RY vs. DRY

DRY means “Don’t Repeat Yourself”.

The wisdom in DRY is that subsuming all similar code[1] into one routine ensures that subsequent fixes/changes affect all uses of that particular code.

DRY is a *maintenance engineering* issue.

But, it *makes sense* to repeat yourself when *designing* a system. Repetition & cloning are ways to compose ideas into solutions.

Additionally, abstraction is not at the top of mind of “normal” people. Abstraction is taught – as a technique – in computer science school.

RY is a design issue.

DRY – and abstraction – should be automated. This is already done with Git, but in-the-small. We need to extend our automation facilities to detect instances of RY at the design level (code level, if you will[2]).

Much of RY can be converted into DRY using edit scripts. The unsolved part is that of detecting RY that can be collapsed into DRY routines.

We see CSE technology – common subexpression elimination – used in compilers (see the Dragon Book).

Paul Bassett’s Frames technology[3] attacked RY in a different way. Frames saw light as a product in the company Netron. Netron’s product was used in some banks.

From my very light understanding, NiCad (Cordy, Roy) might be related.

All of the bits and pieces are in place for automated DRY...

---

1 What does “similar code” mean? The definition is kind of mushy. Code can be parameterized. Programmers try to “abstract” code by parameterizing it. “Similar code”, then, means any code that produces the desired result after having been parameterized and being given the appropriate parameters. I argue that every parameterization makes code hard to read. Parameterized Types make code even less readable – I’ve seen code that couldn’t be understood because of copious amounts of parameterization. Each parameterization makes the resulting parameterized routine more “tricky” to understand. I argue, elsewhere, that parameters should be entirely eschewed. I argue that a hierarchy of pipelines is a better construct than parameterization.

2 I argue, elsewhere, that code is a poor vehicle for expressing Design.

3 https://www.amazon.ca/Framing-Software-Reuse-Lessons-1996-08-05/dp/B01K17JFS4/ref=sr_1_4?dchild=1&keywords=paul+bassett&qid=1600861493&sr=8-4