References

- Davis, Alan M. 1995. *201 Principles of Software Development.* New York: McGraw-Hill.
- Glass, Robert L. 1979. *The Power of Peonage.* Computing Trends.

PEOPLE

Fact 1

The most important factor in software work is *not* the tools and techniques used by the programmers, but rather the quality of the programmers themselves.

Discussion

People matter in building software. That's the message of this particular fact. Tools matter. Techniques also matter. Process, yet again, matters. But head and shoulders above all those other things that matter are people.

This message is as old as the software field itself. It has emerged from, and appears in, so many software research studies and position papers over the years that, by now, it should be one of the most important software "eternal truths." Yet we in the software field keep forgetting it. We advocate process as the be-all and end-all of software development. We promote tools as breakthroughs in our ability to create software. We aggregate a miscellaneous collection of techniques, call that aggregate a methodology, and insist that thousands of programmers read about it, take classes in it, have their noses rubbed in it through drill and practice, and then employ it on high-profile projects. All in the name of tools/techniques/process over people.

We even revert, from time to time, to anti-people approaches. We treat people like interchangeable cogs on an assembly line. We claim that people work better when too-tight schedules and too-binding constraints are imposed on them. We deny our programmers even the most fundamental elements of trust and then expect them to trust us in telling them what to do.

In this regard, it is interesting to look at the Software Engineering Institute (SEI) and its software process, the Capability Maturity Model. The CMM assumes that good process is the way to good software. It lays out a plethora of key process

areas and a set of stair steps through which software organizations are urged to progress, all based on that fundamental assumption. What makes the CMM particularly interesting is that after a few years of its existence and after it had been semi-institutionalized by the U.S. Department of Defense as a way of improving software organizations and after others had copied the DoD's approaches, only then did the SEI begin to examine people and their importance in building software. There is now an SEI People Capability Maturity Model. But it is far less well known and far less well utilized than the process CMM. Once again, in the minds of many software engineering professionals, process is more important than people, sometimes spectacularly more important. It seems as if we will never learn.

Controversy

The controversy regarding the importance of people is subtle. Everyone pays lip service to the notion that people are important. Nearly everyone agrees, at a superficial level, that people trump tools, techniques, and process. And yet we keep behaving as if it were not true. Perhaps it's because people are a harder problem to address than tools, techniques, and process. Perhaps it's like one of those "little moron" jokes. (In one sixty-year-old joke in that series, a little moron seems to be looking for something under a lamp post. When asked what he is doing, he replies "I lost my keys." "Where did you lose them?" he is asked. "Over there," says the little moron, pointing off to the side. "Then why are you looking under the lamp post?" "Because," says the little moron, "the light is better here.")

We in the software field, all of us technologists at heart, would prefer to invent new technologies to make our jobs easier. Even if we know, deep down inside, that the people issue is a more important one to work.

Sources

The most prominent expression of the importance of people comes from the front cover illustration of Barry Boehm's classic book *Software Engineering Economics* (1981). There, he lays out a bar chart of the factors that contribute to doing a good job of software work. And, lo and behold, the longest bar on the chart represents the quality of the people doing the work. People, the chart tells us, are far more important than whatever tools, techniques, languages, and—yes—processes those people are using.

Perhaps the most important expression of this point is the also-classic book *Peopleware* (DeMarco and Lister 1999). As you might guess from the title, the entire book is about the importance of people in the software field. It says things

like "The major problems of our work are not so much technological as sociological in nature" and goes so far as to say that looking at technology first is a "High-Tech Illusion." You can't read *Peopleware* without coming away with the belief that people matter a whole lot more than any other factor in the software field.

The most succinct expression of the importance of people is in Davis (1995), where the author states simply, "People are the key to success." The most recent expressions of the importance of people come from the Agile Development movement, where people say things like "Peel back the facade of rigorous methodology projects and ask why the project was successful, and the answer [is] people" (Highsmith 2002). And the earliest expressions of the importance of people come from authors like Bucher (1975), who said, "The prime factor in affecting the reliability of software is in the selection, motivation, and management of the personnel who design and maintain it," and Rubey (1978), who said, "When all is said and done, the ultimate factor in software productivity is the capability of the individual software practitioner."

But perhaps my favorite place where people were identified as the most important factor in software work was an obscure article on a vitally important issue. The issue was, "If your life depended on a particular piece of software, what would you want to know about it?" Bollinger responded, "More than anything else, I would want to know that the person who wrote the software was both highly intelligent, and possessed by an extremely rigorous, almost fanatical desire to make their program work the way it should. Everything else to me is secondary. . . ." (2001).

References

- Boehm, Barry. 1981. *Software Engineering Economics.* Englewood Cliffs, NJ: Prentice-Hall.
- Bollinger, Terry. 2001. "On Inspection vs. Testing." *Software Practitioner,* Sept.
- Bucher, D. E. W. 1975. "Maintenance of the Computer Sciences Teleprocessing System." Proceedings of the International Conference on Reliable Software, Seattle, WA, April.
- Davis, Alan M. 1995. *201 Principles of Software Development.* New York: McGraw-Hill.
- DeMarco, Tom, and Timothy Lister. 1999. *Peopleware.* 2d ed. New York: Dorset House.