

Process Centric Development to Improve Quality of Service (QoS) in Building Distributed Applications

K. Krishna Mohan¹, A.Srividya², A. K. Verma³ and Ravi Kumar Gedela⁴

Abstract - In a competitive business landscape, large organizations such as insurance companies, banks etc are under high pressure to innovate, improvise and differentiate their products and services while continuing to reduce the time-to market for new product introductions. For example, in banks operating multiple lines of businesses, generating a Single view of the customer is a critical from different perspectives due the systems developer over a period of time and existence of disconnected systems with an enterprise. Therefore to increase revenues and cost optimization, it is important build enterprise systems more closely with the business requirements with re-use of existing systems. While building distributed based applications, it is important to take into account the proven processes like Rational Unified Process (RUP) to mitigate the risks and increase the reliability of the system. Experience in developing applications in Java Enterprise Edition (JEE) with customized RUP is presented in this paper. RUP is adopted into an onsite-offshore development model along with ISO 9001 and SEI CMM Level 5 standards. This paper provides RUP approach to achieve increased reliability with higher productivity with lower defect density and competitiveness through cost effective custom software solutions.

Index Terms - Rational Unified Process, Java Enterprise Edition (JEE), Phases, Disciplines, Reliability, Testing, Artifacts, Use Case and Model, metrics, Offshore Development Center (ODC).

1. INTRODUCTION

development has expanded rapidly in recent years and has brought a wake of changes that impact application development projects [1]. The adoption of a new process for delivery excellence within an organization is critical to meet time-to-market conditions and it is a significant undertaking. It requires careful customization to match the organization culture, accommodate any existing procedures, and obtain buy-in among the key stakeholders and users of the process. Many organizations have initiated a program to standardize the software development process using rational tools and RUP. A natural extension of adoption of RUP would be extending the same to offshore development to reduce the total cost of

¹RS, Reliability Engineering Group, Department of Electrical Engineering, Indian Institute of Technology Bombay

²³Professor, Reliability Engineering Group, Department of Electrical Engineering, Indian Institute of Technology,

⁴HCU – SAP, Satyam Computer Services Limited, Hyderabad
E-Mail: ¹kkm@ee.iitb.ac.in, ²asvidya@ee.iitb.ac.in,

³akv@ee.iitb.ac.in and ⁴Ravikumar_Gedela@satyam.com

ownership (TCO) with improved reliability with higher productivity. However, with the comprehensive nature of RUP comes significant complexity regarding the process steps and types of artifacts produced at each step. The authors therefore created this paper to cover the elements of a RUP-based development process that are vital to a successful development projects.

2. BACKGROUND AND MOTIVATION

National Research Council Canada [2] and several other organizations reveal that the process oriented development is necessary to improve the reliability and productivity and decrease the cost, thereby increase the operational efficiency. This paper provides an approach to adopt RUP in building applications and focusing on various areas of software development. It discusses the approach citing examples of the work done by authors in two areas: Requirements Gathering/Modeling and testing (Manual and automation) phases.

A Proof-of-Concept (PoC) with financial domain application is developed and tested to address the RUP approach to demonstrate the benefits. The PoC was developed in SOA - J2EE platform with RUP and adoption of RUP for several projects the results obtained and conclusions are being shared in this paper.

3. GENERIC ENGAGEMENT MODEL (ONSITE-OFFSHORE MODEL WITHOUT RUP)

Based on the working experience of the authors it is well accepted that most of the service organizations follow the onsite-offshore model to reduce the development and maintenance costs. Development in India or other low cost centers (here after referred as offshore development) teams are effectively leveraged the onsite-offshore model to provide value based services. This model is true for most of the North American and European companies/customers.

Most of the organizations successful in following Iterative Incremental development methodology in order to meet the customer requirements through an early and continuous delivery of software to end users at regular intervals. The lifecycle in iterative development is composed of several iterations in sequence. Each iteration is a self-contained mini-release composed of activities such as requirements, analysis, design, development, and testing. The final system is built by adding and releasing new features in each iteration. Every iteration ends up with:

1. The delivery of stable release
2. A visual model of emerging product
3. Lessons learned to incorporate into next iteration
4. Customer sign-off on implementation of requirements

The diagram in figure 1 depicts the iterative and incremental development process that is being followed currently.

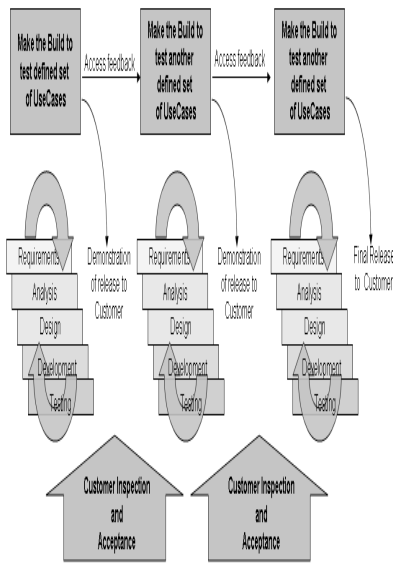


Figure 1: Generic Engagement Model

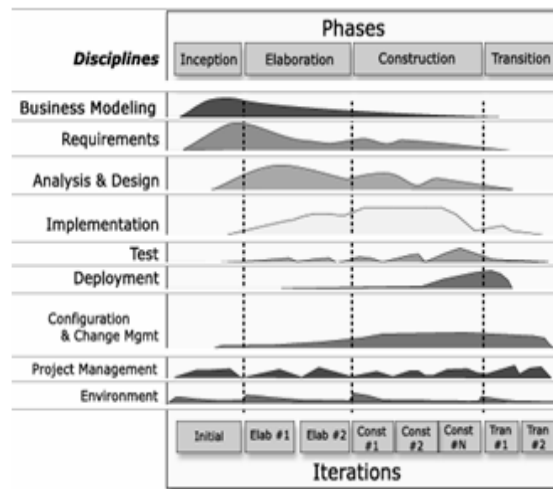
The key characteristics of such developments are:

1. The development is framework driven. It is necessary that the onsite and offshore team have understood the framework on which the development is based. The framework is in terms of the base architecture, definition of common components, clear understanding of the code and design practices and so on.
2. The onsite team manages the scope. Any new features are discussed and negotiated with the customer. They are put on wait till next release if they are impacting the schedule.
3. Involvement of business users in the development at early stages is key. From our experience, such projects are successful with full involvement of the business people. The onsite team's one of the main responsibility is to engage with business for clarifications of requirements and JAD sessions
4. Planning is done iteratively. There is a possibility that such development may fall in the "death by planning trap". The plan is iterative, the milestones are fixed and they are clearly communicated to all stakeholders. A high-level, coarse-grained view of the project is developed during the initial iteration. It shows the total number of planned iterations across all the phases, and key milestone dates for each of these iterations. A fine grained plan is made for each iteration.
5. There is lot of emphasis on separate test team. This team works with the development team and continuous testing is done during the development.
6. The configuration management plan is clearly laid out. The activities are more frequent. Generally, in such projects, the offshore team delivers code drops once in three days. The onsite team continuously integrates the code drops.

7. The product works from day one. The functionality may be very minimal but the product is built very frequently.
8. Effective risk management plan. The risks are monitored on weekly basis and are communicated to the customer.

4. RATIONAL UNIFIED PROCESS

RUP is a comprehensive framework; it is a more or less complete set of process elements that has to be tailored to each case as no project needs the complete set of elements. IBM Rational has in fact done some of the tailoring of the original Unified Process by the development of RUP [3]. The Rational Unified Process® [4,5] or RUP® product is a software engineering process. It provides a disciplined approach to assigning tasks and responsibilities within a development organization. Its goal is to ensure the production of high-quality software that meets the needs of its end users within a predictable schedule and budget.



Source: IBM Corporation
Figure 2: Rational Unified Process

The figure 2 illustrates the overall architecture of the RUP, which has two dimensions:

1. The horizontal axis represents time and shows the lifecycle aspects of the process as it unfolds. This first dimension illustrates the dynamic aspect of the process as it's enacted and is expressed in terms of phases, iterations, and milestones.
2. The vertical axis represents disciplines that logically group activities by nature. This second dimension portrays the static aspect of the process-how it's described in terms of process components, disciplines, activities, workflows, artifacts, and roles.

The graph shows how the emphasis varies over time. For example, in early iterations you spend more time on requirements; in later iterations you spend more time on implementation.

5. CUSTOMIZED RUP FOR ONSITE-OFFSHORE MODEL DELIVERY

As described above, RUP is iterative, use case based, Architecture driven process, it can be customized for an organization. The following Figure 3 customized for a leading bank in North America and which is operational.

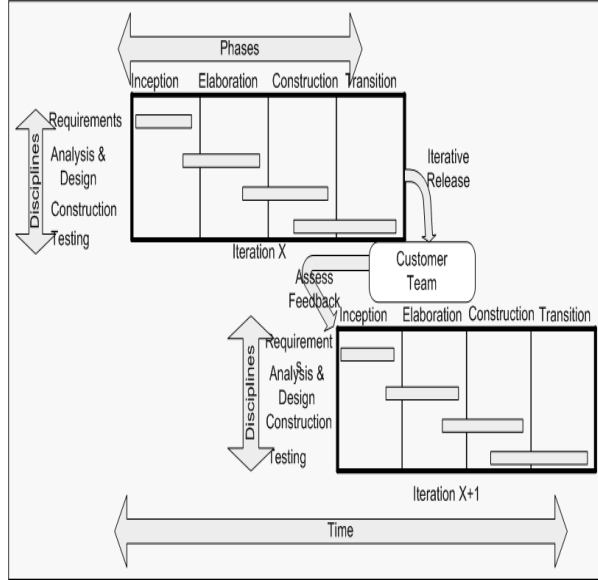


Figure 3: Customized RUP

Four phases are defined for a project lifecycle, which proceed in order:

- **Inception:** the beginning phase of the project with priorities on achieving concurrence among all stakeholders on the lifecycle objectives for the project.
- **Elaboration:** focuses on finalizing the system and software architecture and requirements
- **Construction:** building phase of the project that implements what's been laid out during the elaboration phase to produce an alpha-tested stable system
- **Transition:** final phase of the project that makes the system production ready and prepares the user community for use

The following diagram presents the RUP phase-end deliverables in an application environment. Note that these do not constitute the full set of artifacts normally produced during a project that uses RUP. Based on the previous engagements experience, authors suggests ensuring a reliable and repeatable development with the artifacts mentioned in Figure 4 at high level.

Project Entry	Inception	Elaboration	Construction	Transition
	Communication Plan	Architecture Document	Application Code	Application Deployment Plan
	Project Plan	Design Approach Document	Code Documentation	Project Build
	Resource Plan	Application Test Plan	Application Integration	Installation Artifacts
	Team Structure	Application Deployment Plan	Application Test Plan	
	Vision Document	Coding Standards	Application Deployment Plan	
	Wire Frames	Detailed Design Document		
	Risks List	Communication Plan		
	Base Requirement Document	Project Plan		
		Resource Plan		
		Team Structure		
		Base Requirement Document		
		Wire Frames		

Figure 4: RUP Phase-end deliverables

As described in the previous section, key activities are carried in different phases (Inception, Elaboration, Construction and Transition) of RUP as depicted in the above diagram. During the project delivery, offshore is viewed as an extended development facility. To leverage the onsite-offshore model and effective delivery efficiency, the set of activities are distributed in between onsite and offshore facilities.

During the Inception phase, the project vision is established based on the requirements from business users and deliver Vision document. The next key activity of constructing the use cases from the requirements that are captured in the Inception phase is carried out in the Elaboration phase. In this phase, the following activities are carried out.

1. Complete set of use cases (UC) will be planned against various iterations and deliver Use-Case document with Iteration Plan.
2. High level design and Low Level Design will be generated as indicated in the Fig 4.

To leverage the offshore felicity effectively, key activities are being carried out from the elaboration iteration (X+1) and construction (X) will be carried out simultaneously. The (X+1) iteration of Elaboration phase will be carried out at onsite and the X iteration of Construction will be carried out at Offshore. After completion of the construction iteration X, the transition phase will be initiated. During the Transition phase, Integration System Testing (IST) execution will be carried out. After the completion of initial iterations of Transition phase, (X+1) iteration of Construction phase would start. This phase will be followed by (X+1) iteration of Transition phase. After completion of planned iterations, Quality Assurance Test (QAT) will be conducted in an integrated environment (which labeled as Transition III in the diagram Fig 3). After meeting the required QAT criteria, the project will be deployed/ transitioned to production.

6. CASE STUDY

This section explains the work done in building the Proof-of-Concept of a financial services application.

To narrate the simplified version of the application use case

diagram is depicted. Use cases drive the whole development process. With each iteration, Use cases drive the work in analysis, design, implementation and test [6, 7]. This paper addresses the requirements phase and testing phase as the project is completely developed and currently in production and does not addresses the other phases in detail. The use case of the application is shown in the figure 5.

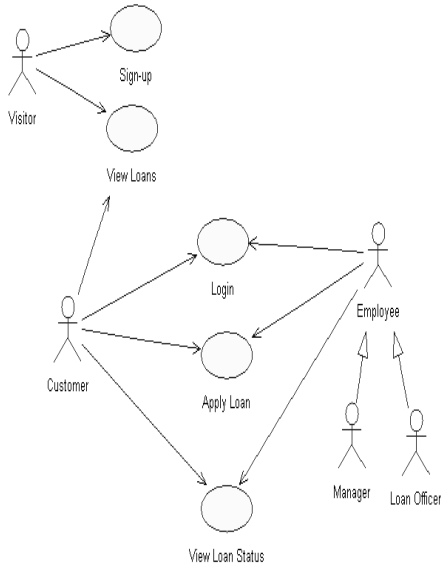


Figure 5: Use Cases of the application, few cases are considered for PoC

6.1 Logical Architecture of Credit Management Application with Design Patterns

The implementation model of the PoC with proven Design Patterns is as follows: The thin-client application users uses the browser to access the application. The request from the browser passes, on (over HTTP) to the Presentation Layer which implements the MVC design pattern. The industry wide proven open source framework ‘struts’ used to realize the presentation layer.

The Business Layer implements business processes for different modules as Java Objects and Enterprise Java Beans (EJB) which encapsulates the business logic. The Data Layer implements the Data Access Object (DAO) pattern. The data access components encapsulate the data sources from the business layer as shown in Figure 6.

6.2 Deployment of the Credit Management Application

The deployment of the Credit Management Application is as follows in the figure 7.

The application is developed in Java/J2EE running on Windows XP server and database is Oracle 9i. The IBM HTTP Server, IBM WebSphere Application Server and the Oracle Database were running on the on different systems system with Clustering. The clustering mechanism was chosen to demonstrate the Failover in the event if any WebSphere Application Server is down. The following Tables 2, 3, 4, 5, 6, 7 and 8 represent the application behavior at runtime.

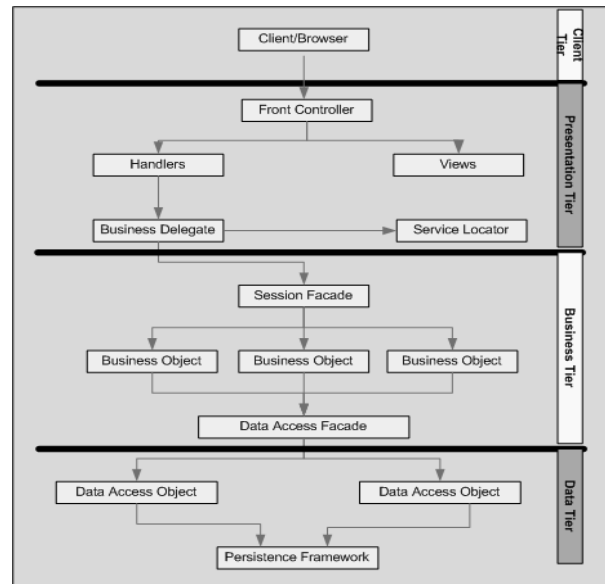


Figure 6: Logical Architecture of Trader Tax monitor with Design Patterns

Components and frameworks selected for J2EE version of PoC are provided in Table 1

Layer	Presentation	Business	Data
Technology Options	Jakarta Struts 1.2	Enterprise Java Beans (EJB) and POJOs	Hibernate

Table 1: J2EE selection

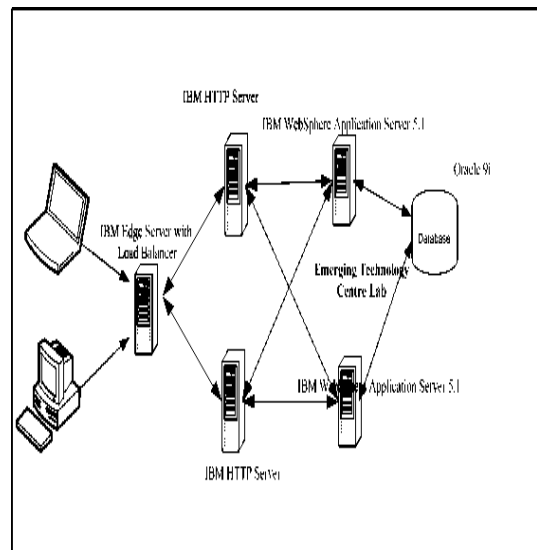


Figure 7: Deployment of the Credit Management Application

6.3 Database Design

CMS_USER_DETAILS			
Column Name	Type	Null?	Constraint
USER_ID	VARCHAR2(20)	NOT NULL	PRIMARY KEY
FIRST_NAME	VARCHAR2(20)	NOT NULL	
LAST_NAME	VARCHAR2(20)	NOT NULL	
STREET1	VARCHAR2(20)	NOT NULL	
STREET2	VARCHAR2(20)		
DISTRICT	VARCHAR2(20)	NOT NULL	
STATE	VARCHAR2(20)	NOT NULL	
COUNTRY	VARCHAR2(20)	NOT NULL	
EMAIL	VARCHAR2(20)		
PHONE	VARCHAR2(20)		
PIN	VARCHAR2(20)	NOT NULL	

Table 2

CMS_LOGIN			
Column Name	Type	Null?	Constraint
USER_ID	VARCHAR2(20)	NOT NULL	REFERENCES CMS_USER_DETAILS
PASSWORD	VARCHAR2(20)		
ROLE	VARCHAR2(20)		

Table 3

CMS_LOAN_DOCUMENTS			
Column Name	Type	Null?	Constraint
DOCUMENT_ID	NUMBER	NOT NULL	PRIMARY KEY
DOCUMENT_NAME	VARCHAR2(20)		
DOCUMENT_DESCRIPTION	VARCHAR2(20)		

Table 4

CMS_LOAN_TYPE			
Column Name	Type	Null?	Constraint
LOAN_TYPE_ID	NUMBER	NOT NULL	PRIMARY KEY
LOAN_TYPE	VARCHAR2(20)		
LOAN_DESCRIPTION	VARCHAR2(20)		

Table 5

CMS_LOAN_TYPE_DOCUMENTS			
Column Name	Type	Null?	Constraint
LOAN_TYPE_ID	NUMBER		REFERENCES CMS_LOAN_TYPE
DOCUMENT_ID	NUMBER		REFERENCES CMS_LOAN_DOCUMENTS

Table 6

CMS_LOAN_APPLICATION			
Column Name	Type	Null?	Constraint
LOAN_ID	VARCHAR2(20)	NOT NULL	PRIMARY KEY
LOAN_TYPE_ID	VARCHAR2(20)		REFERENCES CMS_LOAN_TYPE
USER_ID	VARCHAR2(20)		REFERENCES USER_DETAILS
LOAN_AMOUNT	VARCHAR2(20)	NOT NULL	
TENURE	VARCHAR2(20)	NOT NULL	
APP_DATE	VARCHAR2(20)	NOT NULL	
STATUS	VARCHAR2(20)		
OFFICER_APPROVAL_DATE	VARCHAR2(20)		
OFFICER_APPROVAL_REMARKS	VARCHAR2(20)		
OFFICER_SUGGESTED	VARCHAR2(20)		

MANAGE R_APPROVA L_DATE	VARC HAR2(20)		
MANAGE R_APPROVA L_REMARK	VARC HAR2(20)		
MANAGE R_SUGGEST ED_AMOUN T	VARC HAR2(20)		

Table 7

CMS_LOAN_TYPE			
Column Name	Type	Null?	Constraint
LOAN_T YPE_ID	NUMBER		REFERENCES CMS_LOAN_T YPE
DOCUME NT_ID	NUMBER		REFERENCES CMS_LOAN_D OCUMENTS
STATUS	VARCHA R(20)		

Table 8

6.4 Sequence diagrams

The following figures 8 – 11 are sequence diagrams for sign up, Login, Sign-out, Loan Application are shown below.

Sequence Diagram for Sign Up

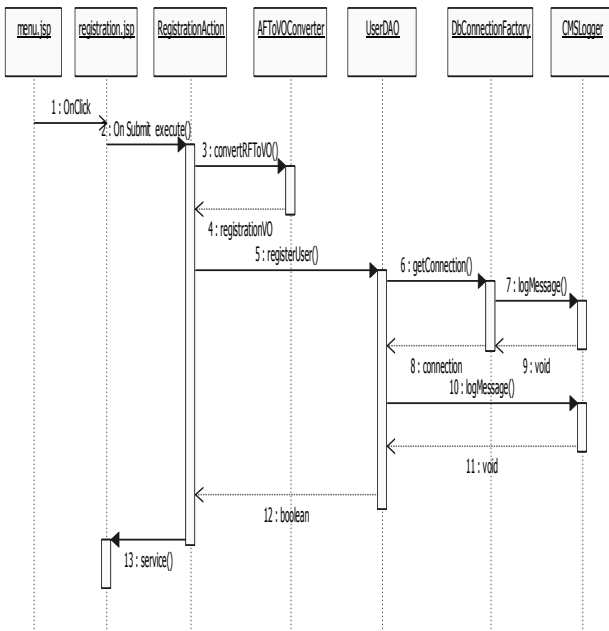


Figure 8: Sequence Diagram for Sign Up Login Sequence Diagram

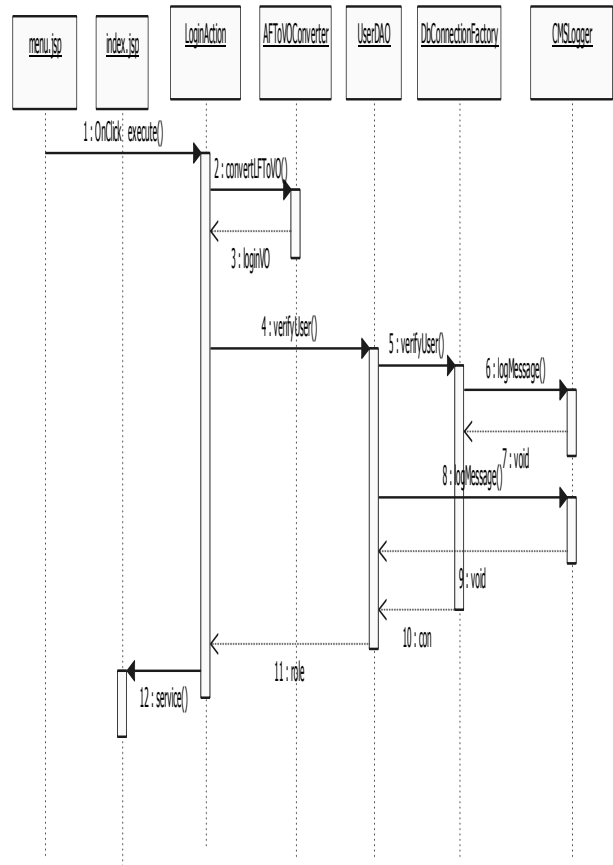


Figure 9: Sequence Diagram for Login

Sign-out

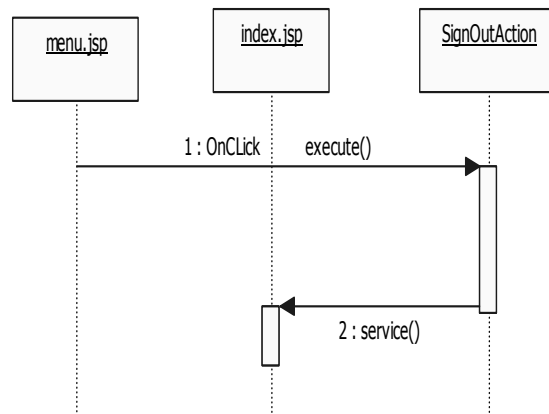


Figure 10: Sequence Diagram for Sign-out Loan Application

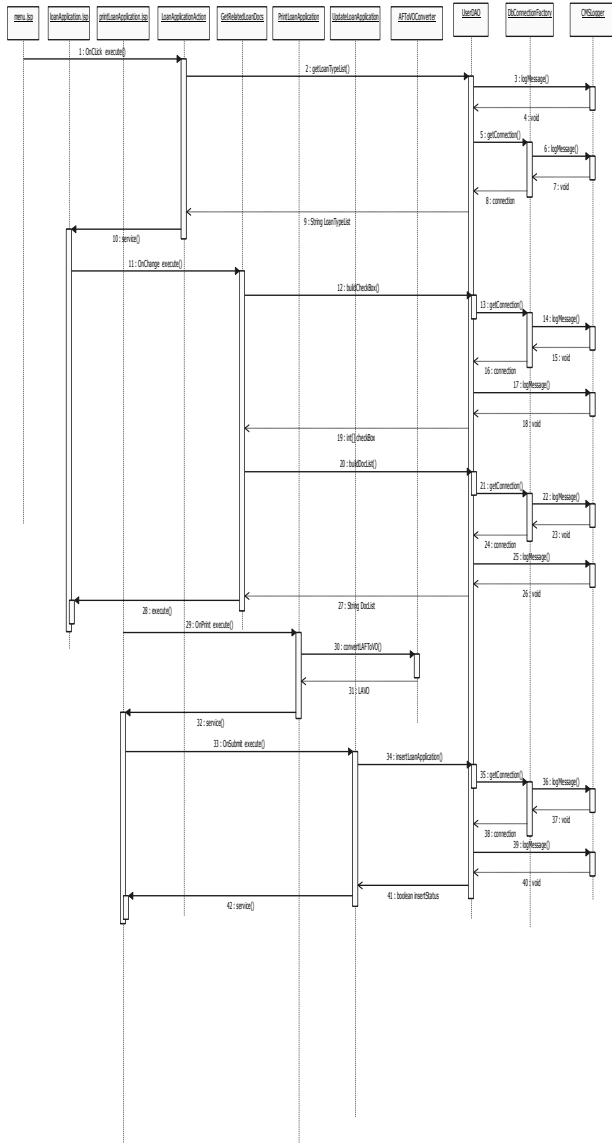


Figure 11: Sequence Diagram for Loan Application

7. REQUIREMENTS FLOW – RUP

The PoC is developed in Java/J2EE with strong object oriented principles. The authors suggest the following requirement flow to capture requirements.

As depicted by the Figure 12, the following major activities are executed for the requirements discipline

- **Analyze the Problem:** identify problem to be solved, system boundaries and constraints, and stakeholders.
- **Understand Stakeholder Needs:** elicit stakeholder requests or “wish list”
- **Identify Actors:** identify people, applications that will interact with the system
- **Find Use Case:** find system processes or set of behavior that product a specific result
- **Develop Use Cases:** document Use-Case diagrams,

Activity Diagrams, etc

- **Develop Supplemental Specifications:** define requirements that cannot be readily captured by the use-case model such as system, regulatory requirements, application standards, etc. These specifications are captured in the Master Requirements document
- **Refine System Definition:** detail use cases and model and prototype the use interface (wire-frame, etc)

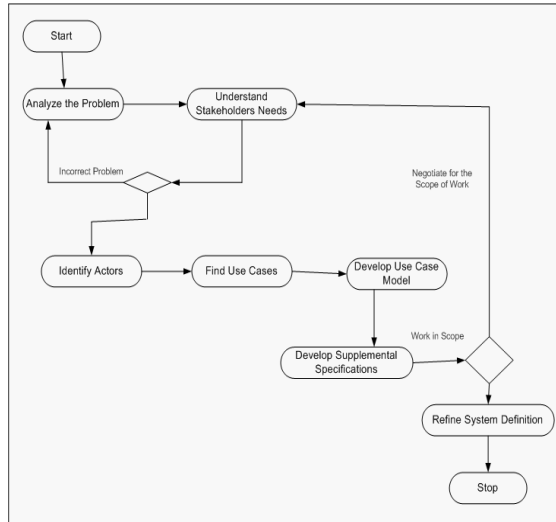


Figure 12: Requirements Flow

8. TESTING WORKFLOW

The following testing workflow was considered during the execution is depicted in the figure 13 and detailed test flow chart is depicted in figure 14.

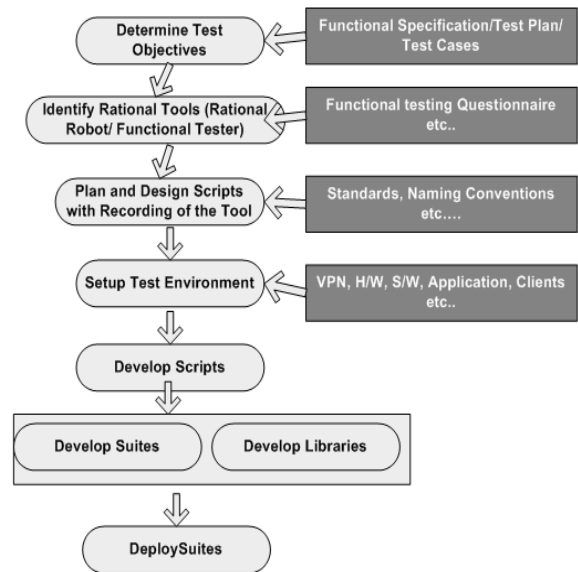


Figure 13: Testing workflow

Key elements to be noted are:

- This is a generic test process flow for various types of tests

to be executed during the project. Test method and type are determined after defining the test criteria.

- Testing can be executed on components, subsystems and the overall system, provided they are in a stable condition (stability)
- The test result will be tuned for another round of test cycle

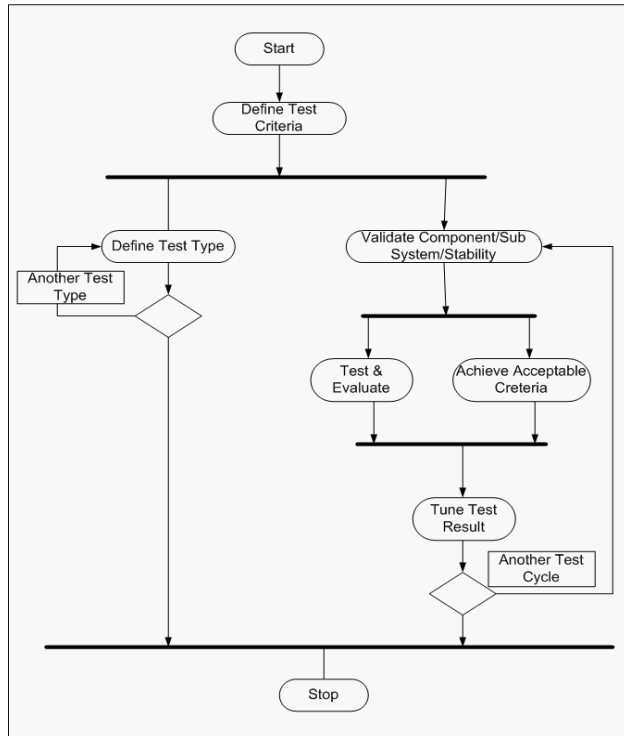


Figure 14: Test Detailed Flow chart

9. Analysis of Metrics

The detailed metric analysis has been performed on three different modules (EFT, REP, SD modules) over three cycles/builds with RUP implementation. The results obtained from the PoC, which is experimented for RUP implementation the number of defects are significantly reduced, in incremental cycles, which are analyzed in graphs depicted in figures 15 – 26 from data collected from the defect Consolidation log for three different modules EFT, REP and SD shown in table 9 - 11.

Defect Type: LG-Logic, CO-Computation, IF-Interface, UI-User interface, ST-Standards, CF-Configuration, DT-Data, SY-System, DC-Documentation, OT-Others

10. RESULTS AND REALIZED BENEFITS

Based on the results obtained from the prototype, it is evident that, downsize of the number of defects occurring in a module from iteration n to iteration (n+1). For example, the number of defects observed in the EFT module in the first iteration is reduced to in the second iteration and also the in the third iteration and also severity of defects. Also the reduction of the defects can also be seen in all the phases of the software development life cycle. The same trend was also observed for other two modules i.e. for REP and SD modules shown in

tables 10 & 11 respectively along with the graphical representation. In this case study we have considered mainly “number of defects” parameter which is effecting the quality of service and hence the reliability.

Also with the current engagement model without RUP, the productivity is 120 LOC/PD. By using the customized RUP approach with test automation, the productivity is increased to 134 LOC/PD, which is approximately increased by 12%. Iterative development resulted in reducing the development duration, without waiting for the complete set of requirement. One set of requirements were developed in first iteration, while the second set of requirements were in elaborated phase, in parallel. By the time the first set finished its integrated testing, the development of second set is initiated. After doing an integration test on the second set, the complete requirements were tested in QAT. The duration of QAT effort was reduced drastically, from 3 months to 1 month. The number of defects in QAT also reduced by 90% compared to the PoC which was not executed based on RUP.

1. Iterative development enables process improvement and maturity in the first few iterations of the same project.
2. The PoC application integrates with another 20 applications which are not available at the time of PoC. Therefore, authors have created/used stubbed environment. This resulted frequent code drops to onsite for code builds into IST environment.
3. Risks are identified and mitigated well in advance. As mentioned in the above, offshore team has dependency not only on the external services, but also dependency on data. Authors recognized the dependency as high risk at the beginning. To mitigate the risk, a pre-IST period of two weeks was provided with the actual IST environment for testing the iterations. This facilitated in minimizing the errors during the IST testing cycle.

10.1 Recommendations

Based on the experience with several customer engagements, authors recommend the following to increase the operational efficiency and delivery excellence with special focus to defect density and other parameters indicated in the graphs.

10.2 Tool Driven Approach

Authors recommend using the following tools (from IBM) for delivery excellence.

Phase	Tool
Requirement Analysis	RequisitePro
Architecture	Rational Software Architect
Design	Rational Software Architect
Coding	Any IDE, should have facility for Rational Plug-in. For example Rational Application Developer (RAD)
Unit Testing	JUnit and NUnit
System Testing	Rational Functional

Phase	Tool
	Tester (RFT)
Performance Testing	Rational Performance Tester
Configuration Management	Rational ClearCase
Defect Management	Rational ClearQuest
RUP Implementation	RUP Builder / Rational Method Composer (RMC)
Documentation	Rational SoDA

Table 12: Recommended tools and technologies Effective Requirements Management

1. By capturing requirements with RequisitePro, the traceability can be made very simple as the same project can be imported into the design and testing.
2. During the development of the project, any changes need to done can be communicated to stakeholders by configuring mailing system with RequisitePro

10.3 Configuration Management

By having access to Clearcase from offshore, the following activities can be done efficiently.

10.3.1 Testing

Authors recommend early involvement in testing.

1. The test process should start as early as in the later stage of the inception phase, but after the first or second iteration of the requirements activities.
2. The majority of testing should be conducted during the construction phase. 10-20% (System, Load and Acceptance Test) will be executed during the transition phase.

10.3.2 Metrics and Measurement

Metrics are absolutely important to measure the project quality and performance. Authors recommend using Rational Test Manager to obtain various metrics to understand the project health. Test Manager also has an integrated reporting engine to generate various graphs and reports.

11. CONCLUSION

Based on the work that was carried by team and the obtained results, it is recommended to use RUP with careful customization to result in significant impact in productivity. In the inception phase of initial iterations, the productivity is low, but it will be improved in the subsequent iterations. In onsite-offshore model, it is recommended to have similar infrastructure to adopt the process efficiently with minimal/no dependency. This process also minimizes the risk as the dependent components would be addressed in the initial phases.

Based on the results obtained from PoC, which is experimented for RUP implementation, the number of defects is significantly reduced, in incremental cycles. This is due to the methodical approach suggested by RUP which is tool driven approach.

In mathematical modeling, the software reliability is inversely proportional to the number of defects. The results obtained in

this paper are indicating the increase of reliability by reducing the number of defects.

REFERENCES

- [1] P. Iyengar, Application Development Is More Global than Ever, publication G00124025, Gartner, 2004; www.gartner.com/resources/124000/124025/application_d ev.pdf.
- [2] Hakan Erdogmus, National Research Council Canada, The Economic Impact of Learning and Flexibility on Process Decisions
- [3] D.Sotirovski, "Heuristics for iterative software development", IEEE Software, May/June 2001, pp.66-73.
- [4] Rational Unified Process: <http://www.ibm.com>
- [5] P. Kruchten, the Rational Unified Process: An Introduction, 2e. Addison-Wesley-Longman, 2000
- [6] Hans Westerheim, Geir Kjetil Hanssen: The Introduction and Use of a Tailored Unified Process: 2005 IEEE Software
- [7] Ivar Jacobson, Shaping Software Development: May/June 2002 IEEE Software
- [8] J. G. Proakis and D. G. Manolakis – *Digital Signal Processing – Principles, Algorithms and Applications*; Third Edition; Prentice Hall of India, 2003.

Cycle 1																									
Phase	Unit / Module	Total Defects	Time Spent on Review / Testing Hrs	Re-work Time (Hrs)	Size	No. of Defects - by Severity			No of Defect - by Type										No. of Defects - by Phase of Origin .						
						H	M	L	LG	CO	IF	UI	ST	CF	DT	SY	DC	OT	Require ments	Desi gn	Coding	Testing	IST Support		
Requirements	EFT	2	10	5	140Pg	0	1	1	1										1	2					
Design	EFT	5	15	25	100Pg	5	0	0	5										1	1	4				
Coding	EFT	18	10	20	200Kloc	1	11	6	2				16								3	15			
Unit Testing	EFT	75	75	150	200Kloc	20	20	35	25	10	5	15	15		5									75	
IST Support	EFT	11	10	60	20Kloc	1	3	7	2			5			1	3									11
Cycle 2																									
Phase	Unit / Module	Total Defects	Time Spent on Review / Testing Hrs	Re-work Time (Hrs)	Size	No. of Defects - by Severity			No of Defect - by Type										No. of Defects - by Phase of Origin .						
						H	M	L	LG	CO	IF	UI	ST	CF	DT	SY	DC	OT	Require ments	Desi gn	Coding	Testing	IST Support		
Requirements	EFT	1	5	6	160Pg	0	0	1	1											1					
Design	EFT	2	10	20	130Pg	2	0	0	2									1	1	1					
Coding	EFT	9	15	30	260Kloc	0	6	3	0				9								1	8			
Unit Testing	EFT	25	16	40	258Kloc	5	9	11	5	4	4	7	4		1									25	
IST Support	EFT	7	10	20	20Kloc	0	1	6	1			2			1	3									7
Cycle 3																									
Phase	Unit / Module	Total Defects	Time Spent on Review / Testing Hrs	Re-work Time (Hrs)	Size	No. of Defects - by Severity			No of Defect - by Type										No. of Defects - by Phase of Origin .						
						H	M	L	LG	CO	IF	UI	ST	CF	DT	SY	DC	OT	Require ments	Desi gn	Coding	Testing	IST Support		
Requirements	EFT	0	3	4	190 Pg	0	0	0	0											0					
Design	EFT	0	15	25	180 Pg	0	0	0	0									0	0	0					
Coding	EFT	2	10	48	325 Kloc	0	1	1	0				2								1	1			
Unit Testing	EFT	5	75	42	200Kloc	0	2	3	0	2	0	2	1		0									5	
IST Support	EFT	1	10	30	20Kloc	0	0	1	0			0			0	1									1

Table 9: Defect consolidation log for EFT Module

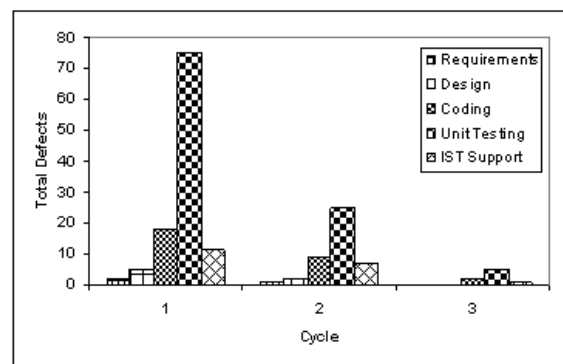


Figure 15: Total defects analysis for EFT module

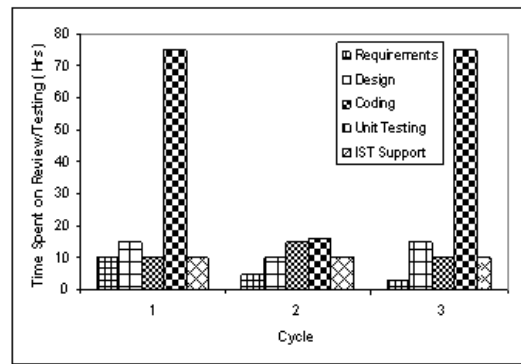


Figure 16: Time spent on Review/testing (Hrs). analysis for EFT modul

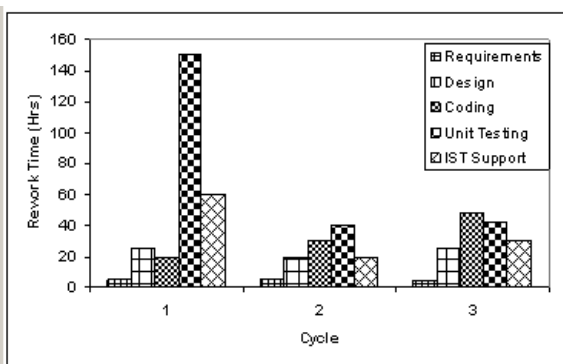


Figure 17: Rework Time (Hrs) analysis for EFT module

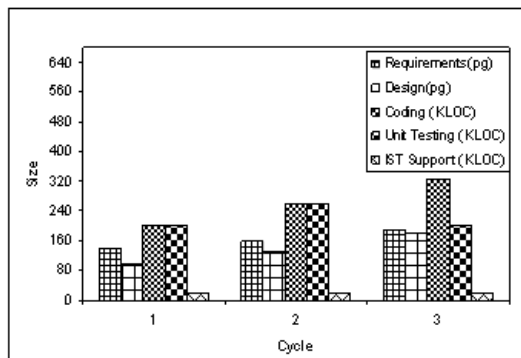


Figure 18 Size (pg/KLOCS) analysis for EFT module

Cycle 1																																										
Phase	Unit / Module	Total Defects	Time Spent on Review / Testing Hrs	Re-work Time (Hrs)	Size	No. of Defects - by Severity			No of Defect – by Type										No. of Defects – by Phase of Origin .																							
						H	M	L	LG	CO	IF	UI	ST	CF	DT	SY	DC	OT	Requirements	Design	Coding	Testing	IST Support																			
Requirements	REP	2	5	5	150Pg	0	2	1											1	2																						
Design	REP	3	5	5	36Pg	0	3	0	3																	1	2															
Coding	REP	6	10	10	100Kloc	2	2	2	2						4											1	5															
Unit Testing	REP	20	40	80	100Kloc	5	10	5	5	5					10																									20		
IST Support	REP	0	10	0	6Kloc	0	0	0																																		
Cycle 2																																										
Phase	Unit / Module	Total Defects	Time Spent on Review / Testing Hrs	Re-work Time (Hrs)	Size	No. of Defects - by Severity			No of Defect – by Type										No. of Defects – by Phase of Origin .																							
						H	M	L	LG	CO	IF	UI	ST	CF	DT	SY	DC	OT	Requirements	Design	Coding	Testing	IST Support																			
Requirements	REP	1	2	9	163 Pg	0	1																			1	1															
Design	REP	1	5	5	45 Pg	0	1	0	1																		0	1														
Coding	REP	2	8	21	138 Kloc	0	1	1	1					1													0	2														
Unit Testing	REP	9	10	14	100Kloc	2	5	2	1	2				4																										9		
IST Support	REP	0	10	0	6Kloc	0	0	0																																		
Cycle 3																																										
Phase	Unit / Module	Total Defects	Time Spent on Review / Testing Hrs	Re-work Time (Hrs)	Size	No. of Defects - by Severity			No of Defect – by Type										No. of Defects – by Phase of Origin .																							
						H	M	L	LG	CO	IF	UI	ST	CF	DT	SY	DC	OT	Requirements	Design	Coding	Testing	IST Support																			
Requirements	REP	0	1	5	192 Pg	0	0																			0	0															
Design	REP	0	5	1	49Pg	0	0	0	1																		0	0														
Coding	REP	1	14	10	170 Kloc	0	0	1	1																		0	1														
Unit Testing	REP	1	10	13	110 K Loc	0	0	1	1	0			0																											1		
IST Support	REP	0	10	0	6Kloc	0	0	0																																		

Table 10: Defect consolidation log for REP Module

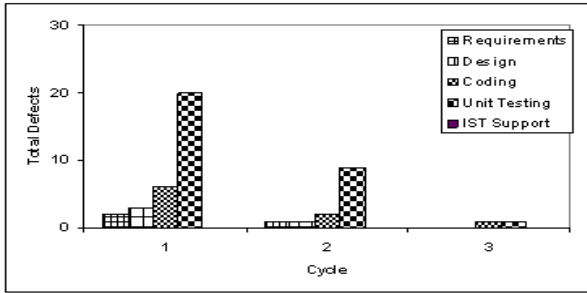


Figure 19: Total defects analysis for REP module

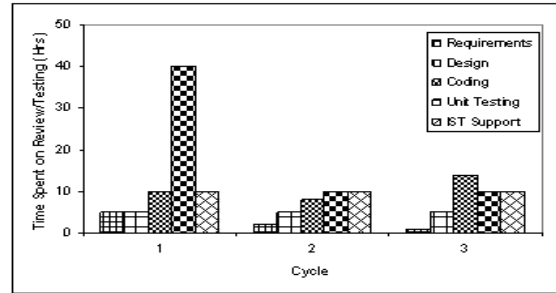


Figure 20: Time spent on Review/testing (Hrs) analysis for REP module

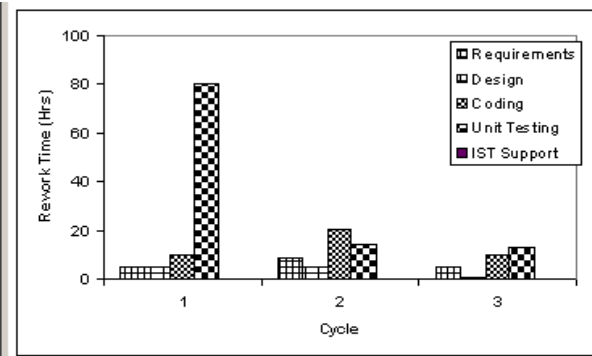


Figure 21: Rework Time (Hrs) analysis for REP module

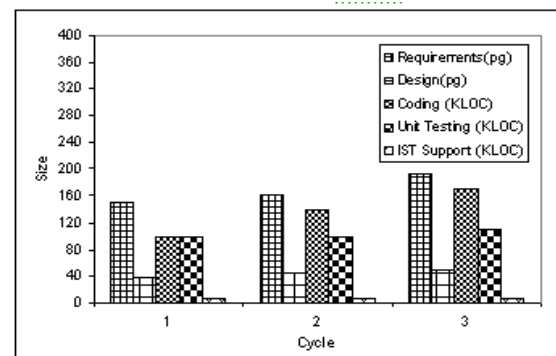


Figure 22: Size (pg/KLOCs) analysis for REP module

Cycle 1																								
Phase	Unit / Module	Total Defects	Time Spent on Review / Testing Hrs	Re-work Time (Hrs)	Size	No. of Defects - by Severity			No of Defect – by Type										No. of Defects – by Phase of Origin .					
						H	M	L	LG	CO	IF	UI	ST	CF	DT	SY	DC	OT	Requirements	Desi gn	Coding	Testing	IST Support	
Requirements	SD	3	15	10	300Pg	0	1	2	3											3				
Design	SD	3	15	15	70Pg	0	1	2	3											1	2			
Coding	SD	20	10	30	150Kloc	4	6	10	4				15								4	16		
Unit Testing	SD	45	40	70	150Kloc	10	15	20	10	15			20										45	
IST Support	SD	4	15	30	10Kloc	0	1	3					3											4
Cycle 2																								
Phase	Unit / Module	Total Defects	Time Spent on Review / Testing Hrs	Re-work Time (Hrs)	Size	No. of Defects - by Severity			No of Defect – by Type										No. of Defects – by Phase of Origin .					
						H	M	L	LG	CO	IF	UI	ST	CF	DT	SY	DC	OT	Requirements	Desi gn	Coding	Testing	IST Support	
Requirements	SD	2	10	8	315 pg	0	1	1	2											2				
Design	SD	2	11	10	70Pg	0	1	1	2											0	2			
Coding	SD	11	8	16	190 Kloc	1	4	6	2			9									1	10		
Unit Testing	SD	14	17	30	156 KLOC	1	9	4	2	5		6											14	
IST Support	SD	1	5	15	10Kloc	0	0	1				1												1
Cycle 3																								
Phase	Unit / Module	Total Defects	Time Spent on Review / Testing Hrs	Re-work Time (Hrs)	Size	No. of Defects - by Severity			No of Defect – by Type										No. of Defects – by Phase of Origin .					
						H	M	L	LG	CO	IF	UI	ST	CF	DT	SY	DC	OT	Requirements	Desi gn	Coding	Testing	IST Support	
Requirements	SD	0	5	3	290 Pg	0	0	0	0											0				
Design	SD	0	9	9	70Pg	0	0	0	0											0	0			
Coding	SD	2	8	14	205 KLOC	0	1	1	1			1									0	2		
Unit Testing	SD	2	14	26	163 KLOC	0	1	1	0	0		1											2	
IST Support	SD	0	5	15	10Kloc	0	0	0				0												0

Table 11: Defect consolidation log for SD Module

Continued on page no. 60