No Hypoxic Heroes, Please!
Biological Limits on Cowboy Programmers

Lewis Gray
Abelia Corporation

What do you say when someone rejects IEEE/IEA 12207, the Software Engineering Institute Capability Maturity Model℠ (CMM)®, and every other process standard? Here is a response.

James Bach's article "The Microdynamics of Process Evolution" [1] moved me to write a response to the entire collection of methodological arguments it exemplifies. Along with other like-minded arguments, Bach's article promotes heroism as a substitute for process. My goal here is to point out an inherent biological limit to dependence on heroes.

Mountaineers who climb high mountains, like Mount Everest, experience an insidious debilitation called *hypoxia* that impairs judgment—even the ability to detect the impairment. Software managers and developers experience a common problem that is similar in some ways to hypoxia. It limits what heroes can be expected to do.

When managers take away process standards as development tools, they take away the very tools that developers need to cope with the problem.

Hypoxia and Stress

I have been reading a lot about Mount Everest in the last year, including Jon Krakauer's *Into Thin Air*, a tragic story about the death of five people near the summit in 1996 [2]. Two of them were widely admired, professional mountaineers and guides. All of them were fit and well trained. Their exceptional drive and focus pushed them through miserable conditions all the way to the top of the highest mountain on earth. Their behavior during the climb is what most of us mean by the word heroic.

Krakauer reports that a major factor in the deaths of these five heroic people was hypoxia (lack of oxygen). High on Mount Everest, in the death zone above 25,000 feet, the amount of oxygen in the air drops to only one third of what it is at sea level. When the human body is deprived of oxygen to this extent, it always breaks down. Even Sherpas in Nepal, for example, live well below the altitudes of the camp sites on the way to the summit. Everyone who goes to the summit of Mount Everest becomes seriously hypoxic.

Hypoxic people not only do not think well, they also do not know that they do not think well. Anticipating this problem in 1996, the guides set strict rules for how and when their group members would make their summit attempts. In effect, the rules were intended to replace judgment at the most dangerous part of the climb near the summit. One of the rules was to turn around and head back down the mountain at 2 p.m. on summit day, no matter how close to the summit anyone might be at that time.

In the everyday world, there is a common medical condition—stress—that is similar in some ways to hypoxia. Heavy stress impairs our thinking and judgment. We find that we cannot identify and weigh alternatives like we can when we are calm. As with hypoxia, under heavy stress, we often simplify our options into black-or-white problems with an obvious solution. Then, we quickly seize the solution so we can get on to the next problem. It feels right, and we feel like efficient problem solvers.

But from past experience, we all know that this approach to decision-making is seductive and defective.

Lists Are an Antidote

There is a popular antidote for poor decision-making under stress: lists. All kinds of lists can help, from grocery lists to to-do lists. Project managers use checklists to estimate and control projects. Pilots use checklists to prepare for flight. Scuba divers use checklists before going into the water.

Everyone uses lists for the same basic reason. We all recognize that when we are preoccupied, under pressure, or distracted, we forget things and make errors in judgment. Lists are like the rules that the Mount Everest climbers imposed before their climb. We need them to simulate good judgment at certain critical times.

Many modern software engineering standards, like ISO/IEC 12207 (Information Technology – Software Life Cycle Processes), MIL-STD-498 (Software Development and Documentation), quality standards such as the ISO 9000 series, and process documents like the CMM, are just lists. Speaking as one of the designers of MIL-STD-498 and IEEE/EIA 12207, I can report that these standards were designed to be checklists of tasks to consider during software project planning.

We instinctively use lists for survival. The motivation for using standards is not just good form—not just being the best. We need standards and other lists to avoid disaster (although they may be useful for more than that).

Process Standards

In a development situation, you or I might choose not to do some task in a standard because it might not be appropriate for the project or organization. In many modern standards, the only mandatory activity is tailoring the standard to your particular needs.

Modern process standards are not designed to replace professional skill or experience in software development.

An earlier version of this article, "Gray Rebuts Bach: No Cowboy Programmers!" appeared in IEEE's Computer *(April 1998), pp. 102-103, 105.*

Pilots know how to fly before they are hired by airlines. They do not use checklists as do-it-yourself flying manuals. No one should expect standards like IEEE/EIA 12207 to be do-it-yourself software development manuals for novices. For the software professional, process standards are "pilot checklists" to get software development off the ground. The value of putting the tasks in a standard is it forces standard users to at least acknowledge, and better yet, to attempt to understand the possible negative consequences of not doing the tasks in the standard on their projects. This is the heart of the tailoring process. It is a critical part of successful project planning.

Modern process standards like IEEE/EIA 12207 are more useful today than their predecessor standards such as DOD-STD-2167A and DOD-STD-1679A. The earlier standards did not reflect current thinking that process standards can be standard checklists.

Why use a standard when you can develop your own personal checklist? One reason is that hundreds or thousands of software professionals have contributed tens of thousands of comments designed to polish a standard like MIL-STD-498. It does not seem sensible to many people to completely ignore these insights and start a list from scratch based only on personal, necessarily more-limited experiences. It is sensible to start your own list with a good standard.

Compliance with Standards

So what about a hard-hearted auditor who objects to any deletion of any requirement in a standard such as the ISO 9000 series or the CMM? Perhaps the auditor will not let you tailor the standard even though you feel that some requirements are inappropriate to your particular project.

Does not the audit refute the claim that modern process standards are designed to be checklists for use as memory aids by skilled software professionals? Does not the audit show that the standards are full of requirements that must be satisfied even when it does not make sense to do so, that they are really employed to substitute the standard writer's judgment for the judgment of real people on the project?

Actually, it does not. The audit is imposed (directly or indirectly) by buyers, who are customers. A company might voluntarily submit to an audit—an ISO 9000 audit, for example—to certify or register a quality system, but it would only do so with the expectation that the audit results would favorably impress potential customers. Foolish buyers or foolish auditors might insist that developers do foolish things, and they might be more of a nuisance wielding a standard than they would be without it. But buyers and auditors are not under the control of the standard.

The only rule for many modern process standards is to tailor them to suit your development conditions. Now, let us say that someone does this poorly—should we blame the authors of the standard?

Hypoxic Heroes Are Vulnerable

When people argue that all process standards hinder software development, as Bach does, they are promoting a "cowboy" approach that glorifies heroic individuals. According to his logic, you cannot be a hero using a process standard. There are no heroes without risk. In fact, the bigger the risk, the bigger the hero and the more the stress.

Without process standards to nag them at times of stress, when they need them the most, cowboy developers will push past their biological limits with no help in sight. It is like putting climbers into the death zone on Mount Everest with no rules for what to do on summit day.

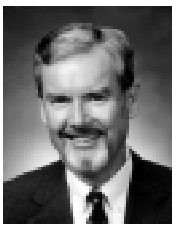
Stress will cut away their competence. They will not notice. And because they rely only on themselves, they will make bad decisions. Now, some of us resolve, instinctively, to follow a common rule, to check off our mental lists before we act. If I read Krakauer correctly, a big part of the reason that the climbers died on Mount Everest in May 1996 was that, tragically, in their impaired, hypoxic state, they broke their own rules.

The lesson I see for software development is that organizations and projects need process standards the most when their employees are most under pressure and have little time for thought. That is when everyone hits a biological limit and when it is most dangerous to let heroes run free without rules or guidelines.

Acknowledgments

Thanks to James Bach for the spirited, good-natured E-mail debate that helped me to identify the most important objections to my arguments and for his generosity in publishing my original rebuttal in his column. Thanks to Kirk L. Kroeker and the other editors at *Computer* for tightening and smoothing my original language. ◆

About the Author



Lewis Gray is president of Abelia Corporation. He is also a software process improvement coach and long-time teacher of software development standards. He has 17 years experience developing software systems for government, industry, and academia. He was a leader in the development of MIL-STD-498 and in the development of IEEE/EIA 12207. He is the author of many technical papers on process improvement and software engineering. He is the only instructor outside the Software Engineering Institute who is authorized to teach the TXM model of technology introduction. He holds a doctorate in the philosophy of science from Indiana University.

Abelia Corporation
12224 Grassy Hill Court
Fairfax, VA 22033-2819
Voice: 703-591-5247
Fax: 703-591-5005
E-mail: lewis@abelia.com
Internet: http://www.abelia.com

References

1. Bach, James, "Microdynamics of Process Evolution," *Computer*, February 1998, pp. 111-113.
2. Krakauer, Jon, *Into Thin Air*, Villard, New York, 1997.