

## Tabular Guidelines for System Development Methodologies

AJAY JANGRA<sup>1</sup> DEEPA SHARMA<sup>2</sup> JASBIR SINGH<sup>3</sup> and PRIYANKA<sup>4</sup>

<sup>1,2,3</sup> CSE Deptt. U.I.E.T. Kurukshetra University, Kurukshetra (INDIA)

<sup>4</sup>ECE deptt. Kurukshetra Institute of Technology and Management, Kurukshetra (INDIA)  
er\_jangra@yahoo.co.in, reach4deepa@gmail.com, jasbir.singh02@gmail.com,  
priyanka.jangra@gmail.com

(Acceptance Date 8th June, 2010)

### Abstract

Software Engineering works as systematic, disciplined, quantifiable tool to the development, operation, and maintenance of software. In software development life cycle many models have been developed to evaluate and improve their capabilities. This paper proposed two new D-tables (decision tables *i.e.* D<sub>1</sub>, D<sub>2</sub>) which provide necessary guidelines to the developer/ organisation on decision making regarding selecting System Development methodology (SDM) by "Comparing traditional and object oriented SDM". This work is novel in the sense it comprises traditional and object oriented SDM models, analyzing with respect to pros. & cons. and guide the developer to select particular SDM approach. The required result may depend on organization's decision that how well they create software according to how they define and execute their processes. In this paper, we focus on different approaches of SDM and try to find out the suitability of different SDM approaches in software.

*Key words:* software development methodology (SDM), SDLC, Object-oriented system design, user, developer, SRS.

### Objectives :

- The aim of this paper is to provide a helpful knowledge about the flexibility of both SDM to know that which approach is useful at the SRS time for the management.
  - To know the frequency of updation and their effect on software project management.
  - To provide guidelines for selecting SDM
- from user, developer point of view.
  - Study gains in different phases of SDM to know which approach is more advantageous.

### Introduction

Software process means a set of activities required to produce a software system.

Software engineering processes are composed of many activities like: *Requirement analysis*: -It is the first task which may require skill and experience to recognize incomplete, contradictory requirement. *Specification*: It describes the software to be written in mathematically rigorous way. *Software Architecture*: It ensures that system will meet the requirement and future requirement can be addressed. *Implementation*: Structuring a design to code is implementation. *Testing*: It is the major part which gives us security that product meets to requirements<sup>1,4,6</sup>.

#### *SDLC Methodology Steps :*

Objectives of systems development are to create quality information systems. Although SDLC has different forms and models, it follows certain steps which could have the same name in one methodology but they are treated in a different manner. Here are some of the basic steps (whatever the model is), development should go through these stages as these decides the outcome of the any SDLC model. *Planning* – Everything starts with a concept “What do do? & how to do?” This is done in planning. *Design* –it is time to design a rough plan to decide the initial specifications of the software to be created. *Implementation* – is the time to work on the plan. Some developers follow the standard plan of developing software will work on the plan and present them for approval. *Testing* – aimed in cleaning the software of all the bugs altogether. *Acceptance* – When the software is released to be used, acceptance means the software is implemented as an added tool. *Maintenance* – All SDLC models include maintenance because there is no surety that software will work perfectly when the software is implemented

in public. *Disposal* –when software is being old-fashioned, it is not just all deletion of files. Project managers should determine which file to protect and dispose. These seven steps could be even bigger or expanded depending on the SDLC model that has been followed by different developers<sup>2,6,7</sup>.

#### *Literature Survey :*

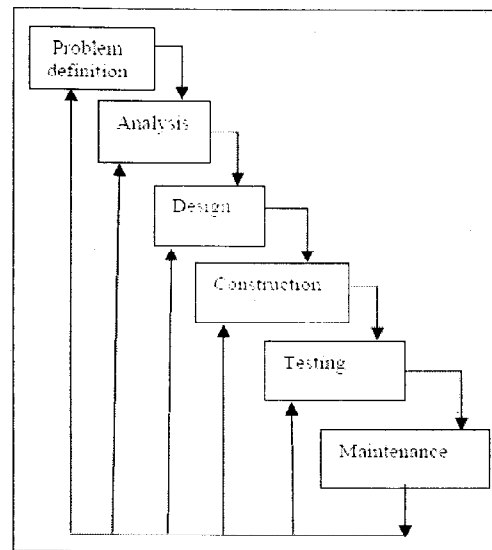
A. Enthoven and Henry Rowan developed a heuristic process for managing large information systems projects<sup>3</sup>. A methodology is a procedure for resolving the problems of the current system or for building a new one. Since the 1960's many descriptions of the classic software development life cycle have appeared<sup>4-7</sup>. Royce [1970] began the formulation of the software life cycle The SDLC is more commonly known as Structured Systems Analysis & Design. Structured methodologies allow the analyst to break down complicated systems into smaller, clearly defined and more manageable parts. The structured systems development life cycle is a step by step approach that moves from one phase to another commonly known as the waterfall model because the development is always moving forward with no mechanism in place to go backward. The first object-oriented languages came into existence during the 1960's and 1970's with Simula and Smalltalk. However, it was not until several years later that the Object-Oriented Analysis and Design (OOAD) methodology came into being<sup>5</sup>. First in 1982 Object-Oriented Design emerged as independent topic<sup>13</sup>, and later in 1988 Object-Oriented Analysis was introduced by S. Shlaer and S. Mellor (1988) and S. Bailin<sup>14</sup>. Many different object-oriented analysis and design

methods evolved since then by such well known individuals as (J. Rumbaugh, 1991), P. Coad and (E. Yourdon, 1991<sup>15</sup>) and many others. The OOAD methodology uses an object-oriented perspective rather than a functional perspective as in the SSAD methodology. An object is a person, place or thing initially drawn from the problem domain which has three aspects to it: what it knows (its identity and certain attributes), who it knows (relationships to other objects) and what it does (its methods it is responsible for performing<sup>16</sup> on its data). So in this methodology we think in terms of things (objects) rather than functions.

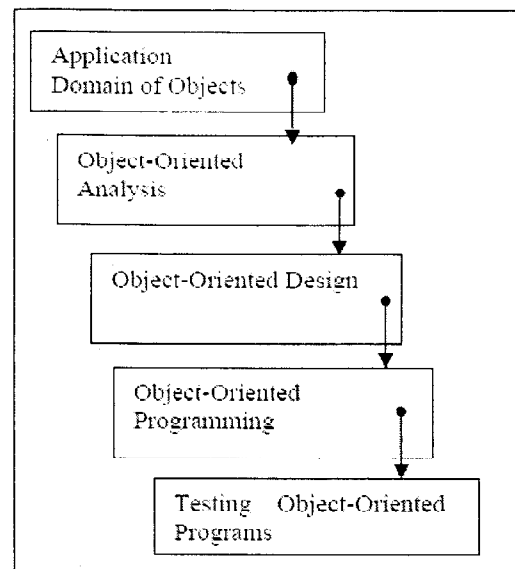
*Systems development methodologies/approaches:*

**CASE –I Traditional Systems Development:** It is a structured systems analysis and design (SSAD) technique which used data flow diagramming and entity modelling. Systems development life cycle (SDLC) provides overall framework for managing systems development process. Traditional approach also called structured system development, structured analysis and design technique (SADT) which includes information engineering (IE). All projects use some variation of SDM and two main approaches to SDM. Predictive approach – assumes project can be planned out in advance and there should be low technical risk. Adaptive approach – more flexible, assumes project cannot be planned out in advance which means requirements are uncertain and there should be high technical risk. An **advantage** of this scheme is that It's a systematic approach which makes easier to plan and manage the tasks of systems development & **Disadvantages** it is costly,

time consuming, inflexible<sup>1,2,4,6,7</sup>.



Conventional SDLC



SDLC for OOSD

**CASE –II Object-Oriented System Development:** An object-oriented system development is differs from traditional methodologies by combining data and procedures into one unified object rather than separately modelling business processes and data. It is also called OOA, OOD, and OOP which views information system as collection of interacting objects that work together to accomplish tasks. Object-oriented analysis (OOA) defines types of objects users deal with and shows use cases are required to complete tasks. Object-oriented design (OOD) defines object types needed to communicate with people and devices in system and shows how objects interact to complete tasks and refines each type of object for implementation with specific language of environment. Object-oriented programming (OOP) writing statements in programming language to define what each type of object does. Advantages of Object-Oriented approach are reuse, low cost, improved maintainability and quality and it can be used with rapid prototyping. Disadvantages of Object-oriented approach are that because technologies and methodologies are not mature or stable, potential benefits may not be attained and there is large investment in training required.<sup>11,13,14,15</sup>

#### *Comparative analysis of case I and case II:*

To increase the productivity, quality of products, services and maintain customer loyalty system can be developed only with proper

planning, analysis, design and implementation. This process is termed as Systems Development Life Cycle (SDLC). The objective of Systems analysis and design is to develop an effective Information System to reach organizational needs whether it go for a traditional approach or object-oriented approach to develop software according to requirements. Proposed D table in the next segment briefly analyze two approaches with their strengths and weaknesses respectively.

**D-tables:** In this paper we proposed two D-tables (  $D_1$  &  $D_2$  ). The proposed  $D_1$  table analyzes a-to-z functional behavior of traditional and object oriented approach with respect to certain functional **parameters** which aggregates user and developer vision about the characteristics and behavior of software.

In the next section  $D_2$  table is shown where quality and performance with comparative parameters of traditional and object oriented approach are shown. This table may use to help developer and user to contrast traditional and object oriented approach. From developer point of view it suggests behavioral and functional tools to respective system development approach. And in user point of view it helps to understand the user about complementary process and tools to fulfill requirements. These two tables ( $D_1$  &  $D_2$ ) also help user and developer to make a fully understood & an ambiguous SRS conformity.

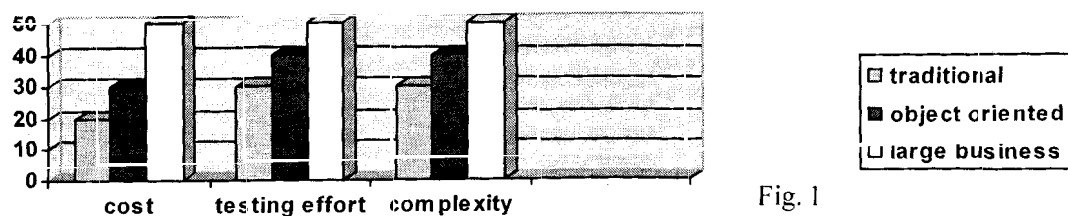


Fig. 1

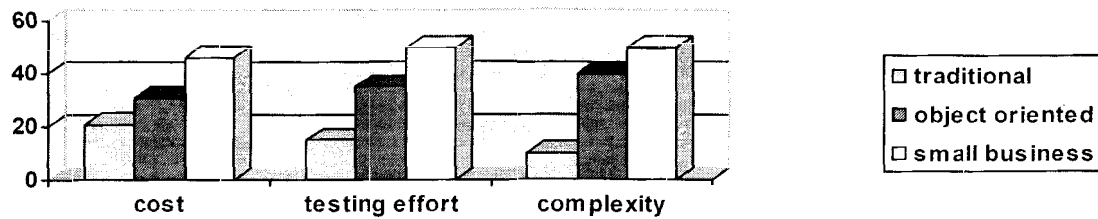


Fig. 2

D<sub>1</sub>-Table

Parameters	Traditional approach	Object oriented approach
System view	Data or process centric	Views system as a collection of object, data and process
System adoption	Adopts step by step approach to SDLC phases	No clear cut step from one phase to another
Approach	Views a system as a top down approach	Bottom up approach to system development
Required Documentation	Specific documentation are produced after each phase	No specific documentation after each phase
Models	Uses three models and diagrams i.e DFD, ERD, Structure chart	Class diagram, object, sequence diagram, component and deployment diagram
Behavior of models	Diagrams are simple and ease	Diagrams, charts are complex
Repository of data	A single data dictionary	No single data repository data.
I/O definition	Input and output of system are identified from DFD	Input and output are scattered in many sequence diagram
Process	Introduce use of modeling technique to describe process and data	It views system as a collection of process by using UML
Process, data and functional models	Process model and data models used	A functional model is urban in analysis phase itself.
Code Reusability	No reuse facility	Encapsulation allow reuse
Uniqueness	Do not have unique concepts thus leading to insecurity.	Polymorphism, inheritance, encapsulation and dynamic binding are the concepts
Qualities	Application specific, tested easy	Flexible, efficient and simply transformable into code

D<sub>2</sub> -Table

Type	Example	Testing	Metrics	Model
Traditional SDLC	SDLC, SSADM	System, unit testing and integration testing	LOC, Function Count, Function Point, Cyclomatic Complexity	DFD, ERD
Object oriented	RUP	Unit & system is simple integration is different and more complex	Class metrics, Method metrics Coupling metrics, Inheritance metrics	UML

As shown in above fig. (1,2) that in large business the cost of traditional approach is less than the object oriented and the same position is with the testing effort done by the programmers and the complexity of the system. In spite of that in small business the situation is almost opposite with the traditional and the object oriented approach. Cost, testing effort and complexity are higher in object oriented approach rather than traditional.

### Conclusion

Traditional and OOSD are completely different in terms of the models and diagrams. With the help of proposed D tables (D1, D2) we found OOSD does not require any documentation after each phase and structured approach mainly poses three diagrams. These tables help user and developers to mutually decide which approach is more beneficial and suitable to particular application. Mainly these tables help to understand both user & developers about both approach and adopt the one which is more appropriate to satisfy the requirement of system to develop good quality software.

### Reference

1. A. Rama Mohan Reddy, Dr. P. Govindarajulu, Dr. M. M. Naidu "A Process Model for Software Architecture", *IJCSNS International Journal of Computer Science and Network Security*, vol.7 no. 4, April (2007).
2. John W. Satzinger, Stephen D. Burd, Robert B. Jackson "Systems Analysis and Design in a Changing World", Fourth Edition (2006).
3. G. and Anderson, D., Management information systems: Solving business problems with information technology. (4th ed.). New York: McGraw-Hill Irwin (2006).
4. Darren Dalcier, Oddur Benediktsson, Helgi Thorbergsson, "Development Life Cycle Management: A Multiproject Experiment," ecbs, pp. 289-296, 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05), (2005).
5. Larman, C., Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, 3rd ed., Prentice Hall, Upper Saddle River,

- New Jersey (2004).
6. Craig Larman, Victor R. Basili, "Iterative and Incremental Development: A Brief History," *Computer*, vol. 36, no. 6, pp. 47-56, June (2003).
  7. Walt Scacchi, Institute for Software Research, University of California, Irvine "Process Models in Software Engineering", Encyclopedia of Software Engineering, 2<sup>nd</sup> Edition, John Wiley and Sons, Inc, New York, December (2001).
  8. Scacchi, W., Understanding Software Process Redesign using Modeling, Analysis and Simulation. *Software Process-Improvement and Practice* 5(2/3):183-195 (2000).
  9. Gross, Paul F. Systems Analysis and Design for Management (1976).
  10. W. A. Hosier, "Pitfalls and safeguards in real-time digital systems with emphasis on programming," IRE Trans. Eng. Management, pp. 99-115, June 1961; in Proc. 9th Int. Conf. Software Engineering, IEEE, Mar. (1987).
  11. Royce, W.W., "Managing the Development of Large Software Systems: Concepts and Techniques," Proc. WESCON, August (1970).
  12. Boehm, B. W., Software Engineering Economics, Prentice-Hall, Englewood Cliffs, N. J. (1981).
  13. Booch, G., "Object-Oriented Design", *Ada Letters Volume 1 (3)*, 64-76 (1982).
  14. Bailin, S., Remarks on Object-Oriented Requirements Specification, Computer Technology Associates, Laurel, MD (1988).
  15. Coad, P. and Yourdon, E., Object-Oriented Analysis, 2nd ed., Yourdon Press, Englewood Cliffs, New Jersey (1991).
  16. Norman, R., Object-Oriented Systems Analysis and Design, Prentice Hall, Upper Saddle River, New Jersey (1996).
  17. A. G. Sutcliffe, Object-Oriented Systems Development: survey of structured methods **Information and Software Technology** Volume 33, Issue 6 (July/Aug. 1991).