Chapter 4 & 5 Summary

Chris Nutter*

CPSC 362

—

Timestamps

10/08/2020 - 01:58:55 PM

Figures of each model are at the end of the document.

Contents

1 Chapter 4

Prescriptive Process Model is about constructing the order of software development where each task is set in order with specified guidelines. *The Waterfall Model* sets a sequential method to development with no traversal to earlier or later stages (similar to a waterfall that goes straight down and not back up again). *Incremental Process Model* relies on both linear and parallel processing directions to be applied; linear being the sequence of events while jumping around through increments #n. *Evolutionary Process Model* is very iterative which constantly improves upon over time before reaching the final product. It keeps adding to the itself which makes the project bigger and closer to the final product. *Prototyping* is an important step to test your project and practically mandatory for releasing a product. Prototyping is often sent out to testers as a way of getting accurate reception of a product without in-house teams to influence judgement. For example a team creates a math game

*Dedicated to @QuesoGrande a.k.a. Jared D.

for 1st graders. The team will give the product to 1st graders to see how they like the product. *The Spiral Model* is a combination of prototyping with the Waterfall Model method which is very useful for fast development of versions for teams that are stuck in a time constraint. *Concurrent Model or sometimes concurrent engineering* is the repetitive nature of any process model mentioned. *Specialized Process Models* take one or more attributes of other models into one model. *Component-Based Development* is a special model that has many Spiral Model features with pre-packaged software components. There are 5 stages to the component-based development model.

1. Available component-based products are researched and evaluated for the application domain in question.
2. Component integration issues are considered.
3. A software architecture is designed to accommodate the components.
4. Components are integrated into the architecture.
5. Comprehensive testing is conducted to ensure proper functionality.

Aspect-Oriented Software Development is the process and methodological way of defining, specifying, designing, and constructing *aspects*. *Aspects* are ways of subrouting and localizing the expression of crosscutting concerns. *Concerns* are the required parameters of a customer. Putting it all together, *crosscutting concerns* are concerns that are cut across various functions, features, and information bases. *Unified Process* recognises importance with customer communication towards the end goal.

1. *Inception* or communication and planning
2. *Elaboration* or planning and modelling
3. *Construction*
4. *Transition* or construction to deployment
5. *Production* or software implement

Personal Software Process is similar to unified process but for personal development.

1. *Planning* requires isolation of requirements
2. *High-level design* requires component construction and design
3. *High-level design review* requires verification methods to find error

4. *Development* is for design refining.

5. *Postmortem* is the collected measure of each stage to analyse effectiveness.

Team Software Process is similar to PSP but requires a further team-based curriculum to tackle. Various team building suggestions include moral boosting seminars, planning goals accordingly, accelerate process by coaching, providing guidance in a respectable manner, and facilitate university teaching.

2 Chapter 5

Agility is very important and widely spread across software engineering. Ivar Jacobson says "Everyone is agile. An agile team is a nimble team able to appropriately respond to changes." Agility involves the philosophy of team structures and team attitudes being used among the members including technicians, programmers, design, everyone. Agility should be applied to any software process but it is essential to understand how agility is adapted to various different processes' tasks. Agile development processes lowers the development cost compared to conventional software processes and also increases development schedule processes.

Agile software process is mentioned as the manner of addressing a number of assumption about the majority of software projects. Agility principles are important for understanding and working towards and agile software process.

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

4. Business people and developers must work together daily throughout the project.

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

7. Working software is the primary measure of progress.

8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity—the art of maximizing the amount of work not done—is essential.
11. The best architectures, requirements, and designs emerge from self organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

The agile software process is very important to a proper work ethic that is fast and efficient for all parties.

The XP Process is the object-oriented approach as a preferred development paradigm. It is labeled as the Extreme Programming for a reason. Planning the process requires listening to the people and technical members. Design which requires rigorous principles to follow by which are **keeping it simple**. Coding which is expanitory which is preliminary design implemented over time. It's not immediately ported over and needs tweaking and subtle improvements/adjustments over time. Lastly, testing which is crucial to a healthy design. Many errors will be made so adjusting and finding acceptable levels is appropriate at this step. *Scrum* is a development method that are consistent with the agile "manifesto" and are framed into 5 basic activities.

1. Requirements
2. Analysis
3. Design
4. Evolution
5. Delivery.

Each framework allocates different important step to an appropriate manifesto. *Agile Unified Process* adopts a "serial in the large" and "iterative in the small" philosophy. Adopting the previous iteration's activities and working upon them in this one is essential and allows for updated output.

A proper agile philosophy permits a proper engineering ethic for everyone. These tools help facilitate a proper agile working habit for all members which is important for succeeding in the engineering field. Ever since the dawn of humans, we have been all been collaborating using different strengths to solve the same problems.

3 Images

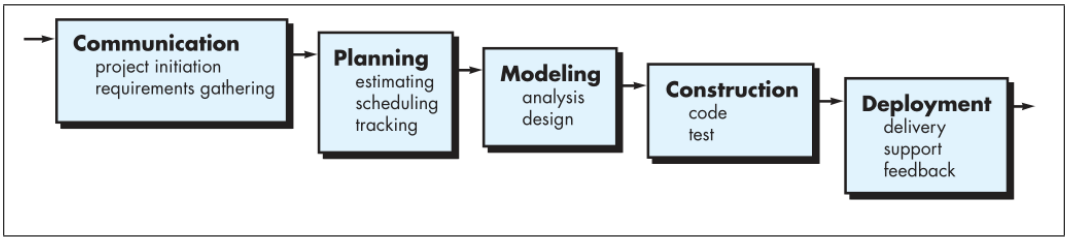

Figure 1: Waterfall Model

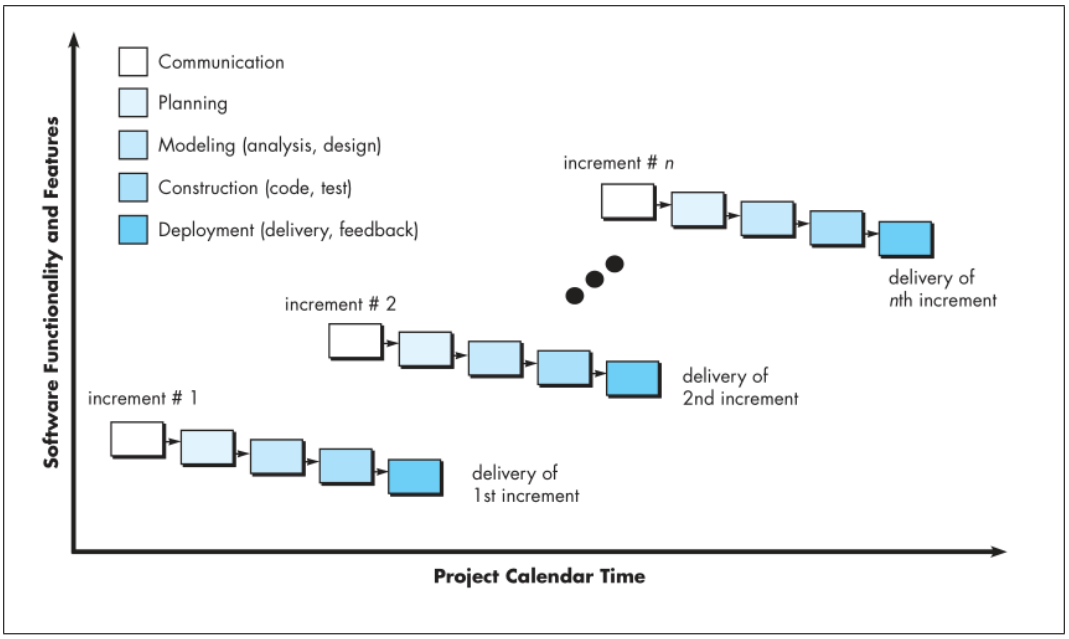

Figure 2: Incremental Model

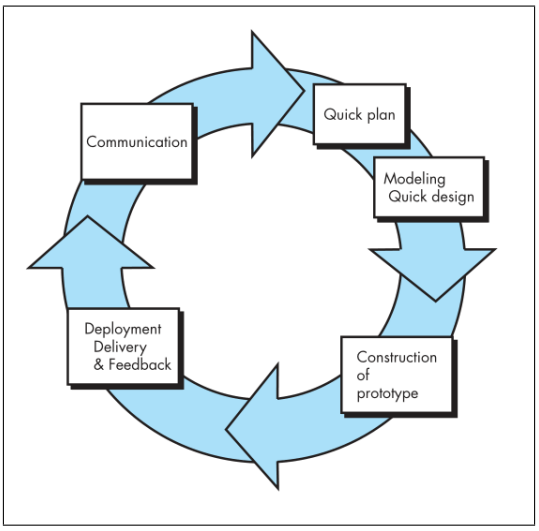

Figure 3: Prototyping

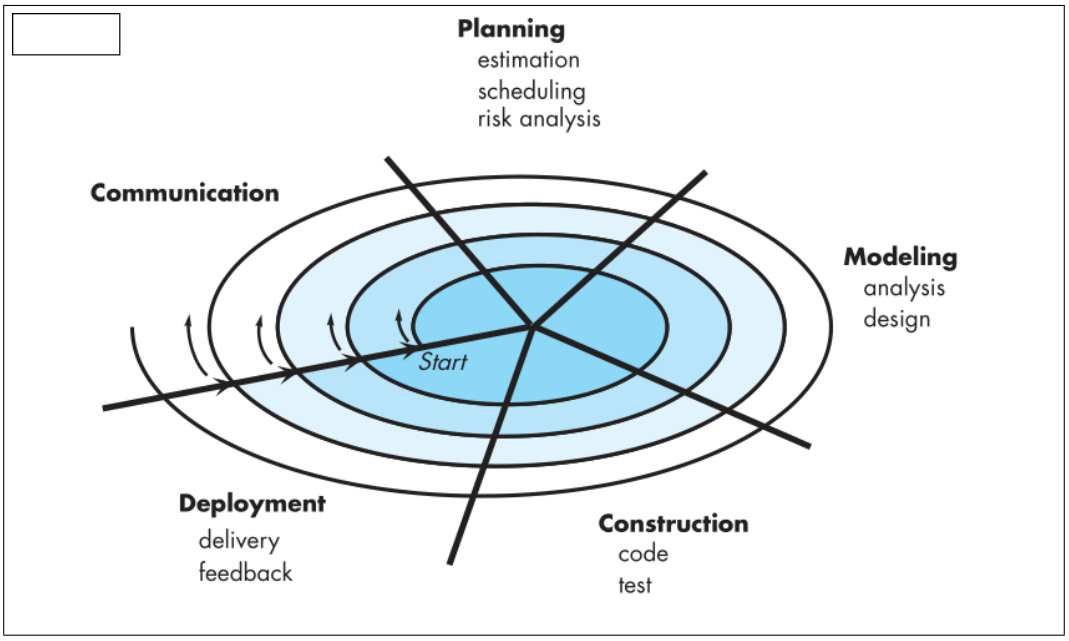

Figure 4: Spiral Model

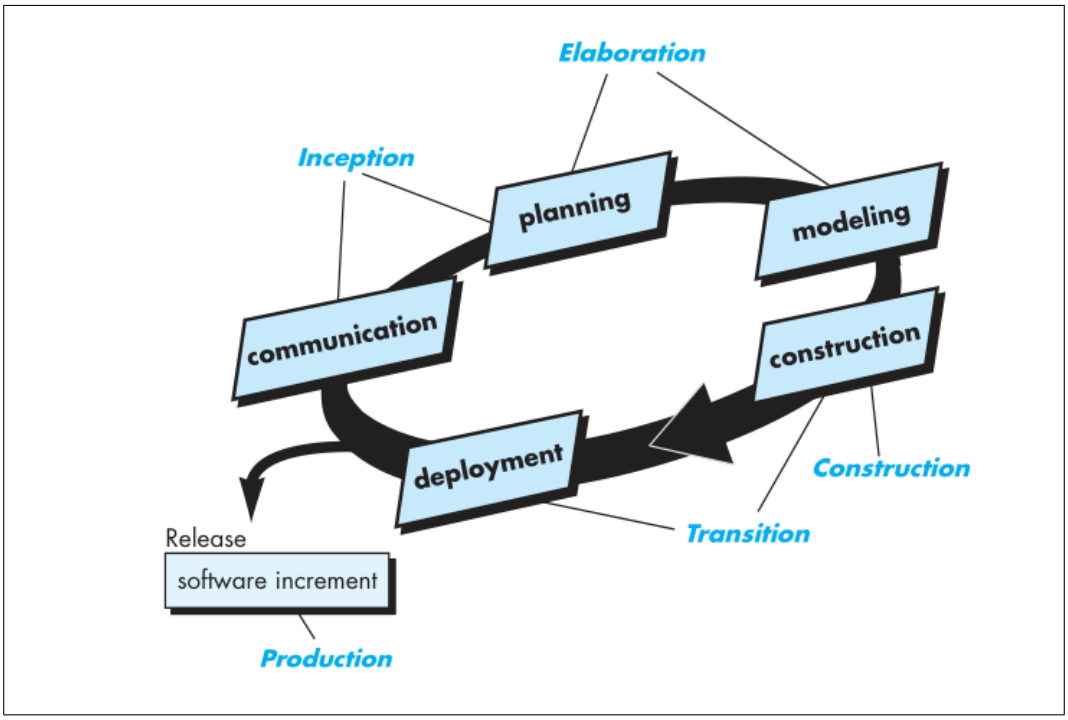

Figure 5: Unified Model