Embrace Change

... an attitude assumed in ExtremeProgramming and the subtitle of KentBeck's manifesto on the same, ISBN 0201616416. Discussion of the concept continues here while the book is explored under the complete title, ExtremeProgrammingExplainedEmbraceChange.

... this attitude, tested in ExtremeProgramming is too valuable to be restricted to Software-Development only, it should extend to the whole business process ExtremeOpenBusiness and beyond.

Before you start design, get all the requirements.

Before you start development, freeze the requirements.

When even seemingly appropriate changes come along, resist them: they will lengthen your schedule and no-one will remember anything but that you were late.

Build a Change Control Process that ensures that proposed changes are looked at carefully and that no change is made unless it has passed intense scrutiny.

Oh yes ... and deliver a product that doesn't even satisfy the person who asked for it, because it's obsolete the day it comes out.

Face it: the requirements will **not** be known at the beginning, and they **will** change along the way. To be truly successful in software development, we must recognize this and ride the wave. The ExtremeWay is to build a system of people and code that will accommodate change as a natural process. We do this not by building for the future (YouArentGonnaNeedIt), but by building for change. Amongst our weapons ... amongst our weaponry ... are such elements as ... UnitTests and AcceptanceTests. We DoTheSimplestThingThatCouldPossiblyWork and then RefactorMercilessly. We learn to EmbraceChange. -- RonJeffries

---

I had a great conversation with a lady on the airplane about the way she runs her business, I run mine, and XP runs software development. The stability-by-living-with-built-in-change-response became the theme of our discussion. Having visited C3 project, I have this sensation that they are picking up the knife by the handle instead of the blade, while much of the industry is still figuring out how to safely pick up the knife by the blade. (I have to cogitate more on what I saw, but that's the direction I am thinking.) -- AlistairCockburn

Sometimes, I think people forget about the difference between *tracking* (traceability of who changed what, where, when, and why) and *control* (no changes are made without formal approval under intense scrutiny) and force the latter down their throats when the former would be more than sufficient. A good set of development tools, combined with a mere modicum of discipline, can easily fulfil most of your tracking needs without stifling progress (and morale) in a quagmire of totalitarian control. Granted, there are times and places where control is needed; I'm just observing that it's often overextended when traceability is mostly sufficient. -- BradAppleton

Certainly, here, we're talking about that reaction to change which is to hold it back. Regarding "tracking", perhaps you could help me out. If you're talking about TrackingRequirementsChanges, remember that I come from a place where the customer is queen and can say whatever she wants. Perhaps you would write a little page on TrackingRequirementsChanges saying how one might do it and (most of all) why. Thanks. -- RonJeffries

I'm not talking about tracking of requirements per se, but of anything and everything folks may attempt to track and control. This includes, but is not limited to things like requirements, changes to requirements, code, changes to code, and just about anything else you can think of. The subject originally came up because you mentioned formal change control procedures at the top of this page. That's just one good example of many in which there is often a tendency to go overboard with control when tracking will suffice (most of which can be done unintrusively by a decent SCM tool).

As for the "customer is queen" comment, I don't quite see the relation between that and what I was referring to above. I was, in fact, emphatically agreeing with the EmbraceChange comments above.

Regarding *how* to track requirements, if it's something that makes sense for you for the given project, I recommend tracking it the same way you would track defects and/or changes: with a tracking tool (I definitely don't recommend trying to manually maintain so-called "requirements traceability matrices" - there are decent tools for this, like RequisitePro and DOORS) . -- BradAppleton

---

Brad, I'm guessing from your comments that you know how to track requirements, **and why someone would want to**. In my world, my customer just says "do this next" and we say "ok". I haven't a clue why requirements tracking is in any way desirable, or what it means to track them. My request was that you share a summary of that knowledge with us here. -- RonJeffries

Eg: "Sometimes the current state of the system is easier to understand if you know how it got there." -- DaveHarris

True all too often. But not for a system in ExtremeNormalForm. -- KielHodges

You use Envy for version control, right? Why? Why wouldn't you want to keep your non-code software artifacts under version control as well? If they are non-text, then there may be issues of whether or not meaningful "diffs" can be constructed (it depends on the tool). But the same reasons for version-controlling the source and tracking fixes/defects apply to all non-source artifacts as well. You track changes to them the same way for the most part.

That's how you track requirements *changes*. To track the requirements themselves, if the customer demands traceability matrices (ouch) then get a tool. Otherwise, do whatever works best for you, so long as you track it. If you track it by having one or more test-cases represent each of one or more requirements being "implemented", that's great (though that only tracks successful implementation, as opposed to providing traceability to code & design; I'd use a tracking system for the latter as well if it was being insisted upon). -- BradAppleton

---

RequirementsTracking questioned.

I just came across RecoveryOrientedComputing - http://roc.cs.berkeley.edu/ and http://seotop.in.ua/. It immediately struck me that what they're doing is a form of EmbraceChange - or in their case, 'embrace failure'. Rather than emphasize proactive design for fault avoidance, it's about designing for reactive recovery from any failure. They describe their approach as moving the focus from Moore's law to Peres' law, this quote:

"If a problem has no solution, it may not be a problem, but a fact, not to be solved, but to be coped with over time." -- Shimon Peres

... which seems quite apposite to 'Embrace Change'

This quote is bilinked to <-> http://www.usemod.com/cgi-bin/mb.pl?GhostTown, where I use it as an argument to integrate community-friendly business into the wiki, instead of letting the poor admin alone with blindly fighting Spam.

Spam (in Wikis) is an ever increasing phenomenon, that reflects the inherent economic value in wikis. That is "a fact",.. "to be coped with over time". So

EmbraceChange and let the wikizen participate in their wealth-creating contributions.

Those active "vested" wikizens will share the workload with the admins to defend their strongholds. You are invited to develop this concept as <->http://aboutus.org/ExtremeOpenBusiness. -- FridemarPache

---