Midterm Study Guide — CS 466 Software Processes

Jared Dyreson
California State University, Fullerton

March 6, 2021

Contents

[1]Lecture 3 — Pg. 10 Case Study
[2]Book Chapter 4 — Case Study
[3]Lecture 4 — Pgs. 8 - 13 are a regurgitation of the book

Abstract

This is by no means a comprehensive assessment of the first six lectures/chapters but a condensed version of them. That would defeat the purpose of this document. Please consult the external resources as described in the footnotes through out this study guide. By the way, the book is fantastic, you should read it. Proudly written in vim and LaTeX. Good luck and have fun.

1 Chapter 1 — Introduction

1.1 General Notes — Book

- Specifications are important
- Difference between *mass manufacturing* (Waterfall) and *predictable manufacturing* (Agile/IID).
- Factors that contribute to the demise of Waterfall. Clients:
  - Are not sure what they want
  - Have difficulty stating what they want and know
  - Details of what [they] want will only be revealed during development
  - Details are too complex
  - Change their mind during development
  - External forces lead to changes or enhancement requests
- Agile development is nimble and does not rely on monolithic structures to hinder its performance

1.2 General Notes — Slides

- Project manager is supposed to define **clear objectives** and must also be **adaptive/reactive** when issues arise.
- Make sure you have the proper budget and time period in which the system must be completed including any interim milestones that must be met.
- The IMS is used for both the development team and the client, making sure the project is on track for completion.
- Tasks have; duration (hours), budget (money), and a measure of how status will be reported (50-50, % complete, etc)
- These tasks show that they're; ahead, on, or behind schedule. Overrun can occur if tasks are not completed in the set time frame.

- Programs that successfully manager their schedule buffers tend to be successful (uhh, yeah?)
- IMS helps track down program staffing (hours and budget must be allocated, therefore number of engineers must be known). It is paramount that the correct number of SWE's are deployed on a project at any given time. [4]
- When planning; look for risks (mitigate most if not all), use the least amount of the budget, properly schedule the development team and create a strong startup process. [5]
- As a software manager, you need to successfully utilize the IMS and IMP to create a SBP that conforms to the time table allotted.
- During the startup (while establishing a baseline schedule/budget), the proposal information is used to:
  - Create SBP
  - Support IMS Planning
  - Justify your organization's budget to program management
  - Solidify the WBS you want to use to manage the program
  - Create SDP
- All software activities must have defined hours allotted and an associated budget. Please refrain from using LoE (Level of Effort) in an IMS unless it is explicitly stated to do so. Hard to quantify.
- Risks always need to be evaluated and need to be accounted for in your cost baseline
- These include; program size, performance, future obsolescence, hardware availability and sub-contractor issues.
- Program organization is highly useful to know as it determines the success of a project. Affects ability to implement iterative or an Agile approach.[6]
- Test in the environment that the software will be used [7]. Most expensive errors are found in production. It is paramount that integration test environment accurately reflects the target environment.
- The software process can be contained in the SDP or referenced in other documents included, with proper citations.

[4]Lecture 2 — Pg. 11 - 13
[5]Lecture 2 — Pg. 14
[6]Lecture 2 — Pg. 36 & 37
[7]Tuffix is tested in Ubuntu

1.3 Key Terms

- Change Management Plan — describes how program artifacts will be controlled and how change will be managed during the development of the product
- IMP — Integrated Master Plan, which comprises a hierarchy of program events. Each event is supported by specific accomplishments. Event-driven plan. Structure in which the **IMS** conforms to.
- IMS — Integrated Master Schedule, which details work/planning packages which are necessary to support the **IMP's** events. Time table.
- Work Package — implies the task can be worked within the next several months
- Planning Package — implies the task to be worked on at some future date on the program.
- Critical Chain Project Management — Minimize program schedule impact and keep insight into task implementation. You need to take into consideration Murphy's Law.
- STD — Software Technical Lead
- SPM — Software Project Management
- PMO — Program Office
- SBP — Software Build Plan. Breaks the software major capabilities/features that will be developed (backlog — Agile speak)
- Point of Departure Build — Software baseline for foundation of project and theoretical data to be fed in the form of tests to ensure behavior is correct.
- Framework Enhancement Build — Modifications to the baseline to improve functionality and extendibility of the environment being used (OpenGL — graphics programming, libgc — C library that several programs like OpenGL utilize). See the layering effect.
- Capability/Features Builds — Once the skeleton is created in the form of frameworks, add more components that utilize this newly created foundation, such as graphics and must have features.
- BOE — Basis of Estimate. How many lines of code will be developed and the hours associated with the task's completion. Needs high-level description of what each capability/feature will achieve
- WBS — Work Breakdown Structure. All activities are organized base on hours/budget for activities that nee to be performed as part of program execution. [8]. Software management, support, development, test, and maintenance activities need to be created.
  - SW Organization with respect to the program
  - High-level SW schedule with key milestones
  - SW risks associated the proposed development
  - SW measurements and analysis plan (new, modified, and reused development into account)

[8]See Lecture 2 — Pg. 18 for example and Pg. 28 - 33 for Decomposition

  - SW Configuration management plan (only tested software gets pushed to production)
  - SWE Environment (in-house lab facilities)
  - SW Development paradigm (Agile, EVO, UP, etc)
  - SW Process (IMS, RAM, etc) and artifacts produced. External documents to this one
  - Declaration of FOSS used in development

- SW — Software
- SWE — Software Engineer(ing)
- SDP — Software Development Plan. Captures the management approach and engineering environment effort associated with the program:
- FOSS — Free and Open Source Software
- SPM — Software Program Manager

2 Chapter 2 — Iterative & Evolutionary Development[9]

2.1 General Notes — Book

- *All* software is compiled across the team into *one* iteration release.
- These builds are internal and not pushed to production (PoC — Proof of Concept)
- Goes well with version control systems such as Git. Bugs can be traced to release builds.
- Most people now ascribe to IID however some DoD contracts still conform to the Waterfall methodology.
- It is best to mix and match risk and client driven development, best of both worlds.
- If a timebox cannot be met, the overall scope of the iteration is reduced (not all timeboxes need to be of equal length)
- IID embraces change not chaos. Needless changes are prohibited.
- Software Engineers need only know the qualities of the final build, not the individual mechanisms that make up the system during initial planning

2.2 General Notes — Slides

- Iterative approaches include constructing design, programming, testing and requirement analysis/specification in several intervals. Nothing is static[10].
- The heart of IID ensures that each iteration builds on top of each other, ensuring only tested code can move into production.
- Allows for multiple teams to work concurrently
- Regression tests pinpoint bugs easily
- New capabilities can be debugged
- Baseline ensures everyone is on the same page
- System features can be deployed in increments, allowing for rigorous testing
- At least three internal iterations are completed before delivery to the customer
- Iterations can last from (1 week to 6 months)[11].
- Iterations can be attributed self-contained mini-projects, isolated from one another.

[9] Lecture 3 — Pg. 10 Case Study
[10] Lecture 3 — Pg. 2 for IID diagram
[11] Author states that the recommended length is 1 wk to 6 wks

2.3 Key Terms

- Adaptive development — Element adapt in response to feedback form prior work (user, tests, developers, etc.)
- Client-driven iterative development — Clients choose the contents of the next iteration based on what they perceive is the best business approach
- Cone of uncertainty — An initial phase (early requirements change) of high uncertainty which drops as time passes and information accumulates.
- EVO — First iterative and evolutionary method, starting in the 1960s. Plans iterations by highest value-to-cost ratio, and strongly promotes the unambiguous definition of quality requirements. Program requirements dynamically change over the lifetime of the project's development and are not frozen at the start.
- Evolutionary iterative development — Same as adaptive however the customer provides constructive criticism for future iterations.
- Incremental delivery — Practice of repeatedly delivering a system into production in a series of expanding capabilities.
- Iteration Release — A stable, integrated and tested *partially* complete system.
- Iteration — Self-contained mini-project composed of activities such as requirement analysis, design, programming, and testing.
- Iterative Development — Building software in which the overall lifecycle is composed of several iteration in sequence
- Iterative and Incremental Development (IID) — Growing a system via iterations
- Program SW Configuration Plan — Details how SW artifacts are controlled and how the change of the artifacts is managed. Includes requirements traceability, design documentation, etc.
- Risk-driven iterative development — Riskiest, most difficult elements of a project are chose for the early iterations. [12]
- Timeboxing — Practice of fixing the iteration end date and not allowing it to change. Round robin approach. Four variables that determine the success of the project; time, scope, resources, and quality.
- UP or RUP — [Rational] Unified Process. Focuses on driving down the riskiest elements and the creation of the core architecture of the project.
- Firm-Fixed Price — the type of contract in which the person buying a product or service pays the seller a fixed amount that does not vary even if unexpected costs arise or additional resources are needed.
- Cost Plus Program — an agreement to reimburse a company for expenses incurred plus a specific amount of profit, usually stated as a percentage of the contract's full price.

[12]E.G — The system be able to handle 5,000 simultaneous transactions

3 Chapter 3 — Agile & Iterative Development

3.1 General Notes — Slides

- Agile is a subset of IID methods
- Small timeboxes
- KISS (Keep it Simple Stupid); direct communication, self-directed teams, and working code[13].
- Job of the project manager is to promote he vision and having open communication (just reading this made me feel like Agile is some sort of cult ritual)
- The entire team needs to participate
- Agile can be used on a larger scale
- The entire premise is to constantly be aware of the current needs of the project

3.2 Key Terms

- Ceremony — Amount of documentation, formal steps, reviews, etc
- Cycles — Number and lengths of iterations [14]
- The Agile Manifesto — Like the Communist one, just less bloodshed. The Agile Principles.
- Defined Process — Many predefined and sequential activities
- Empirical Process — Based on frequency measurement and dynamic responses to variable events[15]
- SCRUM — Self-organized teams, with daily standup meetings and daily team measurement. Iterations are in 4 week blocks, with a demo to show for
- XP — Emphasizes collaboration (via peer programming, team working in a common project room), constant refactoring of the code, and test-driven development (practice of developing test cases prior to developing the code). Four values: Communication, simplicity, feedback, and courage
- Crystal Family — Defines project complexity based on the criticality of the end-product and size of staff required to complete the project
  - E6: A project requiring a staff of 1-6 individuals and in which a failure would result in a loss of essential money.
  - E100: A project requiring a staff of 41-100 individuals and in which a failure would result in a loss of life!

[13] Do not employ in romantic relationships
[14] Lecture 4 — Pg. 3 for diagrams of comparisons
[15] Lecture 4 — Pg. 11

4 Motivation & Agile Case Study[16]

Abstract

These chapters are starting to echo each other; I am not adding redundant information.

4.1 General Notes — Slides[17]

- As the complexity of the project increases, so does the amount of requirement creep occurs [18].
- Motivations for IID[19].
- Work that has a shorter deadline gets done faster and better[20].
- People remember slipped dates, but not slipped features
- Tackle small levels of complexity in a short period of time

[16]Book Chapter 4 — Case Study
[17]Lecture 4 — Pgs. 8 - 13 are a regurgitation of the book
[18]Lecture 4 — Pg. 2
[19]Lecture 4 — Pg. 3
[20]Parkinson's Law