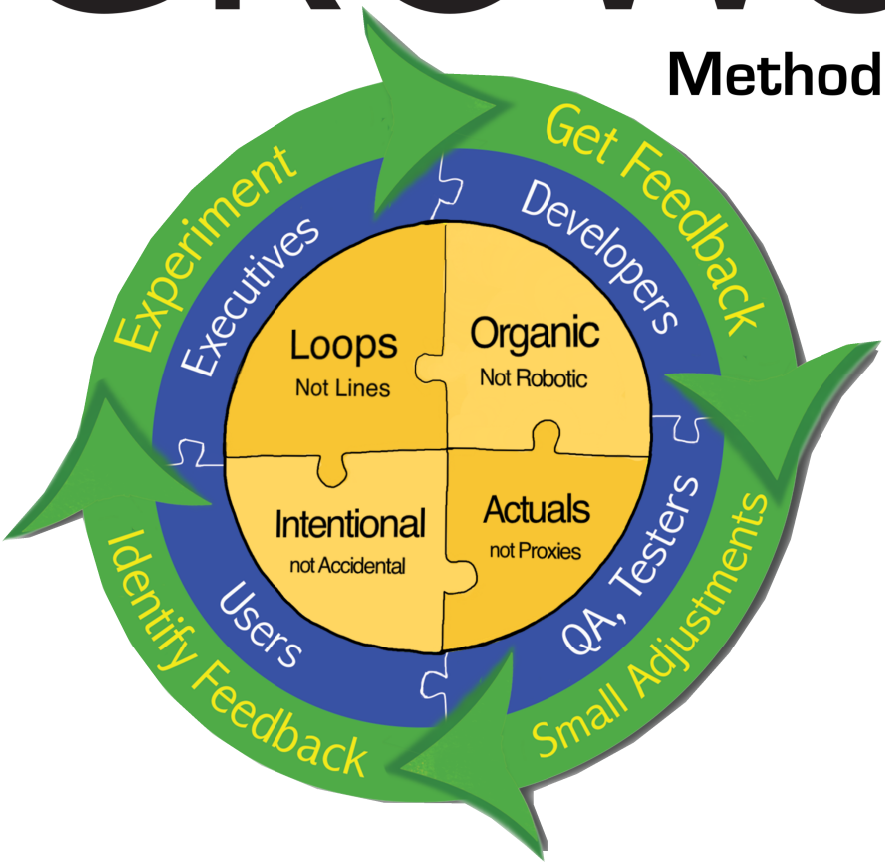
# GROWS

**Method®**




# Learning Journal

The best defense against constant change is constant learning. But not just hit-or-miss learning of random topics as they come up. Instead, you want to work on a more deliberate, personal approach to learning and increasing your skills. Just like in the financial world, always "pay yourself first."

To help you be more deliberate in your learning, we're going to take the following steps:

1. Set a Regular Investment
2. Use S.M.A.R.T Goals
3. Diversify Topics
4. Create a Plan

# Set a Regular Investment

It's important to make a deliberate investment in your learning, in your career. That means working on it regularly. It's helpful to set up a ritual to get into the habit: pick a consistent, quiet, time and place to work.

When and where is your scheduled learning time:

_____

_____

Now once you get there, you don't want to waste any of your precious, carved-out time trying to figure out what to do.  Instead, figure out ahead of time what you're interested in, what you want to work on, what your next steps should be.

Jot down a few ideas of what you think you might want to do next. Don't worry about committing to this yet, we're going to go through some steps to help refine this list. For now, just start with something:

_____

_____

_____

# Use S.M.A.R.T Goals

First, it's far too easy to make a list of sweeping, grand goals, such as "I want to learn functional programming" or "I want to do agile." Neither of those is going to work very well for you.  Instead, use the consultant's trick of S.M.A.R.T. goals--goals that are:

**S**pecific
**M**easurable
**A**ttainable
**R**elevant
**T**ime-boxed

So instead of "I want to learn functional programming," for example, you might instead phrase that as "I will write 'Hello World' in Elm by next week."  Elm is a functional programming language for front-end web applications. That's specific: it's a particular functional language. The idea of a "Hello World" program is measurable, it either works and displays the string *Hello World* or it doesn't.  This goal is probably attainable, "Hello World" in any language is a suitable beginning step. It's relevant to your goal to learn functional programming, and it's time-boxed as you've specified a deadline for yourself.

Now go through your list above and convert those lofty ideas into perhaps more specific S.M.A.R.T. goals:

_____

_____

_____

_____

_____

_____

# Diversify Topics

The next thing to consider is whether your choice of topics is appropriate. Certainly you want to focus on areas that are necessary for your current work environment, or that will be important to your career in this job or the next. But don't forget diversification: branch out to include topics such as:

- new programming languages / development environments
- new techniques (functional, pipeline vs objects, MVC and MVVC, reactive and async, etc.)
- ideas from different industries
- non-technical topics (management, psychology, economics, etc.)

Given that, what topics would you now like to add to your list? Add them here, and remember to be S.M.A.R.T. about it:

_____

_____

_____

_____

_____

_____

_____

# Create a Plan

That list is perhaps getting a little longer and more daunting now, it's time for a plan. You'll want to think a little more long-term, and decide what you want to do:

- Now: what's the next action you can take
- Next Year: what do you want to accomplish by next year
- Five Years from now: it would be impossible to be specific this far out, but in more general terms, where do you see yourself? Think about what might be different in your career, and what you'd like to be doing, five years from now.

Go ahead and jot down these thoughts:

_____

_____

_____

_____

_____

_____

_____

_____

# Rebalance

Finally, as with any plan, the plan itself is garbage ;) No plan survives contact with the enemy. But the planning itself is essential.

Over time you'll need to take a look at your plan and make corrections. You tried that technology, but it didn't work out. You discovered this other new and shiny thing that looks better. The market has changed. Some other skill is now in hot demand, and this other skill less so. These are all normal and expected changes, you just need to adjust your plan to match the latest version of reality.

Make a date for 3 months from now to revisit and revise your plan:

_____

Add that to your calendar. In your re-planning session, repeat these steps, including adding a date 3 months from then to revised your plan again.

Congratulations, you are now ahead of most of your colleagues.

# Making it Stick

It takes more than thirty days to ingrain a new habit. The key to starting any new habit is to touch it every day, even if only for a minute.

Comedian Jerry Seinfield advocates a technique he calls:

> *Don't Break the Chain*

Put up a large calendar on the wall. As you engage in your new habit, however briefly, mark off that day on the calendar. A large, visible calendar serves as a constant reminder to engage in this new activity daily—don't break the chain.

Within a month or two, it will become a new habit.

# What is the GROWS Method®?

The GROWS Method® is a set of habits and approaches to integrate effective software development and learning within an organization. It's based on several key concepts:

- Continuous Development using Tracer Bullet Development to grow software in a steady, robust and responsive manner
- Fast Experiments that provide feedback loops to answer technical questions and drive practice adoption and process pruning
- Continuous Learning using a skill stages model and habits to guide personal and organizational growth and development
- Inclusiveness to ensure that all parts of the organization work together, effectively

# From Seeds to Success

A *seed* is a generative principle that can advise and inform you in many different contexts. Once you understand the nature of a seed, you can apply that idea in multiple situations. This will help you answer questions and solve problems depending on your unique situations. There are four seeds in the GROWS Method®:

1. **Loops, not Lines:** A naive approach often follows very linear, predictive thinking. Unfortunately the real world does not work that way.
2. **Organic, not Robotic:** Systems are not built, they evolve. The evolving, non-linear and learning nature of software and digital work is closer to an organic process.
3. I**ntentional, not Accidental:** Many critical activities are often neglected and left to chance, where they might happen poorly or not at all. Instead, you want to be very intentional.
4. **Actuals, not Proxies:** You want access to the real data, not the proxy. Actual capability, not a piece of paper. Actual shipping software that generates value, not numbers on a spreadsheet.

# For More Information

This is just the beginning! For more information on The GROWS Method® and our workshops and books, please contact us at:

admin@growsmethod.com