Software Process Model

-is a set of related activities that leads to the production of the software. These activities may involve the development of the software from the scratch, or, modifying an existing system.

-In this section a number of general process models are introduced and they are presented from an architectural viewpoint. These models can be used to explain different approaches to software development. They can be considered as process frameworks that may be extended and adapted to create more specific software engineering processes. In this chapter the following process models will be introduced

-A (software/system) *process model* is a description of the sequence of activities carried out in an SE project, and the relative order of these activities.

Software Process

A software process (also knows as software methodology) is a set of related activities that leads to the production of the software. These activities may involve the development of the software from the scratch, or, modifying an existing system.

Any software process must include the following four activities:

In practice, they include sub-activities such as requirements validation, architectural design, unit testing, …etc.

There are also **supporting activities** such as configuration and change management, quality assurance, project management, user experience.

Along with **other activities** aim to **improve** the above activities by introducing new techniques, tools, following the best practice, process standardization (so the diversity of software processes is reduced), etc.

When we talk about a process, we usually talk about the activities in it. However, a process also includes the process description, which includes:

Software process is complex, it relies on making decisions. There's no ideal process and most organizations have developed their own software process.

For example, an organization works on critical systems has a very structured process, while with business systems, with rapidly changing requirements, a less formal, flexible process is likely to be more effective.

Software Process Models

A software process model is a simplified representation of a software process. Each model represents a process from a specific perspective.

We're going to take a quick glance about very general process models. These generic models are abstractions of the process that can be used to explain different approaches to the software development. They can be adapted and extended to create more specific processes.

Some methodologies are sometimes known as ***software development life cycle****(SDLC) methodologies, though this term could also be used more generally to refer to any methodology.*

Waterfall Model

The waterfall model is a sequential approach, where each fundamental activity of a process represented as a separate phase, arranged in linear order.

In the waterfall model, you must plan and schedule all of the activities before starting working on them (plan-driven process).

Plan-driven process is a process where all the activities are planned first, and the progress is measured against the plan. While the agile process, planning is incremental and it's easier to change the process to reflect requirement changes.

The phases of the waterfall model are: **Requirements, Design, Implementation, Testing, and Maintenance.**

The Waterfall Model

The Nature of Waterfall Phases

In principle, the result of each phase is one or more documents that should be approved and the next phase shouldn't be started until the previous phase has completely been finished.

In practice, however, these phases overlap and feed information to each other. For example, during design, problems with requirements can be identified, and during coding, some of the design problems can be found, etc.

The software process therefore is not a simple linear but involves feedback from one phase to another. So, documents produced in each phase may then have to be modified to reflect the changes made.

When To Use?

In principle, the waterfall model should only be applied when requirements are well understood and unlikely to change radically during development as this model has a relatively rigid structure which makes it relatively hard to accommodate change when the process in underway.

Prototyping

A prototype is a version of a system or part of the system that's developed quickly to check the customer's requirements or feasibility of some design decisions.

So, a prototype is useful when a customer or developer is not sure of the requirements, or of algorithms, efficiency, business rules, response time, etc.

In prototyping, the client is involved throughout the development process, which increases the likelihood of client acceptance of the final implementation.

While some prototypes are developed with the expectation that they will be discarded, it is possible in some cases to evolve from prototype to working system.

A software prototype can be used:

[1] In the **requirements engineering**, a prototype can help with the elicitation and validation of system requirements.

It allows the users to experiment with the system, and so, refine the requirements. They may get new ideas for requirements, and find areas of strength and weakness in the software.

Furthermore, as the prototype is developed, it may reveal errors and in the requirements. The specification maybe then modified to reflect the changes.

[2] In the **system design**, a prototype can help to carry out deign experiments to check the feasibility of a proposed design.

For example, a database design may be prototype-d and tested to check it supports efficient data access for the most common user queries.

The process of prototype development

The phases of a prototype are:

Prototyping is not a standalone, complete development methodology, but rather an approach to be used in the context of a full methodology (such as incremental, spiral, etc).

Incremental Development

Incremental development is based on the idea of developing an initial implementation, exposing this to user feedback, and evolving it through several versions until an acceptable system has been developed.

The activities of a process are not separated but interleaved with feedback involved across those activities.

The Incremental Development Model

Each system increment reflects a piece of the functionality that is needed by the customer. Generally, the early increments of the system should include the most important or most urgently required functionality.

This means that the customer can evaluate the system at early stage in the development to see if it delivers what’s required. If not, then only the current increment has to be changed and, possibly, new functionality defined for later increments.

Incremental Vs Waterfall Model

Incremental software development is better than a waterfall approach for most business, e-commerce, and personal systems.

By developing the software incrementally, it is cheaper and easier to make changes in the software as it is being developed.

Compared to the waterfall model, incremental development has three important benefits:

It can be a plan-driven or agile, or both

Incremental development is one of the most common approaches. This approach can be either a plan-driven or agile, or both.

In a plan-driven approach, the system increments are identified in advance, but, in the agile approach, only the early increments are identified and the development of later increments depends on the progress and customer priorities.

It's not a problem-free

But, it's not a problem-free …

Spiral Model

The spiral model is a risk-driven where the process is represented as spiral rather than a sequence of activities.

It was designed to include the best features from the waterfall and prototyping models, and introduces a new component; risk-assessment.

Each loop (from *review* till *service* — see figure below) in the spiral represents a phase. Thus the first loop might be concerned with system feasibility, the next loop might be concerned with the requirements definition, the next loop with system design, and so on.

The spiral model

Each loop in the spiral is split into four sectors:

Spiral model has been very influential in helping people think about iteration in software processes and introducing the risk-driven approach to development. In practice, however, the model is rarely used.

Iterative Development

Iterative development model aims to develop a system through building small portions of all the features, across all components.

We build a product which meets the initial scope and release it quickly for customer feedback. An early version with limited features important to establish market and get customer feedback.

In each increment, a slice of system features is delivered, passing through the requirements till the deployment.

The phases of iterative development

The phases of iterative development are:

All the phases will be done once, while the construction phase will be incrementally visited for each increment; for each slice of system features.

Agile

Agility is flexibility, it is a state of dynamic, adapted to the specific circumstances.

The agile methods refers to a group of software development models based on the incremental and iterative approach, in which the increments are small and typically, new releases of the system are created and made available to customers every few weeks.

The principles of agile methods

They involve customers in the development process to propose requirements changes. They minimize documentation by using informal communications rather than formal meetings with written documents.

They are best suited for application where the requirements change rapidly during the development process.

There are a number of different agile methods available such as: Scrum, Crystal, Agile Modeling (AM), Extreme Programming (XP), etc.

Increment Vs Iterative Vs Agile

You might be asking about the difference between incremental, iterative and agile models.

Each increment in the incremental approach builds a complete feature of the software, while in iterative, it builds small portions of all the features.

An agile approach combines the incremental and iterative approach by building a small portion of each feature, one by one, and then both gradually adding features and increasing their completeness.

Reuse-oriented Software Engineering

It’s attempting to reuse an existing design or code (probably also tested) that’s similar to what’s required. It’s then modified, and incorporated to the new system.

The Reuse-oriented software engineering model

Although the initial “requirements specification” phase and the “validation ” phase are comparable with other software processes, the intermediate phases in a reuse-oriented process are different. These phases are:

There are basically three types of software components that can be used in a reuse-oriented process:

It’s has an obvious advantage, But!

Reuse-oriented software engineering has an obvious advantage of reducing the amount of software to be developed and therefore reduced cost and risks, and usually leads to faster delivery.

However, requirements compromises can’t be avoided, which may lead to a system that does not meet the real needs of users.

Furthermore, some control over the system evolution might also be lost as new versions of the reusable components are not under the control of the organization using them.

Summary

Waterfall

It’s useful when the requirements are clear, or following a very structured process as in critical systems which needs a detailed, precise, and accurate documents describes the system to be produced.

Not good when requirements are ambiguous, and doesn’t support frequent interaction with the customers for feedback and proposing changes. It’s not suitable for large projects that might take long time to be developed and delivered.

Prototype

Again, it’s an early sample, or release of a product built to test a concept or to act as a thing to be replicated or learned from.

This is very useful when requirements aren’t clear, and the interactions with the customer and experimenting an initial version of the software results in high satisfaction and a clearance of what to be implemented.

It’s downsides are, good tools need to be acquired for quick development (like coding) in order to complete a prototype. In addition, the costs for for training the development team on prototyping may be high.

Incremental & Iterative

They’re suited for large projects, less expensive to the change of requirements as they support customer interactions with each increment.

Initial versions of the software are produced early, which facilitates customer evaluation and feedback.

They don’t fit into small projects, or projects that waterfall are best suited for; A structured process with a detailed, and accurate description of the system.

Spiral

It’s good for high risky or large projects where the requirements are ambiguous. The risks might be due to cost, schedule, performance, user interfaces, etc.

Risk analysis requires highly specific expertise, and project’s success is highly dependent on the risk analysis phase. It doesn’t work well for smaller projects.

Agile

It suits small-medium size project, with rapidly changes in the requirements as customer is involved during each phase.

Very limited planning is required to get started with the project. It helps the company in saving time and money (as result of customer physical interaction in each phase). The daily meetings make it possible to measure productivity.

Difficult to scale up to large projects where documentation is essential. A highly skilled team is also needed.

If team members aren’t committed, the project will either never complete or fail. And there’s always a limitation in time, like in increments, meetings, etc.

Process Activities

The four basic process activities of specification, development, validation, and evolution are organized differently in different development processes.

In the waterfall model, they are organized in sequence, while in incremental development they are interleaved. How these activities are performed might depend on the type of software, people involved in development, etc.