ISSN:0975-9646

Empirical Analysis for Web Based Projects - Software Components Reuse Techniques

[1] Prof. Vuppu Padmakar [2] Dr. B V Ramana Murthy,

[1] *Department of Computer Science and Engineering, Chilkur Balaji Institute of Technology, Hyderabad, India*
[2] *Department of Computer Science and Engineering, Jyotishmathi College of Technology and Science, Shamirpet, Hyderabad, India*

Abstract-The basic connect of systematic software reuse is simple. Develop systems of components of a reasonable size and reuse them. Then extend the idea of component system beyond code alone to requirements, analysis models, design, and test. All the stages of the software development process are subject to reuse. Developers can save problem- solving effort all along the development chain. They can minimize redundant work. They can enhance the reliability of their work because each reused component system has already been reviewed and inspected in the course of its original development. Code components have passed unit and system test elsewhere and often have stood the test of use in the field. By these means developers can reduce development time from years to months, or to weeks instead of months.

Keywords-Component, web engineering, azail

[1] INTRODUCTION & RELATED WORK.

The experience at companies such as AT & T, Brooklyn Union Gas, Ericsson, GTE, Hewlett-Packard, IBM, Motorola, NEC and Toshiba show that significant cost and time savings result from systematic reuse. Other companies, those that are doing nothing in particular about reuse, provide a base line.
Several organizations have obtained reuse levels around 90% in certain projects or areas:
AT & T: 40 – 92 % in Telecom operation support system software.
Brooklyn Union Gas: 90 – 95 % in a process layer, and 67% in a user interface and business Object layer.
Ericsson AXE: 90 % in hundreds of customer-specific configurations.
Motorola: 85 % reuse and a 10: 1 productivity savings ratio in compiler and compiler-tool test Suites.
Many organizations have achieved through reuse persuades us that management may expect substantial gains:
Time to market: reductions of 2 to 5 times
Defect density: reductions of 5 to 10 times
Maintenance cost: reductions of 5 to 10 times
Overall software development cost: reduction of around 15% to as much as 75% for long-term projects.

Components are sometimes referred to as assets or work products. While the terms refer to the same underlying reality, they carry somewhat different connotations. Components suggests interfaces and packaging Asset brings to mind matters of ownership and management work product highlights the fact that a components is a unit in a cycle of work, the software life cycle.

Revolution in application development: The growing popularity and availability of component-based software technologies is fueling a change in the habits and expectations of millions of programmers. New application development tools and technologies have made components the key to reusing larger grained objects to build application rapidly. These technologies include Micro Soft Visual Basic, ActiveX and OLE, SUN's JAVA and CORBA interface definition language. Internet computing using applets and scripting languages such as VB Scripts and Java Scripts make it easy to develop and quickly deploy novel interactive applications across the enterprise. Component objects models and distributed computing infrastructure in the form of OMGs. CORBA middle ware technologies or Microsoft operating system support for the distributed components object model and OLE technology enable more complex distributed large-grain objects and components to e used. These technologies define and mange component interfaces separately from component implementations.
Practical reuse has also been quite successful with non-object-oriented languages such as COBAL and FORTRAN. These non-object-oriented components-based technologies reinter face the fact the successful reuse is not really about object oriented languages or class libraries. While object oriented languages have many of the qualities sought when developing components, they are not sufficient in themselves. There is a growing commercial market for components providing larger chunks of functionality than typical object classes do called ActiveX components or OLE components OCXs.
As an increasing number of these component-based applications are constructed and deployed by independent developers. Business objects and components will be defined and constructed by separate groups, yet must work together to meet business information system needs.

Systematic Approaches:

1. Engineering
2. Process
3. Organizational
4. Business-oriented

1. **Engineering:** The technology and methods deficiencies include

a) Means to identify clearly elements of the models that describe requirements, architecture, analysis, design, test, and implementation along the development stream. Clear identification underlines that ability either to reuse them or to allow them to be candidates for replacement by reusable component systems.
b) Lack of components to reuse. This category covers a host of obstacles: failure to select and strengthen components for reuse in the first place lack of techniques to package, document, classify, and identify components inadequate design and implementation of library systems poor access to component libraries for potential reuses.
c) Lack of flexibility in potentially reusable components if a component is rigid; it fits few or sometimes no reuse opportunities. In the past our methods for designing a flexible, layered architecture have been immature. Our ability to adapt a component to fit a new need or a new architecture has been limited.
d) Lack of tools to carry out reuse procedures. A number of new tools are needed tools that can be integrated into reuse-oriented support environments.

2. **Process:** In the engineering and technology level, the traditional process of software development is itself deficient in opportunities to encourage reuse. Nowhere in most of the processes that are used to day is there a point where developer sit down and ask themselves. The potential role of the architect in reuse has not been defined. Similarly, the role of a reuse engineer or a reusable component engineer has not been worked out. The places in the process at which developers might consider inserting component systems have not been built in. after analysis, design, or code components have been blocked out, review, inspection and walkthrough procedures fail to contemplate reuse.

3. **Organizational:** very few organizations systematically practice reuse as an established best practice. One reason is that they focus on one project at a time. Reuse requires a border focus. The management group has to look ahead, focusing on a set of projects that cover an application area, that is, that they believe process some characteristics in common. This area is a domain from this domain someone – a domain engineer has to identify that reusable elements and carry on from there.

4. **Business:** Reuse takes capital and funding it takes capital to finance domain engineering. The building of components systems strong enough to justify reuse, and the creation of in house libraries of components. These operations tie up capital until projects that reuse the components pay for them. It takes funding to provide education, training and access to vendor supplied components. It takes money to cope with an unstable domain as when the initial domain is poorly defined or when similar domains in different organizations must be merged. It may take money to penetrate the legal and social reasons for sharing or not sharing software.

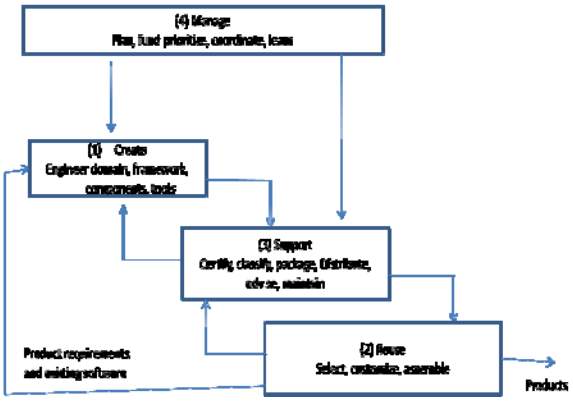
[2] REUSE INVOLVE CONCURRENT PROCESSES

The reuse community has come to understand on the basis of its experience that making systematic reuse effective requires major changes in the way organizations develop software. In the past the software process has focused on developing each application from scratch. At most, individual developers have shared code on an ad hoc basis.

The new way links many application development projects with processes that identify and create reusable assets. To do so, they must overhaul their business and organizational structures. We have come to understand that this significant organizational change can be thought of in terms of business process reengineering. It is rethinking of everything pertaining to software from there stand point of those who ultimately benefit from good software obtained quickly reliably and inexpensively.

Substantial reuse requires, first of all, that reusable assets be identified in terms of a system architecture. Then the assets must be created and appropriately packaged and stocked. Potential users must have confidence in the components integrity, secondly an organization must refashion its systems engineering process so that developer can identify opportunities for reuse and work selected components into the process.

Systematic software reuse is thus the purposeful creation, management, support, and reuse of assets. As illustrated in figure below this can be expressed in terms of four concurrent processes. We call the people in the reusable asset processes, creators, and those in the development projects, reusers.



Create: This process identifies and provides reusable assets appropriate to the needs of the reusers. These assets may be new, reengineered, or purchased of various kinds such as

code, interfaces, architectures, tests, tools and so on. This process may include activities such as inventory and analysis of existing applications and assets, domain analysis, architecture, definition, assessment of reusers needs technology evolution reusable asset testing and packaging.

Reuse This process uses the reusable assets to produce applications or products. Activities include the examination of domain models and reusable assets, the collection and analysis of end-user needs the design and implementation of additional components adaptation of provided assets, and the construction and testing of complete applications.

Support: This process supports the overall set of processes and manages and maintains the reusable asset collection. Activities may include the certification of submitted reusable assets. Classification and indexing in some library, announcing and distributing the asset, providing additional documentation, collecting feedback and defect reports from reusers.

Manage: This process plans, initiates, resources, tracks and coordinates the other processes. Activities include setting priorities and schedules for new asset construction, analyzing the impact and resolving conflict concerning alternative routes when a needed asset is not available, establishing training and setting direction.

Domain engineering:

In most reuse programs to date, a key activity associated with the create process is a fairly systematic way of identifying potentially reusable assets, and an architecture to enable their reuse. This activity is called domain engineering in the systematic reuse community. The development of reuse process is also sometimes called application system engineering. The essence of systematic software reuse is that initial investment by the creator to identify and carefully structure reusable assets will enable reusers to build application rapidly and cost effectively.

Domain engineering reflects the idea that sharing between related applications occurs in one or more application domain or problem domain or solution domains. Reuse of the assets then occurs during a subsequent application system engineering phase.

Sometimes domain engineering has been loosely described as just lke ordinary systems engineering such as structure analysis structured design or object oriented analysis object oriented design except that it applies to a family of systems rather than just one. It is like systems engineering but it is also more that one of kind systems engineering. It seeks the family of similar systems that can inhabit a domain. As a result domain engineering is more complex that established systems engineering. Therefore management should not turn to it without forethought and should establish domain engineering. Therefore management should not turn to it without forethought and should establish domain engineering only when it foresees a business benefit in reuse.

Application System Engineering:

This activity has long existed in the form of building applications from scratch, possibly with the aid of a few back pocket programs. The goal now is to make use of the extensive set of reusable assets that have been provided. The intent is to build the application much more rapidly and cost effectively.

Application system engineering specializes and assembles these components into application. These applications are largely constrained to fit the architecture and the components. Typical applications usually consist of components from several different sets of components.

Starting from the models of the architecture and reusable components, the reusers puts together available reusable assets to meet at least the bulk of the new set of requirements. This is sometimes called a delta implementation because it is an outgrowth of what already exists.

The reusers have to find and specialize components by exploiting a variability mechanisms provided. If it is not possible to meet all the new requirements with the available reusable components additional programming will be needed. This programming may be done by the creator, producing new reusable components or by the reusers.

Finally the components are integrated and the application tested.

Domain engineering	Application system Engineering
Define and scope domain	Do delta analysis and design relative to domain model and architecture
Analysis examples needs trends	Use component systems as starting point
Develop domain model and architecture	Find specialize and integrate components
Structure commonality and variability	Exploit variability mechanism language generators.
Engineer reusable component systems languages and tools	

[3] MOTIVATION AND BACKGROUND

Web-based systems [1] and applications deliver a complex array of content and functionality, to a broad population of end-users. Web engineering is the process that is used to create high-quality web applications. Web engineering is not a perfect clone of a software engineering, but it borrows many of software engineering fundamental concepts and principles. In addition, the web engineering process emphasizes similar technical and management activities are conducted, but the overriding Philosophy dictates a disciplined approach to the development of a computer-based system. Web engineers and non-technical content developers [2] create the web applications. As web becomes increasingly

integrated in business strategies, for small and large companies, the need to build reliable, usable, and adaptable systems grows in importance.

Contrary to popular belief, architecture is an important aspect of agile software development efforts, just like traditional efforts, and is a critical part of scaling agile approaches to meet the real-world needs of modern organizations. But, agile approach architecture a bit differently than traditionalists do architecture provides the foundation from which systems are built and an architectural model defines the vision on which your architecture is based. The scope of architecture can be that of a single application, of a family of applications, for an organization, or for an infrastructure such as the Internet that is shared by many organizations. Regardless of the scope, my experience is that you can take an agile approach to the modeling, development, and evolution of architecture.

An agile approach: Focus on people, not technology or techniques [3]

Keep it simple

Work iteratively and incrementally

Roll up your sleeves

Build it before you talk about it

An Agile Approach

First and foremost, the values, principles, and practices of Agile Modeling (AM) should help to guide your enterprise architecture modeling and documentation efforts. This is just a good start though these issues are:

Focus on people, not technology or techniques

Keep it simple

Work iteratively and incrementally

Roll up your sleeves

Look at the whole picture

Make enterprise architecture attractive to your customers

Potential Problems With The Agile Approach

No approach is perfect, including this one. We would like to address the issues:

It does not include an explicit way to ensure compliancy (although having enterprise architects embedded on the teams goes a long way towards this). It depends on people being responsible.

It requires you to actively strive to keep things simple.

It requires you to accept an agile approach to modeling and documentation[8].

Web Engineering

The World Wide Web and the Internet that empowers it are arguably the most important developments in the history of computing. The technologies have drawn us all into the information age. They have become integral to daily life in the first decade of the twenty-first century. For those who can remember a world without the web, the chaotic growth of the technology hackers backs to another era- the early days of software.

[4] ATTRIBUTES OF WEB-BASED SYSTEMS AND APPLICATIONS:

In the early days of World Wide Web "Web Sites" consisted of little more than a set of linked hypertext files, that presented information, using text and limited graphics. As time passed, HTML was augmented by Development Tools (e.g. XML, Java) that enabled web engineers, to provide computing capability along with information. Thus web-based systems and applications were born. Today, web application has evolved into sophisticated computing tools that not only provide standalone function to the end user, but also have been integrated with corporate databases and business applications[10]. There is little debate that web applications are different from many other categories of computer software. Powell summarizes the primary differences, when he states that web-based systems "involve a mixture between print publishing and software development, between marketing and computing, between internal communications and external relations, and between art and technology." . The following attributes are encountered in the web applications.

Network Intensiveness: A web application resides on a network and must serve the needs of a diverse community of clients.

Concurrency: A large number of users may access the web application at one time. In many cases, the pattern of usage among end-user will vary greatly.

Unpredictable Load: The number of users of the web application may vary by orders of magnitude, from day to day.

Performance: If a web application user must wait too long, he or she may decide to go elsewhere.

Availability: Users of popular web application often demand access on a "24/7/365" basis.

Data Driven: The primary function of many web applications is to use hypermedia to present text, graphics, audio, and video content to the end-user. In addition, web applications are commonly used to access information that exists on databases that were not originally an integrated part of the web-based environment.

Content Sensitive: The quality and aesthetic nature of content remains an important determinant of the quality of a web application.

Continuous Evolution: Unlike conventional application software that evolves over a series of planned, chronologically spaced releases, web application evolves continuously.

Immediacy: The compiling needs to get software to market quickly. It is a characteristic of many application domains.

Security: Because web applications are available via network access, it is difficult, if not impossible, to limit the population of end-users who may access the application. In order to protect the sensitive content and provide secure modes of data transmission, strong security measures must be implemented throughout the infrastructure, that supports a web application and within the application itself.

Web Application Engineering Layers: The development of web-based systems and applications incorporates Specialized Process Models, Software Engineering Methods adapted to the characteristics of web application development and a set of important enabling technologies. Process, methods and technologies provide a layered approach to web engineering that is conceptually identical to the software engineering layer.

Process: Web engineering process embraces the agile development philosophy [9].

Agile development emphasizes a lean development approach that incorporates rapid development cycles. Aoyama [64] describes the motivation for the agile approach in the following manner: The Internet changed software development's [65] top priority from what to when. Reduced time-to-market has become the competitive edge that leading companies strive for. Thus, reducing the development cycle is, now, one of the software engineering's most important missions. Even when rapid cycle times dominate development thinking, it is important to recognize that the problem must still be analyzed, a design should be developed, implementation should proceed in an incremental fashion, and an organized testing approach must be initiated. However, these framework activities must be defined within a process that (1) embraces change (2) encourages the creativity and independence of development staff and strong interaction with web application stakeholders, (3) builds systems using small development teams, and (4) emphasizes evolutionary or incremental development, using short development cycles.

Web Methods: The web engineering methods landscape encompasses a set of technical tasks that enable a web engineer to understand, characterize, and then build a high-quality web application. Web methods can be categorized in the following manner:

Communication Methods: Communication Methods define the approach used to facilitate communication between web engineers and all other web application stakeholders. Communication techniques are particularly important during requirements gathering and whenever a web application increment is to be evaluated.

Requirements Analysis: They provide a basis for understanding the content to be delivered by a web application, the function to be provided for the end-user and the modes of interaction that each class of user will require for navigation through the web application.

Design Methods: They encompass a series of design techniques that address web application content, application and information architecture, and interface design and navigation structure.

Testing Methods: They incorporate formal technical reviews of both the content and design model and a wide array of testing techniques that address component level and architectural issues, navigation testing, usability testing and configuration testing.

Tools and Technology: They encompass a wide array of content description and modeling languages.

Web Engineering Framework: To be effective, any engineering process must be adaptable[5]. That is, the organization of the project team, the modes of communication among team members, the engineering activities and tasks to be performed, the information that is collected and created and themethods used to produce a high-quality product must all be adapted to the people doing the work.

Web Application is often delivered incrementally: That is, framework activities will occur repeatedly as each increment is engineered and delivered.

Changes will occur frequently: These changes may occur as a result of the evaluation of a delivered increment or as a consequence of changing business condition.

Business Analysis: Business analysis defines the business/organizational context for the web application. In addition, stakeholders are identified, potential changes in business environment or requirements are predicted and integration between the web application and other business applications, databases and functions are also designed.

Formulation: Formulation is a requirements gathering activity, involving all stakeholders. The intent is to describe the problem that the web application is to solve using the best information available. In addition an attempt is made to identify areas of uncertainty and where potential changes will occur.

Planning: The project plan for the web application increment is created. The plan consists of a task definition and a timeline schedule for the time period, projected for the development of the web application increment.

Modeling: Conventional Software Engineering Analysis and Design tasks are adapted to web application development, merged and then into the web engineering modeling activity. The intent is to develop "rapid" analysis and design models that define requirements and at the same time, represent a web application that will satisfy them.

Construction: Web engineering tools and technology are applied to construct the web application that has been modeled. Once the web application increment has been constructed, a series of rapid tests are conducted to ensure that errors in designs are uncovered.

Homepage: Web application should contain useful information or a simple listing of links that lead a user to more detail at lower level.

Page Layout: It varies depending upon the type of web application being developed.

Multimedia: Multimedia options are effective options for web application.

Engineering Best Practices: Web engineering teams are, sometimes, under enormous time pressure and will try to take short-cuts, but a set of fundamental best practices adopted from the software engineering practices should be applied, if industry quality web applications are to be built[7].

Product Objectives: It is essential to understand the business needs and product objectives, even if the details of the web applications are vague: Many web application developers, erroneously, believe that vague requirements relieve them from the need to be sure that the system, they are about to engineer, has a legitimate business purpose. The end result is good technical work that results in the wrong system, built for the wrong reasons, for the wrong audience.

If stakeholders cannot enunciate a business need for the web application, proceed with extreme caution. If stakeholders struggle to identify a set of clear objectives for the product, do not proceed until they can.

User Interaction: The user interaction with the web application should be described using a scenario-based approach: stakeholders must be convinced to develop use-cases to request how various actors will interact with the web application. These scenarios can then be used for project planning and tracking to guide analysis and design modeling, and as important input for the design of tests.

Project Plan: Project Plan, should be developed even if it is very brief. Then the plan has to be based on a predefined process framework that is acceptable to all stakeholders. Because project timeliness is short, schedule granularity should be fine.

Modeling: Modeling demands time spending to ascertain what it is being done to build: Generally, comprehensive analysis and design models are not developed during web engineering. However, UML class and sequence diagrams along with other selected UML notation may provide invaluable insight.

Review the models for consistency and quality: Formal technical reviews should be conducted throughout a web-engineering project. The time spent on reviews pays important dividends because; it often eliminates rework and results in a web application that exhibits high quality, thereby increasing customer satisfaction.

Tools and Technology: Tools and Technology that enable to construct the system with as many reusable components as possible: A wide array of web application tools is available for virtually every aspect of web application construction. Many of these tools enable a web engineer to build significant portions of the application using reusable components.

Testing: Don't rely on early users to debug the web application design comprehensive test and execute them before releasing the system. Users of a web application will often give it one chance. If it fails to

Perform, they move elsewhere-never to return. It is for this reason that "test first, then deploy" should be an overriding philosophy, even if deadlines must be stretched.

Other Software: Web application software is different from other categories of computer software.

Objectives: If stakeholders struggle to identify a set of clear objectives for the product, it is not advisable proceed, until they can.

Generic Process: The generic process framework-communication, planning, modeling, and deployment are applicable to web engineering[6].

Implementation: Having understood the significance of web engineering for successful development of web applications, the fundamental question that arises, for any web developer is, the concepts relevant for agile, various attributes that are vital for designing web applications, their relative significance and the emphasis required for each attribute, for the successful implementation of the web application using agile methodology. The present study is a humble beginning, to explore the concepts of web attributes and the agile & component technology methodology.

Keeping in view of the elements of web engineering and the process of developing web based projects based on different approaches like

1.Traditional T: Uses traditional approach –Waterfall model
2. Application A: Uses Programming Languages from scratch
3. Component C:Uses component for reuse
4. Agile and Component based AC: Reuse & Incremental along with client interaction

Analytic Hierarchy Process: The Pair wise Comparison Method was developed by Saaty (1980) in the context of the Analytic Hierarchy Process (AHP). This method involves pairwise comparisons to create a ratio matrix. It takes, as input, the pair wise comparisons and produces the relative weights as output

[5] DEVELOPMENT OF THE PAIR WISE COMPARISON MATRIX:

The method employs an Underlying scale with values ranging from 1 to 9 to rate the relative preferences for two criteria (see table).

Intensity of Importance	Definition
1	Equal importance
2	Equal to moderate importance
3	Moderate Importance
4	Moderate to strong importance
5	Strong importance
6	Strong to very strong importance
7	Very strong importance
8	Very to extremely strong importance
9	Extreme importance

Source: Saaty Scale for pair wise comparison

[6] DEVELOPMENT OF THE PAIR WISE COMPARISON MATRIX

This step involves the following operations:

(a) Sum the values in each column of the pair wise comparison matrix;

	T	A	C	A&C
T	1	1/3	1/7	1/9
A	3	1	1/7	1/8
C	7	7	1	1/9
A & C	9	8	9	1

(b) Divide each element in the matrix by its column total (the resulting matrix is referred to as the normalized pair wise comparison matrix);

	T	A	C	A&C
T	0.05	0.02	0.01	0.08
A	0.15	0.06	0.01	0.09
C	0.35	0.44	0.10	0.08
A C	0.45	0.50	0.88	0.75

Compute the average of the elements in each row of the normalized matrix, that is, divide the sum of normalized scores for each row .

	T	A	C	A&C	w
T	0.05	0.02	0.01	0.08	0.04
A	0.15	0.06	0.01	0.09	0.08
C	0.35	0.44	0.10	0.08	0.24
A C	0.45	0.50	0.88	0.75	0.65

These averages provide an estimate of the relative weights of the criteria being compared. AHP technique is used to find the weights of web based development approaches and the results were as follows

1.Traditional T: Uses traditional approach –Waterfall model

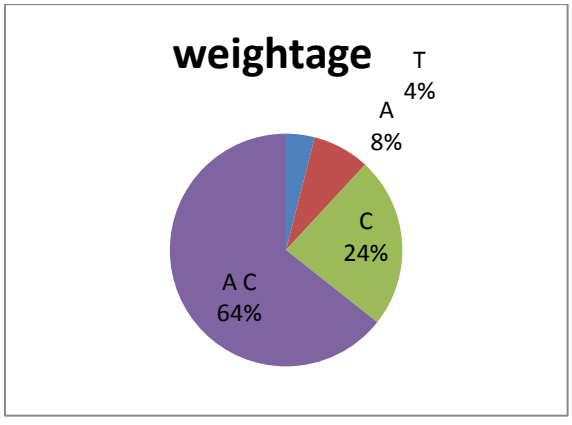
2. Application A:Uses Programming Languages from scratch

3. Component C:Uses component for reuse

4. Agile and Component based AC: Reuse & Incremental along with client interaction

Approach		Weights in %
Traditional	T	0.04
Application	A	0.08
Component	C	0.24
Agile and Component	AC	0.65

And the graph has been depicted for the above table which is as follows



The graph shows that the approach for web based projects is more significant for Agile and Component base which takes 65% of the weight age when compared to other approaches

[7] CONCLUSION

There are many SDLC models such as Agile, RAD and Waterfall etc. used in various organizations depending upon the conditions prevailing in it like v-model gives the verification and validation for organization and it is very useful for organization. All these different software development models have their own advantages and disadvantages. Nevertheless, in the contemporary commercial software development world, the fusion of all these methodologies is incorporated. Timing is very crucial in software development. If a delay happens in The development phase, the market could be taken over by the competitor. Also if a bug' filled product is launched in a short period of time (quicker than the competitors), it may affect the reputation of the company. So, there should be a tradeoff between the development time and the quality of the product. Customers don't expect a bug free product but they expect a User-friendly product that results in Customer Ecstasy!

REFERENCE

[1] Powell,T.A., Website Engineering Prentice Hall,1999.
[2] Pressman, R. S.,"Can Internet Based Applications be Engineered?" IEEE Software, September 1998, pp. 104-110.
[3] The Agile Alliiance Home Page ,hhtp//www.agilealliance.org/home
[4] Ambler ,S, "what is Agile Modelling http://www.agilemodeling.com/index.htm.
[5] Cockburn A.,Agile Software Development: Addison Weley
[6] Cockburn and J HighSmith ., What is Agile Software Development The People Factor "IEEE computing Vol 34 pp 131-133
[7] DeMarco ,.T,and T Listener .Peopleware second edition
[8] DeMarco ,.T and Boehm,"The Agile Method s Fray "IEEE Computer Vol 35 pp 90-92.
[9] HighSmith ., J. Agile Software Ecosystem Addision–Wesley.
[10] Highsmith J., "The Methodology Debate" Part -1 Vol 14.