

Diabologic: It's the Scope, Creep!

by Frank Dolinar

Why is writing software so difficult and often unpredictable?

There's a fundamental truth that programmers, systems analysts, and other people in the trade (i.e. people like me), learn early in their careers. If you want a project to ***ever*** be completed, develop only the things that you and the client have agreed to, in writing, and signed.

Everything else is interesting, possibly amusing, fluff that -- by definition -- has nothing to do with the project at hand. More to the point, these additional "goodies" are often things that the programmer thinks are wonderful, and wants to show off, but that the client has not agreed to and very likely won't pay for.

Users tend to have a backwards perception of how difficult it is to develop working software. It works like this: If the user thinks the requested task is really simple (e.g. a mailing list), they've probably internalized so much of what they do that they no longer think about it, they just do it. This means that the programmer must earn his keep by asking a litany of questions to learn what the user really wants and to understand just how complex it is. On the other hand, if the user thinks the requested task is complex (e.g. a mailing list) they will provide documentation of every little detail they can think of to make sure you don't miss anything. It's still a bit of work to go through all the detail, but it's a lot easier than having to generate it all yourself.

The process works equally well (or badly, depending on your point of view) whether you're developing new software or maintaining existing software.

Because software that's actually being used by the client is likely to need occasional and ongoing modification, the projects typically continue to grow, and grow, and grow.

It's a process called "Scope Creep".

The scope of a project is the agreed upon output (screens and reports), the processing (what the program does with the data to generate the output), where it gets its data (from existing databases or input from users), and how the data is validated (does this value make sense in this context?).

It's not so bad when the additions and/or modifications happen after the initial project has been completed and is being used. That process is called software "maintenance". But, scope creep can become a serious problem before the first version of the project has been completed and the client wants you to add... just one more thing. Please. (Sometimes they omit the "Please" and decline to pay you unless you add that thing, and the next, and the next.) (See the second paragraph.)

Now sometimes these additions are simple, make sense, and are easily included. In that case, nearly every software geek I know will simply make the requested change and be done with it.

The real problem arises when the client asks for something that requires major overhaul of the project design to implement. The ensuing discussion can sometimes be difficult.

One of the approaches for dealing with this is called "Agile Development". It is fundamental common sense applied to software development. There are seven goals to making it work:

- Beginning Agility: *What are you trying to accomplish & how do you get it done?*
- Feeding Agility: *Revisions & questions are good because the project continues to evolve.*
- Deliver what users want: *Talk with the client constantly; deliver something useful regularly.*
- Agile feedback: *Get feedback in ways other than from the customer.*
- Agile coding: *Specific practices for writing (and delivering) good, maintainable code.*
- Agile debugging: *Warnings & exceptions are errors; error messages should make sense.*
- Agile collaboration: *Regular (usually daily) team meetings to review status and for planning*

This makes so much sense, I think it should be an industry standard. It's an approach that alleviates problems quickly, eliminates scope creep, and delivers an effective, usable project to the client.

The goals above are taken from the book [Practices of an Agile Developer](#) by Venkat Subramaniam and Andy Hunt. For a good, and more detailed review of the book, see Cory Foy's Slashdot article, which is available at: <http://news.slashdot.org/article.pl?sid=06/11/29/1449208>