I DON'T DO NEW YEAR'S RESOLUTIONS—THAT'S WATERFALL

Something about the change of a year makes us want to prepare for the next and make plans for how the new year will be better. As part of those plans, people often resolve to make big improvements at the start of the year and then quickly break those resolutions—sometimes the very next day.

For this reason, I gave up New Year's resolutions, but I am still committed to continuously improving. This is true in my own life and in software development. Let me explain.

Missing Big Targets

To me, the tradition of making New Year's resolutions is very much like traditional [mainframe waterfall development](): you plan your enhancements—in this case, resolutions—and deliver them only once a year.

The funny thing about resolutions is that you've already spent a year with whatever behavior your resolution corrects, but only now do you intend to change it, drastically and immediately. In effect, you have wasted months waiting to make a change when you could have been gradually working on it day by day and without added pressure. Now that the focus is on, you decide to go big, and it rarely works out the way you intended.

Constantly judging against the one big goal you want to meet makes it much easier to toss in the towel and quit. The goal is too large. It is binary—you either do the whole goal or nothing. But there is a better approach.

Attaining Small Goals

What if, instead, as the year progressed, you made minor changes and evaluated them each time? You could find out what works, what doesn't, and then modify your approach for the next few weeks.

You could do this with the idea that the incremental "deliverable" achieved every few weeks isn't the end goal—it's progress toward the end goal. You could go for smaller, attainable steps that continuously build into something that brings you closer to where you ultimately want to be.

Most importantly, as the year progresses, you can incrementally benefit by building on successes and learning from errors. This becomes a self-reinforcing feedback loop.

Setting Incremental Software Goals

If this makes sense for self-improvement, it can also make sense for your mainframe software projects. The psychology behind it is the same: Instead of continuing to set up large, binary, pass/fail projects with far-out dates that only build up the pressure you're already under, consider starting on a project now and making incremental improvements with greater quality, velocity, and efficiency.

Break the project into smaller goals that can be met in short time frames, such as two-week sprints, and deliver the benefits along the way. As problems are found, adjust and move on. Define a clear goal that you can meet quickly and soon—that is much more motivating than staring at a huge, far-off goal.

Set resolution goals like these:

- **Resolve to have small, valuable incremental code drops instead of large deliverables:** If you feel they won't have value to the end user, you can at least have code that can be tested independently to make sure it is working before more code is added.
- **Resolve to have two-week sprints instead of long development cycles:** These deliverables need to be frequent, serving as checkpoints so you don't go too far down the wrong path before catching issues. Two weeks seems to be the sweet spot that provides enough time to produce code that is worthwhile, but short enough to get quick feedback.
- **Resolve to embrace change and adjust as necessary:** Frequent checkpoints can help you see what changes are needed and help you avoid the project inertia that can prevent change. Don't' fear changes. Instead, resolve that any necessary adjustments to your design, the team, and procedures must be done immediately—it's always easier to do them sooner than later.
- **Resolve that you will be able to say it's "done enough" for now instead of holding out for perfection:** This can be a very hard resolution to keep. With incremental delivery, you have a fixed amount of work and functionality. It won't be perfect, it may not be finished, but you need to resolve to be comfortable with those smaller incremental improvements, trusting that they will build as you go.
- **Resolve to have some way to measure success:** Don't track progress blindly. You can use key performance indicator (KPI) metrics to improve with each sprint.
- **Resolve to test as you go:** I've saved what might be the most important resolution for last, which is fitting, as testing is so often done last. Shift-left automated testing will allow you to test more frequently and eliminate the usual test phase that looms at the end.

Meeting these small goals is what we call continuous improvement, a hallmark of Agile

development. It's a simple idea, and there is no better time than the present to start. So, if you are going to make a resolution, make it that you will start using the incremental power of Agile today!