Alex Groce (agroce@gmail.com), Oregon State University

Vernor Vinge's *A Deepness in the Sky* is a science-fiction novel. It won the Hugo Award for the best science fiction or fantasy novel of 2000, as well as the Prometheus Award and John W. Campbell Memorial Award for that year. It was nominated for the Nebula Award (similar in prestige to the Hugo, but voted on by science fiction writers only, rather than readers), the Arthur C. Clarke Award, and the Locus Award. It's an exciting tale of adventure, deception, temptation, space travel and warfare, first contact with an alien race, and the birth, decline, and fall of civilizations.

It may not be the *only* science-fiction novel that has *software engineering* as a major topic, but it is certainly the most prominent. Computer science, and particularly artificial intelligence, of course, is critical to many science fiction novels, ranging from the earliest robot stories to the "cyberpunk" visions of the 1980s. For the most part, however, software in science fiction is an uncreated magical MacGuffin, and there are few instances of anything even resembling actual software development in the genre. One notable exception is Rudy Rucker's *The Hacker and the Ants*, which surprisingly involves a hacker who actually writes code in realistic programming languages, complete with discussion of compilers. A second possible exception is much older, and the topic of a column yet to be written. The details in Rucker's book presumably come from Rucker's background as a math and computer science professor. Like Rucker, Vinge has a background that helps with software realism: he's a (now retired) math and computer science professor as well. Vinge's book, however, goes beyond simply having a realistic view of software engineering as it is practiced today. Vinge's plot (and even theme) relies on the hypothesis that Brooks' "No Silver Bullet" essay is correct, with a vengeance.

Early in the novel, one of the protagonists in the far future setting begins to learn the honored and valuable skills of a "Programmer-Archeologist." The job of a programmer-archeologist, of course, is to understand the endless complexities and layers of ancient code that make up the software ecosystem of Vinge's future. It's a dangerous system:

> "So behind all the top-level interfaces was layer under layer of support. Some of that software had been designed for wildly different situations. Every so often, the inconsistencies caused fatal accidents. Despite the romance of spaceflight, the most common accidents were simply caused by ancient, misused programs finally getting their revenge."

One definitely gets the feeling that there are no loop invariant annotations, formally proven pre-conditions and post-conditions, or even very good documentation in Vinge's universe. Our viewpoint character proposes the obvious: "We should rewrite it all." But his mentors inform him that he and a thousand as good as him would take a century to rewrite the code, and it would just be a different mess, if they did. So, the characters adapt code they've found for one cryogenic crisis five hundred years earlier, that's "almost precisely" what they need, “with minor revisions.” In the end, secrets, trapdoors, and undocumented features of an ancient sensor network system are the device by which a significant portion of the plot moves. This is a universe where model checking and static analysis have striven in vain.

In one sense, Vinge’s future vision of software engineering is just a classic science fictional “If this goes on…” extrapolation. Everyone who has even been lost in a vast maze of directories, build systems, conflicting configurations, and abandoned branches in the source repository of a major software system can imagine scaling that feeling up and finding a world much like the software labyrinth of Vinge’s world. Those of us who have seen Heartbleed and “goto fail” can all too easily imagine a hidden empire of trapdoors and exploits in legacy code that emerges not from years or decades of development, but from *centuries* of work.

There’s more than simple extrapolation going on, however. The impossibility of artificial intelligence (which seems to be Vinge’s proposal for the only plausible “silver bullet”) isn’t just a background assumption of the novel. It’s a sorrow and a threat to human potential, and the limitations the failures of software engineering have introduced tempt the above-mentioned Programmer-Archaeologist to consider monstrous means of fixing the problem. While no software engineer will probably learn any new techniques for building systems from this book, it does inspire serious thought about the implications of the difficulties of building software. It’s no surprise that *A Deepness in the Sky* has been discussed in numerous blogs on programming and software development, including some instances of computer scientists arguing that Vinge’s setting is unrealistic because something we already have is the real silver bullet. It never hurts to have good fiction that can also inspire speculation about the larger implications of our field.

Perhaps more importantly, *A Deepness in the Sky* is a good science fiction novel that is likely to bring some talented young readers to computer science (and software engineering specifically) just as the rockets and robots novels of the Golden Age of SF helped produce a generation of engineers and scientists. If science fiction is, as some have argued, “the dreams our stuff is made of” and a major inspiration for modern technical innovation, it can only be a good thing to have a few dreams that are about software engineering. The future of software engineering is likely to be more extraordinary if a few kids who would have dreamed of being starship pilots instead dream of being Programmer-Archaeologists.