Critical Factors Characterizing Projects and Lifecycle Models

30th Pacific NW Software Quality Conference (PNSQC)  
October 8-10, 2012

Kal Toth, kalmanctoth@gmail.com  
Herman Migliore, herm@cecs.pdx.edu

Context (1)
Methodologists / proponents of development lifecycles:
- Boost their favorite approaches ... often with built-in bias
- Rarely a balanced perspective i.e. both benefits and challenges
- Rarely discuss which problems they are best suited to solve or why

Assertion: A given lifecycle is not “best” for every project!

Flow of this Paper (contents)
Puts forward 8 critical factors characterizing lifecycles & projects
- Factored out existing process cultures, competencies, and biases

Describes suggested process for matching lifecycles & projects

Characterizes five common / generic lifecycles:
- Waterfall, Incremental*, Spiral*, Evolutionary*, Agile* (iterative variants)

Assesses & depicts lifecycles in terms of the 8 factors

Illustrates selection process with 2 hypothetical projects

8 Critical Factors Characterizing SW Dev.

- Quality/Maintainability: Completeness, sufficiency and currency properties of the processes, delivered software, and delivered documentation (specs, design, test etc.)
- Application Domain: Relative problem difficulty ranging from casual web-sites, games, financial transaction systems, health IT systems, medical devices, aircraft navigation systems, space vehicles
- Size and Complexity: small, simple, linear programs < 1000K vs. large, complex systems > 500K LOC (size and complexity tend to correlate)
- Uncertain Requirements: Degree of requirements precision / ambiguity whether documented or not
- Requirements Volatility: Rate at which customer, context, and functional / non-functional requirements change (may be related to prior item)
- User Involvement: Users review and approve documents vs. getting intensely involved in writing user stories, requirements specs, design, software development, testing, and acceptance ...
- Urgency/Time to Market: Relative urgency to deliver to market or to the customer
- Progress Visibility: May be provided by way of informal functional demonstrations, high level progress reports, reporting of tasks, modules, and deliverable completion levels, various metrics

High

Medium

Low
Postulated Model

1. [Pre-condition]: each lifecycle characterized in terms of 8 critical factors
   - collect, categorize and analyze data from a large sample of actual projects
2. For each project, compile collected data from stakeholders (incl. cust/users):
   - objectives, context, assumptions, resources, constraints and priorities
3. Prune candidate lifecycles eliminating the obvious including incompatibilities with competencies and culture
4. From compiled project data, assess/estimate nominal values for the 8 critical characterizing factors
5. Apply a matching algorithm to compare project’s characterization data with each lifecycle model’s profile
   - Goal: estimate the “degree of fit” selecting the “best”
6. Conduct sensitivity and trade-off analyses:
   - Vary project characterization data
   - Incorporate project costing and scheduling estimates

Figure 1: Waterfall Lifecycle Model [11]

Waterfall

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Benefits</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relatively sequential with development phases, major milestones, &amp; specified deliverables reviewed by stakeholders</td>
<td>Fosters thorough requirements, architecture and design before implementation</td>
<td>Not very adaptive to project changes or market demands</td>
</tr>
<tr>
<td>At each phase loop back to prev. phase to correct problems Formal change control procedures to correct problems in earlier phases which may modify costs and schedule</td>
<td>Formalizes documentation and deliverables which facilitates project and contract management</td>
<td>Project visibility limited to documentation</td>
</tr>
</tbody>
</table>

Figure 2: Waterfall Lifecycle Model Characterized

Figure 3: Iterative Lifecycle Models

Iterative

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Benefits</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variants: WP, Incremental, Spiral, Evolutionary and Agile</td>
<td>Parallel / concurrent development allows better schedules than waterfall Early discovery of problems Customer feedback – more likely to meet requirements Visibility into progress Process improvement (PI), lessons learned (LL)</td>
<td>Harder to manage project than waterfall Harder to write contracts and subcontracts</td>
</tr>
</tbody>
</table>

Figure 3: Incremental Lifecycle Model [11]
Incremental Lifecycle Model

| Attributes | An iterative process that partitions large complex problems into independent parts, some of which may be mission-critical, and concurrently develops and integrates the parts |
| Requirements & architecture should be stable prior to partitioning and change controls should be in place after baselining |
| Appropriate for multiple delivery and release of capabilities |

| Benefits | Supports concurrent development, partial/progressive deliveries |
| Separate parts can be monitored separately enhancing visibility |

| Shortcomings/Disadvantages | Mapping requirements to increments can be challenging |
| Unanticipated changes to requirements & architecture can break across increments and imply major rework later on |

Figure 4: Incremental Lifecycle Model Characterized

<table>
<thead>
<tr>
<th>Characterizing Factors</th>
<th>Quality / Maintainability</th>
<th>Application Domain</th>
<th>Size / Complexity</th>
<th>Uncertain Requirements</th>
<th>Progress Visibility</th>
<th>User Involvement</th>
<th>Requirements Visibility</th>
<th>Urgency / Time-to-Market</th>
</tr>
</thead>
<tbody>
<tr>
<td>Characterizing Factors</td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Spiral Lifecycle Model

| Attributes | A risk-driven plan-oriented iterative model where each spiral is a development iteration that aims to establish a plan for the next spiral (a.k.a. iteration). Risk assessments prior to each spiral determine the activities scheduled for a given spiral/iteration. Reviews at the end of each iteration include an assessment of "lessons learned" that feed the next spiral |

| Benefits | Early iterations (spirals) systematically focus on consolidating the requirements and exploring technical problem areas through prototyping and simulating. Later iterations transition in more waterfall-like iterations of development – concurrent spirals represent increments of development |

| Shortcomings/Disadvantages | Project management and contracting more challenging as it requires more discipline to incorporate concurrency, risk assessment, and lessons learned |

Figure 6: Spiral Lifecycle Model Characterized

<table>
<thead>
<tr>
<th>Characterizing Factors</th>
<th>Quality / Maintainability</th>
<th>Application Domain</th>
<th>Size / Complexity</th>
<th>Uncertain Requirements</th>
<th>Progress Visibility</th>
<th>User Involvement</th>
<th>Requirements Visibility</th>
<th>Urgency / Time-to-Market</th>
</tr>
</thead>
<tbody>
<tr>
<td>Characterizing Factors</td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Evolutionary Lifecycle Model

Figure 7: Evolutionary Lifecycle Model [11]

Adapted from [7] (considerably simplified)

1. Objectives, Alternatives & Constraints
2. Risk Analysis & Aversion
3. Product Development
4. Planning & Management
5. Develop & Verify Next Level Products
6. Next Spiral
7. Service/Offering
8. Review
9. Evaluate

Cycle 1:
- Analyze
- Design
- Implement
- Test
- Evaluate

Cycle 2:
- Analyze
- Design
- Implement
- Test
- Evaluate

Cycle 3:
- Analyze
- Design
- Implement
- Test
- Evaluate
**Evolutionary Lifecycle Model**

| Attributes | An iterative exploratory development model for solving hard (non-trivial) technical problems and uncertainties. Work products of this model are designed to discover technical solutions and elicit customer/user feedback. Work products of evolutionary development are not considered to be of operational/deliverable quality.

| Benefits/Advantages | Focuses project stakeholders (developers, managers, customers, and users) on feasibility and requirements rather than a solution. Detailed functions and features, as well as product qualification tasks such as reviews and testing can be avoided.

| Shortcomings/Disadvantages | There is a danger that managers and customers assume the prototypes to be of deliverable quality - they are not! And their expectations of actual progress will be inflated.

---

**Agile Lifecycle Model**

| Attributes | An incremental strategy that builds solutions from “stories” over short development iterations (typically 1-2 weeks). Focus is on working software over documentation. Embraces change and close customer involvement. Stories are typically prioritized and put into a backlog. Planning is typically “time-boxed.” Some methods advocate pair-programming (e.g. XP). Often employ “test-driven development” (TDD).

| Benefits/Advantages | Adaptive to change due to light-weight documentation. Higher acceptance rate due to close customer involvement. Informal stories and constant design refactoring reduces time and schedule defining requirements.

| Shortcomings/Disadvantages | Customers don't always participate. Frequent re-factoring can cause brittle systems. Vulnerable to turnover and lack of documentation. Harder to write contracts to meet vaguely stated requirements. May not scale to large, complex and mission-critical projects.

---

**Figure 8: Evolutionary Lifecycle Model Characterized**

<table>
<thead>
<tr>
<th>Characterizing Factors</th>
<th>Quality/Maintainability</th>
<th>Application Domain</th>
<th>Size/Complexity</th>
<th>Uncertain Requirements</th>
<th>Progress Visibility</th>
<th>User Involvement</th>
<th>Volatility</th>
<th>Urgency/Time-to-Market</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 10: Agile Lifecycle Model Characterized**

<table>
<thead>
<tr>
<th>Characterizing Factors</th>
<th>Quality/Maintainability</th>
<th>Application Domain</th>
<th>Size/Complexity</th>
<th>Uncertain Requirements</th>
<th>Progress Visibility</th>
<th>User Involvement</th>
<th>Volatility</th>
<th>Urgency/Time-to-Market</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 9: The Life of an Agile Story [11]**

- **Development Tasks**
  - Get Partner
  - Implementation
  - Unit Tests
  - Integration
  - Plan Design
  - Re-factor

- **A Development Task**
  - Get Partner
  - Implementation
  - Unit Tests
  - Integration

**Summary Characterization of Lifecycle Models**

<table>
<thead>
<tr>
<th>Waterfall</th>
<th>Incremental</th>
<th>Spiral</th>
<th>Evolutionary</th>
<th>Agile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality/Maintainability</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Application Domain</td>
<td>M, H</td>
<td>M, H</td>
<td>M, H</td>
<td>L, M, H</td>
</tr>
<tr>
<td>Size/Complexity</td>
<td>L, M, H</td>
<td>M, H</td>
<td>M, H</td>
<td>L, M, H</td>
</tr>
<tr>
<td>Uncertain Requirements</td>
<td>L</td>
<td>M</td>
<td>M</td>
<td>H</td>
</tr>
<tr>
<td>Progress Visibility</td>
<td>L</td>
<td>M</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>User Involvement</td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>Requirements Volatility</td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>H</td>
</tr>
</tbody>
</table>
Selecting a Lifecycle
(Hypothetical Projects A and B)

Summary

Goal: explored possibilities – not reams of data
Proposed 8 critical factors for characterizing lifecycles & projects
Proposed a process for characterizing lifecycles & projects:

Biggest challenges:
- Collecting & analyzing data to empirically characterize lifecycles
- Semi-quantitative techniques for characterizing new projects
- Developing an effective project-to-lifecycle matching process
  - May be possible to adapt software estimating and COTS selection techniques [refs]

Questions?

Welcome constructive criticism and validation

Hopefully this will motivate research & assessment projects that build on the ideas presented