It's Okay to Reinvent the Wheel

Gene Michael Stover

created Sunday, 2004 August 15
updated Tuesday, 2005 March 15

Contents

1 Introduction

Among programmers, writing a program that duplicates functionality of a program someone has already written is called "reinventing the wheel". Because writing code is so expensive, & because programmers are often eager to write, we're often reminded that we should never reinvent the wheel; we should always reuse code when possible.

We aren't just reminded of the cost of rewriting code. The concept is beaten into our heads. It seems like not a month passes in which I don't hear or read someone scold a programmer for trying to reinvent the wheel. It's a mindless repetition of an otherwise useful maxim. Presented as it is, it's a cynical cliché, marketed as wisdom. It discourages new programmers. I hate it.

So let's take a look at the "Never reinvent the wheel" rule of thumb, when it applies & when it doesn't.

2 When It Applies

2.1 Other People's Money

Reinventing the wheel can be expensive (A.1), so if you are trying to get a job done on a schedule & within a budget, reinventing the wheel might be a bad idea. What's more, estimates for software development time are notoriously optimistic, so reinventing the wheel is an added expense which is probably much greater than you predict it will be.

Reinventing the wheel is a gamble, & if someone is paying you to write software, you should think twice before gambling with their money.

2.2 The Ends Outweigh the Means

Even if you are not working with someone else's money, or if the grimmest pessimistic estimates suggest that you could reinvent the wheel within the budget, you might want to reuse someone else's code. That's because using existing, debugged code can be the fastest way to reach your goal. If the goal is completed software, you might want to reach it as fast as possible. If you get there with time to spare, you can always go back & reinvent things, or even better, you could improve your software's reliability, maybe even extend its functionality. My point here is that even if your budget & schedule could afford a reinvented wheel, you might achieve your goals more quickly if you use existing code.

3 When It Doesn't Apply

Though they cover a lot of situations, there are only two reasons not to reinvent the wheel. There are a lot of reasons to go ahead & reinvent the wheel.

3.1 Quality

Just because someone already wrote software that does what you need doesn't mean it's good enough to use.

When I need software to do a task & I don't want to write it myself, one of my main complaints with the software I find is that it's not documented well, or it does so many things that I can't find the simple feature I need within the

clutter. This is one version of the quality issue: Good quality software does one thing, does it well, & is documented well.

Maybe no one has made software of high quality which does what you need. Maybe you will be the first person to write high-quality software that does this task.

3.2 Licensing

Maybe the software you need exists, but it's license is incompatible with your requirements. This can happen whether you are writing proprietary or free software.

Software licenses can also get in the way if you need certain rights to the software, but the owner won't sell them to you for a price you could afford. Maybe you want to license the source or want some kind of modifications or support, but the cost of those license features are out of your budget.

3.3 Resources

Maybe the pre-existing software doesn't fit within some resource restrictions you have.

At my current contract, we considered using some encryption software, but nearly everything we found was packaged as shared libraries, & one of our requirements for a particular program was that it was a self-contained, single executable file.

I'm sure similar problems arise frequently in restricted computing environments, such as embedded software.

3.4 Learning

An important part of learning how to write a certain kind of software is to write it. Combine this with research & imagination, & you have the *only* way to learn to write a particular kind of software.

For example, if you want to learn how to write a compiler, write a compiler!

If it weren't okay to reinvent the wheel, then nobody could learn to program because no one would be allowed to write a "hello world" program. After all, that program has been written so many times in so many languages that any new "hello world" program is definitely a reinvented wheel.

Reinventing the wheel in the process of learning through experience is the most legitimate reason to reinvent the wheel, but don't do it on the job unless you have a prior agreement with your patron. If you're learning on your own, there's no reason you shouldn't reinvent the wheel (unless you want to learn to re-use existing code).

3.5 Fun

Sometimes, you just want to write a program for yourself, for the sake of programming. In this situation, if you want to reinvent the wheel, why shouldn't you? (Answer: There's no reason you shouldn't, if that's what you want.)

Where Learning is the most practical reason to reinvent the wheel, Fun may be the purest. It's programming for the sake of programming.

4 Conclusion

I'm not saying that it's always a good idea to reinvent the wheel. I'm saying that it's wrong to insist that it's never okay to reinvent the wheel.

If you're considering a wheel-reinvention & someone has scolded you for it, you might want to reconsider. If you still decide to reinvent the wheel, go for it! That guy who shat on your parade doesn't have as much imagination as you. Ignore him.

If you're thinking of telling someone he's a fool for reinventing the wheel, you might want to consider your motivations for doing so. Is he really a naïve programmer about to increase the expense of a professional project, & you're trying to give him some good advice? Then don't just complain that programmers shouldn't reinvent the wheel; instead, give him some real help & encouragement. Maybe you're about to rain on his parade because you're jealous or you're just a cynic; in that case, shut the hell up. The rest of us have heard it all before.

A Glossary

A.1 expensive

Cost isn't always measured directly in money. When developing software, cost can be measured in money, time, effort, stability, or correctness. You can buy stability & correctness for your software by expending time & effort, so cost comes down to money, time, or effort. But time & effort are both money. So cost does come down to money. Nevertheless, when we talk about cost, it doesn't necessarily mean money directly.

B Other File Formats

- This document is available in multi-file HTML format at http://lisp-p.org/reiw/.
- This document is available in Pointless Document Format at http://lisp-p.org/reiw/reiw.pdf.

References