**Late and Over-Budget**

**Introduction**

Why write a book with such a depressing, negative title? Part of the reason is that I tried to write a book with a normal, business-like title “Introduction to Agile Software Development” or some such, at the suggestion of an actual publisher. The trouble was that everything that I tried to write under that title seemed to be to be the most awful, deathly dull stodge. Trying to write about Agile, part of me somehow felt that I should write about it in “business bookese” even though that was not at all how I explained it in my training courses. It took about another 6 months, and a run-in (actually it was a phone conference so it was more of a hang-up/flounce-out) with a company chairman particularly lacking in understanding of the realities of the software development process before I started why I was having so much trouble writing a basic Agile how-to book. The central premise of a basic Agile how to book would, I suppose be, “use these Agile techniques and your life will be get better.” Only trouble was, guess what? It probably won’t.

My experience over the last few years has been working with software development teams, either as a “scrum master” – I don’t care what the zealots say, this essentially boils down to being a project manager – or as an “Agile coach,” doing my best to help development teams improve their process. Over this period I began to notice something slightly strange, or certainly different from what I’d been lead to expect. The majority of books that you read about project management, whether they be Lean, Agile or more “conventional” approaches to project management, such PRINCE2, talk about the enormous advantages that accrue from implementing whichever methods they are pushing. When bad experiences are discussed, they are only discussed as the “before” phase of a project which was then catapulted into joyous productivity.

This doesn’t really chime with my experience of being a project manager. The books simply weren’t describing the projects that I’d worked, no matter how closely we adhered to suggested methodologies, these projects remained places to go to experience pain. Perhaps this is what it would feel like if you were experiencing troubles with your life-partner and the only literature you could find on the subject consisted entirely of romances which ended happily ever after.

My experience of being a project manager is that no matter what methodology is being supposedly or actually implemented, no matter how bought-in to the methodology senior management, customers and stakeholders and team members are, there will still be a perception that the development team is not performing as it should: that it is late and over-budget and desperately needs to try hard and go faster. If you are a project manager responsible for software development and you disagree with this assessment, if you happen to be presiding over a software development team that is perceived by all those is manage it and pay for it as being professional, on-time, to-budget and essentially valuable to the organisation, I don’t really want to argue with you, go, enjoy this while it lasts, make hay while the sun shines. Maybe give this book a look some time in the future if things turn sour. They might.

 The argument of this book is pretty much as follows.

1. Late and over-budget is going to be the rule rather than the exception for most software projects that you ever work on. Most likely it isn’t your fault because:
2. There are at least three really powerful reasons why most project are LAOB (and maybe a fourth):
	1. The complex, sometimes chaotic nature of the problems that software development tries to solve (we’ll talk later about the Cynefin framework)
	2. The basic difference between models of reality i.e. plans, budgets, timescales and actual reality and the bizarre tendency of almost everyone to fixate and worship plans, budgets timescales, rather than reality.
	3. Any team that is performing on-time and to-budget will tend to attract extra work at least at least up to the point where it starts doing badly (the Buckaroo effect).
	4. Possibly because of point 3) below, it suits the management of your organisation to have software development continually perceived as being a “problem” (yes, I’ve read a bit of Machievelli).
3. This makes managing software development a miserable experience, largely because of the powerful desire of human to be consistent and deliver on their commitments.
4. Once you understand all this you can relax (a bit). Relax *and* do a better job. There are things that you can do to make your experience and contribution as a manager of software development a whole lot better.
	1. You can do yourself and your team a favour and concentrate on committing consistently to things that you can actually deliver.
	2. You can concentrate on delivering value to you senior stakeholders and customers and clients.

So this is the argument I’m going to try to expand on, explore and hopefully, finally prove. On the way, I will talk mostly about Agile practices for project management because those are the ones that I’m most familiar with and that I know from personal experience actually work. I’ll also talk about some concepts from Lean and Kanban, although that’s not my personal area of expertise. But, I’ll let you in on a preview, a big part of how start relaxing in step four is to focus on delivering real value to your stakeholders and customers. Of course this isn’t the whole answer sometimes the project that you’re working on delivers no discernible value to anyone, that the only value there is, is the empty value of delivering against the plan. In which case, might I suggest, you have to choose one of the 3 Rs – restructure, run-away or rot. Cheerful huh? We should probably move on.