

The Unbearable Lightness of Programming: a tale of two cultures

Laurent Bossavit, Exoftware mentor (www.exoftware.com)

This is a report from the trenches. The "hooks" by which it is meant to convince are not the author's litany of industry credentials (you won't have heard of most of the companies I worked for), lofty corporate status (I am a developer, technical lead and occasional project manager), or impressive degrees (I'm an autodidact and have none to speak of). Rather, you will recognize in my experiences something of a project you've been on recently, of a former or current team or company – you will remember suffering and never quite understanding why or what went wrong, and listen as if to a campfire story. At any rate this, rather than offering a sermon or a theory, is my modest ambition here.

Over the past two years I have had the opportunity to serve at two small companies as an Extreme Programming "Coach"; in other words, my mission in each case was to pioneer and develop the use of the Extreme Programming approach to software project and quality management. For most companies, and for most people, XP represents a fairly radical departure from the way software development is usually performed. We (by which I mean "the XP community") claim that this rather different way brings far better results than the usual way.

Other than their similar size (under fifty employees) and main business orientation (service businesses), the two companies were quite different, and in particular were home to strikingly different cultures. As the more recent of these two experiences is coming to an end, I am drawing several lessons from comparing the two, lessons that might be useful to people considering XP for themselves or for their business. (A basic familiarity with XP tenets, including its four values of Communication, Feedback, Simplicity and Courage, and its twelve (or so) practices, is presupposed in the rest of this article.)

Summary results

I want to start by summarizing my impact on both companies: in a nutshell, I would call it "close to nil". I believe that this is for interesting reasons, but right off the bat I intend it as a sort of benchmark for ambitious developers or project managers aspiring to become XP Coaches.

If you are a reasonably skilled technician, working in what ought to be good conditions for introducing such a radical shift in mentalities as XP, and achieve essentially nothing; there is no reason for you to feel bad about it. I've been there, and I suspect many people have been there - even among those who report successes with XP.

Whitepaper

There's nothing paradoxical about this, of course, as I hope reporting on my own experiences will make clear. As a community, our commitment to such a community-bound, culture-bound approach as XP is fuelled mainly on partial successes. These serve as intermediate goals and satisfy us, once reached, that we might in fact hope to achieve something that could be viewed as unequivocal, complete success: a thorough reorientation of a business culture, such that the business in question markedly outperforms its competitors in a manner clearly attributable to the XP mindset.

Both businesses where I was called on to introduce XP failed as businesses; that is, terminated all personnel or filed for the local equivalent of bankruptcy protection. I want to view this as a "failure" of XP in the strong sense of success suggested above, even in the presence of partial successes.

As we move forward, I will highlight relevant episodes showing how culture contributed both to the partial successes and to the ultimate failure. I will also throw in the occasional metaphor or bit of speculative theorizing... to get some relief from the inevitable anger and embarrassment I will suffer at recounting true stories of software development. (Don't tell my mom I work in software, she thinks I sell used cars.)

Two cultures

My first contact with XP came around as a result of a clash of cultures. Specifically, my family's culture - which places value on spending quality time with my two children - and my employer's culture - which among other things valued significant dedication from its employees as manifested in the form of 60-hour working weeks (or worse) had been clashing for quite some time. XP's insistence on sustainable pace was therefore one of the first things to capture my attention.

Yet another way in which the culture in that company differed from mine was that they favoured bucking French labour laws and having to hand out largish checks to disgruntled employees as a result, over the peaceable but potentially lengthy resolution of differences. And so I found myself, at one point, with just enough time on my hands between employers to dig into what this XP thing was all about, and found enough of a fit with my own values that I couldn't wait to use it in my next position.

Similarly, I found a large cultural gap between my next two positions. One was a start-up in the "marketing over the Web" space - that was just before the whole dot-com bust when this was still a proposition investors could be attracted to - with a typical age range of 22 to 28; I was older than the CEO by five years, but then pretty much everyone was older than the CEO. We were hip and we were going to take over the world.

The second was a former "body shop" which had been around for five years and had experienced as many changes of business strategy. At the time I came in, they had converted to doing mostly fixed-fee, in-house projects, mixing new development and third-party maintenance, and were mostly targeting customers in finance and banking, with some residual activity in smart cards technology. The marketing material also loudly proclaimed us as "object experts", although most of the projects involved C or some not very object-oriented C++.

Relativism

A number of respected writers on software development who also happen to be interested in cultural issues, for instance Jerry Weinberg or Alistair Cockburn, tend to a stance of "cultural relativism". At a fairly high level of description this amounts to saying that no culture is good or bad in the absolute. In more concrete terms and as applied to software, this is the view that we cannot identify "best practices" irrespective of the context, including the cultural context, in which they are applied; and that, in a given business, what matters is not so much what software development process is in use, as whether their particular process and their particular culture enables it to achieve results that ensure its survival as a business.

Both authors above have noted that the intrinsic difficulty of software projects is determined largely by customer demands (for quality, timeliness etc.) and by overall project size. As a simplifying model, Jerry Weinberg ranks broad categories of software development cultures on a five-step scale, mapping roughly to the CMM scale of one to five: Variable, Routine, Steering, Anticipating and Congruent. A Variable culture, for instance, is one in which programmers "just do their thing", and some software is shipped which, you know, basically works (even though it's got a lot of bugs). A Routine culture (the other common cultural pattern) is one in which software development is largely a rote affair, following what is usually a management-imposed process or lifecycle. For a cultural relativist, as long as the business stakeholders are happy with the results, one should refrain from passing judgment on the culture.

Things are different, of course, if due to external conditions (such as stock markets taking repeated nose-dives) the business in question sits up and decides it's got a problem; or if, external conditions being the same, such a business decides to tackle bigger or harder projects. Then, say even the most tolerant cultural relativists, chances are the business will have to embark on a process of change. This, of course, is where XP or other methodologies supposedly come in. It is a good reason to reflect on how culture and process interact, because a process change will always involve a cultural change.

And this observation conveniently gets me out of having to declare in favour of or against relativism, which is good, as I haven't yet formed a definite opinion in the matter. Since some businesses will have to change their culture in any case, and since my main professional interest is now in assisting businesses embarking on such changes, I can be content to study how culture affects XP, my preferred approach to a solution. And vice versa. I will, however, attempt to refrain from any critical assessment of either culture - and merely make note of my observations and hypotheses.

Results

I wrote about Bless, the Web and marketing guys, in a "fictional case study" for the first French XP book. We were a cool company, and I was really sorry to see it fail, though we did fail cleanly at least. We saw it coming three months in advance, management cleaned up the accounts and gave everyone notice. My account was presented as fictional because I was supposed to present an "ideal" XP project, or at any rate a successful one. Project Canon fell short on many counts, but it was enough of a partial success to serve as inspiration for a case study, and it highlighted which parts of the culture at Bless were a good fit for XP.

Whitepaper

I wanted to characterize each culture in three words each representing a positive value - that struck me as a good exercise to start getting at the corporate culture in your company, should you wish to do that. For Bless I came up with Passion, Daring and Glamour.

Of course Passion and Daring do not (even together) equate with XP's underlying value of Courage. We showed Passion when we, a very young business, went after glamorous large accounts, such as the local arm of one of the Big Five. We showed Daring when, upon being asked to deliver a corporate Web site brimming full with functionality in a drastically compressed schedule - we were due to go live on September 30th, as I recall, the company having been formally incorporated late in July - we decided to roll our own content management solution because all of the commercial offerings out there looked bloated and unsuited to the functionality that was asked for.

We showed Daring, of course, when we decided to adopt XP for part of that project after I showed up one day with packs of index cards and a gleam in my eye. I'd like to think that was also one of our rare instances of Courage. It worked well, too, to the extent that XP's approach to iterative planning and insistence on small, simple, thoroughly tested solutions delivered at regular intervals permitted us to make that impossible deadline... well, almost.

We dropped the ball though, in a clear illustration of what the difference between Daring and Courage is, when we failed to expand the process to the rest of the technical team in spite of our success with it and our many quality problems. Or again when we failed to keep our (not on-site) customer in the loop with respect to these schedule and quality problems.

I'm using Glamour for the last value because, in my opinion, we valued some aesthetic qualities of the software we were producing, but only those visible on the surface. We were (collectively, in the aggregate) blind to underlying elegance of the code, which strikes me as a common trait in many corporate cultures. So we discounted most measures of internal quality, such as level of abstraction, modularity, lack of redundancy.

Glamour also stood in the way of Courage insofar as we preferred to go with "standard" technology choices for our software development: J2EE application servers backed with "big name" SQL databases. My suggestions to look at alternative languages, alternative database technologies such as object databases, or even that JSP might not be the best Web development technology were discounted as "too off-the-wall" for our customers.

Similarly, at one point a management issue brought nearly all development to a halt - at a time when the customer's demands for timely delivery were growing ever more strident - as our CEO, who had until now been overseeing directly our mostly self-organized technical team, decided over our protests to bring in a CTO; a nice person with a prestigious degree who immediately proceeded to make enemies of virtually everyone on the team. As we later realized, the appearance of having "a person in control" had been valued more - by many of us, not merely the person at the top - than actually being in control of our development process.

Going back over the previous few paragraphs, it strikes me that they might possibly be construed as overly critical, overly negative. I want to stress again that I very much enjoyed myself there, how much I liked this culture, identified with it and subscribed to some, if not all, of its values. My point is that there was

Whitepaper

an imperfect match between our culture, our practices, and our market. The culture could perfectly well have been made to work, given different practices or a different market.

In the end, the downfall of this company was mostly as a result of investing much of our venture capital in ventures that had nothing to do with our core business - we opened up a complex of business offices, for instance, which at one time was even conceived of as an incubator for start-ups. One might say that this had more to do with management incompetence than with culture, and that at any rate we're straying far from the topic of software processes.

But in fact I think our behaviour showed a consistent pattern, whether in our software department or in other business activities. I might just be fooling myself, of course. But when I examine the cultural features that seem to be required by XP and compare them to that held by Bless, it strikes me that the main thing we were "missing" (in the non-judgmental sense of being different from a culture where XP would have been more effective) was a systematic preoccupation with results.

Enjoyment

In XP, you find at all levels a principle of looking first to the end result of what you're setting out to do; objectives are defined and precisely, formally measured in acceptance testing and unit testing. You might therefore think that a cool-headed culture with a view to the bottom line at all times might be a more fertile terrain for planting the seeds of XP. These were my thoughts as well, as I moved on to my next place of employment.

They turned out to be partially true to my expectations in being a culture of Economy. There's a maxim often applied to such cultures, that "they know the cost of everything and the value of nothing". Well, that wasn't true in our case. We knew the cost of many things; and thus, we avoided doing of some things that we knew to be costly, such as upgrading the LAN or the workstations, or programming in pairs. But we also did many things that we knew to be valuable, such as bidding on contracts based on short- term revenue opportunities, even if they weren't quite a match for our core business orientation or available skill sets.

The latter in particular wasn't so much of a problem because we were confident of being able to grow into new competencies - we were, to introduce another value, a culture of Potential. Our mix of experiences was skewed largely to junior engineers. Even our more experienced people were rarely put to work on projects that called upon previous experience with similar technologies; it was more often the case that adaptability was a key skill, as when a colleague of mine experienced mostly in SQL and Visual Basic was charged with leading a project involving, among other things, C++ and SNMP.

Here again, I wish to avoid passing judgment on the culture as such, but rather point to a mismatch between culture, practice, and market. A motto of "doing more for less expense" may serve many smaller companies well; so might a policy of presenting challenging work to junior employees. In our case, however, training was one of the areas of economy - with the result that junior employees were rarely up to such challenges.

We were, also, a culture of Conviviality - which you could easily mistake, and I did just that at first, for a culture of Communication. The difference between the

Whitepaper

two is that in Conviviality what is valued is the *act* of sharing information in a group setting - for instance, gathering at the company bar for drinks after the monthly all-hands "status report meeting" - rather than the nature, quantity or quality of the information thus shared.

Conviviality doesn't necessarily extend to honesty - as I experienced, for instance, when a colleague in Sales once asked me if, to help him close a contract deal, I would be willing to claim, in our conversation with the client, a graduate degree I didn't in fact have. (I had the greatest difficulty convincing him that this wasn't a good idea - in the end I had to point out that the correct information was available to any and all from my Web site, and that he would be taking a chance on the client being too dumb to use Google.)

Nor does it necessarily imply openness - the notion, core to XP, of sharing all information relevant to the project, including any problems meeting the schedule, with all of the team including the customer, actively clashed with our standard practice on several occasions. Once, I'd been brought in to manage a project suffering from a lack of visibility. Of course, as soon as visibility improved, people started raising various issues - which was taken by higher management to be a sign of conflict, as a result of which I was removed from heading the project, which was turned over to a colleague of mine who was already "saving" several such projects, and therefore presented little risk of generating conflicts. As for myself, I was reduced in rank to "mere programmer".

As with Bless, and again I might be fooling myself, I am tempted to pin down the eventual failure of this company on one major missing ingredient: a preoccupation with enjoyment. That of the customers, to start with - and we know how much importance XP places on that. I use the word "enjoyment" advisedly, rather than the common "satisfaction". We were hip to "satisfaction" - I was told time and again, "if we fulfil the contractual terms of the project - i.e. satisfy requirements, as stated in the requirements document - of course the customer will be happy". Experience showed this supposed correlation between satisfaction and enjoyment not to hold; several major clients, at any rate, turned out not to be happy with our services and chose not to employ us any more.

Similarly with the enjoyment of the employees. Beyond defensive management, of which we've seen one instance, we certainly committed most of the other sins of Teamicide listed by DeMarco and Lister : fragmenting teams by moving people in and out, fragmenting people's time by having them work on several projects, lowering quality standards to meet unreal deadlines... Part of the XP mindset is the notion that people are more productive when they're having fun. I think we must have subscribed to the more common maxim "If you're not miserable, it can't really be work".

Obliviousness

Looking back on the above, I find that I have merely sampled the ways in which a corporate culture can interfere with a prescribed philosophy and its accompanying practices, such as XP. My preliminary impression is that XP, a deeply pragmatic but also deeply humanistic philosophy of software development, will clash with any culture which isn't primarily focused on the **outcomes** of its activities, and holding itself as a whole accountable for such outcomes; and with any culture which isn't fully committed to **creating joy** in all such activities. But since I could well be wrong on that, I'd like to end on an even more general conclusion.

Whitepaper

I've made a habit in recent days of asking, at companies where I offer my consulting services, what the corporate culture is. I was surprised at how often this drew a blank. Culture, in my opinion, is largely oblivious, in the following sense. At neither company did we ever sit down and reflect upon our own fortunes as being determined by our culture of Passion, Daring and Glamour in one case, or Economy, Conviviality and Potential in the other. But, as I think my experiences show, these values were strongly affecting our chances of success with XP practices, even without our being conscious of them.

So we were oblivious to the ways in which our practices and the external conditions we contended with were interacting with the cultural values which we would most readily have articulated, had we bothered to make the effort. And we were consequently even more oblivious to how our practices and their effectiveness were impacted by the cultural values we weren't even considering, such as an outcome-oriented attitude in one case or a people-oriented attitude in the other. There's a famous quote about cultures which goes, "Culture is what remains when you have forgotten everything else", and that's what I have in mind here.

Reflection and Plasticity

And thus, insofar as I have learned anything from my experiences, what I have learned is this: that the most crucial feature of any successful corporate culture must be a willingness to reflect upon itself, to assess whether it needs to change itself, and if need be to embark on such changes as it finds necessary. It is immensely gratifying to find that these conclusions match that which my friend David Putman comes to in his article in this same issue, even though his is a story of a cultural success with XP while mine is a tale of two failures.

I am loath to use such a loaded term as "emotional maturity", with its implied judgment of value. Nor do I think the prescription to have cultural reflection and cultural change involve every level of the business; rather than be dictated solely by management, is necessarily a universal. For all I know top-down change could work in some industries, and some cultures, though by all reports it has consistently failed to work in the software culture and industry. But one thing I am certain of is that the kind of culture in which XP will thrive isn't one which can be achieved by top-down dictates.

And (I really want to say this loud and clear) though I have failed with XP twice already, I still like it very much. If you'll recall, my sense of "failure" was this: I failed to transform the corporate culture to a point where it could start making use of XP effectively. One of the things I like most about XP now, and which wasn't obvious to me at all when I started out (though I like to think I suspected), is how it makes obvious the plasticity, or lack of such, present in a corporate culture. Its splendidly unorthodox conception of software development serves as a litmus test for cultural problems of this sort.

Conclusions

The problem of how to *create* such plasticity where it doesn't exist, of course, remains entire. For this, I'm afraid; we must look beyond XP and to other disciplines (a small sampling of which I offer, purely as "teasers", in my References). We must, in other words, learn. But, to the extent that the sense I

Whitepaper

gave to "failure" is at all accurate, we now have an interesting sense of "success" to go with it - success *is* learning.

References

Agile Software Development, Alistair Cockburn, Addison Wesley 2002

Jerry Weinberg - <http://www.geraldmweinberg.com/>

Quality Software Management Series, Jerry Weinberg, Dorset House

Seeing Systems, Barry Oshry, Berrett-Koehler 1996

Darwin's Dangerous Idea, Daniel C. Dennett, Simon & Schuster 1996