Volume 5, Issue 5, May 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Comparative Study of Two Software Development Approaches: Traditional and Object Oriented

Shruti Pustake[1], Rajesh Shah[2]
[1]Assistant Professor, [2]Professor
[1, 2] Computer Science & Electronics, Christian Eminent College,
Indore, M.P., India

Abstract: Software projects are developed using a common methodology called System Development Life Cycle. The traditional approach has distinct models that deal with various types of projects, most common is the waterfall model. The other models include Spiral, Iterative, V-shaped etc. These models go through various phases for software development.
Another approach to SDLC is the object-oriented methodology. The object-oriented approach to system development also consists of various phases but it is iterative and incremental in nature. The object oriented analysis and design technique builds, manage and assemble objects into a software system.
This paper discusses the traditional and object oriented approach in software engineering.

Keywords: SDLC, Object Oriented, Analysis, Design, Implementation, UML, SRS.

I. INTRODUCTION

Software development methodologies has been combination of various phases that comprises investigation, analysis, planning, scheduling, activities, roles, tools and techniques, coding, testing, documentation, implementation, training etc. It basically consists of a set of modelling conventions, modelling language, process, that can give guidance to order the activities and offer criteria for monitoring and measuring project activities. A good software development methodology should address at-least the following issues: Planning, Scheduling, Resourcing, Activities involved, Roles, Artifacts and Training.
In traditional approach, the software development life cycle moves from one phase to another. It has a systematic and semantic way of doing things. Even the smallest software project can be developed keeping a methodology in mind such as waterfall model. In this approach a software project goes through a number of phases common to every development regardless of model chosen, starting with information gathering or requirement gathering and ending with maintenance.
Object oriented methodology emerged as a revolution some two decades ago[4]. It is versatile in nature and adapts to the rapidly changing and growing software industry. Object-Oriented techniques are evolved from the semi-structured, partly object oriented to the Unified model, integrated and agile techniques. Object-Oriented Analysis and Design techniques continues to evolve. In this approach, a system is viewed as a set of objects. Object-Oriented methodologies for software development are aimed at viewing, modelling and implementing the system as a group of interacting objects, using the specialized modelling languages, activities and techniques needed to address the specific issues of the object oriented paradigm.

II. AN OVERVIEW OF TRADITIONAL APPROACH:

The most common model used in traditional approach to SDLC is Waterfall model. This model adopts systematic step-by-step approach to different phases and activities involved in SDLC. The activities of a particular phase must complete before moving onto the next phase in the cycle. The core of this approach is the process model that depicts different processes that are the integral part of a software. This process model is presented using Data Flow diagrams and associated tools like Data Dictionary that contains information about the system components that are required to be designed and finally built in to a software system.
SDLC phases of traditional approach models are:

(A) System Analysis:
(i) Preliminary Investigation and Information Gathering: It is the first step in SDLC that identifies the need of the software. Key strategies for eliciting information gathering regarding the user's requirements: (1) asking (2) getting information from the existing information system and (3) prototyping. Various tools are adopted for requirements gathering such as review of literature, onsite observation, interviews etc.
(ii) Structured Analysis: A set of techniques and tools are used by analyst to develop system specifications. These tools are Data Flow Diagram, Data Dictionary, Decision trees and Decision tables etc. Here a new document known as SRS (System Requirement Specification) is built that provides the basis for design and implementation of the software project.

(iii) Feasibility Analysis: Feasibility of the candidate systems is analysed to select the best system that meets the requirement specifications as documented in SRS. Cost-Benefit analysis is performed which identifies the costs and benefits of the given system and after evaluation, the results are interpreted for actions.

(B) System Design:

This phase emphasizes more on translating the performance requirements, as documented in analysis phase, into design specifications. The design goes through logical and physical stages of development.

The logical flow of the system is designed using DFDs and hence the boundaries of the system are defined. The inputs (source), output (destination), databases (data stores) and procedures (data flows) are described that meets the user requirements.

The physical design phase deals with input/output media, database design, backup procedures, designing physical flow of data through the system.

(C) System Implementation:

(i) Coding & Testing: This phase involves actual writing of code for different modules or procedures of the system. These modules collaborate to form the whole system.

After coding, it is needed to test the system against the system requirements as specified in the SRS. This is performed in the testing phase. The system is tested using various testing methods and tools to assure that the system fulfils the requirements.

(ii) Deployment and training: In this phase the end users get the hardware and software of the designed system along with manuals and training.

(iii) Maintenance: Maintenance covers a wide range of activities such as improvement in coding, adding patches of code and correcting design errors, updating documentation and upgrading user support. Maintenance comprise of repairing performance failures or making changes due to previous false assumptions. This phase deals with enhancing the performance of the system in response of user's additional or changing needs.

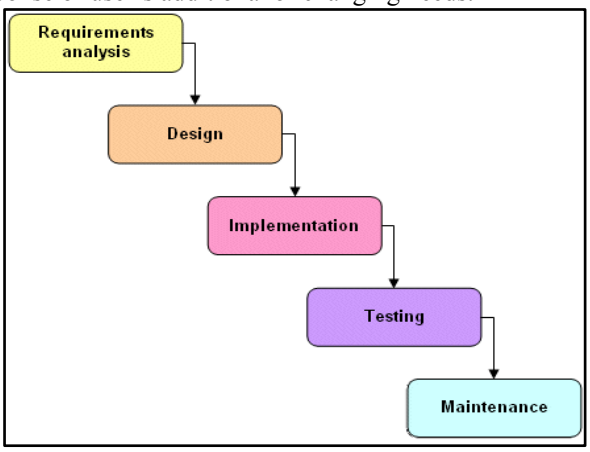

Figure 1: Waterfall Model

III. AN OVERVIEW OF OBJECT-ORIENTED APPROACH

The SDLC is viewed as consisting of several increments or phases: inception, elaboration, construction and transition [5]. In each increment or phase, the developers move through the activities of gathering requirements, analyzing, designing, implementing and testing the system.

The object oriented methodology addresses:

- The concept behind each of the phases of SDLC.
- The individual activities and dataflow within each phase (Functional model).
- The diagrams, documentation and coding.
- The need to model static structure (Object model) and dynamic behaviour (Dynamic model).

SDLC phases of Object Oriented approach:

(A) Analysis:

This phase is aimed at analysing and defining the system to be built. Two models are required to be developed in this phase: Requirements model and Analysis model[1].

In the first model, a conceptual picture of the system using objects of problem domain and specific interface descriptions of the system is developed.

The second model is an architectural view (model) used for analysis of robustness. This model gives a conceptual configuration of the system, consisting of various classes like active controllers, entities, and interfaces. The purpose of this model is to create a base for design and implementation. Each object has its own purpose and together they work to provide total functionality as specified in requirement model. The objects may be combined to form subsystems, which in turn are combined to form the system.

(B) Design and Implementation:

In this phase system is built according to the requirement and analysis models designed in analysis phase.

The design model is the refinement and formalization of the Analysis model. The implementation model depicts the actual code of the system.

(C) UML:

The Object Oriented approach uses some diagramming techniques known as UML (Unified Modeling Language). The UML may be used to visualize, specify, construct, and document the artifacts of a software-intensive system[6]. The focus is on the functional, static and dynamic view of the system.

(1) Use Case Diagram: These diagrams depict the functional view that describes the external behaviour of the system from the user point of view. It is a static description of how the system is used by its users or by other systems. It displays a set of use cases and actors and their relationships.

A use case describes a set of sequences, in which each sequence represents the interaction of the things outside the system (its actors) with the system itself (and its key abstractions.) A use case represents a functional requirement of the system as whole.

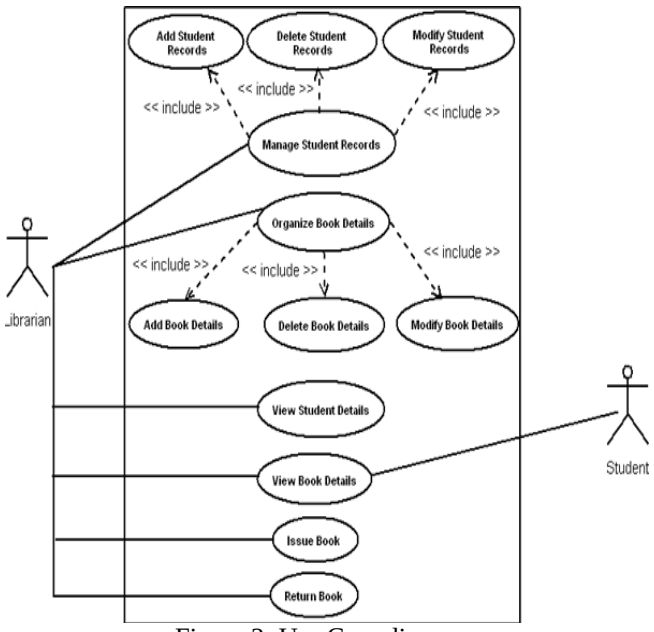

Figure 2: Use Case diagram

(2) Class Diagrams: Class diagrams are the most common diagram found in modelling object-oriented systems. A class diagram displays classes, interfaces and their relationships.

Class diagrams can be drawn at analysis and design levels. The design level class diagrams are more detailed than analysis level class diagrams.

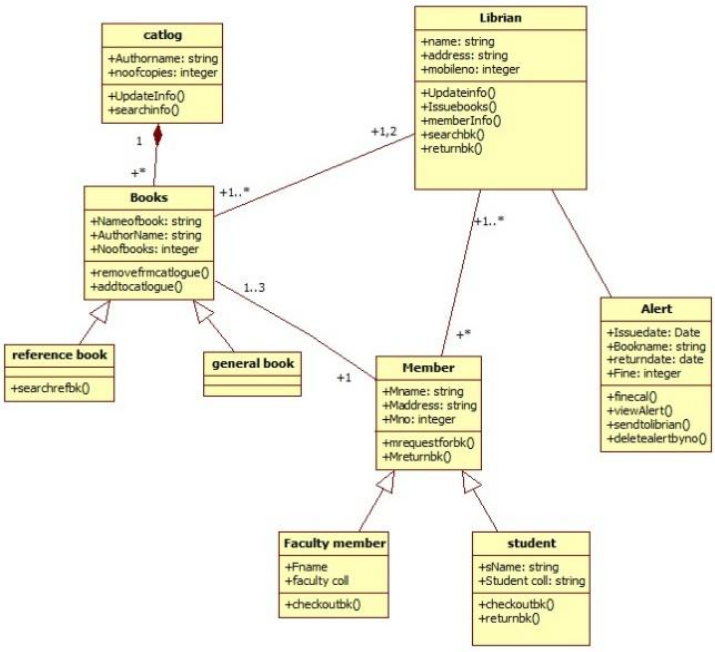

Figure 3: Class diagram

(3) Component Diagram: They show the set of components and their relations. They illustrate the static implementation view of the system.

(4) Deployment Diagram: A set of nodes and their relationships are shown in a deployment diagram.

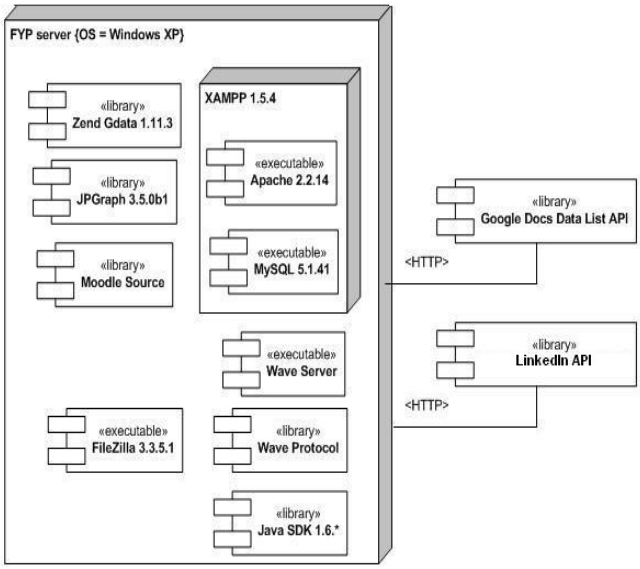

Figure 4: Deployment diagram

(5) Object Diagram: Objects and their relationships are shown in an object diagram. It addresses the static design view from the perspective of real cases.

(6) Sequence Diagram: This diagram emphasizes on time-ordering of messages.

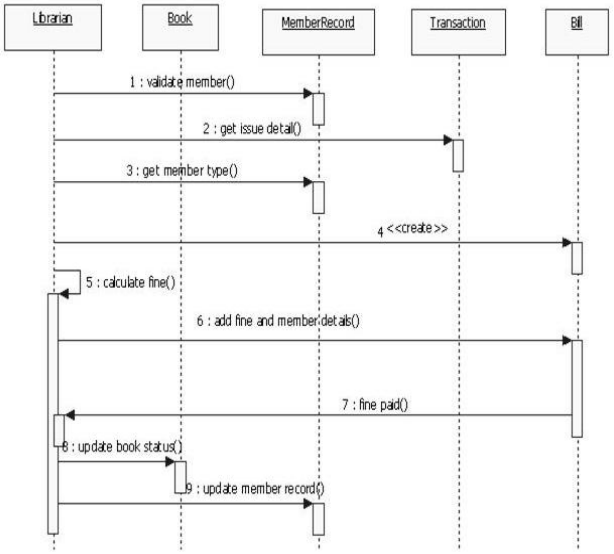

Figure 5: Sequence diagram

(7) Collaboration Diagram: It is an interaction diagram that emphasizes on the structural organization of the objects that send and receive messages. It shows a set of objects, link among those objects, and messages sent and received by those objects.

(8) State Diagram: A state chart diagram shows a state machine, consisting of states, transitions, events, and activities. It illustrates the dynamic view of the system.

All diagrams are refined iteratively until the requirements of the information system are fully defined. Finally the information system is developed through a combination of traditional relational database and object oriented database.

(D) Testing:

Testing is done to check the validity, and verify the functioning of the system as documented in requirement specifications (SRS).

Unit testing is performed to test a specific unit, where a unit can be of varying size from a class up to an entire subsystem. White-box testing is performed when the tester has access to the internal data structures and algorithms including the code that implement these like API testing, Code coverage, Fault injection methods etc.

Black-box testing treats the software as a "black box"—without any knowledge of internal implementation. Black-box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, model-based testing etc.

IV. CONCLUSION

This paper has discussed two approaches to software development. Both share some common phases like analysis, design, implementation and testing. They have their own approach to deal within these phases.

This paper compares both traditional and Object-Oriented approaches to analyse, design and implement the software systems. By doing this we come to conclusion that although the phases in both the approaches are similar but the way to realize these phases in Object Oriented approach is much more different and efficient. In Object oriented approach more emphasis is given on behaviour and functionality of objects of given domain.

A software project can be developed using any of the two reviewed approaches but Object Oriented gives more efficient results.

REFERENCES

[1] Nabil Mohammed Ali Munassar, Dr. A.Govardhan, "Comparison between traditional approach and object-oriented approach in software engineering development", IJACSA, Vol.2,No.6,2011

[2] Mohammad A. Rob, "Issues of structured vs. Object-oriented methodology of systems analysis and design", Issues in Information Systems, Volume V, No1, 2004.

[3] Elias M. Awad, "Systems Analysis and Design",Galgotia Prakashan,Second Edition,1995.

[4] Usman Ali Khan, I.A. Al-Biwedi, Kunal Gupta, "Object-Oriented Software Methodologies : Roadmap to the Future",IJCSI, Vol 8,No 2, 2011.

[5] Dennis, A., Wixon, B.H. and Tegarden, D.(2002), System Analysis and Design: An Object-Oriented Approach,John Wiley & Sons, New York.

[6] Grady Booch, James Rumbaugh, Ivar Jacobson, " The Unified Modeling Language User Guide", Pearson Education.

[7] Grady Booch, "Object-Oriented Analysis & Design with applications", Pearson Education, Second Edition.

[8] Ugrasen Suman, "Software Engineering Concepts and Practices", CENGAGE Learning.

