Information Systems Concepts

Avoiding the Problems

Roman Kontchakov

Birkbeck, University of London

Based on Chapter 3 of Bennett, McRobb and Farmer:
Outline

- Project Lifecycles
  - Section 3.3.1 (pp. 71 – 73)
  - Section 3.3.3 (pp. 76 – 77)
Development as Conversation

- Sales

Customer: I have a need.

Developer: I can help. What do you need?
Development as Conversation

- **Requirements**

  What I need is …

  I understand.

  Customer  Developer

---  Teach Yourself Extreme Programming In 24 Hours.
Development as Conversation

- Design

If we built this would it meet your need?

Yes. Start working.

Customer  Developer

--- Teach Yourself Extreme Programming In 24 Hours.
Development as Conversation

- Build

Customer: What are you doing?

Developer: Programming. Please wait.
Let me check and see. Perfect!

I’m finished. Is this what you wanted?

Customer

Developer

--- Teach Yourself Extreme Programming In 24 Hours.
Development as Conversation

- Launch

Let’s launch it.

Call me if there are any problems.

--- Teach Yourself Extreme Programming In 24 Hours.
Traditional Lifecycle (TLC)

The TLC model is also called the **waterfall** lifecycle model because of the difficulty of returning to an earlier phase.
1. System Engineering:
   - “What is our context?”
   - “How will human, hardware and software get involved in?”

2. Requirements Analysis:
   - “What are we trying to achieve?”
   - “What entities are we dealing with?”
   - “How can we be sure we have the right ones?”
Traditional Lifecycle – Key Questions

3. Design:

- “How are we going to solve the problem?”
- “What hardware and software will we need in the finished system?”
- “How are we going to implement the solution?”
- “What will the source code and supporting files look like?”
- “What rules govern the interfaces between the system components?”
- “Can we remove ambiguity and ensure correctness?”
4. Construction:
   - “How can we code the components to meet the specification?”
   - “How do we write stylish code?”

5. Testing:
   - “Does the finished system satisfy the requirements?”
   - “Can we break the system?”
Traditional Lifecycle – Key Questions

6. Installation:
   - “What do the system administrators have to do?”
   - “How can we educate the end users?”

7. Maintenance:
   - “Can we find and fix the faults?”
   - “Can we improve the system?”
<table>
<thead>
<tr>
<th>Phase</th>
<th>Output deliverables</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Engineering</td>
<td>High level architectural specification</td>
</tr>
<tr>
<td>Requirements Analysis</td>
<td>Requirements specification</td>
</tr>
<tr>
<td></td>
<td>Functional specification</td>
</tr>
<tr>
<td></td>
<td>Acceptance test specification</td>
</tr>
<tr>
<td>Design</td>
<td>Software architecture specification</td>
</tr>
<tr>
<td></td>
<td>System test specification</td>
</tr>
<tr>
<td></td>
<td>Design specification</td>
</tr>
<tr>
<td></td>
<td>Sub-system test specification</td>
</tr>
<tr>
<td></td>
<td>Unit test specification</td>
</tr>
<tr>
<td>Construction</td>
<td>Program code</td>
</tr>
<tr>
<td>Testing</td>
<td>Unit test report</td>
</tr>
<tr>
<td></td>
<td>Sub-system test report</td>
</tr>
<tr>
<td></td>
<td>System test report</td>
</tr>
<tr>
<td></td>
<td>Acceptance test report</td>
</tr>
<tr>
<td></td>
<td>Completed system</td>
</tr>
<tr>
<td>Installation</td>
<td>Installed system</td>
</tr>
<tr>
<td>Maintenance</td>
<td>Change requests</td>
</tr>
<tr>
<td></td>
<td>Change request report</td>
</tr>
</tbody>
</table>
Analysis != Design

- Analysis identifies ‘what’ the system must do
  - The analyst seeks to understand the organization, its requirements and its objectives

- Design specifies ‘how’ it will do it
  - The designer seeks to specify a system that will fit the organization, provide its requirements effectively and assist it to meet its objectives

- It is important to distinguish the two activities and the associated mindset
  - We need to know ‘what’ before we decide ‘how’
Traditional Lifecycle: Strengths

- Tasks in phases may be assigned to specialized teams
- Project progress evaluated at the end of each phase
- Can be used to manage projects with high levels of risks
Traditional Lifecycle: Problems

- Real projects rarely follow such a simple sequential life cycle
- Iterations are almost inevitable in real projects but are expensive and problematic with the TLC
- Lapsed time between systems engineering and the final installation is long
- Unresponsive to changes during project as iteration is difficult

Underlying assumptions:
- *perfect understanding between developers and customers*
- *a crystal ball that foresees the future*
The cost of this form of iteration increases as the project progresses making it impractical and not effective
Iterative Development

- Divide the project into many iterations, each of which can be viewed as a mini-project in its own right.
Incremental Development

- Deliver working, free-standing, useful ‘chunks’ of software, one at a time.
Iterative & Incremental Development: Spiral Model

- Initial requirements gathering and project planning
- Further planning based on user comments
- User evaluation of increments
- Planning
- Risk analysis
- User evaluation
- Software development
- Risk analysis based on initial requirements
- Risk analysis based on user reaction to plan
- Go, no-go decision
- Risk assessment
- Progress towards final system
- Develop first increment
- Develop next increment

Progress towards final system

Develop first increment

Develop next increment
Iterative & Incremental Development: advantages

- **risk mitigation** — making it possible to identify risks earlier and to take action
- **change management** — changes to requirements are expected and properly managed
- **team learning** — all the team can be involved from the start of the project
- **improved quality** — testing begins early and is not done as a ‘big bang’ with no time
Project Lifecycles

Attack major risks early and continuously, or they will attack you.
Take Home Messages

- Project Lifecycles
  - Traditional Lifecycle (TLC) / Waterfall
  - Iterative & Incremental Development / Spiral