Application of the Pareto Principle in Rapid Application Development Model

Vishal Pandey [#1], AvinashBairwa [#2], Sweta Bhattacharya [#3]

School of Information Technology & Engineering
VIT University, Vellore
[1] vishalpandey22@gmail.com
[2] avinash.bairwa2011@vit.ac.in
[3] sweta.b@vit.ac.in

***Abstract*— the Pareto principle or most popularly termed as the 80/20 rule is one of the well-known theories in the field of economics. This rule of thumb was named after the great economist Vilferdo Pareto. The Pareto principle was proposed by a renowned management consultant Joseph M Juran. The rule states that 80% of the required work can be completed in 20% of the time allotted. The idea is to apply this rule of thumb in the Rapid Application Development (RAD) Process model of software engineering. The Rapid application development model integrates end-user in the development using iterative prototyping emphasizing on delivering a series of fully functional prototype to designated user experts. During the application of Pareto Principle the other concepts like the Pareto indifference curve and Pareto efficiency also come into the picture. This enables the development team to invest major amount of time focusing on the major functionalities of the project as per the requirement prioritization of the customer. The paper involves an extensive study on different unsatisfactory projects in terms of time and financial resources and the reasons of failures are analyzed. Based on the possible reasons of failure, a customized RAD model is proposed integrating the 80/20 rule and advanced software development strategies to develop and deploy excellent quality software product in minimum time duration. The proposed methodology is such that its application will directly affect the quality of the end product for the better.**

Keyword - 80/20 rule, Pareto principle, Pareto efficiency, RAD, Rapid Development

I. INTRODUCTION

The Rapid application Development process model is an incremental model which uses concepts like rapid prototyping, Joint Application Development, CASE tools etc. It basically aids creating functional products in less amount of time wherein the entire time span of a project undertaken in RAD process model may be 90-100 days. The time span being so concise, there may be various problems in development such as incomplete requirements set, insufficient time for testing, improper documentation etc. As a result of these problems it is possible that quality of the end product may have been compromised. In this paper we are attempting to minimize such problems by proposing a reduced version of the Rapid Application Development process model using the 80/20 rule of economics. [3][4]

II. LITERATURE REVIEW

As a part of literature review, an intensive study was undertaken on RAD model, Pareto Principle and various failed Software projects were analyzed closely which were developed using the Rapid Application Development Model.

A. RAD Process Model

The RAD process model is a software process model which is used to develop a software system in increments in a short span of time using concepts like prototyping, CASE tools and JAD. [4][5]

B Pareto Principle

The Pareto principle is a well-known rule of thumb in economics which states that 80% of the target can be achieved in 20% of time allotted. [2][3]

The following projects as shown in Table I were developed based on the RAD principle and the reasons of failure of these projects were analyzed to find out the major areas and reasons of effect. The identification of the root cause of the failure would help to find out an appropriate solution to the problem and design a better methodology. [8]

TABLE I
Failed Projects and Reasons of Failure

Various Failed projects:

Project Name	**Reasons Of failure**
DRAMA	• Incomplete set of requirements. • Not commercially viable.
CleanChief	• Less interaction with the targeted end users. • Incomplete requirement set. • Inappropriate understanding of requirements. • Wrong cost estimation.
Chimsoft	• Focusing on unimportant features.
PC Desktop Cleaner	• Not commercially viable. • Too many throw away prototypes
Highlighter	• Too much focus on small aspects but not on the core features.
nBinder	• Less interaction with targeted users.
Net-Herald	• Improper feasibility analysis. • Expensive in long run.
HabitShaper	• Less interaction with end users and other stake holders.
BPL interpreter	• Less interaction with targeted users.
ScreenRest	• To expensive compared to features. • Not commercially viable.

III. PROBLEM DEFINITION

As already known, in Rapid Application Development process model the time span of the project is pre-decided and is very small(less than 100 days) which leads to a major drawback and causes complete project failure. Being more precise the time being less it becomes difficult to decide which feature to given more priority in terms of time keeping in mind the constraints related to client requirements. In such cases the choice of features plays very important role. If decisions are not made correctly the software is deliberate to fail. Even if the choices are made correctly time management has to be done efficiently otherwise resulting in a substandard and compromised software product. The 80/20 rule is used keeping in mind the above mentioned drawbacks and deficiencies and if applied effectively can help to eradicate the issues completely. [4][5]

IV. SOLUTION METHODOLGY AND THE PROPOSED MODEL

A. Division of Teams

Initially, the total number of members or employees working in the software development firm should be divided into teams and sub teams in the first set of iterations. The division of team members should be such that each team work on single iteration and each sub team should be assigned one task of that iteration. The division of team members is done in the following manner as shown in Fig 1.

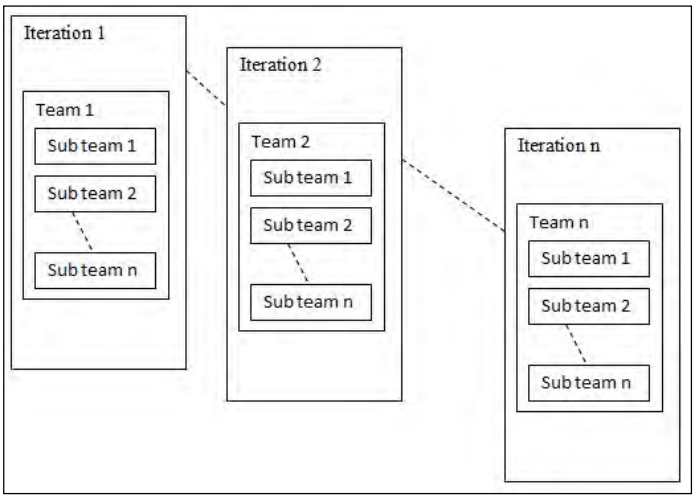

Fig. 1. Division of teams in different iterations

B. Set Requirements Priorities

While the requirements are taken down from the user/client, the user is asked to provide a priority value to every requirement on the basis of the importance of the requirement as provided or mentioned by the client. The priority value may be between 0 and 9 where 0(zero) holds the highest priority and 9 holds the lowest priority. The priority scale may vary depending on the organization where the methodology is being implemented. Once the priorities are set, the requirements are tabulated with their priority value against them and then presented to the client/customer/user for validation and verification. Assuming following priorities as mentioned in Table II is taken into consideration for the Requirement Statements A, B, C, D, E. [6][7]

TABLE II
Requirements Statements and Priorities

Requirement Statements	**Priority**
Requirement Statement A	5
Requirement Statement B	5
Requirement Statement C	3
Requirement Statement D	9
Requirement Statement E	7

C. Set Abstraction Level

Once the requirement priorities are revised by the user the team responsible for implementing the iteration undertakes proper requirements analysis mark each requirement with a value that indicates the abstraction level. The scale of the values here may again vary between 0 and 9, where 9(nine) is considered as the highest level of abstraction and 0 (zero) is considered as the lowest level of abstraction. The values are then added to the previous table and then again revised to reduce chances of error. Table III shows the priority and the abstraction level for each requirement statements. [6][7]

TABLE III
Requirements Statements, Priorities and Abstractions

Requirement Statements	**Priority**	**Abstraction**
Requirement Statement A	3	4
Requirement Statement B	5	2
Requirement Statement C	3	6
Requirement Statement D	9	1
Requirement Statement E	7	1

D. Plot Pareto Indifference Curve

An indifference curve is drawn to predict customer/client priorities and preferences and these are drawn against two different values which in this case are requirement priorities and abstraction level of the requirement. The following indifference curve as shown in Fig 2. is generated using the data in Table III. [3][4]

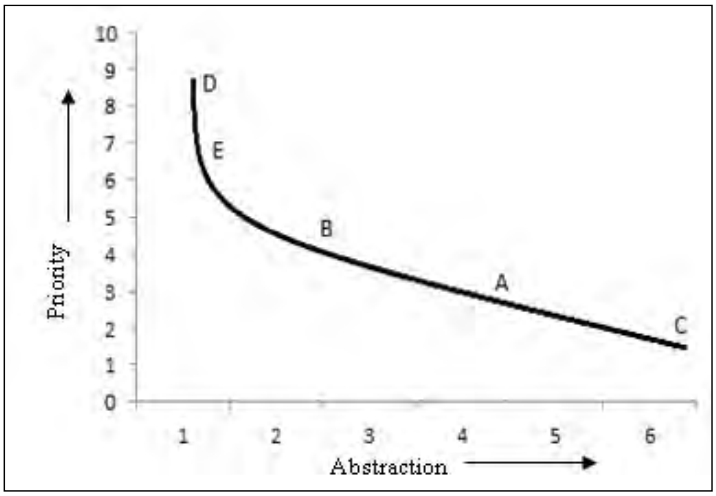

Fig. 2. Pareto Indifference Curve

E. Choose Important Requirements

Once the indifference curve is plotted the next step is to choose the requirements which are more important and feasible for rapid development. This can be done by choosing the requirements which are higher priority and lower abstraction. Requirements falling in such region should be marked or shaded or identified in the indifference curve as shown in Fig 3. [6]

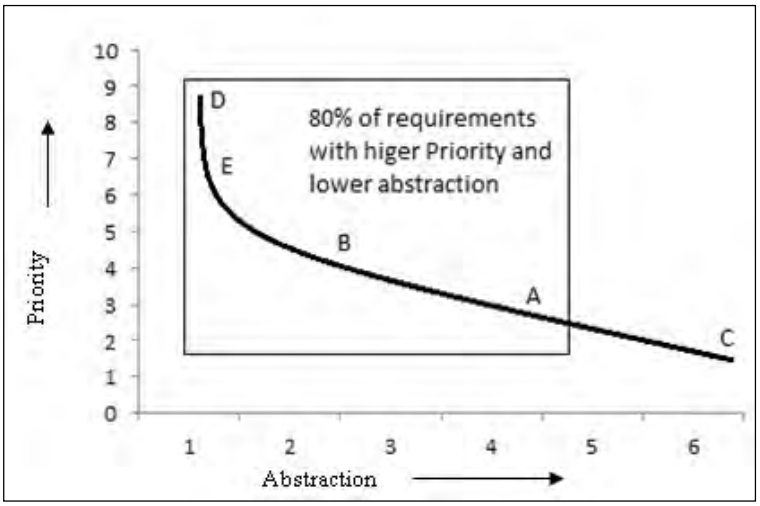

Fig. 3. Pareto Indifference Curve with requirement prioritization

F. Divide Requirements

Based on the markings done on the curve the entire set of requirements is divided in two parts. The first part contains 80% of the requirements having higher priority and lower abstraction level and the rest needs to be kept in the second part as shown in Table IV. [1][2]

TABLE IV
Division of Requirements

Parts	Requirements
Part 1	A, B, D, E
Part 2	C

G. Time Allotment

The time allotment is done on the basis of a small assumption that "The lesser the abstraction the easier and faster the implementation". Hence the first part of the requirements set which contains the 80% of the entire requirement are allotted 20% of time and the rest 80% of time is allotted to the remaining 20% of the requirements. This allotment of time is done in case of each and every task in all the phases of the RAD model henceforth.

Once the above mentioned seven steps are completed the remaining tasks such as designing and construction can be done in the traditional Rapid Application Development process Model, keeping in mind the fact that each task related to the 1st part of the requirements are completed in the allotted 20% timeframe. [3]

V. SUCCESS MEASUREMENT

Going forward in the development, the requirements will be changed to modules. Hence to check the success principle of economics called Pareto Efficiency is used. While testing each module it is extremely important to find out if one module is compromising or disturbing another module. When a module is implemented successfully in code without compromising another module it is said to be Pareto Efficient. To calculate te Pareto efficiency percentage (PEP) the following formula is used:

$$PEP = \frac{En}{n} \times 100$$

Where
PEP=Pareto efficiency percentage
En= Number of Pareto Efficient modules
n= Total number of modules.

Hence the Pareto efficiency percentage can be calculated if the number of Pareto efficient module in a project is known. Also, it can be concluded that "higher the Pareto efficiency percentage the chances of developing a successful project is higher. If the Pareto efficiency comes out to be too less modules which are not Pareto efficient need to be added to the next iteration for revision. [6][3]

VI. RESULTS AND DISCUSSION

The following differences in traditional Rapid Application Development process and the proposed model are identified as shown in Table V. [4][5]

TABLE V
Comparison of Traditional RAD Model and the Proposed Model

Traditional RAD model	**Proposed Model**
There is no provision for prioritizing the requirements by the user or client.	There exists provision to prioritize requirements by the user/client.
There is no provision for setting the abstraction level.	There exists Provision for setting the abstraction level.
No inbuilt concept of indifference graph.	Indifference graph is created on the basis of Abstraction and priorities.
Requirements set are not split into different parts.	Requirements set are divided into two parts on the basis of abstraction and priority.
Requirements may be given uneven time as decided by the team.	Requirements having higher priority and lower abstraction is given 20% of time.
No formal way for deciding the most important requirements.	Introduces a simple but definite way to decide important requirements.
No definite way for checking continuous success.	Success continuously measured with the help of Pareto efficiency percentage.

VII. CONCLUSION

Pareto Principle is a basic rule of thumb in economics which when applied to the Rapid Application Development process model of software engineering gives more definite values to the requirements. These definite values help in identifying the most important requirement which is hence given more focus that would provide a better product compared to the traditional Rapid Application Development model. There is also a provision to keep a check of the success of the current module using the Pareto efficiency percentage. This provides a better management of all modules and implementations. Hence it can be concluded that the proposed model is better than the traditional model**.**

ACKNOWLEDGMENT

The authors would like to thank the faculties of the School of Information Technology and Engineering, VIT University for their support and guidance.

REFERENCES

[1] Yang Wu; Ying Peng, Digital Object Identifier, 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC), pp. 1612 - 1615.
[2] Iqbal, M., Rizwan, M, Application of 80/20 rule in software engineering Waterfall Model, 2nd International Conference on Information and Communication Technologies, 2009, ICICT '09.
[3] Li, B.; Jiang, W.S, "Heuristics genetic algorithm using 80/20 rule," IEEE International Conference on Industrial Technology ICIT 96, pp. 436 – 438.
[4] R. Obermaisser, ; P. Peti, "A Framework for Rapid Application Development of Distributed Embedded Real-Time Systems," IEEE, Vienna University of Technology Vienna Austria, Sept 2003.
[5] Nguyen Hoang Thuan, Jan L.G.Dietz, Tran Van Lang, "Combining DEMO models with RAD's techniques in the analysis phase of software development process", IEEE RIVF International Conference, Nov 2010.
[6] XuZhen , Sun Jizhou , Wu Huabei , MengXiaojing , Tang Shanjiang, "Visual Model-Driven Rapid Development Tool suite for Parallel Applications", WRI World Congress, 2009.
[7] Lican Huang; JianfengNie, "Using Pareto Principle to Improve Efficiency for Selection of Qos Web Services", 7th IEEE Consumer Communications and Networking Conference (CCNC), 2010.
[8] "Lessons learned from 13 failed software products", http://www.successfulsoftware.net/2010/05/27/learning-lessons-from-13-failed-software-products, accessed march 2013.