Midterm

Ponder & Prove 09

# Waterfall

## Aspect – Plan, Control and Monitor Testing

“Without question the biggest user of project resources, whether it be manpower, computer time, or management judgement, is the test phase. It is the phase of greatest risk in terms of dollars and schedule.” [1] Testing involves visual inspection, testing every logic path, and determining when to do the final checkout. [1]

## Essential – Complexity

“From the complexity comes the difficulty of enumerating, much less understanding, all the possible states of the program, and from that comes the unreliability.” [2]

The motivation behind plan, control, and monitor testing is to help manage the essential problem of complexity by enforcing reliability in tested areas.

# Rapid Prototype

## Aspect – Creating a Prototype

“Prototyping is the process of quickly building a model of the final software system, which is used primarily as a communication tool to assess and meet the information needs of the user.” [7]

## Essential – Invisibility

“In spite of progress in restricting and simplifying the structures of software, they remain inherently unvisualizable [sic], thus depriving the mind of some of its most powerful conceptual tools. This lack not only impedes the process of design within one mind, ***it severely hinders communication among minds.***”[2]

“I would go a step further and assert that it is really impossible for clients, even those working with software engineers, to specify completely, precisely, and correctly the exact requirements of a modern software product before having built and tried some versions of the product they are specifying.” [2]

The motivation behind creating a prototype in the Rapid Prototype method is to help manage the essential problem of invisibility by quickly translating from invisible thoughts to a visible prototype that reveals what the software could eventually become. This is a powerful communication tool for specifying and verifying requirements. [2]

# Spiral Model

## Aspect – Flexible, Risk-Based Planning

“The spiral model can accommodate most previous models as special cases and further provides guidance as to which combination of previous models best fits a given software situation.” [3]   
“Each cycle of the spiral begins with the identification of the objectives of the portion of the product being elaborated (performance, functionality, ***ability to accommodate change***, etc.); the alternative means of implementing this portion of the product (design A, design B, reuse, buy, etc.); and the constraints imposed on the application of the alternatives (cost, schedule, interface, etc.)” [3]

## Essential – Changeability

“In short, the software product is embedded in a cultural matrix of applications, users, laws, and machine vehicles. These all change continually, and their changes inexorably force change upon the software product.” [2]

The motivation behind the risk based planning aspect of the Spiral Model is to help manage the essential problem of changeability by explicitly planning to accommodate change.

# Agile – XP

## Aspect – User Stories written by On-site customer

“User stories serve the same purpose as use cases but are not the same. They are used to create time estimates for the release planning meeting. They are also used instead of a large requirements document. User Stories are written by the customers” [4]

“An on-site customer is a must and a valuable recipient of numerous rapid feedbacks in unit-testing and task-estimations.” [5] “Functional tests are written by customers to convince themselves that a system as a whole works as it is expected to.” [5]

## Essential – Invisibility

“In spite of progress in restricting and simplifying the structures of software, they remain inherently unvisualizable [sic], thus depriving the mind of some of its most powerful conceptual tools. This lack not only impedes the process of design within one mind, ***it severely hinders communication among minds.***”[2]

The motivation behind having an on-site customer write user stories and functional tests is to help manage the essential problem of invisibility by providing direct communication between the client and the developers about what the client wants and whether the developer has met those wants.

# Agile – SCRUM

## Aspect – Sprints

“Agile development methodology provides opportunities to assess the direction of a project throughout the development lifecycle. This is achieved through regular cadences of work, known as sprints or iterations, at the end of which teams must present a potentially shippable product increment.” [6]

## Essential – Complexity

“Software entities are more complex for their size than perhaps any other human construct, because no two parts are alike.” [2]

“Harlan Mills proposed that any software system should be grown by ***incremental development***. … I have seen the most dramatic results since I began urging this technique. … I find that teams can grow much more complex entities in four months than they can build.” [2]

The motivation behind the sprint aspect of the Agile – SCRUM method is to help manage the essential problem of complexity by growing the software in iterations.

# Cleanroom

## Aspect – Statistical Quality Measurements

“The Cleanroom process has been designed to carry out [the measurement of statistical quality of delivered software]. It calls for the development of software in increments that permit realistic measurements of statistical quality during development.” [8]

## Essential – Complexity

“From the complexity comes the difficulty of enumerating, much less understanding, all the possible states of the program, and from that comes the unreliability.” [2]

The motivation behind statistical quality measurements in the Cleanroom method is to help manage the essential problem of complexity by replacing debugging with mathematical verification and statistical quality. [8] As programs are inherently complex, debugging of some form will always be needed, and by replacing traditional debugging with the aforementioned methods, Cleanroom seeks to reduce some of this inherent complexity.

# Formal Methods

## Aspect – Formal Language

“A formal method in software development is a method that provides a formal language for describing a software artifact (e.g. specifications, designs, source code) such that formal proofs are possible, in principle, about properties of the artifact so expressed.” [9]

## Essential – Invisibility

“In spite of progress in restricting and simplifying the structures of software, they remain inherently unvisualizable [sic], thus depriving the mind of some of its most powerful conceptual tools. This lack not only ***impedes the process of design*** within one mind, it severely hinders communication among minds.”[2]

The motivation behind having a formal language inside of Formal Methods is to help manage the essential problem of invisibility by providing a strict, and provable method of representing software.

# CMM

## Aspect – Key Process Area – Requirements Management

“The purpose of Requirements Management is to establish a common understanding between the customer and the software project of the customer's requirements that will be addressed by the software project. This agreement with the customer is the basis for planning (as described in Software Project Planning) and managing (as described in Software Project Tracking and Oversight) the software project. Control of the relationship with the customer depends on following an effective change control process (as described in Software Configuration Management)” [10]

## Essential – Complexity

“The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is [as] difficult as ***establishing the detailed technical requirements***, including all the interfaces to people, to machines, and to other software systems. NO other part of the work so cripples the resulting system if done wrong. No other part is more difficult [to] rectify later.” [2]

The motivation for the requirements management key process area aspect of CMM is to help manage the essential problem of complexity by helping companies define a process with which to manage their requirements.

# Bibliography

[1] W. Royce, "Managing the Development of Large Software Systems," Proceedings of the IEEE WESCON, pp. 1-9, Aug. 1970   
[Online] Available: <http://dl.acm.org/citation.cfm?id=41801>

[2] Chapter 16: No Silver Bullet — Essence and Accident in Software Engineering.  
F. Brooks, The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition  
[Online] Available: <http://worrydream.com/refs/Brooks-NoSilverBullet.pdf> *Emphasis added.*

[3] B. Boehm, "A Spiral Model of Software Development and Enhancement," IEEE, xxx, pp. 61-72, May 1988, [Online] Available: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=59&tag=1> *Emphasis Added*

[4] Wells, D., “User Stories”, [Online Available]: <http://www.extremeprogramming.org/rules/userstories.html> *Emphasis Added*

[5] R. Juric, "Extreme Programming and its Development Practices" 22nd Int. Conf. Information Technology ITI 2000, Jun. 2000   
[Online] Available: [http://ieeexplore.ieee.org/xpls/abs\_all.jsp?arnumber=915842&tag=1](https://content.byui.edu/file/fb36352f-44a4-473d-bb81-1e5a2ce36646/1/CS%20432%20PDFs/Extreme%20programming%20and%20its%20development%20practices.pdf) *Emphasis Added*

[6] M. James, "What Is Agile Methodology?," Oct. 2008,  
[Online] Available: [http://agilemethodology.org/](https://content.byui.edu/items/fb36352f-44a4-473d-bb81-1e5a2ce36646/1/?.vi=file&attachment.uuid=6e0aa107-7495-43a3-8dae-c32d87b642f7) *Emphasis Added*

[7] J. Carey, "Prototyping: Alternative Systems Development Methodology, "Information and Software Technology, vol. 32, no. 2, pp. 119-126, Mar. 1990,   
[Online] Available:[http://www.sciencedirect.com/science/article/pii/0950584990901114](https://content.byui.edu/file/fb36352f-44a4-473d-bb81-1e5a2ce36646/1/CS%20432%20PDFs/Prototyping_alternative%20systems.pdf) *Emphasis Added*

[8] H. Mills et al., "Cleanroom Software Engineering," IEEE Software, vol. 4, no. 5, pp. 19-25, Sep. 1987,   
[Online] Available: [http://dx.doi.org/10.1109/MS.1987.231413](https://content.byui.edu/file/fb36352f-44a4-473d-bb81-1e5a2ce36646/1/CS%20432%20PDFs/Cleanroom%20Software%20Engineering.pdf) *Emphasis Added*

[9] R. Vienneau, "A Review of Formal Methods", pp. 7-14, May 1993  
[Online] Available:[https://www.csiac.org/sites/default/files/A%20Review%20of%20Formal%20Methods.pdf](https://content.byui.edu/items/fb36352f-44a4-473d-bb81-1e5a2ce36646/1/?.vi=file&attachment.uuid=ca83b353-b6e6-4f7a-bf7e-2431a44d9fd4) *Emphasis Added*

[10] M. Paulk et al., "Key Practices of the Capability Mature Model SM, Version 1.1,"Technical Report CMU/SEI-93-TR-025, Feb. 1993.  
[Online] Available: [http://dx.doi.org.byui.idm.oclc.org/10.1109/52.219617](https://content.byui.edu/file/fb36352f-44a4-473d-bb81-1e5a2ce36646/1/08%20-%20Paulk%20-%20Key%20Practices%20of%20the%20CMM.pdf) *Emphasis Added*