Chapter3.
Software Project Management

3.1 Introduction

- Software project management is concerned with activities involved in ensuring that software is delivered on time and on schedule and in accordance with the requirements of the organisations developing and procuring the software.
- Project management is needed because software development is always subject to budget and schedule constraints that are set by the organisation developing the software.

- Software project management is the important task of planning, directing, motivating, and coordinating a group of professionals to accomplish software development.
- Software project management uses many concepts from management in general, but it also has some concerns unique to software development.
- One such concern is project visibility. The lack of visibility of the software product during software development makes it hard to manage. Many of the techniques in software management are aimed at overcoming this lack of visibility.

Project Success Criteria

- Deliver the software to the customer at the agreed time.
- Keep overall costs within budget.
- Deliver software that meets the customer’s expectations.
- Maintain a happy and well-functioning development team.

Management Activities

- Project planning
  - Project managers are responsible for planning. estimating and scheduling project development and assigning people to tasks.
- Risk management
  - Project managers assess the risks that may affect a project, monitor these risks and take action when problems arise.
- People management
  - Project managers have to choose people for their team and establish ways of working that leads to effective team performance
- Reporting
  - Project managers are usually responsible for reporting on the progress of a project to customers and to the managers of the company developing the software.

3.2 Management Approaches

- A basic issue in software project management is whether the process or the project is the essential feature being managed.
- In process-oriented management, the management of the small tasks in the software life cycle is emphasized.
- In project management, the team achieving the project is emphasized.
- This results in important differences in viewpoint. In a process management approach, if the team does not follow the prescribed software life cycle, this would be a major difficulty.
- In a project management approach, success or failure is directly attributed to the team.

3.3 Team Approaches

- Organizing a group of people into an efficient and effective team can be a difficult task. Letting a team develop its own paradigm can be risky. Choosing a team organization based on the project and the team members may help avoid disaster.
- One aspect of a team is the amount of structure in the team. While some groups of programmers can work very independently, other groups need strong structure to make progress. In a strongly structured team, small assignments are made to each member. In a weakly structured team, the tasks are usually of longer duration and more open-ended.
- Some teams consist of people with similar skills. Other teams are composed of people with different expertise that are grouped into a team based on the need for specific skills for a project.

Managing People

- People are an organisation’s most important assets.
- The tasks of a manager are essentially people-oriented. Unless there is some understanding of people, management will be unsuccessful.
- Poor people management is an important contributor to project failure.

- People Management Factors
  - Consistency
    - Team members should all be treated in a comparable way without favourites or discrimination.
  - Respect
    - Different team members have different skills and these differences should be respected.
  - Inclusion
    - Involve all team members and make sure that people’s views are considered.
  - Honesty
    - You should always be honest about what is going well and what is going badly in a project.

Motivating People

- An important role of a manager is to motivate the people working on a project.
- Motivation means organizing the work and the working environment to encourage people to work effectively.
  - If people are not motivated, they will not be interested in the work they are doing. They will work slowly, be more likely to make mistakes and will not contribute to the broader goals of the team or the organization.
- Motivation is a complex issue but it appears that their are different types of motivation based on:
  - Basic needs (e.g. food, sleep, etc.);
  - Personal needs (e.g. respect, self-esteem);
    - Recognition of achievements;
    - Appropriate rewards.
  - Social needs (e.g. to be accepted as part of a group).
    - Provide communal facilities;
    - Allow informal communications e.g. via social networking

Teamwork

- Most software engineering is a group activity
  - The development schedule for most non-trivial software projects is such that they cannot be completed by one person working alone.
- A good group is cohesive and has a team spirit. The people involved are motivated by the success of the group as well as by their own personal goals.
- Group interaction is a key determinant of group performance.

CHIEF PROGRAMMER TEAMS

- IBM developed the chief programmer team concept. It assigns specific roles to members of the team. The chief programmer is the best programmer and leads the team. Nonprogrammers are used on the team for documentation and clerical duties.

EXAMPLE 3.1

Draw a high-level process model for a hierarchical team organization. (See Fig. 3-1.)

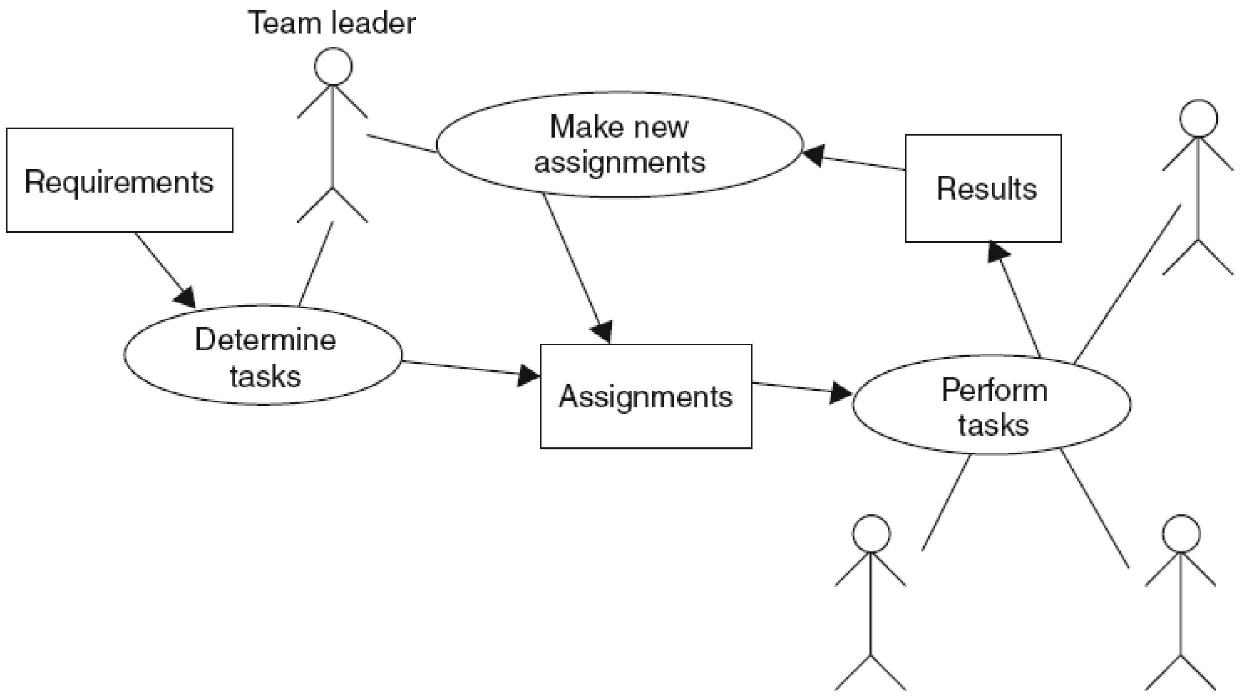


Fig. 3-1. High-level process model for hierarchical team structure.

CHIEF PROGRAMMER TEAMS

EXAMPLE 3.2

Company WRT has an IT department with a few experienced software developers and many new programmers. The IT manager has decided to use highly structured teams using a process approach to managing. Each team will be led by an experienced software developer. Each team member will be given a set of tasks weekly. The team leader will continually review progress and make new assignments.

3.4 Critical Practices

- Most studies of software development have identified sets of practices that seem critical for success. The following 16 critical success practices come from the Software Project Managers Network (www.spmn.com):
    - Adopt continuous risk management.
    - Estimate cost and schedule empirically.
    - Use metrics to manage.
    - Track earned value.
    - Track defects against quality targets.
    - Treat people as the most important resource.
    - Adopt life cycle configuration management.
    - Manage and trace requirements.
    - Use system-based software design.
    - Ensure data and database interoperability.
    - Define and control interfaces.
    - Design twice, code once.
    - Assess reuse risks and costs.
    - Inspect requirements and design.
    - Manage testing as a continuous process.
    - Compile and smoke-test(the first run of a SW after construction or a critical change) frequently.

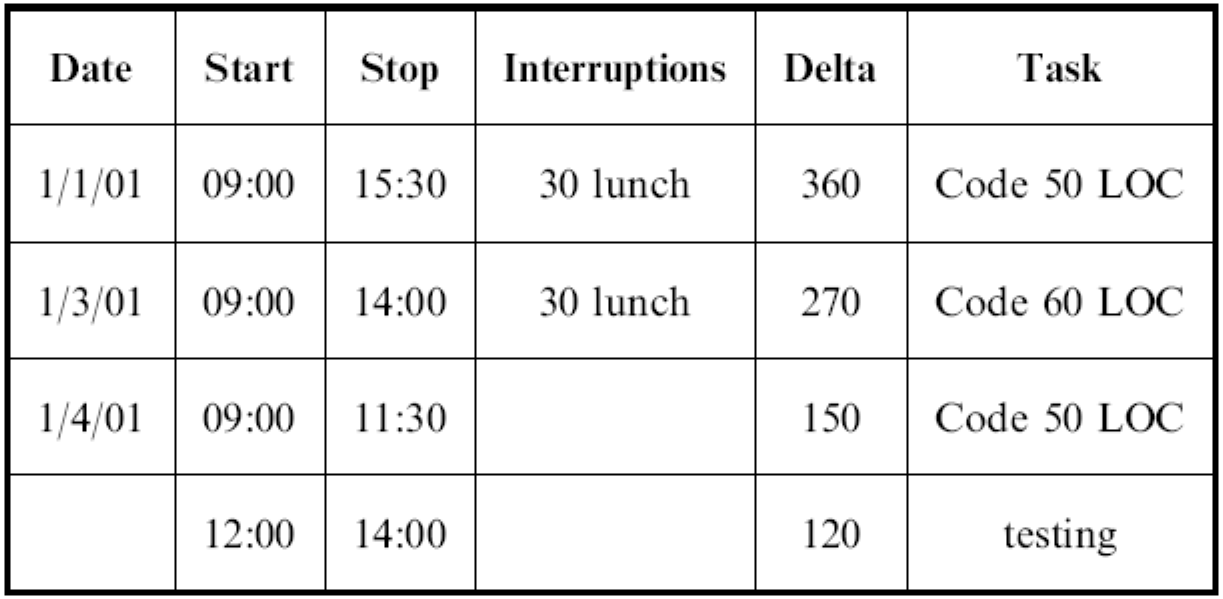
3.6 Personal Software Process

- Watts Humphrey has developed the Personal Software Process to improve the skills of the individual software engineer. His approach has the individual maintain personal time logs to monitor and measure the individual’s skills.
- One result of this is measuring an individual’s productivity. The usual measure of productivity is lines of code produced per day (LOC/day). Additionally, errors are timed and recorded.
- This allows an individual to learn where errors are made and to assess different techniques for their effect on productivity and error rates.

3.6 Personal Software Process

EXAMPLE 3.4

Programmer X recorded this time log.



Date	Start	Stop	Interruptions	Delta	Task
1/1/01	09:00	15:30	30 lunch	360	Code 50 LOC
1/3/01	09:00	14:00	30 lunch	270	Code 60 LOC
1/4/01	09:00	11:30		150	Code 50 LOC
	12:00	14:00		120	testing

The programmer spent 360 + 270 + 150 + 120 = 900 minutes to write and test a program of 160 LOC. Assuming 5 hours per day (300 minutes/day), X spent effectively 3 days to program 160 LOC. This gives a productivity of 53 LOC/day. When X's manager schedules a week to code a 1000 = LOC project, X is able to estimate that the project will take about 4 weeks.

3.7 Earned Value Analysis

- One approach to measuring progress in a software project is to calculate how much has been accomplished. This is called earned value analysis.
- It is basically the percentage of the estimated time that has been completed.

BASIC MEASURES

- Budgeted Cost of Work (BCW): The estimated effort for each work task.
- Budgeted Cost of Work Scheduled (BCWS): The sum of the estimated effort for each work task that was scheduled to be completed by the specified time.
- Budget at Completion (BAC): The total of the BCWS and thus the estimate of the total effort for the project.
- Planned Value (PV): The percentage of the total estimated effort that is assigned to a particular work task; PV = BCW/BAC.
- Budgeted Cost of Work Performed (BCWP): The sum of the estimated efforts for the work tasks that have been completed by the specified time.
- Actual Cost of Work Performed (ACWP): The sum of the actual efforts for the work tasks that have been completed.

3.7 Earned Value Analysis

PROGRESS INDICATORS

- Earned Value (EV) = BCWP/BAC
  = The sum of the PVs for all completed work tasks
  = PC = Percent complete
- Schedule Performance Index(SPI) = BCWP/BCWS
- Schedule Variance (SV) = BCWP – BCWS
- Cost Performance Index(CPI) = BCWP/ACWP
- Cost Variance (CV) = BCWP – ACWP

3.7 Earned Value Analysis

EXAMPLE 3.5

Company LMN is partway through its project. The job log below indicates the current status of the project.

Work Task	Estimate Effort (programmer-days)	Actual Effort So Far (programmer-days)	Estimated Completion Date	Actual Date Completed
1	5	10	1/25/01	2/1/01
2	25	20	2/15/01	2/15/01
3	120	80	5/15/01	
4	40	50	4/15/01	4/1/01
5	60	50	7/1/01	
6	80	70	9/01/01	

The BAC is the sum of the estimations. BAC = 330 days. BAC is an estimate of the total work. On 4/1/01, tasks 1,2, and 4 have been completed. The BCWP is the sum of the BCWS for those tasks. So BCWP is 70 days. The earned value (EV) is 70/330, or 21.2 percent. On 4/1/01 tasks 1 and 2 were scheduled to be completed and 1,2, and 4 were actually completed. So BCWP is 70 days and BCWS is 30 days. Thus, SPI is 70/30, or 233 percent. The SV = 70 days 30 days = +40 days, or 40 programmer-days ahead. The ACWP is the sum of actual effort for tasks 1, 2, and 4. So, ACWP is 80 programmer-days. CPI is 70/80 = 87.5 percent. The CV = 70 programmer-days 80 programmer-days = 10 programmer-days, or 10 programmer-days behind.

3.7 Earned Value Analysis

EXAMPLE 3.6

On 7/1/01, assume that task3 has also been completed using 140 days of actual effort, so BCWP is 190 and EV is 190/330, or 57.5 percent. On 7/1/01, tasks 1, 2, 3, and 4 were actually completed. So BCWP is 190 days and BCWS is 250 days. Thus, SPI is 190/250 = 76 percent. The SV is 190 programmer-days 250 programmer-days = 60 programmer-days, or 60 programmer days behind. ACWP is the sum of actual effort for 1, 2, 3, and 4. So ACWP is 220 programmer-days. Tasks 1 through 5 were scheduled to have been completed, but only 1 through 4 were actually completed. CPI is 190/220 = 86.3 percent, and CV is 190–220, or 30 programmer-days behind.

3.8 Error Tracking

- One excellent management practice is error tracking, which is keeping track of the errors that have occurred and the inter-error times (the time between occurrences of the errors).
- This can be used to make decisions about when to release software. An additional effect of tracking and publicizing the error rate is to make the software developers aware of the significance of errors and error reduction. Making the errors and error detection visible encourages testers and developers to keep error reduction as a goal.
- The error rate is the inverse of the inter-error time. That is, if errors occur every 2 days, then the instantaneous error rate is 0.5 errors/day.
- Usually, most errors are corrected (the faults removed), and thus the error rates should go down and the inter-error times should be increasing. Plotting this data can show trends in the error rate (errors found per unit time). The trend can be used to estimate future error rates.

3.8 Error Tracking

EXAMPLE 3.7

Consider the following error data (given as the times between errors): 4, 3, 5, 6, 4, 6, 7. The instantaneous error rates are the inverses of the inter-error times: 0.25, 0.33, 0.20, 0.17, 0.25, 0.17, and 0.14. Plotting these against error number gives a downward curve, as shown in Figure 3-2. This suggests that the actual error rate is decreasing.

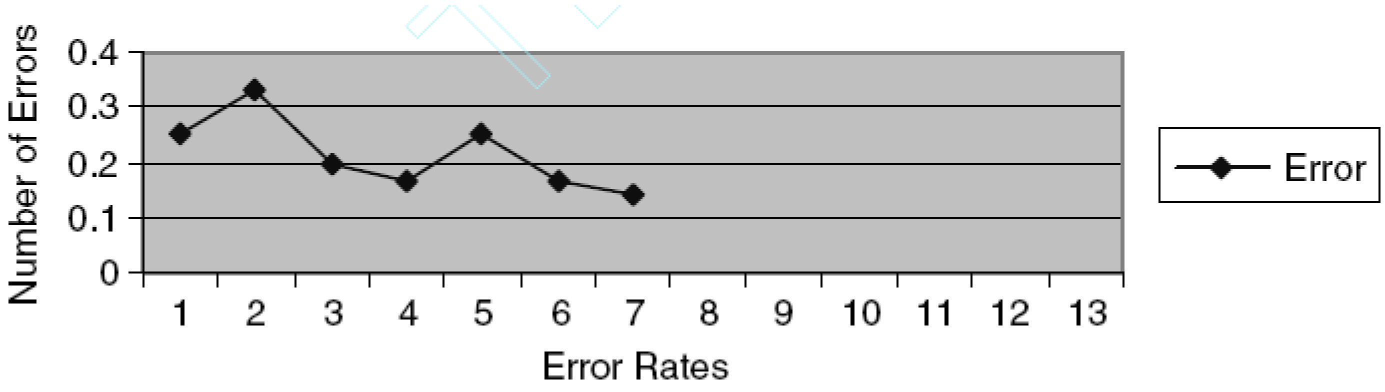


Fig. 3-2. Plot of error rates.

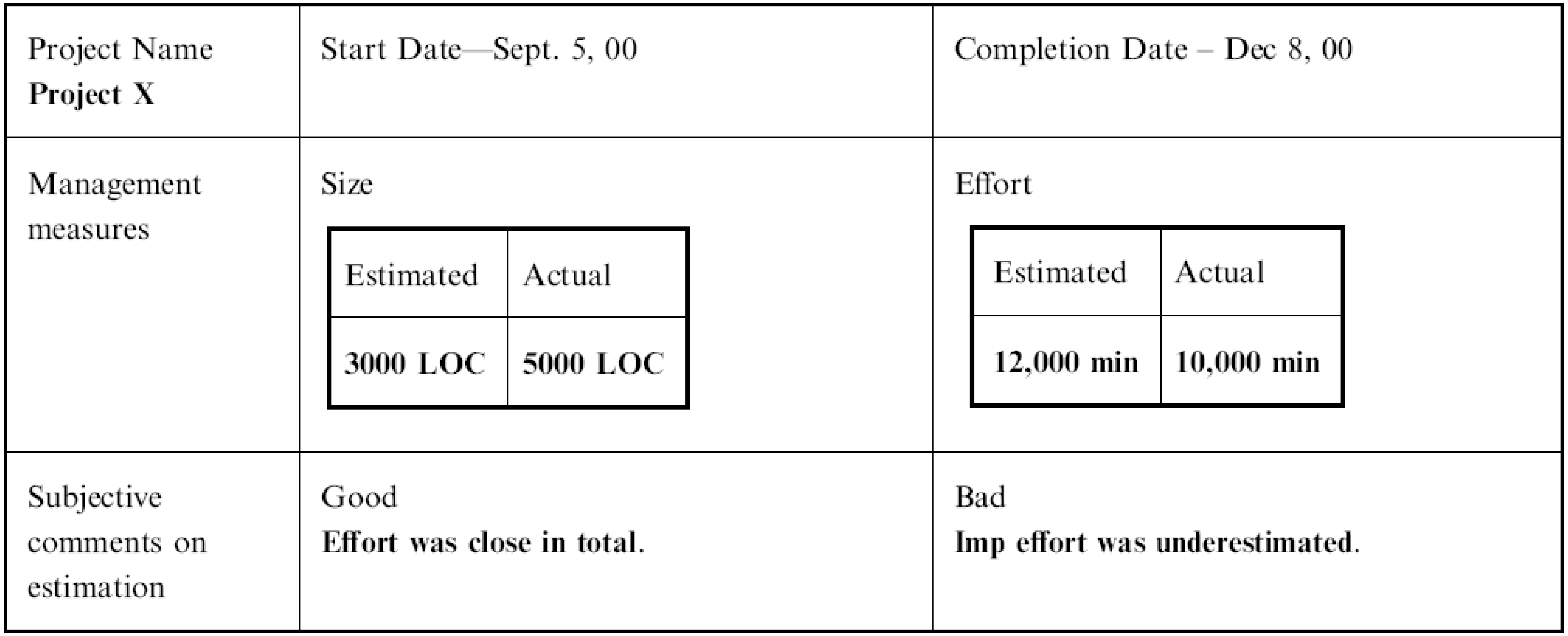
A straight line through the points would intersect the axis about 11. Since this implies that the error rate would go to zero at the eleventh error, an estimate of the total number of errors in this software would be 11 errors. Since seven errors have been found, this suggests that there may be four more errors in the software.

3.9 Postmortem Reviews

- One critical aspect of software development is to learn from your mistakes and successes. In software development, this is called a postmortem. It consists of assembling key people from the development and the users groups. Issues consist of quality, schedule, and software process. A formal report needs to be produced and distributed.

EXAMPLE 3.8

Company JKL produced the postmortem.



Project Name **Project X**	Start Date—Sept. 5, 00	Completion Date – Dec 8, 00
Management measures	Size Estimated — Actual **3000 LOC** — **5000 LOC**	Effort Estimated — Actual **12,000 min** — **10,000 min**
Subjective comments on estimation	Good **Effort was close in total.**	Bad **Imp effort was underestimated.**

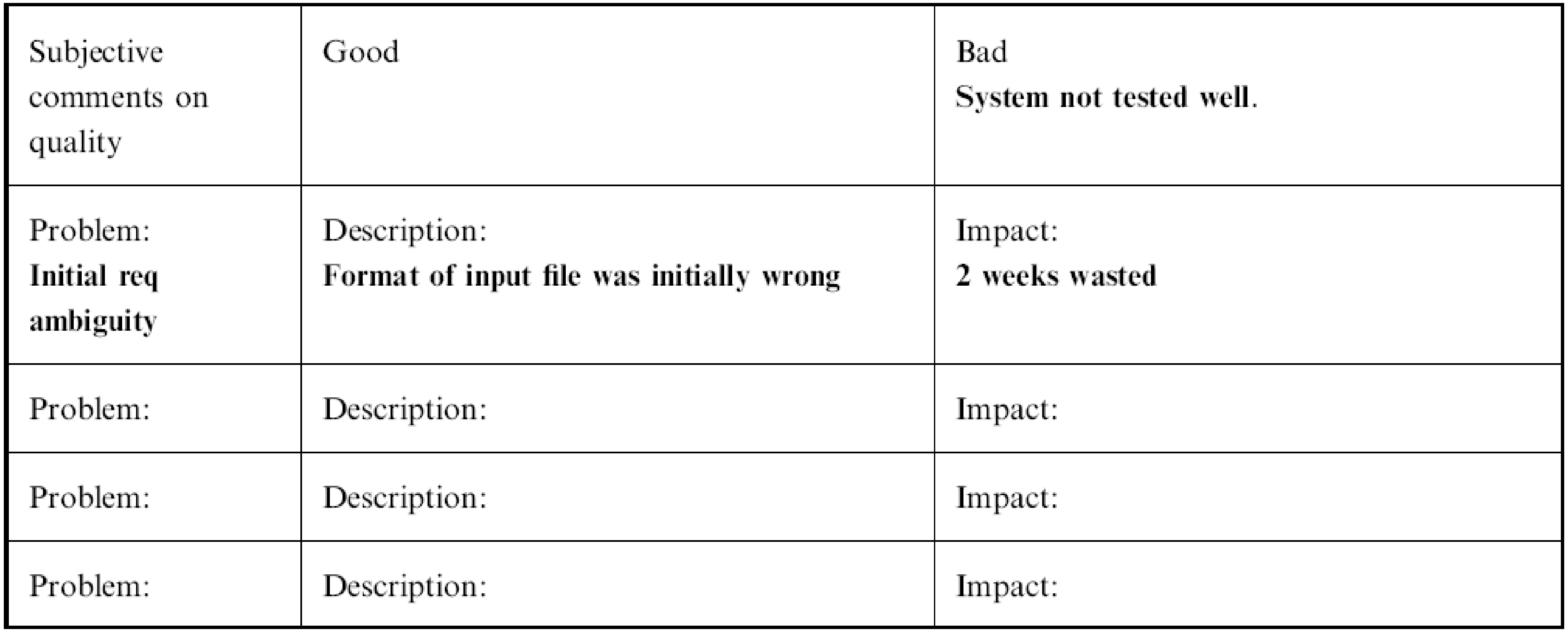
3.9 Postmortem Reviews

Subjective comments on process	Good	Bad **Team members did not complete asgn on time.**
Subjective comments on schedule	Good	Bad **Not enough time for imp.**
Quality	Errors found.	

Req	Design	Unit	Integ	Postdel
			30	

	ave	max
mccabe	**4**	**30**
Method/class	**6**	**10**
Attributes/class	**10**	**15**
LOC/class	**150**	**500**

3.9 Postmortem Reviews



Subjective comments on quality	Good	Bad **System not tested well.**
Problem: **Initial req ambiguity**	Description: **Format of input file was initially wrong**	Impact: **2 weeks wasted**
Problem:	Description:	Impact:
Problem:	Description:	Impact:
Problem:	Description:	Impact: