CLOSURE REPORT

FOR

ACIC PROJECT

Version 1.0 approved

Pankaj Jalote
Ravi Anand

Infosys Inc.

December – 2000

TABLE OF CONTENTS

1. GENERAL INFORMATION

Project Code	Xxxxx
Life Cycle	Development, Full life cycle
Business Domain	Finance. Web-based application for managing accounts
Project Leader/Module Leader	Xxxxxx
Business Manager	Xxxxxx
Software Quality Adviser	Xxxxxx

2. PERFORMANCE SUMMARY

Performance Parameter	**Actual**	**Estimated**	**Deviation**	**Reasons for Deviation (if large)**
Total effort (person-days)	597	501	19%	Two major change requests that came
Peak Team Size	9	9	0	N/A
Start Date	03 Apr 2000	03 Apr 2000	0	N/A
End Date	03 Nov 2000	30 Nov 2000	27 days	Two major change requests consumed more than 5% of the effort
Quality (number of defects delivered per FP)	0.002	0.0125		Quality improved because of defect prevention and use of incremental process
Productivity	58	57	2%	N/A
Cost of quality	31.4%	33%	5%	N/A
Defect injection rate	0.022	0.03	-26%	Improved because of defect prevention
Defect removal efficiency	97.4	97	Small	N/A

3. PROCESS DETAILS

Process Tailoring	• Rational Unified Process was employed • Development and analysis were done iteratively –3 iterations for development and 2 for design and analysis • Requirements traceability was done through Requisite Pro tool

4. TOOLS USED

Notes on Tools Used	• External tools: VSS, VJA, Requisite Pro, MSP • Internal tools: BugsBunny, WAR

5. RISK MANAGEMENT

Risks identified at the start of the project

Risk 1	Lack of support from database architect and database administrator of the customer
Risk 2	Improper use of RUP, as it is being used for the first time
Risk 3	Personnel attrition
Risk 4	Problems with working on customer's database over the link

Risks encountered during the project

Risk 1	Impact of conversion to VAJ 3.0
Risk 2	Lack of support from database architect and database administrator of the customer
Risk 3	Improper use of RUP, as it is being used for the first time
Risk 4	Personnel attrition

Notes on Risk Mitigation

Risk 1: Clearly articulating the risk helped in customer agreeing to postpone the conversion with proper budgeting of its impact.

Risk 2: Mitigation strategies of careful and advance planning and employing the on-site coordinator were effective.

Risk 3: Training the team in RUP was effective. So was keeping the customer informed.

Risk 4: Remained as a risk, although it did not materialize. Impact would have been minimal because multiple people were kept informed of each critical activity.

6. SIZE

	Estimated	**Actual**
Number of simple use cases	5	5
Number of medium use cases	9	9
Number of complex use cases	12	12

Notes on Estimation

Classification Criteria: The standard definition of simple, medium, and complex was used for classifying the use cases. This worked fine.

Final System Size in FP

The size of the final source is measured in LOC. It is normalized to FP by using the published conversion tables. For Java, the published tables suggest that 21 LOC equals 1 FP and for COBOL, 107 LOC equal 1 FP.

Output Language	Size in LOC	Size in FP
Java	33,865	1,612
COBOL	1,241	12

7. SCHEDULE

Phase	Actual Elapsed Time (days)	Estimated Time (days)	% Slippage	Reasons for Slippage
Requirements	28.67	31	-6.5	
High-level design	0	0	0.0	
Detailed design	38.8	42	-6.7	
Coding	132	135	-1.6	
Unit testing	9	10	-9.3	
Total – Build	141	144	-2.1	
Integration test	40	40	0	
System testing	15	0	0.0	
Acceptance testing	30	10	200	AT completion was extended on customer's request

8. EFFORT

Distribution over Life-Cycle Stages

Stage	Task	Review	Rework	Total
Requirements	210	10	60	280
High-level design	0	0	0	0
Detailed design	652	14	29.5	695.5
Coding	1188.0	39.5	76.5	1304.0
Unit testing	129.5	0.0	17.0	146.5
Integration testing	567.5	6.0	160.5	734.0
System testing	90.0	0.0	0.0	90

Stage	Task	Review	Rework	Total
Acceptance testing	336.5	0.0	0.0	336.5
Total –LC stages	3173.5	69.5	343.5	3586.5
Project management	733.1	0.0	0.0	733.1
Training	104.5	0.0	0.0	104.5
CM	317.0	0.0	0.0	317.0
Misc.	488.5	0.0	0.0	488.5
Total –mgmt, training, and misc.	1643.0	0.0	0.0	1643.0
Total effort (Person-hours)	4816.5	69.5	343.5	5229.5
Total effort (Persons-months)	25.76	0.37	1.84	27.97

Cost of Quality

$$COQ = \frac{\text{Review effort + rework effort + test effort + training effort}}{\text{Total effort}} * 100$$

$$= \frac{69.5 + 343.5 + 129.5 + 567.5 + 90 + 336.5 + 104.5}{5229.5} * 100$$

$$= 31.4\%$$

Effort Distribution and Actual Versus Estimated

	Actual		Estimated			
Stage	**Effort (person-hours)**	**%**	**Effort (person-hours)**	**%**	**% Deviation**	**Reasons or Deviation**
Requirements	280	5.35	475.0	10	-41	Overestimated this effort (data from earlier project did not help because it did not have this phase)
Design (HLD and detailed)	695.5	13.30	569.0	12	22	Design took more time because team was inexperienced with Rational Rose and OOAD.
Coding	1304.0	24.94	1235.3	26	6	
Unit testing	146.5	2.80	142.5	3	3	
Integration testing	734.0	14.04	331.0	7	121	Much effort spent on fixing bugs introduced during reconciliation with Synergy and Window Resized code.
System testing	90.0	1.72	95.0	2	-5	

Stage	Actual Effort (person-hours)	Actual %	Estimated Effort (person-hours)	Estimated %	% Deviation	Reasons or Deviation
Acceptance testing	336.6	6.43	285.0	6	18	Acceptance testing was not completed on Nov. 3 and was extended until Nov 23 due to delays from customer
Total –LC stages	3586.5	68.58	3132.8	66	14	
Project management	733.1	14.02	713.0	15	3	
Training	104.5	2.00	455.0	10	-77	
CM	317.0	6.06	142.0	3	123	Deviation due to reconciliation issues
Misc.	488.5	9.34	285.0	6	71	More because of training
Total –mgmt, training, and misc.	1643.0	31.42	1595.0	34	3.01	
Total	5229.5	100	4727.8	100	10.6	

9. DEFECTS

Defect Distribution

Stage Detected	Actual Number of Defects	% of Total Defects Found	Estimated Number of Defects	% of Total Estimated Defects	% Deviation
Req. and design review	11	10	29	20	-62
Code review	58	50	29	20	20
Unit testing	15	13	57	40	-73
Integration and system testing	29	25	25	17	16
Acceptance testing	3	2	5	3	-30
Total	116	100	145	100	-20

Reasons for Deviation

1. Defect prevention reduced the defect injection rate in later stages, resulting in overall reduction in the defect injection rate.

2. In the earlier project from which estimates were derived, fewer code reviews were done and there was a heavier reliance on UT. In this project, because code reviews were done more rigorously and widely, more defects were found in reviews, leading to a substantial decrease in the defects found in unit testing.

Defect Removal Efficiencies

Defects Detection Stage	**Defects Injection Stage**			**Defect Removal Efficiency**
	Req.	**Build**	**Design**	
Req. review	5			100%
Design review	0	6		100%
Code review	0	0	58	55% (58 / 58 + 15 + 29 + 3)
Unit testing	0	0	15	32% (15 / 15 + 29 + 3)
Integration/system testing	0	0	29	93% (29 / 29 + 3)
Acceptance testing	0	0	3	100%

Overall Defect Removal Efficiency = 113 / 116 = 97.4%

Distribution by Severity

Sequence Number	**Severity**	**Number of Defects**	**% of Total Defects**
1	Cosmetic	26	22.4
2	Minor	51	44
3	Major	36	31
4	Critical	3	2.6
5	Others	--	--
	Total	116	

Distribution by Defect Type

Sequence Number	**Defect Type**	**Number of Defects**	**% of Total Defects**
1	Logic	33	28.4
2	Standards	29	25
3	Performance	24	20.7
4	Redundant code	14	12
5	User interface	9	7.7
6	Architecture	4	3.5
7	Consistency	2	1.7
8	Reusability	1	0.9
	Total	116	

10. CAUSAL ANALYSIS AND LESSONS LEARNED

There were very few large deviations in the process performance; the actual performance was close to what was expected. The reasons for the deviations, where they are large, are given along with the deviation. Some key lessons learned are:

1. Incremental or phased development is extremely helpul in achieving higher quality and productivity because data from the first phase can be used to improve the remaining phases through defect prevention.
2. Defect prevention can substantially reduce the defect injection rate. In terms of effort also, defect prevention pays off handsomely; by putting in a few hours of effort, up to 5 to 10 time effort savings can be achieved in the form of reduced rework effort.
3. If a change request has a major impact, discussion with the customer using a detailed impact analysis can be very helpful in setting the right expectations and doing a proper cost-benefit analysis (which may result in postponement of the change, as happened in this project).
4. The defect removal efficiencies of code reviews and unit testing are very low. Processes for both, and implementation of these processes, need to be reviewed to improve these numbers. In this project, the system/integration testing compensated for the poor performance of reviews and unit testing. However, for larger projects, this may not be possible and poor performance in reviews and unit testing can have adverse effects on quality.

11. PROCESS ASSETS SUBMITTED

Project management plan, project schedule, configuration management plan, Java coding standards, code review checklist, integration plan review checklist, impact analysis checklist, causal analysis reports for defect prevention.

12. REFERENCES

Omitted.