Requirements Change: What's the Alternative?

Alan M. Davis
U of Colorado at Colo Sprs
College of Business
PO Box 7150
Colorado Springs, CO
80933-7150 USA
adavis@uccs.edu

Nur Nurmuliani
U. of Technology, Sydney
Faculty of Info Technology
PO Box 123
Broadway, NSW 2007
AUSTRALIA
nur@igreen.net

Sooyong Park
Sogang University
Department of Computer
Science and Engineering
Seoul, SOUTH KOREA
sypark@sogang.ac.kr

Didar Zowghi
U. of Technology, Sydney
Faculty of Info Technology
PO Box 123
Broadway, NSW 2007
AUSTRALIA
didar@it.uts.au.edu

Abstract

Numerous studies have shown that a software project's cost, schedule and defect density escalate as the rate of requirements change increases. Yet none of these studies have explored the effects of not making requirements changes in response to changes in user needs. This paper explains why a project incurs just as much, if not more, risk when requirements changes are suppressed.

1. Introduction

Although many papers have analyzed negative effects of requirements change, none have addressed positive effects of requirements change. This paper differentiates between requirement and need. Although the two terms are synonyms according to thesauri, we use requirement to indicate a documented, externally-observable characteristic of a desired system [1], and need to indicate the actual need of users, customers, market, etc. Requirements change encompasses

- requirements volatility, a term defined as a measure of the number of requirements changes (additions, deletions, and modifications) [2] divided by the number of requirements for a given period of time [3], and
- requirements creep, a term defined by Jones [4] as "frequent changes in requirements," and by Carter, et al. [5] as changes that result "in extensions to and alterations of the software's functionality and scope."

Three factors contribute to requirements change: (1) changes to actual need, (2) changes to stakeholders' perception of what is needed [6], and (3) changes to the requirements document. Instances of the first case go unnoticed, but occur nonetheless; in fact, we can do nothing to alter their rate of occurrence. Instances of the second case emerge as a result of requirements elicitation, prototyping, thinking, seeing an early version of the system, and so on. Obviously we can reduce their rate of occurrence by avoiding practices that surface requirements, but that is mere folly. Instances of the third case are 100% controllable; we can choose to make such changes or not. So, what do authors mean when they discuss mechanisms to "control" requirements change? Figure 1 shows the eight situations related to changes. This figure shows how it is generally good to change requirements when needs change (YYY) and generally good to not change requirements when needs are not changing (NYN and NNN).

2. Why Do Requirements Change?

After requirements are agreed to, they will change as the result of [1]: (1) Stakeholders review the document and that will trigger the desire for something new, (2) stake holders play with a prototype and that will trigger

		Critical Needs Changing			
		Yes		No	
		Perceptions Changing		Perceptions Changing	
		Yes	No	Yes	No
Requirements Changing	Yes	YYY: Req'ts changes made in response to changing needs	YNY: Req'ts changes being made for capricious reasons	NYY: Req'ts changes being made for capricious reasons	NNY: Req'ts changes being made for capricious reasons
	No	YYN: Req'ts changes not made in spite of changing needs	YNN: Cannot respond to changing needs that we don't know about	NYN: No changes made to req'ts because no needs are changing	NNN: No changes made to req'ts because no real needs are changing

Figure 1. Kinds of Requirements Change

the desire for something new, (3) the situation in which the system is used changes, and (4) customers use the current version of the system and that will trigger the desire for something else they'd like to have done. Nurmuliani, et al. [7] investigated how developers classify requirements changes. Those who classified requirements changes by the reason for change (60% of the test subjects) used a combination of these categories: (1) changes to product strategy, (2) changes to environment, (3) scope reduction, (4) design improvement, (5) realization that a requirement had been missing, (6) unclear requirement statement, (7) realization that original requirements were not testable, or (8) enhancement.

Carter, et al., [5] provide insight into how risk analysis is essential for determining which requirements changes should be accepted for inclusion in an iteration. Lam and Shankararaman [8] provide many suggestions on how to manage requirements change, including being more aware of changing user and customer needs (as opposed to documented requirements), performing risk analysis of making changes (also see Antón and Potts [9], who make an excellent case for performing a benefits analysis of requirements change as well), identifying conflicts among requirements and among requirements changes, and performing overt prioritization and triage [10].

On average, between 20% [4] and 53% [11] of the original requirements change by the time the project completes. Yourdon [12] and Young [13] report that projects became over-budget and/or late whenever requirements changes exceed ½-1% per month. We have no doubt that this is true, but seriously question the alternative, namely to suppress requirements changes as needs evolve (box YYN in Figure 1).

3. Perceived Negative Impact of Making Requirements Changes

Most authors who discuss impact of requirements change emphasize negative impacts of such change, e.g.

- "Requirements volatility (RV) is generally considered an undesirable property" [14].
- "Changing requirements are recognized as a major cause of project failure" [8].
- "Requirements volatility causes the software to have a higher defect density" [15].
- "Change . . . is often seen as a menace to established order" [16].
- "Requirements volatility is considered to be a major risk to . . . complex software projects" [7].

This is logical when considering only the cost of satisfying documented requirements. We know for example that the cost to develop a system (C_D) is proportional to the number and complexity of the elements of the set of originally specified requirements (R) and the number of and complexity of the elements of the set of changes made to those requirements (ΔR) during the development process, $C_D \sim R, \Delta R$. To see how silly it is to focus just on the cost to develop, we can minimize it quite easily: just minimize R. This may seem ridiculous, but when we talk about how requirements change causes problems on projects, aren't we suggesting that we find a way to minimize ΔR, which is just as ridiculous?

4. Negative Impact of Not Making Requirements Changes

It makes little sense to try to minimize C_D; after all, it is trivial as shown previously. What we should be doing is trying to minimize the cost to satisfy (C_S) the customers'/market's needs. And, this cost is proportional to the number of elements in and the complexity of the elements of the set of needs (N) and the number of elements in and the complexity of the elements of the set of changes to those needs (ΔN), i.e., $C_S \sim N, \Delta N$. But more importantly, C_S is proportional to C_D and the degree of discordance (δ) between R and N, and ΔR and ΔN, i.e., $C_S \sim C_D, \delta(R, N), \delta(\Delta R, \Delta N)$. Replacing C_D, we get, $C_S \sim R, \Delta R, \delta(R, N), \delta(\Delta R, \Delta N)$. Since we cannot control N or ΔN, attempts to minimize ΔR are moot. The only way to minimize C_S is to minimize discordance of ΔR and ΔN regardless of the negative side effect on C_D.

The preponderance of data indicates that the more requirements change, the higher the cost, the longer the development, and the higher the defect density. We will use the term negative project success factors to capture the concept of customer dissatisfaction, in general, and higher cost, longer development, and higher defect density, specifically, as shown in Figure 2[1]. However, let us examine in more detail the phenomena occurring at points A, B, and C on the graph of Figure 2.

- At point A, requirements are not changing (bottom row of Figure 1), so there appears to be no negative impact on project success. However, what if needs are changing, and we inhibit change to requirements? This is common among organizations who think the solution to escalating costs is to prevent requirements change. Figure 3 expands point A. Note that if needs are not changing, point A on Figure 2 is accurate and minimal, but if requirements are not being updated and needs are changing, then negative project success factors grow (right side of Figure 3 culminating at point A′).

---

[1] Axes on this and subsequent figures are void of scale; the non-decreasing nature of the graph is important, not its shape. As the granularity of requirements increases, the graph's shape becomes more predicttable, as demonstrated by Lavazza and Valetto [17].

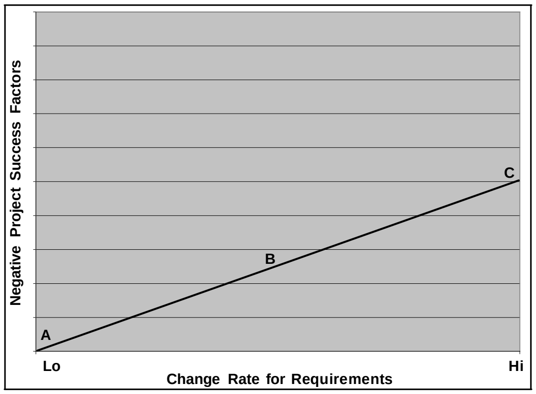

Figure 2. Relationship of Req'ts Change to Negative Project Success Factors

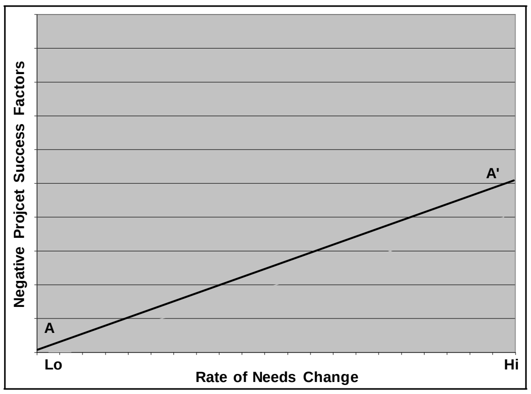

Figure 3. Relationship of Needs Change to Negative Project Success Factors at Point A

- At point C, documented requirements are changing (top row of Figure 1), so there are negative impacts on project success factors. However, what if actual needs are not changing, and we are simply making changes to the documented requirements for non-project critical reasons such as poor leadership, ego gratification, and so on? Figure 4 expands the situation occurring at point C. Note that if actual needs indeed are changing, point C on Figure 2 is accurate but is a minimal impact point, not a maximum impact point. If actual needs are not changing, but documented requirements are being changed nonetheless, then negative project success factors grow even more (left side of Figure 4, corresponding to boxes YNY, NYY, and NNY, culminating at point C′).
- At point B, documented requirements are changing moderately, so there are moderate impacts on project success factors. However, what if actual needs are changing more rapidly? Or less rapidly? In both cases, project success becomes less likely. Figure 5 expands the situation occurring at point B (left side of the figure shows escalating risk when needs are not changing; right side shows escalating risk when needs are changing more rapidly than the documented requirements are changing). Note that point B on Figure 2 is a minimal impact point, not a moderate impact point.

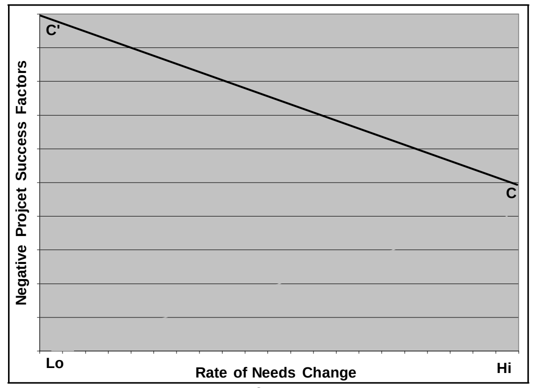

Figure 4. Relationship of Needs Change to Negative Project Success Factors at Point C

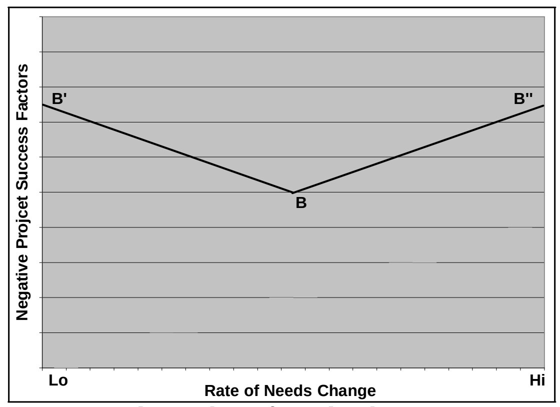

Figure 5. Relationship of Needs Change to Negative Project Success Factors at Point B

The phenomena at these points make it clear what is happening: requirements change must remain proportional to change in actual needs. It is quite easy to control requirements change: simply do not make them. The result will be a project that completes on schedule, within budget, and with minimal defect density, the subjects that are so often correlated with "project success." However, if we consider the ultimate (most critical) defect to be a failure to meet a customer need, then requirements changes must be proportional to needs changes (regardless of the timing of changes as reported in [15][18]), and will result in the lowest critical defect density. Putting Figs 3, 4, and 5 together gives us Figure 6. Note that points A, B, and C of Figure 2 all become minimal points in Figure 6 when examined from the requirements change axis. This becomes more obvious when we rotate Figure 6 so that the "change rate for requirements" axis is facing us, as shown in Figure 7.

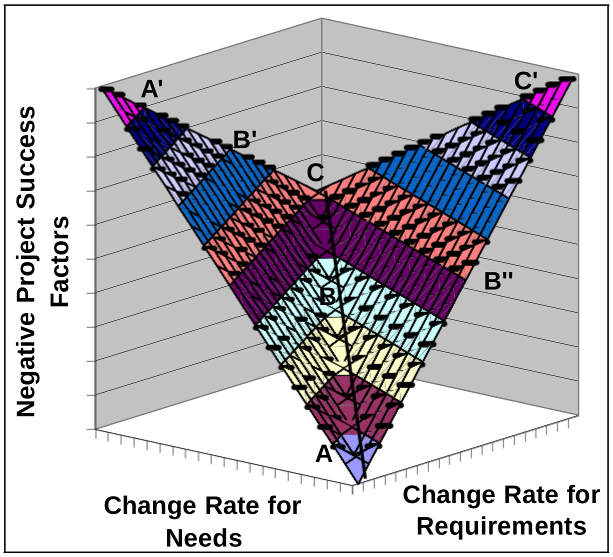

Figure 6. Overall Relationship of Req'ts Change and Needs Change to Negative Project Success Factors

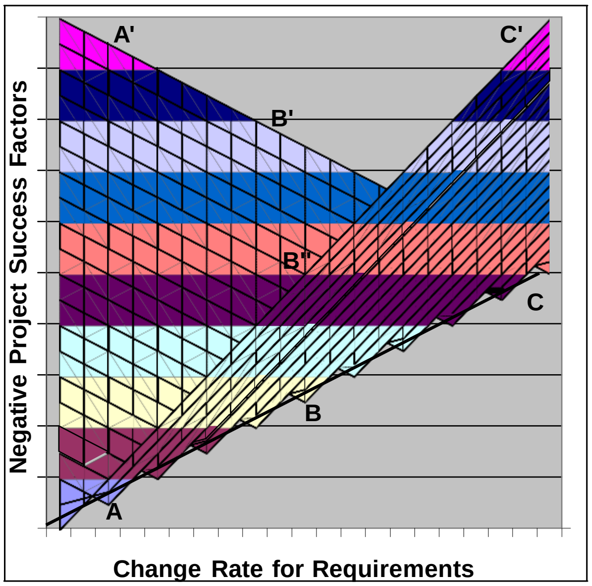

Figure 7. Relationship of Req'ts Change to Negative Project Success Factors

5. Summary & Conclusions

Many researchers have accurately proclaimed that as requirements change, the likelihood of project failure increases. This paper argues that although this phenomenon is important, equal attention should be given to changes to users' and customers' needs, over which we have no control. As critical changes occur to these needs, we have a choice of incorporating them in our requirements (thus increasing project risk but simultaneously decreasing product risk [19]) or ignoring them (thus decreasing project risk but simultaneously increasing product risk [19]).

References

[1] Davis, A., Just Enough Requirements Management, New York: Dorset House, 2005.

[2] Costello, R., and D.-B. Liu, "Metrics for Requirements Engineering," J of Sys and Soft, 29, 1 (Apr 1995), 39-63.

[3] Nurmuliani, N., et al., "Requirements Volatility and its Impact on Change Effort: Evidence-Based Research in Software Development Projects," 2006 Australian Workshop on Requirements Engineering, Adelaide, Australia: U of South Australia, 2006.

[4] Jones, C., "Strategies for Managing Requirements Creep," IEEE Computer, 29, 5 (May 1996), pp. 92-94.

[5] Carter, R., et al., "Evolving Beyond Requirements Creep: A Risk-Based Evolutionary Prototyping Model," Intern'l Symp on Requirements Eng'g, Los Alamitos, CA: IEEE Computer Society Press, 2001, pp. 94-101.

[6] Davis, A., and K. Nori, "Requirements, Plato's Cave, and Perceptions of Reality," IEEE Work on Requirements Eng'g for Services (REFS), Los Alamitos, CA: IEEE Computer Society Press, 2007.

[7] Nurmuliani, N., et al., "Using Card Sorting Techniques to Classify Requirements Change," Inter'l Conf on Requirements Eng'g, Los Alamitos, CA: IEEE Computer Society Press, 2004, pp. 240-248.

[8] Lam, W., and V. Shankararaman, "Requirements Change: A Dissection of Management Issues," Inter'l Work on Requirements Eng'g: Foundations for Software Quality, Heidelberg, Germ, June 1999; also in 25th Euromicro Conf, Los Alamitos, CA: IEEE CS Press, pp. 244-251.

[9] Antón, A., and C. Potts, "Functional Paleontology: The Evolution of User-Visible System Services," IEEE Trans Software Engineering, 29, 2 (Feb 2003), pp. 151-166.

[10] Davis, A., "The Art of Requirements Triage," IEEE Computer, 36, 3 (March 2003), pp. 42-49.

[11] Standish Group, The CHAOS Report, www.standishgroup.com, 1995.

[12] Yourdon, E., "A New Perspective on Metrics," Total Metrics, 1, 4 (September 2000), pp. 1, 6.

[13] Young, R., Effective Requirements Practices, Reading, MA: Addison-Wesley, 2001.

[14] Nurmuliani, N., et al., "Analysis of Requirements Volatility during Software Development Life Cycle," Austral Soft Eng'g Conf, Los Alamitos, CA: IEEE CS Press, 2004, pp. 28-37.

[15] Malaiya, Y., and J. Denton, "Requirements Volatility and Defect Density," Inter'l Symp on Soft Reliability Eng'g, Los Alamitos, CA: IEEE CS Press, 1999, pp. 285-294.

[16] Regev, G., et al., "Creativity and the Age-Old Resistance to Change Problem in RE," IEEE Inter'l Conf on Requirements Eng'g, Los Alamitos, CA: IEEE Computer Society Press, 2006.

[17] Lavazza, L., and G. Valetto, "Requirements-Based Estimation of Change Costs," Empirical Software Engineering, 5, 3 (November 2000), pp. 229-243.

[18] Javed, T., et al., "A Study to Investigate the Impact of Requirements Instability on Software Defects," ACM Software Engineering Notes, 29, 3 (May 2004), pp. 1-7.

[19] Gorschek, T., and A. Davis, "Assessing the Quality Requirements Process Changes," Inter'l Work on Requirements Eng'g Foundations for Soft Quality, Porto, Portugal, 2005.