REPORT

Analysis of SDLC Models for Embedded Systems

Software Development Life Cycle (SDLC) model is defined as a frame work used to describe the activities to be performed at each stage of a software development project.Different SDLC models include Waterfall model, Vmodel, Iterative model, Incremental model, Spiral model, Agile Methods.

AGILE METHOD

Agile method for software development is an evolutionary approach where requirements and solutions evolve through collaboration between selforganizing, cross-functional teams within an effective governance framework to provide solutions in cost effective and timely manner which meets the changing needs of stakeholder.

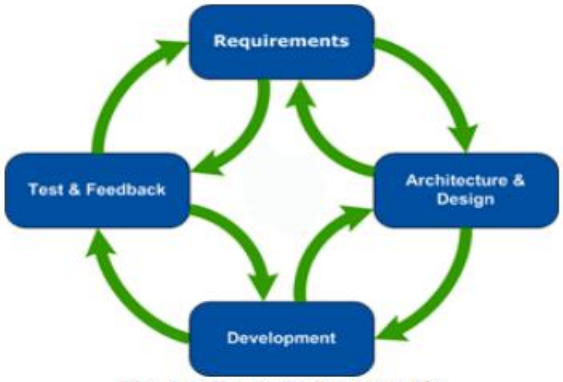

Fig. 1 Agile method process [8]

The major agile development methods include Extreme Programming (XP), Scrum, Dynamic Systems Development Method (DSDM), Adaptive Software Development (ASD), Crystal and Feature Driven Development (FDD).

XP (Extreme Programming):

Phases: Exploration, Planning, Iterations to Release, Productionizing, Maintenance, and Death.

Focus on iterative development and frequent releases.

Scrum:

Phases: Pre-game, Development, Post-game.

Emphasizes flexibility, adaptability, and productivity.

Involves changing requirements, time frames, resources, and technology.

Crystal Method:

Different colors represent methodology heaviness based on project size and criticality:

Crystal Clear: Small projects.

Crystal Orange: Medium-sized projects.

Crystal Red: Larger projects.

FDD (Feature-Driven Development):

Focuses on design and building phases.

Emphasizes quality through frequent, tangible deliveries and project monitoring.

DSDM (Dynamic Systems Development Method):

Fixes time and resources, adjusts functionality.

ASD (Adaptive Software Development):

Phases: Speculate (planning), Collaborate (teamwork), Learn (acknowledge and react to mistakes).

Provides a framework to prevent chaos while fostering emergence and creativity.

Fig. 2 Agile methodologies [9]

SDLC SELECTION FACTOR

Factors for Selection: Type of information system, software size and complexity, system modularity, module integrity, quality standards, cumulative project cost, resource availability.

Model Criteria:

Waterfall: Suitable for complex systems, strong management, good documentation, and component reusability; not ideal for unclear user requirements.

V-shaped: Similar to Waterfall; excellent management and documentation but poor cost limitations.

Spiral: Good for unclear requirements and unfamiliar technology; high costs and skill limitations.

Incremental/Iterative: Great for complex systems, unclear requirements, unfamiliar technology; excellent management, scheduling, and stakeholder visibility.

Agile: Strong short-term scheduling and cost management, better reliability and stakeholder visibility; limited skills, documentation, and component reusability.

APPLICATION OF AGILE METHODS

Flexibility and Adaptability: Agile promotes a flexible working environment for self-organized teams to maximize customer value.

Aspects of Agile in Embedded Systems:

Rapid Cycles: Regular cycles creating executable deliverables; embedded systems require granular feature sets.

Focus on Coding: Emphasizes coding over planning/documentation; embedded systems need extensive documentation.

Continuous Refactoring: Improves code quality; beneficial if initial architecture limits large-scale refactoring.

Team Communication: Extensive communication minimizes hardware shortages, aids documentation.

Limited Customer Interaction: Customer proxies can help align development with customer needs.

Test-Driven Development: Early testing reduces defects, unit-level debugging easier; system-level debugging complex.

Continuous Planning: Helps manage feature sets, staffing, delivery dates, and performance optimizations.

LEAN AGILE APPROACH

Principles: Minimizes waste, combines Lean with Agile for better embedded software development.

Core Tenets:

Eliminate Waste: Focus on removing unnecessary elements.

Empower the Team: Build reliable, capable teams.

Amplify Learning: Foster learning activities.

Build Integrity: Ensure team integration and project integrity.

Fast Delivery: Develop projects quickly to meet schedules.

Holistic View: Integrate steps and systems for comprehensive project development.

User Stories: Capture requirements to prioritize features and deliver business value.

Software Development Life Cycle early phases and quality metrics: A Systematic Literature Review

The paper systematically examines the early phases of the Software Development Life Cycle (SDLC) and emphasizes the importance of quality metrics in improving software processes. Through a thorough review of over 200 publications, it identifies that early defect detection is critical to enhancing team efficiency and reducing waste. The key early phases of SDLC discussed include requirement gathering, system design, and architecture design. The paper underscores that focusing on these phases and integrating appropriate quality metrics can lead to better project outcomes.

It highlights the evolving methodologies in software development, with a particular focus on Agile methodologies. Agile's iterative and flexible nature is contrasted with traditional methods, showing its advantages in adaptability and response to changing requirements. However, the paper notes that Agile methods require the careful incorporation of quality metrics to maintain the reliability and quality of the software, despite the rapid development cycles.

The review aims to provide a framework for categorizing metrics based on different process methodologies. This framework helps in effectively sorting and applying

metrics according to the specific stages of the development process. By defining these metrics clearly, the paper offers a structured approach to assess development stages and establish a base set of metrics for evaluating software processes. This approach is intended to maintain high quality standards and ensure that development projects meet their intended goals efficiently.

Overall, the paper provides a comprehensive examination of the early phases of the SDLC and the role of quality metrics. It emphasizes the necessity for detailed planning and robust metrics, particularly when using Agile methodologies, to guide the development process and enhance the quality of software products. These insights are valuable for optimizing development processes and achieving better outcomes in software projects.