1010 GUIDE–A SIMPLE FRAMEWORK FOR SELECTING THE APPROPRIATE SDLC FOR A SOFTWARE PROJECT

Vaibhav V Prakash and Danny Sunderesan

Master’s student, Software Engineering, Erik Jonsson School of Engineering, Department of Computer Science and Engineering, The University of Texas at Dallas, USA.

ABSTRACT

1010 guide is essentially a flowchart to help software managers choose an efficient software project management methodology based on the metrics they have. Differentiation between critical and non-critical projects which is followed by choosing the pre-defined metrics for the team. Finally a table corresponds to the options chosen and results in a straightforward selection of the appropriate SDLC based on the values given.

KEYWORDS

Software Engineering, Software development life cycle, Software Project Planning and Management

1. INTRODUCTION

Building software systems can become easily complex and unpredictable if not tracked on a systematic basis since uncertainty is inherent in them. The only way to reduce uncertainty and adapt to changing requirements is to select the right software development life cycle (SDLC). The metrics for building software differ depending on the type of system being built viz. application software, real-time systems, embedded software, distributed system etc. Here we present a simple algorithmic framework which will help choosing the most appropriate SDLC for the software system being built. This is particularly aimed at Program/Project managers in the software industry who are responsible for driving and delivering projects.
2.1010 FRAMEWORK

There is no simple framework currently for selecting the right SDLC. However, there are many bits and pieces which have to connect together in order for a Program/Project manager to effectively decide which process is to be adapted. The goal of this paper is very simple – to make the selection process of a SDLC easy. Here we have tried to reach the balance between a too simplistic model and a very sophisticated model. A single SDLC cannot be applied to any project. There will always be pro’s and con’s and deciding which one can be a daunting and confusing task. 1010 guide essentially mitigates the confusion and gives a clearer flow to the thought process. In the end, the decision still rests upon the individual Program/Project manager to use their discretion based upon their project circumstances. 1010 guide will help them make a smarter decision.

3.METHODOLOGY

Here, we use the divide and conquer technique. We have shortlisted 15 of the most prominent software development methodologies. By using this technique, in the first iteration we short list 7 or lesser (there can be overlaps). Then we list two to three major plus points of each methodology and leave it to the user’s discretion to choose the best which suit their needs based on the plus points. Yes, it’s that simple.

Flowchart for the thought process
What kind of software are you building?

- CRITICAL
  - Embedded
  - Application Software
  - Real-time

- NON-CRITICAL
  - Embedded
  - Application Software
  - Real-time

What are the metrics for your project?

- Scope
  - Large
  - Small

- Cost
  - Constrained
  - Flexible

- Time
  - Constrained
  - Flexible

- Team experience
  - Experienced
  - New

- Requirements
  - Changing
  - Fixed

- Result
  - High
  - Low

- Organisational
density
  - High
  - Low

- Soft maintenance
density
  - Easy deliver
  - Hard deliver

- Culture & Structure
  - Monolithic, consolidated architectural
    standards
  - Microservices, cloud-based
    standards
  - Internal, parallelised
    standards
  - External, distributed
    standards

- Team size
  - Small
  - Large

- Criticality
  - Critical
  - Non-critical

- Uncertainty
  - High
  - Low

- Product
  - New
  - Improvement
### 4. PREDEFINED TABLES

<table>
<thead>
<tr>
<th>Scope</th>
<th>Cost</th>
<th>Time</th>
<th>Methodology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large</td>
<td>Constrained</td>
<td>Constrained</td>
<td>9 (d) / 4 with 15*</td>
</tr>
<tr>
<td>Large</td>
<td>Flexible</td>
<td>Flexible</td>
<td>3 / 10 with 15*</td>
</tr>
<tr>
<td>Large</td>
<td>Flexible</td>
<td>Constrained</td>
<td>6 with 15*</td>
</tr>
<tr>
<td>Large</td>
<td>Constrained</td>
<td>Flexible</td>
<td>9 (c) with 15*</td>
</tr>
<tr>
<td>Small</td>
<td>Constrained</td>
<td>Constrained</td>
<td>1</td>
</tr>
<tr>
<td>Small</td>
<td>Flexible</td>
<td>Flexible</td>
<td>13</td>
</tr>
<tr>
<td>Small</td>
<td>Flexible</td>
<td>Constrained</td>
<td>6</td>
</tr>
<tr>
<td>Small</td>
<td>Constrained</td>
<td>Flexible</td>
<td>11</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Team maturity</th>
<th>Organization maturity</th>
<th>Methodology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Low</td>
<td>9 (b)</td>
</tr>
<tr>
<td>High</td>
<td>High</td>
<td>14</td>
</tr>
<tr>
<td>Low</td>
<td>High</td>
<td>7 (c)</td>
</tr>
<tr>
<td>High</td>
<td>Low</td>
<td>12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Team size</th>
<th>Methodology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual and small</td>
<td>7 (b)</td>
</tr>
<tr>
<td>Virtual and big</td>
<td>7 (c)</td>
</tr>
<tr>
<td>Real and small</td>
<td>7 (a)</td>
</tr>
<tr>
<td>Real and big</td>
<td>12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Culture and Structure</th>
<th>Methodology</th>
</tr>
</thead>
<tbody>
<tr>
<td>H/Predefined standards</td>
<td>4 / 7 (c)</td>
</tr>
<tr>
<td>H/Individual freedom</td>
<td>4 / 7 (b)</td>
</tr>
<tr>
<td>V/Predefined standards</td>
<td>1</td>
</tr>
<tr>
<td>V/Individual freedom</td>
<td>7 (a)</td>
</tr>
<tr>
<td>Maintenance</td>
<td>Methodology</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Maintain after delivery</td>
<td>12</td>
</tr>
<tr>
<td>Deliver only</td>
<td>1 / 8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requirements</th>
<th>End Result</th>
<th>Methodology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed</td>
<td>Progressive</td>
<td>1 / 3</td>
</tr>
<tr>
<td>Fixed</td>
<td>Deliver only</td>
<td>1</td>
</tr>
<tr>
<td>Changing</td>
<td>Progressive</td>
<td>4</td>
</tr>
<tr>
<td>Changing</td>
<td>Deliver only</td>
<td>4 / 5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Uncertainty</th>
<th>Methodology</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>3 / 7</td>
</tr>
<tr>
<td>Low</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Product</th>
<th>Methodology</th>
</tr>
</thead>
<tbody>
<tr>
<td>New(Innovation)</td>
<td>4 / 5</td>
</tr>
<tr>
<td>Improve/Modify existing</td>
<td>6</td>
</tr>
</tbody>
</table>

15* - If hardware components are involved

5. LEGEND

1. Waterfall development (Simplicity)[1]
   - Small projects
   - Fixed requirements
   - One-time final result
2. Prototyping[2]
   • Changing requirements
   • Progressive results
   • Good for critical projects with emphasis on testing/quality assurance

3. Incremental development[3]
   • Progressive results
   • Good when time is constrained
   • Changing requirements/High uncertainty

4. Iterative and incremental development[3]
   • Good when scope is large
   • Team is real and big
   • Product is new

5. Spiral development[4]
   • Good for large & critical projects
   • Progressive results/Innovation
   • Flexible requirements

   • Time is constrained
   • Improving product
   • Flexible requirements
• Good for integrating modules

   a) DSDM
      • Customer involvement
      • Quick and repeated deliverables
   b) Kanban
      • Responsiveness to changing requirements
      • Good for horizontal structure and individual freedom
   c) Scrum
      • Changing requirements
      • Increased collaboration between customer and intra-team
      • Excellent when the team is new

   • Time is constrained&One-time delivery
   • Fixed requirements and small scope

9. Lightweight methodologies[8]
   a) Adaptive software development
      • Good for new products
      • Progressive results
   b) Crystal Clear[9]
      • Reflective improvement
      • New teams relying on automation tools
• Large teams requiring collaboration

c) Extreme Programming[10]
• Cost saving
• Good for small projects with changing requirements
• Reliable and quick delivery
d) FDD[11]
• Cost saving
• Time saving
• Requirements have to be fixed
e) ICONIX[12]
• Use Case driven
• Different modules following OO design

• Large complex projects
• Good for recovering from failed projects
• Creates progressive business value

11. Incremental funding methodology[14]
• Cost is constrained.

12. Structured system analysis and design[15]
• Good for maintaining after delivery
• Good if the there are multiple modules with dependencies
13. Slow programming[16]

- Time is high
- Team is stressed
- Design is complex


- Adaptable framework
- Primarily iterative
- De-facto for many projects

15. V-model/Hybrid V(System Engineering)[18]

- Highly adaptable
- Interfacing hardware/software
- Innovation/Fault tolerance can be incorporated

6. SURVEY

This method was given to a total of 40 (3 to 5 people in a team) teams at the Erik Jonsson School of Engineering, The University of Texas at Dallas. All the teams were graduate students and had an array of projects going on in Software Systems. This was the feedback we got from them.

Will you use this in selecting the SDLC for your project?

YES – 100%

Would you be ok if this framework was made the default standard in selecting the SDLC for a project?
How would you rate this framework?

Must read & helpful – 92%
Not sure – 6%
Not helpful – 2%

7. CONCLUSION

1010guide greatly simplifies the process of selecting the correct SDLC. This method is especially helpful for managers who have less time on their hands and the project has too many parameters to be taken into account before the planning framework.

8. ACKNOWLEDGEMENT

We would like to thank our Professor, Rym Z Wenkstern, Director of Software Engineering at The University of Texas at Dallas for her guidance and support.

9. REFERENCES


