e-ISSN: 2349-9745
p-ISSN: 2393-8161
Scientific Journal Impact Factor (SJIF): 1.711
International Journal of Modern Trends in Engineering and Research
www.ijmter.com

Comparative study of Waterfall model with RAD model

Sunil D. Mone[1]
Assistant Professor in Computer Science,R. C. Patel ACS College, Shirpur-425 405(MS)

Abstract: Among numerous software development architecture or para diagrams it is important to choose proper architecture. To choose proper architecture, need to consider some criteria based on which we can compare two or more architectures.
In this paper I am trying to compare and see feasibility of Water fall model and RAD model based on criteria like Systematic Development, Team sized required, Speed, Involvement of user, Complexity, Parallel Development, Success Ratio, Passions Required, Capacity to handle large project.
It seen that when time is more important in that case RAD model is important while when focus is on systematic Waterfall model is more important.
Keywords: Architecture, paradiagrams, waterfall model, RAD model, featuers, feasibility, systematic developemtn, economic

I. INTRODUCTION

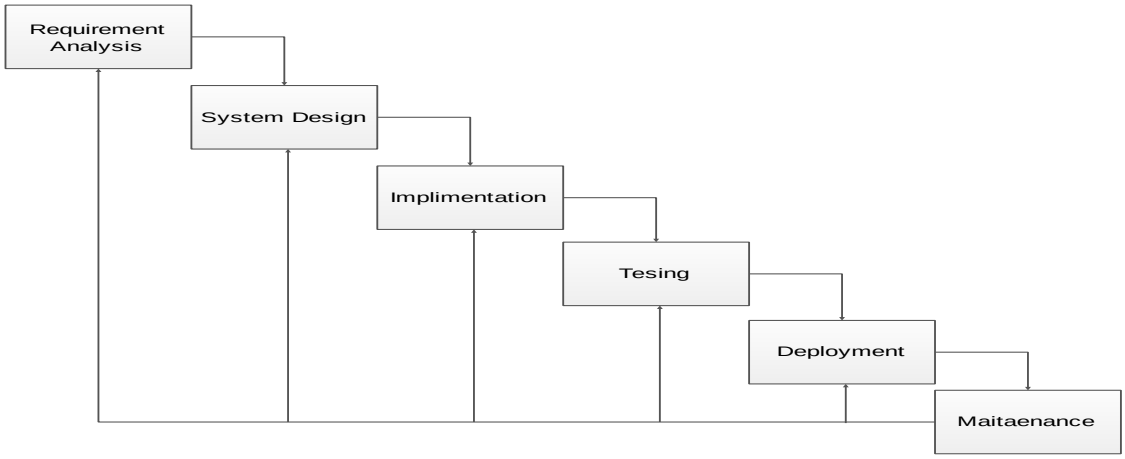
Ther are numerus software development architechtures or software development paradiagrams. Mostly it seen that improper selection of architectuer may lead to unpreditably increase in cost and time or even may leads to failuare.
Architecture design plays a prominent role in software engineering processes. Regardless of the variations in software engineering processes, software architecture provides the skeleton and constraints for software implementation. Software architecture may be defined as "the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution" (Std, 2000)
For development of software selection of proper architecure is little bit tedious job, but one can make it simple by listing it's requirements and then based on their requirement one can compare two or more architecures and architecure which satify more requirements may be the best solution for development.

II. RELATED WORK

The evolution of these architecture paradigms has deep and subtle influences and impacts on software systems, in the way components are designed and structured and in the way component interactions are implemented. The evolution is driven by business integration needs both in terms of enterprise application integration and distributed systems integration. (Guijun Wang, 2004)
"The difference between design pattern and idioms involves the scope at which they solve problems and their language specificity. From the scope point of view, idioms are just smaller patterns. From the language point of view, idioms apply to specific language whereas the design patterns apply to multiple languages using the same methodology." (Žilvinas VAIRA, 2011)
It is increasingly common to use programmable computers in applications where their failure could be life threatening and could result in extensive damage. For example, computers now have safety-critical functions in both military and civilian aircraft, in nuclear plants, and in medical devices.. The safety aspects of computer-based systems as increasingly important as the use of software escalate because of its convenience and flexibility. Incorrect requirements have been identified as a major cause of software accidents and it appears that current software safety standards do not place a proportionate emphasis upon this causal factor. (S. Phani Kumar, 2010)
The improvement has been observed as software development source, in each of parameter, which showed that after application of 80/20 rule in software model (Waterfall model), the considerable improvement was seen which made the software process model (Waterfall model) more useful and efficient (as is done through the application of 80/20 rule in software process model (Waterfall

model), as the effort is reduced and overall performance of the software process has been increased to much extent and efficiently. Thus, as a result, we achieved Software Process Model Improvement, Effort Reduction, Software Process Performance Enhancement (Optimization). (Rizwan, 2009)
Waterfall Model is easy to manage due to the rigidity of the model as each phase has specific deliverables and a review process. It works well for smaller projects where requirements are very well understood. (Rajendra Ganpatrao Sabale, (July 2012))
Waterfall model is simple and easy to understand and use. It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process. In this model phases are processed and completed one at a time. Phases do not overlap. Waterfall model works well for smaller projects where requirements are very well understood. Iterative model is much better model of the software process. It allows feedback to proceeding stages. It can be used for project wherein the requirements are not well understood. (Gupta, March 2015)

III. WATERFALL MODEL

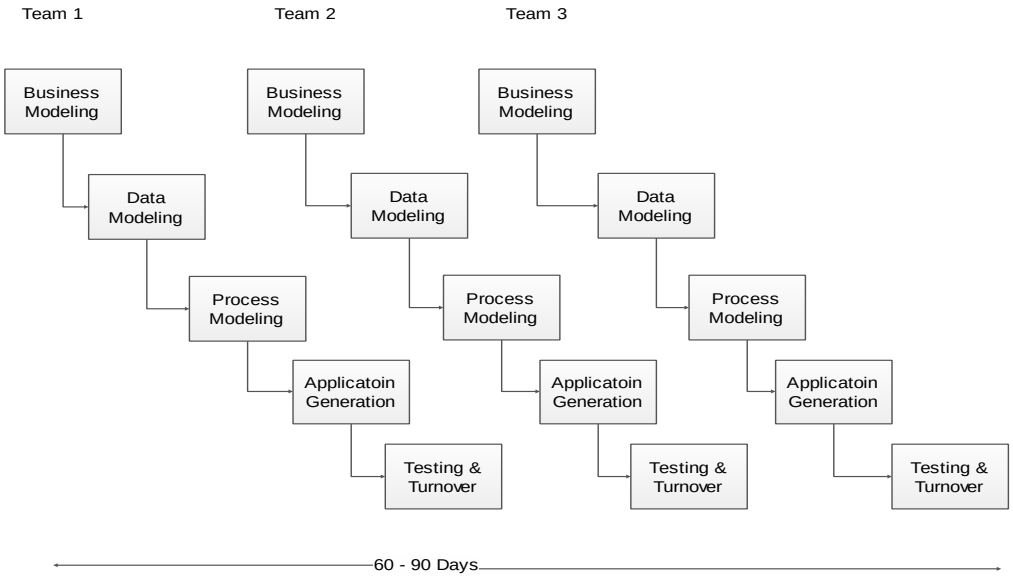


1) Requirement Analysis:- In this phase requirements are collected from user analyzed by communicating with customer and then prepare proper documentation which will helpful for next phases.
2) System design:- Whole system is partitioned into process also define interface between process. Followed by with the help of designing tools logic of process and interface between them is represented.
3) System implementation:- In this phase logical system is converted into physical system by coding followed by testing individual module.
4) System Testing:- Series of test cases are prepared and whole system is tested after integrating all modules.
5) Deployment:- system is not only implemented or installed but also gain faith from users by giving them training and satisfying quires about newly implementing system.
6) Maintenance:- After implementation when system runs it may goes maintenance due to change in policy or some bugs found at work, system will be maintained.

Features

- Waterfall model is oldest software development architecture. It also know as linear sequential life cycle model.
- This model is very simple to understand.
- One can not move to next phase until completion of current phase.
- Waterfall model provide systematic and sequential approach of software development begins from requirement gathering and improvement through design, coding/ implementation, testing , deployment and maintenance.

IV. RAD (RAPID APPLICATION DEVELOPMENT MODEL)



1) Business modeling:- The information flow is between various functions of business is identified. A complete business analysis is performed to find important information for business in ways that answers the following questions.

- how dose information obtain
- what information is obtain
- who obtain this information
- In which functioning of unit this information is used
- Who process this information

2) Data modeling:- The information obtained in the above phase is reviewed with customer and analyzed with the help of customer and users to form data objects. In this phase attributes of data object is identified also identified relationship between various data objects or entities.

3) Process Modeling:- Process are define to manipulate data objects including adding, modifying, deleting or retrieving a data objects.

4) Application Generation:- Actual coding is developed and system is build by using tools to converts the process and data models into actual prototype.

5) Testing and turnover:- each component followed by components after interfacing is tested.

Features

- It is linear sequential software process model.
- It develop software must faster.
- Complete development is expected to complete within 60 – 90 days only.
- Major functions can be addressed or handled by separate RAD teams in parallel.

V. CRITERIA WISE FEASIBILITY OF APPROPRIATE MODEL

Sr. No.	Criteria	Feasibility	
		Waterfall model	RAD Model
01	Systematic Development	High	Medium
02	Team sized required	Low	Medium
03	Speed	Low	High
04	Involvement of user	Low	Medium
05	Complexity	Low	Medium

06	Parallel Development	Low	High
07	Success Ratio /	High	Medium
08	Economical	High	Low
09	Passions Required	High	Low
10	Capacity to handle large project	Medium	High

VI. CONCLUSION

1. When time is more important than money RAD model is preferred (which develop software using parallelism)
2. When size of project is large, one can prefer RAD model instead of waterfall model.
3. When systematic development and low risk(success ratio) is expected and user as well as customer have passions then one can prefer waterfall model instead of RAD model.
4. When you have sufficient expert, one can prefer to develop software using RAD model instead of waterfall model.

REFERENCES

[1] Guijun Wang, C. K. (2004). Architecture Paradigms and Their Influences and Impacts on componant based software system . (Proceedings of the 37th Hawaii International Conference on System Sciences - 2004).

[2] Gupta, M. A. (March 2015). A Comparision between differnt type of software development life cycle models in softwre engineering. *International Journal of Advanced Technology in Engineering and Science Volume No 03, Special Issue No. 01 ISSN (online): 2348 – 7550* , 624-631.

[3] Pressman, R. (Dec.2009). *SOFTWARE ENGINEERING 6E: A Practitioner's Approach.* McGraw-Hill Higher Education.

[4] Rajendra Ganpatrao Sabale, D. A. ((July 2012)). Comparative Study of Prototype Model For Software Engineering With System Development Life Cycle. *IOSR Journal of Engineering (IOSRJEN) ISSN: 2250-3021 Volume 2, Issue 7* , PP 21-24.

[5] Rizwan, M. I. (2009). APPLICATION OF 80/20 RULE IN SOFTWARE ENGINEERING WATERFALL. *978-1-4244-4609-4/09/$25.00 ©2009 IEEE* .

[6] S. Phani Kumar, D. R. (2010). A Methodology for Building Safer Software based Critical. *IEEE 2nd International Advance Computing Conference 978-1-4244-4791-6/10/$25.00 c* , 422-429.

[7] Singh, K. A. (2007). *Software Engineering (3rd ed.),.* New Age International Publishers.

[8] Std, 1.-2. I. (2000). Recommended Practice for Architectural Description of Software Intensive Systems,. *IEEE Std 1471-2000* .

[9] Žilvinas VAIRA, A. C. (2011). Software Engineering Paradigm Independent Design Problems, GoF 23 Design Patterns, and Aspect Design. *INFORMATICA, Vol. 22, No. 2,* , 289–317.

