The Software Process and its Problems

- The software process is the way we produce software. It incorporates the software life-cycle model, the tools used, and the individuals building the software.

- Recall transparencies 12 – 16

Different organizations have different software processes

When in case of prestigious projects large firms do not carry through one of the approaches advocated in this lecture course, this tends to reach the news headlines.

Just in Case You Wanted to Know

Why does the software process vary so drastically from organization to organization? A major reason is lack of software engineering skills. All too many software professionals simply do not keep up to date. They continue to develop software the Olde Fashioned Way, because they do not know any other way.

Another reason for differences in the software process is that many software managers are excellent managers, but know precious little about software development or maintenance. Their lack of technical knowledge can result in the project slipping so badly behind schedule that there is no point in continuing. This is frequently the reason that many software projects are never completed.

Another reason for differences among processes is management outlook. For example, one organization may decide that it is better to deliver a product on time, even if it is not adequately tested. Given the identical circumstances, a different organization might conclude that the risk of delivering that product without comprehensive testing would be far greater than taking the time to test the product thoroughly.

The bottom line is that software is developed by people, and the process within a given organization depends first and foremost on the individuals working in that organization. If those individuals are ethical, hard-working, intelligent, sensible, and up-to-date, then the chances are good that the software process within that organization will be satisfactory. Unfortunately, the converse is equally true.

1. The Ariane V disaster.
   Loss: $5 * 10^9$ DM at least. A national enquiry started into its causes; a sign of health of the French industry!

2. The, at the time new, DENVER AIRPORT passenger luggage transfer system disaster. Took one extra year to complete.
   Loss: 1.1 Million \$ per day in rent!

3. IBM's planned new passenger aircraft security positioning and routing management system for the whole of the U.S. civil airspace – IT NEVER FUNCTIONED.
   Loss: $2.5 * 10^9$ \$.

4. A well-known south-german electronics firm (by which several of you will be hired in future).

   (a) It took over the locomotive branch of KRUPP-MAK at Kiel and wrecked it. A separate

talk is planned for next week on this disaster. Initial losses: $\sim$ 100 million DM.

(b) It tried to build a system for the electronic surveillance and management of the entire railway network around the Hamburg-Altona station. At "delivery" train traffic with Kiel from Hamburg broke down entirely. The Hamburg-Altona station never regained its old status.

Other organizations follow the lines taught during this course.

Consider, e.g,, the issue of documentation:

- They perform careful design activities, checking and rechecking designs before coding commences, and give detailed description of each module to the programmers.

- Test cases are preplanned, logged, and test data carefully filed.

- Once the product enters the maintenance phase, any suggested change must be proposed in writing, with detailed reasons for the change.

- The change can be made only with written authorization, and the modification is not integrated into the product until the documentation is updated and approved.

Phases of the Software Process

- Regardless of the exact procedure, the software process broadly follows the phases outlined in figure 1.1. on transparency 13: requirements, specification, planning, design, implementation, maintenance, and retirement.

- Some of these are known by other names:
  - the requirements and specification phases are also called systems analysis.
  - The maintenance phase is also called operations mode.
  - The design phase is broken down into: architectural design (the structure of the software product) and detailed design (what the modules, in terms of which the software is structured, do).

The Need for Testing and Documentation

- Recall transparencies 17 and 18.

- In the preceding lines there is NO SEPARATE TESTING PHASE.

- Testing is not a separate phase, but takes place all the way through software production, i.e. during each of the phases:

  – during a phase itself
  – at the end of each phase (verification)
  – before the product is handed out to the client (validation)

  Similarly, there is NO SEPARATE DOCUMENTATION PHASE.

  – The documentation for each phase of the software development process must be completed by the

team responsible for their phase – BEFORE THE NEXT PHASE BEGINS!

– The documentation must always be UPDATED to reflect the CURRENT version of the product.

Why?

1. Postponed documentation is almost never completed.
2. Individuals responsible for a phase tend to be transferred or leave the firm.
3. During development, the product is constantly changing.E.g., its design is normally modified during the implementation phase (due to new info). UNLESS its design has been fully documented, modifying the design is EXTREMELY DIFFICULT.
4. It will be extremely difficult for the original designers to document their design AFTER modification if no original documentation exists.

Software Life-Cycle Models

- The separate phases of the software life cycle model often does not progress in the order suggested in fig 1.1 on transparency 13.

JUST IN CASE YOU WANTED TO KNOW

In practice, software development does not always progress as smoothly as suggested here. That is to say, the various phases of the life cycle can overlap. Instead of a clean division into eight separate phases, what frequently happens is: "Specify some, design some, code some, then specify some more, design some more, and so on."

Such an incremental process (Section 3.4) can be most productive, provided that the process is strictly followed. What happens all too often, however, is that some pieces are coded before they have been properly designed, and some parts are designed before they have been properly specified.

For this reason, I advocate phasewise software development, unless your organization is well managed and can handle an incremental process. But many organizations cannot work incrementally, and for them I strongly recommend a strict phasewise approach.

- The order and the extent in which these separate phases are carried out differ according to the various software life cycle models adopted.

  These are:

  – The build-and-fix model. This naive and stupid model is used by any programmer in his youth;

when adopted by a firm it ALWAYS leads to DISASTER.

- The waterfall model. Characterized by a strict separation of phases until faults are discovered. Then backtracking takes place to previous phases until the earliest occurrence of those faults are localized, correction has taken place and the process is continued iteratively.
- The rapid prototyping model. Discussed below. It consists of building a working model displaying the same functional (I/O) behavior as a subset of the software product.
- The incremental model. Characterized above in the "Just in Case... Box" by "Specify some, design some, code some, then specify some more, design some more, and so on."
- The spiral model. Characterized by also planning minimal (financial) risk via the extended use of prototypes and other means.

Client, Developer, User

- The client is the individual organization who wants a product to be developed.
- The developers are the members of the organization responsible for producing the product.
- The user is the person or persons on whose behalf the client has commissioned the product and who will utilize the software.
- If client and developers belong to the same firm this is called internal software development.
- With contract software, client and developers belong to totally different organizations.

Requirements Phase – Characterization And Difficulties

Characterization:

- Software development is expensive.
- At any stage of the process, if the client stops believing the system is cost effective, development will immediately terminate. Below we assume that the client feels this cost is justified.
- At the initial meetings – called concept exploration – the client outlines the product as he or she conceptualizes it.
- However, from the developers' viewpoint, the client's description may be vague, unreasonable, contradictory or impossible to achieve.

THE TASK OF THE DEVELOPERS DURING THIS PHASE IS TO FIND OUT EXACTLY WHAT THE CLIENT REALLY NEEDS and which constraints exist.

- E.g., the product should cost less than $ 370.000 and should be completed within 14 months.

- Other constraints are reliability (the Siemens locomotives ordered by Norway should be operational 97% of the time), and size (should fit on a personal computer).

- In subsequent meetings between members of the development team and the client team, the functionality of the proposed product is successively refined and analyzed for technical feasibility and financial justification.

Difficulties During the Requirements Phase

What happens when the requirement phase is carried out improperly?

- After delivery of the product the client calls the developer and says: "*I know that this is what I asked for, but it isn't what I really needed.*"
- How can this happen, and why?
- The client may not truly know what's going on in his or her own organization (Does Kohl know what really happens in Germany? Did Mercedes-Benz' director know his test-drivers fooled him when testing model A?).
- E.g. is does not make sense to ask for a faster operating system if the cause of bad response time is a badly designed database.

The Need For Rapid Prototyping

- The MAJOR REASON that the client so frequently asks for a wrong product is that
  SOFTWARE IS SO HORRIBLY COMPLEX !

- If it is difficult for a developer to have a model of software and its functionality, how much more difficult is it for the client who is a computer illiterate?

- One way of dealing with this problem is: rapid prototyping.

- A rapid prototype is a piece of software hurriedly put together that incorporates much of the functionality of the target product, yet omits those aspects generally invisible to the client (s.a. file updating, error handling).

- Client and users then experiment with the

prototypes to find out whether it meets his or her needs, indeed.

- The rapid prototype can then be changed until client and users are satisfied that it encapsulates the functionality they desire.

- You will appreciate the difficulties encountered during the requirement phase once you have tried to find out what your client really needs in case of your team project.

Requirements Phase Testing

- Recall transparencies 19 and 20.

- During the requirement phase the SQA (Software Quality Assurance) team should assure that the product satisfies THE ACTUAL NEEDS OF THE CLIENT (and not what he or she says is needed – see above).

- Thus, the SQA team must verify with the client that the final version of the rapid prototype is totally satisfactory.

- Forces beyond control of the development team may, however, necessitate changes in requirements WHILE THE PRODUCT IS BEING DEVELOPED:

  – Further development then stops until the required modifications to the partially finished product are made.

- This is called the MOVING TARGET PROBLEM.
- However, the major cause of this problem is a client who keeps on changing his or her mind. NOTHING CAN BE DONE ABOUT THIS if the client has sufficient CLOUT!

Specification Phase – Characteristics And Difficulties

- Once the client agrees that the developers understand the requirements, the specification document is drawn up by the specification team.

- The specification document (or specification(s)) explicitly describes the functionality of the product, i.e. precisely what the product is supposed to do, and what "constraints" the product must satisfy.

- It includes all inputs to the product and the outputs required.

- E.g., in case of a payroll product, inputs must include pay scales of each employee, data from a time clock, and information from personnel files so that taxes can be computed correctly.
  Outputs will be paychecks and reports s.a. Social Security deductions.

- The specification document of the product constitutes a CONTRACT, and may form the basis of a lawsuit, for:

- The software developers are deemed to have completed the contract when they deliver a product that satisfies the acceptance criteria of the specification document.

- Thus, the specification document should NOT include imprecise terms like "suitable", "convenient", "ample" and "optimal".

- For, otherwise, it cannot support a lawsuit.

Difficulties During The Specification Phase

- Specifications may be ambiguous – contain sentences that may have more than one possible valid interpretation.

- E.g., consider the specification: *"A part record and a plant record are read from the database. If IT contains the letter "A" directly followed by the letter "Q", then compute the cost of transporting that part to that plant."*
  To what does "IT" in the preceding sentence refer to?

- Specifications may be incomplete – relevant facts or requirements are omitted.

- E.g. the specification may not state what actions to undertake when the input data contains errors.

- Specifications may be contradictory – i.e. inconsistent.

- E.g., a staff document contains the sentence: "*employees working more than 50 hours/week get extra pay/hour.*" However, at another place in that document, it is stated that employees working less than 55 hours/week get no extra pay/hour.

- SOFTWARE DEVELOPMENT CANNOT PROCEED UNTIL SUCH PROBLEMS IN THE SPECIFICATION HAVE BEEN CORRECTED.

- You will appreciate these difficulties once you have tried to remove the ambiguities, sources of incompleteness and contradictions from your team project's description.

Specification Phase Testing

- Recall that 65% of all faults in a delivered product are due to an incorrect or misinterpreted specification or requirement document.

- Therefore, the SQA group must carefully check the specification, looking for contradiction, ambiguities and signs of incompleteness.

- It must ensure that the specification is feasible (e.g. fits the clients disc capacity).

- The specification should be testable, i.e., every statement in the specification document must be traced to a statement in the requirements document. This is called tracability. And also the converse should hold.

How is this Goal Achieved?

- Checking the specification document is done by means of a REVIEW during which specification team and the client are present.

- The reviewers go through the specification document ensuring that that there are no misunderstandings about the document.

- There are two types of review:
  - Walkthroughs, and
  - Inspections.

  These vary from each other in degree of depth.

Walkthroughs and Inspections

- Members of the walkthrough team should be experienced senior technical staff because they tend to find the important faults.

- Each reviewer should study the material and develop two lists:
  - Of the items the reviewer does not understand.
  - Of the items the reviewer believes are incorrect.

- An inspection goes far beyond a review and has 5 formal steps:
  1. An overview phase: an overview of the document is given.
  2. During the preparation phase, participants try to understand the document in detail.
  3. During inspection, it is ensured every item is covered and every branch taken at least once. Fault finding commences. A written report is produced of all findings.

4. During the rework phase, the individuals responsible for the document resolve all faults and problems raised in the written report.
5. In the follow-up phase, the moderator ensures every single issue raised has been satisfactorily resolved and no new faults have been introduced.

- IF MORE THAN 5% OF THE MATERIAL HAS BEEN REWORKED, THE TEAM RECONVENES FOR A 100% INSPECTION (as a further guarantee that no new failures have been introduced).

Planning Phase – Characteristics

1. No client authorizes a software project without knowing HOW LONG THE PROJECT WILL TAKE and HOW MUCH IT WILL COST.
   - This is equally important from the developer's viewpoint because:
     - If the developers underestimate the cost of a product, the client pays the agreed fee and the developers' firm loses money.
     - If the developers overestimate this cost, the client may turn the project down or go to another software firm.
   - Similar issues hold for the duration estimation of a project.

2. The developers need to assign appropriate personnel to the various development phases, i.e the developers MUST PLAN AHEAD. A software projects management plan – SPMP – is drawn up, reflecting:

- deliverables (what the client is going to get).
- milestones (when the client gets them).
- budget (how much it will cost).

3. The SPMP includes:

   - Life-cycle model used
   - Organizational structure of development team
   - Who is responsible for what
   - Managerial objectives or priorities (delivering on time and/or correctness of the product)
   - Detailed schedules, budget resource allocation)

4. Planning phase testing is done by reviewers similar to those of the specification document.

Design Phase – Characteristics

- The specifications spell out WHAT the product is to do. The design phase determines HOW the product does this starting from the specification document.

  1. The internal structure of the software product is determined
  2. Algorithms are selected
  3. Data structures are chosen
  4. Inputs to and outputs from the product are determined

- Now the internal data flow can be determined.

- More precisely:

  – The product is decomposed into modules (independent pieces of code with well-defined interfaces to the product). An object is a preferred kind of module.

  - For each module, its designer specifies what is has to do and how it does this.
  - The interface of each function within the module (i.e. arguments passed to and supplied by the function) are specified in detail.
    * E.g., a function measures the water-level in a reactor and causes an alarm to sound if the level is too low.

- During decomposition, the design team keeps a careful record of the design decisions made, because of the following reasons:

  1. During design, there will be times when a dead end is reached. Making a record of the design decisions helps to backtrack upon them and redesign certain pieces.
  2. Maintenance. Designers have to compromise, putting together a design that can be extended in many reasonable ways without the need for redesign.

- In a product that undergoes major enhancement (recall 50% of a good product may be changed

within 5 years), time will come when its design SIMPLY CANNOT HANDLE FURTHER CHANGES. Then it is time to REDESIGN the product.
If the redesign team has a record of the reasons for all original design decisions, its job will be easier.

- Major output of the design consists of two parts:
  - architectural design – a description of the product in terms of its modules,
  - detailed design – a description of each module; these descriptions are subsequently given to the programmers for implementation.

Design Phase Testing

- Every part of a design should be linked to a statement in the specification document.
- Moreover, every statement in the specification document should be reflected in the design.
- Design reviews are similar to the reviews specifications undergo.

Implementation Phase

- During the implementation phase, the various modules of the design are coded into the programming language specified in the specification document.

- The major source of documentation is the source code itself, suitably documented.

- The programmer should also provide additional documentation to assist in the maintenance phase in regression testing. This documentation consists of all the test cases against which the code was tested, the expected results and the actual outputs produced.

- Modules should be tested by the programmer him/herself while they are being implemented – called desk checking – and after they have been implemented they are run against test cases. A

code review is a powerful method for detecting programming faults. Here the programmers explains his code to the SQA team, the review being similar to those discussed in case of the specification and design phases.

Integration Phase – Characteristics

- The next stage is to combine the implemented modules and to determine whether the product as a whole functions correctly, i.e. satisfies the needs laid down in the specification document.

- Later on it will be described why implementation and integration are no separate phases but should be performed in parallel. Basically this is to detect and eliminate faults as soon as possible.

Integration Phase Testing

- First integration testing is being performed: this is the name for checking that the modules combine correctly to achieve a product that satisfies its specifications. In particular the module interfaces are checked, i.e. whether number, order and types of formal parameters match the number, order and types of the actual parameters. (In a fully typed language this is done by the compiler and linker.)

- The product testing is carried out: This consists of 3 phases:

  - The functionality of the product as a whole is tested against the specification document. E.g., whether the constraints listed therein, s.a. maximal response time, are satisfied.
  - Then the robustness of the product is tested, by submitting intentionally erroneous input data to determine whether the product will crash

or whether its error-handling capabilities are adequate.
  - Finally the documentation and source code are tested for completeness and internal consistency.

- Last but not least, the software is delivered for acceptance testing to the client, who tests the software on the actual hardware using actual data, as opposed to test data.
  - There is a significant difference between test cases, which by their very nature are artificial, and actual data.
  - Therefore, a software product cannot be considered to satisfy its specifications UNTIL THE PRODUCT HAS PASSED ITS ACCEPTANCE TESTS!

- Sometimes, earlier versions of the complete product are shipped to the client for testing on site – this is called alpha testing.
  - The corrected alpha version is called the beta version and is usually close to the final product.

Maintenance Phase – Characteristics

- Once the product has been accepted by the client, any changes constitute maintenance.

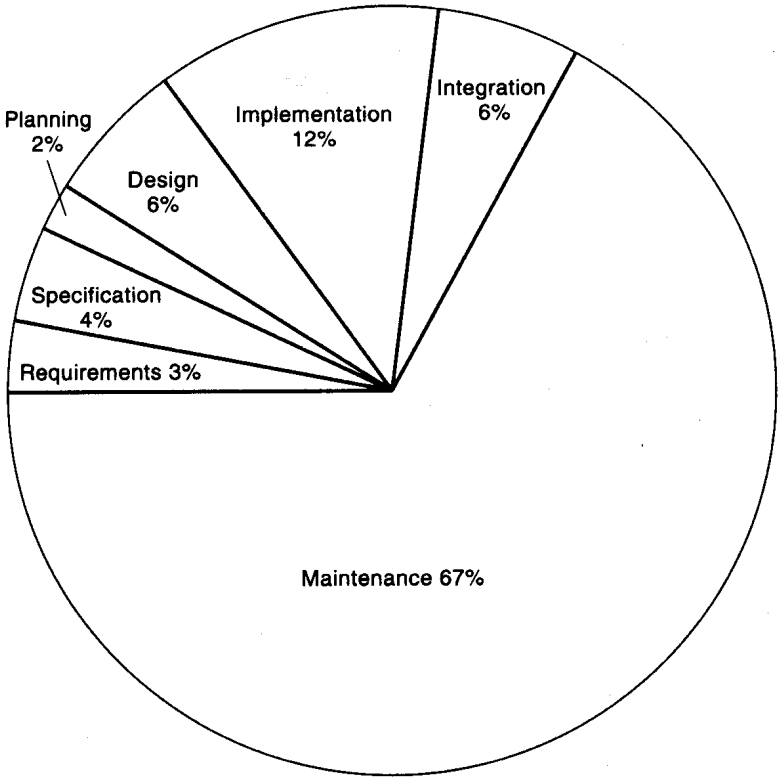

Figure 1.2 Approximate relative costs of phases of software production process.

- Because maintenance measures 67% of the entire software production process, THE ENTIRE SOFTWARE DEVELOPMENT EFFORT MUST BE CARRIED OUT IN SUCH A WAY AS TO MINIMIZE THE IMPACT OF THE INEVITABLE FUTURE MAINTENANCE.
  - E.g., the design should, as much as possible, take future enhancements into account and coding should be performed from the same point of view of future change.

Maintenance Phase – Testing

- There are two aspects to the testing of changes in operations mode.
  - The first is that the required changes are correctly implemented.
  - The second is that no other inadvertent changes in the product result because of negative interaction with the unchanged parts of the code. Testing this is called regression testing.

- The need for regression testing explains why it is necessary that all previous test cases be retained, together with the results of running them.