Calculating Your Return on Investment from More Effective Requirements Management

by Dean Leffingwell

Abstract

There is strong evidence to suggest that reducing requirements errors may be the single most effective action application developers can take to improve project outcomes and assist us in our goal ***of delivering quality software, on time and on budget***. This article highlights some of the empirical evidence and demonstrates that an investment in more effective requirements management can produce substantial rewards.

The Software Crisis Continues Unabated

In a June 1996 article in Fortune magazine cleverly titled " The Trouble with Software Is …It Sucks "[1], industry pundit Stewart Alsop lambasted our industry with his comments on the poor state of software quality and reliability. While many in the industry take offense at yet another broadside, a recent study by the Standish Group[2], a well respected market research firm, provided an even more sobering perspective. Specifically, according to the Standish Group survey (over 352 companies reporting on over 8,000 software projects):

- 31% of all software projects are canceled before completed ($81 billion waste)
- 53% of projects will cost 189% of estimates
- 9% on time and on budget (large companies)
- 16% on time and on budget (small companies)

In a further step to help understand the problem, the Standish Group survey also asked its respondents to identify the causes of these failures. According to the respondents, the top three reasons why projects were "impaired" are identified in Table 1 below.

Project Impairment Factors	% of Responses
1) Lack of User Input	12.8%
2) Incomplete Requirements and Specifications	12.3%
3) Changing Requirements and Specifications	11.8%

Table 1: Standish Group Project Impairment Factors

It seems that our inability to work more effectively with users to better understand their requirements, coupled with weak engineering discipline in managing requirements, is the leading cause of software failures.

The High Cost of Requirement Errors

Studies performed at GTE, TRW, and IBM measured and assigned costs to errors occurring at various phases of the project life-cycle[3]. These statistics were confirmed in later studies.[4] Although these studies were run independently, they all reached roughly the same conclusion: If a unit cost of *one* is assigned to the effort required to detect and repair an error during the coding stage, then the cost to detect and repair an error during the requirements stage is between *five* to *ten* times less. Furthermore, the cost to detect and repair an error during the maintenance stage is *twenty* times more. Figure 1 below illustrates a summary of the results.

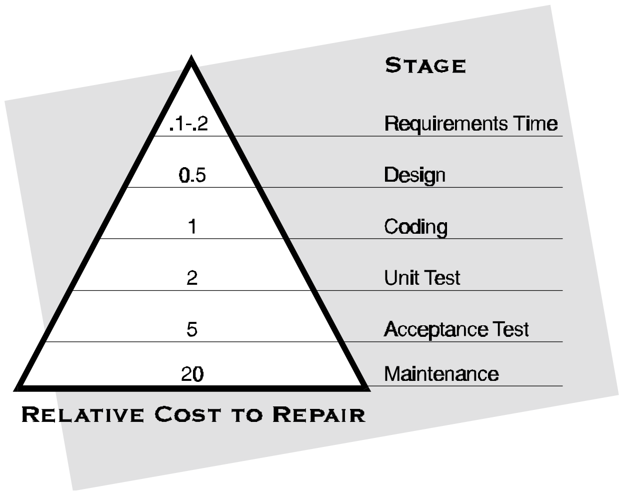

***Figure 1- As much as a 200:1 cost savings results from finding errors in the requirements stage versus finding errors in the maintenance stage of the software life-cycle*.**

The reasons for this large difference is that many of these errors are not detected until well after they have been made. This delay in error discovery means that the cost to repair includes the cost to correct the offending error and to correct subsequent investments in the error. These investments include redesign of code, documentation rewrite, and the cost to rework or replace software in the field.

Requirement errors are the most common errors

These studies illustrate that errors made in the requirements phase are extremely expensive to repair. If such errors occurred infrequently, then the contribution to overall cost would not be significant. However, requirements errors are indeed the largest class of errors typically found in a complex software project. In a study of a US Air Force project by Sheldon[5], errors were classified by source. It was found that requirements errors comprised 41% of the errors discovered, while logic design errors made up only 28% of the total error count. Other studies back this result. For example, Tavolato and Vincena, quoting Tom DeMarco, report that 56% of all bugs can be traced to errors made during the requirements stage[6].

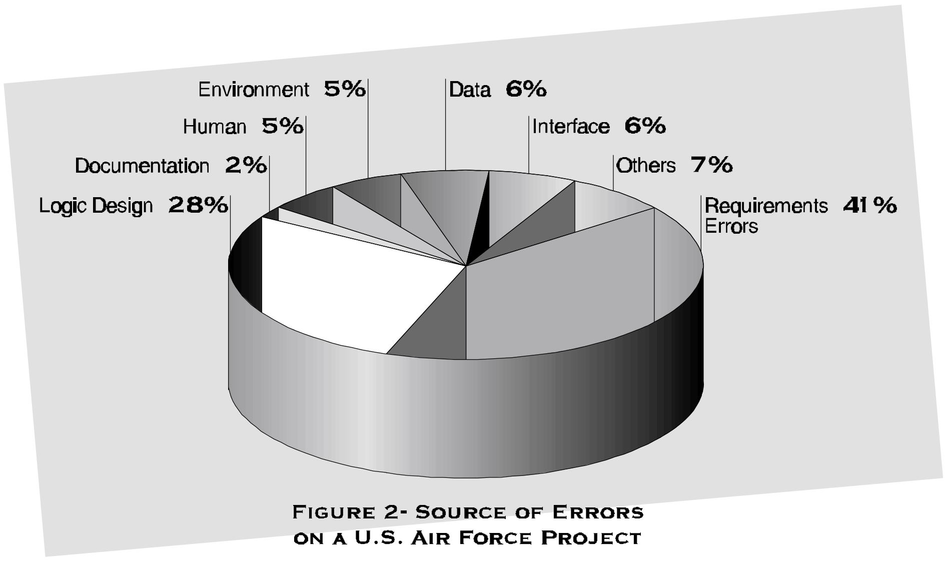

FIGURE 2- SOURCE OF ERRORS ON A U.S. AIR FORCE PROJECT

Requirement Errors and Rework Costs

In a study performed at Raytheon, Dion reported that approximately 40% of the total project budget was spent in rework costs[7] . Boehm reports that the cost of rework can approach 50% for the largest software projects. Because of their large number, and the multiplying effect, *finding and fixing requirement errors consumes between 70% - 85% of total project rework costs.*

Reducing Requirement Errors

There is no silver bullet with which to make your requirement errors go away. However, organizations have demonstrated that the following techniques are effective in reducing these types of errors:

- More effective requirements elicitation
- Walkthroughs and reviews of requirements with customers and end users
- Better organization and documentation of requirements
- Accessible requirements repository, fostering improved team communication
- Improved reporting processes
- Better requirements-based testing and requirements traceability

In order to become proficient in these activities, an investment in both tools and training for key project personnel will be required.

Estimating Typical Project and Rework Costs

Consider an application which contains 50,000 lines of code and is produced by a staff of six developers supported by one program manager, two software testers and one quality assurance person. To determine the product time line, let's assume coding is the critical path (probably a defensible assumption in most organizations!). Based upon a productivity rate of 350 lines of debugged source code per month[8], then the project timeline is 24 months. (i.e. 50,000 LOC÷350 LOC/month÷6 developers). If the loaded cost per developer/tester/QA personnel per month is $10,000, then that creates a total project budget of approximately $2.4 Million. Of this cost, even if only 30% of the project is invested in rework (an optimistic assumption?), then the total rework budget is $720,000. If 70% of the rework is due to requirement errors, then the total rework costs related to requirement errors are over half a million dollars!

	Typical Project	Your Project
Number of lines of code	**50,000**	
Number of developers	**6**	
Person months of coding	**142.9**	
Time to market	**24 calendar months**	
Loaded cost/team member/mo	**$10,000**	
Total project labor cost	**$2,400,000**	
Total rework costs (30%)	**$720,000**	
Total requirements errors rework cost (≈70%)	**$504,000**	

Table 1: Project Cost Estimates for a Software Project

Your Investment

What's to be done to rework - this valueless project activity? Simple, REDUCE REQUIREMENTS ERRORS. The techniques demonstrated above could be expected to have a dramatic effect on reducing requirements errors. However, as with any meaningful process, an investment in both tools and training is required for success. Let's assume that an organization purchases Requisite's RequisitePro toolkit and training in both requirements management principles and tool usage for the project staff. This would require an investment of approximately $19,900 (ten people *($995 tool+$995 training)). How would this investment pay off?

The Bottom Line

No one can predict exactly how effective your organization will be in reducing requirement errors. However, it seems reasonable to assume that a 20% or more reduction in requirement errors can be accomplished at various levels of organizational maturity. Because of the multiplying effect, *any* such reduction can have a dramatic overall effect to your project's bottom line as Table 2 below shows.

	Case 1	Case	Case 3	Your Project
Requirements error reduction percentage	**10%**	**20%**	**40%**	
Cost savings on typical project	**$50,400**	**$100,800**	**$201,600**	
Months shaved off time-to-market	**1.0**	**1.7**	**2.5**	
Payback time for investment in months	**9.5**	**4.7**	**2.4**	
% Return on investment over life of *First project*	**153%**	**407%**	**913%**	

Table 2: Return on Investment of More Effective Requirements Management

Summary

Table 2 above shows that the investment in requirements management tools and training can provide a truly extraordinary payback even on the first project. Thereafter, the team's ability to reduce requirements errors will continue to pay a substantial dividend on future projects.

But these hard costs exclude the intangible costs associated with a requirement error. Intangible costs include lack of features that could have been delivered had the project's resources not been devoted to rework, loss of confidence on the part of customers, and accompanying lost and unrecoverable market share, revenue and profit.

Taken together, these costs clearly demonstrate that a company cannot afford to ignore the benefits of better requirements management!

Bibliography

[1] Alsop. S, "The Trouble with Software Is …It Sucks", Fortune Magazine, June 10, 1996

[2] The Standish Group, Chaos Report, 1994

[3] Davis, A., "Software Requirements - Objects, Functions and States", Prentice Hall, 1993

[4] Boehm, B. and Papaccio, C. "Understanding and Controlling Software Costs*", IEEE Transactions of Software Engineering*, Oct 1988

[5] Sheldon, F. et al, "Reliability Measurement from Theory to Practice", *IEEE Software,* July 1992

[6] Tavolato, P., and K. Vincena. "A Prototyping Methodology and Its Tool." In *Approaches to Prototyping,* R Budde et al., eds., Berlin: Springer-Verlag, 1984.

[7] Dion, R. "Process Improvement and the Corporate Balance Sheet," *IEEE Software,* (July 1993),pp. 28-35

[8] Grady, R. "Practical Software Metrics for Project Management and Process Improvement", Prentice-Hall, Inc. 1992