[](https://kjmackey.files.wordpress.com/2016/11/gear-wheel-310906_1280.png)In the distant past that represents the early part of my career articles were written about how IT, software development in particular, might be the last place automation took hold. Acres of column inches talked of how everything else was being automated by computers, but software developers felt their work was too complex and resisted the march of progress that was sweeping away everyone else's jobs.

Even so, there were companies building Computer-Aided Software Engineering (CASE) tools. There were even classifications dividing such tools into lowerCASE and upperCASE. I'll leave it to the historians (or perhaps paleontologists) among us to provide the definitions for these classifications.

Speeding forward to the near-past and the present, we find automation taken for granted, or perhaps not even thought of at all.

The 'scaffolding' provided by Microsoft's .NET framework that adds substantial support code for object-relational mapping, the Javascript frameworks (nodeJS, angular—version 1 or 2, etc.) that inject code beneath the tokens supplied by programmers, the database queries generated by analytical & dashboarding tools, even by the humble Excel spreadsheets used by so many, all attest to how automation has helped augment human interaction with computers and the striving to get them to do what is wanted.

The title of this entry is borrowed from a McKnight Consulting presentation I saw some years ago, a cartoon which showed a white-coated individual in a glass case with an "In Case of Emergency Break Glass" sign on the wall beside it.

The context was data science and the thrust of the visual was "Automate everything you can. Use humans where necessary."

In the era of high-speed data, high-volume data, highly-varied data, this remains true. When faced with processing millions of messages an hour, a minute, a...oh dear...second, code is needed. Lots of it. Simple in-line code.

More than a decade and a half ago I was making the case for metadata-driven data movement applications. My thought was a few data flows that would configure themselves at runtime, taking this path or that, depending on the metadata.

A young woman on my team offered a better way. Using the same metadata she generated specific jobs for each type of data flow. Simple. Fast. No runtime decisions to get in the way. New metadata? Just re-run the generation process. Simple.

I've learned my lesson. The tools I use have changed. XML & XSLT for code generation (code is really just text, right?, so XSLT was a good fit) have given way to template code with embedded keywords and Python scripts that know how to apply metadata to those templates.

However you do it, if you're faced with code that needs to process data at volume & speed, take the time upfront to determine what it is you need and see where you can use code generation as a "force multiplier"—a term I despise because of its use by a CEO I once had the misfortune to work for.

But it applies here.