? (e.g., as increments)

Notes:

Example Requirements Spec Sections

- Assumptions and Expected Changes
  - What kinds of requirements are likely to change in future versions of the system?
  - What kinds of requirements are expected to be the same (not change) in future versions of the system?
- Sources
  - Annotated list of reference material and personnel answering: How do I find out more about the requirements or system context?
    - Where is detailed information about X specified?
    - Who represents the customer for capability Y?

Notes:

End

Notes: