ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

Analysis of Process Model in Software Development

Neeraj Jaiswal[1], Rajesh Shah[2]

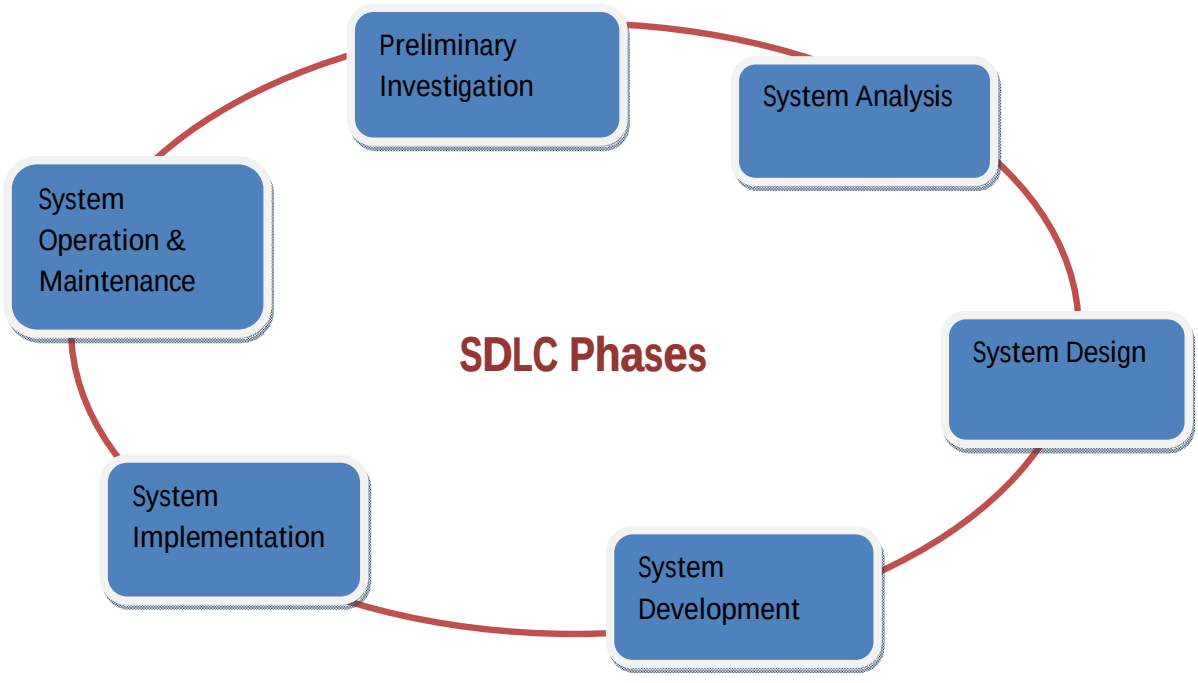
Professor, Dept. of Comp Science & Elex, Christian Eminent College, DAVV, Indore, MP, India[1,2]

ABSTRACT: In the present scenario all software systems are lacking because they cannot be built with mathematical or physical certainty, Hence in this research paper the comparison of various software development models has been discussed. Software development methodology (SDLC) in software engineering is a framework that is used to structure plan and control the process of developing an important system. Choosing the right SDLC methodology for your project is as important to the success of the project. Today all live in the era of computer technology. This paper also identity the basic problems in the spiral, waterfall, and iterative models. These models has own advantages and disadvantages. This research paper compare all four models on the basis of some key points which will be helpful to develop a successful software project with the help of comparison any one can select his/her own type of model for his/her project development Keywords- S DLC, phase of SDLC models, Comparative analysis of model, four models.

KEYWORDS: Software Development Process, models, Comparative analysis of models

I. INTRODUCTION

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.



ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

Preliminary Investigation:

- Determine if a new system is needed
- Three primary tasks:
  - Define the problem
    - By observation and interview, determine what information is needed by whom, when, where and why
  - Suggest alternative solutions
  - Prepare a short report

System Analysis:

- In depth study of the existing system to determine what the new system should do.
  - Expand on data gathered in Phase 1
- In addition to observation and interviews, examine:
  - Formal lines of authority (org chart)
  - Standard operating procedures
  - How information flows
  - Reasons for any inefficiencies

System Design Tools Used:

- Uses specifications from the systems analysis to design alternative systems
- Evaluate alternatives based upon:
  - Economic feasibility - Do benefits justify costs?
  - Technical feasibility - Is reliable technology and training available?
  - Operational feasibility - Will the managers and users support it?

System Development:

- Build the system to the design specifications
  - Develop the software
    - Purchase off-the-shelf software OR
    - Write custom software
  - Acquire the hardware
  - Test the new system
    - Module (unit) test - tests each part of system
    - Integration testing - tests system as one unit
  - Create manuals for users and operators

System Implementation:

- Convert from old system to new system
- Train users
- Compile final documentation
- Evaluate the new system

Operations & Maintenance:

- Evaluation Methods
  - Systems audit - performance compared to original specifications
  - Periodic evaluation - “checkups” from time to time, modifications if necessary

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

II. RELATED WORK

A framework that describes the activities performed at each stage of a software development project

A. Waterfall Model:

. Requirements – defines needed information, function, behavior, performance and interfaces.
- Design – data structures, software architecture, interface representations, algorithmic details.
- Implementation – source code, database, user documentation, testing.

Waterfall Strengths:

- Easy to understand, easy to use.
- Provides structure to inexperienced staff.
- Milestones are well understood.
- Sets requirements stability.
- Good for management control (plan, staff, track).
- Works well when quality is more important than cost or schedule.

Waterfall Deficiencies:

- All requirements must be known upfront
- Deliverables created for each phase are considered frozen – inhibits flexibility
- Can give a false impression of progress
- Does not reflect problem-solving nature of software development – iterations of phases
- Integration is one big bang at the end
- Little opportunity for customer to preview the system (until it may be too late)

When to use the Waterfall Model:

- Requirements are very well known
- Product definition is stable
- Technology is understood
- New version of an existing product
- Porting an existing product to a new platform.

B. RAPID APPLICATION MODEL (RAD)

- Requirements planning phase (a workshop utilizing structured discussion of business problems).

- User description phase – automated tools capture information from users.
- Construction phase – productivity tools, such as code generators, screen generators, etc. inside a time-box. ("Do until done").
- Cutover phase -- installation of the system, user acceptance testing and user training.

RAD Strengths:

- Reduced cycle time and improved productivity with fewer people means lower costs
- Time-box approach mitigates cost and schedule risk
- Customer involved throughout the complete cycle minimizes risk of not achieving customer satisfaction and business needs

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

- Focus moves from documentation to code (WYSIWYG).
- Uses modeling concepts to capture information about business, data, and processes.

RAD Weaknesses:

- Accelerated development process must give quick responses to the user
- Risk of never achieving closure
- Hard to use with legacy systems
- Requires a system that can be modularized
- Developers and customers must be committed to rapid-fire activities in an abbreviated time frame.

When to use RAD:

- Reasonably well-known requirements
- User involved throughout the life cycle
- Project can be time-boxed
- Functionality delivered in increments
- High performance not required
- Low technical risks
- System can be modularized

C. Spiral Model

The spiral model is similar to the incremental model, with more emphases placed on risk analysis. The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation. A software project repeatedly passes through these phases in iterations (called Spirals in this model). The baseline spiral, starting in the planning phase, requirements are gathered and risk is assessed. Each subsequent spiral builds on the baseline spiral. Requirements are gathered during the planning phase. In the risk analysis phase, a process is undertaken to identify risk and alternate solutions. A prototype is produced at the end of the risk analysis phase. Software is produced in the engineering phase, along with testing at the end of the phase. The evaluation phase allows the
customer to evaluate the output of the project to date before the project continues to the next spiral.In the spiral model, the angular component represents progress, and the radius of the spiral represents cost.

Advantages

1. High amount of risk analysis.
2. Good for large and mission-critical projects.
3. Software is produced early in the software life cycle.

Disadvantages

1. Can be a costly model to use.
2. Risk analysis requires highly specific expertise.
3. Project's success is highly dependent on the risk analysis phase.
4. Doesn't work well for smaller projects.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

III. PROPOSED METHODOLOGY

There are various methodologies. They are as follows

A. **Data Collections:** The challenge of collecting software engineering data is to make sure that the collected data can provide useful information for project, process, and quality management and, at the same time, that the data collection process will not be a burden on development teams. Therefore, it is important to consider carefully what data to collect. The data must be based on well-defined metrics and models, which are used to drive improvements. Therefore, the goals of the data collection should be established and the questions of interest should be defined before any data is collected. Data classification schemes to be used and the level of precision must be carefully specified. The collection form or template and data fields should be pretested. The amount of data to be collected and the number of metrics to be used need not be overwhelming. It is more important that the information extracted from the data be focused, accurate, and useful than that it be plentiful. Without being metrics driven, over-collection of data could be wasteful. Over collection of data is quite common when people start to measure software without an a priori specification of purpose, objectives, profound versus trivial issues, and metrics and models.

B. Data Analysis: Analysis of data is a process of inspecting, cleaning, transforming, and modelling data with the goal of discovering useful information, suggesting conclusions, and supporting decision-making. Data analysis has multiple facets and approaches, encompassing diverse techniques under a variety of names, in different business, science, and social science domains. Data mining is a particular data analysis technique that focuses on modelling and knowledge discovery for Software Engineering Models rather than purely descriptive purposes.

IV. CONCLUSION AND FUTURE WORK

After completing this research , it is concluded that :

A. Planning in early stage is good for all models.

B. Returning to an earlier phase is not possible in waterfall model but in other two model this is possible.

C. Handling of large project is good in Spiral model not in other two model.

D. Detailed Documentation is Necessary for waterfall and RAD model.

E. Flexibility to change code is possible in Spiral and RAD model but not possible in Waterfall model.

F. Team size will be small in Spiral and RAD model but Waterfall model not worked with small team.

REFERENCES

[1] Rajesh Shah et al **,"** A Comparative Study of two Software development Approach Traditional and object Oriented ",page no-1383-1387,ISSN-2277128X," International Journal of Advanced Research in Computer Science and Software Engineering**",**VOl -5 Issue-5.

[2] Ambriola, V., R. Conradi and A. Fuggetta, Assessing process-centered software engineering environments, ACM Trans. Softw. Eng. Methodol. 6, 3, 283-328, 1997.

[3] Balzer, R., Transformational Implementation: An Example, IEEE Trans. Software Engineering, 7, 1, 3-14,1981.

[4] Basili, V. R., and A. J. Turner, Iterative Enhancement: A Practical Technique for Software Development, IEEE Trans. Software Engineering, 1,4, 390-396, 1975.

[5] B. Curtis, H. Krasner, V. Shen, and N. Iscoe, On Building Software Process Models Under the Lamppost, Proc. 9th. Intern. Conf. Software Engineering, IEEE Computer Society, Monterey, CA, 96-103, 1987.

[6] Curtis, B., H. Krasner, and N. Iscoe, A Field Study of the Software Design Process for Large Systems, Communications ACM, 31, 11, 1268-1287, November, 1988.

[7] Hekmatpour, S., Experience with Evolutionary Prototyping in a Large Software Project, ACM Software Engineering Notes, 12,1, 38-41 1987.