Midterm Study Guide — CS 466 Software Processes

Jared Dyreson
California State University, Fullerton

March 4, 2021

Contents

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. [1]

[1] Lecture 2 — Pg. 11 - 13

- 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. [2]
- 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.

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.

[2] Lecture 2 — Pg. 14

- 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. [3]. Software management, support, development, test, and maintenance activities need to be created.
- SW — Software
- SDP — Software Development Plan
- SPM — Software Program Manager

[3]See Lecture 2 — Pg. 18 for example

2 Chapter 2 — Iterative & Evolutionary

2.1 General Notes

- *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.

2.2 Key Terms

- Iterative Development — Building software in which the overall lifecycle is composed of several iteration in sequence
- Iteration — Self-contained mini-project composed of activities such as requirement analysis, design, programming, and testing.
- Iteration Release — A stable, integrated and tested *partially* complete system.
- Iterative and Incremental Development (IID) — Growing a system via iterations
- Risk-driven iterative development — Riskiest, most difficult elements of a project are chose for the early iterations.
- Client-driven iterative development — Clients choose the contents of the next iteration based on what they perceive is the best business approach
- Timeboxing — Practice of fixing the iteration end date and not allowing it to change
- Evolutionary iterative development — Requirements can change over the duration of the project's lifetime rather than using frozen requirements defined in the beginning
- Adaptive development — Element adapt in response to feedback form prior work (user, tests, developers, etc.)
- Cone of uncertainty — An initial phase (early requirements change) of high uncertainty which drops as time passes and information accumulates.

- Incremental delivery — Practice of repeatedly delivering a system into production in a series of expanding capabilities.
- 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.
- UP or RUP — Unified Process. Focuses on driving down the riskiest elements and the creation of the core architecture of the project.