

Test Drive Your Knowledge

Answer the following questions with the knowledge you have now. Then answer them again at the end of this chapter.

1. What current trends in software development are good examples of systems thinking?
2. When there is a problem, how do you decide who to blame?
3. When are analogies useful? When are they harmful?

4.1 So Many Principles!

Because this book is focused on changing your intentions and worldview, we rely on principles as much as practices. For this reason, we've ended up with three sets of principles:

- **Systems thinking principles**—helping to guide all your thoughts, ideas, speech, intentions and worldview
- **Professionalism principles**—defining what it means to be a professional in the new world
- **Servant leadership principles**—explaining how it looks and feels to be a true servant leader

In this chapter, we'll examine the systems thinking principles. In many ways, these principles are meant to bring us back to the foundations of our worldview and intentions, rather than focusing too much on practices. All four parts of the path are important, but world view and intention are the most important.

These principles are not codified by academic systems thinking scholars. While we (the authors) are certainly guided and inspired by the scholars we referenced in Chap. 2—The Scholars of Systems Thinking, we do not attribute these principles back to them as a whole. Instead, we’ve created a new set of principles to guide the application of systems thinking to the field of software development. Principles to help correct the ills (and illth) of today’s problems in our beloved field.

The use of principles can help to elevate your thinking away from “What should I do?” practices towards “How can I see differently?” views and intentions.

1. Trust = Speed
2. Avoid best practices
3. Beware of immense power of analogies
4. Blame the system, not the person
5. Treat people like people, not machines
6. Acknowledge your boundaries
7. Relation-ness matters more than thing-ness

4.1.1 Systems Thinking Principle 1: Trust = Speed

Trust and control are incompatible because the core of trust involves freedom. To trust people is to count on their sense of integrity, believing that they will choose to act in a trustworthy manner, while recognizing the possibility that they may choose to betray the trust.

Robert C. Solomon and Fernando Flores (Solomon 2001)

In a flurry of methodologies, practices, team structures and manifestos, we often forget one factor—**trust**. If we don’t trust the members of our team, it doesn’t really matter what processes we use, we’re never going to be very good. And we’re never going to be fast. It’s amazing how much you can strip away in terms of documents and processes when you trust each other.

Too often, we have a kind of persistent “hurt feelings” in the workplace. “Well, I did trust someone once, but they let me down, so now I need everyone to fill out this twenty page form before I can do anything.” It is true that when we have an environment of trust, certain people will take advantage of that. But piling processes and documents on top of a functioning team to ensure that no one can ever “get away” with things is not the answer.

Having a team that trusts one another improves everything—quality, speed, happiness at work. We’ve worked in situations where the product owner truly trusts the team to execute on his wishes. The results are nothing short of magical. Things get done quickly. Quality is through the roof. In one case, a product owner said “It’s like the team is inside my head. They know exactly what I want even before I can communicate it completely.”

Trusting someone is a risk. Anyone who has worked for more than a couple of months out of school has had their trust broken by a coworker, boss, client or

vendor. There is no magic to shielding yourself from broken trust. But the benefits that you gain from trusting people far outweigh the problems that happen when your trust is broken. Your goal should always be to find ways to trust people more and sooner, in whatever way possible.

4.1.2 Systems Thinking Principle 2: Avoid Best Practices

If anyone were to study without theory such a company, i.e. without knowing what questions to ask, he would be tempted to copy the company, on the pretext that “they must be doing some things right.” To copy is to invite disaster.

W. Edwards Deming (Deming 1994)

Much of what currently passes for IS theory is nothing more than “what works in practice.

Peter Checkland, Sue Holwell (Checkland 1997)

Adapt. Don’t adopt.

Timothy Lister

“Best practices” is a term we use persistently in software development, and especially in consulting. It implies stuff that’s been tried before and proven successful. It worked for those guys, and it’ll work for you.

But it’s a dirty lie. You try the best practice and it doesn’t live up to the advertising. So, you ask for your money back. The expert’s response? “You did it wrong.” You made some small mistake in the application of that best practice that prevented you from getting value from it.

The truth is that best practices should not, should never, be followed blindly. They are, at best, interesting stories from someone who experienced success. They are, at worst, a crutch or a set of handcuffs that cause teams to do stuff that they know is wrong, but “the consultant told us to do it.” There are no “best practices.”

There are, as Tom and Mary Poppendieck (Poppendieck 2003) say, only “good practices in context.” We would add “. . . in context of the right worldview and intentions.” Don’t get caught up with what the industry best practices are. Treat them with curiosity and good humor.

Mike Myatt, executive advisor and *Forbes* columnist, has a twist on best practices that might be useful. He says that best practices are never “best,” but in fact we should be looking for “next practices,” ways to take what we learn from elsewhere and apply it to our own situations, projects, teams, companies.¹

Most purveyors of best practices talk about the need to “adopt” their particular method. They want you to buy into it, to take on that mantle and become an evangelist of it yourself. There is a certain extent to which you should use the set

¹bit.ly/bestpracticesaint.

of practices without questioning it. “It works for everyone else, why should you be different?” is the expert’s question. But we’d like you to question the experts, to try to understand whether these “best practices” are really valid in your circumstance or not. Do you need to change them? Do you need to “adapt” them to what is unique about your world?

Probably the reason most experts are fearful of people adapting (rather than adopting) is that there might be a tendency to change the new practices to be as similar to the old practices (that definitely weren’t working) as possible. This is not a good goal, of course. But it does happen. But rather than saying “Let’s adopt this new process without question,” we can distinguish between good ways to adapt and bad ways to adapt. It comes down to intent. If you are adapting the new process to make it like the old one, question yourself. What is your intention? Are you just afraid to change? Are you being lazy in your thinking? Are you trying to create a “hybrid” of your old way and the new way just because you’re afraid to make the leap? Instead, find ways to adapt that help the new process work better in your environment. Are you making the adaptation because of some special element in your environment? Is there a business reason you are adapting it? If so, great. That’s a good reason to adapt.

One of our systems thinking scholars, Peter Checkland, showed how our thinking can be flawed when we “adopt” new processes. One of Checkland’s students came up to him after a lecture one day and said that he had used Checkland’s soft systems methodology and it worked. Checkland immediately scolded the student. “How do you know it was the process that worked? How do you know it wasn’t your team that worked?” The student was taken aback, of course. (He was trying to compliment the guy, after all.) But Checkland was trying to tell him that a process cannot “work” on its own, it depends on the context of who is using it and the particular situation. It is impossible to say that, once you have a success, you “have evidence” that the process worked. You have no evidence. All you know is that with that team, in that circumstance to solve that particular problem, using that process, it worked. That’s all you can say.

It comes down to one fact: *success never comes from within the practices themselves. The success is in the relation between the people and the practices*, as we show in Fig. 4.1.

Taking this point of view is problematic to consultants. What do consultants do if not box up best practices and sell them to clients?

We have a lot of work to do to explain what a “post-best practices world” would look like. We promise to talk through many of these issues in this book. For now, let’s just leave our advice as “avoid best practices.”²

²When we talk about avoiding blind adherence to best practices, you might say “Well, sure, but no one really thinks like that, do they?” Yes. Yes, they do. We’ve seen many speeches, articles and books that make this very mistake. Here is a recent example of an Agile blogger and coach offering

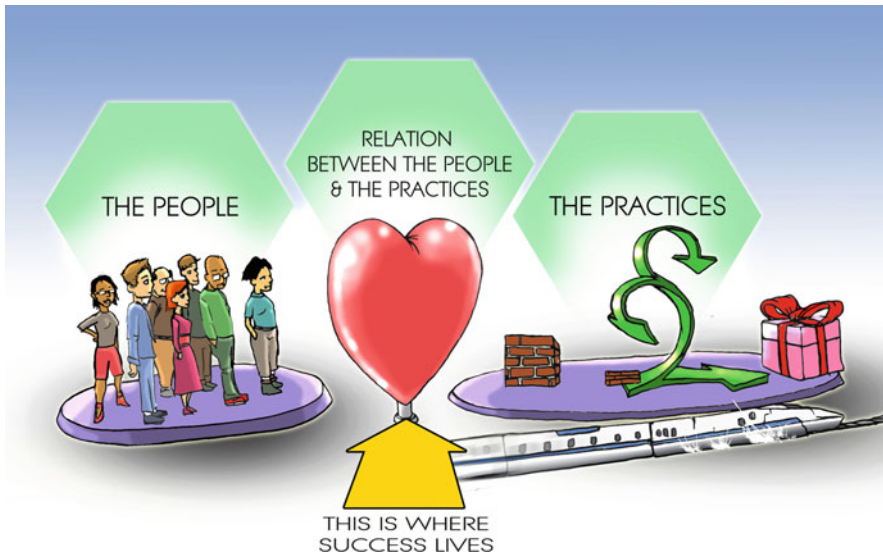


Fig. 4.1 The relation between people and the practices is where success truly lives. © 2017 by Daryl Kulak and Hong Li—reprinted with permission

4.1.3 Systems Thinking Principle 3: Beware the Immense Power of Analogies

All metaphors...offer ways of not seeing as well as ways of seeing.
Michael C. Jackson, systems science pioneer

First of all, it is certainly true that analogies (and metaphors) can be very useful in explaining concepts to people. But we want to point out that they can also be dangerous. Analogies run rampant in our software world. Maybe because software is so ethereal, but we love to construct analogies to explain why we're doing what we're doing. Most prominent is the "house building analogy." "You see, if you're building a house, you need to start with the foundation, and only then can you start on the framing and the electrical circuits."

Other people tend to use more organic analogies, like a gardener and her garden. Plant the seeds, watch them grow, don't yell at the seeds to grow faster, etc.

exactly the rigid "best practices" advice we are talking about here. His post is called "**Modifying Scrum—You THINK You Know Better...**" Now, it's bad enough that one so-called expert thinks this way. But his blog post was uncritically linked to by a number of other Agile coaches. Mechanical thinking is prevalent among Agile "experts." And why not? Being mechanical about Agile "best practices" makes life a whole lot easier...for the coach. (This is a real blog post...we did not make it up...swear to God.)

bit.ly/scrumknowbetter

At one level, these analogies seem harmless. Analogies can make whatever you're saying sound right. If you like a waterfall software model, you can use lots of physical building metaphors: house-building, bridge-building, road-construction. They work great in getting a point across.

But, we tend to take analogies too far. Analogies break down easily. If software is like building a house, and the architectural framework is the house's concrete foundation, then what is the equivalent to the sidewalk in front of the house? What is the connection of the analogy to the cedar siding? They don't relate, of course. Analogies can be very brittle and you never know when they will cause you problems.

Think of the house-building analogy again. In a waterfall lifecycle, it makes sense to complete the foundation, and only then move on to the frame. That is the only way to build a house. But is it the only way to build software? No. We can create "just enough" foundation and build a bit of value on top of it. You probably wouldn't build a house that way. So our house-building analogy fails us.

The biggest example we have of an analogy taken too far in our software world is Lean³ software development. We try to make a comparison of how software is "similar to" producing automobiles at the rate of demand. The analogy breaks down almost immediately (is one feature equal to one car or one part?), but surprisingly, it has made great inroads and has many prophets.

In systems thinking, there is something stronger than an analogy. Ludwig von Bertalanffy (Bertalanffy 1968) provides us with examples of analogies and their more powerful cousin, the **homology**. He talks about how the flow of electricity and the flow of fluid can be studied using the *same set of laws*. All the physical equations that govern the flow of water through a tube work the same way for the flow of electricity through a cable. This is a super-powerful alternative to the analogy. It is an analogy that doesn't break down, it doesn't have exceptions, it's just *the same*.

The homology is very refreshing, compared to what we try to do in the software world "It's sorta like steering a cruise ship but it's also sorta like renting a backhoe." Whaaaa?? Instead, software development needs a good **homology**. We intend to supply one in this book with Robert Rosen's M-R model (Rosen 1991).

4.1.4 Systems Thinking Principle 4: Blame the System, Not the Person

In systems thinking we are always, naturally, trying to "think about the system." If we are developing software, we're trying to think about how a sprint proceeds, who does what, how we make progress, how we recover from missteps, how to move

³More about Lean at EnterpriseAgile.biz.

faster and how we figure out what it all means. When we're hiring people, there is a hiring system. You might also call it a "hiring process." But the process is just part of the system. A process diagram, showing the tasks, handoffs and responsibilities is a small part of the overall system. A system involves the people. Not people as interchangeable automatons, but real live people—with attitudes, emotions, personal goals, hidden agendas, relationships, worldviews and intentions. The system has to take all this into account.

Within a system, there are lots of interconnections. That means that you can fix what you think is the problem, and it can cause additional problems elsewhere. That's why we use causal loop diagrams. But it also means that the person who is sitting right directly where the problem occurred may look like the culprit, but they're really not. The problem likely originated elsewhere. In fact, W. Edwards Deming (Deming 1992) said throughout his life that almost all the problems that corporations encounter are a problem with the system, not a problem with a particular person. This seems so counter-intuitive at first. We all want to know "Who's to blame?" But it doesn't help to figure that out. It helps more to "blame the system."

Blaming the person comes directly from our best practice mentality. "I told you to use this Scrum best practice, which is guaranteed to work! You screwed it up, so you must have done your job poorly. Get out!"

4.1.5 Systems Thinking Principle 5: Treat People Like People, Not Like Machines

How is it, then that our organizations are less adaptable, less creative and less inspiring than we are? Because they are hostages to an ideology that is, in a real sense, inhuman.

—Gary Hamel (Hamel 2012)

In systems thinking, we try to find ways to affirm that people are more than just machines set up to do a job. They are people—complex, thinking, feeling, relationship-building, unpredictable, ambitious people.

Are you, as a manager, treating your people like people? If you are managing your team "by the metrics" or managing by results, you are probably treating them like machines. If you have tightly defined roles and responsibilities, you are treating them like machines. If you expect the team to follow a set of best practices (Do Scrum or else!), you are treating them like machines.

Machine-oriented teams are obvious by the number of "yes people" you can observe. "Yes sir, you are right. In fact, you are always right!"

Management problem solving in a machine-oriented team looks like replacing defective parts (people).

Most American corporations and teams are mechanical today. This is not necessarily a problem, as long as everyone knows that the organization is a machine. The executives must admit “we’ve created a machine.” In this environment, any professions of “We deeply care about our people” or “Our people are our greatest asset” should be avoided. You’ve created a machine organization, you do not care about your people. The other thing that should be avoided is the expectation that people will think for themselves. It is very difficult to think for one’s self in a machine organization. This type of organization can work in two circumstances. The first is that the work must be simple. The second is acknowledging that the machine will wear down over time. It will achieve some level of efficiency, and perhaps even success. Then the “money machine” will wear down and will stop working. Trying to get it started up again will be hard or impossible.

Look at the music industry, for example. Music executives have, for decades, seen new artists as “money machines.” They look to squeeze profits out of the artists and then throw them away. That’s a perfectly fine business model, but it will not last. The music labels are seeing the destruction of their businesses today, and are seemingly helpless to stop it. They cannot. The machine has worn down. It will never be back the way it was. Those executives must move on to the next opportunity.

The machine organization must obey Newton’s laws of thermodynamics. Everything must eventually yield to entropy and disintegrate from year to year. It will wear out and run down.

In the software industry, we often hear terms like the “software factory.” A name like this may indicate that people see the team mechanically, like a money machine. Running a machine-model organization can be a successful and dignified endeavor. But everyone involved needs to acknowledge that it is a machine, and any part of it is dispensable, and sooner or later will need to be dismantled and replaced.

The opposite of machine-model management is treating people like people, also known as *servant leadership*. This means acknowledging the fuzzy side of people. It does NOT mean having cake parties, throwing confetti, giving awards or other “fun committee” ideas. It means providing people with control over their own work. It means connecting people with a shared vision and common purpose. And giving them a chance to get really, really good at what they do. Autonomy, purpose, mastery (Pink 2011). Now that’s a party!

4.1.6 Systems Thinking Principle 6: Acknowledge Your Boundaries

Boundaries are an important part of systems thinking. Scholars like C. West Churchman (Churchman 1979), Werner Ulrich (Ulrich 1996) and Gerald Midgley (Midgley 2000) have dedicated their lives to this concept.

Boundaries are part of our mental models. They are certain limits that must not be crossed. And they are often problematic. They keep us from seeing the whole picture. They cripple us from acting on information in a sensible way.

Let's look at an Agile example.

A team is producing software quickly and it is exactly what the product owner (PO) wants. The communication between product owner and other team members is clear, instantaneous and precise. At a demo as the software nears a production release, some other executives from the company attend. The president of the company looks shocked. "This software isn't right!" he exclaims to the team. The product owner is as surprised as anyone.

The team assumed that they should communicate with their product owner AND ONLY their product owner. Sure, the PO made a mistake by not asking the right questions of the company president. But the team also erred by assuming that they should not need to look beyond their product owner. This is a boundary built right into the Scrum framework.

You can see this situation as a problem with seeing a boundary in the wrong place. The team saw the boundary of their user community as one person—the PO. They did not think they needed to see past the PO into the broader user community, other stakeholders, executives, and potential end users who may not have had perfect communication with the PO. The team's boundary caused them to miss their goal completely, even though succeeding beautifully with their Scrum compliance.

Each of us needs to constantly look at the boundaries we've placed around us and listen hard when someone tells us those boundaries are wrong. The impossible happens to some industry, project and person every day. The worst reaction is to think it's still impossible when it's already happening, as we've seen with so many companies who have been blown out of their leadership position by one of their competitors actively and currently doing "the obviously impossible."

4.1.7 Systems Thinking Principle 7: Relation-ness Matters More Than Thing-ness

The aim of science is not things in themselves but the relations between things; outside these relations there is no reality knowable.

Henri Poincare

What do we mean by "silos" in an organization? Silos are when people and groups have tightly defined roles and their communication with other people/groups is formal and regulated.

A deeper meaning of silos is that it is a **lack of relations between people and groups**. The highest functioning organization will have the richest relations. If people inside a company know and trust each other, that company will be able to operate efficiently. And if the people inside the company have rich relationships with the customers, they will also be effective in the long term serving those customers. As they say, it is all about who you know, and also how rich your relationship is with them.

The richness of relationships on a team is also important. But something we get from Robert Rosen's M-R model (Rosen 1991) in systems thinking is this: the relations in an organism (fields, connections, etc.) are actually more important than the things (cells, organs, etc.). Rosen shows how the "life" is in the relations, not in the physics or chemistry of an organism (or organization).

This goes against all of science thus far, and it is also a slap in the face for our usual methods of leading people. What do we do when we have a problem? We try to figure out "who is the problem?" Once we know that, we can get rid of "the problem" and everything will be right again. But what if the problem is in the relations between the team members? Something in the embedded culture of the team? Then, getting rid of a "problem person" won't be the solution, it will have to be a deeper change.

Many soft skills consultants talk about changing the culture of a team. But it isn't easy. It definitely won't result from some classroom exercise, or knowledge of additional concepts, or, god forbid, a ropes and ladders adventure in the wilderness together. Soft skills consultants have tried many things, except to actually work with teams in real life to fix the problems of a bad culture. As far as we know, this is the only way to fix a culture.

4.2 Principles for Your Worldview

Use these principles to guide your own worldview. The systems thinking principles can be effective reminders in conversations as your teams are moving to more productive processes. "Are we acknowledging our boundaries in this case?" "Are we being mechanical with this best practices approach?" Principles are more effective tools to enforce intention and worldview than are practices. Hew more to the principles than to the practices whenever possible.

We've created a poster of the "Systems Thinking Principles" for you to print and use on our website EnterpriseAgile.biz if you think it will help your team connect to a helpful worldview and intention for transformation.

Test Drive Your Knowledge Again

Answer the following questions with the knowledge you have after reading this chapter. Have your answers changed?

1. What current trends in software development are good examples of systems thinking?
2. When there is a problem, how do you decide who to blame?
3. When are analogies useful? When are they harmful?

References

- Bertalanffy L (1968) General systems theory: foundations, development, applications. George Braziller, New York
- Checkland P, Holwell S (1997) Information, systems and information systems: making sense of the field. John Wiley and Sons, Chichester
- Churchman CW (1979) The systems approach and its enemies. Basic Books, New York
- Deming WE (1992) Out of the crisis. MIT Press, Cambridge, MA
- Deming WE (1994) The new economics for industry, government, education. Massachusetts Institute of Technology Center for Advanced Educational Services, Cambridge, MA
- Hamel G (2012) What matters now: how to win in a world of relentless change, ferocious competition and unstoppable innovation. Jossey-Bass, San Francisco
- Midgley G (2000) Systemic intervention: philosophy, methodology and practice. Kluwer Academic, New York
- Pink D (2011) Drive: the surprising truth about what motivates us. Riverhead Books, New York
- Poppendieck M, Poppendieck T (2003) Lean software development: an agile toolkit. Addison Wesley, Boston, MA
- Rosen R (1991) Life, itself: a comprehensive inquiry into the nature, origin and fabrication of life. Columbia University Press, New York
- Solomon RC, Flores F (2001) Building trust in business, politics, relationships and life. Oxford University Press, New York
- Ulrich W (1996) Critical systems thinking for citizens: a research proposal. University of Hull, Hull

The Journey to Enterprise Agility
Systems Thinking and Organizational Legacy

Kulak, D.; Li, H.

2017, XIX, 286 p. 35 illus., 32 illus. in color., Hardcover

ISBN: 978-3-319-54086-3