The Prophet Alexander

John Favaro

Computer Programming, Italian Edition, March 2004

"A prophet is not without honor except in his own country, among his own relatives, and in his own house." (Mark 6:4)

"Beware of false prophets" (Jeremiah 23:9-40)

A brother of mine is an architect (www.johnsonfavaro.com). One day early in 2003, while contemplating a few questions in software development, I decided to ask him a simple question.

"What can you tell me about Christopher Alexander?"

Alexander, of course, is now such a legend in the software engineering community that he hardly needs an introduction. But for the few who are still unaware of him, he is an emeritus professor of architecture (University of California at Berkeley). His dissertation at Harvard in 1964, *Notes on the Synthesis of Form*, received the first Gold Medal for research ever given by the American Institute of Architects and established him as an important voice in modern architecture. Then he wrote a book in 1977 called *A Pattern Language*, which contained 253 solutions to problems in architectural design, ranging from the design of a kitchen to the design of entire cities. The book was a great success and created an entire movement. His influence in those years was enormous in the architectural community.

But the big surprise was that the book and its ideas then became an enormous success in our own software engineering community. In the late 1980s a number of software engineers, including Kent Beck, the father of Extreme Programming, began to read Alexander and see possibilities to apply his ideas to software. This interest culminated, as we all know, in the publication of the book *Design Patterns* in 1995 by four authors who became known as the Gang of Four. The book justifiably became required reading for any practitioner in object-oriented development, and it cemented Alexander's reputation in the software community. His status grew to legendary proportions. His work was cited by the great software engineering experts such as Tom deMarco; Grady Booch, one of the fathers of UML, spoke of having "Alexandrian epiphanies" in the mountains of Colorado.

And so it was that on that day, contemplating this reverential status of Alexander among software engineers, I asked myself: "I wonder what the architects think of Alexander?"

My brother's response to my question was a shock: "Oh, he was a kind of guru twenty or thirty years ago, but nothing much ever came of it."

I was astonished! How is it possible that a man who whose status is nearly god-like in the software community is viewed in this way in his own community? But further investigation confirmed that it was indeed the case: “Techies avidly discuss Alexander’s oeuvre on the Web. Yet Alexander’s own colleagues in the American architectural establishment will have nothing to do with him,” wrote Wendy Kohn [1] in the *Wilson Quarterly*. (Alexander is not the first architect to be treated this way in his own community. The book *Frank Lloyd Wright: Prophet Without Honor*, written in 1975, notes that Wright was never officially recognized in his home state of Wisconsin.)

Through discussions with my brother and further reading I was gradually able to piece together the reasons for the current antipathy toward Alexander in the architectural profession. It seems that there are three principal “sins” for which he is most reproached. The first of these is that he tried to take the art out of architecture. “He wanted to catalogue all the patterns of architecture in the history of the world in the belief that there would be a scientific way to get to good design,” explained my brother. “Of course it didn't work because architecture is an art not a science.” And he added a warning to us: “I have a feeling that computer applications are also as much an art as a science, since they're about tailoring to specific needs, making choices and judgments, relating choices to a set of values on behalf of the client, etc.”

The second sin consisted of not producing many concrete results from his theories – which, in the case of an architect, means buildings. “The buildings he actually produced (not very many, small residences) ended up being ultra-conservative (very Eurocentric, in fact, like little Austrian cottages) and not very remarkable,” commented my brother. The architect Moshe Safdie reiterated this view [2]: “People go from the ideas, which they could interpret freely, to the solutions, which they see as somewhat anachronistic, somewhat craft-related, stylistically somewhat Victorian. They see him as a reactionary.” Kohl adds: “He has a vision. But if we cannot experience his vision in *built* form, Alexander is bound to realize his greatest fear – that his ideas will remain pristine, whole, even beautiful, and on the printed page only.” Furthermore, Alexander seems to have trouble admitting that beautiful buildings can be and have been built *without* using his ideas. “The simultaneously intimate and all-knowing tone of his writing sounds unbearably condescending to practitioners who take pride in having invented some of their own solutions to the problems of architecture,” writes Kohn.

This last remark brings us to the third “sin.” There is something messianic about the man and his message that inspires the formation of cults. A glance at his website (www.patternlanguage.com) reveals mystical, New Age phrases like “spirit growing in the world” and “making wholeness heals the maker.” Kohn writes, “It comes as no surprise that Alexander is not tolerant of others’ ideas,” and relates the story of a doubting student who was told by Alexander to discover his own psychological problems that were preventing him from seeing the truth.

So why *is* Alexander so revered in the software profession when he is so controversial in the architectural profession? Three possibilities come to mind:

- Everybody in the architectural profession is stupid and doesn’t see how right Alexander’s ideas are;
- Everybody in the software profession is stupid and doesn’t see how wrong Alexander’s ideas are;
- There is some kind of essential difference between software and architecture at work here.

I suspect that the answer has something to do with the third possibility. Software is essentially an *engineering* discipline, with – despite frequent characterizations to the contrary – a far smaller artistic component than is found in architecture. It is this artistic component that is responsible for so much of the controversy.

Nonetheless, some lessons may be drawn for the software community out of this controversy. The first is that, despite the uncontested value of concepts such as design patterns – and they should be part of every serious programmer's repertoire – programming will never be reduced to a mechanical activity. There will always be room for programmers to aspire to the highest levels of human inspiration, creativity, and innovation in their work. The critic John Ruskin once said, "We require from buildings, as from men, two kinds of goodness: first, the doing their practical duty well; then that they be graceful and pleasing in doing it; which last is itself another form of duty." It wouldn't hurt to think about software in the same way.

The second lesson is that, where a constructed building is the ultimate demonstration of an architect's theories, a running program is the ultimate demonstration of a theory in software engineering. It is fun to read and write about exotic and abstract approaches to software engineering ("applying the theory of chaos to software development") but in the end all that counts is the running program. The recent agile approaches deserve credit for bringing the focus back to this fundamental fact.

But perhaps the most important lesson is the last: even more than the architectural community, our own community is vulnerable to fads, trends, and the formation of cults. Time and again we have succumbed to the latest fashion – structured programming, artificial intelligence, object-orientation, UML – with religious fervor, creating movements whose members, certain of their unique grasp of the Truth, will tolerate no dissent. We are in danger of seeing this happen today in the agile movement. At a panel during the *Fourth International Conference on Extreme Programming and Agile Methods* in Genoa in May 2003, M.I.T. professor Michael Cusumano compared the agile movement to a religious movement and remarked, "I think we're seeing the beginnings of a priesthood here."

The XP community is well aware of the cult issue and the problem it represents. In November 2003 there was a vigorous debate on the Yahoo XP discussion list on the topic of "XP and Religion," (including a rather interesting diversion on the relationship between XP and the Buddhist notion of "the middle way"). A central point in the debate was how widespread the impression is that agile practitioners are intolerant of dissenting opinions, effectively having formed a cult. They do well to be aware of this problem and try to resolve it, for it risks repelling people who would otherwise be attracted to its undeniably fine features.

Architecture is as old as civilization and there is no place in it for intolerant cults. Software engineering is young, with so much to be learned and experimented – and even less room for such counter-productive, and ultimately oppressive, phenomena.

Resources

[1] Wendy Kohn, "The Lost Prophet of Architecture," *Wilson Quarterly*, Summer 2002.

[2] Emily Eakin, "A Bid to Rebuild the World on a Human Scale," *International Herald Tribune*, 16 July 2003.