Microspeak: Fun fork

devblogs.microsoft.com/oldnewthing/20240625-00

June 25, 2024

Raymond Chen

I noted some time ago in the discussion of the Microspeak term *party* that a *party branch* is a branch in which the usual procedures for code changes don't apply, or at least apply less strictly than normal.

A closely-related term for this type of branch is a *fun fork*.[1] The idea is the same: It's a branch where you can make changes more freely than normal. But a *fun fork* is even looser than a *party branch*: The idea of the fun fork is that it will never be merged back to the parent branch. Do whatever you want! Nobody will care!

Usually, a *fun fork* is created so that a team can experiment with a large, complex feature. If the experiment proves successful, the team can move the changes from the fun fork to a product branch. The actual code in the fun fork is not usually taken verbatim directly into the product. It will probably be cleaned up, say by removing things that didn't pan out,[2] removing dead code, refactoring for maintainability. It might even be reimplemented entirely from scratch based on information learned in the fun fork. It is the ingestion into the product branch that undergoes the standard change scrutiny that applies to all changes to the product.

[1] Even thought the term "fork" implies that the changes are going into a clone of the main repo, in practice, the "fork" is usually a branch of the main repo. This makes it possible to use existing infrastructure for managing branches and resources associated with branches, like build resources, ISO and VHD production, and symbol indexing. The term "fork" comes from an old internal Microsoft source control system that did not support branches, so the only option was to fork the entire code base.

[2] For example, the team may have implemented three different ways of doing something, so they could see which one works best. After they decide which one to use, they can delete the other two. You might have a trio of functions called `DoSomething`, `DoSomethingDifferently`, and `DoSomethingWithFeeling`. You decide that "Differently" is the best one, so you delete the other two and rename `DoSomethingDifferently` to `DoSomething`.

Or maybe the team implemented an object called a `WidgetManager`, but as the code evolved, the manager didn't do that much managing any more, so they renamed it to `WidgetSelector` to match what it actually does.