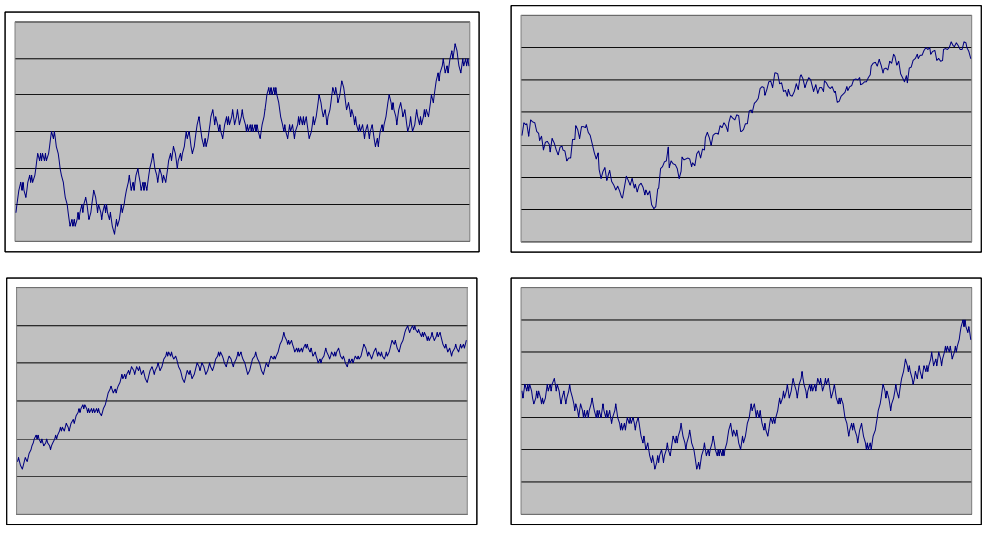
The Piranha Pond

John Favaro

Computer Programming, Italian Edition, CP134, May 2004

In a recent column [1] an interesting economic analogy (the theory of options) was found to be very useful for explaining an important aspect of agile software development projects: the value of flexibility. In this column we will exploit a different economic analogy to explore another aspect of agile software development projects: the manner in which information is absorbed and transformed into implementation [2]. The dynamics of agile development projects can be difficult to explain because they lead to principles that seem strange and counter-intuitive. One such principle is known informally as YAGNI ("You Aren't Going to Need It"). The YAGNI principle states essentially that a project should not plan for the future; instead, it should concentrate only on the present. Such a principle seems to run so contrary to common notions of good software project management and planning that it is not surprising that many people acquire a view of agile software development as poorly disciplined at best, and chaotic at worst.

Yet once the dynamics of agile development are understood, such a principle makes a good deal of sense. Economists went through a similar process of discovery over the last century as the dynamics of markets were gradually understood, and their experience can provide some useful insights. It was just over one hundred years ago that Louis Bachelier made an amazing claim: the movements of stock markets are random. *Random?* Impossible, everybody cried: markets aren't random, they obviously respond to important events like profit announcements and mergers! But there was a problem: it really *was* impossible to distinguish real market price movements from random movements. You don't believe it? Then try yourself: only one of the stock charts in Figure 1 is real – all the others were generated randomly. I'll bet that you can't tell which is the real one.



Only one of these charts is real

John Favaro
Via Gamerra, 21
56123 Pisa - Italy

Tel +39 050 55 60 74
Fax +39 050 6143 1145
john@favaro.net

www.favaro.net/john/

But even if it's true, the question remains: *why* is it true? The answer arrived in 1964 with the Efficient Market Hypothesis formulated by Fama. According to the hypothesis, the participants in the market are always hungry for new information. As soon as new information arrives, it is quickly disseminated, analyzed, and digested. Like a pool of piranhas, investors pounce on it and strip it to the bones. What happens to this information? It is immediately converted into market prices, so that the state of the market always completely reflects the information available. But think about what that implies: the future has been completely separated from the past. By absorbing all past information in an efficient and complete manner, the market has nothing more to say about the future. And so like a drunken sailor, the market stumbles from one new piece of new information to another, completely absorbing it and waiting for the next, unpredictable arrival. This *random walk* (its official name) is the phenomenon that characterizes efficient markets.

The dynamics of agile software development projects are surprisingly similar to what I have just described. Agile projects strive to be *efficient* projects, disseminating, absorbing, and digesting new information as quickly as possible. To see this, just take a look at the core practices: collective ownership, daily meetings, pair programming, the absence of fixed roles that tend to compartmentalize information. Agile projects don't wait to implement new information, either. In contrast to BDUF (*Big Design Up Front*), implementation is undertaken as soon as possible and all implications immediately considered.

Do you see what happens? Like efficient markets, efficient projects separate the past from the future, following their own form of random walk. On the one hand, the system implements *everything* that is currently known. Core practices such as test-first programming, refactoring, and the daily build ensure that the system is always up to date, ready to react to any new information. On the other hand, the system carries no *extra* functionality, implemented only out of fear of what "might" happen. Agile developers realize that such "hooks" only make the system more complex and difficult to understand. Above all, they waste resources on something that *might not even be needed* (YAGNI principle) – resources that could be used to implement new requirements when they are *really* known. This is all part of a new approach to requirements management already discussed in a previous column [3].

Like efficient markets, efficient projects demonstrate an important principle: the best way to prepare for the future is to take good care of the past.

P.S. the upper-right hand chart tracks the S&P500 stock index from November 2002 to November 2003. I generated the others by flipping a coin 365 times.

Resources

[1] J. Favaro, "The Tomato Garden," *Computer Programming*, Italian Edition, CP133, April 2004.

[2] J. Favaro, "Efficient Markets, Efficient Projects, and Predicting the Future," *Proceedings 5th International Conference on Agile Methods*, Garmisch, June 2004.

[3] J. Favaro, "A Quiet Revolution in Requirements Management," *Computer Programming*, Italian Edition, CP124, May 2003.