Editor in Chief: Steve McConnell ■ Construx Software ■ software@construx.com

Cargo Cult Software Engineering

In the South Seas there is a cargo cult of people. During the war they saw airplanes with lots of good materials, and they want the same thing to happen now. So they've arranged to make things like runways, to put fires along the sides of the runways, to make a wooden hut for a man to sit in, with two wooden pieces on his head for headphones and bars of bamboo sticking out like antennas—he's the controller—and they wait for the airplanes to land. They're doing everything right. The form is perfect. It looks exactly the way it looked before. But it doesn't work. No airplanes land. So I call these things cargo cult science, because they follow all the apparent precepts and forms of scientific investigation, but they're missing something essential, because the planes don't land.—Richard Feynman, in Surely You're Joking, Mr. Feynman! *WW Norton & Company, New York, reprint ed., 1997*

I find it useful to draw a contrast between two different organizational development styles: *process-oriented* and *commitment-oriented* development. Process-oriented development achieves its effectiveness through skillful planning, carefully defined processes, efficient use of available time, and skillful application of software engineering best practices. This style of development succeeds because the organization that uses it is constantly improving. Even if its early attempts are ineffective, steady attention to process means each successive attempt will work better than the previous one.

Commitment-oriented development goes by several names, including *hero-oriented development* and *individual empowerment*. Commitment-oriented organizations are characterized by hiring the best possible people; asking them for total commitment to their projects; empowering them with nearly complete autonomy; motivating them to an extreme degree; and then seeing that they work 60, 80, or 100 hours a week until the project is finished. Commitment-oriented development derives its potency from its tremendous motivational ability; study after study has found that individual motivation is by far the largest single contributor to productivity. Developers make voluntary, personal commitments to the projects they work on, and they often go to extraordinary lengths to make their projects succeed.

Organizational Imposters

When used knowledgeably, either development style can produce high-quality software economically and quickly. However, both development styles have pathological look-alikes that don't work nearly as well and that can be difficult to distinguish from the genuine articles.

The process-imposter organization bases its practices on a slavish devotion to process

IEEE Software

for process's sake. These organizations look at process-oriented organizations, such as NASA's Software Engineering Laboratory and IBM's former Federal Systems Division, and observe that those organizations generate lots of documents and hold frequent meetings. The imposters conclude that if they generate an equivalent number of documents and hold a comparable number of meetings, they will be similarly successful. If they generate more documentation and hold more meetings, they will be even more successful! But they don't understand that the documentation and the meetings are not responsible for the success; these are the side effects of a few specific, effective processes. I call these organizations bureaucratic because they put the form of software processes above the substance. Their misuse of process is demotivating, which hurts productivity. And they're not very enjoyable to work for.

The commitment-imposter organization focuses primarily on motivating people to work long hours. These organizations look at successful companies such as Microsoft and observe that they generate very little documentation, offer stock options to employees, and then require mountains of overtime. They conclude that if they, too, minimize documentation, offer stock options, and require extensive overtime, they will be successful. The less documentation and the more overtime the better! But these organizations miss the fact that Microsoft and other successful commitment-oriented companies don't require overtime. They hire people who love to create software. They team these people with other people who love to create software just as much as they do. They provide lavish organizational support and rewards for creating software. And then they turn them loose. The natural outcome is that software developers and managers choose to work long hours voluntarily. Imposter organizations confuse the effect (long hours) with the cause (high motivation). I call the imposter organizations sweatshops because they emphasize working hard rather than working smart, and they tend to be chaotic and ineffective. They're not very enjoyable to work for, either.

Cargo Cult Organizations

At first glance, these two kinds of imposter organizations appear to be exact opposites. One is incredibly bureaucratic, and the other is incredibly chaotic. But one key similarity is actually more important than their superficial differences: Neither is very effective because neither understands what really makes its projects succeed or fail. They go through the motions of looking like effective organizations that are stylistically similar. But without any real understanding of why the practices work, they are essentially just sticking pieces of bamboo in their ears and hoping their projects will land safely. Many of their projects end up crashing, because these are just two different varieties of cargo cult software engineering, similar in their lack of understanding of what makes software projects work.

Cargo cult software engineering is easy to identify. Its engineer proponents justify their practices by saying, "We've always done it this way in the past," or "Our company standards require us to do it this way"—even when those ways make no sense. They refuse to acknowledge the trade-offs involved in either process-oriented or commitment-oriented development. Both have strengths and weaknesses. When presented with more effective, new practices, cargo cult software engineers prefer to stay in their wooden huts of familiar, comfortable, and not necessarily effective work habits. "Doing the same thing again and again and expecting different results is a sign of insanity," the old saying goes. It's also a sign of cargo cult software engineering.

The Real Debate

In this magazine and in many other publications, we spend our time debating whether process is good or individual empowerment (in other words, commitment-oriented development) might be better. This is a false dichotomy. Process is good, and

so is individual empowerment. The two can exist side by side. Process-oriented organizations can ask for an extreme commitment on specific projects. Commitment-oriented organizations can use software engineering practices skillfully.

The difference between these two approaches really comes down to differences of style and personality. I have worked on several projects of each style and have liked different things about each style. Some developers enjoy working methodically on an 8-to-5 schedule, which is more common in process-oriented companies. Other developers enjoy the focus and excitement that comes with making a 24×7 commitment to a project. Commitment-oriented projects are more exciting on average, but a process-oriented project can be just as exciting when it has a well-defined and inspiring mission. Process-oriented organizations seem to degenerate into their pathological look-alikes less often than commitment-oriented organizations do, but either style can work well if it is skillfully planned and executed.

The fact that both project styles have pathological look-alikes has muddied the debate. Some projects conducted in each style succeed, and some fail. That lets a process advocate point to process successes and commitment failures and claim that process is the key to success. It lets the commitment advocate do the same thing, in reverse.

The issue that has fallen by the wayside while we've been debating is so blatant that, like Edgar Allen Poe's *Purloined Letter*, we have overlooked it. We should not be debating process versus commitment; we should be debating competence versus incompetence. The real difference is not which style we choose, but what education, training, and understanding we bring to bear on the project. Rather than sticking with the old, misdirected debate, we should look for ways to raise the average level of developer and manager competence. That will improve our chances of success regardless of which development style we choose.

In the Next Issue

Requirements Engineering

Validating Requirements for Voice Communication

How Videoconferencing and Computer Support Affect Requirements Negotiation

A Reference Model for Requirements and Specifications

Also: **Automated Support for GQM Measurement**
Linguistic Rules for OO Analysis

In Future Issues

CMM across the Enterprise: Reports from a Range of CMM Level 5 Companies

Process Diversity: Other Ways to Build Software

Malicious IT: The Software vs. the People

Software Engineering in the Small

Applying Usability Techniques during Development

Recent Developments in Software Estimation

Global Software Development

IEEE Software