Abstract

This paper proposes a collaboration model designed to facilitate principled negotiation among stakeholders when estimating software projects. This approach builds on the concepts introduced by this author [1] utilizing the concepts of estimating uncertainty and principled negotiation of Boehm [2] and McConnell [3]. The landing zone and air traffic control metaphor is used to visualize project sponsors and solution providers working closely together to target and jointly arrive at mutually acceptable software estimates and plans. The proposed approach is iterative, constantly feeding back newly discovered and negotiated project information to adjust estimates and project commitments in order to stay within the project’s “landing zone”. This strategy has been developed to help customers and solution providers achieve mutually acceptable project success more consistently than less principled techniques in common practice.
**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!
Context (2)

Prudent engineering manager should seek objective evidence:
  ▪ Which life-cycle process(es) best support the project's unique needs?
  ▪ Which … yield positive ROI given core competencies?
  ▪ What are the risks of project overrun and downstream failures?

Should we take the project? retrain? pass?
Aim of this Paper

Advance the discovery of a process that will select [most] appropriate lifecycle for the planned project

- Assumption: Certain critical factors characterize lifecycles & projects
- Hypothesis: a practical characterization matching process exists

Preliminary Remarks:

- Herein postulating and describing a possible approach
- Encourage motivated practitioners to explore this problem further
- Hopeful side-effects of this paper:
  - Recognition that any given lifecycle is not suitable for all projects
  - Practitioners should consider adapting & combining lifecycles
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]
  ▪ Their distinguishing features, merits, shortcomings
  ▪ 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 (reqts, 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 intensively 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
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

> 1970 refinement of earlier stagewise model (had no feedback between stages).

> Primary difficulty was the emphasis on fully elaborated documents as completion criteria for early requirements and design phases.

  -- Assumes that users fully understand his/her needs

  -- Problems arise when, in testing, it is realized that requirements are missing or poorly defined.

> Phases are associated with baselines and milestones. This model was adopted early on as a standard (prescriptive) by the DoD for software contracting (DoD Std 2167-A), mainly for measurement and control of projects
# Waterfall Model

<table>
<thead>
<tr>
<th>Waterfall</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Attributes</strong></td>
<td>Relatively sequential with development phases, major milestones, &amp; specified deliverables reviewed by stakeholders. 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>
</tr>
<tr>
<td><strong>Benefits Advantages</strong></td>
<td>Fosters thorough requirements, architecture and design before implementation. Formalizes documentation and deliverables which facilitates project and contract management.</td>
</tr>
<tr>
<td><strong>Shortcomings Disadvantages</strong></td>
<td>Not very adaptive to project changes or market demands. Project visibility limited to documentation. Customer and user feedback and refinement (too) late to incorporate lessons learned into the current project.</td>
</tr>
</tbody>
</table>
Figure 2: Waterfall 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 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>
## Iterative Lifecycle Models

<table>
<thead>
<tr>
<th>Iterative</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Attributes</strong></td>
</tr>
<tr>
<td>Variants: WP, Incremental, Spiral, Evolutionary and Agile</td>
</tr>
<tr>
<td>Repeated cycles, ongoing rework</td>
</tr>
<tr>
<td>Parallel / concurrent development</td>
</tr>
<tr>
<td><strong>Benefits</strong></td>
</tr>
<tr>
<td><strong>Advantages</strong></td>
</tr>
<tr>
<td>Parallel / concurrent development allows better schedules than waterfall</td>
</tr>
<tr>
<td>Early discovery of problems</td>
</tr>
<tr>
<td>Customer feedback – more likely to meet requirements</td>
</tr>
<tr>
<td>Visibility into progress</td>
</tr>
<tr>
<td>Process improvement (PI), lessons learned (LL)</td>
</tr>
<tr>
<td><strong>Shortcomings</strong></td>
</tr>
<tr>
<td><strong>Disadvantages</strong></td>
</tr>
<tr>
<td>Harder to manage project than waterfall</td>
</tr>
<tr>
<td>Harder to write contracts and subcontracts</td>
</tr>
</tbody>
</table>
Figure 3: Incremental Lifecycle Model [11]
## Incremental Lifecycle Model

<table>
<thead>
<tr>
<th>Incremental</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Attributes</strong></td>
</tr>
<tr>
<td><strong>Benefits/Advantages</strong></td>
</tr>
<tr>
<td><strong>Shortcomings/Disadvantages</strong></td>
</tr>
</tbody>
</table>
Figure 4: Incremental Lifecycle Model Characterized

Characterizing Factors | Quality/Maintainability | Application Domain | Size/Complexity | Uncertain Requirements | Progress Visibility | User Involvement | Requirements Volatility | Urgency/Time-to-Market
---|---|---|---|---|---|---|---|---
High |  |  |  |  |  |  |  |  
Medium |  |  |  |  |  |  |  |  
Low |  |  |  |  |  |  |  |  

[Diagram showing the incremental lifecycle model with factors and levels]
Figure 5: Boehm’s Spiral Model [11]
Adapted from [7] (considerably simplified)

1. Objectives, Alternatives & Constraints
   Cycle Definition
   Project Planning
   Develop & Verify Next Level Products
   Cumulative Cost

2. Risk Analysis & Aversion
   Risk Assessment
   Prototyping
   Sim. Modeling

3. Product Development

4. Planning & Management
   Commitment
   next spiral
   R = Review

R = Review
## Spiral Lifecycle Model

<table>
<thead>
<tr>
<th>Spiral</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
</tr>
<tr>
<td>Benefits Advantages</td>
</tr>
<tr>
<td>Shortcomings Disadvantages</td>
</tr>
</tbody>
</table>
Figure 6: Spiral Lifecycle Model Characterized
When you cannot specify requirements fully in advance, this provides a way to investigate alternatives before committing to full-scale development.

Some products do not lend themselves to partitioning into discrete pieces (like compilers).
# Evolutionary Lifecycle Model

<table>
<thead>
<tr>
<th><strong>Evolutionary</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Attributes</strong></td>
</tr>
<tr>
<td><strong>Benefits/Advantages</strong></td>
</tr>
<tr>
<td><strong>Shortcomings/Disadvantages</strong></td>
</tr>
</tbody>
</table>
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>Requirements Volatility</th>
<th>Urgency / Time-to-Market</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<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>

Arrows indicating the evolution of factors from low to high.
Figure 9: The Life of an Agile Story [11]
# 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 10: Agile Lifecycle Model Characterized

Characterizing Factors:
- Quality / Maintainability
- Application Domain
- Size / Complexity
- Uncertain Requirements
- Progress Visibility
- User Involvement
- Requirements Volatility
- Urgency / Time-to-Market
## Summary Characterization of Lifecycle Models

<table>
<thead>
<tr>
<th></th>
<th>Waterfall</th>
<th>Incremental</th>
<th>Spiral</th>
<th>Evolutionary</th>
<th>Agile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality/Maintainability</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>L-M</td>
<td>L-M</td>
</tr>
<tr>
<td>Application Domain</td>
<td>M, H</td>
<td>M, H</td>
<td>M, H</td>
<td>L, M, H</td>
<td>L</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>
<td>L</td>
</tr>
<tr>
<td>Uncertain Requirements</td>
<td>L</td>
<td>M</td>
<td>M</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>Progress Visibility</td>
<td>L</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>H</td>
</tr>
<tr>
<td>User Involvement</td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>M</td>
<td>H</td>
</tr>
<tr>
<td>Requirements Volatility</td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>Urgency</td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>H</td>
<td>H</td>
</tr>
</tbody>
</table>
Selecting a Lifecycle
(Hypothetical Projects A and B)
Figure 11 attempts to visualize the project-to-lifecycle matching process. The characterized project of eight estimates, one for each factor, is represented as a single line graph that is superimposed over each of the (five) previously characterized lifecycles figures. This figure illustrates, for example, that the Waterfall Model meets or exceed the project’s needs in 5 out of 8 areas; while the Spiral and Agile models meet or exceed project needs in 7 areas (note that the failing matches are in different areas).
Figure 11 attempts to visualize the project-to-lifecycle matching process. The characterized project of eight estimates, one for each factor, is represented as a single line graph that is superimposed over each of the (five) previously characterized lifecycles figures. This figure illustrates, for example, that the Waterfall Model meets or exceed the project’s needs in 5 out of 8 areas; while the Spiral and Agile models meet or exceed project needs in 7 areas (note that the failing matches are in different areas).
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