Stephan Schwab 2012

Essays
On Software Development

Thoughts on Quality, Management, Practices

Essays On Software Development

Thoughts on Quality, Management, Practices

This version was published on 2012-05-02

This is a Leanpub book, for sale at:

http://leanpub.com/essays-on-software-development-2012

Leanpub helps authors to self-publish in-progress ebooks. We call this idea Lean Publishing. To learn more about Lean Publishing, go to: http://leanpub.com/manifesto

To learn more about Leanpub, go to: http://leanpub.com

Contents

Introduction

Over the years I have written many articles on my personal blog. Some are more personal while others are random thoughts or commentary on whatever came to mind. However, there are also posts about software industry topics, which, after some time has passed, become more interesting the older they are.

This book contains these blog posts. I have slightly edited them here and there to improve the formatting or fix some mistakes that made it into the blog.

It is my plan to republish a new edition of this little book once in a while. In future editions some articles may be removed to make room for newer articles. I want this book to stay small.

My personal blog can be found at:

> http://blog.stephan-schwab.com

The chapters of this book are ordered alphabetically. I found it hard to put them into any particular order as that would appear as giving more importance to one topic over another.

Feedback Desired

One motivation for me to publish this book is to get my writing and thoughts in front of a larger audience. I love feedback and a dialog with other members of the software industry community.

The easiest way to get in touch me is to send email. Here is my address:

> sns@caimito.net

You may write in English but if you prefer German or Spanish, don't hesitate to write email in these languages too. I'm fluent in all three.

About Me

My name is Stephan Schwab. I started writing software in 1981 when I was 14 years old. My mother had a Philips P2000M[1] computer at our home in Germany. She used the system to do accounting. Driven by my youthful curiosity I discovered an unused program cartridge that was a BASIC interpreter and from there on I taught myself programming with the help of the manual that came with it.

Today I am a Software Development Consultant and Coach. It appears that I am a writer too :-) Over the years I learned and used about two dozen programming languages ranging from very low-level C up to very high-level Ruby and am fluent in three human languages (German, English, Spanish). So far I have lived about twelve years outside my native Germany - mostly in Latin America and the United States.

As coach I use a whole team approach to help each role (customer, analyst, tester, programmer) to reach their full potential and create great products. A recent focus of my work is Acceptance Test-Driven Development and Activity-Centered Design. I view both techniques as essential tools to create smarter software that actually meets the needs of the users and businesses that pay for their development.

As consulant I offer advice on a broad range of technology topics based on my long- term experience in the field.

One of my more recent achievements in the space of packaged software has been the development of a software product called Caimito One Team (Agile Project Management and Collaboration Tool). In the past I co-designed and co-developed Convince (presentation graphics), ErgoCalc (spreadsheet) and DEO (Documents of Embedded Objects - a platform similar to Microsoft OLE2).

I co-founded and served as Geschaeftsfuehrer (CEO) for SoftStream Development (software development), VentureNET (Internet Service Provider and SAP consultancy), Farside Communications (Internet Service Provider and Network Security consultancy), and DINX (Internet Service Provider).

The picture on the cover shows me riding a horse in the rainforest of Panama where my family and I lived for several years. Visitors of my blog[2] will discover I talk a lot about horses besides software development practices, management, and quality. There is the occasional article about off-road driving too.

[1] http://www.old-computers.com/museum/computer.asp?st=1&c=1042

[2] http://blog.stephan-schwab.com

Management

Pseudo Scientific Management Kills Innovation And Creativity

Numbers in the form of metrics are dangerous. They can become a two-sided sword and despite their usefulness collecting them can backfire on those who collect and publish them. Agile teams frequently show big visible charts[3] on the walls of their team space. The intention is to be transparent and let everybody know what gets achieved and where the problems are.

> *One of the XP values is Communication. There are many ways of communicating within the team, and with people outside the team. We generally prefer conversation for most purposes, but when it comes to trends, history, or sensitive subjects, a good approach can be what we in XP call a Big Visible Chart, or what Alistair Cockburn (Agile Software Development) calls an Information Radiator. A simple chart on the wall can bring important information to the attention of the team, the customer, and everyone else who passes through the area. The chart can provide important information, even politically sensitive information, without getting personalities involved or hurting feelings.* Ron Jeffries in Big Visible Charts[4]

As long as everybody understands that this data is to be used to improve and is not a means to re-introduce scientific management[5] all is fine.

> *I myself participate in this shift as I include lean flow management, queueing theory, Yesterday's Weather and the like in my lectures and classes .. and worry the entire time as I do so. I add chapters on craft, creativity and personalities, not as compensation, just as part of the mix. I don't see others putting those into the mix.* Alistair Cockburn in Taylorism strikes software development[6]

I have knowledge of a large development organization where this very transformation back from open and honest agile practices towards scientific management has already started.

Initially this organization used the common practice of relative sizing of user stories. Story sizes were expressed in points using a fibonacci scale (1, 2, 3, 5, 8, 13). They use a sophisticated issue tracking tool that is available for every team member. All user stories, epics and features are tracked in this tool to make sure that there is good traceability so that it is possible to see how requirements were developed.

[3] http://xprogramming.com/articles/bigvisiblecharts/

[4] http://xprogramming.com/articles/bigvisiblecharts/

[5] http://en.wikipedia.org/wiki/Scientific_management

[6] http://alistair.cockburn.us/Taylorism+strikes+software+development

After a while management discovered that it is difficult or impossible to compare the performance of teams with each other. First they were trying to encourage teams to estimate the "same things" in the "same way" hoping that by soft pressure the numbers would become comparable. They instructed lower ranking managers assigned to each team to help with that.

A while later a big change was made and now relative sizing has been abandoned. One point now is seen as equal to one day per programmer pair which equals to 6 hours per person. They still use the fibonacci scale and explain that by using fibonacci they take into account that for larger estimates (which are now in time) the lower accuracy of the estimate has been factored in.

At the same time the organization wants to define a standard set of tools that should be used by every programmer and tester. They actively discourage or even prohibit the use of non-approved tools thereby killing exploration and personal growth of their employees towards becoming good craftsman.

In my opinion this leads to a culture of fear and in the long-term will kill innovation. How can innovation happen, if all the paths of exploration have been closed except a pre-defined one?

Plus the organization risks that their employees will perceive the message this sends as being "just do what you've been asked to do". Many software developers derive a big portion of their motivation from personal satisfaction and pride of workmanship. This kind of scientific management does not leave room for that as the core principle is that everything is made comparable by simple numbers.

Written May 20, 2011.