III. Software Lifecycles

Software processes and lifecycles
Relative costs of lifecycle phases
Examples of lifecycles and processes
Process maturity scale
Information system development lifecycle
Lifecycle phases

The Software System Lifecycle

A software process is a partially ordered collection of actions, carried out by one or more software engineers, software users, or other software systems in order to accomplish a (software engineering) task.

The software system lifecycle is a software process by which a software system is developed, tested, installed and maintained throughout its useful history.

The concept of software lifecycle is a useful project management tool. A lifecycle consists of phases, each of which is a software process.

Think of lifecycles as coarse-grain software processes. There is a lot of work on fine-grain software processes, such as fixing a bug, extending a module, testing a module, etc.

We focus here on information system development lifecycles.

The Software Lifecycle

• For large software systems, involving >10K lines of code (LOC), the breakdown of costs between different phases is as follows:
  - Requirements Analysis 5%
  - Design 10%
  - Programming-in-the-small 15%
  - Integration 10%
  - Maintenance and Evolution 60%
• The breakdown of costs per phase for small software systems (<5K LOC) has as follows:
  - Specification 10%
  - Decomposition 20%
  - Coding 20%
  - Optimization 15%
  - Testing 25%
  - Validation 10%

Systems analysis and design more important than coding!

What is Described by a Lifecycle?

• The lifecycle describes the temporal, causal and I/O relationships between different lifecycle phases
• The lifecycle concept includes the concept of feedback (returning to a previous phase) as well as moving forward to the next phase
• In the past, the lifecycle concept was applied to the management of complex systems that had some sort of physical hardware as their end product, e.g., missiles, communication networks, spacecraft, etc.
• However, for hardware systems there is a tangible end product that can be measured and observed...

It is not as easy to measure and observe the results of information systems analysis and design.

Lifecycle Models

• History of lifecycle models
  - Stage-wise (Benington, 1956)
  - Waterfall (Royce, 1970)
  - Transformational, automatic (Balzer, 1983)
  - Evolutionary (Basili & Turner, 1975)
  - Transformational, specification to implementation (Lehman, Stenning and Turski, 1984)
  - Spiral (Boehm, 1986)
• Benefits of lifecycle models
  - Process awareness and understanding
  - Order of global activities
  - Improvement in product quality
  - Reduction of software costs
• Deficiencies of lifecycle models
  - Too coarse-grained -- they hide important process detail

The Waterfall Lifecycle Model
Waterfall Life Cycle Deliverables

<table>
<thead>
<tr>
<th>Phase</th>
<th>Output deliverables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification</td>
<td>Requirements spec</td>
</tr>
<tr>
<td>Design</td>
<td>Functionality spec</td>
</tr>
<tr>
<td>Integration</td>
<td>Interface spec</td>
</tr>
<tr>
<td>Construction</td>
<td>System spec</td>
</tr>
<tr>
<td>Testing</td>
<td>Acceptance test</td>
</tr>
<tr>
<td>Maintenance</td>
<td>Change requested reg</td>
</tr>
</tbody>
</table>

Criticisms of the Waterfall Life Cycle Model

- **Advantages**
  - The tasks of a step may be assigned to a specialized team;
  - The progress of the project can be evaluated at the end of each phase and an assessment made as to whether the project should proceed;

- **Criticisms**
  - Inflexible partitioning of the project into distinct steps -- real projects rarely follow it!
  - Iterations are inevitable;
  - It may take too long;
  - Difficult to respond to changing customer requirements.

Generally, this lifecycle model is appropriate when the requirements for a given system are well-understood.

The Waterfall Life Cycle with Iteration

Prototyping

- Built something quickly to explore some aspect of the systems requirements
- The prototype is not intended as the final working system; among other things, it may be incomplete, less resilient (ex. poor performance) than a production system.
- In building a prototype, the objective is to investigate user requirements, in particular:
  - What data should be presented and what data should be captured;
  - To investigate suitable forms of interfaces;
- Also to determine whether a particular implementation platform is appropriate, as well as the efficacy of a language, DBMS or communication infrastructure.

Evaluation of Prototyping

- **Advantages**
  - Early demonstrations of system functionality help identify any misunderstandings between developer and client;
  - Helps identify missing client requirements;
  - Problems with user interfaces can be identified;
  - Early testing of feasibility and usefulness of the system (partially)
- **Problems**
  - The client may not understand the extra effort needed to produce a working production system;
  - May divert attention from functional to solely interface issues;
  - Requires significant user involvement;
  - Managing the prototyping life cycle is not easy;
- **Applicability**
  - For small or medium-size interactive systems
  - For parts of large systems (e.g. user interfaces)
  - For short-lifetime systems.
**Software Processes: Fixing a Bug**

**Step 1: Problem identification**
- During testing, a problem is identified.
  - A problem report is created, including problem identification, responsible personnel etc.
  - Responsible personnel is notified

**Step 2: Problem analysis**
- Perform problem description evaluation, evaluation of software component etc.
  - Propose solutions and describe technical and operational implications

**Step 3: Cost analysis**
- Project manager decides whether to use cost analysis routine
  - If so, perform cost analysis to determine impact in work-months

**Step 4: Schedule analysis...**

**Step 5: Perform change process...**

**Step 6: Close problem report...**

---

**Software Process Programming**

A Testing process

Function AllFunctionsOK(executable, tests);
declare executable executableCode, tests testsSet, 
result derivedResult;
/* executableCode etc are types, undefined here */
All-fn-OK := true;
For case := 1 to #tests do
  derive(executable, tests[case].input, result)
  if ~resultOK(result, testcase[case].output)
    then All-fn-OK := false; exit;
end loop;
end All-Fn-Perf-OK

This only works for highly structured or automated processes

[Osterweil87]
Deciding What to Do:
- Confirm that a problem exists
- Carry out a study to determine if a system can be developed to solve the problem (2 days - 4 weeks)
- A feasibility study looks at the problem at a high level (only takes into account few details)
- The study provides cost and savings estimates for the proposed solution.
- The feasibility study is reviewed by the customer (usually through a manager) and if the review is positive, then a more detailed requirements study is undertaken.

Who Are the Players (“Stakeholders”)?
- Management -- for initiation, approval, control, possibly as users
- End-users (persons who actually use the system on a day-to-day basis) -- they provide input during requirements definition and testing, participate in committees and final system evaluation
- Developers (analysts and programmers): Analysts -- serve as project leaders, perform information analysis, create system requirements and design
- Programmers -- program, test, document, maintain
- System support group -- they are responsible for system maintenance
- Database administrator -- responsible for design and control of one or more databases
- Program librarian -- keeps track of all program files, documentation
- Steering committee -- oversees project to ensure that objectives have been met

Who Are the Players (“Stakeholders”)?
- Management -- for initiation, approval, control, possibly as users
- End-users (persons who actually use the system on a day-to-day basis) -- they provide input during requirements definition and testing, participate in committees and final system evaluation
- Developers (analysts and programmers): Analysts -- serve as project leaders, perform information analysis, create system requirements and design
- Programmers -- program, test, document, maintain
- System support group -- they are responsible for system maintenance
- Database administrator -- responsible for design and control of one or more databases
- Program librarian -- keeps track of all program files, documentation
- Steering committee -- oversees project to ensure that objectives have been met

Who Are the Players (“Stakeholders”)?
- Management -- for initiation, approval, control, possibly as users
- End-users (persons who actually use the system on a day-to-day basis) -- they provide input during requirements definition and testing, participate in committees and final system evaluation
- Developers (analysts and programmers): Analysts -- serve as project leaders, perform information analysis, create system requirements and design
- Programmers -- program, test, document, maintain
- System support group -- they are responsible for system maintenance
- Database administrator -- responsible for design and control of one or more databases
- Program librarian -- keeps track of all program files, documentation
- Steering committee -- oversees project to ensure that objectives have been met

Who Are the Players (“Stakeholders”)?
- Management -- for initiation, approval, control, possibly as users
- End-users (persons who actually use the system on a day-to-day basis) -- they provide input during requirements definition and testing, participate in committees and final system evaluation
- Developers (analysts and programmers): Analysts -- serve as project leaders, perform information analysis, create system requirements and design
- Programmers -- program, test, document, maintain
- System support group -- they are responsible for system maintenance
- Database administrator -- responsible for design and control of one or more databases
- Program librarian -- keeps track of all program files, documentation
- Steering committee -- oversees project to ensure that objectives have been met

Who Are the Players (“Stakeholders”)?
- Management -- for initiation, approval, control, possibly as users
- End-users (persons who actually use the system on a day-to-day basis) -- they provide input during requirements definition and testing, participate in committees and final system evaluation
- Developers (analysts and programmers): Analysts -- serve as project leaders, perform information analysis, create system requirements and design
- Programmers -- program, test, document, maintain
- System support group -- they are responsible for system maintenance
- Database administrator -- responsible for design and control of one or more databases
- Program librarian -- keeps track of all program files, documentation
- Steering committee -- oversees project to ensure that objectives have been met

Who Are the Players (“Stakeholders”)?
- Management -- for initiation, approval, control, possibly as users
- End-users (persons who actually use the system on a day-to-day basis) -- they provide input during requirements definition and testing, participate in committees and final system evaluation
- Developers (analysts and programmers): Analysts -- serve as project leaders, perform information analysis, create system requirements and design
- Programmers -- program, test, document, maintain
- System support group -- they are responsible for system maintenance
- Database administrator -- responsible for design and control of one or more databases
- Program librarian -- keeps track of all program files, documentation
- Steering committee -- oversees project to ensure that objectives have been met
Additional Readings

