Different Life Cycle Model Summary

Introduction

It is well-known that without a well-defined project lifecycle, it is difficult to successfully build a quality project. There are various software development lifecycle (SDLC) models that can be used in our projects. Generally, these models fall into two categories: predictive lifecycles and adaptive lifecycles. In a predictive lifecycle, the design and implementation of system functionality do not begin until the requirements have been thoroughly identified and documented. In contrast, adaptive lifecycles do not typically follow a rigid set of steps or phases that must be completed in sequence for a project to progress. These two types of SDLCs can give rise to many other models, such as the waterfall model, parallel development model, iterative lifecycle, evolutionary prototyping, throw-away prototyping, and agile development approaches. This report will introduce the characteristics of different SDLC.

Life Cycle Models

Waterfall Model

In the waterfall model, the SDLC phases are completed in sequence to progress toward a complete system.

Characteristics: Linear and sequential (each phase must be completed before the next begins)

Advantages:

1. Simple to understand and manage;
2. well-suited for projects with well-defined requirements.

Disadvantages:

1. Inflexible to changes;
2. not ideal for complex or long-term projects.

Risks: High risk if requirements change late in the process.

Handling Changes: Changes are difficult to accommodate; may result in significant rework.

Parallel Development Model

Characteristics: Divides the project into sub-projects that can be developed simultaneously.

Advantages: Shortens development time; efficient for large systems.

Disadvantages: Integration challenges; complex to manage.

Risks: Integration risks; potential for sub-project delays.

Handling Changes: More flexible than Waterfall but still challenging to manage mid-development changes.

Iterative Life Cycle

Characteristics: Develops the system in iterations, refining the product with each cycle.

Advantages: Allows for feedback and improvements in each iteration; reduces risks.

Disadvantages: Requires careful planning; can be time-consuming.

Risks: Scope creep; potential for budget overruns.

Handling Changes: Changes can be incorporated in subsequent iterations, improving client satisfaction.

Evolutionary Prototyping

Characteristics: Develops prototypes iteratively until the final system is built.

Advantages: Provides early visibility; allows for user feedback.

Disadvantages: May result in incomplete systems; high user expectations.

Risks: Users may mistake the prototype for the final product.

Handling Changes: Accommodates changes well through iterative prototyping, enhancing client satisfaction.

Throw-away Prototyping

Characteristics: Builds a prototype to understand requirements, which is then discarded.

Advantages: Helps clarify requirements; reduces misunderstandings.

Disadvantages: Wasted effort on prototypes; not suitable for complex systems.

Risks: Potential loss of focus on final product requirements.

Handling Changes: Allows for changes early in the process, reducing risks later on.

Agile Development

Characteristics: Focuses on iterative development with close collaboration with clients.

Advantages: Highly flexible; responsive to changes; high client satisfaction.

Disadvantages: Requires experienced team members; can lead to scope creep.

Risks: Lack of documentation; potential for project delays.

Handling Changes: Agile is designed to accommodate frequent changes, ensuring high client satisfaction and adaptability to evolving requirements.

Summary

Every software development life cycle (SDLC) has its own characteristics and is suited to specific scenarios. In modern society, agile development is the most popular SDLC because it allows for rapid client feedback. This approach helps the development team ensure they are on the right path throughout the project.