2 Software life span models

• Stages through which software goes, from conception to death

• Stages may be very different

• Software = product
  – stages are similar to the stages in the life span of other products
Product lifespan
Software lifespan

• Software is a product
  – sales go through the same lifespan

• Unique proprietary software
  – value follows the same curve

• Names of stages are different
Staged model

Initial development

Evolution

Servicing or Maintenance

Phase-out

Close-down

© 2012 Václav Rajlich  Software Engineering: The Current Practice  Ch. 2
Initial development

• Requirements
• Design
• Implementation
  – similar to waterfall, but of limited duration

• Fundamental decisions
  – technology
    • programming language, coding conventions, libraries,…
  – architecture
    • components, interactions
  – program domain knowledge
    • the knowledge is required for evolution
Evolution

• Adapts the application to the ever-changing user and operating environment
• Adds new features
• Corrects mistakes and misunderstandings
• Responds to both developer and user learning
• Responds to changes in technology
• Program usually grows during evolution
• Both software architecture and software team knowledge make evolution possible
Evolution stops

• Managerial decision
  – business reasons to stop evolution

• Software stabilization
  – problem is solved
  – no reason to continue evolution

• Code decay
Code decay

- Loss of software coherence
- Loss of the software knowledge
  - less coherent software requires more extensive knowledge
  - if the knowledge is lost, the changes will lead to a faster deterioration
- Loss of key personnel = loss of knowledge
- Challenge: eliminate or slow code decay
Servicing = Maintenance

• No additions of new functionality
• Changes are limited to patches and wrappers
  – less costly, but they cause further deterioration
• Process is very different from evolution
  – no need for senior engineers
  – the process is predictable
    • well suited to process measurement and management
Reversal from servicing to evolution

• Expensive, rare

• Not simply a technical problem
  – the knowledge of the software team must also be recovered
Reengineering

- Initial development
  - first running version
  - software changes

- Evolution
  - reengineering
  - code decay

- Servicing
  - servicing discontinued

- Phase-out
  - switch-off

- Close-down
Phase-out

• No more servicing is being undertaken
  – but the system still may be in production

• Users must work around known deficiencies
Close-down

• Software use is disconnected
  – current life of successful software:
  – about 10 to 20 years

• Users are directed towards a replacement

• An ‘exit strategy’ is needed.
  – changing to another system requires retraining
  – what to do with long-lived data?
Versioned staged model

- Used by software with many users

- Evolution is the backbone of the process
  - evolution produces versions
  - versions are serviced, phased-out, closed down
Mozilla Firefox releases

- 2008 – 2009
- Versions 2.0 and 3.0 – serviced in parallel
- Version 3.5 introduced 4/2009 – while version 3.0 still serviced – while version 2.0 in phase-out

<table>
<thead>
<tr>
<th>2.0</th>
<th>3.0</th>
<th>3.5</th>
<th>version #</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>x</td>
<td></td>
<td>2.0.0.12/</td>
<td>2/7/2008</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td></td>
<td>3.0b3/</td>
<td>2/13/2008</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td></td>
<td>3.0b4/</td>
<td>3/11/2008</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td></td>
<td>2.0.0.13/</td>
<td>3/25/2008</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td></td>
<td>3.0b5/</td>
<td>4/9/2008</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td>x</td>
<td>2.0.0.14/</td>
<td>4/15/2008</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td></td>
<td>3.0rc1/</td>
<td>5/15/2008</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td>x</td>
<td>3.0rc2/</td>
<td>6/4/2008</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td>x</td>
<td>3.0rc3/</td>
<td>6/11/2008</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td></td>
<td>3.0/</td>
<td>6/19/2008</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td></td>
<td>2.0.0.15/</td>
<td>6/23/2008</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td></td>
<td>2.0.0.16/</td>
<td>7/11/2008</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td></td>
<td>3.0.1/</td>
<td>7/16/2008</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td></td>
<td>2.0.0.17/</td>
<td>9/17/2008</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td></td>
<td>3.0.2/</td>
<td>9/22/2008</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td>x</td>
<td>3.0.3/</td>
<td>10/7/2008</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td>x</td>
<td>2.0.0.18/</td>
<td>11/11/2008</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td></td>
<td>3.0.4/</td>
<td>11/11/2008</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td>x</td>
<td>2.0.0.19/</td>
<td>12/15/2008</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td></td>
<td>3.0.5/</td>
<td>12/15/2008</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td></td>
<td>2.0.0.20/</td>
<td>12/18/2008</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td>x</td>
<td>3.0.6/</td>
<td>2/2/2009</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td>x</td>
<td>3.0.7/</td>
<td>3/3/2009</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td>x</td>
<td>3.0.8/</td>
<td>3/27/2009</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>3.0.9/</td>
<td>4/9/2009</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td></td>
<td>3.5b4/</td>
<td>4/24/2009</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td></td>
<td>3.0.10/</td>
<td>4/27/2009</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td></td>
<td>3.5b99/</td>
<td>6/7/2009</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td></td>
<td>3.0.11/</td>
<td>6/10/2009</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>3.5rc1/</td>
<td>6/16/2009</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>3.5rc2/</td>
<td>6/17/2009</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>3.5rc3/</td>
<td>6/24/2009</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>3.5/</td>
<td>7/1/2009</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>3.5.1/</td>
<td>7/17/2009</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td></td>
<td>3.0.12/</td>
<td>7/20/2009</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td></td>
<td>3.5.2/</td>
<td>7/30/2009</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>3.0.13/</td>
<td>7/31/2009</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>3.5.3/</td>
<td>8/24/2009</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td></td>
<td>3.0.14/</td>
<td>9/8/2009</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td></td>
<td>3.5.4/</td>
<td>10/19/2009</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td>x</td>
<td>3.0.15/</td>
<td>10/26/2009</td>
</tr>
</tbody>
</table>
Incomplete lifespans

- Discontinued projects
  - stopped during initial development

- Stable domain
  - no need for evolution

- Development starts with evolution
  - a related old software is evolved into new one
Lifecycle vs. lifespan model

• Lifecycle
  – common terminology
  – incorrect: There is no cycle
    • some software discontinued without a replacement

• Lifespan model
  – better terminology
  – less commonly used
V-Model

requirements → system design → unit design → implementation → unit testing → system testing → functional testing → maintenance
Prototyping model

requirements

prototype

corrected requirements

design

implementation

maintenance