Specialized Process Model in Software Engineering

Specialized Process Model

- Specialized process models take on many of the characteristics of one or more of the traditional models.
- However, these models tend to be applied when a specialized or narrowly defined software engineering approach is chosen.

Specialized Process Model

There are 3 types of specialized process models:

1. **Component Based Development**
2. **Formal Methods Model**
3. **Aspect Oriented Software development**

1. Component Based Development

- Commercial off-the-shelf (COTS) software components.
- Developed by vendors who offer them as products, provide targeted functionality with well-defined interfaces that enable the component to be integrated into the software that is to be built.
- **Incorporates many of the characteristics of the spiral model**.
- It is evolutionary in nature, demanding an iterative approach to the creation of software.
- **constructs applications from prepackaged software component**.
- Modeling and construction activities begin with the identification of candidate components.
- **These components can be designed as either conventional software modules or object-oriented classes or packages of classes**.

1. Component Based Development

component-based development model incorporates the following steps:

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

1. Component Based Development

The component-based development model leads to software reuse, and reusability provides software engineers with several measurable benefits.

Software engineering team can achieve a reduction in development cycle time as well as a reduction in project cost if component reuse becomes part of your culture.

2. Formal Methods Model

- The formal methods model encompasses a set of activities that leads to formal mathematical specification of computer software.
- Formal methods enable to specify, develop, and verify a computer-based system by applying a rigorous, mathematical notation.
- A variation on this approach, called cleanroom software engineering is currently applied by some software development organizations
- Ambiguity, incompleteness, and inconsistency can be discovered and corrected more easily, through the application of mathematical analysis.
- When formal methods are used during design, they serve as a basis for program verification and therefore enable you to discover and correct errors that might otherwise go undetected.

2. Formal Methods Model

Disadvantages:

- The development of formal models is quite time consuming and expensive.
- Because few software developers have the necessary background to apply formal methods, extensive training is required.
- It is difficult to use the models as a communication mechanism for technically unsophisticated customers.

3. Aspect Oriented Software Development

- Regardless of the software process that is chosen, the builders of complex software invariably implement a set of localized features, functions, and information content.
- **These localized software characteristics are modeled as components and then constructed within the context of a system architecture**.
- certain concerns span the entire architecture.
- Some concerns are high-level properties of a system, Other concerns affect functions, while others are systemic.

3. Aspect Oriented Software Development

- When concerns cut across multiple system functions, features, and information, they are often referred to as crosscutting concerns.

- Aspectual requirements define those crosscutting concerns that have an impact across the software architecture.

- Aspect-oriented software development (AOSD), often referred to as aspect-oriented programming (AOP), is a relatively new software engineering paradigm that provides **a process and methodological approach for defining, specifying, designing, and constructing aspects**.

3. Aspect Oriented Software Development

- Adopt characteristics of both evolutionary and concurrent process models.
- The evolutionary model is appropriate as aspects are identified and then constructed. The parallel nature of concurrent development is essential because aspects are engineered independently of localized software components and yet, aspects have a direct impact on these components.
- It is essential to instantiate asynchronous communication between the software process activities applied to the engineering and construction of aspects and components.