

Origins of Agile Methods

Richard P. Gabriel

Agile methods seem to all be solving a pair of simultaneous problems: creating in an artistic mode while at the same time doing careful engineering. Developing software systems is a creative activity requiring the techniques of science, engineering, and art; and software, unlike art, is also required to perform via execution on (networked) computer hardware. Let's look at these two problems a bit.

Unless a particular system is being implemented afresh after many successful implementations, its actual requirements cannot be determined until its creators are in the midst of its creation, and users of the software cannot know what they desire until they can see what is possible and what it is like to use it. In most cases, interesting new software presents numerous creative challenges, and there are no special ways to handle them in software. Creative activity requires identifying some triggers—facts, thoughts, initial approaches, metaphors, actors, roles, snippets of code, snippets of design, reminders, half-remembered thoughts while listening to customers—doing some construction—be it writing lines or sentences, making some brush strokes, performing some initial cuts—and then finding more or better triggers and refining what is already there. Every act of creation requires triggers.

...art is simply making things in one's own way, guided by the skills and inclinations at hand, experiences, the materials at hand, and the triggers that present themselves. Triggers play the role that most see as creativity or self-expression. A trigger is any thing, place, person, rhythm, or image that presents itself, or metaphor that comes to mind that leads the maker to make a work; often the trigger appears in the final work, and if the work contains a lot of private triggers, it is sometimes considered hermetic.

—Richard P. Gabriel, *Writers' Workshops & the Work of Making Things*

The poet, Richard Hugo, says it this way:

A poem can be said to have two subjects, the initiating or triggering subject, which starts the poem or "causes" the poem to be written, and the real or generated subject, which the poem comes to say or mean, and which is generated or discovered in the poem during the writing. That's not quite right because it suggests that the poet recognizes the real subject. The poet may not be aware of what the real subject is but only [has] some instinctive feeling that the poem is done.

—Richard Hugo, *The Triggering Town*

But just as importantly, the completion of an act of creation consists of a response to what has already been created along the way. The work itself supplies further triggers, and seeing it manifested in the real world, one can see whether one's imagination before construction was sufficient. Hugo, speaking of poetry, says it this way:

The poet's relation to the triggering subject should never be as strong as (must be weaker than) his relation to his words. The words should not serve the subject. The subject should serve the words.

—Richard Hugo, *The Triggering Town*

But a computer program is not like a poem or sculpture: A computer program must run properly and with adequate response for its needs. When building something that's never been built before, the only way to get

it right is to get it right every step of the way, by repairing anything that is new and broken as soon as it's written and by making sure that nothing that is already ok becomes broken. In writing, this means proofreading each sentence as soon as it's complete, if not sooner. When a paragraph is done, proofread it. If possible, engage other eyes and minds to read what has been written already, and if it makes sense, use a writers' workshop to test the final product.

And in a situation where the users of a software system are not its developers, the system needs to be "proofread" every step of the way by those users. Therefore, not only should the software be subject to continual early testing of small additions, those additions should be given to the users.

By making small releases, the creators are also able to gather more and, as well, more accurate triggers, thereby enabling them to get closer to a good piece than if they were to rely on fewer triggers or triggers gathered over a short, initial period. And the course corrections done by small releases enables more eyes and minds to supply triggers—those of the users.

Agile methodologies don't usually describe what they are for this way, and different methodologies address these points a little differently, but if you look at those methodologies, they are trying to gather more triggers through spreading out the gathering period and through engaging crowds or mobs, on one hand, and to be careful to build correctly and simply all through the process on the other. The magic comes from also being able to factor in a response to what has been already created—not only is this further source of triggers but the work itself to teach its creators about itself.

But what is missing from the whole process is what happens next. The agile methodologies are able to create very good first drafts, ones that might also be good shippable drafts. In the written arts, a first draft is followed by numerous revisions, workshops, private readings and discussions, copyediting, etc. This is part of completing a work of art. The agile methodologies are sometimes silent on this part of the job.

Nevertheless, I believe this way of looking at the creation of software—as gathering triggers, responding to existing work, and building correctly all along the way—will illuminate our understanding of agile methods and how to apply and evolve them.