# Total Quality Management and Software

Adrian Kosmaczewski

2007-12-18

Total Quality Management is one of the founding pillars of modern mass-production economy, of which the software industry is by far the youngest (and most rebel) child. This article will provide a short discussion on some TQM principles and about their applicability to software projects.

## TQM Principles

Joel Spolsky has written a brilliant introduction to software testing in his famous "Joel on Software" blog:

> Software has bugs. CPUs are outrageously finicky. They absolutely refuse to deal with things that they weren't taught to deal with explicitly, and they tend to refuse in the most childish of ways. When my laptop is away from home, it tends to crash a lot because it can't find the network printer it's used to finding. What a baby. It probably comes down to a single line of code somewhere with a teensy tiny almost insignificant bug in it.

(Spolsky, 2000)

The software industry has been a victim of its own success. Software companies have experienced the fastest growing rates in the history of capitalism, and this rush to conquer juicy worldwide markets, and to offer services to literally billions of human beings has, more often than not, been done at the expense of proper quality guidelines.

William Deming proposed 14 key principles that form the basis of what became TQM in the second half of the 20th century:

1. Create constancy of purpose for the improvement of product and service (…)
2. Adopt a new philosophy of cooperation (win-win) (…)
3. Cease dependence on mass inspection to achieve quality. Instead, improve the process and build quality into the product in the first place.
4. End the practice of awarding business on the basis of price tag alone. Instead, minimize total cost in the long run. (…)

5. Improve constantly, and forever, the system of production, service, planning, of any activity. (…)
6. Institute training for skills.
7. Adopt and institute leadership for the management of people, recognizing their different abilities, capabilities, and aspiration. (…)
8. Drive out fear and build trust so that everyone can work more effectively.
9. Break down barriers between departments. Abolish competition and build a win-win system of cooperation within the organization. (…)
10. Eliminate slogans, exhortations, and targets asking for zero defects or new levels of productivity. (…)
11. Eliminate numerical goals, numerical quotas and management by objectives. (…)
12. Remove barriers (…) abolishing the annual rating or merit system that ranks people and creates competition and conflict.
13. Institute a vigorous program of education and self-improvement.
14. Put everybody in the company to work to accomplish the transformation. The transformation is everybody's job.

(Wikipedia, emphasis added)

In the following section, I will discuss how these principles overlap and intersect, and how they can be applied (and have been applied historically) in software companies.

## Focus on the People

From the above list, some commonalities appear for all and each one of the principles: the biggest one in my opinion is the focus on the people. As such, knowing that software is a pure mind product, completely intangible and tremendously flexible, I think more than ever that software is a human & social process. This focus in people is particularly evident in principles 2, 6, 7, 8, 9, 10, 13 and 14.

For example, reducing the causes of internal and external conflict in the organization (principles 12, 9, 2), makes software developers able to collaborate with other departments or companies, creating integrated software suites that become de facto standards in their respective industries. A very extreme example of this phenomenon is Microsoft's Office System, which was created from separate products created by different teams, such as Word and Excel. The synergy created by the developers working together, not only unifying the UI look & feel or the programming model, but also increasing the interoperability of both systems, created a product that effectively is much more than just two products together.

Moreover, the Peopleware book by DeMarco and Lister showed that historically,

the most successful software companies have been those that excelled in creating a human-centric environment:

> In 1982, (Mitchell Kapor) founded Lotus Development Corporation, for which he is most noted. While there, he revolutionized corporate workplace culture by making diversity and inclusivity top priorities in his goal for creating an environment that attracted and retained employees. There were many "firsts" for Lotus, including being the first company to sponsor an AIDS Walk event in the mid-80's and refusing to do business with South Africa due to Apartheid.

(Sterling-Hoffman)

Thanks to a sharp hiring process, a series of innovations in their flagship spreadsheet product, and a progressive corporate culture, Lotus dominated the software landscape of the 80s. Today, Google follows very closely Lotus' steps (Google, 2007a), and their brilliant results in the last few years seem to confirm this trend. Google applies principle 13 very strongly, allowing their employees to use 20% of their time in their own projects (Google, 2007b). This is resulting in an incredible amount of code, used internally and also released as open-source projects, such as the MacFUSE project (Google Mac Blog, 2007):

> Google is a fantastic company to work for. I could cite numerous reasons why. Take the concept of "20 percent time." Google engineers are encouraged to spend 20 percent of their time pursuing projects they're passionate about. I started one such exciting project some time back, and I'm pleased to announce that Google is releasing the fruits of this project as an open source contribution to the Macintosh community. That project is MacFUSE, a Mac OS X version of the popular FUSE (File System in User Space) mechanism, which was created for Linux and subsequently ported to FreeBSD.

## Conclusion

Deming's principles are today, more than ever, extremely important in the software industry. With a very high rate of turnover and burnout, software companies are faced with the choice of stop considering their staff as a "resource" but rather as an "asset". The most successful companies in the field are not only those that are able to cut costs effectively (following the fourth principle) but also those that arrive to empower their staff, increasing creativity, well-being and innovation, which ultimately leads to market leadership and economic success.

## References

DeMarco, Tom & Lister, Timothy, "Peopleware - Productive Projects and Teams, 2nd Edition", 1999, Dorset House Publishing, ISBN 0-932633-43-9

Google, “Top 10 Reasons to Work at Google”, 2007a [Internet] http://www.google.com/jobs/reasons.html (Accessed April 5th, 2007)

Google, “What’s it like to work in Engineering, Operations, & IT?”, 2007b, [Internet] http://www.google.com/support/jobs/bin/static.py?page=about.html (Accessed April 5th, 2007)

Google Mac Blog, “Taming Mac OS X File Systems”, January 11th, 2007, [Internet] http://googlemac.blogspot.com/2007/01/taming-mac-os-x-file-systems.html (Accessed April 5th, 2007)

Lewis, W.; Veerapillai, G.; “Software Testing and Continuous Quality Improvement”, Auerbach Publications, 2005, ISBN 0-8493-2524-2

Spolsky, J.; “Top Five (Wrong) Reasons You Don’t Have Testers”, [Internet] http://www.joelonsoftware.com/articles/fog0000000067.html (Accessed April 5th, 2007)

Sterling-Hoffman, “Opening Doors To Higher Education”, [Internet] http://www.sterlinghoffman.com/newsletter/articles/article140.html (Accessed April 5th, 2007)

Wikipedia, “William Edward Deming”, [Internet] http://en.wikipedia.org/wiki/W._Edwards_Deming (Accessed April 5th, 2007)