The Software Process

Madiba Hudson-Quansah

Contents

Chapter 1

Process Models

1.1 Generic Process Model

Definition 1.1.1: Process Model

A model/framework in which activities, actions, and tasks reside within and define relationships betwixt them.

Definition 1.1.2: Process Flow

Describes how the framework activities and the actions, and tasks within them are organized with respect to sequence and time.

Definition 1.1.3: Linear Process Flow

The process activities are completed in sequence (one after another) from Communication to Deployment.

Definition 1.1.4: Iterative Process Flow

Like a linear process flow, but one or more activities are repeated before moving to the next activity.

Definition 1.1.5: Evolutionary Process Flow

Executes activities in a circular manner, with each circuit of the five activities leading to a more complete version of the software.

Definition 1.1.6: Parallel Process Flow

Executes one or more activities in parallel

1.1.1 Defining a Framework Activity

Definition 1.1.7: Software Engineering Action

A task that is part of a framework activity

Definition 1.1.8: Task Set

Defines the actual work to be done to accomplish the objectives of a software engineering action.

A framework activity is made up of up several software engineering action, which themselves are made up of task sets. Task sets should be identified based on the best actions that accommodate the needs of the project and characteristics of the team.

1.1.2 Process Patterns

Definition 1.1.9: Process Pattern

Describes a process related problem that is encountered during software engineering work, identifies the context in which the problem occurs, and suggests one or more proven solutions to the problem

There are three types of process patterns:

Task Patterns - Defines a problem associated with a software engineering task or work task, i.e Elicitation, Design, etc. Example RequirementsGathering

Stage Patterns - Defines a problem associated with a framework activity for the process, i.e. Communication, Construction, etc. Incorporate multiple task patterns as they encompass several software actions and work tasks. Example EstablishingCommunication.

Phase Patterns - Defines the sequence of framework activities that occurs within the process, even when the overall flow of activities is iterative in nature, for example SpiralModel and Prototyping.

Ambler proposes the following process pattern template:

Pattern Name - A meaningful name describing the pattern in the context of the software engineering process.

Forces - The environment in which the pattern encountered, and issues that make the problem visible and may affect its solution.

Type - The type of pattern, i.e. Task, Stage, or Phase.

Initial Context - The conditions where the process pattern is applicable. Prior to the application of the pattern

- What organizational or team-related activities have already occurred.
- What is the entry state for the process
- What software engineering / project information already exists.

Problem - The specific problem to be solved by the pattern.

Solution - Describes how to implement the pattern successfully, including how the initial state of the project is modified as a result of the initiation of the pattern, how software engineering / project information is transformed as a result of the pattern.

Resulting Context - Describes the state of the project after the pattern has been applied. Includes:

- What organizational or team-related activities must have occurred.
- What is the exit state for the process.
- What software engineering / project information has been developed.

Related Patterns - Provides a list of related process patterns

Known Uses / Examples - Indicates specific instances where the pattern is applicable.

1.2 Process Assessment and Improvement

A number of different approaches to software process assessment and improvement have been proposed, including:

Standard CMMI Assessment Method for Process Improvement (SCAMPI) - Provides a five step **process assessment** model that incorporates five phases:

- Initiating

- Diagnosing
- Establishing
- Acting
- Learning

CMM-Based Appraisal for Internal Process Improvement (CBA IPI) - Provides a diagnostic technique for **assessing relative maturity of a software organization**, uses SEI CMM as the basis for the assessment.

SPICE (ISO/IEC 15504) - A standard that defines a set of requirements for **software process assessment**. Intended to **assist organizations in developing and objective evaluation on the efficacy of any defined software process**.

ISO 9001:2000 for Software - A generic standard that applies to any organization that wants to improve the **overall quality** of its products, systems or services.

1.3 Prescriptive Process Models

Definition 1.3.1: Prescriptive Process Model

Advocates for an orderly approach to software engineering, these models prescribe a set of process elements (framework activities, software engineering actions, tasks, work products, etc.) for each product.

1.3.1 Waterfall Model

Definition 1.3.2: Waterfall Model

A linear model that suggests a systematic, sequential approach to software development that begins with customer speciation of requirements (Communication) and progresses linearly through all the framework sections, culminating in ongoing support of the completed software.

Good for situations where:

- Requirements are well understood
- Product definition is stable

1.3.2 V Model

Definition 1.3.3: V Model

A model that extends the waterfall model by associating a testing phase for each corresponding state of development.

Good for situations where:

- Critical systems are being developed, i.e. robustness is a key concern
- Requirements are well understood
- Product definition is stable

This model and the waterfall model fail as:

- Real projects rarely follow the sequential flow.
- Requirements are rarely stable and explicitly stated.
- The customer must have patience to wait for the software to be delivered after the end of the whole process

1.3.3 Incremental Process Models

Definition 1.3.4: Incremental Process Model

A model that combines elements of linear and parallel process flows, applying linear sequences in a staggered fashion, producing deliverable increments of the software, similar to the evolutionary process flow.

When the incremental model is used, the first increment is often a core product, i.e. basic requirements are addressed with additional features added in subsequent increments. This allows for evaluation of increments by stakeholders to gain feedback and improve the software.

Good for situations where:

- Staffing is unavailable for a complete implementation by the deadline.
- Requirements are relatively well understood.
- Overall scope of the development effort doesn't fit into a linear process, i.e. large complex projects.
- The customer needs a limited set of software features quickly.

Fails when:

- The customer is unsure of what they want.
- The customer is unable to provide feedback on the software.

1.3.4 Evolutionary Process Models

Definition 1.3.5: Evolutionary Process Model

An iterative model characterized in a manner that allows for the development of increasingly more complete versions of the software.

1.3.4.0.1 Prototyping Model

Definition 1.3.6: Prototyping Model

A model that allows for the development of a prototype, a partial implementation of the software that is used to better understand the requirements of the software. This model begins with Communication, identifying known requirements and areas where further clarification is needed. Then a prototyping iteration is planned, and modelling in the form of quick-design, which focuses on a representation of the parts of the software that will be visible to the end user. Using this design a prototype is constructed, deployed and evaluated by stakeholders. Stakeholder feedback is taken in to improve the prototype in the next prototype iteration.

Prototyping is mostly used as a technique that can be implemented in any other process model, rather than a stand-alone model.

Good for situations where:

- The customer has a general idea of what they want but does not identify detailed requirements for functions and features.
- The developer may be unsure of specific design choices, the efficiency of an algorithm, or the adaptability of an operating system.

Disadvantages include:

- Stakeholders see what appears to be a finished product unaware that the prototype is held together with glue and duct tape.
- Implementation compromises and hacks may be made to meet deadlines which can lead to complacency allowing these hacks to become permanent.

1.3.4.0.2 Spiral Model

Definition 1.3.7: Spiral Model

A model that combines the iterative nature of prototyping with the controlled and systematic nature of the waterfall model, providing the potential for rapid development of increasingly more complete versions of the software.

Using this model, the software is developed in a series of evolutionary releases, with early iterations being closer to a prototype than a final product, and later iterations generating increasingly more complete versions of the software.

Good for:

- Large, complex projects as it allows for the development of a prototype that can be used to identify and mitigate risks.
- Projects where the requirements are not well understood.
- Projects where the requirements are likely to change.
- Projects where the software must be developed quickly.
- A consideration of risk is important.

Fails when:

- Risk analysis is not performed properly thus risk goes unmitigated and unmanaged.
- The customer cannot be convinced that the evolutionary process is controllable.

1.3.4.0.3 Concurrent Models

Definition 1.3.8: Concurrent Model

A software process model that represents iterative and concurrent elements of other process models, allowing different software engineering activities to exist simultaneously but in different states.

The concurrent development model (also called concurrent engineering) enables software teams to work on multiple activities in parallel. Each activity (e.g., modeling, communication, construction) can exist in one of several states: inactive, under development, awaiting changes, or done. The model creates a process network where transitions between states are triggered by events generated at various points in the network.

Good for:

- Projects requiring parallel development activities
- Providing an accurate representation of a project's current state
- Adapting to changes in any phase of development
- Incorporating elements from other process models (spiral, prototyping, etc.)
- Complex systems where different components may be in different stages of development

Fails when:

- Team coordination is poor
- State transitions and events aren't clearly defined
- Project management lacks experience with concurrent development approaches
- Team size is too small to effectively work on parallel activities
- The project is simple enough that sequential development would be more efficient

Chapter 2

Agile Development