

Readings For Problem-Solving Leadership



Second Edition

Esther Derby

Don Gray

Johanna Rothman

Gerald M. Weinberg

PSL Reader

Esther Derby, Don Gray, Johanna Rothman and
Gerald M. Weinberg

This book is for sale at <http://leanpub.com/pslreader>

This version was published on 2017-10-28



Leanpub

This is a [Leanpub](#) book. Leanpub empowers authors and publishers with the Lean Publishing process. [Lean Publishing](#) is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

© 2013 - 2017 Esther Derby, Don Gray, Johanna Rothman and
Gerald M. Weinberg

Contents

Part 1: Leadership Styles	1
Managing In Mayberry	2
Do We Have to Choose Between Management and Leadership?	19
Beyond Blaming: Congruence in Large System Development Projects	22
 Part 2: Designing Interventions for Change	 47
Change, Transitions, and Messes	48
Change is Inevitable	64
The Satir Change Model: Interventions	66
Learning by Design: Constructing Experiential Learning Programs	68
Climbing Out of Technical Debt	76
Seven Lessons from a Top Down Change	81

CONTENTS

Part 3: Observing Yourself and Others 86

 This Title May Change at Any Time! (How Do You Feel About That?) 87

 Beyond Belief 92

 Seeing the Other Person’s Big Picture 96

 Seeing Your Own Big Picture 101

Part 4: Appreciating Individual Differences 106

 The Identified Patient Pattern 107

 Approaching a Conflict in Style 118

Part 5: Building Relationships 122

 The Secret Ingredients of High Morale 123

 Peer-to-Peer Feedback 127

 Why Not Ask Why? 132

 How2 Create a Buddy (Informal Mentoring) Program . . 140

 Getting Some Good Out Of Bad Interviewing 146

Part 6: Group Work 151

 Four Different Ways of Participating 152

CONTENTS

Choosing Facilitation 158

How to Improve Meetings When You’re Not in Charge . 160

Understanding and Creating Safety 163

Collaboration: It’s more than facilitated meetings 172

Part 7: Problem Solving 177

Always Be Second 178

Reasons 182

What’s on Your Not-To-Do List? 188

Staying Sharp 191

The Exception is the Rule 195

Shifting the Burden – Whose Monkey is it? 199

Solving Other People’s Problems 206

Bibliography: Problem Solving Leadership 210

More From Esther Derby 220

More From Don Gray 222

More From Johanna Rothman 224

More from Gerald M. Weinberg 226

Part 1: Leadership Styles

Managing in Mayberry

Do We Have to Choose Between Management and Leadership?

Beyond Blaming

Managing In Mayberry

© 2001 Don Gray and Dan Starr

Near the Blue Ridge Mountains in North Carolina, not far from where you think it should be, there really is a town called Mayberry.

Although the main highway bypassed the town years ago, the namesake for the popular 1960s television series is still a bustling community, and a fair amount of traffic enters Mayberry's downtown from the north on the US Highway 52 business spur every morning. In town for a week of consulting work, we were able to observe the recent road construction along that route and watched a trio of local citizens demonstrate their own unique management styles. Let's take a look at how these characters traffic management closely parallels common styles of *software project* management.

When road work just north of town closed Business 52, all the traffic entering town from the north had to take the 52 bypass around to the west side of town and enter the downtown on Key Street. Unfortunately, this meant traffic would have to make a left turn onto Key Street, crossing fairly busy east-west traffic (see Figure 1).

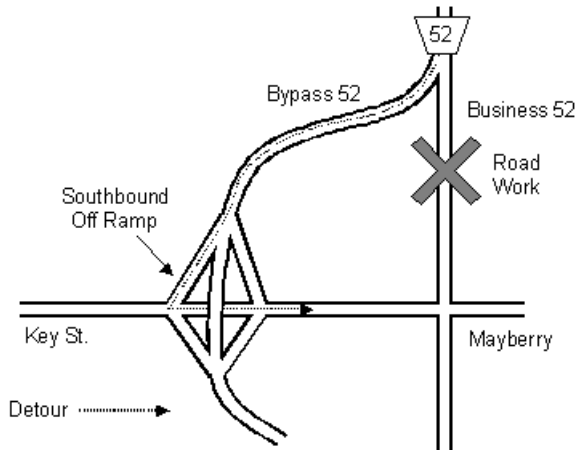


Figure 1

The town council feared that during the morning rush the traffic waiting to make the left turn onto Key Street would back up on the southbound off-ramp all the way to Highway 52 itself. This could cause a serious accident, since Highway 52 has a 65 mph speed limit. So the council decided to station one police officer and one or two rescue squad volunteers at the intersection to make sure that traffic on the ramp did not back up.

Three Approaches to Managing

Being a take-charge guy, the officer on duty (we'll call him Barney) arrived at the scene Monday and quickly sized up the situation. He decided that what was needed was a traffic light at the intersection of Key Street and the ramps. Since it would take the county months to approve a light, he decided to operate as a "human traffic light," directing traffic manually. Each direction got its turn: westbound Key (including left turns onto the southbound ramp), then eastbound Key (including right turns onto the southbound ramp), then the off-ramp (which could turn either way onto Key).

Barney's plan didn't actually work all that well. Traffic stalled in both directions on Key Street. And there were a couple of close calls on the ramp; traffic backed up almost onto Highway 52 once when Barney let a few cars turn left onto Key Street. By the end of rush hour he was hot, tired, and a little discouraged, and he had written a fistful of citations to drivers for making unmentionably rude gestures at a law enforcement officer.

On Tuesday, one of the rescue squad volunteers (a helpful local woman known as Aunt Bea) said she knew how to take care of the situation. She figured that traffic could probably take care of itself as long as drivers didn't have to cross each other's paths. So she let traffic go both ways on Key Street, and let people make right turns onto and off the ramps. When somebody had to turn left, she'd stop the other lanes and let them go. Aunt Bee's approach worked better than Barney's (at least nobody made rude gestures at her), but there was still a lot more congestion than we expected, and by the end of rush hour Bee was glowing profusely.

On Wednesday Sheriff Andy showed up, bringing a lawn chair and a thermos of lemonade. He set up the lawn chair on a shady spot from which he could see the intersection and a fair way down the off-ramp, and sat down to sip lemonade. When traffic started to back up on the ramp, he got up, stopped Key Street traffic, and let the ramp empty; then he went back to his lemonade. Other than that, Andy pretty much didn't seem to *do* anything. Despite his apparent inaction, the intersection just seemed to work. People were calm and relaxed, with the drivers making right turns creating breaks for others making left turns, and everything worked a lot like it did before anyone showed up to help—just a little better.

Putting on our consultant hats, we realized we'd just witnessed three distinct styles of management—Barney's *micromanagement*, Aunt Bee's *motherly* management, and Andy's *masterly* management. Since these styles are also common in software project management. Let's look at each of them in more detail, and see

what we can apply to our own software projects.

A Question of Style

Each of our managers made different assumptions that shaped their style—in particular, assumptions about the *people* being managed, and about the *role* of the manager. These assumptions determined how they approached the critical activities of managing. In his book, *Quality Software Management, Vol. 1: Systems Thinking*, Jerry Weinberg highlights five critical activities:

1. understanding *the problem to be solved*,
2. *planning* the solution approach,
3. *observing* what the people being managed are actually doing,
4. using *rules and process models* to determine what to do next, and
5. taking action*** to guide the group toward its goal.

Together these activities form a feedback system that “steers” the project team. How they are executed (i.e., what the manager defines as the problem, how the manager plans, what observations get made, which rules get followed, and how the corrective actions get taken) makes all the difference—determining just where the team will go, how the team members will feel about the software project as a whole, and ultimately how satisfactory the results will be.

Micromanagement

Barney practiced micromanagement, which is based on the assumption that the manager has to see to it that everything gets done. Most micromanagers don’t deliberately meddle out of a need to be

in control; they're just operating under the assumption that if they don't do it, it won't get done. Micromanagers also tend to make the related assumption that those being managed will do what they're told to do; no more, no less.

These assumptions describe machines better than they do humans. Indeed, when Barney said we needed "human traffic lights," he was describing a situation in which both the manager and those being managed were more mechanical than human. Perhaps this is why so many good programmers become micromanagers when they get their first promotion—they're just "programming" the "bio-robots" who work for them!

Using Weinberg's model, we can see how Barney's assumptions defined his view of the critical management activities:

1. The ***problem to be solved*** was to personally make sure everything was done in an orderly fashion.
2. The ***plan*** that followed was for Barney to pretty much do everything himself. He would personally direct the movements of each and every vehicle. This meant that the plan had to be simple enough that he could be in control of its execution at all times.
3. Even with the simple plan, Barney was far too busy directing traffic to ***observe*** much. Standing in the middle of the intersection, he wasn't in the right position to see up the ramp when traffic began to back up onto Highway 52.
4. Even if he had made better observations, his manager-centered ***process model*** didn't allow him to do much. The underlying assumption that he was personally responsible for each and every car going through the intersection meant that he couldn't delegate much - he couldn't count on the drivers to do anything other than what he told them to do.
5. Barney's ***actions*** were pretty limited; because he had to control each vehicle, he couldn't leave his spot in the middle

of the intersection. In the end, he couldn't do much beyond try harder at what he was already doing—waving his arms more frantically at the folks, in the hopes that they'd get through faster.

Because the manager must make (or at least approve of) all decisions, only one thing happens at a time and everything else lines up waiting for a turn. When simplicity, centralized information, and oversight are turned from virtues into vices, it creates a choke point that affects project planning and execution.

Simplicity Since the entire project plan must be under the control of the manager at all times, the plan must be simple enough that a single person can comprehend it in its entirety. This sets an upper bound on project complexity—if the problem to be solved is beyond this bound, the manager has to simplify it somehow (e.g., letting traffic go in only one direction at a time). This serialization of activities is a common simplification in micromanaged projects as well, and it wastes both effort and time. When serialization isn't enough, the manager may start leaving “non-essential” activities out of the project plan. Micromanagers are notorious for oversimplifying, to the point where their software project plans may leave out something essential for a successful product launch.

Centralized information Since the manager is the only one who can make a decision, it's critical that he get lots of quality information about how the project is doing. Unfortunately, the only observations allowed are those that the *manager* puts in the project plan—but that manager's far too busy making each and every decision to actually observe much of anything. So in practice, micromanagers are often flying blind, making decisions on little or no actual information.

Oversight The need to get explicit approval for each action adds to the amount of time required to accomplish tasks. So micro-management tends to be inefficient, with a lot of people waiting

around for the manager to tell them what to do next. The manager-as-bottleneck is a key structural problem. The practice also leads to people problems, such as initiative squelching. The manager's assumption implies that the people being managed have nothing to contribute beyond the functions defined for them by the manager. What if the workers want to do something other than follow the rules—because they see a better way or a problem with the plan? Forget it. The micromanager will not allow it to happen. This creates short tempers and long days for those who are micromanaged.

Most people don't like this style of management. Some will respond with a sort of dead, mechanical compliance, waiting dutifully for their next set of instructions from the manager. Others may choose some form of subtle rebellion, such as "working to rule"—following the manager's instructions to the letter, no more, no less, even when those instructions are clearly a recipe for failure. And others will rebel more openly, taking advantage of the manager's continual distraction to get away with whatever they can. Alas, these responses to micromanagement tend to set up a positive feedback loop that reinforce the micromanager's assumptions and leads to even more micromanagement. Micromanagers tend to be very busy people.

So, is micromanagement ever appropriate? Certainly, when the problem to be solved is small enough for one manager to truly comprehend the entire project plan, and the people doing the work are willing to follow each and every command of the manager. While this situation can occur now and then, it's not very common in the software world.

A common cause of micromanagement is the newly promoted, technically competent manager rushing in to help a floundering employee or rescue a particular part of a software project. This creates a co-dependent dynamic where the manager becomes the rescuer, and the employee becomes helpless. This ensures that the next time there is a problem, the manager will step in again, and so

on, until something happens to break the pattern.

While micromanaged projects can (and often do) result in successful product launches, it's often more in spite of their management than because of it. There ought to be a more efficient and less aggravating way to handle the situation.

Motherly Management

Aunt Bea chose a kinder, gentler style that we call motherly managing, allowing the drivers to do some things for themselves, and helping them when she thought they needed help. But her underlying assumption was still pretty close to Barney's: the people being managed might be able to do a few routine things without being told, but all significant decisions—especially when there was some form of contention or competition—were still firmly under her control.

If the micromanager views the people being managed as machines, the motherly manager sees them more like children, able to do a few routine things but still needing protection from anything potentially dangerous. Like the micromanager, the motherly manager is not necessarily malicious or desperately in need of control. Aunt Bea had no great need to have power over the drivers; she just knew that they couldn't make major decisions without her help. She simply couldn't visualize the situation where one person could be turning left into the gap created by another turning right, because she couldn't see who was controlling the relationship, and she knew that two drivers certainly couldn't cooperate without somebody to coordinate them.

Aunt Bea's motherly assumptions defined her view of the key management activities:

1. The ***problem to be solved*** was something like "take care of the people who have to cross other traffic." Like Barney, she

saw the problem in personal terms; it was her problem, not the drivers' problem.

2. Because Aunt Bea saw the drivers as human beings who could do a few things for themselves, her **plan** was a bit less rigid than Barney's. She could allow at least a few routine things to happen in parallel, but under exceptional conditions she would take full control of everything, which meant reverting to serial execution.
3. Aunt Bea's more distributed plan required somewhat more sophisticated **observations** than Barney's. She had to observe those situations in which her help was needed—in particular, left turns. Notice that she *wasn't* observing whether people were having trouble making left turns; her underlying assumption said that a left turn signal was a request for help. Like Barney, she spent her time in the middle of the intersection, a point from which she couldn't see up the ramp very well.
4. Because of her motherly assumption that the people being managed couldn't handle any form of contention or conflict, Aunt Bea's **process models** dictated that she must personally resolve these things. So her response to just about any out-of-the-ordinary condition was to stop traffic and go back to taking turns.
5. Like Barney, Aunt Bee was working from a very limited set of **actions**, in part restricted by her need to be in the position of control at the center of the intersection. If those actions didn't work, about all she could do was more of what she was already doing.

Like micromanagement, motherly management can work when its underlying assumptions are true and the problem and solution aren't too complex. Trouble is, most software development shops aren't day care centers, and most development is non-routine and requires that a lot of conflicts be resolved. Interfaces, partitioning,

decomposition, protocols—these are all “left turns” in the view of a motherly manager, who must personally make sure that everybody plays well together. This creates a structural problem similar to micromanagement. Similar, but also different. Since some work can take place independently under motherly management, the manager is less of a choke point than in the case of micromanagement.

But because the process is still highly manager-centric, the actual amount of work that can be done in parallel is often less than expected. We end up with a process that’s very nearly effective: almost parallel, relatively observant, and coming awfully close to giving workers independent responsibility:

Parallel (almost) Only pre-defined “routine” things can take place in parallel. As long as traffic went straight ahead or turned right, Aunt Bea’s plan seemed to work. But she couldn’t predict how many people would want to turn left. When lots of people started turning left, her plan fell apart. In the same way, the actual performance of a motherly-managed software project depends heavily on just how much of the development is really “routine” with no need for interactions or conflict resolution. If there are a lot more “exceptions” than expected, a lot of developers working in parallel according to the project plan may be sitting on their hands waiting for the manager to make a decision. This can make a project plan that was *parallel* in theory become *serial* in practice.

Myopic Motherly managers make more observations than micromanagers, but they still confine those observations to specific conditions noted in the project plan. If the conditions defined by the manager are in fact not the key exceptions that need to be managed, the motherly manager will be spending time and energy observing the wrong thing, while missing the observations that are really necessary for project success.

Nannying Motherly management can be less oppressive than micromanagement for the people being managed, because the “mother” allows her “children” to do a few things on their own. The individual

developers can go ahead as long as they aren't going against the flow or getting into conflicts. But at the first indication that something non-standard is going on, the whole process stops until the manager decides what to do. The manager must handle all the decisions that really matter—and this squelches the individual contribution to solving the overall problem just about as effectively as micromanagement. There is a great deal of variation here—a manager who views the employees as teenagers is less openly controlling than one who views them as toddlers. Still, most of the people who work in the software business have college degrees, and we wonder if we're making the best use of their expensive educations when we manage them as though they were children.

If we are going to find a style that's more efficient and effective than *micro* and *motherly*, we must start by changing our underlying assumptions. Barney sees the people being managed as machines to be programmed; Bea sees them as children to be helped. Now let's see what happens when Andy views them as adult human beings.

Masterly Management

Andy took an approach that at first didn't look like "management" at all. He just sat in his chair, sipping lemonade and watching traffic on the Highway 52 off-ramp. When it started backing up badly, he strolled out into the intersection, stopped traffic on Key Street, and let the off-ramp clear; then he went back to his lemonade. He seemed to be "working" a lot less than Barney or Aunt Bee, yet traffic flowed smoothly. We refer to Andy's style as *masterly*management — because of our three traffic controllers, only he was truly the master of the situation.

The keys to Andy's management style were his underlying assumptions: that drivers are adults, that most of the time they can take care of themselves, and that his role as a manager is to support these competent adults so they can do the real work of getting

themselves safely through the intersection. This is vastly different from Barney's and Aunt Bea's assumption. Andy felt secure enough about his own competence and the drivers' know-how that he could remove himself from the center of the job.

Because Andy did not place himself at the center of the management task, he could be much more flexible and effective at the key management activities:

1. Andy saw the ***problem to be solved*** as moving traffic efficiently and safely through the intersection. He also realized that most of the time this intersection didn't need any help; people made turns here every day without any supervision. What made this a unique problem that might require some management intervention? The detour increased traffic on the Highway 52 off-ramp, and that might, on occasion, cause traffic on the ramp to back up onto the highway and cause a safety hazard. Notice the difference—while Barney and Aunt Bea defined the problem in terms of what they had to do, Andy defined the problem in terms of results, independent of who actually “did the work.” By doing this, Andy positioned himself to observe and “steer” the system that did work, rather than as the person doing the work.
2. With his understanding of the real problem to be solved, Andy was able to construct an effective ***plan*** for its solution. The drivers could be responsible for getting themselves through the intersection. He and his “management team” would monitor the off-ramp and make sure that it could be emptied when (and if) it backed up far enough to pose a safety hazard. While Barney might accuse Andy of not having much of a plan, the fact is that Andy's simple-looking plan actually allowed some very complex things to happen. Because he didn't attempt to control low-level actions by the drivers, Andy's plan delegated management work to individual drivers. This allowed them to operate in parallel,

which they did—drivers waiting to turn left off the ramp took advantage of gaps in traffic created by drivers turning right.

3. Now that he had both a problem statement and a plan, Andy could identify which **observations** he needed to make. To keep traffic from backing up onto Highway 52, he had to watch the ramp—not the intersection. So he positioned himself off to the side, where he could see the ramp. This is another critical difference in Andy's style. Standing in the middle of the intersection, Barney and Aunt Bea were taking in a great deal of information—most of it irrelevant to solving the real problem. They weren't in the right place to make the observations that really matter. Of course, Andy didn't ignore what was happening in the intersection—but he didn't make the intersection his primary focus.
4. Andy's management style used two **process models**. First, if traffic's backing up on the off-ramp, stop traffic on Key Street and allow the ramp to drain. Second, if something blocks the intersection, get it out of the way immediately. The rest of the time, Andy's process model says "let the drivers take care of themselves."

Both of these models are more subtle than they look. The first model allows Andy to do some fine-tuning as the morning progresses. How far up the ramp is "too far" for traffic to back up? At first he took a conservative approach, draining the ramp when it was backed up about halfway to the highway. Later, after observing how quickly Key Street traffic could be stopped to drain the ramp, he changed his definition of "too far" to something more like three-quarters of the way up the ramp. This meant even fewer interventions were needed, because often traffic would back up to the halfway point and then drain back down by itself.

The second model contains a flexible definition of just what triggers action. Andy's looking for a symptom, which could have a variety of root causes. If something blocks the intersection (e.g., a driver too

timid to turn left), Andy's model will handle it.

1. Finally, Andy took a lot less “overt” **action** than either Barney or Aunt Bea. Most of the time it appeared that he was doing nothing at all. Yet, when action was required, he knew what action was appropriate and effective. But it would be wrong to say that Andy's actions were simpler than Barney's or Aunt Bea's. In fact, his infrequent interventions required *more* skill. After all, Barney and Aunt Bea were already standing in the middle of the intersection, and had the drivers' complete attention. Andy had to enter an intersection full of moving vehicles, get the drivers' attention, temporarily interrupt their self-management, get the drivers to carry out his instructions, and finally re-establish the self-managing system. This is a task requiring some skill.

Like the other two styles we've discussed, masterly management works when its underlying assumptions are valid. In software development, where the people being managed are skilled, competent, educated adults, these assumptions are usually *true*. Masterly management, therefore, addresses the structural and behavioral problems we saw with micro and motherly management:

- The delegation inherent in the plan means that most contentions and minor conflicts get solved without the manager's intervention, so most of the time the people aren't waiting for the manager's attention. When a problem does require the manager's attention, that problem doesn't have to wait in line behind a bunch of minor conflicts.
- This support for parallel activities means that masterly management can work with projects that are just too complicated to be understood in all their detail by a single manager—and most software projects would fall into that category.

- Because the people being managed are also delegated a self-management job, they are able to contribute observations that a micro or motherly manager is likely to miss.
- Masterly management involves managing the *project* rather than the *individuals*. Most of the time, the people doing the work are free to pick their own methods within some basic guidelines (for instance, driving on the correct side of the road, or using the corporate standard tool set). This allows creative energy that might otherwise be spent on finding ways to “beat the system” to instead go toward creating profitable products.

In short, a masterly manager like Andy observes and steers a system. If the problem is well understood, the plan is appropriate, and the people doing the work are competent, the controller often doesn't need to do much. Unlike micro and motherly managers, masterly managers spend most of their time in observation and thought rather than in frantic activity. But don't be fooled—when Andy was sitting in his chair sipping lemonade, he was more effectively in control of the situation than either Barney or Aunt Bea.

If masterly management is so good, why don't we see it more often? Because in some ways it's unsettling, especially for the manager:

Looks can be deceiving Masterly managed projects often give a certain appearance of chaos. When Andy managed the intersection, traffic was turning every which way, which was disturbing compared to the neat and orderly behavior when Barney was in charge. However, more traffic moved through the intersection, and did so more safely, under Andy's chaotic-looking management style. Many software projects already look like chaos. Will going to masterly management make them more so? We doubt it; we suspect that much of the apparent chaos in software development comes from resistance to micro and motherly management.

Power is as power does Masterly management requires a different mindset. Most people associate the word *manager* with the word *power*. Yet moving from micromanagement to masterly management involves giving up much of the apparent power and authority of the managerial position, and giving it to the people being managed. The masterly manager has more real power, according to writer Barry Oshry (quoted in Weinberg's book *Becoming a Technical Leader*), if we define power as the ability "to act in ways which enhance the capacity of our systems to thrive and develop in their environment."

Measuring what counts In some organizations (particularly those where micromanagement is the rule), a masterly manager may have a hard time getting promoted. After all, you won't be doing much visible *managing* compared to the micro and motherly managers around you, and it will be easy for the micromanager who makes promotion decisions to conclude that the project succeeded in spite of your "inaction," not because of it.

But masterly management also has rewards. Masterly managers often don't have to work as frantically as micro and motherly managers. As a masterly manager, you're less likely to find yourself in the office at three in the morning, trying to resolve yet another trivial issue. And you'll get the satisfaction of knowing that you're truly an effective leader when the project team says, "We did this ourselves."

Micro, Motherly, or Masterly Management

The best way to determine your management style is to ask questions and observe what is happening.

- Do the people reporting to you scatter like leaves in the wind when you show up? Do you feel like they are performing to

the letter of the law and not the spirit? Do you jump in and start coding when there is a problem? If so, you're probably micromanaging.

- Do you organize workflow for a minimum of interaction so things go smoothly in the team? Do you step in and try to make everything all right for everybody? In crunch mode, do you revert to micromanaging? Your heart may be in the right place, but you may be in motherly managing mode.
- Do you spend a fair amount of time observing what is happening, thinking about the impact the events will have on your team and project, and planning what to do? If so, you may be masterly managing.

If you would like to change your management style, there are some important questions to think about. First, how did you come to have your current management style? For most of us, the way we manage is influenced by the people who've managed us, and by the environment in which we manage. Acknowledging these influences, and the constraints of your current work situation, may help you determine whether it's time for new models. It's important, too, to examine how you *feel* about your style. If you're happy with the status quo, change may not be necessary. But if you feel overworked, and seem to be constantly fighting fires, then maybe a change is in order.

And finally, what would you *like* to have happen? We saw that Barney, Bea, and Andy's view of the "problem at hand" shaped their unique responses, and the same is true for you. Once you know what you would like to have happen, you can create and implement the plans that will allow you to achieve your goals and keep your traffic running smoothly.

Do We Have to Choose Between Management and Leadership?

© 2007 Esther Derby

In a recent discussion on the state of a software company, a programmer declared, “We don’t need managers around here, we need leaders!” I’m always puzzled by statements like this.

“How do you see the difference between management and leadership?” I asked.

“Managers do things right, and leaders do the right thing,” the programmer replied, repeating a Warren Bennis quote.

“But what do they do differently?” I pressed.

“Managers manage, and leaders lead,” the programmer replied with conviction.

Here’s how leadership professor John Kotter describes the difference between management and leadership (which I paraphrase here):

Management is:

- establishing timetables and steps for achieving needed results and allocating resources to make it happen.
- creating structure, staffing and delegating responsibility, and having the authority to accomplish goals.
- monitoring results, identifying deviations, and planning and organizing to solve problems.
- producing key results expected by various stakeholders.

Leadership is:

- establishing direction, and developing a vision for the future.
- aligning people, modeling the vision, influencing, and creating teams and coalitions.
- inspiring people to overcome barriers to change by satisfying basic human needs.
- producing useful change.

Reading these lists, it's clear to me that organizations need both.

Here's an example. A test manager takes a job with a new testing group. He talks with his team, his manager, and the internal and external customers for his unit's work. Based on what he hears, he articulates a mission for the group: "We provide assessments of product quality and help product owners understand risks." That's leadership—setting a direction.

He works with the team to identify all the work they're currently doing, work that's in queue, and projects scheduled for the next several months. Together, they assess what they can accomplish, what they won't do, and whether they have the right mix of skills to do the work. That's management.

He supports the team as it self-organizes to accomplish the work. The organizing part is management (done by the team), while supporting self-organization is leadership—meeting human needs for autonomy.

The test manager works with the team to identify the resources they need—machines, tools, and training—and then adjusts the budget to acquire the necessary resources. That's management.

He's showing leadership when he meets with members of the team to understand their aspirations and help them articulate professional development goals. When they work together to build skills into daily work, that's management.

As the team works to test its products, the manager and the team work together to develop metrics and dash boards that show test progress and communicate the quality of the product—management again.

He makes sure the development manager and product owner define release criteria, leading through influence. He also brings change to the way the company makes ship decisions. When a testing project starts slipping, he pulls the team together to assess the issues and re-plan their approach—management, according to Kotter’s definition.

And so it goes—a little management here, some leadership there. The balance shifts, depending on the situation. The test manager combines management and leadership activities to attend to people and accomplish meaningful work.

I’ve worked with people who were all leadership. When they lacked management behaviors—follow-through and attention to practical implementation—they left chaos in their wakes (and didn’t actually produce much useful change). I’ve also worked with people who were mostly management, which only worked when they had enough personal warmth to navigate human relationships. (In accounting areas, you don’t necessarily want creative ideas or big charisma—think Enron.)

Viewing leadership and management as dichotomous sets up a false choice. Most positions in organizations need both, and that’s what effective managers deliver.

Beyond Blaming: Congruence in Large System Development Projects

©1996 Jean McLendon and Gerald M. Weinberg, www.satir.org¹
and www.geraldweinberg.com²

“England, though at present enjoying a very high state of prosperity, still shows some symptoms of a decaying nation. Propose to an Englishman any principle, or any instrument, however admirable, and you will observe that the whole effort of the English mind is directed to finding a difficulty, a defect, or an impossibility in it. If you speak to him of a machine for peeling a potato, he will pronounce it impossible; if you peel a potato with it before his eyes, he will declare it useless, because it will not slice a pineapple. Import the same principle or show the same machine to an American, or to one of our colonists, and you will observe that the whole effort of his mind is to find some new application of the principle, some new use for the instrument.” – Charles Babbage, 1852

As early as 1852, Charles Babbage could see symptoms of decay and infer from them a vision of future performance. In so doing he provides a perfect description of the blaming style of communication, which emerges in ‘decaying’ organizations—be they nations or software engineering organizations. What is a “blaming style of communication,” and why is it important in systems development?

¹<http://www.satir.org>

²<http://www.geraldweinberg.com>

What Is Congruence?

Congruence is a concept that describes the human experience of alignment between the internal and external—what is thought and felt (the internal), and what is said and how it is said (the external).

In order to operate congruently in the world, you need to take into account three general factors: self (the internal world), other (the immediate external world of people), and context (the larger external world of things, structures, processes, laws, and cultures).

* **Self:** You must consider your own needs and capabilities. Suppose you are a manager who doesn't trust anyone else's judgment, so you try to attend every technical meeting. Doing this, you're likely to overload all your available time, and then be unable to do the managerial job, nor to make real technical contributions in any case.

* **Other:** You must consider the needs and capabilities of other people. For instance, if you are a programmer who refuses to be bothered to write readable code, then testing and maintenance of your code will be a great burden, if not an impossibility.

* **Context:** You must consider the reality of the context in which you are operating. For instance, if you are a manager who insists on sticking with an old design that no longer has the capacity to handle the task, your project may be doomed no matter how hard everyone works. Or, if you are a manager in a start-up company and spend money as if the company had a billion-dollar cash balance, your organization may be out of business before its software product is ready for market.

Congruence is integrity at the most basic level and thus has immense value to a project and each individual in it. Without integrity, we cannot build trust; without trust, we don't feel safe; without safety we have a hard time being congruent. Thus, congruence reinforces congruence in a powerful loop which improves the chances of producing a quality product, on time, and within budget.

On the other hand, the same loop causes incongruence to reinforce incongruence. If a project is allowed to ride such a downward spiral, the integrity of information is destroyed. Soon it becomes impossible for anyone to know what is really happening. Such projects invariably fail, and when they fail, they are invariably found to have been keeping two sets of “books.” Their external picture is not congruent with their internal picture, and they die. Or worse yet, live forever—the living dead.

If congruence is so important for project success, why aren’t all projects congruent? One reason is that congruence is not without a price. Another is that congruence usually involves risk. The level of risk is somewhat contingent on the kind of congruence being demonstrated—mental or emotional.

Mental congruence

In the United States, it’s relatively easy to express our thoughts with out too much incrimination—freedom of speech was a foundation upon which the country was built. Even so, there may be a price to pay for “speaking up.” For example, differing with a colleague or someone in authority at the wrong time can put us on a fast track to isolation, reprimands, reduced opportunities, and subtle door closings. Thus, we’ve all learned the importance of being careful about what we say where and to whom. Saying the wrong thing can lead to heated debates, followed by proclamations of who is right or wrong and who is good or bad. At that point, we’ve lost most possibilities for enhanced understanding and effective communication.

Emotional congruence

In our culture, feelings are reserved for athletic events, celebrations, funerals, near death experiences, deeply felt spiritual experiences,

fight, and exchanges between intimate others, the very young and the very old. We even have many feelings about our feelings and some of the strongest have to do with shame and embarrassment over having them. Feelings are personal and lie close to our heart, where we are tender and vulnerable. No wonder we have all become so skilled at denying our feelings—which necessarily makes us incongruent.

Suppose you are a developer who is scared that you won't be able to deliver a product when you promised. You try to tell your manager about your fear, but he tells you in no uncertain terms what will happen to you if you don't express more confidence. "Why are you so negative? Aren't you a team player?" One way to protect yourself from such negative responses is to live in your head. Perhaps you say, "It's just an estimate; I'm not attached to it," meaning you won't be hurt because you've distanced yourself sufficiently to ward off anything that might hint at rejection. But, though you deny your scared feeling to your manager, you still feel it, squashed down inside. You can stand back away from your ideas, but you always remain standing in your feelings. And, of course, you have been incongruent, and deprived your manager of your best information.

When you share your feelings, your heart-self is being presented to the outer world—exposed to the elements. When you're scared and express your fear while maintaining consideration for the other person (your manager) and the context (the project), you are being congruent. Your critical issue here is, "Can I share my feelings and still be in control?" If the environment of your project is blaming, it threatens to remove your control if you tell the truth—so the temptation to lie about your feelings and your ideas increases. That's why blaming cultures lead to "double books," and that's how they lead to failure.

What is Blaming?

In a congruent organization, your manager asks, “Where does your project stand?” and you answer, “I’m rather scared that I’m not going to make my schedule.” This starts a problem-solving discussion, out of which the two of you make new plans to get the project back on track. In a blaming organization, however, your manager may well tell you that only inferior people lack confidence. In that case, problem-solving will be replaced by blame-avoidance.

From a writer’s point of view, congruent interactions aren’t very dramatic; people just act sensibly, are considerate of one another, get their work done, and enjoy what they’re doing. That kind of behavior might not make as good a soap opera scene as your manager throwing a tantrum and you cringing in the corner, but it definitely makes a better project.

Not that a blaming culture conducts every interaction in a dramatic, blaming way. Under ordinary circumstances, congruent coping is the rule, but if circumstances were always ordinary, we wouldn’t need managers. When feelings of self-esteem are low, they are manifest much more dramatically in characteristic incongruent coping styles: blaming, placating, being superreasonable, loving or hating, and acting irrelevant. We can’t deal with all of these in a short article[1], so let’s discuss blaming, perhaps the most common and most directly destructive of the coping styles.

Under stress, people tend to lose their balance, and one or more of these three essential components may be ignored, leading to a characteristic incongruent coping style. For example, when people fail to take other people into account, they fall into a blaming posture. Here is a typical blaming action you may see in software organizations (italicized words are stressed in this style of speaking—because multiple stressed words in a sentence are a linguistic sign of blaming[2]):

Manager, as programmer arrives late for a meeting:
“You’re *always* late. You *never* show *any* consideration
for *other* people.”

Why is this incongruent? If the manager really is feeling and thinking that the programmer is always late and inconsiderate, isn’t she being congruent by saying so? Yes, but that isn’t what this manager said. She didn’t say, “It’s my impression that you’re always late to my meetings.” Instead, she pronounced her impression of lateness as if it were a scientific fact, never offering the possibility that the programmer might have a different impression. She generalized experience in her meetings as if they necessarily applied to all meetings, never allowing for the possibility that her experience might not be the only one that counts.

If the manager really is feeling and thinking that the programmer is always late and inconsiderate, she might say, “I think that you’re always late, and I feel that you’re not being considerate of me and the others. Is this your perception, too?” (And leave out the stressed words.) Even better management style would be to give the programmer a chance to provide a different perception before launching into interpretation. At the very least, that prevents embarrassment in situations such as the following:

Manager, as programmer arrives late for a meeting:
“It seems to me that you’re always late. Is this your
perception, too?”

Programmer: “Yes, and I feel bad about it. The reason I’m always late is that I have to donate blood for my 9-year-old son, who’s dying of leukemia, and the only time they take donations is just before this meeting.”

Manager: “I’m sorry about your son. I didn’t know about it. Let’s figure out a new meeting schedule so you don’t have to be late.”

More generally, it allows for the possibility that there may be other considerations that count besides those of this one manager. For example, perhaps the programmer is coming from a meeting with customers—a regularly scheduled meeting which overlaps the manager’s meeting.

But what if the programmer really is always late, with no reasonable explanation? Isn’t the manager then entitled to blame the programmer? Not really, because this situation is not about entitlement, but about getting the project done. For that purpose, the problem is most effectively resolved using a non-blaming confrontation with the facts about the unacceptable behavior. By foregoing blaming, the manager keeps the communication clear and open, maximizing the chance that the programmer will receive the intended message. And, of course, receiving the intended message maximizes the chance (though it doesn’t guarantee) that the problem will be solved.

When blaming, problem-solving is less likely because the facts of the case become a minor issue—the major issue in blaming is “who is important and who is insignificant.” When blaming, a person is saying, in effect, “I am everything, you are nothing.” Of course, this stance comes not from really thinking “I am everything,” but just the opposite. Directing the attention at another person—and blaming is often accompanied by a pointed finger—is a self-protective device to distract others from the inadequacy the blamer feels.

Like all incongruent coping, blaming is reinforced by feelings of low self-esteem. When you blame, you attempt to build yourself up by tearing down others because you don’t have the confidence that you can amount to much—or even survive—any other way.

Blaming usually fools people who are unsophisticated, or whose own self-esteem is at a low ebb. The knowledgeable observer, however, sees the amount of blaming as a sure measure of how inadequate the blamer feels. Moreover, if blaming is the preferred project communication style, then it becomes a measure of how

far an environment has degenerated—how little communication is being directed at the project’s issues, compared to the amount that is being directed to puffing up the communicator’s weak self-esteem.

In a blaming organization, it’s not merely the managers who blame, as illustrated by these examples:

Programmer, when asked by a manager to volunteer to talk to a job applicant: ‘Why don’t you do it yourself? I’m not going to do your job for you. If you were better organized, you wouldn’t need to ask me such things.’

Customer, when project manager asks about the possibility of revising the requirements: “You never get the requirements right the first time. If I told you once, I told you a thousand times: Do the job right the first time, then you won’t bother me with revisions.”

(To test your understanding of the blaming style of communication, you might try to improve the congruence of these examples.)

How Blaming Hurts a Project

Of course, people are not perfect, so it’s impossible to conduct a large project without occasions on which people cope incongruently. Normal project management can deal with these situations—when they are exceptional. But when the whole environment encourages blame, each new situation further elaborates the incongruence. Fred Brooks[3] once asked, “How does a one-year project get to be two years late?” His answer was “one day at a time.” Our answer is “one incongruent communication at a time,” as the following example illustrates:

One of the developers was developing a module that produced a printed report when it was tested. The manager put a lot of pressure on the developer to be ready on time, with no excuses allowed. The programmer produced the report, and the manager was pleased

(though he didn't show it, of course—it was “just an expected part of the job” in this blaming culture).

A month later, other people tried to use this module and discovered that it was not finished after all. The developer had used a word-processor to produce a fake report that looked just like a correct test report should look. He thought this would buy him time (it was a month, after all, until anybody found out) to finish the module. Unfortunately, since he was in over his head, a month wasn't enough time.

The manager blamed the programmer. The programmer said nothing, because in this culture of blame, saying something only brought further streams of blame down on your head. The person who reported this incident said that in this organization, failure is not allowed under any circumstances. People who have problems in a project and can foresee slippage are unable to cry HELP! and receive appropriate assistance. According to the managers, each programmer is responsible for meeting the deadlines that the programmer agreed to. Inaccurate estimation is “not allowed” and perfection is to be achieved from day one—otherwise you are put in the pillory of blame. In this situation, fake test reports are the rule, not the exception.

Blaming is the dark secret underlying the failure of many projects. A blaming culture hurts a project in at least six major ways:

1. People commit to plans they know they cannot achieve, at least to delay blame.
2. People hide facts that managers need to control the project, as in the above example.
3. When problems are finally revealed, people avoid coming forth with creative solution ideas, for fear they will be blamed if the ideas don't work, or simply appear dumb at first glance.
4. In day-to-day operations, a major portion of everyone's effort is devoted to positioning themselves so they will not be accused when the time of reckoning arrives.

5. Those people who somehow feel safe enough to focus on the job at hand find themselves spending large amounts of time checking up on the reliability of others' communications.
6. People feel bad most of the time, and spend a lot of time fiddling with unproductive tasks or simply staring at the walls.

What Incongruence Looks and Feels Like

Organizations can be changed from a culture of blame to a culture of congruence. To make this change, the first step is measurement, or at least detection—but how to measure blaming? Actually, an experienced consultant can detect a blaming organization within a few minutes of contact, because symptoms are everywhere. Indeed, people within the organization already know it's a blaming culture—but of course within a blaming culture, blame is undiscussable, and moreover, the undiscussability is also undiscussable.[4] Paradoxically, the existence of undiscussability makes blaming easy to detect. The manager of one project issued a memo saying that there would be no more discussion of project morale, and that he would entertain no questions on the subject because everyone should be grateful to be working on such a terrific project. This could happen only in a blaming organization.

Executives

A culture of blame usually starts at the top. Members of the top level of management are inclined to see the other people in the organization as the source of all problems. The employees are seen as “ungrateful” for the jobs, pay, benefits, and opportunities management has bestowed on them. They are seen to “lack an appropriate work ethic,” “not know the value of a dollar,” “have authority problems,” and “resist change.” These perceptions leave

upper management in a predicament: “Do I fire them, or do I fire the people who hired them?”

Such managers feel that they are trying to realize a vision without getting the necessary support, which leaves them out on a limb. The internal kinesthetic experience of these executives is normally a dull and chronic headache—unless the profit margin is really down. In that case, they have more acute feelings, like pain in the chest and burning in the gut. Their low self esteem reflects outwardly in the form of frequent downsizing, re-engineering, avoiding serious problems, futile memos, and, of course, humiliating of subordinates. Towards themselves, they often practice addictive and self-destructive behaviors (which cannot be discussed, but are always the subject of gossip).

Middle Management

When the top leadership is incongruent, middle managers constantly receive mixed messages. Project managers are told of their importance, then find that their seniors have bypassed them to intervene directly in projects or change the rules without consulting them. They feel as if they are living on a roller coaster—unable to predict whether a particular day or week will be an upper or downer. After being publicly humiliated a few times, they decide their best strategy is to try to stay out of trouble by not ever rocking the boat. Even though they cannot perform at their best, they try to appear important and extremely useful.

In the blaming organization, top managers try (perhaps unconsciously) to teach their middle managers their own blaming attitudes. When one project manager complained of her inability to get the developers to work faster, the Vice-President of Development said, “If your dog won’t jump high enough, get a bigger stick to beat it with.” Living in hail of such incongruence from above, middle managers’ survival issues stay close to the surface. As they did when

they were children, they figure out how to either appease, please, or avoid the power owners. By so doing they insure their survival—and pass the blame on to lower levels.

Employees

At the bottom rung of a blaming organization, employees are usually looking for someplace else to work unless the company is in a stable condition with little competition—or if their retirement is within view. The way to survive is to hide out and appear only to pick up a regular pay check.

Employees are discouraged from thinking creatively—new ideas are interpreted as blaming the management or attempting to usurp their power and prerogatives. Employees are not rewarded for industriousness—but they are frequently punished for perceived “laziness.” Employees cannot seem to find their managers—except when there are problems. Then, the major efforts are directed as attaching blame rather than solving the problem at hand.

The style of blaming varies from organization to organization. It can be harsh, vindictive, direct or indirect—but it is always contagious. Some organizations have polished their blaming style to a high degree of subtlety—without raised voices, merely by a look, or a memo, or an e-mail message, or a phone call, or a visit if things are really bad. In other organizations, the blame is loud, angry, and frequently done in front of an audience of peers—ensuring that all get the message of who is right, who is good, who is in charge, and who should become invisible.

In such an environment, defensiveness becomes pervasive. To those without formal power and authority, it seems that those with power really don’t care about them—and would banish them with no feeling at all. Thus they feel justified in retaliating (in advance, and in secret), and in avoiding their managers and their problems.

Regardless of the style, blaming from the top always generates fear, malaise, errors, accidents, and passive-aggressive responses from the bottom. Those on the bottom feel small and act from a place of powerlessness. The lack of emotional safety effectively erodes the trust level and makes any attempt at congruence extremely risky. This environment sounds awful and it is—both for the person who has regressed into emotional immaturity and, sadly, for the person at the top who is doing the blaming.

Those on the bottom of any large organization can easily come to feel a sense of dependency on those above them in the hierarchy. When blaming is the primary mode of dealing with people, this dependency is exacerbated. Then, out of a feeling of dependency, people easily generate a feeling of hostility. As this hostility grows so does the debilitating experience of shame—that overly critical judge that lies latent in all humans.

What Congruence Would Look/Feel Like

Most people who have experienced a congruent organization won't tolerate the misery of working in a blaming organization. But many people haven't ever had that experience, and have a hard time believing what a congruent organization is really like. Let's look at what would happen if a healthy dose of congruence could be magically applied on a large scale to an incongruent project organization.

Executives

If we could magically install congruence in the internal programs of those blaming executives, their style would shift dramatically. For example, if they would truly consider the others involved in their communication, they would be more likely to believe in the intent

of people to contribute, to be productive, to belong, and to learn—and would take deviations from this ideal as evidence of ineffective management. Their belief in the inherent value of all people along with a healthy respect for the constraints of the work context would engender energy, hope, appreciation, understanding, and gratitude among their employees.

An executive who truly does not believe in the good intentions of the employees will be likely to say, ‘No excuses! You will get this done on October First.’ But, with employees whose intentions are bad, this style (or any other style) isn’t really going to work.

A congruent executive who truly does believe in the good intentions of the employees will be likely to say, “We need this badly by October First. What do you need from us to help get it?” This kind of mutuality and support enlivens a genuine “can do” feeling that increases the chance that a project meets its goals—and that nobody has to make false promises to escape abusive blaming.

When the top level managers sustain their commitment to congruence, they see that most workers appreciate the opportunity the business provides them in developing skills, meaning, relationships, and monetary rewards. They also know how to cope when the occasional worker doesn’t seem appreciative or even productive. Managers who know how to use their power congruently generally get the results they seek—not perfection, which they know not to expect.

These leaders know they have a special kind of power—power they use with awareness and sensitivity. They do not resist accountability to those they lead, but demonstrate the same level of integrity they seek from others. And if they cannot match the levels of commitment they request from others, they are open about that. They know they are sometimes going to be weak and vulnerable and need support—perhaps even to see the value of their own visions. They use their awareness of this human reality to nurture their capacity to empathize and to have compassion for themselves and

others.

Congruent executives know that their principal job is developing their organization's capability, not just pushing the same old shoddy products and services out the door. They involve themselves seriously in organizational improvement efforts while simultaneously involving others in the organization to ground these efforts in real-life, practical operational input and decision making. They know that synergy is needed for organizational development, and they know that synergy comes from high quality connections among people—regardless of level.

Middle Management

When the top folks begin to operate from congruence, the middle managers receive direct, clear messages—not mixed messages with double meanings. Communications are more open, making it is easier to know more about what's really going on. Given higher quality information, they know more about how to be useful, so they can more easily join their leaders in their visions. Knowing more clearly the strategic directions desired and feeling that they count in this process frees them to contribute more generously and thoughtfully—rather than merely playing safe. Success becomes a goal that all can share.

Given their unique vantage points, middle people have useful input to assist in predicting problems, projecting realistic time lines, and forecasting trends. What they see, hear, think and feel is valued, and they are in a position to initiate behaviors that prevent project weaknesses from growing into project failure. They know the necessity of interdependence, so if major problems do develop, they can be counted on to provide—and seek-truthful information. They are not ashamed or afraid to work for those who employ them. Indeed, they have pride about their commitment to the organization—and know it is a commitment not so much as to schedules and budgets,

but to the truth about schedules and budgets.

Because congruence at the top trickles down, middle managers take notice of the difference in their leaders. They respond to the modeling by passing it on to their constituencies. Everyone in the organization knows what is at stake in doing each job well, so everyone feels safe to tell what is wrong, what is getting in the way, and what is needed to fix it. Honest reporting of facts and feelings is genuinely appreciated, and do not put people at risk of being humiliated or losing their jobs. That's why congruent organizations deliver their projects as promised.

Congruent middle managers encourage high quality communication. Their belief in people's ability to learn and change towards more congruence make those around them responsive. With congruence radiating from the center of the organization, everyone can have a place, position, and function of importance and value—so things get done, and done right.

Employees

Working at the front line of a business where the top leadership is congruent, is entirely a different experience from working in a blaming organization. Commitment and energy are the norm, not the exception practiced by new employees until they “learn the way things are around here.”

Congruent organizations hold to an ideology that doing well in the marketplace is connected to doing well with employees as well as with customers. The perspective of the leadership includes a global consciousness about the existence of multiple non-linear factors, the importance of connections among all the various parts of the whole, and the necessity of all parts knowing their value. Workers feel that this is a company going somewhere, where growth is a natural state and everyone's efforts count.

Workers in a congruent organization tend to have a long range view and can usually maneuver as needed to meet the changing needs of clients and customers. Employees trust that what they see and hear is real. They share in the enthusiasm of creating a future. They may not like everything that happens—for instance, they don't always feel that they are rewarded adequately for what they give—but they don't feel that there is a chronic pattern of undercutting, diminishing, discrediting, and devaluing them and what they do. They can risk congruence knowing that it will act as a catalyst for optimizing successful outcomes that benefit everyone.

Congruence is the bright secret underlying the success of many projects. A congruent culture helps a project in at least six major ways:

1. People commit to plans they know only after open negotiation, so plans are more likely to be realistic in the first place.
2. People come forth readily with facts that managers need to control the project, as soon as they are known, so managers can act early and act small to correct the problems.
3. When problems are revealed, people readily come forth with creative solution ideas, increasing the chances for quick and effective solutions.
4. A major portion of everyone's effort is devoted to getting their jobs done, and helping other get their jobs done.
5. Because human fallibility is considered normal, an appropriate—but small—amount of time is spent assuring the reliability of communications.
6. People feel good most of the time, and thus are productive most of the time.

Congruence in Large Systems Development Efforts

In the course of developing systems, people engage in numerous acts of communication—about requirements, schedules, interpersonal problems, designs, progress, and just about anything else. That's why effective individual communication is important in all projects, large and small. That being said, effective communication becomes even more important as the size of the development effort grows. The number of necessary communications goes up non-linearly with the size of the project, so the effect of imperfect communication style is magnified. Thus, if the quality of individual communications remains fixed while the project grows, the overall quality of communication will go down.

For instance, a certain level of congruent communication might be adequate for a producing a product with 25,000 lines of code, yet be totally unacceptable for a product with 2,500,000 lines of code. In order to develop larger and/or more complex systems, then, it's not sufficient to pay attention to technical issues—accepting that the existing communication style will be adequate. Managers must also improve the project's communication culture, and thus they must pay more attention to congruence.

To make matters worse, unless we manage well, tougher projects tend to diminish congruence—because stress tends to rise when the expectation of quality rises. We are not always utterly logical creatures, but have feelings as well as thoughts in response to tougher assignments. When these inner feelings are strong enough, they translate into characteristic styles of coping with the stress. If our characteristic style is incongruent, communications become less effective and the job becomes even more difficult, creating a vicious cycle.[5]

Congruence, of course, is but one of the factors in effective communication—other factors include such things as timeliness, memory, proper

audience, and accuracy of data. But without congruence, your efforts to improve these more “logical” factors will always be seriously undermined, along with your ability to build bigger, more complex, or more reliable systems.

Achieving Congruence

When Deming said, “Drive fear out of the workplace,” we think he was talking about changing the blaming organization to the congruent organization. This kind of change is made by one person at a time—hopefully starting at the top—and one step at a time. The steps can be broken down into six ‘A’s’, as follows: Awareness, Acceptance, Authorship, Articulation, Application, Activism. Let’s look at how each of these steps takes place in the context of an individual trying to change a blaming organization.

Awareness

Awareness says, “This is happening. This is real.” Awareness comes from experience, when I allow myself to experience the world around me as it is—not as it is supposed to be, or I wish it to be, or someone else tells me they want it to be.

Awareness is always the first step, and probably the hardest, because generally we’re not aware that we’re not aware. Here’s a personal example of how lack of awareness stops the change process before it can even start:

Jerry was attending a project meeting in a software company—a meeting called by the company President to find out what was going on in a late project. After some coaxing, one of the developers said that she was afraid to go to Nat, the Development Manager, with problems, because of the reception she got. Nat got red in the face, stood up, and shouted angrily, “How can you say that? My

door is always open to hear your problems! The only thing I won't tolerate is if you're all emotional when you come, or if you don't have a proposed solution!"

In the calmest voice he could manage (it's hard to stay calm when someone is being so angry, even if it's not directed at you), Jerry turned to the President and asked if Nat ever came to him with problems. When the President said yes, Jerry asked if the Nat was always calm and carrying a proposed solution. Before the President could answer, Nat interrupted: 'Why would I come with a problem if it wasn't important enough to get excited about? And, if I had a solution, why would I come to him?"

Although it was now clear to everyone else in the room that Nat was demanding that others "do as I say, not as I do," he was unable to see the incongruence. Lacking awareness, there was no way Nat was going to change—and indeed he never did change, up to the time the President released him to seek greener pastures.

Nat's case is quite typical. Since incongruence is a defense, incongruent people erect all kinds of shields that close off information about congruence. Their own incongruence, and that of others, is invisible—it is accepted, especially if it is the norm in the organization. This invisibility makes it very hard to reach them with any kind of information on the subject.

In other words, when you're being incongruent, you're losing your ability to take in what's going on in the world (inner or outer). So, you don't know that you need changing. And, even if you did, you haven't a clue what to change to. No wonder it is so difficult to transform an incongruent culture, when the very first step—awareness—is so hard to come by.

Awareness comes from experience, when you allow yourself to experience the world around you as it is—not as it is supposed to be, or you wish it to be, or someone else tells you they want it to be. But in the blaming organization, where people shield themselves from experience, becoming aware usually requires help. Helping others

become aware takes the skill to develop safe environments and to build relationships. It takes the patience and caring to watch for signs of awareness and help build on them. It also takes a belief and a commitment that “part of my job is to help the people on my team to develop– the most important part.” If you don’t believe this, then certainly don’t try to help people become aware. Otherwise, you’ll find yourself saying, “You aren’t aware of what a lousy employee you are, but I’m going to make you be aware!”

But awareness of the overall situation is not sufficient—you also need self-awareness. Self-awareness says, “This is me. This is mine.” You may be fully aware of the blaming, but as long as you merely say, “This is a blaming organization,” you’re not doing anything to change it. When you say, “I am a part of this blaming organization,” you move forward. You own the blaming as a part of yourself and your behavior—not just something that “they” do (to you).

Self-awareness is often followed by depression or shame or guilt. Some people react with anger, at themselves or at any convenient target. Yet Self-awareness is empowering—the thought that since I own it, it’s mine to do something with.

Acceptance

Acceptance moves the change process beyond self-blaming and says, “I’m not a bad person because I do this. My intentions are good, though my actions may not be effective.” Acceptance means that you understand that taking responsibility is not the same as blaming yourself. Thus, you have mercy on yourself and your all-too-human imperfection. You stop being angry. You forgive yourself for not doing better in the past, based on your present understanding and standards. And, as you forgive and accept yourself, you gain compassion for the others involved—thereby increasing the chance that you can communicate with them and effect change.

At the point when you’re trying to reach acceptance, it’s critical

that you not be punished or humiliated by someone else. You need a little help in getting off your own back, or else you think so little of yourself that you couldn't possibly do anything about the situation. Of course, in a blaming organization, you may have a hard time avoiding this kind of punishment, which is why authorship and acceptance are usually done internally, and kept internal for some time.

Authorship

Authorship is the first decision point, when you say, "I have choices. I can do something about this." With some encouragement, you accept that you are responsible for choice in your life. You understand that you don't have to react, but that you can choose your response—that you create, in large part, your own interpersonal context. You know there are some parts of the context that you can control and some that you can't; and you know accurately which is which.

Articulation

Articulation is the public commitment to change, and says, "I'm going public with this (for accountability and support)." Articulation is ineffective if attempted before the prerequisites are in place. If you can't accept yourself or how you have reflected yourself out to the world, or if you don't know that you have choices or feel you can gain support for those choices, then speaking out is merely ineffective bravado.

But, when the prerequisites are in place, you cannot be effective by keeping silent—you must decide to speak out. In the process of speaking you transform your inner awareness to another kind of experience. You hear yourself and you notice the response you get

from others. You make public, if you will, your self—your mental and emotional position.

Initially, of course, you must seek out safe places to disclose your truer and more honest expressions of your thoughts and feelings. When you become more grounded in the power of your true self then you can seek the kind of support that challenges and confronts you, as opposed to the kind of support that coddles and consoles.

Initial steps of articulating congruence are often awkward. That's why a responsive and receptive listener satisfies one of the requirements for promoting the development of congruence.

Application

Application says, "These are my choices (my new ways of coping)." You learn to be congruent yourself, first in your most immediate, safe, and encouraging context. Then you expand the contexts in which you can respond congruently. Don't try to "not be incongruent." This paradoxical command only invokes the incongruence of perfectionism. ("If I can't be perfectly congruent all the time, I'm worthless.") Focus on congruence, practice congruence, and the incongruence "muscles" will simply atrophy.

With support and practice you can begin to use and test out congruence in your immediate relationships. We suggest that you continue to design for success, so that initially these tests of your new skill are done within environments where you will more likely be given the benefit of the doubt. As you experience success, then you can be centered even in more turbulent and conflicted arenas. In other words, once you "get the call," don't march into the president's office and announce that henceforth, all the guilty parties must stop blaming, or else.

Activism

Activism says, “Now that I can make a difference in myself and my most familiar world, I’m going to help spread this throughout the organization.” Activism is applied leadership, starting at the point at which you have enough competence at being congruent to reach out and be proactive–anticipating, initiating, instigating–but not inflicting. You cannot operate from an incongruent position and force other people to be congruent. (“I have to blame them, because they’re so blaming. Once they change, then I’ll be able to change.”)

In any case, you don’t have to inflict congruence on anyone. Congruence is contagious–when directed consciously to creating a safe, nurturing, productive environment. It may spread more slowly than you’d like, but once it starts moving, it’s hard to stop.

Notes:

1. For more on how blaming and other incongruent styles impact software work, see Weinberg, G. M. (1994). *Quality Software Management: Congruent Action*. New York: Dorset House.
2. For more on incongruence and verbal patterns, see Hardin, S. E. (1989). *Success with the Gentle Art of Verbal Self-Defense*. Englewood Cliffs, NJ: Prentice-Hall.
3. Brooks, F.P. (1982). *The Mythical Man-Month*. Reading, MA: Addison-Wesley.
4. For more on undiscussability, see Argyris, C. (1993). *Knowledge for action: a guide to overcoming barriers to organizational change*. San Francisco: Jossey-Bass.
5. for more on such system dynamics of projects, see, for example:

Senge, P. M. (1990). *The Fifth Discipline: The Art & Practice of the Learning Organization*. New Turk: Doubleday.

Weinberg, G. M. (1975). *An Introduction to General Systems Thinking*. New York: Wiley-Interscience.

Weinberg, C. M. (1991). *Quality Software Management: Systems Thinking*. New York: Dorset House.

Part 2: Designing Interventions for Change

Changes, Transitions and Messes

Change is Inevitable

The Satir Change Model: Interventions

Learning by Design: Constructing Experiential Learning Programs

Climbing out of Technical Debt

Lessons In Top Down Change

Change, Transitions, and Messes

©2012-17 Don Gray

The team I worked with was scattered in a cube farm with 5 foot walls. Well, except for the George and Ken who chose to work in a large storage closet.

After the first sprint, we finally moved to a common area. Still in cubes, but with short walls so everyone could see everyone else. Except of course, the two guys in the closet.

It took some influencing, but eventually George and Ken moved too. Some people just don't seem to be in a hurry to change.

In looking back at this event I notice the team members experienced change and transitioned to a new way of working together. Managers made plans using change models. The success of the change hinged on the combination of these events and activities.

Understanding Change

We can use the Satir Change Model to step through what the team members experienced.

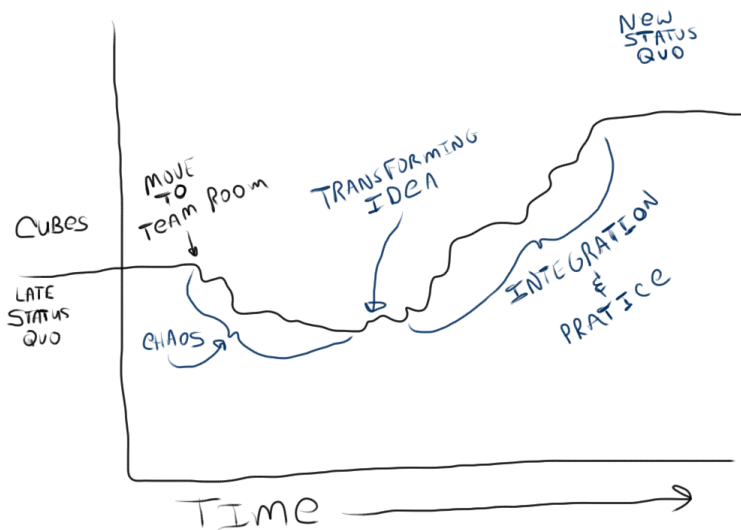


Figure 1 - Satir Change Model

Late Status Quo - things hum along nicely. Everyone knows what's expected from them and how to do it. In a word - comfortable. Eventually along comes a

Foreign Element - this event shocks the system. It could be a manager forcing a team to move to a team room. It could be changing from gated to incremental and iterative development.

Chaos - the system now operates in ways we cannot predict. People no longer know exactly what's expected, and if they have an idea, they don't know how to do it. Team members try many ideas to get a semblance of order.

Transforming Idea - the team discovers how the change benefits them. They discover osmotic communication. When they need quiet to focus, earbuds covered by noise canceling headsets appear! (I've seen it done) Interestingly, when conversations started, off came the headsets and out came the earbuds.

Integration and Practice - the team experiments with their environment. They add mood lighting. Their working agreement gets amended to include quiet time for deep work. Mob programming emerges as a new practice for the team.

New Status Quo - now the team has integrated the change to the team room and have become comfortable in it. They know what to do, and how to do it. This eventually becomes the next Late Status Quo.

Sometimes, it seems like the people asking other people to change believe change happens like flipping a switch. When they see people holding on to what used to work, flailing, or trying out many new ideas they think something is wrong—either with the change or the other people. But this process is normal, natural, inevitable.

This represents the change “happy path”. The journey from Late to New Status Quo contains many hazards and opportunities to revert back to the Late Status Quo.

Failure Paths During change

Many designed changes never change anything. Other change initiatives look promising for a while, but as time passes people revert to working “the way we’ve always worked around here.” In the “move to team room” scenario these would be the team never moving into the team room, and the team moving to the team room, but slowly reverting to staying in their cubes.

What happens in these situations? Actually, many paths exist between starting a change, and ending back in the old status quo. How many paths can you find in this figure that lead back to the old status quo?

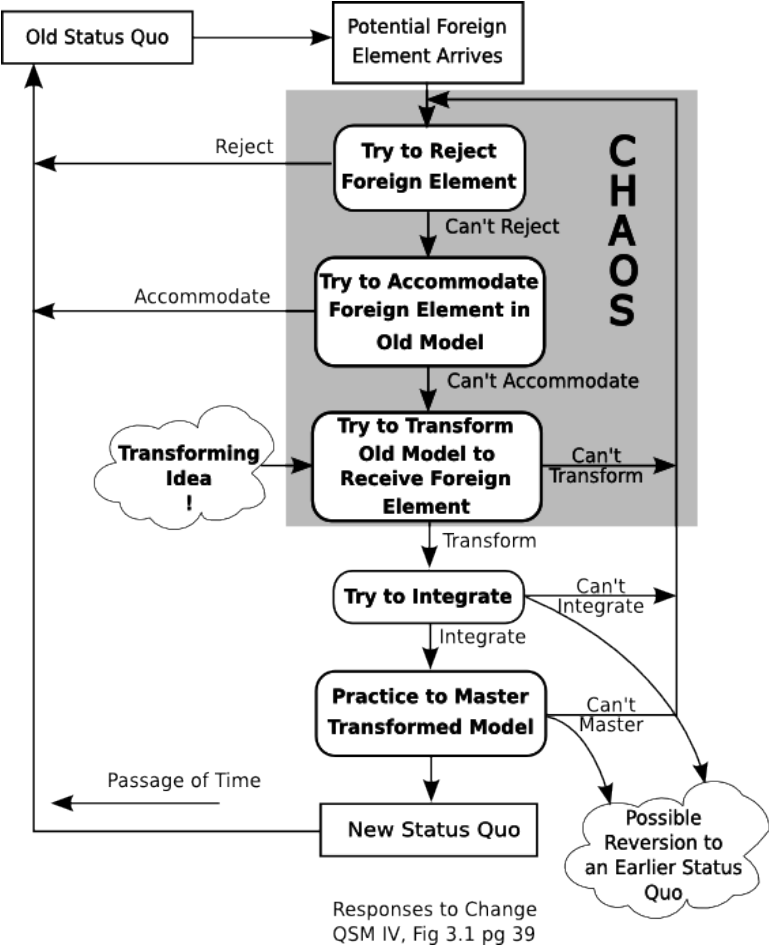


Figure 2 - Change Choice Points

Now you can see what traps change initiatives and takes them off track, usually back to the old status quo.

Responding to Changes

You've probably noticed people behave differently. **Differences become pronounced during change.** People start to act out in ways that don't make sense to us. But if we look at peoples' temperaments, we might have an idea what's happening within them. Based on the *Jungian psychological types*, [David Keirsey's four temperaments](#)³ allow us to understand weaknesses and strengths during change.

CHANGE AND TEMPERAMENT¹

The Visionary (NT) likes working with ideas. NT Visionaries are most interested in designing, rather than implementing, change. They like to provoke with ideas, even during Chaos when such provocation is inappropriate and may cause much pain and confusion.

The Catalyst (NF) likes working with people to help them grow, but is concerned that people should not suffer from change. They have a tendency not to let people experience their own pain, so they may short-circuit Integration by trying to be helpful. They are such team players that they may want everyone to do the same thing, even if their personalities are different. Also, they may want everyone to do something at the same time, even if people are at different stages of the change process.

The Organizer (SJ) likes order and system. The important thing [to them] is not just doing it, but doing it right. .. They are best at carrying the transformation into actual practice, long after the NT Visionaries have gotten bored. Although SJ Organizers tend to fear quick change, they may push for quick closure. An example is getting firm commitments during Chaos when it is clearly inappropriate. They may also stifle all change by requiring that success be provable in advance.

³<https://www.amazon.com/Please-Understand-Temperament-Character-Intelligence/dp/1885705026/>

The Troubleshooter (SP) likes getting the job done. [They] want quick fixes, not elaborate plans. They are the least likely to deny the foreign element, because they see it as an opportunity to swing into action. For SP Troubleshooters, change should be fast. They don't want to get stuck with something that's dragging on and boring. As a result they are impatient with planning. SP Troubleshooters may provoke change for its own sake, piling one change on another, even during Chaos. Impatient with Integration and Practice, they may drop out if change seems too slow.

It's important to notice that **each temperament has both strengths and weaknesses during change**. We can utilize different strengths to help achieve the desired *New Status Quo*. Knowing where the change currently resides on the Satir Change Model allows us to understand why team members act like they do.

These descriptions aren't "one size fits all." People have many facets so don't expect all NTs to act the same way during change. At minimum these descriptions could be thought of as "*tendencies*". Sometimes they'll be dead on. While not *truth*, they can be useful.

In addition to Temperaments, different people [process change at different rates](#)⁴, and other parts of our personality like [survival rules, family background and self-esteem](#)⁵ affect our responses to change.

Change can seem simple and straight forward when drawn and diagrammed with nice squares, straight lines, and check lists. But when we look at what actually happens, change is anything but straight forward.

⁴<https://www.donaldegray.com/change-quotients/>

⁵<https://www.donaldegray.com/this-title-may-be-changed-at-any-time-how-do-you-feel-about-that/>

Organizations Change - People Transition

Googling “**organizational change**” returns almost 6 million hits. The LinkedIn Organizational Change Practitioners contains 26,125 members. With this much information and so many people practicing organizational change, you’d think we’d be good at it.

But that’s not what I usually experience in my work. Why does this gap exist?

A Handful of Change Models¹

What models exist for changing organizations?

- **The Diffusion Model**, which says change more or less happens. A common example of this would be communities of practice in an organization. People choose to meet and share ideas and how they accomplish their tasks. Attendees can choose or adopt new ideas and practices, or not.
- **The Hole-in-the-Floor Model**, which says change is dropped on changees by planners upstairs. Process improvement committees and “You will now be Agile” mandates fall into this category.
- **The Newtonian Model**, which introduces the concept of external motivation to change. Often this takes the form of upper managers creating a sense of urgency, walking around with gantt charts checking to make sure every thing’s reported on schedule, blaming resistors and beating up the laggards.
- **The Learning Curve Model**, which considers the time to adapt to something new. On the job training and mentoring fall into this category.

Other models exist, but often they fall into one of the above categories.

A Little More About Organizations

We can consider an [organization](#)⁶ as “an administrative and functional structure (as a business or a political party)”.

This means we change an organization by re-arranging its structure. We add a new department (Quality Control), insert a new role (Director of Quality), reassign people, update the HR manuals, and re-draw the development process. Implement the new structure and we're done! Sort of. If the change doesn't seem to work fast enough, we can increase the pressure or put motivational posters on the wall.

But What About the People?

While organizations can change at the drop of a new org chart, people can't. We miss the old familiar way, aren't really sure what's going on, and where this might all finish. In [Managing Transitions, Making the Most of Change](#)⁷ William Bridges identifies these three zones as the Ending, the Neutral Zone, and the New Beginning.

- **Ending** - Letting go of the old ways and the old identity people had. This first phase of transition is an ending, and the time when you need to help people to deal with their losses.
- **Neutral Zone** - Going to an in-between time when the old is gone, but the new isn't fully operational. We call this time the “neutral zone”: it's when critical psychological realignments and repatterning take place.
- **New Beginning** - Coming out of transition and making a new beginning. This is when people develop the new identity, experience the new energy, and discover the new sense of purpose that make the change begin to work.

It looks like this:

⁶<http://www.merriam-webster.com/dictionary/organization>

⁷<http://www.amazon.com/Managing-Transitions-Making-Most-Change/dp/0738213802/>

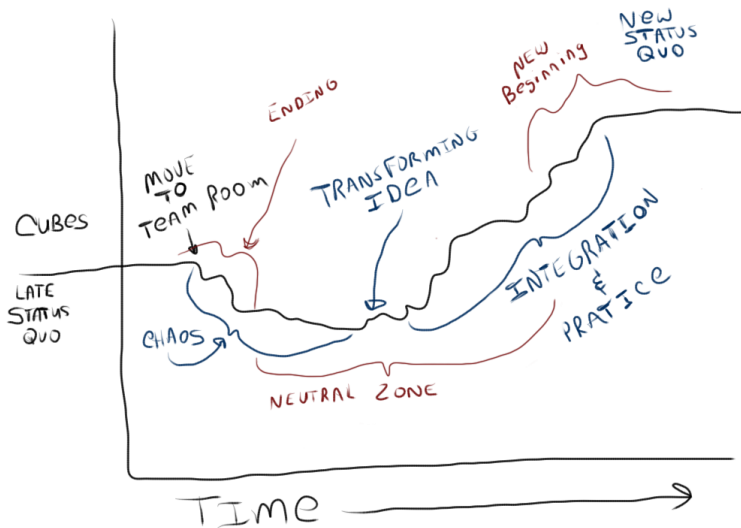


Figure 3 - Transition and Change

Using our previous example of moving a cross functional development team into a team room:

Moving to the team room represents the **change**. Make the announcement and move. I've seen teams pick up their computers, manuals and move. I've worked where "facilities" moved the equipment and books over the weekend. Either move has a discrete start and end. The change announcement starts the **transition** process.

When team members hear they're moving to a team room, they wonder: "What will it be like?", "How will this affect my ability to concentrate?", "If I play along, how long will it take before we move back into cubes?"

After the equipment arrives and they're sitting in the team room, they start to adapt to their new surroundings. They may establish times when the team tries to stay quiet. Two or three may go to the whiteboard and have a conversation about a piece of code. They develop simple rules for handling interruptions.

Eventually the team room becomes the norm. Team members become comfortable with the change to the team room. I've seen those most opposed to moving become ardent supporters for the change.

Handy Reminders

Companies *change*

People *transition*.

Not everyone in the organization completes the transition started by the change.

Those who do make the transition from ending to new beginning do so *at different rates*.

"You need all three phases, and in that order for a transition to work. The **phases don't happen separately; they often go on at the same time**. ... Perhaps it would be more accurate to think of them as three *processes* and to say that the transition cannot be completed until all three have taken place."³

Constructive Chaos - The Neutral Zone

If you're shifting software development paradigms, use a clutch!

Team Powerhouse had problems. The developers didn't know how much work they could get done in a sprint. The testers received the code about a day before the sprint review. About half the stories got pushed to the next sprint. All in all, it seemed normal to me. They had just formed, and some chaos seemed reasonable.

Say What?

When organizations change, the old way of working doesn't work any more. Work flows change. People get put in **new groups**. **Agreement hasn't formed** on what we're going to do, much less on **how we're going to do it**.

Chaos reigns. We're far from doing and agreeing. In a picture it looks like:

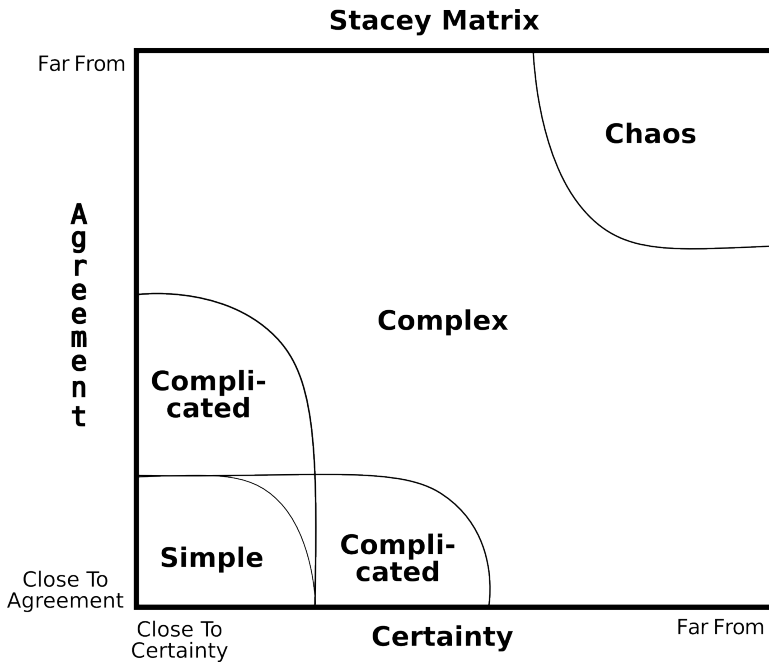


Figure 4 - Simple to Chaos

How Does This Apply to Software Development?

You can put almost any two inter-related labels on the axes. Common ones I use in software development include “**Requirements/Technology**” and “**What/How**”. The closer to agreement you can get, the less chaos exists.

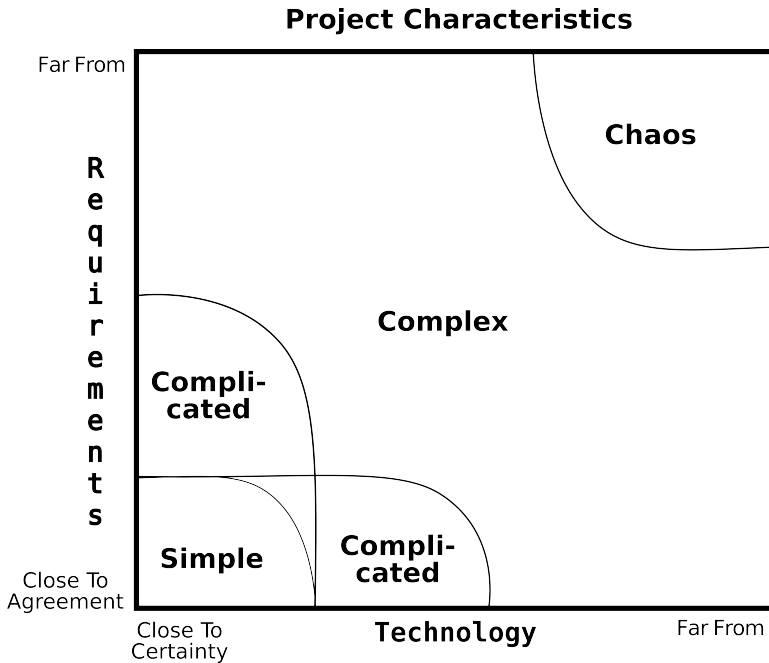


Figure 5 - Project Categories

While Chaos may seem bad, it can provide energy and freedom to innovate if the organization has created [safety](http://www.donaldegray.com/tag/safety/)⁸ around the **change**.

We need to recognize, understand and not be surprised by the neutral zone for several reasons.

First, if you don't understand and expect it, **you're more likely to try to rush through or even bypass** the neutral zone. And to be discouraged when you find that doesn't work.

Second, you may be **frightened in this no-man's-land** and try to escape. ... To abandon the situation, however, is to abort a transition, both personally and organizationally - and to **jeopardize the change**.

Third, if you escape prematurely from the neutral zone, you'll not

⁸<http://www.donaldegray.com/tag/safety/>

only compromise the change but also lose a great opportunity. ... so here let me simply say that **the gap between the old and the new is the time when innovation is most possible and when the organization can most easily be revitalized**

The neutral zone is thus both a dangerous and an opportune place, and it is the very core of the transition process.

So What to Do?

- **Understand** your employees will experience some chaos during the neutral zone.
- Chaos **will initially lower productivity**⁹, even though the change will (hopefully) improve productivity in the future.
- **Create safety** so employees can **explore and innovate** without fear during Chaos.
- Work with a **Change Artist** to plan the change, and then follow on as people transition.

Organizational Changes Make Messes

As we've seen, people approach and respond differently to change. Change starts getting messy when **our change model doesn't map to the reality we deal with.**

Three Types of Change

Human Systems Dynamics¹⁰ posits three types of change: **Static**, **Dynamic**, and **Dynamical**.

I've summarized the differences in the types of change in this table. The descriptions come from **HSD**. I added the equations (for us math/engineer types) and the graphs of what the change might look like.

⁹<https://www.donaldegray.com/developing-developer-skills/>

¹⁰<http://www.hsdinstitute.org/index.html>

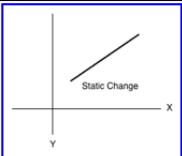
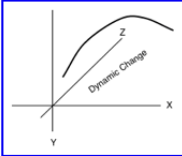
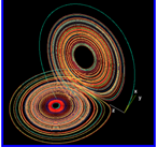
Name	Description	Equation(s)	Looks Like
Static	Static change is the simplest, being two-dimensional, it depends on direction and force. It is also predictable. Static change is about moving from Point A to Point B by applying force.	$ax^n+ax^{n-1}+...+c$	
Dynamic	Dynamic change is more complicated, being multi-dimensional. It can best be described as moving along a smooth trajectory toward a predictable end point. Like water shooting out of a hose, if pressure and angle are known, you can predict height and distance of the arc of water.	$ax^nb^y^m+ax^{(n-1)}by^{(m-1)}+...+c$	
Dynamical	Dynamical change is complex and results from multiple forces acting in unpredictable ways, generating surprising outcomes. Think about water dripping from a faucet. The rate of drops depend on too many factors to to predict, precisely, when each drop will fall. The amount of deposit in the pipes; the temperature, wind, and humidity in the room; and the amount of water in the pipe interact in unpredictable ways to determine when drops will fall—non-predictable and maddening in the middle of the night.	$\frac{dx}{dt} = \sigma (y - x)$ $\frac{dy}{dt} = x (\rho - z) - y$ $\frac{dz}{dt} = x y - \beta z$	

Figure 6 - Change Categories

Let The Mess Begin!

So what happens? Some executive, somewhere, **decides things need to change!** People get assigned to new departments or teams. A new reporting structure gets defined. The charts are drawn, powerpoint slides created, and “all hands” sessions scheduled. What model is management using? The mess starts shortly after the meetings.

In my experience managers usually expect static change behavior. Start here. Go there. Done. You’re in a new team. We moved your equipment and materials to the new space. What’s the problem? Other than everyone having to work through the **the Change Model?**

Some people will work through the **Change Model** quickly. But on many levels we are dealing with very personal deep seated stuff... the stuff that anchors us as people and defines who we are in relation to the world.

This means other people will respond in a **Dynamical** fashion.

And **hence the mess**. The mismatch between the change response expected and the change response in the domain.

Productivity goes lower than it needed to. People get labelled “resistors”. **Malicious compliance** might occur. **Management pushes harder** to make sure the change happens.

If the wheels don’t spin completely off the change, it may get sidetracked. Sometimes management declares success and moves to the next hot topic.

As a **change artist**, I’m pro-change. Change includes planning.

As a **systems thinker**, I’m pro-models. Models are neither naughty nor nice. They’re the simplification of the world we carry in our head that enable us to function.

As an **engineer**, I’m pro-reality. In this case I define reality as using data to determine if our models reflect what’s happening the domain.

What to Do? If you’re planning or involved with a change:

- Remember everyone goes through the **Change Model** at different rates, but everyone goes through.
- Be prepared for the **Failure Paths**.
- Check your assumptions on the **type of change** you’re working with.

¹Quality Software Management Vol 4 Anticipating Change, © 1997 Gerald M. Weinberg, ISBN 0-932633-32-3 pp 60-61 If you prefer ebooks Becoming a Change Artist contains this material and more

²You can read more about these models in Quality Software Management Vol 4 Anticipating Change, © 1997 Gerald M. Weinberg,

ISBN 0-932633-32-3 pp 3-14 If you prefer ebooks [Becoming a Change Artist](#)¹¹ contains this material and more

³Managing Transitions, Making the Most of Change, ©2003 William Bridges and Associates, ISBN 978-0-7382-0824-4 page 9

⁴ Ibid., 8-9

The equations for the Lorenz attractor (the picture for dynamical change) came from https://en.wikipedia.org/wiki/Lorenz_attractor.

The picture of the Lorenz Attractor is from Wikimedia commons - [Lorenz_Ro28-200px.png](#)

¹¹<http://www.smashwords.com/books/view/47707?ref=JerryWeinberg>">
[Becoming a Change Artist](#)

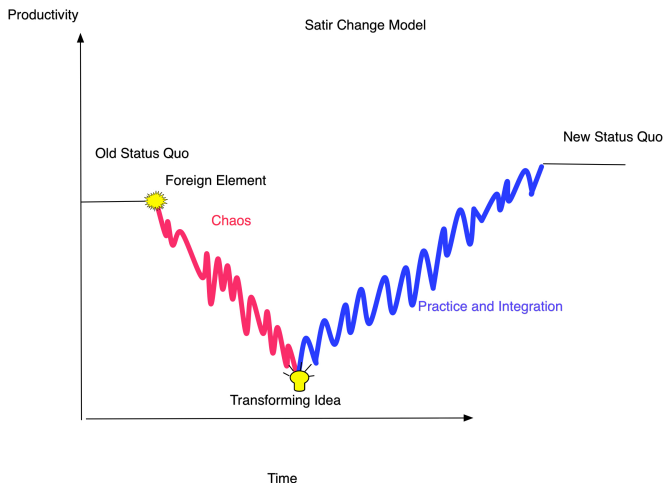
Change is Inevitable

©2011 Johanna Rothman

It's funny. In my consulting work, I meet a lot of people who think that change never happens in their organizations. Or they say, "Only death and taxes are sure things."

But I know that change is the only sure thing in life. We change from the time we are born until we die. Sometimes—indeed, often—we instigate these changes. Every time I go to the gym, I'm instigating a change that I want. If you're thinking of changing how you develop or test or manage, you're instigating change. Sometimes change happens to you. But change is inevitable.

We go through a process of change. Virginia Satir captured this with her change model.



We start at a certain level of capability in Old Status Quo until we encounter a Foreign Element. If we want that change, such as a new job, we've created our own foreign element. In my case, my inner ear hemorrhage was my foreign element. Then, we stay in Chaos for a while, until we get a Transforming Idea. Sometimes, we stay in Chaos for seconds. Sometimes, we stay in Chaos for months or years.

Once we find that Transforming idea, we move on to Practice and Integration, and then finally, we reach a New Status Quo, where we are at a new level of capability.

For those of you who are introducing agile to your organizations, or are introducing some other organizational change—this is why organizational change is so hard and so expensive. Every single person goes through the change model at his or her own pace. And, every single person must go through the change model. If something happens during chaos to shove the person back into Old Status Quo, well, you're back at the beginning. Good luck.

Change happens all the time. Sometimes we don't notice. Sometimes we adapt without thinking too hard, because we see a transforming idea and it makes so much sense to us that we move directly into practice and integration and the new status quo. And, sometimes, change hits us with a 2x4. "Hey, you paying attention?" Uh, I am now!

The Satir Change Model: Interventions

The Satir Model does not say that change will always follow the five stages. The change process can be short-circuited at any stage, or a stage may be prolonged or shortened. Thus, if a change artist wants to help change to happen successfully, interventions may be needed.

The change artist who would be helpful must learn to work in the following ways:

- to listen actively
- to respond honestly and clearly
- to set clear boundaries between themselves and others (their pain, not mine)
- be willing and able to let them struggle with their own chaos
- by activating possibilities in each person.

The type of intervention needed at each stage is different. Some change artists are good at only one stage, and shouldn't be used in others:

Status quo: Use ideas or measurements to provoke.

Foreign element: Use persistence to overcome avoidance and denial.

Chaos: Help to avoid any long-term decisions or easy escape back to status quo.

Integration: Support, hold hands, teach (but not too much). Provide opportunities to practice in safe environment.

New Status Quo: Provide steady stream of well-timed little foreign elements, not too often, not too seldom, but just right.

Change and Temperament

Different temperaments tend to have different styles of coping with change.

NT The Visionary

- Likes: Working with ideas, designing change.
- Concerns: Change must be optimal and fair.
- Problems: Loses interest at implementation time. Can't distinguish designing from doing.

NF The Catalyst

- Likes: Working with people to help them grow.
- Concerns: People will not suffer from change.
- Problems: Wants everyone to do the same thing. Wants everyone to do it at the same time.

SJ The Organizer/Implementor

- Likes: Order and system, getting it done right.
- Concerns: Change can be messy and inefficient.
- Problems: May fear quick change. May require advance proof of success.

SP The Trouble-Shooter/Negotiator

- Likes: Quick fixes, getting it done.
- Concerns: Getting stuck with something boring.
- Problems: May provoke change for its own sake. May drop out if change is too slow.

Learning by Design: Constructing Experiential Learning Programs

Daniela Weinberg and Gerald M. Weinberg Weinberg and Weinberg

How can we assure that our students really learn? If we want to improve the learning process, we'll have to decide what we mean by "learn". Recall a time when you learned something new – a new skill, a new technique, a new word, a new hobby, a new way of interacting with people – and consider the following three questions:

First, why did you bother? You knew that learning would change you, and that change meant stress. Why would anyone volunteer for additional stress?

Second, how did you feel during the learning process? Was it difficult or easy? Was it a delight or a bore?

Third, what was the outcome? Did your behavior change? Did it stay changed or did it fade? Would you consider it a successful learning experience?

A Typical Learning Experience

A typical learning experience we observed was Dani's learning how to use a word processor. Let's consider Dani's answers to the three questions. Why bother? Jerry and his secretary, Judy, made a quick transition from the typewriter to the new machine. Dany observed the process and felt curious about this new addition to the family,

as well as increasingly excluded by the new language around the office. Not to be conquered by a glorified typewriter, she asked for instruction.

How did you feel? Dani's first reactions to the new technology were mostly negative. The keyboard was not laid out exactly like a typewriter's, nor did it have the same "feel." There was no carriage return, the familiar typing sound was absent, and the small video screen had a nasty habit of displaying cryptic error messages such as:

62, FILE NOT FOUND, 00, 00

And with all her intelligence, Dani could not seem to memorize the simplest formatting commands but had to look up everything in the manual, over and over. No question about it, this was a stressful situation that made her feel lost, dumb, and angry!

What was the outcome? With time and practice, Dani's irritation gradually gave way to excitement, which she noticed when she found herself actually looking forward to the next session at the machine. She knew she had mastered it when she realized that she no longer referred to the manual, or even to her little sheaf of penciled notes.

Piaget's Model of the Learning Process

How can we explain this process of learning? And, once explained, can we generalize the model to apply to other kinds of learning?

Jean Piaget, the Swiss philosopher and psychologist, proposed that learning was a process of constructing – not receiving – new knowledge. The learner could not simply sit back passively, and obtain learning through the manipulations of another person – a "teacher." The learner was neither an "empty vessel" to be filled

with knowledge or a “blank slate” on which knowledge could be inscribed, but a system that learned through its own interaction with its environment.

What we like best about Piaget’s model is the way it makes the learner an active participant in the learning process. The process begins when the learning is provoked – the learner examines his or her existing cognitive models because of some pressure from the outside environment. It might be provoked by necessity, as when a businesswoman struggles to learn Japanese to retain a major overseas client. It might be simple curiosity or, as in Dani’s case, a feeling of being left out of the group. Whatever the case, learning doesn’t just happen. Learning is the result of decision and action by the learner. (And not learning can also be a conscious decision and action.)

Once the decision is made, the learner plunges into an unfamiliar sea filled with unknown flora and fauna. At first, nothing fits or even makes sense in the learner’s vision of the world. The world seems filled with cryptic messages. Piaget called this stage disequilibrium. The learner is literally off balance in the new environment.

What to do? From the disequilibrium state, we may try to restore equilibrium without learning by changing the environment. When Dani said “this is nothing more than a glorified typewriter,” she was bending the environment so she could retain her own model of the world. Piaget called this process assimilation, which was one pole on a continuum of choices we can make to restore equilibrium.

At the other pole, we can throw out our old model and embrace the new one totally. Piaget called this choice accommodation. Jerry was closer to the accommodation pole because he could no longer type competently on the typewriter without ruining 50 sheets of paper.

Generally, neither pole by itself provides a satisfactory outcome. Instead, we move between these poles in a dynamic equilibrium that Piaget called self-regulation. We partially surrender our earlier

models of the world, and at the same time we twist the world into their static shape. In the process, we build for ourselves a new model of reality that somehow reconciles the old and the new. Although Dani no longer calls the word processor a typewriter, she still needs a printout and a red pencil before she can “really” edit her work.

When we complete the self-regulation process, we have constructed new knowledge. And then the process begins anew – moving from a state of comfort, into a provocation, through a state of discomfort, and finally into a new region of comfort.

The Learning Cycle

As trainers and educators, we can use the Piaget model to design teaching strategies that allow this process to develop. We must first put our students into a provocative environment. We must encourage them to experiment – to play with the materials in that environment. We must challenge some of their preexisting notions of the world – to provoke disequilibrium, but without utterly frightening them away. We must support and guide them through the self-regulation process. And finally, we must provide opportunities for them to consolidate and feel comfortable about their newly-constructed knowledge.

The teaching design we use to accomplish all this is called the learning cycle. The original learning cycle was developed by Robert Karplus of the University of California at Berkeley to assist students in the development of logical thought. Robert Fuller worked with Karplus at Berkeley and brought the learning cycle idea back to the University of Nebraska, where it was modified for college instruction by the ADAPT faculty. In our own work, we have adapted the learning cycle for adult learners. In all its variations, the learning cycle has three phases, called exploration, invention, and application.

The Exploration Phase

In the exploration phase, students – usually working in small groups – are permitted to explore a new environment on their own, with minimal intervention on the part of the instructor. The exploration phase is the learner's personal encounter with the new material.

Working with these materials, the students attempt to complete an apparently simple task. They soon find, however, that there is some fly in the ointment. Resolving the difficulty requires even more interaction with the materials, as well as active discussion with peers. Eventually, the students complete the task and possess a set of data about the new environment.

Here's an example of an exploration taken from one of our workshops for systems analysts. We want the students to learn a new way of designing questions, one that will be effective at extracting information from users in the fuzzy early stages of the system development process. We give them a work order for a new system, with instructions to prepare a list of questions for the user. The user is played by one of the instructors, who carefully answers their questions without volunteering anything not covered by their questions. The students are aware of some difficulties as they interview the user, but other difficulties become apparent only after they are given a complete list of all the information they might have obtained if their questions had been better framed.

The instructor's principal role in the exploration phase is designing a task that will force the students to call their existing models into question. Sometimes these tasks work too well, so the instructor may have to encourage a student who feels unable to proceed, but this is rare. Sometimes the instructor plays a role, as in our fuzzy-question simulation. Mostly, however, the instructor simply observes the explorations, gathering data to be used in the next phase, invention.

Invention

The invention phase brings the entire group together, with the instructor playing a leading role. The group tries to make sense of the data generated during the exploration phase. They are now working in an analytical mode, trying to generalize from their data by inventing new concepts or tools. The instructor may provide the standard technical terms for these “inventions,” or present a model in current use, or even deliver a mini-lecture to illuminate or integrate the students’ inventions.

For instance, in our fuzzy question simulation, the instructor might guide the discussion by listing students’ examples of information that their questions failed to elicit. Alongside each piece of information, the students write questions that might have succeeded. The instructor then asks the students to identify systematic differences between the successful and unsuccessful questions. Out of ten principles that the instructor had in mind, the students might develop seven on their own, get two more with tiny hints from the instructor, and need a bit of a push to catch the last. The instructor might then finish the invention by relating all ten principles to a model based on information theory.

Regardless of the instructor’s contribution, the knowledge that emerges has been constructed by the students themselves, rather than provided by the instructor. The instructor may have provided conventional names, or accelerated the convergence of invention, but has in no sense filled the empty student vessels with knowledge. That is done by the student’s own active involvement.

Application

The application phase completes the learning cycle by creating the opportunity for the students to interact with the world once again –

this time using their newly acquired models. As in the exploration phase, the instructor merely provides a structure and observes the students working in their small groups. In our fuzzy-question learning cycle, the application involves a work order for another system, more or less repeating the structure of the exploration phase.

In many cases, the application phase for one learning cycle becomes transformed into the exploration phase of the next. The fuzzy-question learning cycle is actually a series of case studies. In each case, the students progress a bit further by applying their learnings from the previous case – then run into trouble on the next type of difficulty. Such a series of linked learning cycles makes learning more efficient, but, more important, integrates different conceptual models.

Experience with the Learning Cycle

We have used the learning cycle approach in a great variety of educational settings, ranging from undergraduate college courses to intensive residential workshops for technical leaders in industry. Whenever principles are to be learned, rather than simple procedures, the learning cycle is appropriate and effective.

Virtually any subject can be learned using the learning cycle approach. Our experience, and the experience of our colleagues, has included anthropology, computer science, management, English, systems analysis, philosophy, logic, economics, communication skills, and leadership training.

We have used the learning cycle effectively in groups ranging from 5 to 150. Learning cycles may be designed for a variety of available resources, including what the students carry around in their heads. A learning cycle may be built into a one-hour class meeting or planned to run for a full 16-week semester or a 6-day workshop.

It may be integrated with readings, films, field trips, laboratory experiments, guest experts, and even lectures.

Learning versus Teaching

Piaget's model highlights the difference between learning and teaching. In other educational approaches, the "teacher" acts on the "student," hoping to transfer knowledge from one to the other. As one wag described the lecture method, "It's a way of getting material from the teacher's notes into the student's notes – without passing through the brain of either one." Not all lectures for all students may share this characteristic, but it's certainly a frequently observed phenomenon – one that's simply not possible with the learning cycle. In the learning cycle approach, the material must pass through the student's brain because the student acts on and interacts with the material to be learned – transforming the material rather than mechanically copying it.

In the learning cycle, the student has responsibility for her or his own learning, which is virtually a requirement for success with adult learners. And through this responsibility, all students, whatever their age, develop a greater sense of ownership of the products of the learning process. Not only is understanding of the material much deeper, but retention is far better than in other approaches. Experiments show that most of what is taught is rapidly forgotten, but what is learned stays learned – until it's unlearned in some new disequilibrating environment.

References

SCIS, Teacher's Handbook, 1974 Lawrence Hall of Science University of California Berkeley, CA 94720 This is the original publication containing the idea of the learning cycle as developed by Karplus.

Piagetian Programs in Higher Education, 6th printing, 1982 ADAPT Program

Climbing Out of Technical Debt

© 2002 Johanna Rothman

Have you ever had a conversation like this one?

Vice President: In the last release, you were able to bring the release date by over a month by cutting the testing. Do that again, ok?

Project Manager: Sorry, boss, we can't do that. For the past three releases we've shortchanged the design work and the testing. We can't cut any time from this project. In fact, it's time to re-architect the system, redesign it, and do the testing we've put off.

Vice President: NO WAY!

Project Manager: Hey, that's just the way it is. We haven't taken the time to finish this product yet, and our shortcuts have come home to roost. Our technical debt is too high to work this release the way we've worked the others.

Technical debt, as defined by my colleague Dave Smith, is the debt a company “owes” to a product they persisted in shipping in an incomplete or unstable condition. This is a situation referred to in Hunt and Thomas’s book *The Pragmatic Programmer* as “software rot” or by Gerald M. Weinberg in *Quality Software*

Management, Volume 4 as “design maintenance debt.” The software isn’t necessarily rotten—the product is incomplete.

As the technical debt increases, the load on the customer support staff becomes overwhelming, and the developers have trouble adding or changing system features. When developers are stuck on new development and support is overloaded, you face difficult choices about what to do now: have developers support the current product; ignore the current product and continue with development; some combination of support and development; or stop developing and fix what you’ve already got.

Project managers, development managers and test managers often can see this coming. They know there is a window of opportunity to fix the product before the technical debt overwhelms the company’s ability to do new product development.

Look for these signs of technical debt:

- You ship a product and then have to put out a point release before it’s even left the CD duplication house.
- Testing time increases disproportionately to the development time.
- The developers tell you that part of the product needs to be re-architected, because the current architecture doesn’t support the current requirements, never mind the additional requirements.
- Developers refuse to touch a part of the product saying, “No one but Fred can touch that. I know Fred left three years ago, but no one else can work on it, because we don’t understand it.”
- Developers spend more time supporting current customers by solving customer problems, fixing defects, and answering customer questions, than they spend developing new product.

- You hire more testers than you have developers, and you're still not sure you've tested the product.
- Your developers threaten to leave because all they do is "maintenance".
- You ship the product not because it's ready, but because the developers' families have abducted them to go on vacation, or the developers are too exhausted to continue the crunch mode you've been in.
- You stop testing because you don't want to find more defects.
- Your cost to fix a defect continues to increase, from release to release. (If you'd like to see a picture of this dynamic, check out Weinberg's *Quality Software Management*, Volume 4, reference below.

Recognize your technical debt, and start planning what to do. Before you can plan, you need to know a little about how you came to have this technical debt problem. These questions may help you understand how your product acquired its technical debt:

How do you staff projects? Do you assign enough people at the time they are needed, or are some positions never assigned or assigned late to a project? If you never assign enough testers, or you don't have an architect, you will incur technical debt. Don't blindly accept people starting on your projects late, because you can't make up time in a project. Have people start when they're supposed to start, or recalculate the project schedule.

How do you design your projects? Do you first architect the entire product for its entire lifetime? Do you iterate on the design for several projects? If you don't architect the product, and update the architecture as necessary, or if you don't do a high level design before starting to code, you will not have a congruous product. One way to recognize an incongruous product is when you have multiple ways of doing the same thing. At one organization, the developers couldn't agree on how to open and save data files. Since

they couldn't agree, every time a developer wanted to open or save their data file, they had their own open and save functions. When they moved from a command line interface (CLI) to a GUI interface, they had no easy way to integrate the open and save functionality.

How do you test the product? Do you have a way to start testing the product at the beginning of the project? Do you only have manual tests through the graphical user interface? If you start limited testing late, you're guaranteeing yourself technical debt. You have many opportunities to test early and often: requirements inspections, design reviews, code inspection or peer review, unit tests, daily builds and smoke tests (tests to verify the daily build works at least a little bit), peer review of fixes in addition to system test. Decide what makes sense for your environment, and add test capability as early in the project as possible.

How do you organize your projects? Do you have functional managers who hand off the product from the analysts to the developers to the testers to the manufacturing folks? If so, no one has the same view of what the product should be. At a minimum, assign an overall project manager or program manager, so someone is pulling the entire project together. Projects without project managers are highly risky, and in my experience, always fail. In addition, projects without a sponsor or champion have a difficult time finishing.

How do you know what done means? Do you have some objective criteria to know when the project is complete? If not, you're more likely to release an incomplete product. Develop release criteria for every project, so that everyone on the project, and the project sponsor understands what your company will receive for its project money.

How do you run your projects? Do you stick to only one type of lifecycle and one process? You may need to choose other lifecycles and/or processes to avoid future technical debt. Many project managers are comfortable with a particular lifecycle and/or process, and it can be difficult for those project managers to switch to a

different lifecycle for a project. If you're used to a waterfall lifecycle, you'll have to learn to plan for an iterative or incremental lifecycle differently. And, you'll have to monitor your projects differently.

If you've changed your lifecycle, you may need to change your process. What you do and when you do it will make the lifecycle successful and keep down technical debt. If you've never planned when to do inspections and reviews, start with something small, such as peer review of defect fixes during system test. I've found that a nightly build and smoke test helps find defects faster, and helps everyone realize when the product is not ready to ship.

Although it feels as if you're moving faster when you take shortcuts on a project, in reality, your technical debt prevents you from making project progress. You might be able to shortchange your product on one or two releases, but at some point your product technical debt is like credit card debt — if you don't pay it off in a planned way, the debt buries you.

Observe and measure your products, to know when you should act to reduce or avoid technical debt.

For more information, see

Hunt, Andrew and David Thomas, *The Pragmatic Programmer*. Addison-Wesley, Boston, MA 1999.

Weinberg, Gerald M., *Quality Software Management, Volume 4*. Dorset House, New York, 1998.

Seven Lessons from a Top Down Change

© 2007 Esther Derby

You'd think that since I'm president of a one-person company, I could change anything in my office in a snap. But a recent incident reminded me that change is always a process.

My dog, Pudge, comes to the office with me every day. Until recently, she spent her day on a blanket by the door where she chewed her on her bone between naps. When she wasn't chewing or napping, she tugged the blanket around the floor. But this day, I'd stepped on the blanket and slipped one too many times—and frankly, the blanket was looking ratty!

So I decided to replace it.

Lesson 1: A change usually represents someone's best idea on how to solve a problem or respond to an external event.

I decided that a cushy new dog bed would solve several problems at once. It would stay in one place, look better, and provide a more comfortable nest for Pudge.

That's usually the case with a top-down change. Someone is offering the best solution they know at the time to improve the situation. No matter what it looks like from below, people who initiate change hope and intend to improve the situation. But unless managers follow the rest of these lessons, it's unlikely that the people who are asked to change will appreciate this truth.

Lesson 2: One person's carefully considered idea is another person's incomprehensible surprise.

When the new dog bed arrived, Pudge was curious about the box. She was even curious about its contents. But when I replaced her old blanket with the new bed, her curiosity ended. She picked up her bone and headed into the hall for a good chew. Then, one by one, she moved the rest of her toys out into the hall.

The people charged with formulating new strategies spend time thinking through the issues, economics, benefits, and risks. By the time they decide to implement a change, they have a thorough understanding. They've worked through questions, risks, and their own objections. And then they forget that other people have not had as much time to mull over the issues and examine the pros, cons, and imperatives.

It is very important for people to understand the thinking behind a change. They need information and time to digest it.

Lesson 3: The people seeking a change may value different things than the people expected to change.

When I bought a new dog bed, I was looking for neatness, containment, and easy washing. Although Pudge can't tell me, I suspect that she valued the ability to tug the blanket around and actually enjoy its odor (she's a dog, after all!).

When a mid-western company was purchased by a larger firm based in New York, headquarters assumed that the acquired company would appreciate new opportunities for advancement and travel. They were sure the mid-westerners would find their direct approach a breath of fresh air. But the mid-westerners valued balancing work and family life. The mid-westerners believed that being considerate was part of maintaining long-term working relationships. To them, the New Yorkers' directness seemed disrespectful and rude.

People who ask others to change may not understand what value they are asking people to give up. And in fact, they may not appreciate or even notice what's valuable to the people they expect

to change.

Lesson 4: Change always involves loss.

I had every intention of providing something wonderful for Pudge when I bought her a new dog bed. But she didn't see it that way. She lost her familiar blanket.

Even when there is long-term gain, the people affected by change will experience loss. They may lose established routines, patterns, and relationships. They may lose prestige, their sense of competence, or belonging. When you plan a change, think about who is benefiting and who is losing as a result of the change, and then be prepared to empathize with people's sense of loss.

Lesson 5: Choice or coercion depends on where you sit.

I chose to change my dog's blanket for a new dog bed. She didn't have a choice—she wasn't even consulted! "She's a dog!" you may say. That's true. Too often, though, intelligent adults are treated the same way when it comes to organizational changes.

People don't resist change. Change that's presented as "Follow, or be fired!" feels like coercion. And most people resist coercion.

Along with providing information, engage the affected people in designing the change. Engagement increases support. People will still feel a sense of loss, yet they are less likely to feel like victims when they have information and some input in shaping the change. And the people closest to the work are likely to have ideas that will improve the implementation—if you bother to ask them.

Lesson 6: How people respond to a change is a rich source of information.

When people don't do what we want them to do, change management professionals call it "resistance." Resistance is something to overcome. If we look at it another way, how people respond to change is a source of information to refine ideas and shape implementation.

When people don't do as we want or expect, it may be because:

- they don't know how to do what they're being asked to do.
- they don't feel they have time to learn the new way and still meet existing goals and targets.
- they believe the existing way is better.
- they don't think the new way will work.
- they believe the new way will cause harm—to customers, the company, the employees, etc.
- they don't like or don't respect the person requesting the change.
- the new suggestion is counter-intuitive given people's existing models of how the world works.
- the new way runs counter to existing reward structures or other organizational systems.

Once you learn about the reasons behind “resistance,” use that information to adapt the change, provide needed training, or review potential roadblocks. Acting on the source of the response increases the chances the change will succeed.

Lesson 7: People change to retain something they value. The second day after I introduced the new bed, Pudge followed me to the office but stopped at the door. I enticed her onto the new bed with a biscuit. She stepped gingerly onto the bed, picked up the biscuit and left. We repeated this routine for three days. Finally, after four days spent alone in the hall, she came into the office—not because she liked the bed, but because she wanted to be close to her human.

That's true for people, too. Most people don't change because there's a good argument for a change, or even when there's overwhelming data supporting the reasons to change (witness the number of people who still smoke cigarettes). People change to retain something they value. It may be an association with people or

the organization. It may be a steady paycheck. Take time to find out what people value and how the change relates to what they value.

So what does my experience with Pudge and her new dog bed teach us about change? Lack of enthusiasm for a change proposal may not mean it's a bad idea. But it's a good indication that there's work left to be done in introducing the idea, understanding what people value, and incorporating their ideas.

This article originally appeared in insights Vol. 5 No 1.

Part 3: Observing Yourself and Others

This Title May Change at Any Time! (How Do You Feel About That?)

Beyond Belief

Seeing the Other Person's Big Picture

Seeing Your Own Big Picture

This Title May Change at Any Time! (How Do You Feel About That?)

©2010 Don Gray

Three of my favorite quotes about change and translations:

“The only person who likes change is a wet baby.” This change corrects a problem so I’m OK with it.

“Everyone likes change, when someone else is doing it.” I’m OK with the status quo. You do the changing.

“The only universal constant is change.” Get used to changing. The universe constantly changes so you need to change too.

If change is so inevitable, why do people, teams and organizations experience so much difficulty changing? We should be good at it!

In fact, we are good at change, but in our own particular way. Any given change carries two factors. The requested change, and how we feel about that change. Recently while cooking, I grabbed the handle of a pan that had been in a 350° oven. It didn’t take long before I let go of the handle. I had no trouble accepting the need for this change.

Not all changes, though, are as universally accepted as dropping the handle of a hot pan. Suppose everyone were required to shave their head. This change wouldn’t bother me and I’d readily shave my head. Johanna Rothman assures me she’d have a big problem with this change and be very slow to change. This typifies “You go ahead and change. I’m happy with the status quo.”

CHANGE QUOTIENTS

If we ignore neurologically requested changes (like letting go of the hot pan handle), we can view the ratio of the change/willingness to change as a quotient. As we average all the quotients over many situations for one individual, we arrive at a statistical value I'll call that person's Change Quotient. The Change Quotient relates to how open a person is to change.

Change Quotients vary from 0 to 1. Values close to 0 indicate a preference for slow changes and values approaching 1 indicate people who readily change. We each have a unique change quotient. 0 doesn't mean "bad", and 1 doesn't mean "good". The values simply indicate a preference.

People with lower change quotients often view those with high change quotients as chaotic and "flighty". I was once asked to "make up my mind" when in fact I'd already made my mind up and shared the results twice while the other person was still thinking.

Conversely, "glacial" describes low change quotients to those having change quotients approaching 1. These people seem to reject instantly any suggestion to change, even something as simple as removing their boot from your bare toes.

Several things contribute to our Change Quotient. Any proposed change runs a gauntlet of survival rules, self-esteem, and personality/family checklists. The gauntlet forms our "willingness to change."

Survival rules perform as "mental reflex actions", the things we do, the thoughts we think, and feelings we have, that we don't consciously think about. We build most of our survival rules by repeated assertion, usually by our parents or respected adults, of things to keep us alive. "Never make a decision until you have all the facts." "Don't be wishy-washy—make up your mind." "Be your own man—don't let others push you around."

Self-esteem reflects how we currently feel about ourselves. This fluctuates based on our physical state (rested/tired, healthy/sick), our mental state (happy/sad, excited/bored) and how we feel others

view us.

Our family creates our initial world view. Some families view consistency and slowness to change as admirable qualities. Others feel adaptability and responsiveness are desirable attributes. Each one of us starts with our family's view of change.

Personality builds on this foundation. We add our experiences, successes and failures, to build new rules about change. Which kinds of change we can accommodate (or welcome) quickly, which changes we need to consider, which changes we want to avoid.

RESPONDING TO CHANGE

Some years ago I found out that I was not really in charge of everything, and actually in control of very little. This led me to

Don's Dismal Dilemma: *How will I achieve my goals, when I'm not in charge or control?*

Physical systems follow patterns established eons ago. Friends do what suits them. My clients determine when and where I'll work with them.

As time progressed, I found the one thing in the universe I have complete control over: me. And so I found

Don's Delightful Discovery: *I can't control what happens to me, but I can control me, and how I respond to what happens to me.*

As Virginia Satir said, "We can direct our efforts to change what we can and to work out creative ways to live with what we can't change." [New Peoplemaking, pg 7]

When we quit changing with our environment, we face extinction. We need to update our mental models so they conform to current reality. As we grow and mature, our realities change. Our mental models need to reflect the changes. Mental models formed in childhood and not updated make for interesting adult behavior. Dad warned me about getting "Hardening of the Attitudes". (Notice the survival rule that affects my change quotient?)

Changing allows me to achieve my goals. If the goal doesn't change, and external events (by definition beyond my control) change the situation, I need to change my actions (I control these) to bring the results in line with achieving my goal. Or perhaps my actions aren't producing the results needed to achieve the goal. Again change becomes necessary to realign with my goal. If I don't change what I'm doing, my actions will take me away from my goal.

***“Insanity: doing the same thing over and over again and expecting different results.”** *Albert Einstein*

Occasionally I need to change my goals. As I learn more, what used to be important may not be important now. Changing goals allows me to work on what's important to me.

So what to do? The answer depends if you are the changer or the changee.

FOR THE CHANGER ...

Asking other people to change can appear as “Inflicting Help.” Insisting other people change borders on foolhardy. Think about the following:

You've already thought through the change, they haven't. Allow some soak time.

Are you really solving a problem? Check out “Solving Other People's Problems.”

Expect people to react differently at different times to different change requests.

FOR THE CHANGEES ...

Rhonda's Second Revelation: “When change is inevitable, we struggle most to keep what we value most.” – Jerry Weinberg

Remember everyone has a different Change Quotient. What you see as change, they may view as stable.

The Helpful Rule reminds us “Regardless of how it looks, people are trying to be helpful.”

Share with the changer what is most valuable to you.

I’d like to share my favorite quote about change:

“Change happens one person at a time.” Virginia Satir

And you can always change your mind about how you change.

Beyond Belief

© 2001 Esther Derby

This article originally appeared in STQE, March/April 2001.

Let me tell you a little story, a true story, about how our beliefs influence what we see in the world and affect our ability to solve problems.

Two years ago my friend Julia, who was forty-four and a bit portly at the time, starting experiencing troubling physical symptoms. She was fatigued, depressed, and generally uncomfortable. After several weeks, she went to the doctor. The doctor didn't find anything specifically wrong.

Julia was sent home with a vague diagnosis and a prescription for Prozac. After a while her mood lifted and she felt less tired, but the discomfort continued. Finally, after several months and several more visits, her doctor determined she had a fibroid tumor that was increasing in size. He decided to remove the tumor.

Julia wasn't happy to be facing surgery, but was relieved that after seven months of discomfort there was a diagnosis and a concrete plan. Two days before surgery, Julia went in for an ultrasound to precisely locate the tumor.

Based on the ultrasound results, Julia's surgery was cancelled. Julia was sent home to prepare for the birth of her daughter—who arrived, full-term, two months later.

Now I'm willing to bet that you guessed the end of this story by the middle of the second paragraph. It's obvious...if you don't already have any particular beliefs about Julia.

Julia and her doctor, however, did have a belief, built up over years, that Julia would never become pregnant. And over the course of six

months of office visits and medical exams, no one ever suggested pregnancy as the cause of Julia's symptoms.

We could say that the medical staff were incompetent, but I would say they suffered from a belief problem. Their belief caused them to overlook information that was readily available—and also limited their application of the information they were using as they diagnosed the cause of Julia's symptoms.

What does this have to do with software?

We all have beliefs about the world and other filters that affect what information we take in. Our beliefs, built up through education and experience, form the internal maps that help navigate the world we live in. Our internal maps can enable us recognize and categorize the vast flood of sensory inputs and think quickly. And often they are very helpful as general models of how the world works.

Other times, our beliefs keep us from seeing what is blindingly obvious to someone with a different set of eyes. It's "as plain as the nose on your face" to someone looking at it without our particular set of blinders.

Take Tom the test manager, for instance, assigned to a team that had always operated on participative and consensus-based decision making. Tom's framework for managing relied on his belief that, as a manager, he should entertain input from the group but make all final decisions on his own.

Soon after Tom was assigned to the group, the team was assigned to finish an evaluation of testing tools. Tom read the reports and listened to the group discussion, then closed his office door and decided which tool he favored. At the next team meeting, as he discussed his decision, he reminded the group that "we decided this at our last meeting." Tom didn't notice that most of the other heads in the room were shaking back and forth, indicating "no, we didn't."

Was Tom a bad manager? Maybe, but it's hard to say based on one incident. What we see is that because of his belief about how

decisions should be made, Tom didn't ask questions that might have given him direct information about how the group operated, and he also filtered out valuable non-verbal information that would have given him additional clues. As a result, he was far less than effective in working with the team...at least until he became aware that his map didn't match the territory.

We often don't consciously account for the existence of our internal maps, which makes them more likely to trip us up—just as Julia and her doctor, and Tom, our test manager, stumbled when their maps didn't show all the ups and downs of the territory.

Our thinking process happens so fast that it's extremely difficult to pause the process in the middle and ask, "What unconscious beliefs, filters or maps are influencing me right now?" The challenge is to pause between the time we reach an initial conclusion and the time we act on that conclusion...kind of like how we test a piece of software before we ship it to understand the quality of the product and the risk associated with releasing it in its current state. I have four questions I use for this pause in the mental process:

1. What have I seen or heard that led me to this belief? This question reminds me to really look at what data my response is based on. If I hear myself saying something like "because its always been like that..." I send up a tiny little internal red flag
2. Am I willing to consider that my belief or conclusion may be mistaken? If I'm not willing to consider that I might be wrong, it's a sign that I'm reacting out of a belief I'm pretty attached to...and it's a clear sign I need to go to the next question.
3. What are three other possible interpretations of this situation? If I can't think of any other interpretations, it's time to get some help shaking up my assumptions. I find a colleague I trust and we brainstorm as many different interpretations as we can.

4. What would I do differently if one of these other interpretations were true?

This gives me a wider range of responses to choose from, and increases the chance I'll choose one that will help solve the problem. When I start to test my conclusions, I can surface and examine my beliefs—my assumptions—about the situation. If I'm willing to admit that my initial interpretation might be inadequate, I can gather more information and represent the situation more accurately. And when I do that I open up the possibility of making better decisions, working more effectively with people, and—coincidentally—building better software.

Seeing the Other Person's Big Picture

©2000 Gerald M. Weinberg

You're entering a new situation, and you're ready to gather the Big Picture of the other people involved. Whatever you do, don't try the following process without first getting a Big Picture of yourself, as discussed in an earlier article. If you're not personally centered, this whole process will sound hollow and even smarmy.

Which others' Big Pictures? Well, who will the significant others be? Anybody I omit from this survey will potentially appear on stage at a critical juncture and spoil my best-laid plans. The people I usually have to consider are Dani, my wife and business partner; Sweetie and Ruby, my German Shepherd dogs and biggest supporters; Lois and Susie, my coworkers; other colleagues in my network, such as my PSL faculty colleagues; my customer, the one who's going to pay my bills. In this column, however, I'll focus on my clients, the ones I'm going to work with on this assignment.

I'll look for the answers to the three Big Picture questions: - How do they happen to be here? (Past) - How do they feel about being here? (Present) - What would they like to have happen? (Future)

How do they happen to be here? (Past)

When someone talks about past consultants, they've given me a free head start without my having to ask one of my "past" questions, such as,

Did Darlene choose to be here, or was she forced by me, or some other factor, like her boss?

What has been her past history on this job? What knowledge does she have that I can tap into? What prejudgments has she made about the nature of this task? Has she had early personal or cultural experiences that might affect the way she works on this job? With me? These are not excuses for poor performance, but things I have to understand to work well with Darlene.

What's been her past experience with me? With other contractors? What preconceptions does she bring to the table as a result of these experiences?

How do they feel about being here? (Present)

In this instance, I knew right away that this organization "had consultants before, but none of them made any difference." Obviously, Darlene felt that this was an important thing to say, but I didn't know why she brought this up so early in our relationship:

Does she have some reservations, or forebodings, about this assignment? About me? Does our doing this assignment conflict with something else she wants to do?

Is she eager to be here? Is she looking forward to working with me on the task that I've agreed to do?

Is she clear about what's going to be required of her if I take this assignment?

How's her self-esteem? Does she feel able to control her situation and accomplish her personal goals, or does she feel powerless?

However she's feeling, is hers the right mood for me to succeed in this job? If not, what steps can I take to help her get into the right mood?

I often seek this information by asking, "And what does that tell you

about my tour of duty?" Here are some of the answers I've received from Darlene and other people, at other times:

Aaron: You don't have a chance, so I'm not going to waste any time helping you.

Bonnie: You're going to need my help if it's going to turn out differently this time.

Carter: It's nothing personal, but this will be another of those management vision things, full of sound and fury and going nowhere.

Darlene: I'm really excited, because you're different from any of the consultants we've had before. This time, our consultant is really going to make things better around here.

Each of these answers is full of information, but I'm going to work differently with each of these people.

What would they like to have happen? (Future)

First, though, I have to know the answer to the third question, "What would you like to have happen?"

Why did X agree to work with me on this assignment? The experience? The challenge? Fear of the boss?

What will success look like, to X? Is it aligned with my success criteria? Did previous consultants solve problems that X failed to solve, thus making X look like a failure?

How long does X want me to be on this assignment? Will I be able to stay long enough to see it through? If the customer extends the project, will X be laughing or crying?

My responses

Assuming each of them genuinely hoped something would change, but knowing that each felt differently about my being here, I would construct different responses, perhaps as follows:

Aaron:

You don't have a chance, so I'm not going to waste any time helping you.

Me: I can understand your feeling. I'll do my best not to waste any of your time, but if I should happen to come up with something that might save you some time, would you be interested in hearing about it?

Bonnie: You're going to need my help if it's going to turn out differently this time.

Me: Great! What sort of help do you think you can give me?

Carter: It's nothing personal, but this will be another of those management vision things, full of sound and fury and going nowhere.

Me: Yes, I've sure seen my share of futile, grandiose projects. I personally think that big changes result from an accumulation of small changes. Would you be willing to work with me on some small thing that would help you in some way? Then we could see if we're wasting our time, or if things might be different this time.

Darlene: I'm really excited, because you're different from any of the consultants we've had before. This time, our consultant is really going to make things better around here.

Me: I'm flattered. Thank you. In what way do you think I'm different from the others, and why do you think that will help?

As a result of learning their Big Picture, I'm no longer knocked off balance. Instead, I'm well centered and already beginning to create a method of working appropriately with each of my clients.

Question and answer

Q: How do you come up with such responses in real time? They make sense when I read them, but in the moment, I often go blank.

A: There's a pattern, but it won't work if you think it's a formula. You must remain creative in order to fill in the pattern, so the first thing you must always do is center yourself. Then, find a way to connect with the emotional content of what they're saying, relating your own emotional state to theirs. Only then can you proceed to the content – what they want to have happen, and you might do next to move toward what they want.

Seeing Your Own Big Picture

©2000 Gerald M. Weinberg

The editor of Contract Professional chose the name for my column there, “The Big Picture.” He told me he chose the name “because you (Jerry) look at the business of contracting and consulting and the people skills involved, which translate across all skill sets and even industries” – in short, The Big Picture.

That’s flattering – but why would you want to look at the Big Picture? If you’re like me, you’re often called into an assignment because you’re supposed to be an “expert.” You know what an expert is: “someone who avoids all the small mistakes while committing a grand blunder.” So, before I get down to the nitty-gritty of a new assignment, I like to place everything in a grand array. I always make mistakes in my assignments, but this way I can hope they’ll all be small mistakes.

My favorite method of approaching the Big Picture is first to break down the question into three parts: Self, Other, and Context. In this column, I’ll start with Self – that is, the Big Picture of yourself.

Focusing on myself, I then ask three three questions I learned from the famous family therapist, Virginia Satir: - How do I happen to be here? (Past) - How do I feel about being here? (Present) - What would I like to have happen? (Future)

How do I happen to be here?

Here are some Big Picture questions that make an enormous difference in how I approach an assignment.

If it's the first assignment with this client, how did I make the connection? Was it through a third party, or through a direct contact by the client?

If it's a repeat, what impressions did I leave the previous times I was here? Did I leave friends, or enemies? Are my old contacts still viable? What assumptions am I carrying over from the previous assignments?

Did I get the contract I wanted, or did I have to make some concessions that might come back to haunt me?

How do I feel about being here?

Am I here reluctantly? Do I have some reservations, or forebodings, about this assignment?

Am I eager to be here? Am I looking forward to the task that I've agreed to do?

Am I puzzled about what's expected of me, or is the assignment clear? How sure am I of the assignment?

How sure am I of myself – of my ability to provide value for value received?

However I'm feeling, is this the right mood for succeeding in this job? If not, what steps will I have to take to get in the right mood?

What would I like to have happen?

Why did I take this assignment? For the money? The experience? The challenge? The possibility of a future reference? If I don't have my mission in mind whenever I choose a course of action, the client may be happy with my work, but I'll come away with a hollow feeling.

What will success look like, to me? If I come away with a pile of money but a poor reference, will I be satisfied? How about an ecstatic client who's enormously impressed by my repeating a solution I've done so often it bores me into a trance?

How long do I want to be here? If the client extends the project, will I be laughing or crying?

Using the Big Picture of Yourself

By using these three questions to assess my own state before I start an assignment, I've enormously increased my level of satisfaction. I use them to survey my state before I agree to any contract, new or renewed. On one occasion, for instance, I found I was about to renew a long-standing contract with a nice 15% increase in my daily fee. When I checked my feeling, however, I realized that I had negotiated for the wrong thing. Much of the time on the old contract, I felt that I was doing a fine job in solving the wrong problem, and I don't find this very satisfying. I didn't mind the extra 15%, but what I really wanted was more involvement in defining my own assignments.

Armed with improved self-knowledge, I halted the negotiation process and asked for more leeway, which the client was only too happy to grant. I was prepared to sacrifice at least some of the 15% increase, but the client insisted that I take it. He commented, "Now that you'll be helping do the right things, rather than just doing things right, you'll be worth at least that much more to us."

Self-assessment doesn't always pay off this directly. On another renewal, early in my career, my attempt to get more leeway in defining my work led to an irreconcilable difference between me and my client. This client knew – or thought he did – exactly what his problems were, yet I felt his poor problem definition limited my ability to be successful in my terms.

At that time in my life, what I wanted most was experience with certain types of problems and a few outstanding references. On my first assignment with this client, I had helped solve a problem that didn't vaguely resemble what I really wanted to work on. And, though the solution was innovative and successful, it didn't really help the client with his true problem – since he was working on the wrong problem to begin with. He attributed his lack of satisfaction to some unspecified shortcoming in my work, and was reluctant to give me a sterling reference.

And, of course, he was right. The shortcoming in my work was my failure to assess the Big Picture – both his and mine – before I took the assignment. As we negotiated for a follow-on, the three questions show me that the extra money he offered wasn't adequate to overcome my bad feelings about working with him again. Negotiations broke down, but at least I didn't waste another six months of my life struggling for something I didn't really want. It took me a few weeks to get a new assignment, and that cost me a few bucks. The cost is long forgotten, but I still savor the memory of my satisfaction with the new assignment – what I learned, what I earned, and how it put my professional life back on my own track.

Questions and answers

Q: It's all very nice to say that I ought to be centered in myself before I make big decisions, but whenever I get into some sort of negotiation, I lose track of myself, what I want, and what's good for me. What do you suggest?

A: This is too big a question to answer entirely, but the first part is easy. When you first notice that you're starting to lose yourself, STOP whatever you're doing. Then concentrate on how you're breathing, and switch to smooth, regular breathing.

Q: I've tried the breathing thing, and sometimes it works. But

sometimes it doesn't, like when another person is blasting at me in a loud voice. What should I do?

A: If you can't get your breathing under control, find a way to leave the situation. If you wish to continue, come back later. If you find yourself unable to leave, then that's a sure sign you must leave, now, and not come back. This is not the situation for you.

Part 4: Appreciating Individual Differences

The Identified Patient Pattern

Approaching Conflict In Style

The Identified Patient Pattern

©2006 Don Gray and Jerry Weinberg

Engelbert frowned, trying to understand why Pamela had been acting strangely. Her programming skills were among the best in the company. She had a way of getting things completed. That's why he made her project lead for Uberdenke's next UDCRM product release.

With only two weeks left until the ship date, Pamela's personality had shifted. Normally calm and composed, she had been seen crying after meetings. Occasionally he could hear her screaming at the programmers working for her. Something had to be done. He was going to have to figure out what was wrong with Pamela. And fix it.

Engelbert's decision casts Pamela as an "Identified Patient." The Identified Patient becomes the focal point for the Engelbert's intervention work to "solve the problem". When Pamela is "cured", Engelbert thinks, he can return to his normal schedule. Until then, he has more important work to do.

What happens when Engelbert becomes involved with the Identified Patient Pattern? And is Pamela's behavior change really the problem? Could there be a fundamental problem that Engelbert is overlooking? What effect does Pamela's behavior have on the project?

Identified Patient Dynamics

Trying to answer these questions requires considering several points. As Pamela's behavior becomes more pronounced:

- People become distracted.
- Extra work gets created (trying to “cure” Pamela) and time gets spent doing non-productive tasks (appeasing Pamela, hiding, gossiping).
- The less useful work gets done.

This seems straight forward enough. But how do these relate to each other? Using a diagram of effects we can visually represent the process as

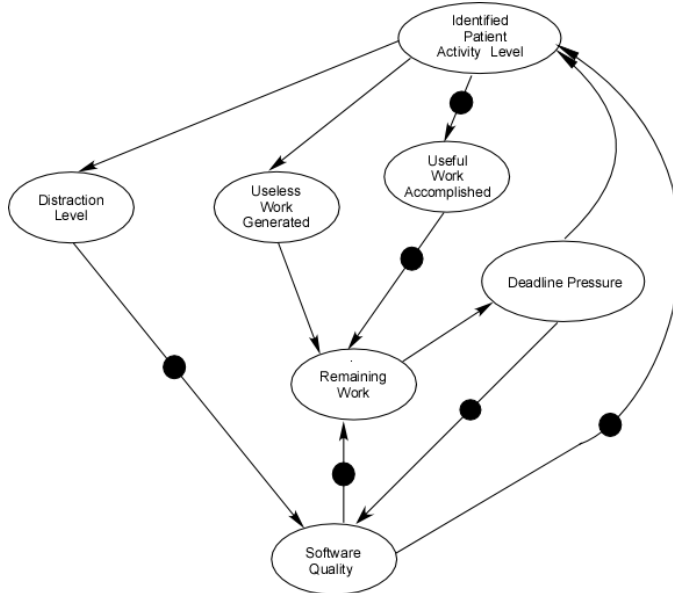


Figure 1 - Identified Patient Activity

This diagram shows the downstream effects these actions cause. Eventually these actions adversely impact Remaining Work, which increases Deadline Pressure, which increases Identified Patient Activity Level. And the dynamic starts over again with amplification due to the prior looping. The loops are all negative feedback loops. This means once Pamela (or anyone) becomes the Identified Patient,

the system starts down hill, and continues until it hits a natural limit and stabilizes or collapses.

Without knowing it, Engelbert and the UDCRM team become embroiled in activities that add more pressure to system, thereby creating more Identified Patient Activity Level on Pamela's part. By trying to "cure" Pamela, Engelbert engages in treating the symptom, not fixing the problem. In this case, Deadline Pressure continues to build, creating more stress and exacerbating the Identified Patient Activity Level.

All Stressed Up and No Where to Go

What does Pamela's worldview look like? She's the project lead for the next release of UDCRM. The project started with a bad release date (see "The Liar's Contest"). Engelbert knows this ("No Exit). The project team can look at the remaining work and see they won't make the ship date.

Uberdenke has two intertwined systems: the formal hierarchical system with nice boxes and lines, and the informal "shadow" system formed by acquaintances, friends, antagonists, history, and working relationships. The intertwined systems can be congruent or incongruent. When the systems are incongruent, the informal system struggles to compensate—not always effectively—as long as possible. When the informal system can no longer compensate, it doesn't degrade slowly, it collapses.

In this case the formal system won't acknowledge the inevitable until it's too late. The project team (informal system) knows they can't meet the date, but for a variety of reasons don't feel able to update the formal system with their reality. The incongruence between the formal and informal system takes a mental and physical toll on the employees by creating stress. To protect the others from blame, one of the people may actively (but perhaps not consciously) adopt the role of Identified Patient by "acting out" in bizarre ways. This behavior serves to distract management's attention from the other people, but at the same time distracts them from the true,

underlying problems. In the short run, then, Pamela is rewarded for her bizarre behavior, and reinforced to continue “protecting others.”

As project lead Pamela embodies the stress and manifests it via her behavior. As her stress level goes up, her behavior becomes more erratic. As her behavior becomes more erratic, the less remaining work gets finished (see above figure), which increases the project tension and so on.

When Engelbert starts trying to “cure” Pamela, two problems occur. First, the Identified Patient dynamic starts, engaging the negative feedback loops. Second, Engelbert’s attention becomes focused on Pamela, and misses the opportunity to look for a fundamental problem. Trying to “cure” Pamela may momentarily reduce pain, but—because the real problems are not being addressed—this ultimately leads to system collapse and much more pain.

Combining Pamela’s stress and Engelbert’s actions results in the following diagram.

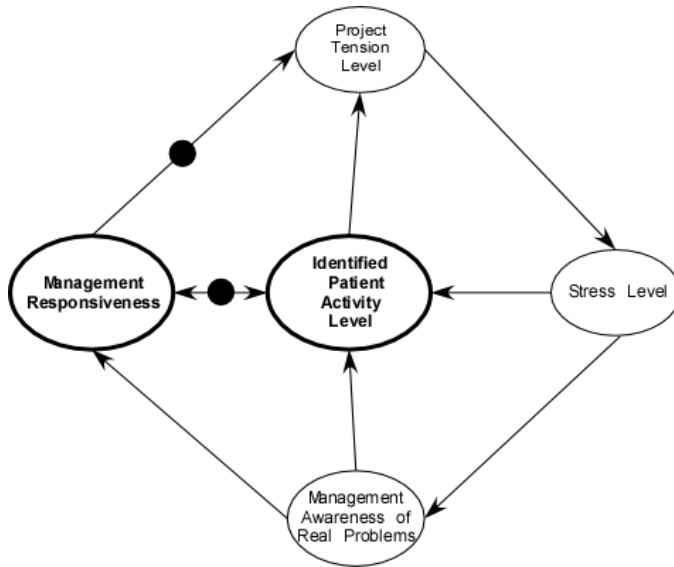


Figure 2 - Triangulation

Once again we see two negative self-reinforcing loops:

- Pamela's Tension/Stress/Patient Loop
- Engelbert's Tension/Stress/Patient/Responsiveness Loop

Both loops keep on keeping on, resulting in more of what we already have. This results from the Identified Patient's second order nature. Second order nature doesn't mean the tension and stress aren't real. It means some other problem causes this problem. Trying to cure an Identified Patient is like scratching athlete's foot. You're doing something that feels good, but when you quit scratching you still have athlete's foot, and wounds from the scratching, which has also spread the fungus more widely.

To improve the situation, Engelbert needs to move from the negative reinforcing loop to the Tension/Stress/Awareness/Responsiveness balancing loop.

What You See isn't What You Get

To “cure” Pamela, Engelbert needs to determine what’s making Pamela act out. Difficulties with this include:

- Something in the formal system doesn’t want to know what the informal system knows.
- Pamela’s behavior masks the real problem.
- Engelbert may not have the interpersonal skills needed for this.

Management acknowledging the delivery slip would relieve the deadline pressure and stress wouldn’t build. Unfortunately managers promote what they want to hear with such comments as:

- I don’t want to hear problems, I want hear solutions.
- Our competitor can do this, why can’t we?
- You’ll just have to work smarter.
- If you can’t do this, I’ll find someone who can.

Comments like these ensure that managers only hear good news, until there’s no possible way to hide the bad news. Then they ask, “Why didn’t they tell me sooner?”

Pamela’s behavior indicates a problem exists, but doesn’t give a clue what it might be. Pamela could be acting out because of:

- Work stress (our example).
- Triangulation (compensating and covering for a fellow employee).
- Problems at home.
- Substance abuse.
- Sexual harassment at work.
- Some other unresolved problem.

Most software development managers made it there by having great technical skills, not great people skills. If Engelbert discovers the problem isn't work related, he'll need help. Dealing with personal problems in a work environment can be a legal minefield, and is best left to people trained to do it.

In the extreme case, Engelbert may choose to blame Pamela for the project's problems. While incongruent and counterproductive in the long term, this tack appears to have short-term benefits. Pamela becomes the scapegoat, and hopefully the problems will leave when she does. In the mean time, a new delivery date will be set (while in panic mode). This temporarily relieves the Deadline Pressure. This dynamic may execute several times, until eventually some new delivery date finally provides the time necessary to complete the project. Or, the project collapses under all this extra effort and emotion.

Dealing with an Identified Patient

Identified Patients exist everywhere. The tip-off comes when you hear or think, "Things would be better if this person would just leave." In reality, that person leaving won't change the underlying system, and someone else will take on the role. The key to success starts with understanding the Identified Patient functions like a canary in a mine. They provide an early indication something is wrong and needs to be dealt with, but replacing the dead canary will not clean the toxins out of the air.

Unfortunately, unlike the canary whose death tells the miners that the air is probably poisonous, the Identified Patient doesn't indicate what problem exists. One thing for Engelbert to consider: the closer he gets to finding the real problem, the more Pamela may act out.

One of Engelbert's options involves getting the UDCRM team to focus on the remaining work, thereby reducing the attention given to the Identified Patient Activity Level. This becomes an exercise in active support and barrier removal, not platitudes and posters about success through working harder. If the Identified Patient Activity

Level gets even larger, this indicates that something else is going on.

If Engelbert can remove Pamela from the system, it might allow the problem to show. Of course removing Pamela might cause other problems which would again mask the original problem provoking Pamela's behavior. Another person becoming the Identified Patient would indicate underlying system problems. If the Identified Patient Activity Level prevents any useful work from being accomplished, this may be the first action to take.

Hopefully, Engelbert will be able to find and fix the problem that created the IP. Engelbert needs to be aware some fixes actually exacerbate the symptomatic problem. Common fixes that fit this dynamic include:

- Adding tools leads to the “worse before better” dynamic as productivity drops while people get used to the new tools.
- Splitting tasks leads to the communications/interface dynamic where information needs to flow through more paths and permutes while doing so.
- Splitting people between tasks leads to context swapping overhead, that period of time it takes to remember where you were and get back into the flow.
- Adding new people to the project basically incorporates all the above dynamics.

Stress Reduction

Engelbert has several options available to relieve the Deadline Pressure causing Pamela's behavior. He can use any one of three project leverage points to change the amount of Deadline Pressure:

- Features - If the number of features decreases, the Remaining Work decreases, which decreases Deadline Pressure.

- Quality Goal - If the acceptable quality is reduced, the Remaining Work reduces, which reduces Deadline Pressure.
- Desired Ship Date - As the Desired Ship Date gets bigger (more days to ship), Deadline Pressure goes down.

This looks like:

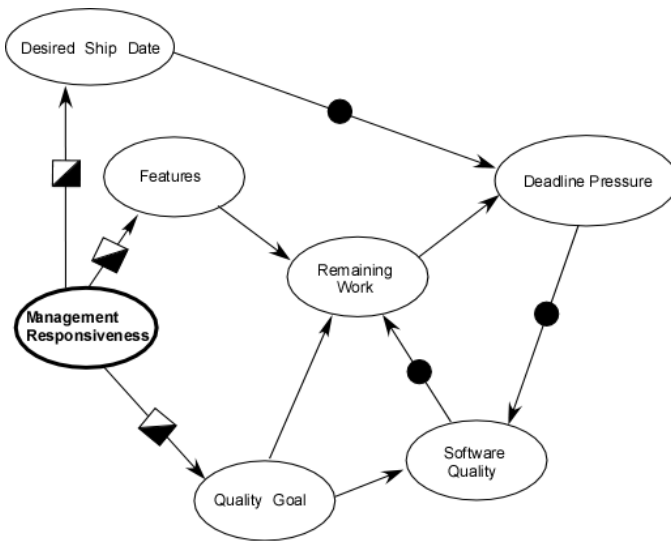


Figure 3 - Intervention Points

At this level we now have balancing feedback loops that help stabilize the system. Selecting the best leverage point or combination involves determining how the Uberdenke clients can best be served.

What to do if you're the IP

Identified Patient behavior has several possible causes. These include:

- Boredom - which results in irrelevant behavior
- Triangulation - based on placating behavior. Could be caused by

- Incongruent formal/informal systems
- Compensating for other people in the system.
- Really bad managers using blame to cope with bad results from poor decisions.

Identified Patient behavior is a symptom, not a cause. If management does recognize the behavior, they'll most likely try to fix you, not the cause. Unless you want to be "fixed", it's important to recognize the source of your incongruent behavior. This enables you to work on becoming more congruent and changing the source of the Identified Patient behavior.

"If what you're doing isn't working, try something else."Marvin's First Law¹

If you're bored, change yourself. Try and find a more challenging position in the company. Take up a hobby that will provide the challenge you're looking for. If necessary, consider changing companies. Get interested in solving some real problem that's hindering others.

Triangulating due to incongruent systems doesn't accomplish anything. The systems eventually align when the informal system collapses and the formal system has no choice. This usually happens too late to do meaningful risk management and damage control. Much pain accompanies the collapse. Trying to alert the formal system to impending doom may get you labeled as "Nay-sayer", "Not a team player" and such. If this happens, consider changing companies.

While a nice concept for those taught to "play nice with the other kids", triangulating by covering for other people boomerangs at two levels. You're not going to be able to do the work for two people, the amount of remaining work will go up, leading to deadline stress, and the incongruent systems dynamic comes into play. Along the way, you'll become exhausted and your health, both mentally and physically will deteriorate. This reduces your ability to produce and

the cycle starts over again. It's OK to occasionally help other people. But if helping other people gets in the way of getting your job done, learn to say no, or find another company to work for. Whatever you do, do it directly. There's enough confusion without adding triangles to the mix.

Lastly, if you can't change the system, and changing companies isn't an option (and sometimes it isn't), recognizing the dynamics that create the Identified Patient behavior can help you cope as congruently as possible.

And in conclusion

Identified Patients indicate something is wrong. Trying to "cure" Identified Patients seems important, but lowers the amount of useful work accomplished. Locating and correcting the fundamental problem is the best way to "cure" the Identified Patient. Engelbert needs to remember that correcting the fundamental problem may temporarily cause other problems.

¹Weinberg, G.M., *The Secrets of Consulting*. 1986, New York: Dorset House Publishing

Approaching a Conflict in Style

© 2006 Esther Derby

Jim, the test manager, started the coordination meeting with Pam, the development manager, by stating that he needed her team to turn over all the code on the first Monday in September. In a previous meeting, they'd discussed having the code complete in October, but Jim's statement sounded like a demand to Pam rather than a starting point for discussion.

Pam asked Jim what was behind the change, and when he said he wanted to begin testing early, Pam was thrilled.

"That's great," Pam said. "Early testing will really help us. We won't have all the code done until the October date we discussed earlier, but we'll have features ready to test starting in August. I can turn over features every two weeks from August through September."

"No, I need *all* the code for early testing the first week in September," Jim reiterated.

"Is the issue that you don't have anyone to assign to testing earlier?" Pam probed. Jim shook his head. "No, we need the code all at once."

Pam asked more questions to understand Jim's concerns and offered more options, but Jim stood firm.

Later, Pam mused to herself, "*It's almost as if he needs me to lose in order for him to win. I offered everything I could think of so the situation would work for both of us. Now we'll have to hash this mess out with the VP. Why does Jim always have to have his way?*"

Meanwhile, Jim was thinking, "*Why did Pam try to weasel out of this? If I agree to her options, she wins.*"

Scenes similar to this one play out in business every day. The people and the topic may be different, but the ways Pam and Jim are approaching their differences represent common approaches to conflict:

- Collaborative Problem Solving—Pam is approaching her conflict with Jim by trying to find options that will work for both of them. She's looking for the interests behind Jim's position, sharing her interests, and looking for options that satisfy both parties.
- Competition—Jim, on the other hand, is approaching the conflict with one aim in mind: achieving his goal. He's not willing to explore other options; he's intent on pressing his preferred solution. If he gets his way, Pam doesn't get hers. In addition to Pam's Collaborative Problem Solving approach and Jim's Competition approach, there are three other common approaches to conflict:
- Yielding—In this style, one person yields, accommodating the other person's wishes without pressing his or her own interests.
- Avoidance—Sometimes people do everything they can to avoid a conflict. They pretend the difference doesn't exist to avoid the unpleasantness of confrontation.
- Compromise—In compromise, people try to meet halfway. Each gives up some of what he wants and achieves some of what he wants. Compromise is common, though not always satisfying since no one is completely happy with the solution. All of these are valid and useful ways to approach conflict in some situations. And each can be destructive when misapplied.

In the story about Pam and Jim, Jim could have achieved his stated interest had he been willing to look for more options to meet the goal of early testing. His desire to prevail—competition— in this

situation will damage his relationship with Pam, and may hurt his reputation with the VP.

In Pam's approach—collaborative problem-solving, while appropriate in this situation, might not be helpful when there's a clear downside to meeting the other's interest—for example if the other person wants to pursue an illegal or unethical action. Pam's collaborative approach also takes time in order to uncover interests, generate options, and reach a mutually satisfying outcome. It's worth the time when long-term relationships are at stake, but may not be when time is of the essence or the relationship is transitory. (If a store clerk in the airport wants to talk on the phone with a friend instead of serving you, and you have a plane to catch, you probably won't use a collaborative problem solving approach. You just want to pay for your item and be on your way.)

Likewise, yielding is fine when one person doesn't have much investment in the outcome and the other person does. Yielding hurts when it's habitual—one person always gives in to the other. Others may perceive habitual yielders as doormats and walk all over them. An example in the workplace is when someone always says "yes" to all his manager's requests without discussing risks and negotiating. The long term cost of habitual yielding is resentment, depression, anger, and contempt.

Avoidance may be the best policy when there's nothing to be gained by working through an issue. For example, one manager walked away from a conflict with a peer when they couldn't agree on a testing standard. He saw that the situation would correct itself as soon as the standard (which he believed was misguided) was published to the organization, and that arguing with his peer would only damage their relationship.

We often hear that compromise is the ideal, and sometimes it is. But looking for compromise often ends in a half-horse, half-camel solution that isn't fully satisfying to anyone. Compromise leads people to miss novel solutions that can satisfy both parties and

may be better than either of the original solutions. Pam could have compromised and agreed to turn over partially completed features, but that wouldn't have worked out well for either Pam or Jim. Compromise is the best option when it's clear that a collaborative solution isn't possible.

Like Pam and Jim, most of us have a preferred style for approaching conflict. Sometimes it works for us—and sometimes it doesn't. When we approach every conflict with the same style, regardless of what's at stake and without consideration for maintaining important relationships, we may win in the short term but lose in the long term. Or we may avoid a difficult conversation but build up resentment. We're all more effective when we develop our ability to approach conflict with the style that suits the situation. Consciously choosing which approach fits best, given the stakes and the relationships, is a winning strategy every time.

Part 5: Building Relationships

The Secret Ingredients of High Morale

Peer to Peer Feedback

Why Not Ask Why?

How to Create a Buddy (Informal Mentoring) Program

Getting Some Good Out of Bad Interviewing

The Secret Ingredients of High Morale

by Esther Derby, 2004

(This column originally appeared on Stickyminds.com)

Jessica and Sean scowled as they headed back to their cubicles after the company spirit meeting.

"I can't believe they wasted two hours of our time with that award ceremony and stupid pep talk," Jessica said. "Talk about demotivating. Morale is bad around here with out wasting our time."

"Yeah," said Sean. "It was like the Oscars for the never-done-nothing crowd. I can't believe they gave out those hokey certificates."

"If they really want to build morale, they'd stop changing priorities every two days and let us get some work done," Sean responded.

"When pigs fly," answered Jessica. "I've got to get back to work... I'll be here 'til midnight getting everything ready for the build."

As Jessica and Sean turned down the hallway, Ted, their manager, peered around the corner to make sure the coast was clear. He hadn't intended to eavesdrop, but he'd just heard an earful. "Do they really think managers are that clueless?" Ted wondered. "I always thought recognition and team spirit helped morale... that and a big pay raise. But maybe I've got it wrong."

Ted may have been hearing something new, but we can't be too hard on Ted. Recognition and rah-rah have been the conventional wisdom for building morale for a long time. Unfortunately, there is no quick fix. Cheerleading is no substitute for the hard work of creating solid morale.

If you're a manager or a team lead and you really want to improve the morale on your team, take heed of this list – inspired by some real experts on what it takes to build morale in software teams – a software team.

Keep workload reasonable. If your team is being asked to “do more with less” (and who isn't these days), it's time to set priorities and decide what not to do. You can only do it all if you don't care what “done” means.

Set a sustainable pace. A 40-hour week will do wonders for morale. Enforced overtime will not. The more overtime people work, the less productive they are.

Avoid multi-tasking. Assigning people to work on several projects at once creates the illusion of progress. In fact, multitasking slows down progress. Most people are motivated by a sense of accomplishment – actually finishing something. Multi-tasking works against a sense of accomplishment because it takes longer to finish everything.

Articulate a clear mission for your group. People want to know that they are working on something worthwhile. Even if you're not in control of the company mission or product mission, you can articulate a mission for your group. Perhaps your group's mission is to “Provide accurate and timely information to management about the quality of the product” or “Create inviting and easy-to-navigate documentation that enables our customers to access all the features of Widget Master.” Say it. Document it. Then stick to it. When you're deciding on goals and how to achieve them, ask yourself and your team “How will this action help us meet the mission of our group?”

Set clear goals. A mission tells the big story – why your group exists. Every group needs goals, too – specific, time-bound achievable goals. Your group's goals may relate to a release, a project, or a service level. People will pull in the same direction when they know

what that direction is. Muddy goals make it hard for people to focus their efforts and contribute to poor morale.

Set clear priorities. Shifting priorities undercut morale. People don't like to throw away the results of their hard work. And switching priorities can have the same effect as multi-tasking – nothing reaches completion. Change priorities often enough, and people will view the newest priority as “flavor of the day.” The reality of business is that external events may dictate changes. Iterative development, with its 3 – 6 week sprints, is one of the ways to manage for accomplishment in a shifting environment. If your organization can't hold to one set of priorities for 3 weeks, it's going to be hard to make forward progress in any direction.

Remove obstacles. This is one of the most powerful morale building tools in a manager's toolkit. Find out what's getting in the way and work to remove the impediment. When people see their managers are making it easier for them to work, morale goes up. Managers can't always remove every obstacle. Let people know what you're trying, and be honest if you can't fix it.

Don't over specify. Give people the goal, set them in the right direction and let them decide how to get there. People will come up with a surprising number of creative ways to achieve the goal. Telling people both what to do and how to do it stifles morale. There's only one thing more de-motivating than over specifying the goal and the method: over specify the method, and not articulating the goal.

Deal with the un-jellers. It's hard enough to build software without someone actively working against the goal. It's a manager's job to field the best team possible. If there is a person whose interpersonal skills are making life hell for the rest of the team, deal with it. Sometimes that means moving someone off the team. Never underestimate the impact an un-jeller will have on the team.

Negotiate reasonable deadlines. We all know that we don't always get to choose the release date. If you're stuck with a hard

date, prioritize the requirements and negotiate scope. Knowing from the get-go that the schedule is impossible to meet is not very motivating.

If you're stuck with a hard date and a hard scope, talk to your team. Tell them you want everyone to work as hard as possible (but not overtime) and that you have serious concerns about meeting the goals even if every one does his best. Ask the team if they have any ideas on how to make the project work. Knowing that you recognize the situation the project is in will help the team remain focused and energized. Working reasonable hours is a better strategy for reaching goals than going on the fabled death march. Pep talks, contests, and certificates won't build morale. They can be fun when things are going well, but when your team is in the pits, they contribute to cynicism. There's no short-term fix or magic formula for boosting morale, but old-fashioned effective management may just do the trick.

Peer-to-Peer Feedback

©2005 Esther Derby

Not long ago, a developer approached me for advice about a problem team member. The developer reported that one team member was causing resentment, alienating other team members, and generally making life difficult for all. No one wanted to work with him.

“What is he doing to cause all this?” I asked. The answer surprised me. “He picks his nose,” the developer said.

“He picks his nose? Have you talked to him?” I asked.

“Of course,” my developer friend replied. “I talked about the importance of manners at our team meeting. And I talked about how we all had to be careful about spreading germs.

“He still picks his nose,” he continued. “It’s gross. The only thing I can think of is to start picking my own nose to see how he likes that.”

Nose picking is an unattractive habit. But the real source of this team’s problem isn’t nose picking. The real problem is that team members don’t know how to have an uncomfortable conversation—peer-to-peer.

How to talk about a difficult subject.

Remember, the over-arching goal of feedback is to improve working and social relationships. When you think of it that way, it’s easier to find a respectful way to deliver a difficult message.

Use “I” messages.

Talk about what you see, and what you feel. Start your feedback with a sentence that starts with “I,” rather than with “you.”

Describe what you have seen and heard.

Stick to the facts of what you have seen and heard. Describe behavior rather than applying a label. For example, “Yesterday in our team meeting I heard you call Sara an idiot.” rather than “Yesterday you were rude.” Labeling the other person only puts him or her on the defensive.

Own your own feelings about the situation.

Some people advise using this formula to give feedback: “When you do X, I feel Y.” But this construction implies that one person is the cause of another’s feelings. No one else can make you have feelings. To remove the implied cause and effect, you might say, “When I hear you call Sara an idiot, I feel like you are disrespecting her,” or “I want to tell you about something that you do that’s a problem for me.” Then describe the behavior.

Talk about the effect the behavior has on you.

People often don’t realize the effect their behavior has on other people. Explain (briefly) how the behavior you are talking about affects you. Explaining the impact gives the feedback receiver

information so they can choose what to do with your feedback. If there's no impact, then a request seems arbitrary. The conversation could start with "When I hear you call Sara an idiot, I feel like you are disrespecting her. I worry that you talk about me that way when I'm not in the room."

Ask for what you want.

If you have a specific change you'd like to see, make a request. You can make a request for behavior to stop, start, or change. For example, "I want you to treat our co-workers with respect and stop calling Sara and our other co-workers idiots."

It's not always easy to give feedback. I still feel anxious when I prepare for a difficult feedback conversation. I have almost always found that the pre-conversation anxiety is worse than the actual event. And the pay off for having the conversation is well worth the effort.

So what happened with the nose-picker?

I advised the developer to have a private conversation with the offending team member. "Give him the benefit of the doubt," I said. "What if he's unaware he's picking his nose? It may be an automatic habit. And even if he's aware he's picking his nose, he may not be aware of how it affects you and other people on the team."

The developer agreed reluctantly, and we worked out a little script. Here's what he decided to say to his nose-picking colleague:

"Joe, this is really awkward for me. I want to tell you about something that you do that's a problem for me."

[Pause]

"I've noticed that during our team meetings, you pick your nose."

[Pause and wait for a response. This may be all you need to say.]

"I have some judgments about nose-picking. I was brought up that it's not appropriate. When I see you picking your nose, I feel worried about you spreading germs. My reaction is getting in the way of our working together."

[Pause and wait for a response. This may do it. "Would you please stop picking your nose while we're working together?"

The next week, he reported back.

"You'll never guess what happened," he said. "You were right, he wasn't even aware he was picking his nose. But it was really awkward," he continued. "He was embarrassed but he was also grateful I told him. I guess I shouldn't have waited so long."

It is hard to address interpersonal and work issues directly—even when the issues aren't as awkward as someone picking his nose. Respectful feedback can improve working relationships. And handling issues directly keeps little irritations from growing into major divisions. "I've noticed that during our team meetings, you pick your nose."

[Pause and wait for a response. This may be all you need to say.]

"I have some judgments about nose-picking. I was brought up that it's not appropriate. When I see you picking your nose, I feel worried about you spreading germs. My reaction is getting in the way of our working together."

*[Pause and wait for a response. This may do it. "Would you please stop picking your nose while we're working together?"**

The next week, he reported back.

"You'll never guess what happened," he said. "You were right, he wasn't even aware he was picking his nose. But it was really

awkward,” he continued. “He was embarrassed but he was also grateful I told him. I guess I shouldn’t have waited so long.”

It *is* hard to address interpersonal and work issues directly—even when the issues aren’t as awkward as someone picking his nose. Respectful feedback can improve working relationships. And handling issues directly keeps little irritations from growing into major divisions.

Why Not Ask Why?

© 2011 Don Gray

It all started with a tweet I posted:

“Why” questions trigger feelings bypassing data input and thinking.
#dontdothat

As this got retweeted, interesting questions started coming my way:

- What about the Five Whys?
- Do you have data?
- What is your context?

All good questions.

“Why” questions have the ability to both gather data and to probe for underlying thoughts and decisions that lead to action. Other interrogatives (what, when, where, how) provide a better way to gather data since they focus on physical items or actions.

So when do “Why” questions work well? How might “Why” questions lead to unexpected results? What can we do about that?

Solving Problems: Toyota and the Five Whys

I was writing process control code for a living when I first heard about the Five Whys. It made sense for finding a problem’s root cause. The example went something like this:

1. Why did the line stop? Because the conveyor gear reduction box froze.
2. Why did the gear reduction box freeze? Because it didn’t get lubricated during the last preventive maintenance.

3. Why didn't it get lubricated during the last maintenance? It's a new piece of equipment and wasn't on the preventive maintenance check list.
4. Why didn't it get added to the maintenance check list when it was installed? Because we don't have a standard way of adding items to the check list.
5. How can we create a standard way of adding items to the check list so this won't happen again?

Asking why uncovers another layer of information that eventually leads to the problem's root cause and allows us to craft a solution to (hopefully) prevent the problem. Since we're starting with observable data, asking why works well here.

Gathering Data: Five-year-old Whys

Anyone who has spent time around children has probably experienced a period of incessant whys.

Why is the sky blue? Because air molecules scatter light from the sun.

Why do the air molecules scatter the light? Because they get in the way of the sun's rays.

Why do they get in the way?

(And so on.)

Often, the question-and-answer process ends with "Because I said so, that's why."

Asking why provides children with new information, and data expands their knowledge, so why works well here—at least until "Because I said so."

When Why Might Not Work Well

Your teammates, managers, and coworkers are neither mechanical processes, which don't care if we talk about them, nor five-year-olds attempting to gather more information about their world. They

come complete with experiences you don't know about and ideas about how things should work. As such, your why questions may trigger in others an emotional response that catches you unaware. What might generate such a response?

An Interaction Model

The Satir Interaction Model¹ provides a framework for understanding how interactions proceed, as shown in Figure 1.

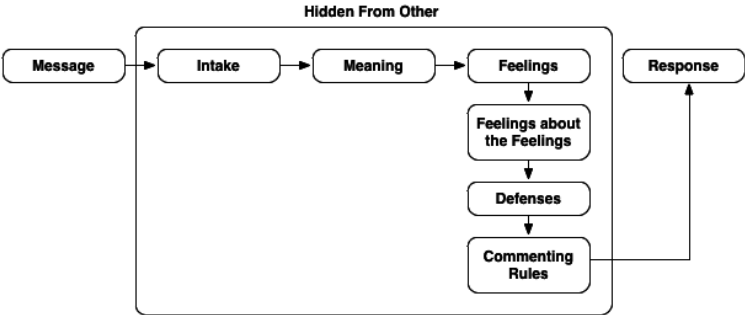


Figure 1 - Satir Interaction Model

Take Tony, for example. You've noticed the build server has been sending emails announcing that the build broke, and Tony usually makes the commit that occurred just before the build that broke. Wanting to be helpful, you head to Tony's cube and ask, "Why do you keep breaking the build?" You're looking for information. The message has been spoken and becomes the input.

Based on how Tony feels and his background, he can infer several different meanings. He may think you're picking on him. He may think you mean "Tony, you're incompetent." He may interpret the question as a request for information.

The meaning Tony chooses determines his feelings about your message. If he thinks you're picking on him, perhaps he will feel afraid or threatened. If he believes you think he's incompetent, he might become defensive, or he might be relieved that you might

help with the problems he's having.

Tony also will have feelings about those feelings based on his background. Perhaps his father taught him not to back down when threatened, or to prove you're right when challenged, or even that accepting help shows weakness. Tony may feel ashamed, angry, confused, or relieved.

Based on Tony's life experiences, he may defend himself by:

- Blaming—"The stupid formatting rules take too long to check. If IT would buy us better computers, this would not happen."
- Placating—"I'm so stupid. I should do better."
- Being super-reasonable—"Has anyone checked the rules on the build server to make sure they agree with how the builds work on my workstation?"
- Feigning irrelevance—"It's almost lunch. Where should I go today?"

After these steps, Tony will work on his response. What can he safely say? What do his family, social, and corporate cultures say about what he can say? Some comments can be career limiting. Does he have an "always be agreeable" rule? These commenting rules affect his verbal response.

Everything between the Intake and Response steps happen internally for Tony, but you can guess what happens based on his response. If Tony replies in a way congruent with your intention—"I'm having problems with formatting. Can you help me figure out what's happening?"—the request for information succeeded. If Tony becomes bellicose, belligerent, or his answer bewilders you, the question triggered a meaning and feeling not related to your request for information.

The Interaction Model and Temperaments

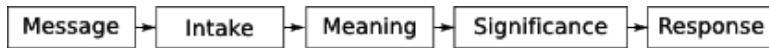


Figure 2 - Simplified Satir Interaction Model

In *Congruent Action*² Gerald M. (Jerry) Weinberg collapses the steps from Feelings through Defenses into a single step, which he calls Significance, and correlates how Keirsey's Temperaments³ (see the sidebar) tend to work through the Interaction Model:

- SJs stay in Intake mode too long.
- NTs tend to go instantly to Meaning.
- NFs tend to jump immediately to Significance.
- SPs go so fast it looks to others as if they jump instantly to Response.

More than half of IT professionals implicitly skip the Intake step based on their personality preference. You want to know “why” to collect data—41.6 percent will look for the meaning behind your question and another 12.1 percent think about how the question makes them feel.

Jerry offers the following suggestions:

- For NTs/NFs, ask, “What did you see or hear that led you to that conclusion?”
- For SJs, ask, “What can we conclude from the data we have so far?”
- For SPs, appeal to their desire to be clever and ask them to teach you how they did it.

Use Data Questions to Gather Data

Questions that start with other interrogative words, such as when, what, where, and how, help people focus on the data aspect of the question.

If we ask Tony a different question, we can help him focus on the data we would like to know—for example, “What steps do you take prior to committing to the build server?” or “How do you decide it’s time to commit your code?” Tony still traverses the Interaction Model, but we’ve explicitly asked for data.

Over the years, I’ve had some why questions bounce back to me with responses that left me wondering how what I said triggered that response. Using the Satir Interaction Model, I’ve learned to unravel the responses. I’ve also learned that if I want data, I should use data-oriented questions that start with how, what, when, and where, and use why as a last choice.

On the other hand, if you want practice unraveling communications, start with why as often as possible.

Temperaments

“There are two types of people: people who divide people into two types and those who don’t.” –Barth’s Distinction

Dividing people into groups is a time-honored tradition. Circa 340 BC, Plato divided people into four groups: Artisans, Guardians, Idealists, and Rationals. Since then, many others, including Aristotle, Galen, Paracelsus, Fromm, and Myers, have found ways to divide people into four groups.

In 1978, David Keirse and Marilyn Bates “developed and described” the temperaments “in modern form.” They “found that selectively combining [N with T/F and S with J/P] produced a descriptive personality system similar to the four temperaments ... described centuries earlier.”⁴

Temperament	Motivation	Strengths	Weaknesses	Percentage in IT Profession
Guardian / SJ	Need to be responsible Value tradition	Reliable Organized Focused Conscientious	Judgmental Controlling Inflexible Close-minded	22.6
Artisan / SP	Need for freedom and action Value being in the moment	Optimism Adventurous Realistic Adaptable	Hyperactive Impatient Impulsive Scattered	9.5
Rational / NT	Need knowledge and competency Value theory and mind	Innovative Inquisitive Analytical Independent	Arrogant Cynical Critical Distant	41.6
Idealist / NF	Need to understand themselves and others Value authenticity and integrity	Compassionate Warm Helpful Idealistic	Hypersensitive Overly emotional Impractical Unrealistic	12.1

Figure 3 - Temperaments in IT

“When the only tool you have is a hammer, it is tempting to treat everything as if it were a nail.” –Abraham Maslow

I like to remind clients that temperaments represent one way of understanding people and how they interact.

References

- ¹The Satir Model: Family Therapy and Beyond, 1991, Science & Behavior Books, Inc., pp 121-129
- ²Quality Software Management, Volume 3, Congruent Action, 1994, Gerald M. Weinberg, pp 108-109
- ³Please Understand Me II, 1998, Prometheus Nemesis Book Company
- ⁴MBTI Manual: A Guide to the Development and Use of the Myers-Briggs Type Indicator, 2003, Isabel Briggs Myers and Mary H. McCauley

This article was originally posted on StickyMinds.com on January

28, 2011

How2 Create a Buddy (Informal Mentoring) Program

© 2004 Johanna Rothman

Introduction

If you've hired new people or transferred people into your group, you know that they're not immediately productive when they start. If you're lucky, they start to be useful in a month, but you most likely spend closer to six months or even a year before they raise the entire group's productivity. One technique for bringing a new employee up to speed quickly is to assign that person to a buddy, someone who will mentor the new employee through the first few days and weeks on the job.

If you'd like to bring new hires up to speed quickly, or if you'd like to cross-train your group, try a buddy system.

Buddies aren't just for the pool

A buddy system at work is nothing like the buddy system you might have experienced in a pool when you were a kid. You didn't get into the pool without your buddy, and when the whistle blew, you and your buddy found each other. You and your swim buddy were there to watch over each other, to make sure you both knew where the one was. This is not your child's swim-buddy system.

A buddy system at work is a technique to help already capable staff learn how to apply their skills more quickly, to reduce the floundering that occurs at work when a new hire starts.

Here's how to think about the knowledge necessary for work:

Functional skills: what the person knows about their job (development, testing, project management, writing, etc.) People learn these skills at school and on the job.

Product Domain Expertise: how people understand the product and the product domain, and how they apply those skills to the product. People learn about the product and its internals by working with products.

Tools and Technology: how well the person knows the tools of the trade: compilers, testing tools, defect tracking tools, databases, and so on. People may learn some basic skills, but most skills are learned on the job.

Industry Knowledge: what the person knows about the kinds of people likely to buy your system, and the general expectations of the your users. People learn about the industry by working in it, on the job learning.

Your new hire already has the functional skills—that's why you hired that person. But the new employee may need product domain expertise, both in the problems the customer has and in how your product solves those problems. Your new hire needs to know how you've customized the tools and technology you use. The more you can help new staff understand your local environment, the faster they will adapt to your environment, and the more quickly they will contribute to the overall efforts.

Requirements for a buddy system

Creating a buddy system requires that you have people who are willing and able to mentor others. In addition to the mentors, you'll need to document a few key pieces of information: the physical layout of the building and the security, where the tools are located

and how to use them, where the project information lives, and some information about the product internals. The more documentation you have, the less the new staff will interrupt the experienced staff. The documentation doesn't have to be a huge honking binder. In fact, if you can keep the information to one web page or one to two printed pages, the information will look accessible and not threatening to the new hire.

Evolution of the buddy from guide into mentorWhen someone new starts at work, that person needs to know everything, from how to use the phone and voicemail systems, where the bathrooms are, where the lab equipment is, to how to use the tools of the trade, such as the defect tracking system or the configuration management system, or even the compilers.

When you first assign a buddy to a new person, the buddy acts as a guide, to explain where everything is. If you've never had a buddy system in place, this is a good time to document the locations of everything, including the physical facilities and tools. Once the locations are documented, the buddy can explain once where to discover more information, and you've already farther ahead for the next new person.

On the first day, it's helpful if the buddy can take the new employee to lunch, preferably with a group of people in your cafeteria. If you have no cafeteria, choose a lunch location that most people use. Sometime during the first day, the buddy can demo your products or applications to the new hire. After the first few days, the new person knows where the cafeteria and the bathrooms are, and has probably started reading product and tool documentation. The buddy is available to answer questions about which platforms to make sure to compile on before checking in code, or how you use certain databases, where specific tools are, and so on.

During the first few weeks, discussing architectural or design tradeoffs with new developers, test strategies and choices with new testers, or project monitoring specifics with new project managers

will help the new employee understand the context of how you work, which choices you've already made, and how to make sure the new employee's work fits into your environment. As the discussion moves past the basics, the buddy has moved from guide to mentor. You may choose to have a formal ending to the buddy relationship after one month. If so, one way I've found useful is to mark the new hire's 1-month anniversary in a group meeting and say, "John is now experienced enough to not need the formal services of a buddy. Thank you Steve, for taking on the role of the buddy. John, you can ask questions of any of us at any time, and hopefully you can take on the role of a buddy to a future new hire."

Pitfalls or common traps

Even when you've planned for a buddy system, things can go wrong. Here are some common problems:

The buddy assigned doesn't want to be a buddy. Some people don't want the responsibility of helping someone new acclimate into a new job. Sometimes, they're not temperamentally suited for the role (such as people who are so shy or introverted they have a difficult time speaking in public), or they are on the critical path for some deliverable, or they don't feel as if they know everything the new person needs to know. If your employee doesn't want to be a buddy, don't demand that the employee be a buddy. Instead, understand why your employee is reluctant, and work on that problem. One caveat: if the employee is too shy or introverted to easily speak with a new hire, find other ways that employee can help new hires. Don't demand someone perform the buddy role when they are nervous around new people.

The buddy inflicts help on the new hire. Some people are so inspired by their buddy role they give unsolicited advice about the new hire's work. The buddy needs to provide advice about how the work is

performed at your organization, not how to design or how to test or anything else, unless the new employee has requested peer review.

The buddy performs the new hire's work. Another form of inspiration is to take over the new hire's work. One of the common reasons I hear for this is, "It would take me less time to do it myself." Well of course it would. That doesn't mean one employee can or should take over the work of another employee. The first few months are when you can expect the new hire to be less productive. Don't allow a buddy to short-circuit the new hire's learning experience. The buddy stops being a buddy before the new hire is ready. If the buddy takes a vacation or has a trip during the first crucial weeks, the new hire is left stranded. The new hire and the buddy have already created a relationship. If the new hire has to create another relationship with a substitute buddy, the new hire is less likely to ask the necessary questions.

Use a buddy system for cross training

If you're concerned about single points of knowledge in your environment, you can also use a buddy system for cross training. In this case, the buddy assumes the new person only needs product domain expertise, or alternative techniques to applying functional skills to the new product domain. When you use a buddy system for cross training, make sure the two people develop a list of milestones where the already-knowledgeable person performs less and less work over time. That way you and the person learning this area both have feedback about how well the new person is learning the product.

When I use a buddy system for cross training, I assume it will take the people about three months to transition the product knowledge, unless you have very short releases.

Summary

Creating a buddy system for a new hire isn't difficult, but does need to be handled with care. Make sure you've chosen a buddy who is willing. Create the minimum set of documentation, and revise it as you hire new people. Set an end date for the formal buddy relationship. Watch for pitfalls so you can guide both the experienced and new employees. A buddy system can dramatically reduce the time a new hire requires to be productive. A new hire will need to ask questions anyway, so make sure you have an effective system in place to deal with those questions.

Getting Some Good Out Of Bad Interviewing

©2007 Jerry Weinberg

Contract professionals, on the average, change jobs more often than employees, so they are involved in lots of interviews. One of our SHAPE forum threads was started by Pat Ferdinandi, an independent consultant, who complained: “I am continually amazed at some of the ridiculous or inappropriate questions I get when interviewed by prospective clients. Keeping a straight face and not losing my cool is sometimes a challenge.”

The Shapers then contributed many examples from their own experience, and I’d like to recount some of them and the lessons they might carry for my readers.

Insensitive and Oblivious

An interviewer made the following statement to Pat: “A psychiatrist has a model that answers to specific questions fall within. So, by having a base model, he can identify that the problem with the marriage is the lazy wife.” Pat, being a wife herself, didn’t care for this analogy. The problem here is, of course, his gender biases, which might exist throughout the company - but even more, his total insensitivity to the person he was interviewing.

Well, not total insensitivity. For that award, I have to turn to a story from a woman who was interviewing with the manager of the department she would work in if she took the assignment. During the interview, he said, “I want you to feel free to come in and talk to me any time, about anything. Think of it like a young girl talking

privately to her father where she can practice her techniques for sexual advances knowing that she's perfectly safe with him, because he's her father."

How did she handle this? Apparently, the same way everybody present handled it when she told us. We sat there for about five minutes with our mouths open, not able to make a sound. She then just walked out. Maybe he's a masher, or maybe he's totally out of touch, but in either case, you don't want to work for him.

Gender bias is bad enough, but maybe you're the right gender. Sexual innuendo is worse, but maybe the interviewer won't find you attractive. Insensitivity, however, eventually snags everyone working in the environment, so stay away even if you're not a "wife" or a daughter.

The Nedlog Rule There's a variation of the Golden Rule that applies to these and a great many other the bad interview lines:

As they do unto others, they will eventually do unto you.

I call this the Nedlog Rule. Here are some examples:

Candidate: "So what happens to the staff from the other sites?" (The company was a merger of several companies)

CIO: "We offer relocation to the ones we want. That's the nice thing: we can start over and define our culture"

The candidate has just learned how he will be handled should he become excess baggage. The interview continued:

Candidate: "Given that, what culture would you want to have?"

CIO: "That's a good question. I haven't really thought about it."

The candidate figured they wouldn't start thinking about the culture they wanted just because they hired him.

Here's another Nedlog example:

Interviewer: "We would like to have you working for us. I already heard about you from them (thumb pointing over shoulder at his employees) ... By the way, one of them, namely George is not very productive, but I can't fire him"

This candidate figured she didn't want to be called "them," nor did she want her performance deficiencies to be discussed with actual or potential co-workers. If they badmouth other employees, you have to wonder what they'll say about you.

Dumb or Ignorant

Sometimes, the questions you're asked tell you all you need to know about the kind of people who will be managing you if you take an assignment. A software consultant was interviewing for an assignment in a hardware organization whose managers knew nothing about software. They asked questions like these:

- You know something about software?
- Do you write computer programs?
- Can you type? How fast?
- How long does it take someone to write a computer program?
- What's a database?
- What programming language should we use for everything?
- What's the one thing we should tell people to do to cut development costs in half?

My own answer to this last question would be, "Get rid of the managers who ask questions like this one." Where do such questions come from? In trying to understand this, I tried to imagine what questions I would ask if I were going to interview, say, a brain surgeon:

- You know something about brains?
- Do you operate on brains?
- Can you cut with a scalpel? How fast?
- How long does it take someone to do surgery on a brain?
- What's a nervous system?
- What surgical tool should we use for every operation?
- What's the one thing we should tell people to do to cut surgical costs in half?

Looking at these questions, I realized that any surgeon would know instantly that I knew nothing about surgery, let alone brain surgery. I would think that the surgeon's next move would be to ask me, "What is your role with respect to the job I'm interviewing for?" If there was any role at all, no decent surgeon would take the job.

Should it be any different if you're a software developer, or tester, or project manager? It's not the lack of knowledge that's the problem; it's the lack of knowledge about the lack of knowledge. I can work for someone who doesn't know beans about my specialities, as long as they know they don't know.

Hidden Agendas

Sometimes, what the interviewer says will reveal to you that they're not really trying to fill a position. Sharon Marsh Roberts identified two common types of hidden agenda interviews. The first are "professional courtesy interviews" - ones which are scheduled because:

1. the interviewer owes someone a favor;
2. the interviewee has some (hopefully worthwhile) connections; and
3. there is time on everyone's schedule.

Sharon opposes this to the “comparison shopper interview”:

1. the interviewer has someone in mind;
2. Human Resources hasn’t approved the decision; and
3. the corporation demands some sort of record of “due diligence”.

Many of the worst interviews fall into these categories, and the best thing you can do (since you’ll never get the job) is save time by getting out quickly. One young contract professional was invited for an interview, believing that she was under consideration for a particular assignment. The interview started badly enough and went downhill fast enough so that she asked the defining question: “Is there something that I should be telling you to explain that I could perform this job?”

The answer communicated bluntly something on the order of, “No, because I’m interviewing someone who’s isn’t qualified for this position.” She gracefully departed. There wasn’t much she could say, and besides, why waste more time?

Conclusion

There are dozens more ways in which interviewers reveal this kind of information, and if my readers clamor for them, perhaps I’ll provide some more. The important thing to know, though, is no matter how inept an interviewer may be, you—the interviewee—can almost always get the information you need to make a decision. Just don’t lose your cool and thereby lose the information.

Part 6: Group Work

Four Different Ways of Participating

Choosing Facilitation

How to Improve Meetings When You're Not in Charge

Understanding And Creating Safety

The ROTI Method of Gauging Meeting Effectiveness

Collaboration Is More Than Meetings

Four Different Ways of Participating

©1999 Gerald M. Weinberg

Contract professionals are seldom sent to classes by their customers, but often pay for their own professional development. As such, they're eager to get full value for their time and tuition.

Moreover, contract professionals often find themselves as instructors in classes, in which case, they're also interested in making sure the class goes well. But not all classes go well, and often it's because the mix of participants isn't ideal. Or, to look at it another way, it's because the instructor doesn't know how to handle certain kinds of participants.

Our Problem Solving Leadership workshops are designed for professional development - specifically, to develop leadership abilities in the participants. Because leadership means so many things to so many people, our participants arrive with different expectations, so one of the first things we must do is clarify why each participant is in the room.

Over the years, I've seen students who fall into four categories - Customers, Learners, Visitors, and Complainers.¹ Each type of student requires a different approach from the instructor.

A **Customer** is someone who arrives with a particular problem to solve, for themselves or concerning some other person(s) or situation. If a student is a Customer, it's possible to gain a relatively clear description of this problem. Here are some examples of Customer problems brought to our PSL Workshop:

"I want to communicate better with my customers when they are trying to tell me their requirements."

“I want to do a better job facilitating meetings.”

“I can’t handle my boss when she criticizes my work.”

“My teammates often don’t listen to my ideas, so that they misunderstand them and reject them.”

“I’m putting together a new project team, and I want to do it without making the same mistakes I made last time.”

The participant as Customer quite clearly wishes to do something about this problem and is seeking help from the class and the instructor. All the instructor has to do to satisfy the Customer is offer a high-quality class. Most instructors wish for a class composed entirely of Customers, but it’s never that easy.

A **Learner** is someone who comes in without a particular problem to solve, but just wanting to learn whatever the class has to offer. Typically, Learners may state their objectives in ways such as these:

“I heard this was a neat class, so I wanted to learn what you had to offer me.”

“I’m interested in teamwork and how to build teams.”

“My best friend at work took this class and wouldn’t tell me anything about it except that I should come and learn whatever I learn. I like that idea.” “I’d like to improve my problem solving skills (without mentioning any particular situation).”

Learners are fun to have in class, because they gobble up everything the instructor has to offer. They never ask that hard question the Customers may ask,

“What’s the relevance of what we just learned to so-and-so.” Nevertheless, they can be a danger to the class because they may drift off in any direction just because it catches their fancy. If the instructor gets distracted by Learners, the Customers start getting angry because they’re not getting what they came for.

A **Visitor** is an uncommitted participant, often involved in the class under some kind of duress, implicit or explicit, and usually because

of the concerns of some other person. Typical statements by Visitors might be:

“My boss told me I needed to learn whatever this class teaches.”

“My company bought three seats in this class, and someone cancelled one of them, so they sent me to fill the empty seat.”

“Everyone in our department has to go to this class.”

“I can’t get promoted unless I’ve taken this class.”

The Visitor has no agenda to participate in class discussions, and works on any exercises in the most minimal way. Attempts by the instructor to improve the Visitor’s participation are likely to be fruitless or to lead to “resistance” that can be disruptive to the whole class. Trying to capture the interest of a Visitor can easily distract the instructor from paying attention to others in the class who are not Visitors.

Instructors need to treat Visitors respectfully, always being open to their contributions, should they decide to start making them. If Visitors do contribute, honor their contributions with complimentary (but not effusive) feedback. Until they volunteer to contribute, however, avoid soliciting their participation or trying to assign them tasks. They have to make their own choice to transform themselves into Customers, or at least Learners.

A **Complainer** does have a particular problem (or list of problems), specific or vague, either concerning themselves or about some other person(s). The Complainer’s problems may resemble the problems brought by the Customer, but it’s not clear that the Complainer actually has any wish or hope that these problems be solved. Often, the instructor can distinguish the Complainer from the Customer by the whining, helpless tone in which the Complainer presents these problems - often repeatedly.

Complainers don’t really believe that their problems can be solved by this class, or possibly even solved at all. Therefore, they should

be treated initially as Visitors, with empathy. If they display any hope that something can be done about their problems, that's the time to get them engaged in the class. Often, this display of hope comes in the form of a tentative question that doesn't refer directly to the Complainer, such as,

"Could this technique be applied to dealing with a boss who doesn't appreciate your work?"

"But this technique couldn't be used on a project under heavy schedule pressure, could it?"

Understanding and recognizing the four types of participant is obviously helpful to you when your job is to instruct a class, but what good does it do if you're a fellow student? And what if you're not in a class, but just a plain vanilla contract professional working on the job?

As a contract professional, I'm seriously interested in my education. When I attend a workshop, I want to get my money's worth - because I'm paying my own tuition and I'm usually not getting paid for my time. (Sometimes I can get my client's to pay me for attending a workshop, but that's a subject for another column.)

Therefore, whenever I'm in a workshop, I watch my fellow students just as I would if I were the instructor. If they're Customers, I watch to see that they're shopping for the same learnings I am, because if they're not, they may steer the workshop into areas that are not related to what I'm shopping for. If they do, I may suggest something to the instructor, such as, "I see that you and Val have a deep interest in this subject, but it's not my main focus right now. Would you be willing to continue off line with those who are really interested?"

If some of my fellow students are Learners, I cautiously enjoy their excursions into the unknown, because I'm always partly a Learner myself. I do my best to protect them from overfocused Customers (including me), while at the same time keeping my own goals in

sight. If necessary, I try to steer things gently, as by offering, “Thank you for that wonderful exploration. I was fascinated, and could probably spend all day exploring it. But I wonder if we couldn’t come back to the subject of ...”

If the instructor has a Visitor or Complainer to cope with, I consider myself a junior partner of the instructor and try to help out, generally off line during breaks. This role comes naturally to me, since we usually lead our own workshops in pairs - one instructor “on stage” and one ready to handle Visitors or Complainers one-on-one outside of the classroom.

For example, in a recent workshop, one of the Visitors was using one of our breaks to try to recruit other students to his cause by convincing them that the workshop was a waste of time - which, of course, would make his boss look like a fool for making him attend. Once I recognized that he was a Visitor, it wasn’t very hard to turn the subject toward his problems with his boss, and away from deprecating the very professional job my co-instructor was doing. Eventually, he realized that he had a choice between spiting his boss by learning nothing, or spiting her by actually learning some things that made him a better professional.

And speaking of professionals, I don’t spend most of my time being a student or instructor, so what does all this have to do with my day-to-day work? If you’re a regular reader of this column, perhaps you know by now that I always have my own learning in mind when I negotiate an assignment. I firmly believe that when I stop learning, I’ll stop being a professional - so, I’m always a student.

Yes, I’m on the job to earn money by helping my client, but I’m also there to learn, so I’m always on the lookout for Customers, Learners, Visitors, and Complainers among my co-workers. I believe that any worthwhile high-tech assignment is too difficult to be accomplished without constant learning by all participants, so I’m always working to enhance the learning environment, for me and for the others.

And, one more thing. Like anybody else, I, too, am susceptible to these stereotyped behaviors:

- becoming too focused on just the immediate job at hand (Customer);
- drifting off the subject into interesting side tasks (Learner);
- detaching myself from the importance of the assignment (Visitor);
- feeling powerless and whining about everything and everybody (Complainer).

None of these roles enhance my performance on the job, so I always try to watch myself, too - and bring myself back to a more professional and effective role.

1 At the suggestion of David Schmaltz, one of our former PSL instructors, I adapted these categories from my teacher and colleague, Bill O'Hanlon. See, *A Brief Guide to Brief Therapy* by Brian Cade and William Hudson O'Hanlon Â©1993 by Norton in NYC.

Choosing Facilitation

© 2003 Johanna Rothman

Meetings are a fact of our lives. Most of the time we don't need a facilitator to help move our meeting along; we can manage to accomplish the goals of the meeting without a formal facilitator. However, there are times when a facilitator makes sense.

Darcy is a middle manager in a startup. They have enough money for the next eight months. For the last three months, the senior managers have closeted themselves in meetings day in, day out. Darcy knows they're trying to define the current strategy and tactics to accomplish the goal: drive enough revenue to break even. If they can break even in eight months, their investors will consider investing just a bit more to overcome the slow economy and the company will succeed. If they can't break even, they'll be shut down.

Darcy's no dummy. Neither are the other people in the company. They all know what these closed-door meetings mean. Darcy is concerned that if the senior management team can't figure out what they're going to do soon, the meetings will turn into layoff-decision meetings.

Darcy's management team needs a little facilitation to help them overcome their inability to come to a decision and move forward to specific tactics and action items.

Senior management teams aren't the only groups who become stuck and need help making decisions. Sometimes, a technical group has the same problem. Desmond, a database developer has an ongoing discussion with George, the GUI developer, and Tina, the tester about how to appropriately design the database upgrade for their product. Desmond, George, and Tina all agree they need an up-

grade. They can't decide how the upgrade should work. Depending on how they choose to implement the upgrade, their work will change, as well as the work the users will have to accomplish. Each of them has different ideas, and each idea is valid. They can't come to a decision, and they have only a week left to decide.

In both of these cases, well-meaning, intelligent people are stuck. Their normal ways of managing their disagreements are not working. Consider choosing a facilitator under these conditions:

- When a group has trouble coming to agreement on a strategy or set of actions.
- When you want to be part of the discussion and decision-making. It's not possible to treat the group fairly if you want to participate and facilitate.
- When you want to explore a previous project (retrospective facilitator) or explore alternatives (meeting facilitator)

You may be able to use people inside your organization as facilitators. Sometimes HR people or others are trained as facilitators. If you're not part of the problem context or solution, you can facilitate the decision-making.

Whatever you do, choose when you require a facilitator. Don't let the problems or conflicts escalate into no decisions, especially when you require a timely decision.

How to Improve Meetings When You're Not in Charge

© Esther Derby, 2004

Are you tired of attending endless meetings where the conversation goes in circles and nothing gets done? Even if you can't stand up and take control, you can nudge the meeting in the right direction from where you sit. Here are some strategies for improving the quality of meetings when you're not in charge.

Ask for an agenda ahead of time. When you receive a meeting notice, ask for an agenda. Make your request in the spirit of the best use of everyone's time: "Knowing the agenda will help me come prepared to participate." You can also say, "Knowing the purpose of the meeting will help me determine whether I can contribute."

Sometimes a request for an agenda is unsettling to the meeting convener... probably because he hasn't thought through the purpose of the meeting. Your request may prompt him clarify in his own mind why he called the meeting. If you're really lucky, he'll realize he didn't really need a meeting at all (it happens!).

Sometimes though, a meeting convener will insist that you must be there, even though he can't provide an agenda. I'm a little skeptical when the meeting convener assures me that I need to be there but can't articulate the purpose of the meeting. If you feel like it's not political suicide, tell the meeting convener that you can't assess the priority of his request against all your other work without an agenda.

Send only one person. Sometimes the person calling the meeting goes overboard to be inclusive. If two or more people from your group are asked to attend a meeting, check to make sure that all

perspectives are really required. In many cases, one person can speak to the interests of the group and you can send a representative. One person can go spend an hour at the meeting and then report to the rest of the group. Fewer people will make the meeting easier to manage and the people from your group who don't go will have a bit more desk time.

Politely decline the invitation. When you don't have relevant knowledge and expertise to contribute or won't be affected by the outcome of the meeting, bow out. Ask to be on the distribution list for meeting notes or other communication.

Offer to take notes. Some times when I talk to people after a meeting, it's hard to believe we were in the same room. Taking notes can help. As a participant you can offer to help by taking notes. Bring your flip chart paper and a marker and take notes so that all can see. Taking notes in public ensures that every one agrees that what is written is what was said.

Facilitate from where you sit. A well-timed question or comment has saved many a meeting. Here's a sampling of tactics that I use to facilitate from the back of the room. One word of caution about facilitating from the back of the room: Do this only if you genuinely want to be helpful. If you're feeling snide, it will come across in your voice.

Request a review of the agenda. When you arrive at a meeting with an overstuffed agenda, make a request to review and prioritize the agenda: "I'm wondering if we have time to cover everything we need to in the time we have. Can we please review and prioritize the agenda before we start?"

Ask for a progress check. When you see that time is getting short, ask for a process check: "I'd like to check on our progress. It's 1:50 and we still have three topics on the agenda. Can we prioritize them since we can probably only do one in 10 minutes?"

Help others participate. You can help the meeting when you help

others participate. If you see a quiet person trying unsuccessfully to break into the conversation, say "I think Jennifer has something to say." Don't force her to speak, but make an opening if she wants to take it. You can also help when a speaker is interrupted: "I think we may have cut Josh off before he had a chance to finish. Josh?" Then Josh can finish his thought if he wants to, and the interrupters will be a bit more aware of their behavior.

Rephrase. Sometimes rephrasing can help when someone is stuck on one point: "What I hear you saying is XYZ. Is that right?" Rephrasing helps people feel heard and can break the logjam.

Comment on what you observe. Sometimes it helps to comment on what you observe: "I think we've covered that already," can help get people moving again.

Summarize. Summarizing important points and decisions can help the group move forward. "Here's what I heard us agree to. Is that right?" Don't be upset if people disagree with what you've said – you've just turned up the fact that people really don't have a common understanding. Once you've surfaced the misunderstanding, it's more likely to be resolved before everyone leaves the room.

You may not be able to solve every meeting problem when you're not in charge. But you can help many meetings to run more smoothly. After a while, people may start following your cues: they'll write an agenda, pay attention to whom they invite and start attending to the interaction. And you'll look good, too.

Understanding and Creating Safety

© 2013 Don Gray

When I talk with other coaches about teams, I hear a lot about “creating safety” and “safe teams”. I don’t hear much about how to do that.

I’ve been flipping through my agile books looking for discussions about teams and safety. Many of the general books such as “Succeeding with Agile” by Mike Cohn and “Agile Software Development: The Cooperative Game” by Alistair Cockburn have sections on teams that contain very good information. But not a word about safety, at least by that word, “safety”. Maybe I don’t have the right books.

While debriefing a coaching simulation the 2010 AYE Conference we listed things coaches did and models coaches might use. Someone said, “Create a safe environment”. I replied, “And how do we do that?” And out came ideas and suggestions on how to do that!

The Wheelbarrow Test

The dictionary definition for safety (noun):
the condition of being protected from or unlikely to cause danger, risk, or injury

Since I can’t put a pound of safety in a wheelbarrow, this definition needs work to reduce the level of abstraction to something that can be measured. Most software development workplaces don’t involve danger or injury but we encounter risk, the possibility that something unpleasant or unwelcome will happen. For example ...

Long ago my manager and I decided it was time to defragment the RA81 on the production Vax 11/750. At that time, the process in-

volved making a tape back up, reformatting the disk, and restoring from the tape back up. The operators had the tape backups, so I just needed to wait until production completed for the day, reformat the drive, initialize the drive, and let the operator restore from tape. Things went to plan, until the operator told me, “Don, we don’t have a restore option available to us.” Suddenly, I realized the risk involved, and I didn’t feel very safe.

More recently a team I worked with needed to restructure a class that most of the application used. This required studying how the other 5 teams had used the class, refactoring the class and then testing the result to verify the application worked as expected. Certainty didn’t exist it could be done, or we should be the team to do the work, but if we didn’t deal with the risk now, it was going to grow and become more difficult to deal with in the future.

Safety – Take two

How about:

Safety means people can take risks and their coworkers/management will support them, especially if setbacks occur.

The above examples had happy endings. Other than me losing a lot of sleep that night, the disk was restored and functional when production started the next morning. The team successfully refactored the class and met their sprint goal. Not all risks have happy endings, and that’s when we need support.

Extending the thought that we can take risks means we can express our opinions. Esther Derby talks about this in *Workplace Safety* (and *We’re Not Talking OSHA*). There she defines safety “as the ability to speak your truth without fear of ridicule, rejection, or retribution.”

Combining the two we arrive at:

Safety means people can take risks and speak their truth without fear of ridicule, rejection, or retribution.

Safety doesn’t automatically happen. It requires support, practice,

and patience. Having a common definition for safety with examples provides a starting point.

But how can we measure safety in the workplace? What does it mean to productivity?

The Safety Check

The Safety Check activity provides an understanding of how safe people in a group feel. The group may be an interdependent team, peer managers solving problems, or perhaps a status meeting including the team, its manager and their director. Each scenario presents different dynamics and you may want to investigate how safety shifts between them.

The Safety Check

1. Introduction – Set the stage for the activity. You'll probably use a slightly different description for a team brain-storming session than a multi-level status meeting. Briefly explain the process.
2. Prior to the meeting create a flip chart that contains 5 safety levels. The levels range from completely safe, to completely unsafe. Put the flip chart on a wall and share the descriptions for the 4 – 0 values.

This might look like:

Value	Description
4	I can share anything I want to at this time.
3	I can share almost anything. I may need to be a little careful.
2	I might share something if it doesn't seem out of line w/ the discussion.
1	I'm probably going to observe. If pigeon-holed, I might say something.
0	Nope. No way I'm sharing anything w/ this group at this time.

Be sure to change the descriptions to fit your context. 3. Handout ballot – some indistinguishable piece of paper. 4. Ask the participants put their safety level on the ballot and fold the ballot in half. Tell them you'll collect the ballots, create a chart, and you personally will dispose of the ballots. 5. Collect the ballots in a hat or other container 6. Create a histogram that shows the ballots – it might look something like this:



The Safety Check measures our truth, our safety. We all have different values for “what makes me feel safe”. Your values apply to you. My values apply to me. As we build the histogram we get a view of how safe we feel. This leads to an understanding of how much risk we’re willing to take and how likely we are to share our truth. Doing the Safety Check first tells us how safe the team feels and if we need to continue with secret ballots.

Knowing how safe people feel starts the data collection.

What Put the Number Where it is?

What has happened that put your safety number where it is? Does your manager yell at you when she doesn’t get the results she’s

hoping for? Maybe you're new to the team and not sure how to work with the others? A team member runs to the manager when things don't go their way? Arguing and verbal fisticuffs during planning sessions? The examples I offer seem negative. They could have as easily been positive examples.

By gathering this data, team members and managers discover the behaviors that hinder or help with safety.

What Does Low Safety Cost?

A technique I learned from Esther Derby sheds light on how low safety affects performance and innovation. To do this activity ask people to fill in the blanks in these sentences:

"When I feel safe, I can _____." "When I don't feel safe, I _____."

The answers generate very interesting results and make the costs associated with unsafe environments visible.

What Would It Take?

By now we know how safe the team feels, how it got that way, and what costs associate with that environment. This next question becomes part of the solutions focus.

"What would it take to raise your number 1 level? If you number is a 1, what would it take to get you to a 2?" Having a list of possible changes provides a path to improved safety.

Creating Safety

Most of the time, most of the people don't put a lot of thought into their behavior. They run an "auto-pilot" program that governs how they respond. If we examine the elements that generate behavior, we get something like this list:

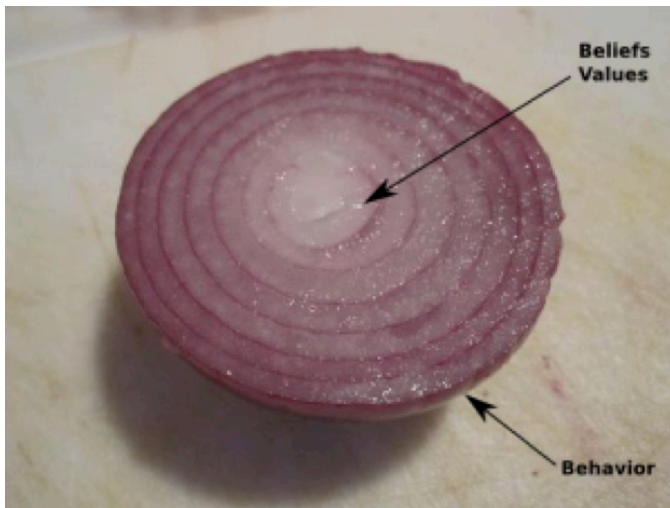
- Physical sensations – information coming into us from "the real world".

- Beliefs and Values – concepts and things important to us
- Feelings – both physical and self-esteem
- Data/facts – information I “know”
- Thoughts – What we do with the data/facts
- Intuition – Hunches based on experience without supporting data (at this time)
- Concerns/Fears
- Desires/Requests – What I want to have happen

If we stopped and consciously processed each step all the time, civilization would screech to a halt. How can we create a safe environment and still create value for our clients?

Beliefs and Values – The Invisible Elephant

We don't walk around the office starting conversations with “Well, I believe that ...” or “Given my values ...”, yet our beliefs and values form the seed for our behavior.



We get some of our values and beliefs from our environment. The culture and family we're part of greatly influence us. Strong

hierarchical relationships make sense in some cultures but seem stifling and ill-informed in cultures that value independence. Another source involves our experience. Behaviors that result in positive outcomes get reinforced and converted into beliefs about how we should behave. Having been part of a lot decisions I have a belief that including everyone's ideas about the decision helps create a better decision.

With a list of values and beliefs we can know what is important to our team, but this list doesn't tell us how to behave.

Team Norms – The Team's Guard Rails

Moving one step closer to safety brings us to team norms. Team norms tell us what behavior we expect from the group. Team norms for getting everyone's ideas might include:

- Respect each other's views.
- Let everyone have a say.
- Don't interrupt while someone is talking.

With this list team members can compare behavior with how the team expects its members to behave.

I've seen these lists referred to working agreements, norms, ground rules. As guard rails they serve to keep behavior in some range and allow team members to point out when a member is not staying within the range of acceptable behavior.

Simple rules – Generating Patterns of Behavior

Like team norms, we create simple rules based on the team's beliefs and values. Simple rules provide guidance for decisions and action when situations are less than predictable. Simple rules generalize to fit most situations eliminating the need for an exhaustive list of acceptable / unacceptable behavior. A simple rule for the getting everyone's ideas might be:

- Seek first to understand, then be understood.

Some guidelines for simple rules include:

- Keep rules to “minimum specifications”.
- Rules should be both generalizable and scalable.
- Each rule should begin with an action verb and be stated in the positive.

The list should be short, seven to nine as a maximum. The list should include at least one rule for these general areas:

- who we are as a group
- how the team will deal with differences
- how the team will exchange ideas and information both within the team and with the organization.

Creating simple rules allows team members to act accordingly and know the team will support them.

Collaboration: It's more than facilitated meetings

©2007 Esther Derby

I've noticed something lately: when people write about collaboration, they discuss facilitated meetings. Well-run meetings that encourage participation and building consensus are certainly valuable, yet there's more to collaboration than meetings. True collaborative assumes shared responsibility, shared ownership, and boosts creativity and learning.

Shared Responsibility

Collaborative teams focus on meeting a shared goal. Everyone's skills and contributions are needed to reach that goal. In the end, they'll either succeed or fail together. No one can say, "I got my tasks done. It's your fault we didn't meet our goal."

Shared Ownership

A colleague told me a story about a co-worker, Bob, who was very protective of his code. If anyone else attempted to refactor or add to Bob's code, he'd pitch a fit and remove all the changes. The result was that Bob was the only one who really knew the details of that module, and all features and fixes had to funnel through Bob. That may mean job security for Bob, but it's risky for the business.

With a team goal, there's no room for "my code" or "my tests." The team shares ownership for its products. Individuals still may know

a part of the system in more depth than others, but everyone has an understanding of the overall system, and that keeps the “truck-factor” risk low.

Cross-functional Learning

Shared goals, shared responsibility, and shared ownership drive cross-functional learning. Josh, a tester, applies the Java scripting skills he uses for test automation to write code for the GUI to help the team meet its goals. When the team members are worried about meeting their goals for test coverage, everyone pitches in to write automated unit tests. Team members learn from each other and expand their skills outside of their functional silos.

Process and Organizational Learning

When teams hold retrospectives, they share their perceptions about technical practices, work processes, and teamwork. Team members learn about the specific topic they are discussing and also about process analysis, process improvement, and influencing change.

Deciding Together

In business, we search for the “right” decision and often rely on experts to reach decisions for the rest of the group. While that’s the right approach some of the time, other times it’s more important to have decisions that the entire group will support and move forward. It’s a matter of looking for a good decision that the team will support rather than the “right” decision that doesn’t go anywhere.

Thinking and Solving Problems Most people in the software field are really good at thinking and figuring out problems alone. Col-

laborative teams make use of each member's insights and ideas to generate multiple solutions and then pick from the best. Looking at problems from different perspectives drives creative solutions. When I listen to teams that are starting to solve problems collaboratively, I hear comments such as "I never would have thought of that" and "Your comment sparked another idea."

Creating Together

As with solving problems, many heads are often better than one when creating. Different people see different visions of what's possible. Team members build on others' ideas and challenge each other to fill gaps and weaknesses in solution candidates.

If better results, lowered risk, and increased learning aren't enough, collaboration is also more fun.

I recently spoke to a domain expert, Sally, who was working with a software team to define features for the company's main product. "After finishing customer research, I used to work on my own to create the perfect feature description," she said. "Inevitably, I'd miss something or write an ambiguous description. Sometimes, I'd find out later that what I described was technically possible, but expensive. Those discussions always led to finger pointing. Now that the team and I are working on the features together, we catch those things early. They have good ideas that I wouldn't have thought of, and we're having fun."

When teams collaborate, there's less pressure on each individual to create the perfect part. Everyone realizes that she has the help and support of the rest of the team, and together they'll create something far better than each could do on her own.

Collaboration Gone Bad

In spite of all these benefits, some people shy away from collaboration. Collaboration isn't for everyone. Some people prefer to work alone and do their best work solo. Trying to force collaboration on people who aren't interested is an exercise in futility (and not very collaborative). Some might have had a bad experience and are wary of trying collaboration again.

Just as collaboration isn't a good fit for every person, collaboration isn't a fit for all work. A work group responsible for installing and configuring servers was urged to work as a team and "be collaborative." The group's work was queued by tickets that specified the setup for each server. Each setup was a one-person job. While there were occasions when it made sense for people in the group to work together, most of the time there wasn't a shared goal that required the effort of more than one person. Their manager decried the lack of collaboration—not seeing that the work didn't require it—and the group members felt their boss was unfairly berating them.

A weak goal can undermine collaboration and lead to churn. One company convened a cross-functional team to "redesign the organization." Each team member had a different idea of the scope and authority of the group. After several weeks of argument and aimless conversation, group members quietly drifted off to more focused and satisfying work. They never reached the point of collaborating because they didn't have a shared idea of where they were going.

A team needs the right mix of skills to collaborate, too. If the group lacks key skills, they won't be able to get the work done, no matter how well they get along and how creative and cooperative they are.

I've also seen collaborative teams struggle when one person didn't follow through on commitments or persisted in working as a lone wolf. Teams that have the skills to do so may hold the person to account or coach her off the team. However, if the team isn't ready

to manage its membership and the coach or manager doesn't step in, that could spell failure for the team.

Also, when the team members have a shared goal but are rewarded and rated as individuals, people have less incentive to cooperate. Depending on how people are rewarded in the company, they may be incented to compete with each other or even undercut each other. Some collaborative teams survive in organizations that stack rank and otherwise foster competition; these teams need some other motivator to overcome the factors that demotivate collaboration.

And collaboration requires specific skills—skills that we don't normally learn in school. Dealing with conflict constructively, giving peer-to-peer feedback, and reaching sustainable agreements are all developmental areas for collaborative teams.

By all means, work to make meetings more effective by using participation and facilitation. But don't forget the other parts of collaboration: shared responsibility and ownership, learning, creating, thinking, and deciding. When we hone our collaboration skills, all of us are smarter together than any of us is alone.

Part 7: Problem Solving

Always Be Second

Reasons

What's on Your Not-To-Do List?

Staying Sharp

The Exception is the Rule

Shifting the Burden - Whose Monkey is it?

Solving Other People's Problems

Always Be Second

© 2002 Gerald M. Weinberg

These days, with all the talk about “internet time,” professional workers are always trying to be the first with new ideas. But is that really the only path to success? Is it, indeed, a very effective path at all? What about being second?

Our culture certainly encourages people to be “first.” In school, the emphasis ranges from who raises a hand first in class to who graduates with the highest grade-point average. The valedictorian - first in the class - gets lots of honors, but the salutatorian - second in the class - has to make the speech (phooey!).

Sports competitions also honor the “winner,” and books on problem solving and business often borrow this metaphor - emphasizing finding the “winning” idea before anybody else. But most problem-solving areas of life are not, in fact, like sports. It’s not having the idea that matters, it’s what you do with the idea that counts.

And, in fact, ideas don’t count that much in sports, either. Anyone can have the idea, “Hit a home run now and win the game,” but there’s a lot more to hitting home runs than simply having the idea. To have a fair chance of hitting a home run, you have to practice, practice, and practice. As in most of life, the key issue is not who is there first with the idea; the key issue is how you develop the idea once you have it. It’s not your ideas alone that win, but ideas plus capability and the will to develop your ideas.

If you think having ideas first is the issue, consider this. Every year, tens of thousands of individuals start businesses based on being the first with a “new” idea. If that was all it took to be wildly successful, many of these firms would quickly displace all of the giant firms and become giants themselves, every year! Yes, we all know stories of

small firms that had a better idea and indeed succeeded at becoming giants, but most of the time, the giants persist and the midgets fall humbly by the wayside.

Giant companies survive the onslaught of little start-ups with one of two strategies:

1. Exploring many parallel ideas and developing the few good ones.
2. Letting others do most of the searching for ideas, then developing the few good ones. What can the individual professional learn from these strategies?

Exploring many parallel ideas depends on having huge amounts of capital, which the individual professional cannot manage. But even giants couldn't afford to have so many explorations in the pipeline if they didn't know how to kill the ones that show no promise, and kill them early. So, for the individual, the lesson of this strategy is to learn how to let go.

For me, letting go means not trying to maintain leading-edge competence in every field in which I was once competent. I used to be one of the world's great IBM 704 assembly language programmers, and I suppose there are still some jobs maintaining such code, but I don't think I'd want them anyway. I've let go of that one, along with dozens of others.

Letting go also means that I don't try to pursue every new fad I happen to hear mentioned at a conference or on the web - which leads me into the second strategy - letting others do most of the searching for me.

In a field where capital costs are low (like software development), chances are better that some small firm will come up with the good idea first because there are tens of thousands of such small firms out there. In that case, the big successful companies don't rely on coming up with every good idea first. Instead, they pursue

an “always be second” policy, and support it. They maintain some minimal competency in a wide variety of areas, so that they’ll readily recognize one of these good ideas when it comes along - from another company, or often from one of their customers. Then, they’ll either copy the idea, buy the rights, or buy the company.

I’m not able to buy companies, or even their licenses, but I am able to learn from them. And, under some circumstances, I can ally with them. That way, even a single individual can pursue an always-be-second strategy. To be a great second, I use a process that resembles the way a breeder produces improved plants and animals, with three essential elements: variation, selection, and retention. Variation provides the new ideas; selection weeds out the ones lacking promise; and retention propagates the ones that survive the selection.

Variation (new ideas) for an individual, comes from using a network to supply information from literally hundreds of sources. By communication with people in my network, I can be sure that few new ideas escape my attention. I have created a huge virtual organization, far larger than any real organization I could afford. The AYE Conference is a central element of that network, where I can meeting many of my “nodes” face-to- face once a year and exchange ideas and experiences.

The network provides some of the selection, too. First of all, any idea that nobody passes on to me is unlikely to be a worthwhile idea. Conversely, when I hear of the same idea from three separate sources, I raise its “score.” Of course, in order for this selection process to be meaningful, I have to consider the reliability of my sources. For example, if I hear an idea supported enthusiastically by certain people, I reduce it’s score. I work hard to keep my network alive and healthy so I can depend on it to do a good job of variation and selection. Ultimately, though, I have to rely on myself to select which ideas are worth retaining and developing, but the whole process is designed to keep me from developing “neat things” that

don't fit the market.

I don't know how to automate the retention and development of ideas, so it takes a lot of work, and I can't afford to develop very many new ideas the way a large organization might. Sometime I can share work by co- developing ideas (seminars, books, articles) with colleagues from my network. But ultimately, I have to develop in my own way, to create the unique service I can sell to my clients. This is the place where it pays me to be first - I may be satisfied being second in the race to market, but I always try to be first in quality.

Reasons

© 2001 Gerald M. Weinberg

[Note: In September, 2000, at the SEI's Software Engineering Symposium in Washington D. C., Jerry was the recipient of the 2000 Stevens Award. The award recipient is recognized for outstanding contributions to the literature or practice of methods for software development. In receiving the award, Jerry gave the Stevens Lecture on Software Development Methods, the purpose of which is to advance the state of software development methods and enhance their continuing evolution. The following column is abstracted from part of his Stevens Lecture.]

I remember one summer day during World War II when my father, Harry, and I were discussing the water restrictions that had been imposed because of drought conditions. I was worried that we might run out of water, and my father said, "Yes, it could happen that we run out of water. Lots of things can run out - water, sugar, meat, gasoline, bread, even air. But there's one thing that will never run out."

"What's that?" I asked, looking for reassurance.

"Reasons," he said. "People will never run out of reasons."

Harry was the first change artist I knew. At that time, he was working for Sears, starting up new stores and installing new procedures in existing stores. He often took me on trips, and I got to see him working. I particularly recall the difficulty he had teaching salespeople about Sears' policy on returned tools and paint.

The policy was simple: The customer was always right, so just replace anything that they didn't like, no questions asked. If possible, find out what they didn't like about it, but never, never question their right to a replacement. I'd watched him coach salespeople

through a number of these transactions, so I thought I understood what he mean by never running out of reasons. Even though the policy was clear - replacement with no questions asked - the customers always had elaborate stories of why the item wasn't satisfactory.

But only later did I realize that it wasn't just the customers he was talking about. He was also talking about the salespeople, who never ceased to have reasons why they couldn't or shouldn't apply the policy in each particular case.

Recently, I found myself recalling that summer day, half-a-century ago, when a client asked me to find out why their Software Engineering Process Group was having so much trouble getting people to adopt new software tools. It couldn't be the tools themselves, they reasoned, because quite a few people had adopted them and liked them. So, I set out to interview both adopters and rejecters, to discover the reasons some were using the tools and some were not. Here are some of the answers I obtained:

Darlene: I installed it because the boss told me to use it.

Porter: The boss told me to use it, so I didn't use it.

Ursula: I installed it because the boss forced me to use it.

Marcy: The boss forced me to use it, so I installed it, but I don't use it. He wouldn't know the difference.

Quentin: I used it because it was like what I used before, so I knew I wouldn't have any trouble adapting to it.

Chuck: Why should I use it? It's nothing new; it's just like what I used before.

Carl: Hey, I used it right away, because it was new and different.

Cynthia: I'm not going to use anything that's new and different. Too many things aren't tested, and something's sure to go wrong.

Mary: Of course I used it. Everyone else was using it.

Roy: Everyone else was using it - what a bore! You won't catch me following the crowd.

Frances: Why should I use it? Nobody else was.

Edgar: Hey, I got to be the first one to use it!

Mort: I couldn't use it. It didn't do all the things I needed.

Alan: The thing I liked best about this tool was that it didn't try to be a Swiss army knife and do everything anyone could possibly want.

Gerri: It was the perfect tool, because it had every feature I could possibly want.

Chico: Every time I hit a key by accident, it would invoke some obscure feature that I didn't want in there in the first place. Finally, I trashed the whole thing.

Orion: I'm so busy, I needed a new tool to save me some time.

Belle: I'm so busy, I don't have time to install and learn a new tool.

May: I'm not that heavily loaded. Why would I need a time-saving tool?

Paul: Well, I wasn't so busy with other things, so I had time to install and learn a new tool.

Earl: It was freeware, so it was a bargain.

Justine: It was shareware, so it couldn't have been any good.

Jacob: This tool costs \$3,000. It must be good, so I'm using it.

Neelie: I'm saving the company \$3,000 by not using it.

Willis: I won't use it because I don't like the way Microsoft makes software.

Samuel: I knew it would be good because Microsoft makes it.

Well, there were more, many more, but that's enough of the infinite reasons to make my point. By this time, you may have noticed that I have arranged these reasons in pairs. Why? So you could see the pattern that I saw:

Every single reason to use the tool was matched by the same reason for not using it - and vice versa!

In other words, these reasons may look like logic, but they're not logic - they're just reasons. In logic, the reasoning comes first, then comes the decision. But in real life, it's usually the other way around - first we make the decision, then we make up whatever reasons we need to "justify" the decision and make it look like logic.

And why would we go to all this trouble? Somewhere along the line, we've learned that emotional reasons aren't good enough, so we have to fool people into thinking we're more rational than we really are - so they'll leave us alone to do what we intend to do in the first place.

I recently read about a fascinating psychological experiment conducted in a large office at the high-speed copier. When there was a line of people waiting to use the copier, an experimenter would walk up with a handful of papers to copy and try to butt into the front of the line.

In one set of trials, the experimenter would say, simply, "I want to copy these." People were indignant, and refused to let him get in front of them. In a second set of trials, the experimenter would say, "I want to copy these because my boss needs them." People then willingly let him go right to the front of the line. In other words, if the experimenter had a good reason, then people were willing to let him go first, even if meant they had to wait longer.

But the fascinating phenomenon was in the third set of trials, where the experimenter would say, "I want to copy these because I need to copy them." And guess what? The people let him go to the front, just as they had done when he had a "good reason," even though this

“reason” was totally vacuous. In repeated trials, the experimenters discovered that any “reason” would work, as long as they used the form, “I want to copy these because X.”

We seem to be conditioned to respond to this kind of pseudo-logic, and we instinctively know how to use it. Imagine telling your boss, “No, I’m not going to use this tool.” You know that the next thing you’re going to hear is “Why not?” So, you’re never dumb enough to “just say no,” but, instead, you’re going to say, “No, I’m not going to use this tool because X,” where you supply the X from the list above or from your own favorites. You might get an argument, but half the time the conversation will simply end there. And, if the boss does continue, you’ll then supply reason Y, then Z, then A, B, C, and D, until she gets tired of the game and quits.

But I’m not telling you all this so you can beat your boss in The Big Game of who gets to tell who what to do. I’m telling you because one of these days - perhaps today - you’re going to be on the other side of the equation. You’re going to be the change artist trying to introduce something new - a tool, a process, a document, a technique, anything new at all. And when you try, you’re going to find yourself faced with an infinitely high wall piled with reasons, mortared in place with that word, “because.” [Or “so,” or other forms of pseudo-logic.]

And what will you do then? Rather than go back and forth with a potentially infinite chain of “why-because,” save yourself some time and energy by recognizing that you will always lose this game, so switch to another. One of these new games is to try to convert to real logic.

When you get the first “because,” simply say, “You’re right. There are lots and lots of reasons why you might not want to do it this way, and you’re exactly right to start raising them. Let’s carry this through.”

Then you take out a sheet of paper and draw a line down the middle. On the left, you label the column “Reasons Why Not.” On the right,

“Reasons Why.” Then you write their reason in the left column and ask for one in the right column. If they can’t come up with one, prime the pump by showing how their own reason could just as well go on the right.

Continue filling out the columns until you have 6 or 8 or 10 reasons on each side, then say, “Okay, now let’s consider what’s the real problem here.” And off you go, possibly getting into real logic for a change.

A second approach is to short-circuit the game right at the beginning. When they start listing “why-not” reasons, you interrupt and say, “You’re right. Not everybody is right for this tool. Since you don’t have the right qualifications, I’ll go look for someone else.”

Sometimes, they’ll let you go - and then at least you’re saving time. But sometimes, they’ll stop you and start giving reasons why they are, indeed, the right people for this tool. And you can be sure they’ll have an infinite number of them.

What's on Your Not-To-Do List?

© 2005 Johanna Rothman

If you're like most of my clients, you have too much to do. Recently, an Engineer Director, Stephanie, explained all the things she "had" to do: monitor the projects, participate in the requirements sessions, draw up a yearly budget, write three performance evaluations, monitor training classes, visit customers, assist technical support and more. Some of the items on her list were clearly her responsibility, but others were not. Her not-responsibilities were preventing her from doing her necessary work. Stephanie needed a to-do list and a not-to-do list.

Here are some signs that you need to create a not-to-do list:

- You're always working overtime.
- You never have time to sit and think.
- You manage crises well, because that's the only kind of management you ever do.
- You're the bottleneck for work that's not getting done on your project.
- Papers sit in your inbox for weeks, and you're hopelessly behind in your email.
- You take refuge in the technical work because you're uncomfortable with the management work you're supposed to do.
- You get depressed looking at the ever-growing pile of paper on your desk.
- Your to-do list is a write-only list; things go on but never come off.

- You are doing more work but your manager is less satisfied with your performance.

Your not-to-do list is the list of everything that's not part of your job to address or solve directly. Here are some suggestions for creating a not-to-do list:

Clarify your role. Try defining your job description with a one-line sentence. What does your company pay you to do? How broadly are you interpreting that? Should you make it narrower? One Director of Engineering said, "I'm responsible for the care and feeding of the engineers and the product." He stopped allowing the salespeople and the support staff to lean on his organization, by putting training and other procedures in place.

Work at your level. Check to see that you're working at the level appropriate for your title. Many people retain pieces of previous work out of habit after a promotion or a reorganization, rather than working at the most appropriate level. If you're now in middle management, think hard about whether you should be doing technical work. If you are doing technical work, who's doing the strategic thinking?

Make decisions when you add value. When it's time to make decisions, check a couple of things: are you the right person to make the decision? Are you too tired to think straight? Will you add value to this decision? If you're the right person to make the decision, clear your head, think the problem through, and then make the decision.

Delegate or manage your paperwork. If you need administrative assistance, or filing assistance, or other paper handling assistance, ask for it. Then, deal with the rest of the paperwork. If you don't know what to do with your humungous pile, throw the whole thing out. If someone needs something from you, they'll be back with another piece of paper.

As you decide what you should do and not do, don't just drop things. Explain your decisions, possibly by negotiating with the

people affected by your decisions. And when you make your not-to-do list practice living with it for a few weeks. See if other people are working on handling the items on your not-to-do list. If no one's working on that list, does it matter, or will your project risks increase? If your risk increases, explain which work you're not going to do to manage the risk.

If you're working on issues that belong to other people, stop. Draw your boundaries, and make the rest of the people in your organization live up to their job descriptions; don't just take on more work.

Staying Sharp

© 2003 Gerald M. Weinberg

I'm not the kind of person who hangs out in nightclubs. In fact, the last nightclub I can remember visiting was in Miami Beach in 1957. What I remember about it is what the stand-up comic said.

After warming up the audience with some rather gross remarks, he commented that early in his life he had learned the motto he had lived by every since: Sound mind; sound body ...â€œ... Take your choice!

How funny to hear it articulated so clearly, but many of us did make this choice early in life. Somehow we got the impression that athletes are stupid and software developers are flabby - and that we must make choose one or the other. Actually, though, a reasonable level of physical health increases the effectiveness of my intellectual work. Increased effectiveness then produces more slack time in which I can pursue healthy practices. So, good health tends to produce better health, at the same time that it produces better mental health.

But this syndrome works both ways. Poor health tends to produce poorer health by diminishing work effectiveness, which in turn causes work to pile up. Piled work causes me to overwork, consume junk food in haste, and generally ignore my physical well being. Eventually, my health becomes even poorer, and the cycle continues unless I can manage to break it in some way. I become, literally, stupid - "in a stupor; deficient in alertness; lacking in the power to absorb ideas."

But this kind of brain dysfunction is merely the grossest kind - akin to the effects of being struck on the braincase by a piano leg. The brain is a complex problem-solving device whose functioning we

still only vaguely understand. We know that the piano leg will put the brain out of commission, as will sickness. But we also know that a computer can be put out of commission with a sledgehammer, or by pulling the plug. What interests me now is some more subtle elements of my brain. Those subtle elements make people want to hire me as a consultant, treat me like royalty, and pay me large sums of money.

My interest in subtle brain factors drew me to reading an article about “personal chemistry.” The author’s list suggested some of these success factors:

Articulate: writing and speaking fluently in at least your native tongue.

Thoughtful: weighing a question for a few seconds before responding.

Bright, informed, sparkling: difficult to define, but obvious if a person doesn’t have it.

Breadth of interest: able to carry on an intelligent conversation without permitting embarrassing gaps because of lack of interest or education. Unfortunately, the author seemed to suggest that you can somehow wipe a veneer of “chemistry” over your otherwise dull, boring self. For instance, he says, “brief reflections give the impression that you have good judgment” - not good judgment, but the impression of good judgment.

At this level of analysis, brain chemistry consists of a set of rules. For example, “count to three before you answer a question, so people will think you are thoughtful.” In the typical steamy working environment I usually encounter, however, this kind of veneer peels quickly, revealing all my ugly lumps and hollows underneath. No, if I truly want to be more articulate, thoughtful, bright, informed, and sparkling, rules won’t suffice. I have to devote some time and effort to the job.

My acquaintances who don’t work with computers tell me that

software people are the dumbest people they know. I have a hard time believing this assertion. We all know that computers aren't dumb - they are an endlessly fascinating subject. But let's face it. There is more to life than computing, and more parts to our brains than those we use in our professional work.

At AYE Conferences, I've repeatedly seen that problem-solving behavior becomes stereotyped when people work in a closed situation. Once they find one or two tricks that work well, they tend to adopt those to the exclusion of all others. I wish we presenters could take more credit, but most of the effectiveness amplifying that takes place at AYE seems to come from exposing the participants to the problem-solving styles of other participants.

My consulting problems are growing more difficult. Systems are growing more complex; needs are growing more demanding; because of past successes, my expectations run high. If I remained at the same level of problem-solving effectiveness, I'd soon accumulate a deadening backlog of unsolved problems. With a little slack time, I have some possibility of "outside" activities that stimulate those parts of my brain I don't ordinarily exercise at work. Without such activities, my problem-solving effectiveness would grow ever more narrow and specialized. New problems would then become unsolvable problems.

My brain is a muscle. Like any muscle, it requires stimulation to remain healthy. If I'm locked into a pattern of work, work, and more work, my brain soon stagnates. Paradoxically, if I want to be more effective at work, I must be less single-minded in my devotion to work. Anything I do that stimulates new segments of my brain will make me a better programmer, or tester, or analyst, or manager, or writer, or consultant.

Many technical folks, seeking this kind of stimulation, enroll in university courses. Some are successful, but some are not. Perhaps the course is dull - not stimulating at all - yet they persist because their employer is paying the tuition and they are embarrassed to

quit.

Or, the course may be too “relevant” to their work - more of the same bland diet they consume every day on the job.

If you want to keep your brain healthy, you might do better seeking your stimulation outside the formal education system. For instance, change your TV-watching habits, not necessarily to something more “intellectual.” Or, if you never watch TV, a little tube time might prove a stimulating change. If you don’t read anything but manuals, pick some paperback at random on the way home and read it - but stop if it’s dull.

If you read frequently, read something different. Or, stop reading for a few days and just open your eyes and ears and nose to the world around you. I find that natural settings always make my brain sparkle.

If you must attend courses or conferences, participate in something your employer would never pay for. That way, you can quit if it’s dull and move onto something healthier for your brain. Sound mind; sound body - it’s not a choice, it’s a mandate.

The Exception is the Rule

© 2005 Gerald M. Weinberg

The other day, I was trying to help a client (let me call them “StartupCompany”) mired in conflicts, exceptions, errors, anomalies, lapses, modifications and other deviations from the norm. These annoying exceptions were playing tricks with my blood pressure, so I had to be wired to a wearable blood pressure computer for twenty-four hours. As if StartupCompany didn’t have enough interruptions, now my wearable computer was inflating a blood pressure cuff at random intervals throughout the day.

Every time the cuff inflated, I petulantly asked myself: Why can’t they run a project like real people living run-of-the-mill, low-blood-pressure lives? That night, I was using the Yellow Pages, and in the A categories in the Yellow Pages index, I chanced to notice a curious pattern. Here are the first few items:

Abortion Services and Alternatives. These were the first two entries in the index. I decided to skip them both, so as not to take sides in the pro-choice/pro-life conflict. I had enough conflicts within StartupCompany.

Abuse - Men, Women, Children. I decided to continue my scan of the index, and this was the next entry. The normal process of family living involves people loving and respecting each other, communicating well, and behaving appropriately according to societal norms. But when people start behaving inappropriately, they need Abuse Services. In StartupCompany, people normally respected one another, communicated well, and behaved appropriately according to societal norms. But they sometimes didn’t, and they lacked “abuse services” for coping.

Academies (including private schools and special education).

When the formal education system doesn't provide special knowledge or handle special cases, private academies and special education are called for. People within StartupCompany often needed to know things they hadn't learned in the public schools, but StartupCompany had no provision for special education.

Accident Prevention. Accidents aren't "supposed" to happen, StartupCompany had accidents. In order to improve, they needed processes to prevent accidents and to mitigate their consequences.

Accordions. Despite what some people think, accordions are perfectly normal, though not everybody learns to play them or appreciate them. Still, StartupCompany could have used some entertainment to lighten the mood once in a while.

Accountants. Accounting is also normal, but, if everything always went according to plan, we wouldn't need to account for things so carefully. We have to protect our financial well-being from mistakes and misbehavior, and that's what accountants do - and also what they should have been doing in StartupCompany.

Acetylene Welding. Some welding is normal, and some is for repairing things that are not supposed to break - but do anyway. StartupCompany lacked a "welding team" to handle lots of stuff that broke.

Acrylic Nails. Most normal people have fingernails, so why is there a nail business? Oh, yes, it's the human interface, and StartupCompany had to cope with conflicting ideas of what made a system beautiful - but they had no special beauty experts to resolve the conflicts. Acting Instruction. We all need to "put on an act" now and then when we're caught by surprise. StartupCompany's people certainly needed training in how to behave in improvisational situations, but there was no acting instruction.

Acupressure/Acupuncture. If we were all healthy all the time, we wouldn't need medical services, and if "normal" Western medical services worked all the time, we wouldn't need acupressure and

acupuncture. So, there are not only abnormal services, but meta-abnormal services - the services when the normal abnormal services fail - certainly true in StartupCompany.

Addressing Service. Have you ever tried to maintain a mailing list? Almost all the work is not the mailing itself, but maintaining the addresses. It's even worse for email, because email services haven't yet evolved "normal" ways of dealing with changes. Gee, neither had StartupCompany. Adjusters. Adjusters, of course, are an abnormal service from the get-go. Without accidents, we wouldn't need insurance, and if things stayed on course, StartupCompany wouldn't have needed risk analysis. But they did.

Adobe Materials and Contractors. Adobe materials may not be "normal" where you live, but here in New Mexico, adobe is a normal building method. StartupCompany, too, has its idiosyncratic processes that are not normal in other projects - and newcomers have to learn about them or pay the price. But StartupCompany had no special services to bring newcomers up to speed.

Adoption Services. Yes, sometimes people are not wanted by their parents, and StartupCompany certainly had some unwanted people. But, they lacked "adoption" services for moving unwanted people around.

Adult Supervisory Care. "Normal" adults can take care of themselves without supervision, and normal workers wouldn't need much managing at all. But StartupCompany had two adults who could not take proper care of themselves, and the managers spent an inordinate amount of time on these two out of a hundred.

I stopped there, sobered by my reading. It was now clear to me that StartupCompany, being a startup, had an overly simplistic picture of what it takes to run a company. I needed an adjustor to adjust my blood pressure - I needed to see that my job as their consultant was to teach them that deviations are normal, and that they (and I) could do what real people do:

- stop whining and deal with them
- create systems to deal with them
- create systems to prevent them

Shifting the Burden – Whose Monkey is it?

© Don Gray, 2005

“Repeatedly curing a system that can cure itself will eventually create a system that can’t.” - Marvin’s Second Great Secret, Jerry Weinberg

“Don, the software’s locked up again! Can you come up here tomorrow and fix it?” George was on the other end of the conversation. George and I had started working together when his employer moved a production line from Florida to Virginia. This move created all sorts of problems¹. The daily struggles getting the hardware, software and process playing nicely together had become a weekly check-in with an occasional on-site visit. This made the request seem a little odd.

Taking a deep breath allowed me to work through some quick thoughts². The software was running on 15 different computers. There wasn’t any way they could all lock up at the same time. And what does “lock up” mean anyway? What’s going to happen different between now and tomorrow, or now and next Monday? And if I drop what I’m currently working on to rush up there tomorrow, am I really helping George? What long-term consequences result from this?

Solution #1 – The Quick Fix

Accepting that a problem is an undesirable difference what we want and what we have, George had a problem. Since I helped George with several other problems, it seemed natural that I should help with problem also. Using a causal loop diagram, the situation looks like this:

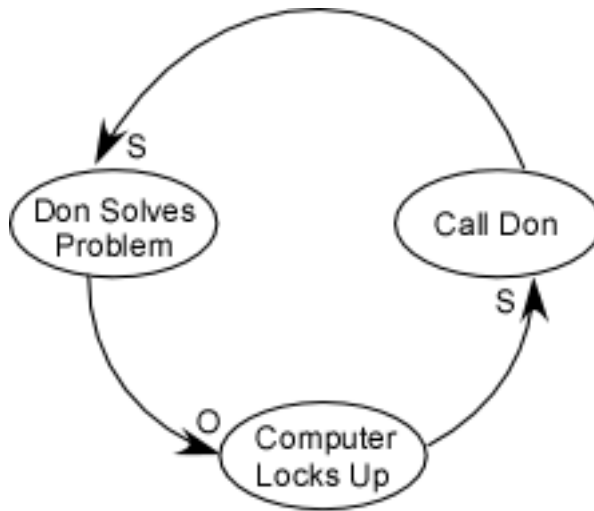


Figure 1 - The Quick Fix

The more the computer locks up, the more Don gets called. The more Don gets called, the more problems he solves. The more problems Don solves, the less the computer locks up. This is a balancing loop that brings stability to George's process. It's also a symptomatic solution, a quick fix to make the pain go away. But following the steps in B1 only solves the immediate problem; it does not solve the underlying fundamental problem. Think of it as treating the symptom, not the disease.

Solution #2 – George Learns to Solve the Problem

There's another possible answer. What does it look like if George solves the problem without calling Don? First George needs to learn "C" much better than he currently knows it. That takes time. This time delay gets represented by the "||" mark on the line. Then he needs to figure out where in the 5000+ lines of code the problem resides. That also takes time. Finally George can solve the problem, but by now something else is broken worse somewhere else, everything gets put on hold, and when George finally gets back, he's lost his train of thought, and has to start solving the

problem again. Diagrammatically we represent it this way:

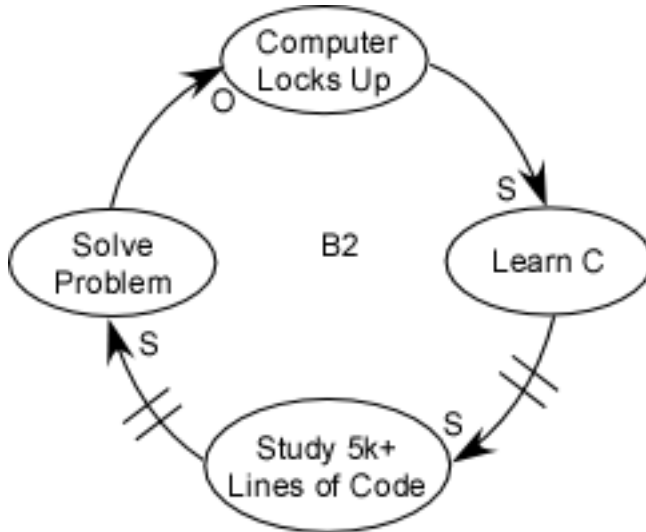


Figure 2 - The Systemic Fix

This is also a balancing loop bringing stability to the system. B2's advantage over B1 is B2 represents a systemic solution. The more the loop executes, the better the system gets at solving the problems. This reduces the system's dependence on outside interventions. The disadvantage of B2 is the time delays involved in learning "C" and understanding the application code. The time delays do get less the more the loop executes,

Solution #3 – Combining the Symptomatic and Systemic Solutions

Since B1 and B2 both contain "Computer Locks Up" we can combine the two causal loop diagrams into a more complete problem-solving picture.

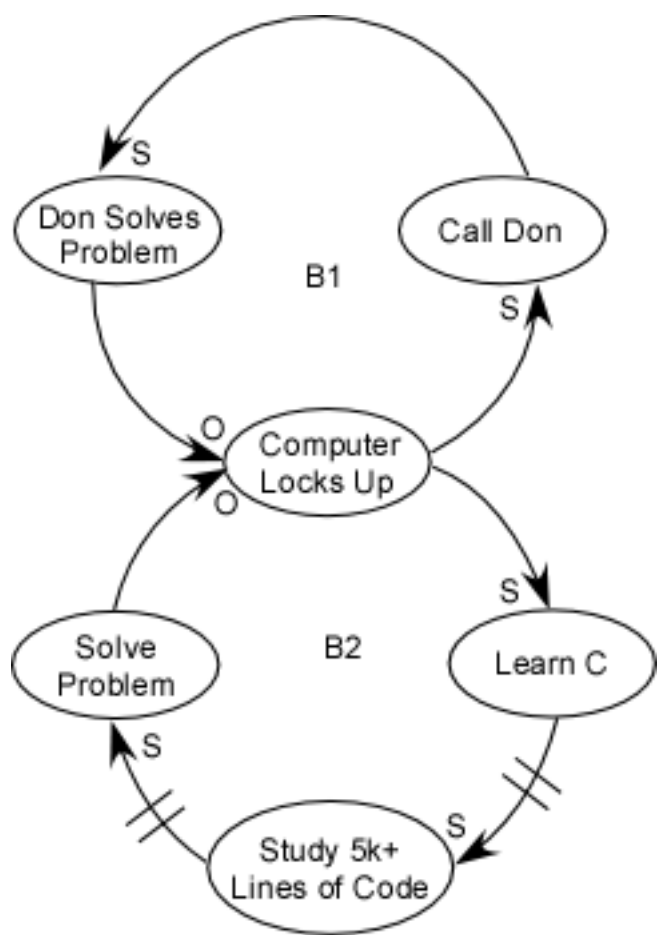


Figure 3 - Combining Actions

This shows the how the event “Computer Locks Up” can trigger one of two responses: a symptomatic response to make the problem go away quickly (B1), or a systemic response (B2) where the system becomes more capable of solving problems without external influence. This pattern occurs often enough that systems people have given it a name, “Shifting the Burden”. I’d been aware of this archetype, but preparing for my AYE Conference “It’s Déjà vu All

Over Again” session sharpened my awareness of what could go wrong.

An Unintended Consequence - Too Much Help

There happens to be a long term reinforcing loop that can show up with this archetype. This occurs when the symptomatic solution (B1) gets all the action. Don becomes better at both “C” and knowing the code base. This makes it easier and quicker for Don to solve the problems, and Don becomes indispensable.

This means the system’s ability to “cure itself” becomes atrophied. George loses his ability to persuade management to give him training. Production becomes accustomed to “great customer service” from Don. The ability to tolerate “pain” goes away, and the quick fix becomes the only fix.

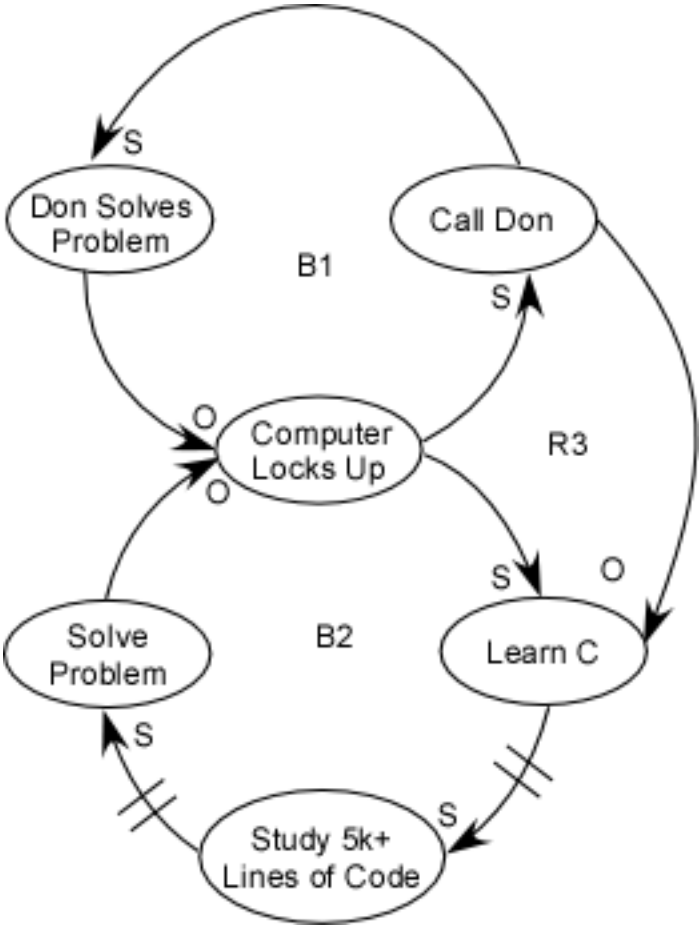


Figure 4 - The Addiction Loop

Invoking the symptomatic solution not only starts B1, it initiates R3. Calling Don reduces the need to learn “C”, thereby weakening the system’s problem solving ability. Carried to the extreme, this becomes an addictive response, and cripples the system.

And The Final Answer Is

Either balancing loop can be chosen to a given problem occurrence.

Handled properly B1 can be used to alleviate acute problems, while invoking B2 to solve chronic problems. This pattern of selectively, consciously choosing how to deal with each problem has resulted in George's ability to solve more code problems and start making enhancements to the code base.

I appreciate Stuart Scott and Jerry Weinberg for their suggestions about this article.

¹ <http://www.donaldegray.com/how-did-this-happen/> ² <http://www.donaldegray.com/just-do-something-stand-there/>

Solving Other People's Problems

© Don Gray 2000

A problem can be a lot of things; perhaps a struggle, a puzzle, or a task. As a consultant, I find it useful to define “problem” as *the difference between what is, and what is wanted*. From that point of view, learning to solve other people’s problems means learning to connect with your client, understand what they think they have, what they think they want, and what they would like to do about it. The experience of living teaches each of us a lot about this kind of problem solving, but the lessons are often murky. In recent years, I’ve been trying become more conscious of my own process so that I can control and accelerate it. Here are some basic principles that I’ve found helpful.

The Pause Principle

When you receive a problem, pause before trying to solve it.

The problem solving starting point should be “Don’t just do something, stand there!” This principle is especially important in the case where there are strong emotions involved. It’s downright critical when those strong emotions belong to you. Pausing creates space for a lot of things: to breathe, to center yourself, to let more information come to you, to notice that the problem is different than you first thought, or to notice a solution sitting in plain view. To pause is not to stop everything. You can be doing a lot of things while you’re pausing. One of the most important is to pay attention.

The Pay Attention Principle

Critical information about the problem will hide in plain view.

I often find that what I need to know to solve a problem is not

explicitly stated. But I also find that it's not always easy for me to hear information that I am told. Communication is a complicated process. One tool I use for checking the process is the Satir Interaction Model. This model decomposes communication into four parts: Intake, Meaning, Significance, and Response. Intake is what we physically see or hear. Meaning is the sum of ideas we think are conveyed by the message. Significance is our how the message impacts us; our emotional reaction to it. Response is what we do in reply or reaction to the message. What I like about this model is that it helps me be aware of a lot of ways that good communication can go bad. If any part of this process goes wrong, the whole communication will be distorted in some way. Certain personality types may be more prone to mistakes in certain steps than are others. I occasionally jump too quickly to into Response, before I've sorted out the other three parts of the process. Knowing that, I find the Pause Principle to help me give the process time and ultimately pay better attention.

Communication is also important because of the next principle: partnership.

The Partnership Principle

When I make the problem my problem alone, that makes more problems for all of us.

The problem I'm presented with isn't really my problem. I am not in the middle of that problem, the other person is. So there are risks whenever I try to help. By helping I may:

1. Deprive that person of a learning opportunity.
2. Take time from my work to do their work.
3. Encourage an ongoing co-dependency between my client and me.
4. Find a "solution" that my client feels no connection with.

The Partnership Principle reminds me to keep my client involved with the solving process. Ideally my client will solve the problem

and I will support them as they do so. The worst case is when I find myself solving a problem so much by myself that by the time I unveil my wonderful solution, the client has forgotten about the problem and moved on. This situation relates to the next principle: passion.

The Passion Principle

Don't care more about solving the problem than the other person does.

I once worked with a company that I believed had a problem. Their training material was sub-standard, their trainer had never used the software, and they were losing money because the sales channel wouldn't promote the class. We (the client and I) initially set out to solve this problem. But less than a week after we dove in I woke up and realized I was the only who cared about solving this problem. They would not commit the resources to deal with the problem. My visions of better training and a better training business notwithstanding, I had to scale down my own enthusiasm to match their level of caring. A client who has no passion about a problem doesn't really have a problem. And when something's not a problem, it's also not a problem not to solve it.

Passion does not belong to the intellect, so you can't solve important problems with intellect alone. You also have to connect emotionally, in some way. You have to see the people involved and find a way to enter their way of seeing the world.

The Person Principle

Every problem is a problem for some person.

Each of the principles above is a reflection of this one. To solve a problem well, you need to discover whose problem it is, and find a solution that works for them. This is not so difficult if you are both the owner and the solver of the problem, but it gets tricky when you're acting as a solver of someone else's problems. Often there are many more people involved in the problem than just the particular person who brought it to you to solve. Furthermore, you aren't only

dealing with the people themselves and the problem itself, but also how these people feel about the problem, about each other, about you, and about the attributes of whatever solution you come up with.

And *that* is *your* problem.

Bibliography: Problem Solving Leadership

Adams, James L. 1974 *Conceptual Blockbusting*. San Francisco: W.H. Freeman. A classic work on problem solving that identifies some of the major blocks—intellectual, emotional, social, and cultural—that interfere with ideation and design.

Connirae Andreas with Tamara Andreas 1995 *Core Transformation, Reaching The Wellspring Within* Moab, UT. Real People Press. ISBN 0-911226-32-X The authors offer a gentle process that allows one to transform personal limitations and reach the wellspring within. The process can be useful whenever it is important to find out what you really values and why. Examples of such times might be negotiating anything, exploring resistance, or discovering hidden motivations.

Block, Peter 1987 *The Empowered Manager: Positive Political Skills at Work*. San Francisco: Jossey-Bass. Clearly and incisively written, this book makes a strong statement in support of participative, as opposed to “bureaucratic”, management styles. Block outlines a vision of how organizations can be and explains ways for people in management positions to make them so.

Bolton, Robert 1979 *People Skills: How to Assert Yourself, Listen to Others, and Resolve Conflicts*. Englewood Cliffs, NJ: Prentice-Hall Other leadership qualities will be negated if you lack “people skills.” Every person, no matter how skilled a leader, could probably benefit from this review of the fundamentals.

Carnegie, Dale 1936 *How to Win Friends and Influence People*. New York: Simon and Schuster. The principles of leadership haven't changed in 70 years, or 7,000 (though there is a revised, modernized edition of this self-help classic). If you find you cannot tolerate Carnegie's rather ordinary rules, you're probably not ready to be a leader among rather ordinary people.

Thomas F. Crum 1987 *The Magic of Conflict, Turning a Life of Work into a Work of Art*. New York. Simon & Schuster. ISBN 0-671-63818-1 The author shares his discoveries in the art of conflict resolution, gained in part, through his many years teaching Aikido, a Japanese martial art. His teaching goes beyond the combative forms to show how the power of harmony and love can work in even the most difficult of situations.

Esther Derby & Diana Larsen 2006 *Agile Retrospectives: Making Good Teams Great*. Pragmatic Bookshelf. Teams don't improve by accident. They improve when they reflect together on what they are doing, what's working, and what to do differently. This book tells you how to help a team inspect and adapt their methods and teamwork for continuous improvement.

[Esther Derby Website](#)¹² And her [Blog](#)¹³ Esther's musings on the wonderful world of software and working with people in software.

Doyle, Michael, and David Straus 1976 *How to Make Meetings Work*. Chicago: Playboy Press Doyle and Straus have developed the "interaction method" for organizing and operating meetings of all types. Using this method, clearly described in this book, dozens of our clients have converted their meetings from the worst of times to the best of times. (For specialized technical meetings, see also Freedman and Weinberg.)

Edwards, Betty 1979 *Drawing on the Right Side of the Brain*. Los Angeles: J.P. Tarcher, Inc. Edwards explains how she teaches

¹²<http://www.estherderby.com>

¹³<http://www.estherderby.com/category/insights>

drawing by using the brain model of hemispheric specialization. An excellent book for developing the generally under-used “creative” right hemisphere—for anybody, aspiring artist or not.

Freedman, Daniel P., and Gerald M. Weinberg 1982 *Handbook of Walkthroughs, Inspections, and Technical Reviews*. New York: Dorset House. Meetings are ritual forms of group interaction which can be designed for different problem solving purposes. This is a handbook on how to design and lead meetings for one type of problem solving - obtaining critical reviews of work in progress. Such reviews can be a source of growth, or great anxiety and conflict, depending on how they are designed and led. (For some general principles of meeting design, see Doyle and Straus.)

Gause, Donald C. and Gerald M. Weinberg 1982 *Are Your Lights On?* New York: Dorset House. In this book, Don and Jerry have tried to teach people some things about the art of problem definition. It's light enough to be a catalyst between systems people and normal human beings. The process is continued in a more formal way in:

1989 *Exploring Requirements: Quality Before Design*. New York: Dorset House. A survey of human processes that can be used in gathering complete, correct, and communicable requirements for a software system, or any other kind of product.

Gordon, Thomas 1977 *Leader Effectiveness Training: The No-Lose Way to Release the Productive Potential of People*. New York: Wyden Books. Gordon, a management consultant, is also the author of the tremendously popular and helpful Parent Effectiveness Training. Many potential problem solving leaders will find useful material in Gordon's “no-lose” approach.

[Don Gray's site and blog](#)¹⁴ Integrating People, Projects, Processes

Gross, Ronald 1979 *The Lifelong Learner* New York: Simon & Schuster An essential handbook for those who take responsibility for their

¹⁴<http://www.donaldegray.com/>

own learning. Full of ideas, suggestions, and specific resources for self-renewal.

Hawken, Paul 1987 *Growing A Business* New York: Simon & Schuster. This delightful and very practical book is by the founder of Smith & Hawken, a mail-order garden tool company known for its integrity, product quality, employee satisfaction, and strong customer focus – all of which the author believes are essential for a successful business. Although the book is written for the small-business person, its lessons apply equally well to companies of any size.

Hollander, Edwin P. 1978 *Leadership Dynamics* New York: Free Press. An entry point into the theoretical and experimental results about leadership. The references are thorough, but inconspicuous, so that you may start with ideas and move to sources if you wish.

Keirsey, David and Marilyn Bates 1978 *Please Understand Me: An Essay on Temperament Styles*. Del Mar, California: Prometheus Nemesis Books. This book gives a detailed and illuminating explanation of the Jungian model of “temperaments.” It includes a self-administered test and scoring system—a simplified version of the well-known Myers-Briggs Type Indicator. It makes a strong argument for respecting human diversity in personality styles.

Kennedy, Eugene 1980 *On Becoming a Counselor*. New York: Continuum Publishing Co. Leaders often find themselves in the role of counselors to people asking for help. Kennedy’s book is directed to those who are not professional counselors, but who often perform this function and want to know at least how to avoid doing harm.

Kroeger, Otto, and Janet M. Thuesen 1992 *Type Talk at Work*. New York: Delacorte Press. This very readable book reviews the Myers-Briggs model and applies it to the workplace, giving many useful and humorous examples.

Lynch, James J. 1985 *The Language of the Heart: The Human Body in Dialogue*. New York: Basic Books. An important and very readable

book about the interrelatedness of language, emotion, and health. Lynch, a psychologist, makes a convincing case for paying attention to the “social membrane” (human interaction) in understanding and treating hypertension and migraine headaches.

McKim, Robert H. 1980 *Experiences in Visual Thinking*. 2nd ed. Monterey, California: Brooks/Cole. This book focuses on visual and other non-cognitive approaches to creative thinking and problem solving. It includes an excellent series of exercises for the reader to practice.

Myers, Isabel Briggs 1980 *Gifts Differing*. Consulting Psychologists Press, Inc., 577 College Ave., Palo Alto, CA 94306 Myers, one of the co-designers of the famous Myers-Briggs Type Indicator, explains the Jungian theory behind it. She discusses how to interpret and make practical use of the revealed types, in This book could be read profitably in conjunction with Keirsey and Bates, Please Understand Me.

NTL Institute *Reading Book for Human Relations Training*, edited by Larry Porter P.O. Box 9155, Rosslyn Station Arlington, VA 22209 Many of our workshop graduates want to learn more about how they interact with other people. We often recommend that they take NTL’s “Human Interaction Laboratory”—and this is the book of readings that they take home from that lab.

Oshry, Barry 1995 *Seeing Systems, Unlocking the Mysteries of Organizational Life*. San Francisco, CA. Berrett-Koehler Publishers. ISBN 1-881052-73-7 Oshry suggests that we need to understand the source of a problem, the kind of organization involved, and the implications of our structural position in that organization. The book is clearly written, thought provoking, and designed for immediate application.

Progoff, Ira 1975 *At a Journal Workshop*. New York: Dialogue House Library If you want to learn more about keeping a journal, here’s a whole book on the subject.

Quenk, Naomi 1993 *Beside Ourselves: Our Hidden Personality in Everyday Life*. Palo Alto, CA. CPP Books (A Division of Consulting Psychologists Press, Inc.) An excellent study of the MBTI “Fourth Function”—the state in which we are not ourselves but operating under stress.

Rogers, Carl 1977 *On Personal Power*. New York: Dell. If you are interested in power and leadership, read this book before you do anything else. Other books by Carl Rogers which will help you on your path to effective leadership include:

1961 *On Becoming a Person*. Boston: Houghton Mifflin

1980 *A Way of Being*. Boston: Houghton Mifflin

Rothman, Johanna. 2012 *Hiring Geeks That Fit*. Rothman Consulting/Leanpub. Your leadership skills are most evident to candidates during the hiring process. Johanna explains how to make the hiring process one that invites people in, not scares them off.

Rothman, Johanna and Esther Derby. 2005 *Behind Closed Doors: Secrets of Great Management*. Pragmatic Bookshelf, Raleigh NC and Dallas TX, 2005. You don’t have to choose between management and leadership, but you do need to master the tactical parts of management. Johanna and Esther show you how.

Rothman, Johanna. 2007 *Manage It! Your Guide to Modern, Pragmatic Project Management*. Pragmatic Bookshelf, Raleigh NC and Dallas TX, 2007. Part of leadership is recognizing the context in which you work, and then managing to make that context as good as you can make it. Johanna helps you recognize your context, and helps you choose what to do next, so your projects can be successful.

Rothman, Johanna. 2009 *Manage Your Project Portfolio: Increase Your Capacity and Finish More Projects*. Pragmatic Bookshelf, Raleigh NC and Dallas TX, 2009. Too many projects? Want to organize them and evaluate them without getting buried under a mountain of statistics? This book will help you collect all your work, decide which projects you should do first, second—and never.

Rothman, Johanna 2013 *Manage Your Job Search*. Rothman Consulting/Leanpub. Treat your job search like a project, dividing it into small steps and obtaining feedback along the way. Discover your traps. Learn from the tips. Best of all, find fulfilling work.

[Johanna Rothman site and multiple blogs](http://www.jrothman.com)¹⁵ Start here for Johanna's articles and blogs.

Russell, Bertrand 1951 *The Conquest of Happiness*. New York: Signet Books Unhappy people are not leaders. Nobel Prize winning philosopher Bertrand Russell tackles the ancient question of how to be happy—and succeeds. This classic is available free on line.

Satir, Virginia, John Banmen, Jane Gerber, Maria Gomori 1991 *The Satir Model, Family Therapy and Beyond*. Science and Behavior Books, INC., Palo Alto, CA 1991 ISBN 8314-0078-1

Satir, Virginia 1983 *Conjoint Family Therapy*. 3rd ed., Palo Alto, Ca.: Science and Behavior Books.

1972 *Peoplemaking*., Palo Alto, Ca: Science and Behavior Books.

1985 *Meditations and Inspirations*., Millbrae, Ca.: Celestial Arts.

1978 *Your Many Faces*., Millbrae, Ca.: Celestial Arts.

1976 *Making Contact*., Millbrae, Ca.: Celestial Arts.

1976 *Self-Esteem*., Millbrae, Ca.: Celestial Arts.

We have been greatly influenced by the work of Virginia Satir. We first became aware of her radical approach to life through Peoplemaking, which is a good survey of her ideas about how we learn to interact with others. Conjoint Family Therapy is more of a comprehensive textbook for therapists, but like all her books, it is written without academic pretension. For a lighter introduction to specific topics of importance to leaders, try one or all of her little books from Celestial Arts.

¹⁵<http://www.jrothman.com>

Waddington, C.H. 1977 *Tools for Thought*. New York: Basic Books. This posthumous work by the brilliant geneticist and teacher introduces ordinary people to sophisticated analytic methods that may be used in problem solving - for example, general systems theory and game theory.

Weinberg, Gerald M. 1986 *Becoming a Technical Leader*. New York: Dorset House. This book is a personalized guide to developing the qualities that make a successful leader. It identifies which leadership skills are most effective in a technical environment and explains why technical people have characteristic trouble in making the transition to a leadership role.

1985 *The Secrets of Consulting*. New York, Dorset House Publishing, Inc.

2001 *More Secrets of Consulting: The Consultant's Self-Esteem Tool Kit*. New York, Dorset House Publishing, Inc. Consulting may be defined as the art of influencing people at their request. *The Secrets of Consulting* takes you behind the scenes of that art, explaining in detail why the world of consulting seems so irrational, and some very practical steps you can take to make it more rational. *More Secrets of Consulting* offers a set of tools to help the consultant operate from a congruent position in all circumstances.

1991 *Quality Software Management, Volume 1: Systems Thinking*. New York: Dorset House.

1992 *Quality Software Management, Volume 2: First-Order Measurement*. New York: Dorset House.

1994 *Quality Software Management, Volume 3: Congruent Action*. New York: Dorset House.

1996 *Quality Software Management, Volume 4: Anticipating Change*. New York: Dorset House. These books apply our teaching about system thinking, observation, and congruent action to the context of software engineering. Volume 1 is based on the type of thinking that Satir applied to families. Volume 2 is structured around Satir's

Interaction Model. Volume 3 is based on Satir's ideas on congruence. Volume 4 is based on her Change Model.

1975 *An Introduction to General Systems Thinking*. New York: John Wiley & Sons.

Weinberg, Gerald M. and Daniela Weinberg 1979 *On the Design of Stable Systems*. [Retitled: *General Principles of System Design*.] New York: Dorset House. These two volumes form a good starting place for those whose interest has been aroused about the subject of general systems thinking. The first volume emphasizes the role of perception in thinking—"it's not the event, but the reaction to the event." The second volume emphasizes the systems principles that can be derived from the elementary need of any system to survive—illuminating the importance of Satir's "survival rules" in the most general possible context.

Weinberg, Gerald M. 2006 *Weinberg on Writing: The Fieldstone Method*. New York: Dorset House Publishing. Writing is an essential leadership skill. This book can help: A reviewer said: "A few years ago I went to the writing section of my local library and checked out every book I could find. I found better books about the business of writing, pitching stories, and fundamental English rules. As for the actual process of writing, this book is far and away better than anything else I have ever read."

Weinberg, Gerald M. 2007 *The Aremac Project*. New York: Little West Press, 2007. This is a story of several young problem-solving leaders and how they develop while coping with some thrilling adventures. A review saidL "A gripping story ripped from tomorrow's headlines. A great cast of characters, not one of them afraid to be smart ... and with 'tude to spare. A fun read as a familiar name in books takes his first swing at the novel ... and scores a home run!"—Mike Shepherd, national best selling author of *Kris Longknife*—Defiant. [Note: Each of Jerry's subsequent novels also concerns the adventures of problem solving leaders. They are all available as ebooks, as are all his non-fiction books.]

Weinberg, Gerald M. Website¹⁶ *The Secrets of Writing and Consulting Blog*. <http://secretsofconsulting.blogspot.com/> For consultants who want to share their secrets, plus occasional thoughts to help both the developing and experienced writer.

¹⁶<http://www.geraldmweinberg.com>

More From Esther Derby

I started my career as a programmer, and over the years I've worn many hats, manager, internal consultant and business owner. From all these perspectives, one thing became clear: our level of individual, team and company success was deeply impacted by our work environment and organizational dynamics. As a result, I have spent the last twenty-five years helping companies design their environment, culture, and human dynamics for optimum success.

How your company's values are reflected in your environment, your culture and organizational dynamics, directly determines the quality and level of your success. When your company environment, culture and organizational dynamics are positive, mutually receptive and reinforcing, your people and teams have the capacity to achieve great things.

I help teams and managers understand what's working and where there are contradictions that sap productivity and stifle innovation. We explore how best to maximize your capacity for achievement by eliminating wasted effort, politics, cynicism, and fear. Together, we achieve a holistic view of your organization, and design your environment to directly enable, support and sustain your agile success, now and into the future. *** I've written over 100 articles, and co-authored two books—Agile Retrospectives: Making Good Teams Great and Behind Closed Doors: Secrets of Great Management. I write about management, leadership, collaboration, organizations and change (or another topic I'm currently exploring). If you'd like to get a taste of how I approach things, many of my articles are posted on this site. If you have particular topic or issue you'd like to explore, email me, and I'll put together a collection of my articles that may be helpful to you.

I also teach workshops and talk to groups all over the world.

I've learned a lot about how organizations work through observation and action research. I also hold an MA in Organizational Leadership and a certificate in Human System Dynamics.

See more at: <http://www.estherderby.com>

More From Don Gray

I started working with clients in 1984 focused on delivering value. My experience crosses a variety of industries, from finance to manufacturing. My clients' sizes range from small startups to Fortune 50 organizations. This background allows me to assist clients as they change their software development practices.

Change contains many dimensions. Often my clients initially focus on a single aspect of their change, for example moving to the Scrum development framework. I help them succeed by ensuring individuals, teams and the company have the necessary ability and motivation to succeed.

Building on my control systems background, I've studied complexity theory, modeling, and network analysis. I balance this with understanding communication, different personality models, and human systems dynamics. I incorporate these diverse subjects into my work and writing. You can find a number of my articles published at Better Software magazine and StickyMinds.com.

Along with Esther, Jerry and Johanna, I helped create and host the AYE Conference and the Change Artistry workshop. I employ the same experiential construction for the public and custom workshops I facilitate.

There are a number [articles](#)¹⁷ at my website and on my [blog](#)¹⁸ about [change](#)¹⁹, [problem solving](#)²⁰, [systems thinking](#)²¹, and [teams](#)²².

¹⁷<http://www.donaldegray.com/article/article/>

¹⁸<http://www.donaldegray.com/category/blog/>

¹⁹<http://www.donaldegray.com/tag/change/>

²⁰<http://www.donaldegray.com/tag/problem-solving/>

²¹<http://www.donaldegray.com/tag/systems-thinking/>

²²<http://www.donaldegray.com/tag/team/>

If you have any questions at all about change, teams, leadership, or just want some feedback or a second opinion, feel free to email me. don@donaldegray.com

More From Johanna Rothman

I consult, speak, and train about all aspects of managing product development. I have a distinctly agile bent. I'm more interested in helping you become more effective than I am in sticking with some specific approach. There's a reason my newsletter is called the "Pragmatic Manager"—that's because I am!

Want to read the other things I've written? Take a look at my [articles](#)²³ and [blogs](#):

[Johanna Rothman's Website](#)²⁴

[Managing Product Development Blog](#)²⁵: Management, especially good management, is hard to do. This blog is for people who want to think about how they manage people, projects, and risk.

[Hiring Technical People Blog](#)²⁶: Hiring technical people and being hired isn't necessarily easy, no matter what the economy is doing. Use the tips here to hire better, or find a new job.

[Create an Adaptable Life Blog](#)²⁷: The only people who don't have to change are the ones who are already buried. Since you're not one of those, see what I've learned in living with and adapting to change.

If you liked this book, you might like the other books I've written:

- [Agile and Lean Program Management: Collaborating Across the Organization](#)²⁸

²³<http://www.jrothman.com/articles/>

²⁴<http://www.jrothman.com>

²⁵<http://www.jrothman.com/blog/mpd>

²⁶<http://www.jrothman.com/blog/http>

²⁷<http://www.createadaptablelife.com/>

²⁸<https://leanpub.com/agileprogrammanagement>

- [Manage Your Job Search](#)²⁹
- [Hiring Geeks That Fit](#)³⁰
- [Manage Your Project Portfolio: Increase Your Capacity and Finish More Projects](#)³¹
- [Manage It! Your Guide to Modern, Pragmatic Project Management](#)³²
- [Behind Closed Doors: Secrets of Great Management](#)³³

I'd like to stay in touch with you. If you don't already subscribe, please sign up for my email newsletter, the [Pragmatic Manager](#)³⁴, on my web site. Please do invite me to connect with you on [LinkedIn](#)³⁵, and follow me on Twitter, [@johannarothman](#).

²⁹<https://leanpub.com/manageyourjobsearch>

³⁰<https://leanpub.com/hiringgeeks>

³¹<http://pragprog.com/book/jrport/manage-your-project-portfolio>

³²<http://pragprog.com/book/jrpm/manage-it>

³³<http://pragprog.com/book/rdbcd/behind-closed-doors>

³⁴<http://www.jrothman.com/pragmaticmanager/>

³⁵<http://www.linkedin.com/in/johannarothman>

More from Gerald M. Weinberg

I was very small when I was born—only 9 pounds or so—and I’ve never gotten over it. Inside me, there’s a little boy who can’t make sense out of the world, but keeps trying. I read about computers when I was about 11, and thought that these “giant brains” might help, so I determined to work with computers when I grew up. This led me to study math and physics, because my guidance counselor told me that computers “had something to do with electronics.” There were no computer classes, or even computers, any place where I found myself. Indeed, I never took a computer course in my life. Yet.

After more than 50 years working with computers, I’ve learned a couple of things, but I still can’t make sense out of most of it. Most of all, I’ve discovered that people are at the bottom of just about every problem—but I think I knew that when I was little, then got talked out of it somewhere along the way. I’ve worked hard at relearning this lesson, and learning how to do something about it. While educating myself, I learned a second principle: I’m the “people” at the bottom of most of my problems.

Anyway, using these insights, I’ve been able to help a lot of people solve problems. I’ve written a lot of books, too, which many people have told me are helpful; but I get much more satisfaction from helping people directly, so I get to know them. For instance, I’ve helped several hundred people write their books, and several thousand find better ways to do their job. I feel very good when I find out that many of these people have learned things from me that they take back with benefit to their family life—possibly because my early family life wasn’t very wholesome.

From my first attempt to create a better family life for myself, I have four children (one, now deceased) and four grandchildren. Although I thought I had screwed up that family life, too, now that the kids have all reached the half-century, things seem to have worked out okay. I did learn a lot from that first attempt, which enabled me to choose Dani for my second—by far the best decision I ever made. We’ve now passed almost five decades of living together, which involved about 7 serious renegotiations of our original, informal marriage contract. We’ve also had 16 children so far, but they’ve all been German Shepherd Dogs.

I like to start new things that help people. Although I’ve written several hundred articles and more than 100 books, my greatest satisfaction is creating real-life learning experiences—schools, camps, institutes, seminars, or development groups. I guess one of the things I am is a social architect. When I’m not starting a new social system, I’m often repairing an old one; in this role, I’m an organizational therapist. And, I also like to work with one-person organizations; perhaps I am at my best as a restorer of works of art.

Anyway, if you’re a work of art that needs a little restoration, I invite you to explore these readings and see if there’s anything we can do for each other. You might also discover more about my work by exploring my website, <http://www.geraldweinberg.com>. Or, you can examine the essays in *The Gift of Time*, edited by Fiona Charles, celebrating my 75th birthday with some truly fine essays from many of my students and colleagues.