1010GUIDE–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 straight forward 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 appropriateSDLC 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.

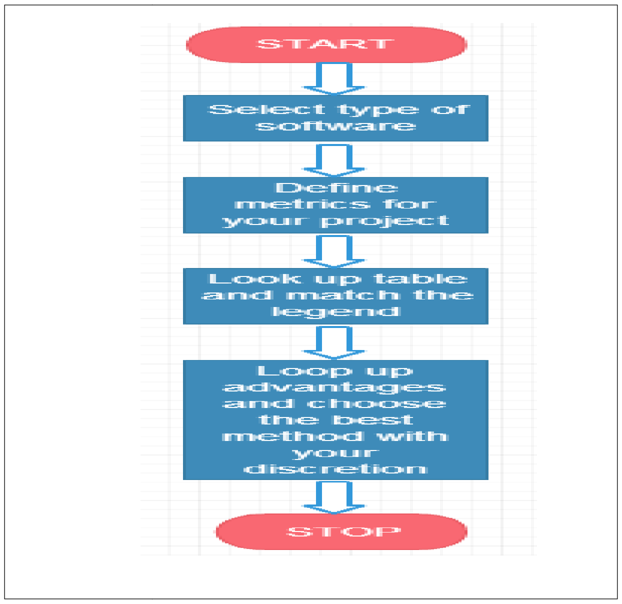
2.1010 FRAMEWORK

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.

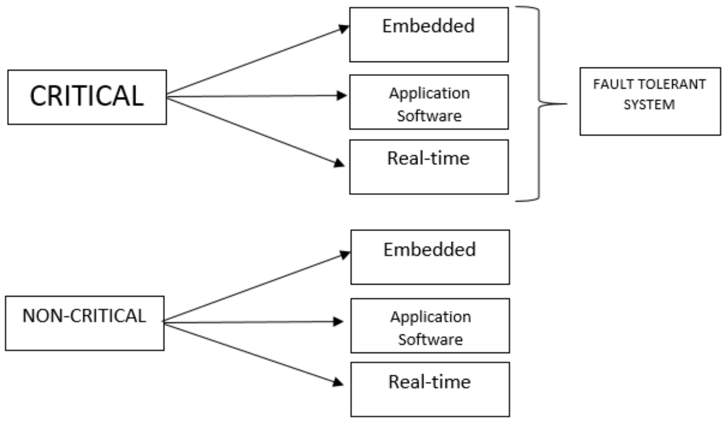
3.METHODOLOGY

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.

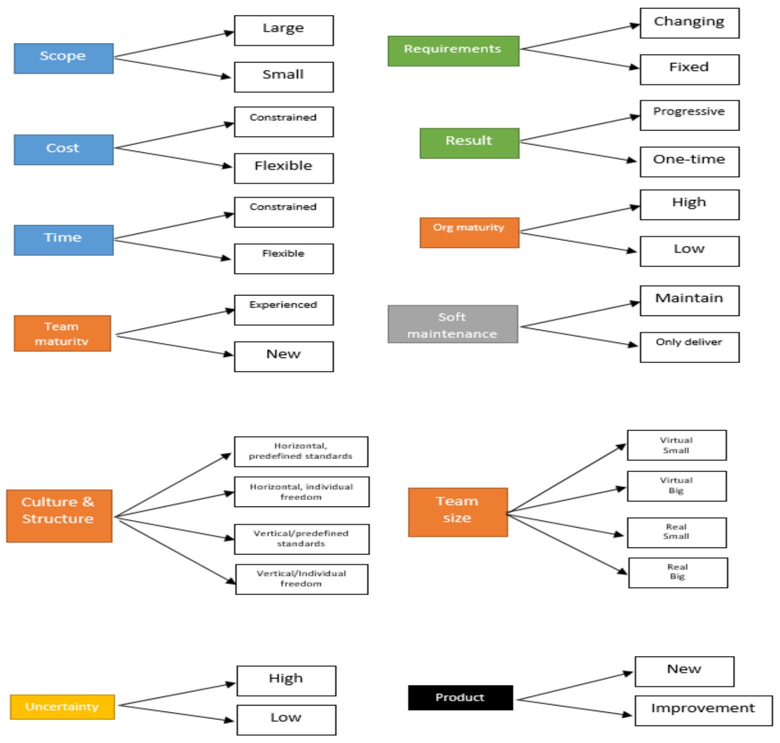
Flowchart for the thought process


START
Select type of software
Define metrics for your project
Look up table and match the legend
Loop up advantages and choose the best method with your discretion
STOP

What kind of software are you building?



What are the metrics for your project?



4.PREDEFINED TABLES

Scope	Cost	Time	Methodology
Large	Constrained	Constrained	9 (d) / 4with 15*
Large	Flexible	Flexible	3 / 10 with 15*
Large	Flexible	Constrained	6 with 15*
Large	Constrained	Flexible	9 (c) with 15*
Small	Constrained	Constrained	1
Small	Flexible	Flexible	13
Small	Flexible	Constrained	6
Small	Constrained	Flexible	11

Team maturity	Organization maturity	Methodology
Low	Low	9 (b)
High	High	14
Low	High	7 (c)
High	Low	12

Team size	Methodology
Virtual and small	7 (b)
Virtual and big	7 (c)
Real and small	7 (a)
Real and big	12

Culture and Structure	Methodology
H/Predefined standards	4 / 7 (c)
H/Individual freedom	4 / 7 (b)
V/ Predefined standards	1
V/ Individual freedom	7 (a)

Maintenance	Methodology
Maintain after delivery	12
Deliver only	1 / 8

Requirements	End Result	Methodology
Fixed	Progressive	1 / 3
Fixed	Deliver only	1
Changing	Progressive	4
Changing	Deliver only	4 / 5

Uncertainty	Methodology
High	3 / 7
Low	2

Product	Methodology
New(Innovation)	4 / 5
Improve/Modify existing	6

15* - If hardware components are involved

5. LEGEND

1. Waterfall development (Simplicity)[1]

- Small projects
- Fixed requirements
- One-time final result

CRITICAL PROJECTS

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

NON-CRITICAL PROJECTS

6. Rapid action development[5]

- Time is constrained
- Improving product
- Flexible requirements

- Good for integrating modules

7. Agile development[6]

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

8. Code and fix[7]

- 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

10. Chaos model[13]

- 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

14. Rational Unified Process[17]

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

Embedded systems (Both Critical and Non-Critical)

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?

YES-94%, NO-6%

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

[1] Benington, Herbert D. (1 October 1983). "Production of Large Computer Programs". IEEE Annals of the History of Computing (IEEE Educational Activities Department) 5 (4): 350–361. doi:10.1109/MAHC.19183.10102. Retrieved 2011-03-21.

[2] Smith MF Software Prototyping: Adoption, Practice and Management. McGraw-Hill, London (1991).

[3] Dr. Alistair Cockburn (May 2008). "Using Both Incremental and Iterative Development". STSC CrossTalk (USAF Software Technology Support Center) 21 (5): 27–30. ISSN 2160-1593. Retrieved 2011-07-20.

[4] Boehm, B, "Spiral Development: Experience, Principles,and Refinements", Special Report CMU/SEI-2000-SR-008, July 2000

[5] Boehm, Barry (May 1988). "A Spiral Model of Software Development". IEEE Computer. Retrieved 1 July 2014.

[6] Abrahamsson, P., Salo, O., Ronkainen, J., &Warsta, J. (2002). Agile Software Development Methods: Review and Analysis. VTT Publications 478.

[7] Hughes, Bob and Cotterell, Mike (2006). Software Project Management, pp.283-289. McGraw Hill Education, Berkshire. ISBN 0-07-710989-9

[8] http://en.wikipedia.org/wiki/Lightweight_methodology

[9] "Crystal Methods Methodology | Infolific". Mariosalexandrou.com. Retrieved 2013-07-25.

[10] "Manifesto for Agile Software Development", Agile Alliance, 2001, webpage: Manifesto-for-Agile-Software-Dev

[11] Coad, P., Lefebvre, E. & De Luca, J. (1999). Java Modeling In Color With UML: Enterprise Components and Process. Prentice Hall International. (ISBN 0-13-011510-X)

[12] Rosenberg, D. & Stephens, M. (2007). Use Case Driven Object Modeling with UML: Theory and Practice. Apress. (ISBN 1590597745)

[13] ACM Digital Library, The chaos model and the chaos cycle, ACM SIGSOFT Software Engineering Notes, Volume 20 Issue 1, Jan. 1995

[14] http://en.wikipedia.org/wiki/Incremental_funding_methodology

[15] Mike Goodland; Karel Riha (20 January 1999). "History of SSADM". SSADM – an Introduction. Retrieved 2010-12-17.

[16] http://www.martinfowler.com/bliki/TechnicalDebt.html

[17] Jacobson, Sten (2002-07-19). "The Rational Objectory Process - A UML-based Software Engineering Process". Rational Software Scandinavia AB. Retrieved 2014-12-17.

[18] Clarus Concept of Operations. Publication No. FHWA-JPO-05-072, Federal Highway Administration (FHWA), 2005