Chapter 10 | Primary, Secondary, Tertiary

Got a plan?

Many don’t. I know that.

Around here, if there is one plan there are three plans. We’ve got to have the primary plan, then the backup plan (Plan B as it maybe) then the other plan for when it all goes entirely wrong.

My husband calls it “belt and suspenders” (“belt and braces” for friends and listeners east of me) yet it goes deeper than that. PST or “Primary, Secondary, Tertiary” is a phrase that acknowledges that things go wrong, and one must be prepared with a primary plan, a secondary or backup plan, and a tertiary or contingency plan. In business, in technology, at sea, and even at home one must be able to sustain operations and life with multiple and simultaneous failures.

Even for long-term professional, even for those reported to be the best in the world, including Primary, Secondary, Tertiary into daily events proves difficult. Sadly, the greatest reminder of incorporating PST into the planning process is a catastrophe. Two things happen when stuff goes wrong. First, we are often slow to recognize failure. Second, we are often slower at asking for help.

That things go wrong has typically been a hard lesson to teach. Maybe 2020 helps us all learn that things do go wrong, horribly wrong, unpredictably wrong, and the cascade of errors, issues, and fallout rarely follows predictions.

From a young age, I sailed. For fun, for sport, for holidays – all three sometimes. Sailing and working at sea reinforces lessons that things go wrong. A friend looked me in the eyes after capsizing a small boat in a cold lake. She asked: “Does this float?” Then let it go. We had a lengthy swim to shore towing a boat. Forty years of sailing influences my software development. You must accept that stuff will go entirely wrong. I have scars in each eyebrow attesting to the kinetic impact of a boom to my head. My husband and I got scolded by an official in Seward Alaska for sailing our boat into our slip at the docks. My perfect and well-practiced maneuver earned a “reckless” from that pudding-headed individual. We let him yell. We had a right defense: one, the motor died the prior day. Two, we maneuvered perfectly and hit nothing.

That’s it. Sailboats have extra sails. They have one or two motors. Sometimes, they have oars. Systems have backup systems. Backup systems have contingency plans. A primary radio, a backup radio, backup communications, flares, float plans, etc.

I know that I am writing about computer programming as I wander in thought to messing about on boats. I write this podcast during a near blizzard in the hills of southern Vermont. I expect that the power will fail today. It failed on a team member in Massachusetts yesterday. Our team has always worked from home – or rather, work from where they are, which really means work from where you want to be. Power failures must be an expected operational activity for our team. Our software supports the financial recovery of natural disasters and our private clients expect us to have the resiliency to work through obstacles.

Our firm adopted the Incident Command System or ICF as promoted by FEMA and other federal agencies. While ICF typically identifies a common hierarchy for responders to an emergency scene, it also involves drafting Incident Action Plans. Our firm wrote incident action plans for as many failures as we could imagine.

Then occasionally someone, like me, breaks something without warning flipping our team to into response mode. I do this on purpose to create a drill and remind people of our role in crisis management – even our own.

The most common failure we face, even after years of practicing, is that we fail to recognize failure. Just like all other human beings, we doubt the data in front of us. We cycle through a series of thoughts.

“That can’t be happening”.

“It was working just fine.”

“I didn’t do anything.”

Getting our brains to accept that something went wrong takes training, experience, and a few ticks on a clock. In the U.S. military this delay is identified as: “From flash to bang” – drawing on the physics related to difference in time it takes for an image to travel (speed of light) versus the time it takes for the sound to travel (speed of sound).

The denial of a problem persists through another phase. As we start to understand something maybe slightly wrong, we willingly think something good will happen next.

“The indicator / dial / error report is wrong, and it will reset.”

“Someone (else) will fix this.”

“This isn’t our problem, but someone else’s problem.”

Nice, normal humans do this. First, we don’t really accept what is happening is happening. Second, we doubt own assessment whilst expecting salvation.

If I were disable the Tom Cat server right now on our development environment it would generate a 504-error message on a web browser for the development team. Tom Cat is a server that sits between our desktops and the Oracle Server with APEX.

Disabling that server would be the “flash” in reference to the military phrase “flash to bang”. Measuring the time between breaking a service and notification from others helps me gauge how well our people step through the first two challenges: recognizing failure, acting on that recognition.

Stephanie “Stevie” Dickerson joined our team as an apprentice programmer earned her title as manager of development services. She spent years with the U.S. Navy as a nuclear machinist mate on the USS George Washington, an aircraft carrier. She joined our team well trained on her primary, secondary, and tertiary actions. There are no *small* problems with nuclear engines. And nuclear machinist mate, one cannot expect salvation from anyone else. Few people run towards a problem in a nuclear engine room. You are the it.

Stevie was on board in May of 2008 when a serious fire broke out that injured 37 sailors. The fire spread via a cableway and ventilation ducting causing extreme temperatures in work areas onboard the ship.

Want to know what happens with a warship, or any ship, at sea during a crisis? The command and crew’s efforts must simultaneously manage the crisis and manage the ship. You have five thousand human beings on a rather expensive ship that must stay afloat and get somewhere safe.

During the 12 hours of firefighting, nearly every other essential function on board ship must continue. The nuclear engines do not turn off with a switch, oh, and they keep the ship moving. Crew needs to eat and poo. The ship must navigate and communicate. Systems damaged during the fire need to restart under contingency operations. Unlike a school bus, you can’t just pull to the side of the road and ask all to jump out the back door– although practicing the jump out of the back door of the yellow school bus tended to be my favorite day in primary school.

If curious read about the aftermath of this fire in 2008 that cost the U.S. taxpayers $70 million to repair. After action reports identified failures with training, failures with preparedness, failures with inspections, failures with following procedures. The most critical called the fire preventable given a cigarette butt was flicked next to improperly stowed flammable materials. The smoker should not have been smoking there. Oh, and the materials should not have been their either.

When discussing this cascading set of failures with Stevie she offered up a few more details, about her personal experience. Her general quarters station during the fire was in “shaft alley” about 7 decks down into the ship, where the propeller shafts transit the hull. Just beyond the bulkhead/wall of her duty station was part of the jet fuel storage area. And near where she worked the fire raged, sailors used a plasma cutting torch to rescue someone.

Failure falls on a spectrum. Catastrophic failures tend to follow a series of smaller and observable failures. I just want to say: embrace failures, study failures, learn from failures. Little failure, little failure, little failure… then oops more failures. Then people get hurt and possibly die.

I am writing about software development though, right? I am. I seem to be discussing sailboats and aircraft carriers.

Here is a trick question – well one I don’t wish to be a trick question, but it appears to be:

The first thing that to do when something goes wrong.

[Play Jeopardy Think! music]

“What is acknowledge the problem?”

We expect people on our team to have a primary plan and several contingencies. When planning an upgrade to production software, I expect to hear a plan for the actions including timelines.

The narrative often begins with:

“We will start at 6 am. I will…” and the steps are detailed. Changes to data in data tables, exporting and importing the APEX application, testing the application. We encourage a four-eyes approach to these evolutions – four eyes, meaning two people. One executing tasks and one watching for errors and missteps.

I then ask: “How do you know if you are failing?”

Maybe I should be more polite and ask: “When do you fall back on to your secondary plan?” What’s wrong with failure? As software developers, we learn early to embrace failure. Nothing works the first time we write it, does it?

A failure, given you don’t burn 8 decks and injury 37 sailors, identifies opportunity and sometimes improvement. Failures make up the thousands of baby steps we take. Failure is not a bad word, nor a bad moment – normally. It must be accepted into our daily life and work.

Want to know what I’d sound like while playing a piano? I have never played the instrument. I am certain that I will fail entirely. Put me on a bench with the sheet music for Beethoven’s *Fur Elise* and I couldn’t play it. Some days, I can find middle “C”. So what? Should I never try? Of course not. I should. I must then accept the difficulties and the failures that will follow as a learn. I grew up with amazing musicians and been watching people play piano all my life. I’d love to play the piano, even badly. On my computer keyboard, I am masterful and fast. My typing and writing will likely not translate to the piano. That’s ok. Failures ought not be a measure of inability or lack of talent or even a barrier.

Failure often feels like a barrier when it should not. I worked for Cisco Systems for years as a field engineer. During those years, my bosses required that I continually earn credentials and certifications. In short, it required reading a book and taking an exam. Or taking a class, reading a book then taking an exam. Each quarter I had to sit in a sterile room with video cameras taking some damn exam. I hated failing. Sweat dripped down my nose. Anxiety clouded my thinking. My friend Sargent First Class Parker, who had already had a long career of taking stupid tests for the U.S. Army said: “You never fail an exam. You either reconnoitering the exam; or you earned the certificate.” Yoda may have said it better, right: “Do or do not, there is no try.”

Do and do again.

Learning to accept and work through failures should be a skill we’re taught. Like falling when skiing, getting soaked while sailing, not knowing where middle “C” is without a glance.

Stevie, when discussing the George Washington, said: “the reason people hid hazardous materials all over the ship was that it often took 4 hours to get them from ship stores. There’d be lines. If you needed paint thinner, you’d have to requisition it, stand in line to get it, then also to return it.

“Storing haz-mat like they did, came about because the right way proved so difficult. It was a work-around to a difficult set of rules.” The rules intended to improve the safety of an aircraft carrier and those who served on her, resulted in one of the many tiny failures initiating the cascade towards catastrophic failure.

That effort to acknowledge failure, to embrace failure may, often, break the cascade before it happens. The simple phrase saying: “I see a problem” serves as a key to unlock so many of these issues.

Of course, New Yorkers, then the rest of the United States heard variants of this statement for nearly two decades now. Today, it serves as a punch line to jokes that are not funny. Two decades of posters and electronic sign boards that read: “See something, say something.” Even I, who did my first ambulance call and fought my first fire at the age of 18 or 19, thought the phrase creepier than useful. Was I to spy on my neighbors? More obviously, one is to report the errant backpack dropped and abandon on Boston’s Boylston Street.

Planning to manage an evolution or event requires a degree of acknowledging the futility of the effort. An old Army adage states: “No plan survives the first bullet.” Frankly, planning has truly improved past this point. Plans must incorporate the likely, the possible, the unlikely, and complete failure.

I’ve led meetings and attended meeting where participants dismiss this level of scrutiny. We can’t plan for failure. It pops into vernacular frequently: failure is not an option. Of course, failure is an option. Failure happens. We write plans to prevent or avoid failure which means failure must be an integral part of plans.

To wrap one’s arms around the belly of failure; to look at failure and name it as a failure; to slog through the recovery of failure; to laugh about past failures; to learn from failures; to feel the sting of failures – all a necessary part of a team’s shared experience.

There should be a quarter-note beat between: “Oops” and the next action.

We not only want to shorten the span of time between flash and bang – between “Oops” and act, we really want to shorten the time between flash and response.

The first thing to do when something goes wrong is to see it, acknowledge it, accept it, name it. The second thing to do is react.

The delay between seeing and acting remains slow enough. Within this cognitive delay hides the art of the magician, the boxer, and the pickpocket. Few of us can catch a dollar bill or a Euro note when it hangs between our fingertips. You know the childhood game, don’t you? One friend says to another, I’ll give you this buck if you can catch it. The fingers close on empty air. The bill flutters like a leaf in autumn. We’re just not fast enough.

We tend to make our responses even slower by first not acknowledging the need to act.

Some professions depend on staff that act without hesitation. When in an ambulance squad bay, or at a fire house, the sound of the alarms announcing an emergency triggers near instantaneous movement. Few firefighters or EMTs sit there saying: “Let’s see if that alarm will clear.” Or “That’s someone else’s problem.” Some citizens expect a response when emergency services are called.

Other people when they see something go wrong tap the dial, reset the alarm, and find a way to argue with facts. I have been in office buildings with the fire alarm going off. I walk out, immediately. I rise, gather my kit and git. That’s not the moment to finish a call, finish a meeting, no. It is a horrible and loud noise with flashing lights. It is screaming at all those near to take immediate action. Yet, the next time you are in such a situation, count the people sitting still.

Want to tell me that the fire alarm disrupts productivity? Want to tell me that the fire alarm is faulty? Well, if each time we react, we can document the issue and make improvements. In other words, fix the alarms. That’s rather like sailors on the George Washington informing command that fetching flammable paint thinner from the secure storage locker is too burdensome, so I put it in this locker or this room. Then other sailor saying: climbing decks to smoke in designated spots requires too much time and too much effort. “I just duck into this locker for a smoke. Everyone does it."

The first plan, Plan A, the primary plan lays out the steps from Flash to Bang then to Objective 1. You know the deal, step 1, step 2, step 3, etc.

Then something goes wrong. We encounter failure.

The first time I saw the phrase “PST” was on a banner at the FedEx International Facility in Anchorage Alaska, where I worked in the mid-1990s. In the middle of the massive facility, the teams did physical stretching exercises and warmups. Supervisors briefed teams on news of the day, expectations, timelines, flight arrivals, etc. One dynamic leader name Rosie shined during these daily drills. He hammered in the importance of PST: Primary, Secondary, Tertiary each and every day. He carried himself with the bearing and charisma of a senior non-commissioned officer. PST stood as his version of Clint Eastwood’s phrase: “Adapt, Improvise, and Overcome” We’ve all heard that phrase in the movie *Heartbreak Ridge*. “PST” echoes the same sentiment. During my brief tenure with the FedEx software development team, stuff went horribly wrong on us: a plane crash at Newark Airport in New Jersey; servers crashing; corrupted databases. Oh, what does a software team have to do after a FedEx plane crashes? The answer involves rebuilding a detailed manifest of every item on board, then coordinating with vendors and insurers through the financial recovery process. That’s all data. Having a MD-11 with millions of Timex products flip upside down on a runway did not exist in anyone’s plans. Why me? I was a member of the software development team living in Anchorage, Alaska, the last stopover for Flight 14.

The discipline of acknowledging rapidly changing facts, adapting, then recognizing new objectives bubbled to the top of the FedEx culture as I then knew it.

When I say: “PST Baby” to my teammates, I recall dozens of small and big failures in my career.

When I ask them: How do know if your succeeding? I need to know that at each step we are all evaluating for failures. Don’t keep following a plan that is failing. Stop, re-assess, and look for contingencies: the secondary plan, the tertiary plan.

What is our application upgrade plan for Sunday mornings? It is formulaic and executes in a few minutes. It is nearly routine.

When it goes wrong, we have customers unable to execute core aspects of their business. They can’t management money, documents, business processes.

You may think that nobody really cares when software developers make mistakes. It is our mistakes that result in data breaches, stolen credit cards, stolen identities, and the like. Software is buried into your phone and into your car. There is software in some doorbells and some televisions. Mistakes software developers make can let dreadful and creepy bastards into your house through these portals. Software mistakes can ruin people’s lives. A minority of people in the United States want to blame software in voting machines for the results of the presidential election. Suddenly, software matters. The geeky tool smiths of the modern age keep cars rolling, control traffic lights, tally election results, and secure our on-line purchases.

We’re not allow to have too many mistakes. We’d better acknowledge our failures quickly so that we render our lives a bit safer.

It’s Sunday morning at 6:30 and you are on our team. Looking at the browser, you discover the application did not upgrade as expected? The clock continues to tick. No way to stop time or stop the clock unlike American Football. 90 minutes remain before a paying customer logs in.

How soon do you recognize you are stuck? That progress is not being made. Another of our team’s polestar phrases is: “When swimming towards shore, make sure the shore is getting closer.” We all have trouble knowing, acknowledging, seeing that even though we are working hard, we are not progressing forward. Like sailing in fog, or flying in fog, the external points of reference disappear. One of the reasons we put four-eye (two people) on critical tasks is so that one can watch the clock. In a two-hour upgrade window, you must finish in about an hour. Why? So you can execute your contingency plans and revert to the last-known good place.

Last-known good place means restoring data from backups. It takes time and it requires additional staff to help. Budget an hour for the tertiary plan. With a two-hour upgrade window on Sunday mornings, we need to be successful with the Primary Plan within the first fifteen to thirty minutes, then test. One person evaluates successes and progress. Within thirty minutes, we need to flip to the Secondary Plan. Then if failure is still present, we need to step back from the effort and restore the system to the pre-upgrade environment – The Tertiary Plan.

Twenty-five years ago, a banner hung in the massive FedEx International Facility in Anchorage Alaska. The Banner read:

> Know your PST
> > Primary
> > Secondary
> > Tertiary

The banner referred to plans, actions, duties, and reinforced the requirements for strong knowledge of the shared mission.

That sign resonated with me. My career in IT and my “fun” activities reinforced the basic lesson that stuff goes wrong. Planning must then incorporate failure. Teams must learn to recognize failure and respond to failure quickly.

Failure exists. It is part of everyday life. It is a part of our work and our learning and even our play. Embrace failure as a friend. Learn to accept we’re not progressing. Learn to accept that the anticipated results differ from reality.

Clint Eastwood informs the young marines in *Heartbreak Ridge* to “Adapt, improvise, and overcome.” A more casual approach to developing a primary plan, a secondary plan, and a tertiary plan. Plans need contingencies. And even contingencies need exit plans.

When our team asks “PST”, we are reminding ourselves that our work matters. Our success matters. Teamwork occasionally involves team-speak, short-hand phrases.