Evolution of Software Development Process Models

Penn State Harrisburg
CMPSC 487W
Fall 2015

Part of this material taken from Steven Schach:
Object-Oriented Software Engineering

Introduction

- Review of lifecycle approaches
- Move toward iterative models
- Agile methods
- Software process improvement
Rapid Prototyping Model

- Rapid prototype
- Analysis
- Design
- Implementation
- Changed requirements
- Postdelivery maintenance
- Retirement

Code and Fix Model

- Implement the first version
- Modify until client is satisfied
- Postdelivery maintenance
- Retirement
Boehm's Spiral Model

Waterfall Model

Traditional SDLC
Waterfall Model

• We often iterate between phases during development.

We typically do not plan for the reverse arrows.

And they happen—whether we plan for them or not.

Waterfall Model

• We find ourselves in big trouble with arcs like this.

This project is probably out of control.
Waterfall Model

• The bigger the arc, the bigger the trouble.

• Instead of fighting the inevitable changes, why not embrace them, and plan for them?

This project is manageable, and we still have flexibility.

Now, how do we get done?
Many Waterfall Models

- When we stack a bunch of these side-by-side, we can state our expectations for each cycle.

<table>
<thead>
<tr>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>Analysis</td>
<td>Require</td>
<td>ments</td>
</tr>
<tr>
<td>Analysis</td>
<td>Design</td>
<td>elation</td>
<td>Analysis</td>
</tr>
<tr>
<td>Design</td>
<td>Coding &amp; Testing</td>
<td>Design</td>
<td>Coding &amp; Testing</td>
</tr>
<tr>
<td>Coding &amp; Testing</td>
<td>Elab #1</td>
<td>Coding &amp; Testing</td>
<td>Const #1</td>
</tr>
<tr>
<td>Elab #2</td>
<td>Const #2</td>
<td>Const #3</td>
<td></td>
</tr>
</tbody>
</table>

Many Waterfall Models

- The vertical columns are phases that encompass a predetermined number of cycles (increments).
Many Waterfall Models

- The end of each phase represents a milestone. Each milestone has its distinct set of deliverables.

Unified Process

- This idea is the basis for the Unified
Unified Process

• UP is a four-phased approach
  – Inception phase
  – Elaboration phase
  – Construction phase
  – Transition phase
• There can be any number of iterations (increments) within each phase.

During each phase, various workflow activities take place at varying degrees depending on the particular phase that is underway.
Unified Process

• The **requirements** workflow aims to determine the client’s needs
• The **analysis** workflow aims to analyze and refine the requirements
• The **design** workflow aims to refine the analysis workflow until the material is in a form that can be implemented by the programmers
  – **Architectural design**: Decompose the product into modules
  – **Detailed design**: Design each module (algorithms)
• The **implementation** workflow aims to deploy operational software in the chosen environment.

The Inception Phase begins the process
Unified Process

• The aim of the Inception Phase is to determine whether the proposed software product is economically viable and to some extent, technologically feasible. You need to:
  – Gain an understanding of the domain
  – Build the business model
  – Delimit the scope of the proposed project
    • Focus on the subset of the business model that is covered by the proposed software product
  – Begin to make the initial business case

• Questions that need to be answered include:
  – Is the proposed software product cost effective?
  – How long will it take to obtain a return on investment?
    • Alternatively, what will be the cost if the organization decides not to develop the proposed software product?
  – If the software product is to be sold in the marketplace, have the necessary marketing studies been performed?
  – Can the proposed software product be delivered in time?
  – What will be the impact if the proposed software product is delivered late?
Unified Process

• The deliverables of the Inception Phase include:
  – The initial version of the business domain model
  – The initial version of the requirements artifacts
  – A preliminary version of the analysis artifacts
  – A preliminary version of the architecture
  – The initial list of risks
  – The initial ordering of the use cases (Chapter 10)
  – The plan for the elaboration phase
  – The initial version of the business case

Unified Process

Stephen Schach says...

• There is insufficient information at the beginning of the
  Inception Phase to plan the entire development
• The only planning that is done at the start of the project is the
  planning for the Inception Phase itself
• For the same reason, the only planning that can be done at
  the end of the Inception Phase is the plan for just the next
  phase, the Elaboration Phase

Note: In the real world, you must have some sort of
overall time schedule with a coarse set of deliverables
and milestones. Otherwise your ROI model is fiction
from the outset.
Unified Process

• The Elaboration Phase follows the Inception Phase

<table>
<thead>
<tr>
<th>Requirements workflow</th>
<th>Analysis workflow</th>
<th>Design workflow</th>
<th>Implementation workflow</th>
<th>Test workflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception phase</td>
<td>Elaboration phase</td>
<td>Construction phase</td>
<td>Transition phase</td>
<td></td>
</tr>
</tbody>
</table>

• The aim of the Elaboration Phase is to refine the initial requirements. You need to:
  – Refine the architecture
  – Monitor the risks and refine their priorities
  – Complete the business model
  – Refine the business case
  – Produce the project management plan

• The major activities of the elaboration phase are refinements or elaborations of the previous phase
Unified Process

• The tasks of the Elaboration Phase correspond to:
  – Completing the requirements workflow, to the extent possible
  – Performing the entire analysis workflow, to the extent possible
  – Starting the design of the architecture

Unified Process

• The deliverables of the Elaboration Phase include:
  – The completed domain model
  – The completed business model
  – The completed requirements artifacts
  – The completed analysis artifacts
  – An updated version of the architecture
  – An updated list of risks
  – The project management plan (for the rest of the project)
  – The completed business case
The aim of the Construction Phase is to produce the first operational-quality version of the software product.
- This is sometimes called the beta release.

The emphasis in this phase is on Implementation and Testing:
- Unit testing of modules
- Integration testing of subsystems
- Product testing of the overall system
Unified Process

- The deliverables of the Construction Phase include:
  - The initial user manual and other manuals, as appropriate
  - All the artifacts including code (beta release versions)
  - The completed architecture
  - The updated risk list
  - The project management plan (for the remainder of the project)
  - Deployment plan
  - Training plan
  - If necessary, the updated business case

Unified Process

The Transition Phase follows the Construction Phase
Unified Process

- The aim of the transition phase is to ensure that the client's requirements have indeed been met
  - Faults in the software product are corrected

- The deliverables of the transition phase include:
  - All the artifacts (final versions)
  - The completed manuals
  - Training

RUP

- This UP was extended by Rational in the Rational Unified Process.

---

(c) Julia M. Lobur

CMPSC 487W
RUP

- RUP is much more involved than UP, and is proprietary.

Enterprise Unified Process

- The extension of RUP, the Enterprise Unified Process (EUP) accounts for the entire asset lifecycle to include project management.
Agile Methods

- Agile processes are a collection of new paradigms characterized by:
  - Less emphasis on analysis and design
  - Earlier implementation (working software is considered more important than documentation)
  - Responsiveness to change
  - Close collaboration with the client

- Invented by seventeen software developers (later dubbed the "Agile Alliance") at a Utah ski resort over two days in February 2001 and produced the Manifesto for Agile Software Development
  - The Agile Alliance did not prescribe a specific life-cycle model
  - Instead, they laid out a group of underlying principles
Agile Methods

• A principle in the *Agile Manifesto* is to deliver working software frequently
  – Ideally every 2 or 3 weeks
• One way of achieving this is to use timeboxing
  – Used for many years as a time-management technique
  – A specific amount of time is set aside for a task
  – Typically 3 weeks for each iteration
  – The team members then do the best job they can during that time

Agile Methods

• Agile processes have had some successes with small-scale software development
  – However, medium- and large-scale software development is very different
• The key decider: the impact of agile processes on postdelivery maintenance
  – Refactoring is an essential component of agile processes
  – Refactoring continues during maintenance
  – Will refactoring increase the cost of post-delivery maintenance, as indicated by preliminary research?
Agile Methods

• Instantiations include:
  – XP - Extreme Programming
  – SCRUM = Iterative development
    + project management
• Agile processes are certainly good when requirements are vague or changing
• Not so good when high reliability and safety are involved.
• Becoming the preferred development approach
  – Many advantages (to be discussed later).

Agile Methods

• Lean Software Development
  – Derives from approach used by Toyota Motors
  – More of a credo than a methodology
• 7 key principles
  1. Eliminate Waste
  2. Build Quality In
  3. Create Knowledge
  4. Defer Commitment
  5. Deliver Fast
  6. Empower the Team (Respect People)
  7. Optimize The Whole (Improve the System)
Agile Methods

#1 Eliminate Waste
Sources of waste:

- unnecessary code or functionality
- starting more than can be completed
- delay in the software development process
- unclear or constantly changing requirements
- bureaucracy slow or ineffective communication
- partially done work
- defects and quality issues
- task switching

Much easier said than done!

Agile Methods

#2 – Build In Quality
Use best quality practices such as:

- Pair Programming
- Test-driven development
- Frequent integration
- Adherence to standards (automated tools help)
- Code inspections

We will look at some of these in a few weeks.
Agile Methods

#3 – Create [Share] Knowledge

• Pair Programming
• Code reviews
• Documentation
• Wikis
• Thoroughly commented code
• Knowledge sharing sessions
• Training
Agile Methods

#4 – Defer Commitment

- Making decisions too early runs the risk that something significant will change.
- Make decisions regarding items that are difficult or impossible to reverse as late as possible.
- Keep decisions about features and the development of those features close together to ensure that the right product is delivered.

#5 – Deliver Fast

- Faster delivery reduces risk of changing requirements.
- Objective of fast delivery reduces temptation to over-engineer.
- Fast delivery is facilitated by reducing waste and building-in quality.
Agile Methods

#6 – Empower the Team (Respect People)

- A consequence of eliminating waste (no bureaucracy) and deciding as late as possible.
- But-- decisions must be communicated to those who need to know (Create knowledge!)
- Respond promptly and courteously to questions.
- Do not dismiss points of view that differ from yours.
- Empathize--try to see things as others see them.


#7 – Optimize The Whole (Improve the System)

- Stay focused on the system being delivered.
- The biggest problems are at the interfaces.
- Understand how the system will be used, by whom, and where.
"The fundamental problem with software development is that the software process is badly managed."

CHAOS reports from 1994 to 2004

- Improving the software process leads to
  - Improved software quality
  - Delivery on time, within budget
- Improved management leads to
  - Improved techniques
- Software process improvement initiatives include:
  - Capability maturity model (CMM) / CMMI
  - ISO 9000-series
  - ISO/IEC 15504
Software Process Improvement

- The CMM is a product of the Software Engineering Institute (SEI) at Carnegie-Mellon University.
- The SEI was established by the Department of Defense in 1984.
- It is a set of strategies for improving the software process
  - SW–CMM for software
  - P–CMM for human resources ("people")
  - SE–CMM for systems engineering
  - IPD–CMM for integrated product development
  - SA–CMM for software acquisition
- These strategies are unified into CMMI (capability maturity model integration)

Software Process Improvement

- The SEI rates the effectiveness of organizations software development processes through a set of 5 "maturity" levels.
- The levels are determined by the mastery of "key process areas."
- The maturity levels as defined by the CMMI are:
  - 1 = Initial
  - 2 = Repeatable
  - 3 = Defined
  - 4 = Managed
  - 5 = Optimized
- A higher number indicates better control over development processes.
Software Process Improvement – CMM Levels

1 = Initial
   – Ad hoc processes. Success depends on individual talent and "heroics."

2 = Repeatable
   – Basic project management processes for tracking cost, schedule, and functionality. Earlier successes on projects repeatable on similar applications and scope.

3 = Defined
   – Standard processes are established and improved over time. Process is customized using standard processes using standardized guidelines. Process is qualitatively predictable.

4 = Managed
   – Quantitative and statistical methods are used to continuously improve processes. Process quality goals are established. Process is quantitatively predictable.

5 = Optimized
   – Established continuous process performance improvement using quantitative process-improvement techniques. Processes are nimble, adaptable and innovative supported by an empowered workforce that is aligned with the business values and objectives of the organization.
Software Process Improvement

SEI Key Process Areas

<table>
<thead>
<tr>
<th>CMMI Category</th>
<th>Process Areas</th>
<th>CMMI Category</th>
<th>Process Areas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
<td>Project Planning</td>
<td>Process</td>
<td>Organizational Process Focus</td>
</tr>
<tr>
<td>Management</td>
<td>Project Monitoring and Control</td>
<td>Management</td>
<td>Organizational Process Definition</td>
</tr>
<tr>
<td></td>
<td>Supplier Agreement Management</td>
<td></td>
<td>Organizational Training</td>
</tr>
<tr>
<td></td>
<td>Integrated Project Management</td>
<td></td>
<td>Organizational Process Performance</td>
</tr>
<tr>
<td></td>
<td>Risk Management</td>
<td></td>
<td>Organizational Innovation and Deployment</td>
</tr>
<tr>
<td></td>
<td>Quantitative Project Management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support</td>
<td>Configuration Management</td>
<td>Engineering</td>
<td>Requirements Management</td>
</tr>
<tr>
<td></td>
<td>Process and Product QA</td>
<td></td>
<td>Requirements Development</td>
</tr>
<tr>
<td></td>
<td>Measurement and Analysis</td>
<td></td>
<td>Technical Solution</td>
</tr>
<tr>
<td></td>
<td>Decision Analysis and Resolution</td>
<td></td>
<td>Product Integration</td>
</tr>
<tr>
<td></td>
<td>Causal Analysis and Resolution</td>
<td></td>
<td>Verification &amp; Validation</td>
</tr>
</tbody>
</table>

- Self-assessments are encouraged, but the "official" rating is done by an SEI-certified auditor.
- Moving from one level to the next can take 18 months to 3 years, but breaking free of Level 1 can take 5 years or more.
- **Important**: The purpose of the CMM/CMMI is process improvement-- not the assignment of a "grade" to a process.
- Unfortunately, it is being used that way, which invites abuse and unethical behavior.
Software Process Improvement

- Other CMMI drawbacks
  - Invites "heavyweight" processes that are unsuitable for some development shops.
  - Too focused on process management and not focused enough on product development.
  - Militates against innovative approaches such as agile processes.
  - You have to meet 100% of the criteria at a certain maturity level to be certified at that level.
    - e.g., you can meet 20% of Level 5, 70% of Level 4, 80% of Level 3, 90% of Level 2, and you would still be at maturity Level 1.

CMMI is an effective tool when used properly.

ISO 9000

- A set of five standards for industrial activities
  - ISO 9001 for quality systems
  - ISO 9000-3, guidelines to apply ISO 9001 to software
- Some overlap with CMM, but they are not identical
  - Not process improvement
- Stresses documenting the process
- Emphasizes measurement and metrics
  - ISO 9000 is required to do business with the EU
  - Also required by many U.S. businesses, including GE
  - More and more U.S. businesses are ISO 9000 certified
ISO/IEC 15504

- Original name: Software Process Improvement Capability dEtermination (SPICE)
- International process improvement initiative
- Started by the British Ministry of Defence (MOD)
- Includes process improvement, software procurement
- Extends and improves CMM, ISO 9000
- A framework, not a method
- CMM, ISO 9000 conform to this framework
- Now referred to as ISO/IEC 15504 or just 15504

Process Improvement Payback

- Hughes Aircraft (Fullerton, CA) spent $500K (1987-90)
  - Savings: $2M per year, moving from level 2 to level 3
- Raytheon moved from level 1 in 1988 to level 3 in 1993
  - Productivity doubled
  - Return of $7.70 per dollar invested in process improvement
- Tata Consultancy Services (India) used ISO 9000 and CMM (1996-90)
  - Errors in estimation decreased from 50% to 15%
  - Effectiveness of reviews increased from 40% to 80%
Process Improvement Payback

- Improving software practices reduce costs an average of ~35% per year.
- The improvement is sustainable for several years, and the potential is much higher.
- The best organizations have sustained cost improvements of 55%+ per year
- Improved software practices shorten schedules an average of ~15-20% per year.
- The best organizations have sustained schedule improvements of ~20-25% per year.


Process Improvement Payback

- Does methodology matter?
  - Poor quality is the single largest cost driver for most projects.
  - Improved software practices improve quality an average of ~40% per year.
  - Improving software practices reduce costs an average of ~35% per year.
  - Improved software practices shorten schedules an average of ~15-20% per year.

Process Improvement Payback

- **Bottom Line:**
  - Improved software practices pay an average ROI of 5-to-1, and continued improvement is sustainable for many years.
  - The best organizations have sustained ROIs of 9-to-1 on software improvement initiatives for many years.

  Consistent, well-managed processes improve worker satisfaction and beget high-quality, reusable products, saving time and money.

Conclusion

- No one-size-fits-all process
- Iterative methods control risk
- Process improvement provides manageability and high quality
- Process improvement usually pays for itself
Web References

• OpenUP
  – www.eclipse.org/epf/
  – http://epf.eclipse.org/wikis/openup/ (OpenUP wiki)
• Software Processes
  – www.sei.cmu.edu (Software Engineering Institute)
• Agile processes
  – www.agilealliance.org
  – agilemanifesto.org
  – martinfowler.com/articles/newMethodology.html
  – leansoftwareengineering.com
  – www.scrumalliance.org
• Architecture
  – www.viewpoints-and-perspectives.info
• General
  – www.multicians.org/thvv/proverbs.html "Software Engineering Proverbs"

Questions?