An Agile Cure for All Ills?

The Voice of Evidence article “What Do We Know about Agile Software Development?” (by Tore Dybå and Torgeir Dingsøyr, Sept./Oct. 2009, pp. 6–9) was enlightening. Now we need articles on what we know about all of software development, and what we don’t know about software development.

Implicit in the article is the fact that one size fits nobody, at least not the agile style. The table does indicate that agile is good for learning through experimentation.

Experimentation isn’t development; at best, it’s research. Research precedes development and should not be mistaken for it. Whatever solution might actually result from agile has zero chance of being the proper one that the entire system needs. Having been in this game for some 45+ years, I’ve seen, and suffered from, way too many management fads and silver bullets du jour—none of which did anything except enrich a few fast-talking consultants. Until software folks realize that they’re merely one part of developing a system, we’ll continue to have the same problems and continue to suffer from fads that solve nothing in spite of the hype.

My son won a team programming contest in high school. The teams had four hours to code solutions to four problems. He got a perfect score from the judges after coding all of them himself. For problems that small, you don’t really need to do anything except conceive of a solution and write it down. The teams with multiple members ran into communications and decision-making problems that slowed them down.

Analogously, small projects will suffer from a “full-boat” CMMI style of instantiation. For larger problems, agile approaches are clearly useless (except as the front end to do research to guide decisions before proceeding). We can never coordinate all the pieces of a humongous problem with an agile approach. It would be refreshing if all the purveyors of universal solutions finally admitted that their approach is good only for selected subsets of problems.

In an article long ago in a commercial software magazine, someone actually went through several “methodologies” and showed where they worked and where they did not, and said why. That inconvenient truth did not change anybody’s mind. Management that had been to a sales pitch seminar remained convinced that their chosen solution was the answer to everything and continued to force it on the programmers. And likewise, consultants and sellers of software oil continue to push their “remedy” as good for curing all ailments. Software folks must realize that they’re just a part of an overall systems development process and stop trying to control things that are beyond their scope and ability. Until all software projects cooperate with the development of the entire system, our resulting systems will continue to have problems.

IEEE Software needs to concentrate on systems architecture and systems development in order to determine what approaches to the software portion of systems might be appropriate.

William Adams
Independent consultant williamadams@ieee.org

Tore Dybå and Torgeir Dingsøyr respond: The response to our Voice of Evidence column on what we know from studies of agile software development includes a number of claims about the suitability of agile methods. Some of these claims require an answer.

First of all, we agree that we need more knowledge of software development in general. Empirical studies of actual development practices will provide better knowledge of what works for whom, when, and how. This applies to traditional as well as to agile development.

Further, we agree that one size doesn’t fit all. Tailoring methods to a project’s needs is likely to improve its outcome. However, there’s little empirical research on which we can base guidelines for tailoring. Also, there are few secondary studies such as ours on agile development that seek to synthesize existing knowledge.

Adams distinguishes between experimentation and development. However, the integration of these has been found to be a defining characteristic in new product development, which shares many characteristics with software development.

Finally, Adams claims that agile development is “clearly useless” for larger problems. There’s no evidence supporting this claim—no empirical research of agile methods for large projects exists. Yet, there are examples of large projects using agile methods.

In response to Hakan Erdogmus’s From the Editor column “Architecture Meets Agility” (Sept./Oct. 2009, pp. 2–4) … I guess it all depends.

The moment you choose a platform, you’ve selected a (high-level) architecture. Most projects don’t have this level of luxury. Later, the choice of an implementation language will tend toward a small set of architectures; the use of certain tools will define certain architectures. These are all levels of architectural choice.

Within each of these, there are usually further project-specific choices to be made. I think the agile discussion really focuses on these levels, not on what I’d call macro-level architecture.

Of course, this more detailed level is not usually hard to achieve, with one key proviso. Attention must be paid to real decoupling and proper encapsulation techniques to allow the flexibility needed for reshaping an architecture in an agile manner. Without this level of expertise and professionalism, we’ll probably see the evolution of the popular Rats’ Nest architecture. I assume that even now there is a cloud-based version of this in development.

George Brooke
Oak Lodge Consulting george@oaklodgeconsulting.co.uk