

Conflicting forces for Software Reliability in e-Business

Ram Chillarege

Chillarege Inc., New York, September 2002

www.chillarege.com

Abstract -- The design point of e-business web application and the market demands for reliability have changed considerably in the past few years. Client ramp rates and business pressures cause conflicting forces, which, if not judiciously managed, can spell an engineering and business disaster.

I. THE NATURE OF RELIABILITY GROWTH

Reliability growth is experienced by products over a considerable length of time. A naive belief is that most of the growth occurs over the period of testing. While there is growth during testing, there is also very substantial growth in reliability over the several months in the field, and over the years through multiple releases. This fact while recognized in the industry, is often not the focus of discussion in software reliability. Academic discussions focus substantially on the testing, modeling, architecture and development process management [1-2]. When viewed in terms of the overall lifetime of a product, the post release maintenance contributes substantially to overall reliability.

To drive home this point let us revisit a result published in a case study which computes reliability from field data [3]. Figure 1 reproduces the reliability growth measured over multiple releases from that case study. It shows that (a) there is substantial reliability growth that occurred in the first three to nine months following the general availability. (b) The second release, which initially started at about the same reliability figure achieved levels better than release 1. The case study was from the early '90s, at the height of the client-server and the desktop business. Their customer growth was almost linear and the market afforded, what now seems like a luxury in time to achieve the reliability growth.

II. THE E-BUSINESS FORCES

e-business applications have dramatically changed the traditional design point for applications. And the market where they operate often stretch the limits of current software engineering capabilities.

A. Force - Ramp Rate

In the client-server era, the ramp rate of client seats was limited to a considerable degree by the IT infrastructure. Thus, deployment of an application would assume a finite ramp up time, and a relatively controlled growth in usage. These limits have for all practical purposes disappeared. Client seats in the web application world can ramp almost indefinitely. The zero foot print client and pervasive availability of network connections have transformed deployment. With a hosting strategy and application services model, web based applications can be deployed worldwide with a presence in all time zones instantly. Added to this, the user community, thanks to the internet, expects a 24x7 operation. Consequently, the luxury of several months to ratchet reliability is vanishing.

B. Force - Time to Market

Business pressures during the dot-com bubble created a sense of urgency that was unforeseen. This led to a behavior which valued time to market as the single most determining criteria for success. The belief that a first mover advantage was critical, as evidenced by a few isolated successes hyped the notion further. Even with the

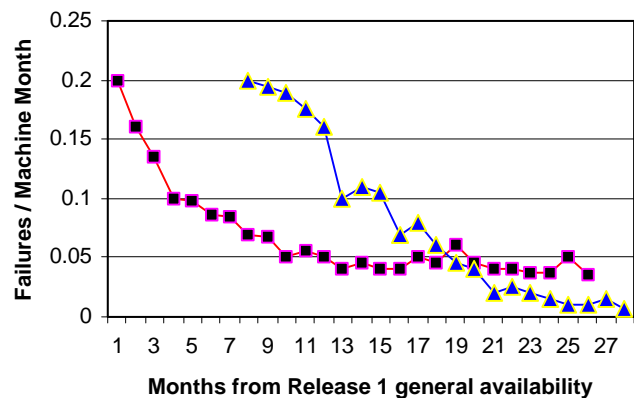


Figure 1. Reproduces reliability growth curves from [3] illustrating the growth experienced during field usage of two consecutive releases from a widely distributed software product.

bubble behind us it is not evident that a reversal of this faith has occurred. Reinforcing the reason for us to examine these forces judiciously.

III. CONFLICTING FORCES

There is a fundamental conflict in the two forces just described. Force A which implies an increasing ramp rate of client seats, shortens the window of time available in the field to ratchet reliability. The only reasonable strategy is to shoot for higher entry levels of reliability at release. While most IT shops will try to stage a rollout, the reality is that the overall window is much shorter, and worse, the ramp quite unpredictable.

Force B which drives shorter time to market, creates pressure from a different dimension. It influences values and methods of engineering which can undermine better judgment.

A remnant of the desktop era created a belief that entry levels of reliability are not critical, while time to market is. The trade-off is never quite so simple, but the rationalization very appealing.

An unfortunate outcome is the birth of what I call the "Quick and Dirty" paradigm. The thesis being that a quick and dirty solution to market would gain time to market and establish an early mover advantage. If ramp rates are low, then there is adequate time to fix software reliability after general availability. A rationalization that seemed to fit well with a growing reverse culture against overburdened processes. The method artificially shortened development cycle time, so long as there was slack at the customer end affording an opportunity to "Fix Later". If all worked well and no major disaster ensued, success was declared. The validity of the thesis, however remains largely untested for it hinges on more than the parameters just discussed.

The higher ramp rates of clients in the e-business world do not afford us such a luxury. But, that does not mean the Quick and Dirty approach is abandoned. Its appealing rationale, albeit faulty, can charm many an unsuspecting manager. The Achilles heel lurks in the shadow of the changing demands from e-business.

A large part of what precipitates the problem in the engineering world, is a belief that producing a quality product takes time. True. But what is equally true, and much less recognized, is that producing a poor quality product can take longer.

IV. CONCLUSIONS

While these conflicts do not stop the development of products and services, they impact cost, quality and development time. As the e-business web application model is still in its early stages several new advancements are underway in software engineering and programming technology. Some of them help modulate the impact from these conflicts. Others can make it worse by forcing the leading and bleeding edge of technology. In conclusion, there are a few observations that capture where we are now and where we are headed.

1. In a nascent business segment distinguishing engineering failures from business failures is not easy from publicly available data. Failure can be caused by either, or both.
2. On the other hand, successful new businesses can still have engineering problems looming over them. It is this segment that critically needs sophisticated software engineering analysis and judgment.
3. The potential crisis of software reliability, as discussed in the paper is temporarily mitigated by an increasingly more robust middle ware layer. Thus, a standard three tier architecture, can deliver significant function with relatively small amount of custom application code. The smaller fraction of application code riding on more robust software infrastructure has temporarily helped new products survive the entry reliability threshold problem.
4. However, as time passes, and the application layers grow, the issues raised in this paper will become more prominent.
5. Finally, new software engineering methods such as Orthogonal Defect Classification, are capable of sophisticated process diagnosis [4] which aid faster management intervention and trade-off.

REFERENCES

- [1] M. R. Lyu, "Software Reliability Handbook", McGraw-Hill, 1995
- [2] J. Musa, "Software Reliability Engineering", McGraw-Hill, 1998
- [3] R. Chillarege, S. Biyani and J. Rosenthal, "Measurement of Failure Rate in Widely Distributed Software", IEEE Digest of papers, ISSRE 1995.
- [4] R. Chillarege, K. R. Prasad, Test and Development Process Retrospective - A Case study using ODC Triggers. IEEE Digest of papers, DSN 2002.