

Agility through Discipline: A Debate



Kent Beck
Three Rivers
Institute

Barry Boehm
University of
Southern California

Kent says: There are those who see agility in software development as a tradeoff, a sort of shaving of the wing spars. By giving up safety or rigor or accountability, we can go faster. From inside Extreme Programming, though, “agility or safety” or “agility or discipline” look like false dichotomies. From inside Extreme Programming, it seems that the only way to achieve the results we seek is to view the world in “both-and” terms instead of “either-or” terms. Why this disconnect?

My dictionary (www.m-w.com) defines discipline as: 3. a field of study; 4. training that corrects, molds, or perfects the mental faculties or moral character; 5. (a) control gained by enforcing obedience or order; (b) orderly or prescribed conduct or pattern of behavior; (c) self-control; 6. a rule or system of rules governing conduct or activity.

Let’s take a look at each of these definitions in the context of a prototypical Extreme Programming team.

A field of study. Extreme Programming describes a set of skills developers or customers should master on the way to becoming effective members of a software development team.

Training that corrects, molds, or perfects the mental faculties or moral character. I won’t make any claims for moral character, but test-driven development, refactoring, pair programming, continuous integration, and weekly planning (with estimation and tracking) all contribute to “perfecting” mental faculties.

Control gained by enforcing obedience or order. Extreme Programming fails on this account. The team is responsible for its own rules. In fact, the first

rule of XP is “They’re just rules.” This rule is not an excuse for unilateral behavior but a reminder that the team must maintain awareness of its own rules and continually refine them in the light of experience.

Orderly or prescribed conduct or pattern of behavior. The team members know what to expect of each other, so the team’s behavior is orderly.

However, the people doing the prescribing are the people doing the doing.

A rule or system of rules governing conduct or activity. Extreme Programming certainly is this. XP teams are conscious of their collective rules for planning, development, integration, and deployment.

As many rules as possible are embedded in tools. For example, the rule that says no code can be released without all the tests passing can be embedded in a script that automatically runs the tests and only releases changes if the tests all pass.

If Extreme Programming is said to be undisciplined, it is only in the sense of “control gained by enforcing obedience or order.” If this narrow definition of discipline is used, I say hooray for undisciplined processes. Efforts to force developers and customers to work according to a procedure developed by those not immediately responsible for results have uniformly failed. However, given a broader view of discipline, Extreme Programming is far more disciplined than most processes, providing a clearer collective picture of what activities are expected and more opportunities for learning. Indeed, without conforming to these positive senses of “discipline,” the social contract of Extreme Programming would rapidly disintegrate.

—Kent Beck

Agility is only possible through greater discipline on the part of everyone involved.
—Kent Beck

Barry says: Once upon a time, in the land of Metaphor, there lived a monkey and an elephant. They both lived on one side of a wide, swiftly flowing river. On both sides of the river there were many fruit trees. The monkey was very agile. He could climb to the top of the fruit trees and eat as much fruit as he needed. The elephant was very tall. He could reach up with his trunk and eat as much fruit as he needed.

But the trees grew taller. Soon the elephant could not reach enough fruit to eat. But he was strong and self-sufficient. He found that when he was hungry, he could just pull down a tree and have fruit to eat.

The monkey watched the elephant pull down most of the fruit trees. He was not happy. He said to the elephant, “Don’t do that! I will climb up the trees and get fruit for you.”

The elephant said, “I am hungry. I am strong. I can do things for myself.”

The monkey said, “You dumb elephant. Soon there will be no trees or fruit for you either.”

The elephant said, “I only work on one problem at a time. Things may change. Maybe more fruit trees will come. Or maybe the trees will get shorter. If they don’t, I’ll work out a solution then.”

The monkey had to agree. He only worked on one problem at a time too.

Soon there were no more fruit trees left on their side of the river. The elephant said, “I have a solution. I will go across the river and pull down trees over there.”

The monkey said, “You dumb elephant. That didn’t work on this side of the river, and it won’t work on the other side of the river. You will starve us both. Let’s duke it out. I am agile and will run rings around you.”

The elephant said, “That is fine with me. I am big and strong and I will pulverize you.”

So they began to duke it out. The monkey ran rings around the elephant. But he was not able to stop the elephant from trying to pulverize him.

The elephant thrashed around with his strong trunk and legs, trying to pulverize the monkey. But the monkey was too agile, and the elephant missed every time.

It was a hot day. Soon the monkey and the elephant got tired. They sat down and tried to figure out what to do next.

The monkey said, “I am agile. I will just scamper across the river and bring back some fruit.”

He got a running start toward the river and went

scamper, scamper, scamper ... splott! The river started to carry him away.

“That dumb monkey!” said the elephant. He waded into the river, picked up the monkey, put him on his back, and waded back to shore. They sat and thought for a long time while the monkey dried out.

Finally the monkey said, “Maybe this is a solution. You can carry me across the river on your back. When we get across, I can climb the trees and get enough food for us both.”

The elephant thought for a moment and then answered, “That sounds like it is worth a try.” So they tried it, and it worked.

And they lived happily ever after, until the end of their days.

**You don’t broaden the definition of “discipline” by rejecting parts of it.
—Barry Boehm**

Some morals to the story:

- Monkeys can do things elephants can’t do.
- Elephants can do things monkeys can’t do.
- Working on one problem at a time may not be a good idea.
- Duking it out may not be a good idea.
- Finding a collaborative win-win solution may be a good idea.

—Barry Boehm

KENT BECK’S RESPONSE TO BARRY BOEHM

The primary moral I take from Barry Boehm’s story is that he sees the discussion of what he calls “plan-oriented” methods in contrast to what I might call “reality-oriented” methods as a competition. I’m afraid I just don’t see it that way. Since he introduced the story form, I’ll respond in kind, although I can’t hope to compete with the charm of two cute creatures learning to get along.

Once there was a dinosaur and a furry little mammal, a bit like a mouse. Then an ice age began. The dinosaur couldn’t regulate its internal temperature, and so it died. The mouse, with its more adaptable metabolism, survived.

The dinosaur and the mouse weren’t in competition. They ate different things, lived in different places. Certainly, the passing of the dinosaur opened up many opportunities for the furry little creature to thrive, but the passing of dinosaur had nothing to do with the mouse. If it hadn’t been a mouse taking over, it would have been cockroaches.

The agilists are reacting to the business, technology, and social environment as we see it. If your climate matches the conditions under which Barry Boehm’s advice works, well, by all means, follow it. If it’s getting chilly, though, think about what else might work better.

BARRY BOEHM'S RESPONSE TO KENT BECK

Kent Beck is right in saying that agility and discipline are not opposites. He is also right in saying that XP requires a significant amount of discipline. He has also been right in recognizing that larger projects may need different types of discipline than those in XP. In *Extreme Programming Explained* (Addison-Wesley, 2000, p. 157), Kent says, "Size clearly matters. You probably couldn't run an XP project with a hundred programmers. Nor fifty. Nor twenty, probably. Ten is definitely doable."

As Jim Highsmith points out in *Agile Software Development Ecosystems* (Addison-Wesley, 2002,

p. 358), about 60 percent of the world's software projects have 10 or fewer people. But when project percentages are multiplied by project sizes to determine the relative amount of work being done, these small projects only account for about 17 percent of the world's software work.

What other disciplines do the people doing the other 83 percent of the work invoke? A good example is The ThoughtWorks lease management case study that Amr Elssamadisy and Gregory Schalliol describe in "Recognizing and Responding to 'Bad Smells' in Extreme Programming," (*Proc. ICSE 2002*, pp. 617-622). In order to get XP to work successfully on a 50-person project, they found it important to "... insist upon including the development and constant updating of a more traditional picture or graphic of the overall application ...," "create a precise list of tasks that *must* be completed ... and then police it rigorously and honestly," and use a factory pattern rather than simple design for foreseeable new features.

Does this sound like "control gained by enforcing obedience or order?" You bet.

I'm not sure where Kent got the data to substantiate his statement, "Efforts to force developers and customers to work according to a procedure developed by those not immediately responsible for results have uniformly failed." Requiring US Department of Defense contractors to achieve CMM Level 3 to compete on contracts has been such an effort, and it has produced quite a few success stories. For data, see the SEI Report CMU/SEI-2001-SR-014, "The 2001 High Maturity Workshop" (www.sei.cmu.edu/publications), and my forthcoming book with Richard Turner, *Balancing Agility and Discipline: A Guide for the Perplexed* (Addison-Wesley, 2003).

But what is most needed is not rhetorical "duking it out," but efforts to synthesize the best from agile and plan-driven methods to address our future challenges of simultaneously achieving high software dependability, agility, and scalability.

Which brings us back to the monkey and the elephant. ■

Kent Beck is the director of the Three Rivers Institute in southern Oregon. Contact him at kent@threeriversinstitute.org.

Barry Boehm is director of the University of Southern California Center for Software Engineering. Contact him at boehm@sunset.usc.edu.

Get the Scoop on
**Extreme
Programming**
in May-June 2003

**IEEE
Software**

Ever wonder what all the XP hype is about? Read these articles to get the inside story from developers who have "been there, done that" with XP:

- The XP Programmer:
The Few-Minutes Programmer**
- Introducing XP into Greenfield
Projects: Lessons Learned**
- Assessing XP at a European
Internet Company**
- Exploring XP for Scientific
Research**