COMP354

Software Engineering

Lecture 1

Overview of Software Engineering

Software Engineering

Software Engineering is a **discipline** for the **systematic production** and **maintenance** of **software** which is developed by a **team** within **time limits** and **cost estimates**.

Aim: Produce software that is useful for **People**

Software Engineering is concerned with

Products: external deliverables & internal products

- software products: final code, test drivers
- paper documents:
  (*external*) user manual, installation guide;
  (*internal* requirements document, architectural design, interface specification, internal module design, test plan

Processes: How is software created? How is quality evaluated and ensured?

Power Tools: editor (`vi, emacs`), debugger (`xxgdb`), configuration management (`rcs`), CASE tools (`sniff`), documentation (`latex, xfig`).

People: technical, managerial, and social skills.

Principles: Guiding lights offering permanence in a rapidly changing discipline.

Software Life Cycle

Feasibility study WHY?

- cost-benefit analysis
- Is it worthwhile doing the project?

Requirements analysis and specification WHAT?

- What should the software do?
- product: requirements, specification and analysis document

Design and specification HOW?

- How should the software do it?
- architectural design
  - overall structure and organization of modules
  - product: architectural design, interface specification
- detailed design
  - choice of data structures and algorithms for module internals
  - product: detailed internal module design

Coding and module testing REALIZE components!

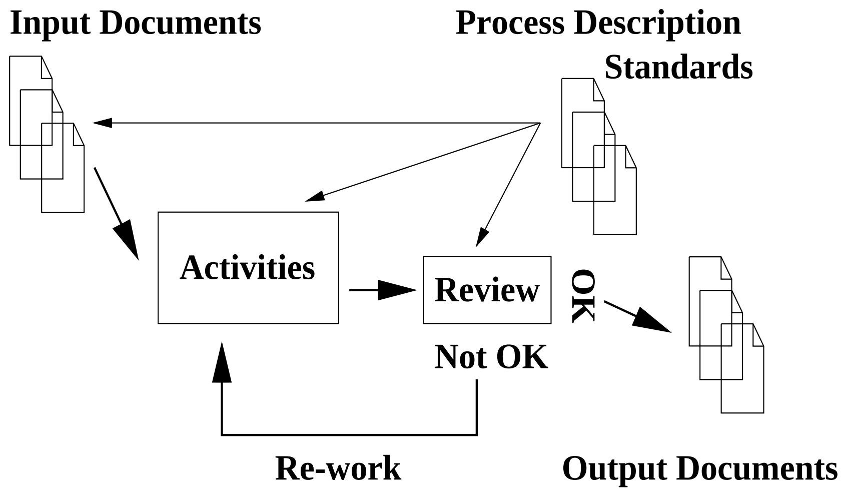
- code and test individual modules
- product: software, test result description

Integration and system testing REALIZE system!

- test whether several modules work together
- test system as a whole
- product: test result description

Delivery and maintenance EVOLVE!

A Lifecycle Phase



For each phase of the life-cycle you should note:

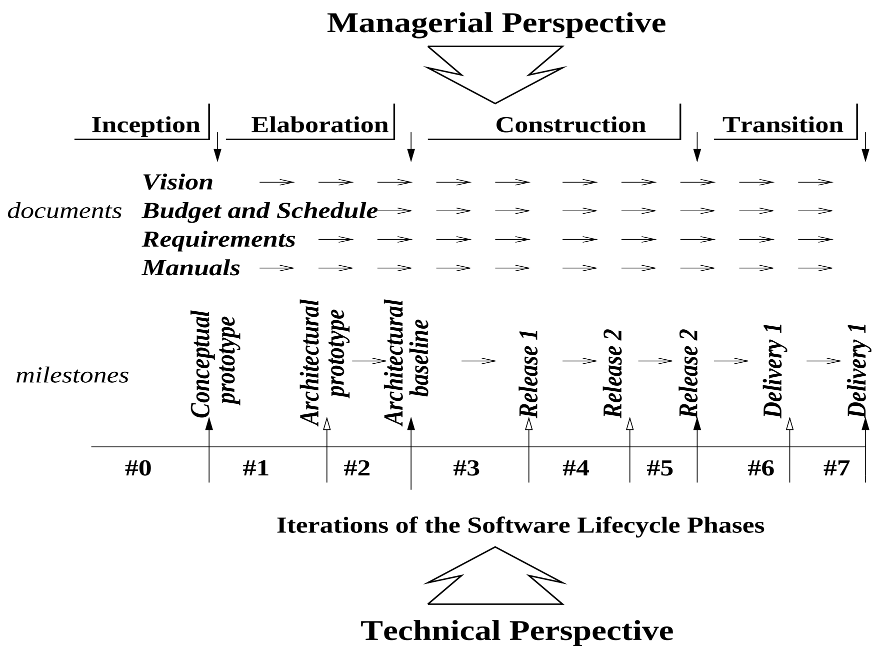
- PRODUCTS/DELIVERABLES
  - what is the aim of the phase
    that is, what does it aim to produce
- ACTIVITIES
  - what steps are taken in the process of producing the deliverables
- AUDIENCE for each deliverable
  - who will use each deliverable
  - how will they use each deliverable
  - what information will they extract from each deliverable
  - is it easy for them to find/understand this information

  NB there are often many different users of each deliverable
- HOW TO REVIEW QUALITY of each deliverable
  - what steps can you take to ensure the quality of each deliverable
    - ∗ during the production of the deliverable
    - ∗ after the production of the deliverable

Managerial vs Technical Perspectives

Managerial Perspective: financial, strategic, commercial, human aspects

The business cycle is
Inception: specify vision, business case, scope
Elaboration: plan activities and resources
Construction: build the product
Transition: deliver the product to the user's community — training, support, maintenance



Technical Perspective: quality, engineering, design

This is our main focus in this course!

Milestones are tangible indicators of progress — often *documents* or *prototypes* — that synchronize the managerial and technical perspectives

Variety of Software Projects

Dimensions along which software projects vary include

business context = developer-customer relationship

- contract work, where the software is for one customer only
- "shrink-wrap" commercial work, where there may be many (or none) potential customers
- internal corporate work, where the customers belong to the same organisation as the developer

size of the development effort: LOC, people, duration

novelty relative to the experience of the developers

- new language or paradigm, eg OO
- new tools, eg CASE tool
- new application domain, eg MIS to avionics
- level of experience working as a team
- level of experience of managers

type of application and application domain: MIS, command and control, embedded real-time, medical, internet — all have different issues and problem areas

These will affect

- the expected productivity during development
- the amount of emphasis given to the different phases of the lifecycle
- the choice of development process
- the risk of unsuccessful completion of the project
- the expected monetary benefits

Skills of a Software Engineer

- programming
  - data structures and algorithms
  - programming languages
  - tools — compilers, editors, debuggers
- communication
  - spoken, written, presentations
  - as a member of a team
  - with external people
    — customer, user, suppliers
- modelling (abstraction) skills
- design
  - be familiar with several approaches
  - be flexible and open to understand different application domains
  - be able to shift between several levels of abstraction
    - application domain jargon and model
    - requirements and specification declarative model
    - architectural design high-level operational model
    - detailed coding
- organization and management
  - schedule and chair team meetings

Team Dynamics

Stages of Team Formation

Forming: establish the team purpose, membership, skills and roles
Storming: develop a strategy or mode of working together
Norming: form a coherent team that collects and processes information
Performing: produce

Conflict Resolution

conflict is healthy — leads to better solutions
conflict resolution selects the **best** from each alternative

Strategies for conflict resolution are

- compromise, often avoids the real issues
- forcing, one person insists of getting their way
- avoidance, hope it will go away
- confrontation — is the *most effective* — pinpoint the real issue of disagreement and discuss merits of the alternatives openly

Meetings

Elements of an Agenda

1. Title
2. Time and location
3. Theme and definition
4. Attendees
5. Topics
   (a) title
   (b) description
   (c) goal

Lay the groundwork well before the meeting.
Define topics clearly.

Four key decisions to be made

1. Who is responsible for solving the problem?
2. How can the rest of us help?
3. What course of action should be taken?
4. What is the deadline?

Outline of Minutes

- date and time
- attendees
- agenda topics discussed
- alternatives presented
- solutions agreed upon
- assignments made and accepted (*be specific*)
- deadlines (*be specific*)
- follow-up actions (including tabled items)

Michael C. Thomsett, The Little Black Book of Business Meetings, American Management Assoc., 1989.

COMP354

Software Engineering

Lecture 2

Tying the Software LifeCycle Together

Tying it all together

Vision and Alignment

getting every phase focussed on same priorities

Traceability

navigating from document to document

Visibility

visibility of *process*: concrete description of the process

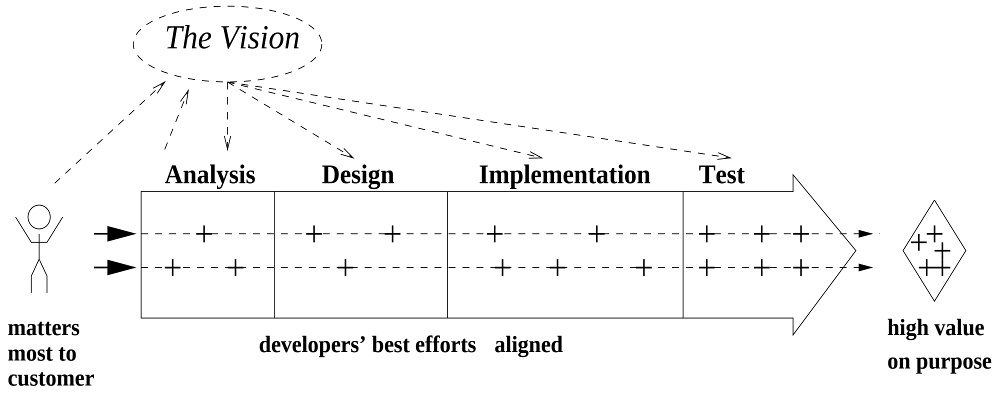
visibility of *project*: concrete evidence of the status of the project

Quality Control

validation: “building the right product”

verification: “building the product right”

Vision and Alignment

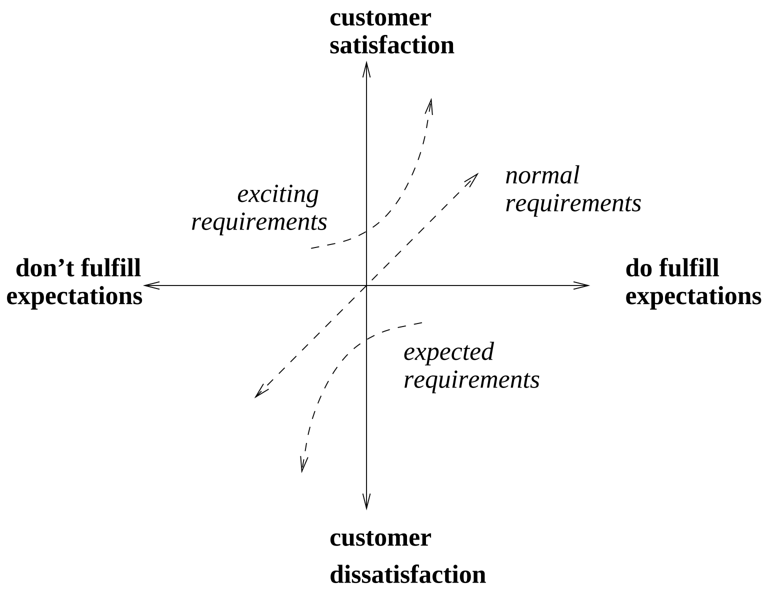


The *vision* captures

- the priorities of the customer
- the special reasons for doing the project
- the enthusiasm of the visionary

It helps everyone focus on the **important qualities**, thus *aligning* the efforts of each phase.

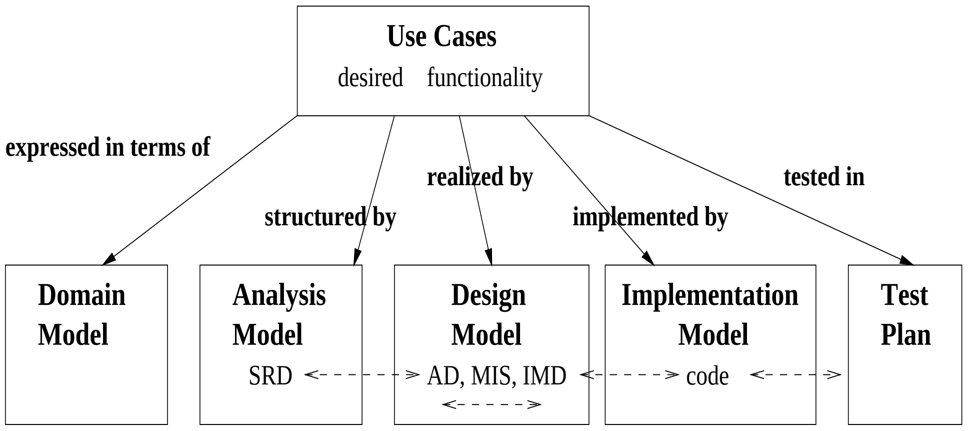
May lead to *great exciting software*, not merely good correct software!



Traceability

Traceability is the ability to find corresponding or related information in other documents or software.

- If users report a defect in the system,how do I find the relevant piece of code, design, requirements, etc that relate to that defect?
- If an upgrade calls for a change in `DialogBox` module of design, how do I find all tests that also need to be changed?
- If a coder believes a module's functionality is incorrectly designed, how does she find the corresponding requirements in SRD to confirm her suspicions?



Often *SRD* or *use cases* are used as central reference point for traceability.

Things that help

- each requirement (etc) should have a unique label
- version management of all products
- table of contents
- indexes
- cross references using unique labels
- links in hypertext documentation systems

Visibility

Visibility means there is tangible evidence of a process or product.

visibility of *process*: concrete description of the process

Not only must you have a process for software development, but you must be *seen* to have a process!

visibility of *project*: concrete evidence of the status of the project

Not only must you be making progress, but you must be *seen* to be making progress!

Milestones provide visibility

Visibility provides re-assurance for managers and positive feedback for developers.

Quality Control

validation: Does the product satisfy the users' needs?
"building the right product"

verification: Does a product, say the code, conform to another product, say the SRD?
"building the product right"

Quality control aims to evaluate and ensure the quality of all products.

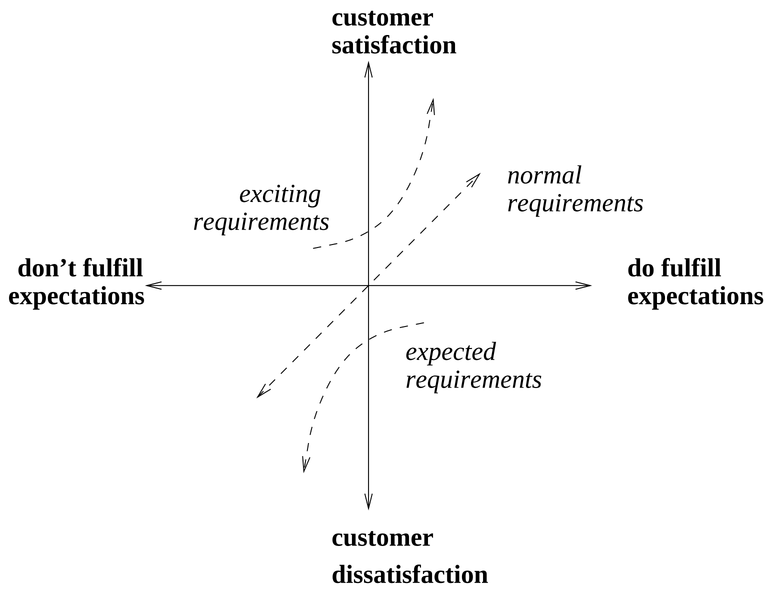
It is cheapest to detect and correct defects as *early as possible* in the lifecycle.

positive quality — a system that will excite the user, satisfy their needs, and make their jobs much easier

neutral quality — a correct system, "zero defects"
(necessary, but not exciting)

negative quality — a system with defects

Aim for *positive* quality!


customer satisfaction
exciting requirements
normal requirements
don't fulfill expectations
do fulfill expectations
expected requirements
customer dissatisfaction