Review Article [Dhanotia, 1(3): May, 2012]
ISSN: 2277-9655

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Rapid Application Development (Rad) Approach with Halt Points

Shruti Dhanotia[1], Ruchi Goyal[2]
[1]Dept. of Computer Science Engineering, Patel College of science & Technology, Indore.
[2]Dept. of Computer Science Engineering, Patel College of science & Technology, Indore.
shruti.becse@gmail.com

Abstract

In the late sixties and early seventies many of the software crisis are faced in software technologies, but in today scenario we are adopting some SOFTWARE ENGINEERING MODEL. The software development process transforms a user's needs into software. Basically software models are use to emerge to specify a user, how to create our documentation part of our project or software. Those software models are described what is the equipment required and what's the functionality are required in our project or we can say that "Software Models are the blue print of our developing projects". So, in this paper, we are introducing some Halt Points (Hps) at every phase of RAD model and FEEDBACK goes through to the starting 1st# test case.
Through these Hps if any error occurred in any phase will solved in same phase then error will not carry over into the next phase while client or user will change in requirement at any phase by jumping to the first required phase.

Introduction

Software engineering (SE) is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software.[1] It is the application of engineering to software because it integrates significant mathematics, computer science and practices whose origins are in engineering.[2] It is also defined as a systematic approach to the analysis, design, assessment, implementation, testing, maintenance and reengineering of software, that is, the application of engineering to software.[3] The term software engineering first appeared in the 1968 NATO Software Engineering Conference, and was meant to provoke thought regarding the perceived "software crisis" at the time.[4][5]

Rapid application development is a software development methodology that involves methods like iterative development and software prototyping. According to Whitten (2004), it is a merger of various structured techniques, especially data-driven Information Engineering, with prototyping techniques to accelerate software systems development.[6]

In rapid application development, structured techniques and prototyping are especially used to define users' requirements and to design the final system. The development process starts with the development of preliminary data models and business process models using structured techniques. In the next stage, requirements are verified using prototyping, eventually to refine the data and process models. These stages are repeated iteratively; further development results in "a combined business requirements and technical design statement to be used for constructing new systems".[6]

RAD approaches may entail compromises in functionality and performance in exchange for enabling faster development and facilitating application maintenance.

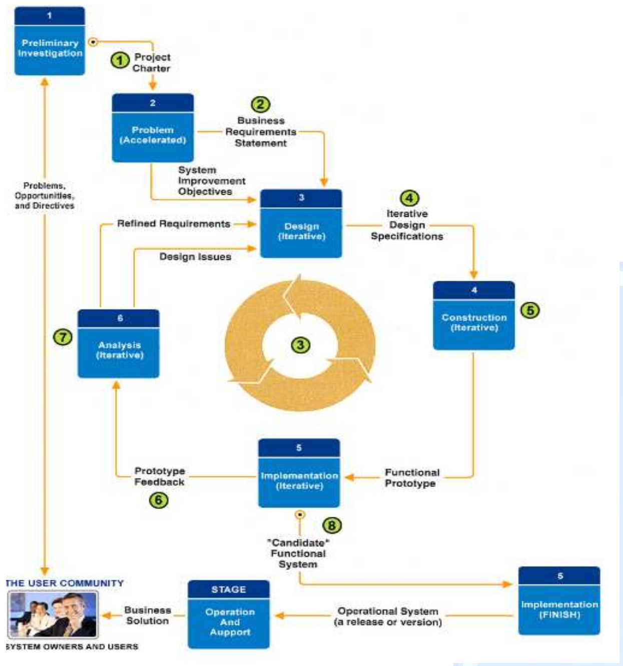

1 Preliminary Investigation
1 Project Charter
2 Problem (Accelerated)
2 Business Requirements Statement
System Improvement Objectives
3 Design (Iterative)
4 Iterative Design Specifications
Refined Requirements
Design Issues
Problems, Opportunities, and Directives
4 Construction (Iterative)
5
6 Analysis (Iterative)
7
3
5 Implementation (Iterative)
Functional Prototype
Prototype Feedback
6
8
"Candidate" Functional System
THE USER COMMUNITY
STAGE Operation And Aupport
Business Solution
Operational System (a release or version)
5 Implementation (FINISH)
SYSTEM OWNERS AND USERS

Fig. 1.1 Process Of RAD Model

System Architecture

A Rapid Application Develop is an incremental software development process model that has extremely short development cycle. It is a high speed version of the linear sequential model in which rapid development is achieved by using component based construction. If requirement are well understood and project scope is constrained, the RAD process enables a development team to create a fully functional system within 60 to 90 days.

RAD model has the following phases [7]:

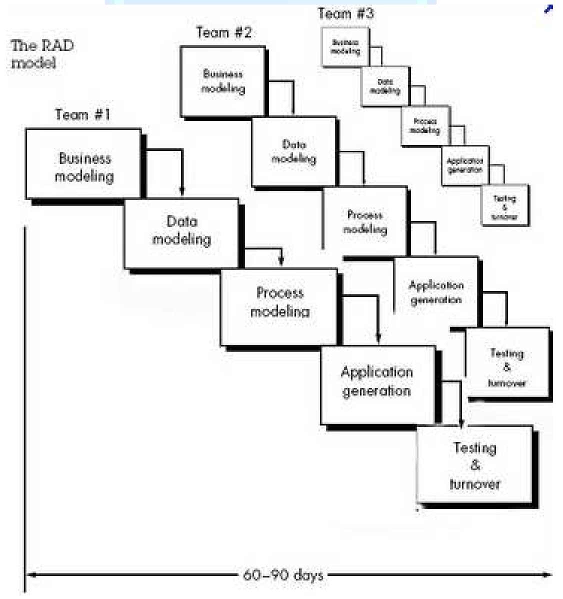
Business Modeling: The information flow among business functions is defined by answering questions like what information drives the business process, what information is generated, who generates it, where does the information go, who process it and so on.

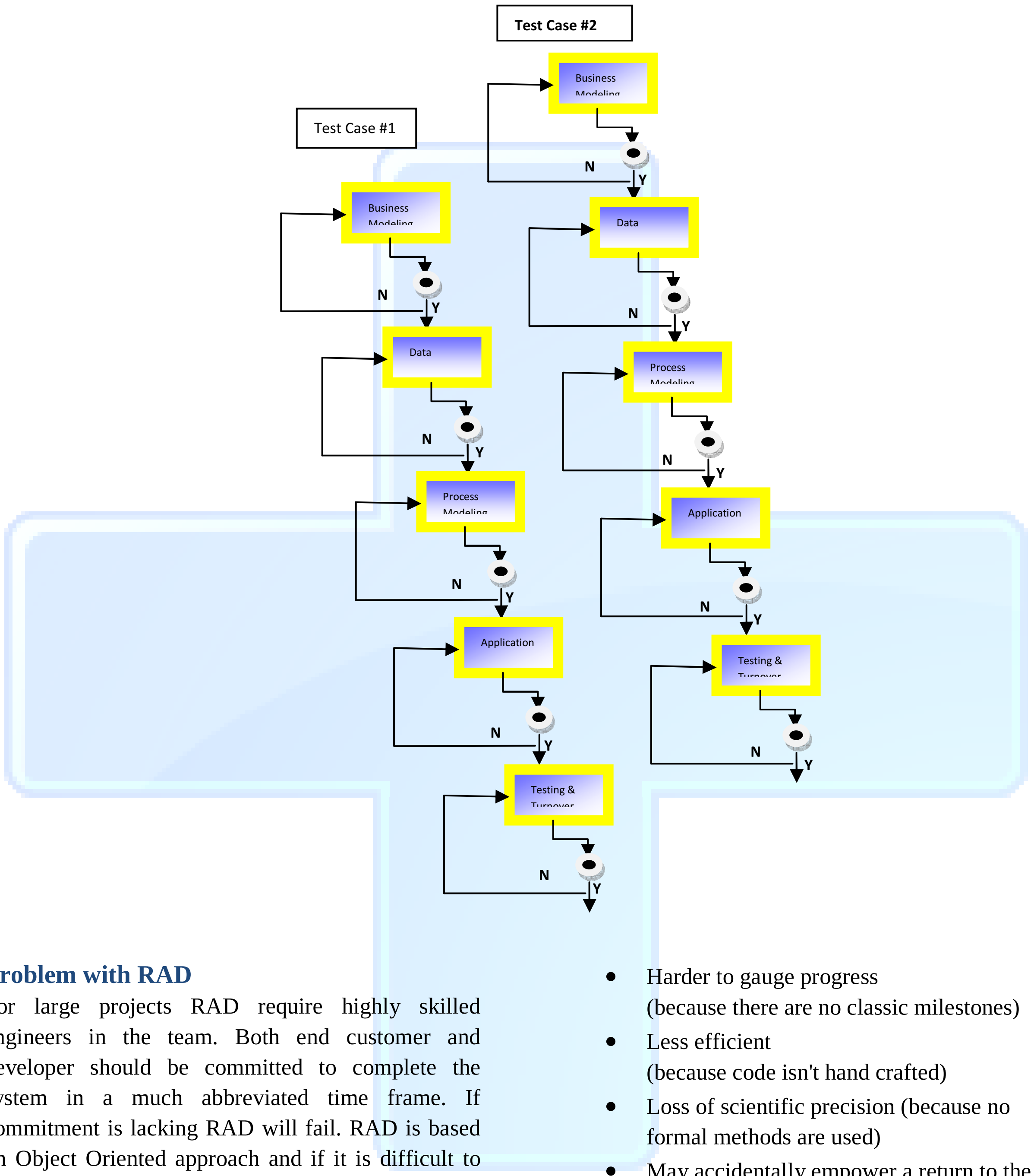
Data Modeling: The information collected from business modeling is refined into a set of data objects (entities) that are needed to support the business. The attributes (character of each entity) are identified and the relation between these data objects (entities) is defined.

Process Modeling: The data object defined in the data modeling phase are transformed to achieve the information flow necessary to implement a business function. Processing descriptions are created for adding, modifying, deleting or retrieving a data object.

Application Generation: Automated tools are used to facilitate construction of the software; even they use the 4th GL techniques.

Testing and Turn over: Many of the programming components have already been tested since RAD emphasis reuse. This reduces overall testing time. But new components must be tested and all interfaces must be fully exercised.





Problem with RAD

For large projects RAD require highly skilled engineers in the team. Both end customer and developer should be committed to complete the system in a much abbreviated time frame. If commitment is lacking RAD will fail. RAD is based on Object Oriented approach and if it is difficult to modularize the project the RAD may not work well.

- Buying may not save money compared to building
- Cost of integrated toolset and hardware to run it
- Harder to gauge progress (because there are no classic milestones)
- Less efficient (because code isn't hand crafted)
- Loss of scientific precision (because no formal methods are used)
- May accidentally empower a return to the uncontrolled practices of the early days of software development
- More defects (because of the "code-like-hell" syndrome)
- Prototype may not scale up, a B-I-G problem

- Reduced features (because of time boxing, software reuse)
- Reliance on third-party components may ...
    a. sacrifice needed functionality
    b. add unneeded functionality
    c. create legal problems
- Requirements may not converge (because the interests of customers and developers may diverge from one iteration to the next)
- Standardized look and feel (undistinguished, lackluster appearance)
- Successful efforts difficult to repeat (no two projects evolve the same way)
- Unwanted features (through reuse of existing components)

Iterative Application Development Methodology

This section describes the development method used; the Researcher's aim is to emphasize aspects of the development process that are distinctive as far as RAD is concerned not to focus on the product. The developers adopted their own in-house commercial Iterative Application Development (IAD)Approach to promote a controlled, structured but flexible development methodology aimed at providing incremental delivery. This involved a series of time-boxed mini iterations and a number of software 'release' and test iterations to provide flexibility to meet the recognized volatile needs of theBusiness environment. They believe this methodology offers all the main benefits of a RAD type approach and is suited to the uncertainty of, and continually changing business requirements. IAD, like RAD, involves prototyping and iterative delivery but without the problems of lack of rigor,
creeping scope and overrun that are perceived as associated with RAD and an iterative development life cycle. Its similarity is extended to using the same main features i.e. JAD (Joint Application Design) workshops, time-boxing, prototyping, intensive user involvement, iterative development and incremental delivery, which they maintain are increasingly used for system functionality development.
The developers believe that a major benefit of an iterative approach to development is that it affords early visibility of the system being developed. Thus early validation of the system by the users and the business analysts provides the flexibility to incorporate user feedback and handle any new or changing requirements within the volatile business environment – a key goal of the RAD approach.

Flowcharts

A Multi Model Methodology

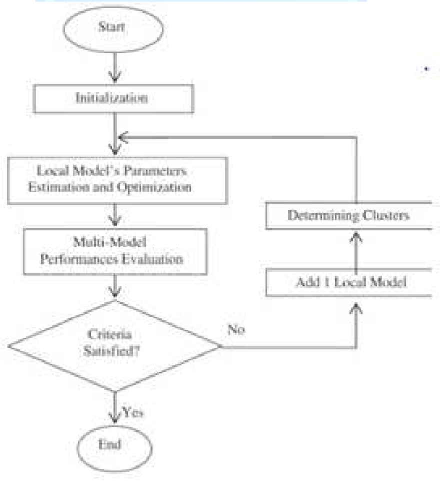

Start
Initialization
Local Model's Parameters Estimation and Optimization
Multi-Model Performances Evaluation
Criteria Satisfied?
No
Yes
End
Determining Clusters
Add 1 Local Model

Fig 1.3 (A) multi-model based identification methodology

B. RAD Model flow charts

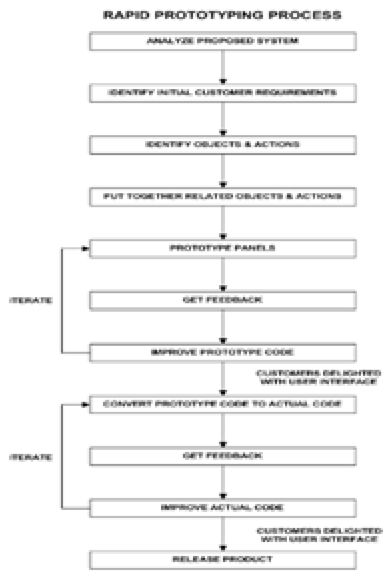

RAPID PROTOTYPING PROCESS
ANALYZE PROPOSED SYSTEM
IDENTIFY INITIAL CUSTOMER REQUIREMENTS
IDENTIFY OBJECTS & ACTIONS
PUT TOGETHER RELATED OBJECTS & ACTIONS
PROTOTYPE PANELS
ITERATE
GET FEEDBACK
IMPROVE PROTOTYPE CODE
CUSTOMERS DELIGHTED WITH USER INTERFACE
CONVERT PROTOTYPE CODE TO ACTUAL CODE
ITERATE
GET FEEDBACK
IMPROVE ACTUAL CODE
CUSTOMERS DELIGHTED WITH USER INTERFACE
RELEASE PRODUCT

Fig.3 (B). Flow chart of RAD

Conclusion

A new modified version is presented in which weakness of RAD model and iterative RAD model are overcome. The new model consist some Halt Point S (Hps) to provide better efficiency to generate the software product. In odified RAD model security constraints are also used in each and every phase of each and every test case of the model so that there is no chance of any software problem at the end user side. Now the problem if any occurred on any Phase of modified iterative RAD model then it will solve on that phase where it is generate.

Future Expansion

In these modified RAD model need strong man power, because we are including Halt Point S (Hps) each and every phase of each and every test case so at that point we need some testers that meet with client and verify the required data that can concluded by the developer. And also here required some more time as compared to without Halt PointS (Hps) RAD model, because after every phase of test case developer go to the client, so these take more time as compaired to without Halt PointS (Hps) RAD model. So in future these weakness of the iterative RAD model can be solve by using some methodology.

References

[1] SWEBOK executive editors, Alain Abran, James W. Moore ; editors, Pierre Bourque, Robert Dupuis. (2004). Pierre Bourque and Robert Dupuis. ed. Guide to the Software Engineering Body of Knowledge 2004 Version. IEEE Computer Society. pp. 1–1. ISBN 0-7695-2330-7.

[2] ACM (2006). "Computing Degrees & Careers". ACM. Retrieved 2010-11-23.

[3] Laplante, Phillip (2007). What Every Engineer Should Know about Software Engineering. Boca Raton: CRC.
ISBN 978-0-8493-7228-5. Retrieved 2011-01-21.

[4] Peter, Naur; Brian Randell (7–11 October 1968). "Software Engineering: Report of a conference sponsored
by the NATO Science Committee" (PDF). Garmisch, Germany: Scientific Affairs Division, NATO. Retrieved
2008-12-26.

[5] Randell, Brian (10 August 2001). "The 1968/69 NATO Software Engineering Reports". Brian Randell's University Homepage. The School of the Computer Sciences, Newcastle University. Retrieved 2008-10-11. "The idea for the first NATO Software Engineering Conference, and in particular that of adopting the then practically unknown term "software engineering" as its (deliberately provocative) title, I

believe came originally from Professor Fritz Bauer."artinez

[6] Hitten, Jeffrey L.; Lonnie D. Bentley, Kevin C. Dittman. (2004). Systems Analysis and Design Methods. 6th edition. ISBN 025619906X.

[7] http://ezinearticles.com/?Rapid-Application-Development-Model-%28RAD%29&id=412312