System Development Life Cycle

Ratna Wardani

Semester Genap, 2013
Outline

- SDLC
- Project Phase
- System Development Methodologies
SDLC

Planning

Implementation

Analysis

Design
Project Phase

1. **Planning**: Why build the system?
   - System request, feasibility analysis, project size estimation

2. **Analysis**: Who, what, when, where will the system be?
   - Requirement gathering, business process modeling

3. **Design**: How will the system work?
   - Program design, user interface design, data design

4. **Implementation**: System construction and delivery
   - System construction, testing, documentation and installation
Planning

1. Identifying **business value** *(System Request)*
   - Lower costs
   - Increase profits

2. Analyze **feasibility**
   - Technical Feasibility
   - Economic Feasibility
   - Organizational Feasibility

3. Develop **workplan and staffing** *(WBS)*
Analysis

1. Requirement gathering by answering the questions:
   - Who will use the system?
   - What will the system do?
   - When will it be used?

2. Investigate the current system

3. Identify possible improvements

4. Develop a concept for new system
Design

1. **Program Design** *(UML Diagrams)*
   - What programs need to be written
   - Exactly what each program will do

2. **User Interface Design**
   - How users interact with system
   - Forms / reports used by the system

3. **Data Design** *(ER Diagrams)*
   - What data is to be stored
   - What format the data will be in
   - Where the data will be stored
Implementation

**Construction**
- New system is built and tested
- Often testing is the longest part

**Testing**
- Unit Testing
- Integration Testing
- System Testing
- User Acceptance Test

**Installation**
- Old system is turned off
- New system is turned on
Processes and Deliverables

<table>
<thead>
<tr>
<th>Process</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td>System Proposal</td>
</tr>
<tr>
<td>Analysis Design</td>
<td>System Specification</td>
</tr>
<tr>
<td>Implementation</td>
<td>New System with Testing/Maintenance Plan</td>
</tr>
</tbody>
</table>
System Development Methodologies
What is Methodology

- A formalized approach to implementing the SDLC (series of steps and deliverables)

- Writing code without a well-thought-out system request may work for small programs, but rarely works for large ones
Major Methodologies

1. **Structured Design**
   - Waterfall method
   - Parallel development

2. **RAD Development**
   - Phased Development
   - Prototyping
   - Throw-away Prototyping

3. **Agile Development**
   - Extreme Programming (XP)
   - Scrum
Structured Design Methodology

Projects move methodically from one to the next step

Generally, a step is finished before the next one begins
Waterfall Method
# Pros-Cons Waterfall Method

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifies systems requirements long before programming</td>
<td>Design must be specified on paper before programming begins</td>
</tr>
<tr>
<td>Begins, it <strong>minimizes change</strong> to the requirements as the project proceed (mature)</td>
<td>Long time between system proposal and delivery of new system</td>
</tr>
<tr>
<td></td>
<td>Rework is very hard</td>
</tr>
</tbody>
</table>
Parallel Development

Addresses problem of time gap between proposal and delivery

General process:
1. Breaks project into parallel subproject
2. Integrates them at the end
Parallel Development

FIGURE 1-3  A Parallel Development-based Methodology

10/13/2013  Ratna W/PSBO2012
Rapid Application Development

1. Phased development
   - A series of versions

2. Prototyping
   - System prototyping

3. Throw-away prototyping
   - Design prototyping
Rapid Application Development

Critical elements to speed up the SDLC:

- CASE tools
- Visual programming languages
- Code generators
RAD: Phased Development

- Break overall system into a series of versions
- Each version has Analysis, Design, and Implementation
- Output from on version is the input to the next
- Incorporate ideas, issues, lessons learned in one version into the next version
RAD: Phased Development
# RAD: Phased Development

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets useful system to users <strong>quickly</strong></td>
<td>Initial system is intentionally incomplete</td>
</tr>
<tr>
<td>Most <strong>important</strong> functions tested most</td>
<td>System requirements expand as users see versions</td>
</tr>
</tbody>
</table>
RAD: Prototyping

- Analysis, Design, Implementation are performed concurrently
- Start with a "quick-and-dirty" prototype
  - Provides minimal functionality
- Repeat process, refining the prototype each time
- Stop when prototype is a working system
RAD: Prototyping
## RAD: Prototyping

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets working system to users <strong>quickly</strong></td>
<td>Fast paced. Hard to conduct careful, methodical analysis</td>
</tr>
<tr>
<td><strong>Reassures users</strong> that the project is progressing</td>
<td>Initial design decisions have long term staying power</td>
</tr>
<tr>
<td><strong>Quickly refines</strong> true requirements</td>
<td>Problems may come to light late in design, requiring re-design</td>
</tr>
</tbody>
</table>
RAD: Throw-Away Prototyping

- Use prototypes only to understand requirements
  - Example: use html to show UI
- Prototype is **not** a working design
- Once requirements are understood, the prototypes are thrown away
- The system is then built using SDLC
RAD: Throw-Away Prototyping

Diagram:
- Planning
- Analysis
- Design
- Implementation
- Design prototype
- System
- Implementation
Agile Development

- Just a few rules that are easy to learn and follow
- Streamline the SDLC
  - Eliminate much of the modeling and documentation
  - Emphasize simple, iterative application development
- Examples include:
  - Extreme Programming (XP)
  - Scrum
  - Dynamic Systems Development Model (DSDM)
Extremem Programming (XP)

“Core Values” of XP

1. Communication – All to All
2. Simplicity – KISS, refactoring
3. Feedback – Embrace Change
4. Courage – Quality First, test and efficient coding
Extreem Programming (XP)

1. User Stories about system do
2. Code small program using defined standards
   - Naming conventions
   - Coding practices
3. User Feedback
4. Repeat
Extreem Programming (XP)
Selecting the Appropriate Methodology

1. Clarity of User Requirements
2. Familiarity with Technology
3. System Complexity
4. System Reliability
5. Short Time Schedules
6. Schedule Visibility
## Selecting the Right Methodology

<table>
<thead>
<tr>
<th>Ability to Develop Systems</th>
<th>Structured Methodologies</th>
<th>RAD Methodologies</th>
<th>Agile Methodologies</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Waterfall</td>
<td>Parallel</td>
<td>Phased</td>
</tr>
<tr>
<td>with Unclear User Requirements</td>
<td>Poor</td>
<td>Poor</td>
<td>Good</td>
</tr>
<tr>
<td>with Unfamiliar Technology</td>
<td>Poor</td>
<td>Poor</td>
<td>Good</td>
</tr>
<tr>
<td>that are Complex</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>that are Reliable</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>with a Short Time Schedule</td>
<td>Poor</td>
<td>Good</td>
<td>Excellent</td>
</tr>
<tr>
<td>with Schedule Visibility</td>
<td>Poor</td>
<td>Poor</td>
<td>Excellent</td>
</tr>
</tbody>
</table>
Suppose you are an analyst for the Roanoke Software Consulting Company (RSCC), a large consulting firm with offices around the world. The company wants to build a new knowledge management system that can identify and track the expertise of individual consultants anywhere in the world based on their education and the various consulting projects on which they have worked. Assume that this is a new idea that never done before been attempted in RSCC or elsewhere. RSCC has an international network, but the offices in each country may use somewhat different hardware and software. RSCC management wants the system up and running within a year.