



## SOFTWARE DEVELOPMENT METHODOLOGY DEBATE

In the conclusion to his contribution to the October The Profession column (“Do Agile Methods Marginalize Problem Solvers?” pp. 120, 118-119), Victor Skowronski correctly states that there are situations in which agile methods are more or less effective. I think, however, that the choice between agile and nonagile approaches is not so simply reduced to whether or not there are known solutions.

As a profession, when we approach the complex process of software engineering, we often get caught up in what I refer to as “either-or-thinking” ([www.allpm.com](http://www.allpm.com)). To be effective, agile methods must be used with a degree of creativity, not as an off-the-shelf methodology applied by the book.

The effective development environment should combine elements that enable different activities throughout the process. Quiet space and the ability to communicate regarding the need for more time to incubate a well thought-out solution are necessary as is the creation of minimal, though not zero, documentation.

Most software engineering does not require the level of creativity and genius of an Isaac Newton or Thomas Aquinas. In fact, if they were on my team, both of these creative geniuses would be required to take some people skills training to make them more effective team members. While there may be a place for a genius who can only operate in isolation, a collaborative problem-solving approach is more appropriate in most circumstances.

Software engineering combines social and technical components. In this time of divisiveness between proponents of one concept or another, political or otherwise, let’s remember that the most effective solutions in this type of endeavor generally involve the



fusion of multiple views and the creative application of the best elements of a number of “isms.”

The question is: How can we use agile as well as other more formal and individualistic methods to promote the most effective software engineering approach for each situation?

*George Pitagorsky  
New York, N.Y.  
pitagorsky@iil.com*

Regarding Victor Skowronski’s article on agile methods, I am surprised that *Computer* would publish an article that is not only misleading, but is also based on erroneous assumptions and premises.

Proponents of agile methods for software engineering would not claim that this approach is applicable for solving research problems. These methods are meant to facilitate engineering software systems.

It appears that the author ignored the most scientific aspects of agile programming, namely the user stories, test-first, regression-based, and incremental development approaches. He concentrated instead on weaker aspects of agile methods, such as pair programming and the open-door policy, which are organizational factors, and which, of course, were proposed for software development, not for scientific discoveries.

Agile methods are not intended to be a panacea for all human discoveries and endeavors. Endeavoring to prove that they won’t work for everyone and everything is trying to prove the obvious.

*Sandeep Shukla  
Blacksburg, Va.  
shukla@vt.edu*

I like to watch the software development methodology wars with an attitude similar to that with which I watch the Sunday morning political shows: Amid all the shouting, there’s probably some small bit of useful information to be had.

Although Victor Skowronski didn’t sound as hardline as some, I felt he was too summarily dismissive of the useful bits from the agile/extreme camp. And using Newton and Aquinas as excuses is very funny, but I’m not sure whether that sort of talent would be blessing or burden on a development team.

Would Skowronski really prefer the product of an isolated, taciturn genius to that of someone, perhaps less talented, who could effectively engage the users? Maybe a good team would accommodate both types.

I would guess that the vast majority of software written outside of universities deals with problems that were solved long ago, possibly in other contexts. Of course, we all enjoy reinventing the wheel in our own idiosyncratic ways, but a little experience can take us a long way. Having a hammer as your only tool makes everything nail-like.

I regard the agile “basic principles” as prioritizations rather than exclusions. For any project more complicated than listing the primes less than 100, you can be pretty sure that, however detailed your planning, there’s going to be something you’ve omitted, gotten wrong, or just plain don’t know at the outset.

It makes sense to determine what needs changing as soon as possible, rather than finding out at the end of a three-month development cycle that you need to completely replace those 10 functions you spent a week documenting. And how would you determine this other than by actually communicating with the customers/clients/stakeholders?

I’ll grant that certain domains in which there’s a possibility of physically damaging someone—like aircraft control or CAT scan software—require special considerations and more for-

We welcome your letters. Send them to  
[computer@computer.org](mailto:computer@computer.org).

mal development approaches. But even there, wouldn't it be better to have something to test earlier rather than later?

Garland Stern  
gstern@spitzinc.com

Having read the well-written and thoughtfully reasoned "Do Agile Methods Marginalize Problem Solvers?," I wonder whether the author has found that on large-scale, highly developmental programs, agile methods are likely to leave some of the most challenging problems in their wake, thus causing project management to eventually seek relaxations or waivers of the related specifications, with increasing schedule slippages in the meantime?

The article suggests that if used on defense contracts, agile methods could exacerbate a dangerous tendency: If project management has signed up for an overly optimistic schedule—and has done so at a cost it cannot feasibly meet—it will fail to staff software development adequately. Then, when challenges remain or problems emerge, management will add software programmers. Unfortunately, unlike adding more hardware engineers when hardware schedules slip, adding software programmers late to an ongoing project often causes the effort to become more disorganized and to slow down.

Software development in large, complex defense procurements tends to be a chronic problem, resistant to each new wave of reforms that defense officials and project managers introduce. One root cause is the tendency of the government customer to overlook a recurrent phenomenon that could be dubbed the "short-sighted surveillance syndrome": Large software projects that go bad, tend to go bad early, much earlier than the customer anticipates, usually in the first six to nine months. Agile methods would seem to limit a contractor's ability to discover problems early in a project, which is ultimately counterproductive for both contractor and customer.

If agile methods continue to propagate, contractors and customer will need to make critical assessments of software development trends very early in the project—and also ensure that test procedures are rigorous and not allowed to become superficial in order to keep problems from surfacing prior to acceptance.

Roland Trope  
New York, N.Y.  
roland.trope@verizon.net

*The author responds:*

I agree with George Pitagorsky that accommodating the different working styles of individuals and the various types of problems they are trying to solve requires different methodologies. The point of my article was that agile methods might not be suitable for individuals with certain working styles and that these individuals may be the ones most needed for particular projects.

Pitagorsky appears to contradict himself in his second paragraph, however, when he states that he would assign Newton and Aquinas to personal skills training. Apparently when there is a mismatch between the people and the method, Pitagorsky would rather change the people than the method. Both Newton and Aquinas were extremely effective in their work, and it is unlikely that better people skills would have made them any better. If anything is broken and needs fixing in this example, it is the method.

Engineering, including software engineering, is the application of scientific principles to the solution of problems. Contrary to Sandeep Shukla's assertion, it is just as rigorous an intellectual challenge as any scientific investigation. As I noted in my article, some software projects require minimal problem solving. If they were the only projects that used agile methods, however, agile methods would be a specialized and minor form of software development.

The technical aspects of agile methods that Sandeep Shukla mentions are not unique to the agile methodology. Neither are they central to agile meth-

ods, as can be seen from the principles of agile methods that I quoted. What is unique and central to agile methods is the social organization. As such, the social organization affects all aspects of the software development process. If some of their effects are negative, that needs to be considered when choosing a development methodology.

When he states that the difference between agile and more formal methods is getting a working program sooner rather than later, Garland Stern assumes that the agile team will have the level of problem-solving skills needed for the task. This may not be the case. By requiring a high level of interpersonal skills, agile methodology reduces the number of qualified programmers relative to more formal methods. With fewer applicants, an agile team may have to accept a lower level of problem-solving skills in its members to staff itself adequately.

The problem of finding agile programmers with adequate problem-solving skills may actually be worse than just a reduced applicant pool. Time spent learning people skills is time not spent learning problem-solving skills—see George Pitagorsky's statement about giving Newton and Aquinas training in people skills. With less time spent learning problem solving, the problem-solving ability of agile programmers may be lower than for programmers in general.

The problem that Roland Trope suggests of developers putting off work on tough problems until too late is a real one. The emphasis on working software favors this tactic. Agile proponents may also argue that delaying work on tough problems may be justified if their solution ends up not being needed. I wonder, however, if this is a characteristic of agile methods or a characteristic of the people who are attracted to agile methods. If it is the latter, formal methods would not prevent the kinds of procrastination that lead to slipped deliveries.

Another problem that is more specific to defense contracts involves ver-

ification and validation. Mission-critical software must be subjected to a rigorous program of testing and inspection. Agile methods do not provide the documentation to support this. To quote Deming, “You cannot inspect quality into a product.” In software terms, this means that you cannot rely solely on after-the-fact testing of the code to validate it. You need a process that verifies the software at each step.



### CURBING THE SPAM PROBLEM

In “Spam and the Social-Technical Gap” (Brian Whitworth and Elizabeth Whitworth, Oct., pp.38-45), the authors discuss charging for returned e-mail as an option for decreasing spam.

In an ideal world, where the senders of spam legitimately owned the e-mail addresses they used, charge-backs might be a reasonable option to consider. However, widespread reports indicate that much, if not nearly all, spam is sent through rogue methods. These tactics range from spoofing return e-mail addresses—either using fake addresses or other peoples’ valid addresses—to leveraging the bad deeds of virus writers to hijack people’s computers and use them as e-mail relays.

Given the reality of today’s world, charging back for returned e-mail would not penalize the spammers, but it would very likely cause additional, grievous harm to innocent victims.

Truly curbing the problem with spam will require changes to the e-mail infrastructure on the Internet. Since basic e-mail protocol largely assumes the sender is honest, it provides little authentication—and it is the resulting

anonymity of senders that has allowed spam to flourish.

*Rob Stitt*  
Lees Summit, Mo.  
[robstitt@computer.org](mailto:robstitt@computer.org)

The authors of the thought-provoking “Spam and the Social-Technical Gap” claim that “without responders, spam would not exist.” However, responders aren’t necessary. Some spammers are simply sociopathic: They try to “hit” as many recipients as possible. They don’t care if anyone responds.

Even ignoring spam from pure sociopaths, there are business models for spam that do not require any responses. Consider this scenario: You’re a Nigerian who has just obtained your first e-mail account. A fellow Nigerian approaches you (in person or by e-mail) and offers a deal: For \$20, he’ll send a phony funds-transfer solicitation letter on your behalf to millions of people worldwide. You’re enticed by the potential gain, so you pay him the money—two months’ wages—and the spam is sent.

No one responds. You know you’ve been hoodwinked, but the swindler has disappeared, and the police don’t care. You’ll never fall for that trick again, but it doesn’t matter because another thousand people like you get e-mail accounts every day. No responses to the spam are required for the scam to succeed and proliferate.

In foretelling a spam-filled future, the authors wrote, “The Internet will then transmit vast amounts of information, but minute amounts of meaning.” Had they used the term “information” correctly, the sentence would read, “The Internet will then transmit vast amounts of data, but minute amounts of information.”

Data is the bits and bytes that computers store and transmit. Information is that aspect of data that conveys meaning and allows people to make decisions.

Thanks to the authors for their article. I hope some of their ideas make their way into the Internet’s infrastructure.

*Jeff Johnson*  
San Francisco, Calif.  
[jjohnson@uiwizards.com](mailto:jjohnson@uiwizards.com)

The article by Brian and Elizabeth Whitworth about spam raises some interesting issues. I hope it will help the Internet community to take some positive action to improve the situation.

One useful suggestion they offer is permitting spam to be rejected, returning a failure message to the sender. This feature exists in at least one available product, Mailwasher, which permits spam to be rejected manually or automatically and applies the usual filtering methods. Rejected spam can be deleted from the server, or it can be bounced by implying that the receiver’s address does not exist. Such messages are only partly downloaded to the user for review, without their attachments and without executing HTML code.

I find such methods a trifle tedious, and I hope that e-mails will have to be paid for one day, using a micropayment system, as I am sure that technical measures will always be circumvented if not very inconvenient for the users. Such payments could be kept within the mail-server operator community, and they could eventually reduce their Internet access fees to compensate for the e-mail charges.

I managed to almost eliminate spam a few months ago by changing my e-mail address, as no spam came to my IEEE Computer Society alias at that time. Alas, spam is now starting to arrive there, so I might need to change the alias again, which will be inconvenient to some of my correspondents.

How about setting up a forwarding aid? The IEEE Computer Society servers could store a reply message that I compose, to be sent in reply to all messages received at the old alias address(es). It would specify my new address in a verbose form such as `<p_dot_wolstenholme_at_sign_computer_dot_org_remove_underscores_and_fix_the_rest>` which would not be easily machine-processed (for a year or two ...).

Generalizing this idea leads naturally to the challenge-based spam defenses the authors mentioned. But this simple scheme could be set up easily and quite soon, as it involves only one organization, and standards issues would not arise.

Peter Wolstenholme  
Var, France

[p.wolstenholme@computer.org](mailto:p.wolstenholme@computer.org)

*The authors respond:*

We thank the writers for their useful comments. Spam affects us all, but if we discuss it we can solve it.

Rob Stitt rightly notes the online reality, but we proposed returning rejected mail, not charging for it. We feel that charging for e-mail is counter to the main Internet gain (free communication) and could enable corruption. Returning rejected e-mail, instead of charging for it, would make online reality more fair.

Stitt hits the nail on the head regarding anonymity. An Internet that permits anonymity allows unaccountable communication, that is, spam. Physical societies do not offer anonymity, as this denies accountability. Society knows and “registers” us by recording births, marriages, deaths, drivers’ licenses, voter registrations, and so on. It gives us privacy from others, but not from itself.

Likewise, a communication system must know who is who to impose communication fairness—that to send you must also receive. However, senders can still be anonymous to others—our second proposal lets receivers deny anonymous channel requests, similar to rejecting telephone calls without caller ID.

Security ensures that a system is used as intended, but legitimacy defines that intent. That users are who they say they are (authentication) is a security issue, but what rights they should have (authority) is a legitimacy issue. Both are needed, as security without legitimacy can become domination.

We agree that curbing spam will require changing the Internet infra-

structure, but a “socially blind” approach to doing so is risky. Social principles, like fairness, shed light on how the infrastructure should change. Given a social specification for online communication, the technology implementation is a secondary issue.

Jeff Johnson’s nonresponding business model did not occur to us. We agree on the difference, but talking about data versus information rather than information versus meaning is a terminology issue. We take information as the freedom of choice in a message, following Claude Shannon and Warren Weaver’s original concept (*The Mathematical Theory of Communication*, University of Illinois Press, 1949).

Using the term “information” for a message’s meaning and substituting “data” for the original concept is confusing, so we use meaning for the human aspect of messaging.

The reject methods Peter Wolstenholme suggests still seem to be just filters. Filters stop the appearance of spam but not the waste. I can block e-mail from a source into my trash so I never see it, but it still travels the Internet, and I still download it. Even for server filters, spam is still transmitted. The 95 percent spam figure in our article was for *transmitted* spam, not *received* spam. Filters solve the per-

## CORRECTION

The article by George Lawton titled “The Lowdown on High-Rise Chips” (Oct., pp. 24-27) mistakenly stated that Matrix Semiconductor is using recrystallized silicon to make its 3D memory chips.

Instead, Matrix explained, it is using multiple layers of deposited polycrystalline silicon to build 3D integrated circuits. Company officials said that the recrystallization approach would add cost and complexity to its products. In their initial application, these products are nonvolatile memories used in cost-sensitive applications such as prerecorded content publishing.

sonal problem, but the systemwide problem remains.

We could “shift house” when the spam wolves arrive, leaving a coded message with our new address, but how long before spam programs read the code? Changing your e-mail alias works, but it’s a hassle, and why should we have to? *It is the spammers who should be running.*

# JOIN A THINK TANK

Looking for a community targeted to your area of expertise? IEEE Computer Society Technical Committees explore a variety of computing niches and provide forums for dialogue among peers. These groups influence our standards development and offer leading conferences in their fields.

Join a community that targets your discipline.

In our Technical Committees, you’re in good company.

[www.computer.org/TCsignup/](http://www.computer.org/TCsignup/)