The Lengthy Introduction to Fr(agile) Service Management

Dana Stoll
IT-Management Professional & Coach

Contents

1 Introduction

1.1 What´s this (Fr)agility all about?

(Fr)agility is …

… simple, natural and state of the art form of managing IT
… combining agile management methods with traditional IT Service Management (e.g. Scrum with ITIL)
… enabling IT Service Management for volatile markets
… focusing on both: results, people and work
… built upon ethics and a working environment worth participating in… valuing both strategies for their respective benefits.
… the way I work and want to work
… not for sale.

Simple?

(Fr)agility gives you an easy to use guideline through the current difficulties when doing IT Service Management in modern market environments. Like connecting with rapid application development. It doesn't keep you busy with a mess of process chains and organization diagrams. It relies on simple principles, which have proven to be very effective. Therefore, they are independent from

tools or actual implementations. Whoever does things the (fr)agile way, can profit from enabling his own mind to get things straight.

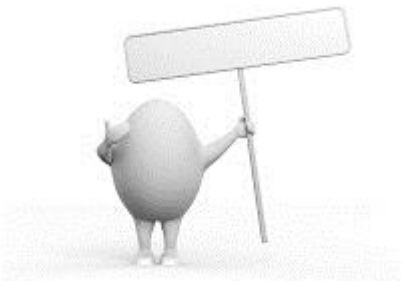
Natural?

(Fr)agility focusses an a view of people and companies as human beings and their natural actions within their environment. (Fr)agility is "green" IT Management :-) So (fr)agility translates complicated terms of IT Service Management into actions which feel like natural behavior to us when we carry them out. Instead of following complicated procedures. We firmly believe that focussing on what people naturally do best and providing decent "defaults" is key to sustainable results.

Traditional?

(Fr)agility doesn't forget about best practice methods which have evolved over the last decades. Instead, (fr)agility presents the important principles of IT Service Management in a modern light, adapting them our volatile markets, which gives you maximum flexibility and allows you to make your own decisions based on your very own situation and experience. The principles and culture which (fr)agility promotes are based on values and beliefs which we deem worth supporting to create a humane environment in a world full of technology. Not following every hype just for the sake of it.

We? Us? I?



(Fr)agility basically means "I", and the term has become my trademark. If I'm refering to "I", then I usually mean myself. I do this because I don't like myself referred to as a method or school of thought. If I'm referring to "we", I'm not cultivating schizophrenia between (fr)agility and I, but am paying tribute to the small group of (to me) very special people who helped in getting the bits and pieces together to comprehend this. Most of the bunch are IT, coaching or leadership people. We're some sort of virtual family. Somtimes "we" or "us" will also refer to you and I. You'll have to decide.

Why do you need to put a label on this?

I wouldn't want to see my first name on everything I do. I also couldn't stand becoming a trend. (Fr)agility deserves a name of its own. Maybe I want to do something else, some when, in the far future.

But this is also not another Agile framework. What's written here is neither ultimate truth nor the only way to do things. I try to find vital points in fixing Agility and IT Service Management together, illustrate what's important and open perspectives how to do it. There are tons of ways to do it successfully. A lot has already been said by Agile methods. Repeating and sticking a different label on it would be a waste. If it works for you, use it. (Fr)agility is nothing more than the way I work. If you think this is "small scale ITIL", okay. If you think this is "Scrum for IT" or "XtremIT", I can live with it. If you think it is "Crystal Clear Red White Server", go for it. (Fr)agility is just a label. It means to fix agile and fragile environments together. That implies making some minor tweaks to Service Management in the early stages of live operations.

Sure, but what qualified particularly you to do this?

It happened as a result of my work experience. During my last 15 years as a graduated computer scientist I have been working in software engineering, running many IT services from game servers to DAX 100 companies, built and headed the IT department ofWEB.DE, at the time the biggest German portal site, for eight years and throughout our major new market crisis, taking additional lectures on management, psychology (and for some strange reason quantum physics), giving lectures on IT service management and ITIL at three universities, even working in controlling, coaching and closely working with some of the most well-known Scrum trainers and IT pros in Germany, even drafted a Scrum certification class for one of them, dug myself into organization theory for doctoral studies (which I somehow never got the time to finish), and been consulting in business informatics for a major German telco company, which I now work for.

To put it short: I encountered these pieces during the tremendous amount of information and work I ate. I don't expect many people to have the exact same schedule. At least in central Europe there is a chance that I know most of them who are just as crazy. Also I usually fix the pieces I find together. At some point I have created (fr)agility, simply to have a label to refer to and collect all this. Besides, the name still sums it up very well.

So, from a top level perspective, what's going wrong?

During the last two decades, professional frameworks concentrated on two factors: maintaining quality (or customer satisfaction at most) and avoiding fraud. Both of these approaches from my experience do not seem to work on a large scale.

Maintaing quality and customer satisfaction has created many quality assurance frameworks, process frameworks, certifications, review processes, assessments and the like. However, they leave you behind with many unresolved issues:

- They work better for *cash cow* environments where everything is ready defined so best practice can be used upon. However there is a huge gap for all product situations which didn't reach that stage yet.

- Innovation always takes place in non-cash-cow environments.

- Reviewing the block busters of online applications during the last five years one can find many examples, where quality was outperformed by time to market. Apparently modern customers are willing to take some slack in terms of quality, if they are able to interact with companies even during the development stages, or simply save money.

- The frameworks have grown huge, they overlap, but just not quite. Thus they bind a lot of effort and brainwork within the same organization, which slows things down, plus requires a lot of people, which you don't always have. So they may become innovation killers.

- Todays markets are fast paced. There has been a shift away from traditional cash cows towards shorter product cycles and market exploitation. Therefore we once more need to reconsider "best practice".

- They abstract business from those who have to conduct it.

How about corporate governance?

Avoiding fraud and laws for enhancing transparency on the other hand seem to have created a vast documentation pile, which is almost as outrageous as the quality framework specifications are comprehensive. Sarbanes Oxley (KonTraG respectively), Basel II all had one consequence: the signed pile of paper which states that things have to be as it has been stated has been reaching new heights. And at the same time the mutually dependent lobby grew, who after signature had to assert truth to the signed paper instead of reality. As we now know, none of them helped to prevent our major financial crises.

Each new framework or layer of abstraction you place upon an organization contains new Chinese whispers about how people at the base really conduct their business. They usually state how things should be, rather than how they really are. Since important people and companies have to testify that things are how they should be, the papers will state just that, because everything else would

mean immediate trouble. If they are not quite how the paper says they should be, then there perhaps may be some trouble at some point in the future. Immediate trouble always wins. And once there is a signature, the lobby of defendants for this version of reality is big. Since this applies to all levels of business right up to the top, the consequences will most likely be "relative", otherwise the "system" would have to jail itself.

And what has all this to do with delivering working IT service processes?

Nothing. People who have to keep stuff up and running take zero orientation for their daily work from all of this. And exactly this is the biggest problem.

Don't tell me agile methods are all good!

No, they aren't. For the most part of it, could even have done better: Leave their own universe, stop pointing towards the "traditionals" and putting the blame on them. Instead of helping to dissect all the mess, engaging in mutual communcation at the same level and thus opening ways for effective collaboration. Which sometimes is difficult, if you have to meet deadlines, I have to admit.

What we need is a proper synthesis of the both, dependent on the portfolio of product situations an organization needs to master, and apply the right tool to the right services. This means bringing agility to IT Service Management, as well as Service Management to Research and Development. And an environment of mutual responsiblity, where people can grow from orientation and thus enjoy contributing, to both.

1.2 What´s the (fr)agile perspective on management?

(Fr)agile management is …

… about people.

Isn’t this approach a little bit obvious?

It obviously isn’t. Karl Weick, a renowned organization expert, already in the seventies wrote, that organization itself does not exist. If you start looking for organization, you end up finding nothing but people and their interactions. This hasn’t changed. However, if you look at current management literature, you find nothing but roles, rules, processes and optimising, and little about people. This only describes what could be done, not what helps people be successful, and not at all what people will actually do.

Even the boundaries of an organization, which we always draw on our white boards as some sort of circle, are visionary. It is the people who act with outside organizations and the environment. In some places, there indeed are automated processes, which assist people with their interactions. There is little operation which is completely automated, and most of it consists of energy or production supplies.

Yes. But processes describe what people should do and give orientation.

From what I have seen, they do not. They are often documented afterward, and seldom used as a guideline for work. I’ve seen many of those process graveyards. If two human beings interact, something way more complex happens than what can be described in process documents. These additional factors have at least as much influence on every single decision and action of human beings as rules and role descriptions.

Process descriptions, social microsystems between people and all other influencing factors (health, family, likes, dislikes, skills, etc.) create conflicts. These conflicts have to be resolved by us working individuals. Adhering to processes is just one piece within the conflict resolution process. Besides, there is a big motivational issue. Many people like to help other people by far more than they are motivated to abide by abstract rules. They cannot do otherwise, because it is a reflex. This is a powerful human trait which can lead to enormous achievements. In terms of procedure you may just call it poor performance.

This isn't really professional behavior, is it?

This is human behavior. If two people interact, they do not behave like Turing Machines, as process charts may illude us. Their behavior is complex. They, more or less, "entangle". This means, there is a chance that they will feel sympathy and start to like each other. This makes communication a lot easier and gives way to new possibilities. In fact, all human life is based on this principle. But it also creates new diversion and places importance on traits like matching ethics, likes and dislikes. If two sympathizers negotiate on a certain subject, there is good chance that they will find some agreement or at least decent compromise, and — for a certain time — will live up to this agreement. Until they become more involved with other "entanglements". During their working routine, these two people will most likely pass the ball on in mutual responsibility, regardless of any process requirements or prohibitions.

Yes, but …

Yes, but. There is also a chance, that the two will not feel any sympathy for each other, their ideas, views or beliefs. This makes conversation difficult, if it's being taken too seriously. And ironically it makes interaction, which we usually call professional, a little easier. If the two converse on a certain subject, it is highly likely that they will not focus on their mutual understanding. But instead of results they may just as well focus on the differences which still remain (and will always remain).

There is never an end to the subtleties of differences in points of view. The phrases which are most commonly heard in such situations start with "Yes, but …". My personal favourite is the completely hypothetical "Yes, but if …", which has no practical relevance whatsoever, but merely constructs illusionist scenarios which might some day at a given time have a tiny chance of creating a problem. People in this state are likely to throw tasks like bullets at each other with an air of "not my job"-attitude. Perfectly justifiable by situational interpretation of their respective role descriptions.

So we behave a bit like quantum systems. There is a certain probability that we will act according to what role descriptions prescribe. But the organization or management itself cannot measure, what this probability really depends on. It remains hidden in our brains, or the mutual brains of our teams, families or any other level of coexistence.

Yes. But therefore we measure compliance, performance and goal indicators, and create incentives

Frankly, this is an illusion. We know that human interaction in organization
creates *emergent* properties. This means new possibilities arise, which are beyond the sum of all our parts. However, on an individual level, we are not capable of measuring them, because they only exist at larger scale.

Even more: If we measure people at an individual level for their performance in away so they are aware of the measurement, we actively ruin organization and its results. I've never seen measurement of poor personal performance take place in organizations which did not directly lead to prohibitive interaction. This always results in poor team interaction, and on a large scale this leads to poor team results. At least poorer as they could be.

Imagine for a minute a team in the following state: One member is, for whatever reason, performing bad. The others, aware of this, decide to make a difference, and by any means do not copy his poor performance. This has a good chance to boost overall performance. If you take the low performer out of the team, the result may just be the opposite of what you expect. There may of course also be a chance, that chit chat about the team member's bad working attitude will distract people more than it's worth. You simply cannot tell from measuring on an individual level. You will have to interact more intelligently if you want to call it "management".

Incentives based on personal performance monitoring are even more harmful. In this case, for our own benefit, we entangle with the measurement more than what common sense would allow for in certain decision situations. You can blame many ludicrous situations in organizations on this. Even with the measurement systems themselves. I've personally encountered the following situation not only once: "Why can't we just correct this number to reflect the proper amount?" "I don't know. Probably somebody's bonus incentive depends on it."

So what do you propose?

Radically speaking: never measure on an individual level and never try to directly influence people's actions.

Instead, at each level of organization (teams, departments, groups, …) provide as much orientation and guideline which is necessary, so people can align their many interactions in order to achieve their mutual targets, as a team. Only people can decide: what they can do best, when they can do it, how they can do it, and when not.

Every team is capable of negotiating this on a daily basis without any or at least with little loss. But the benefit is huge. However only measurable in terms of team performance. If you insist on measurement, do it passively, on team level, and always as a means to improve your own performance, not others. Refrain from drawing quick conclusions on individual performance. Always place developing people's abilities, skills and welfare in life above squeezing for individual results. Create a working environment, which on global scale encourages such behavior.

In other words: If you want great achievements, treat people like intelligent, creative human beings, not like exchangeable role and duty fulfillers. Otherwise, you will never get more than the sum of their parts, and in this case, your organization's only justification of existence left is some shared resource among the people.

1.3 How does (Fr)agility organize IT processes?

(Fr)agility organizes IT processes …

… not.

(Fr)agility ist not yet another process framework for IT Service Management. Pretty much all what can be said on the architecture of IT Service Processes has been written down in readily available literature. You probably know most of it already.

Of what use is (fr)agility then at all?

(Fr)agility is not for use. It only tries you hint you through these huge collections of can-doables. It shows you how to resolve modern conflicts in IT Service Management, and where to place your secret portals for shortcuts to immediate success … Not. It tries to sharpen your view, so you can tell for yourself which patches of these frameworks and management frameworks may be appropriate for your particular IT Service Management situation. If this isn't enough, (fr)agility finds people who can help you get started, but only that. We won't do the job for you. Doing this, (fr)agility focuses on innovative, fast paced product situations, or at least a combination of these with traditional Service Management environments.

Therefore the most prominent piece is some sort of interface between agile development methods like Scrum or XP and strategies of traditional IT Service Management as ITIL is. This is a fairly modern market requirement for which we found many question marks in people's minds out there, and little dedicated literature to start out with. Because only "yet another piece of engineering diagram" will not do this job.

Don't the frameworks already do this? The deal with sizing.

As a rule of thumb, the cake is a lie. Many frameworks nowadays end with certifications. And as part of the certification process, you also subscribe to the framework's code of work ethics. Sometimes they require you to do this even explicitly. I've even encountered contradictory codes of conducts for different certifications, which was the reason, that although studying the syllabus for quite some, I finally left out on many. Lecturing at universities I cannot afford to subscribe to one particular code of conduct and credibly retain an independent approach. I've never seen a certificate doing any hardware, software or people work anyway.

The global trends point toward separation, not integration. This happens, because a theory today is not only one alternative way of thinking about things, but it is a complete school of thought, certification office, brand, market power, philosophy, social body, method of structuring your daily work

routine and general code of conduct at the same time. Since parts of this force have been licensed to other organizations, defending them becomes mutual obligation. Thus we need something a little bit more open than yet another open framework, even if this means leaving the well known path of engineering-like diagrams, which our educational systems dictate us to prefer.

Things become even more difficult, if those frameworks are based on organizations which have been financed on borrowed funds. Investors want to see results, and usually define results in little else than terms of profit. It makes no difference whether this influence comes from financial influence on private companies, political lobbyism or academic sellout.

Well, as coming from the same society, aren't you biased as well?

Of course I am and I hope I am. This is how the human brain works. I would have to fear myself if I weren't. So I have to deal with it. I possess no "ultimate truth" or knowledge for the better. Besides, you're free to go and do things as you see fit, at any time.

(Fr)agility merely deals with different aspects of our professional life and hopefully does it as well as others do their job in their part of the game. If it works, it sometimes makes you think, and then you start to see and do things different.

So what does (fr)agility consider to be the basic principles to have in mind for agility IT Service Management?

There are, in fact, two of them:

A

The Consequences of Complexity on Organizational Contingency

B

The Consequences of the Cynefin model on Service Management

C

A collection of common sense truths which usually are not so common

1.4 The Consequences of Complexity on Organizational Contingency

What on earth is this Contingency thing?

The possibility, that you can deliberately do things. For example a rock. A rock has very little contingency. A dog has a little more contingency. However in resisting its instinct to run after a thrown stick and bring it back, it might badly fail. Humans on the other hand have even more contingency, most of the time, at least.

This does not overcome laziness. It just means they are deliberate in their actions, if the necessity should arise, that they act. This does also apply to organizations.

So what can we do with it?

Everything. Unless complexity hinders us.

I thought Complexity enables us to do things?

Yes, up to a certain point. In this bubble graph an organization is simply depicted as a bubble. "Our" organization and everything which belongs to it is painted in red. Environment gets black. Each circle means there is "one". One means "existence". Thus, if we have one organization, there is one big, red bubble.

If this bubble is empty, it does not have any members. Each member is shown as bubble within this bubble. The concept is recursive. There can of course be more than one member within an organization. Since organizations can also be members let's just call this whole stuff *entities*. Finally, relationships between entities are drawn as connecting lines.

This is standard informatics. Get to the point …

At an initial stage, every organization is built of simple structure. It starts scanning its environment for meaningful candidates for interaction. Because of the organizations low complexity, few members have to deal with a lot of things on the outside. If our internal structure is simpler than our surroundings,

we perceive an abstract image, just like our eye does. So we recognize patterns, which is intelligent.

As our organizational brain develops, structure gradually refines and may interface with environment at higher level. For example whole departments can communicate with departments of foreign companies. Thus increasing complexity step by step enhances our freedom to respond deliberately to our environment.

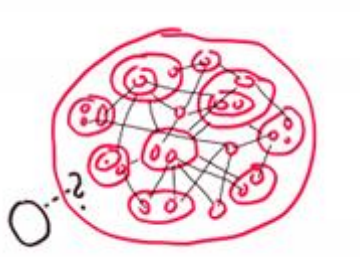
This is nothing new. Again, so why is Complexity a bad thing?

The more complex we get, we need to develop superstructures, which observe ourselves and keep things organized within. Otherwise we would fall apart into our pieces. Or cultivate schizophrenia. So everything within our organization is interwoven in one big mesh.

Processes urge members of our organization into roles and rules, as well as prescribe their interconnections. Or better: all members of an organization will entangle (we had this before), if they interact with others. This is the *Dynamic Structure* of an organization. Dynamic, because entanglements never last eternally. A line then symbolizes this interaction, as long as the entanglement is still active. Practically, there is little difference between processes and entanglement except when looking for somebody to take the blame.

Now get to the point!

Contingency starts to drop rapidly, the more lines we draw between the members within our organization. If entanglements change, the Dynamic Structure of an organization changes. If there are more and more entanglements, decisions and actions of our organization become less deterministic.



The more interconnections we get, the more self-defined and less contingent our organization becomes. We can no longer move. The Dynamic Structure has become rather static. I have even seen companies end up in a state, where building structure seemed to be the main purpose of their further existence, seemingly having lost connection to the outside world. It happens, because decisions and actions are always a result of the existing Dynamic Structure of an organization.No matter how many people speak the same warning, the organization simply cannot hear them, or do nothing about it.

And if we then change the environment?

They will get wet. For it may take days to find out who is entitled to open an umbrella. Let alone who needs to draft new forms for it beforehand. After all, umbrellas do create desires if it rains, so they have to be given out sparsely.

If their market conditions change, there will probably be big trouble. And cake for their competitors.

Again, what does this have to do with IT processes?

Introducing IT Service Management Frameworks has multiple effects on the dynamic structure of an organization:

- New elements increase the complexity, but also give way to new behavior.
- Abstract roles may summarize existing entities, thus decrease complexity, but also vice versa.
- New processes can do both: increase and decrease the number of durable interactions.

Changes to IT Service Management always manipulate the Dynamic Structure of an organization. It depends on your very special situation whether the impact will do you any good.

If your structure is already tied up, and you still add new structure to it, you end up in big trouble. But at least, well structured trouble.

So what would you do?

Whatever you do, always reduce complexity to a point where you can do what is required just fine. That's the simplest solution. The shortest way to express things. The best order. That's enough structure you need. You don't need to outperform yourself, even if it feels good. This keeps you from overstructuring. Overstructuring makes things complicated, lengthy and less ordered.

When introducing IT Service Management components, pick and adapt. Estimate the impact of whatever you introduce to your organization in terms of changes to

its Complexity and Contingency. From my experience, this is your best guide at making the right choice.

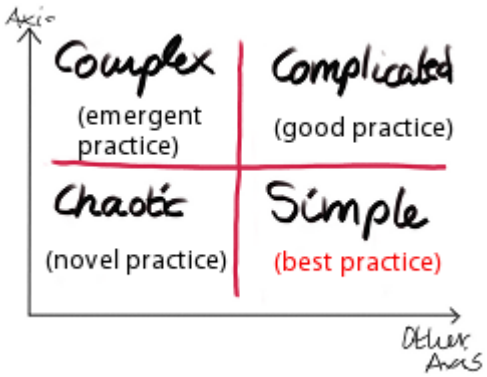
With every role, rule or process ask yourself whether it adds or decreases complexity, and whether it simplifies, enables, mobilizes or paralyses things.

Pick whatever suits your unique situation. Don't forget to consider your environment.

Sure. What is my unique situation?

Have you heard about the Cynefin model?

1.5 The Consequences of the Cynefin model on Service Management



You probably know the Cynefin model by now, for it has been around since 2007. *Cynefin* is a Welsh word and stands for the present or space, which we are living in, taking into account that it results from the interwoven pathways which led us there in our past. What we did has never been wrong. It simply brought us to where we are right now.

The Cynefin model has been invented by Dave Snowden, a Welsh lecturer, consultant and researcher, as a modification of Boisot's I-Space. The model divides organizational environments into four categories: *Chaotic, Complex, Complicated* and *Simple*. The axes typically do not need to be specified, but you could substitute a decreasing velocity of environmental change times knowledge about the composition thereof on *x-axis*, and an increasing level of perceived environmental complexity on *y*.

But this doesn't hold completely, so it's a lot better to use more high quality images to illustrate these stages:

Chaotic

If we start out with anything new, we know very little about it. This is symbolized by our red organization bubble, which shows little internal structure. At the same time, our environment seems to be in a chaotic shape. We cannot recognize any of its structures,

there just seems to be a vast number of objects we cannot assign any meaning to, or discern good from bad. Furthermore what we observe seems to be rapidly changing. Since there is little internal routine and structure to be kept up, we can react pretty fast, and nobody blames us if our first attempts to see the light may seem nothing like funny grimaces to other people.

This applies to almost any new product situation:

- we know nothing about customer acceptance
- we know nothing about trends
- we have little experience from mistakes we already made
- we observe rapid changes in our environment because we can not grasp orientation from its higher level structure

It just all looks like asteroids to us. Everything we do here is novel practice. Without any known places to anchor, we must simply act first, then sense what's happening, and try to act accordingly.

This is usually the stage when traditional IT is requesting a one year development plan from product management for any new product to come to be able to prepare accordingly.

Complex

As our market experience is becoming more mature, we can build up corresponding internal structures, which reflect many of the things from our environment. Not as a whole, but as some sort of abstract pattern. We know how to lift our legs, maybe even stand for a couple of seconds, but we will still have to learn how to even walk. Enough to work with, anyway.

Given our new freedom we realize that there is much more behind things in our surroundings than we were initially able to comprehend, and that everything seems to have its unique features. There seem to be no exact sames and we are almost getting lost in its variety.

At this stage usually a couple of expensive commercial off the shelf vendors come to your door and promise you they have seen, grasped and defeated all the beasts of this rough world, and offer you a software solution which will make it all easy for you to catapult you straight to market leadership. At the same time

they also promise you to have the ideal software solution to administer your internal structures. At least the latter of the two may make you think.

Additionally, your experience may double, if your boss arrives at your office, presenting new targets and recommends you to “grow really fast”.

Bear with me. Only two to go.

Complicated

Things are getting better, when we realize, that many things look the same, behave the same, and we can therefore seperate them from other things in categories. Things are then getting complicated, when we notice that some of them, which basically look the same to us, at a closer look do not quite seem to be as similar as we initially thought.

Anyhow. From the moment we can draw conclusions from higher level categories, we can apply different standards when handling the external objects, which makes life for us a lot easier. After a couple of rounds of corrections and adjustments we can then apply what is considered good practice.

At this point, many talented, gifted, creative people stop working with you, because the job doesn’t have too much to do with creativity anymore, rather than categorizing, administrating, structuring and doing the same things over and over again. Which for some people for no obvious reason and against all better knowledge really identifies with ultimate boredom.

You will have to work through serious struggles between those people who need to keep up the rapid trial and error development of new features of your product to investigate their market acceptance and exploit their share, versus those who are responsible for operating your customer platform. They are separated by “The Threshold”. Typical groups of people who sometimes hardly understand each other are:

- product managers and programmers
- programmers and administrators
- administrators and product managers

- art directors and CEOs with all of the above

But only if they are operating on different sides of The Threshold.
This is also where quality considerations kick in and you should seriously reconsider whether the way you were doing things really was as cool and funky as you initially thought it would be.

Simple

Things become simple, when you're pretty well organized, have developed a higher level of intelligence, implemented a decent routine, trained yourself in perfect shape and possess a voluptuous body of resources. Despite the fact that to your knowledge things are way more complicated than you will ever be able to comprehend you will find your way. Simply by deliberately chosing the level of abstraction you prefer.

If you're lucky and there are enough other people who look at the world from this particular perspective, there is meta-experience which can be shared and distributed, so everybody can profit from it. This is called*best practice*. As there have been many many observations, what works for others will of course decently work for you. **)

***) Best practice will only work if your product experience reached this particular development stage. Best practice comes WITHOUT ANYWARRANTY. Standard disclaimers apply. There is no guarantee that any of this will ever work for you, but recent, industry wide, independent studies have shown best practice methods to perform above average. We can NOT BE HELD RESPONSIBLE if any of this will not work for you. Please do NOT contact us for any problems with any framework we published.*

So what has all this to do with IT Service Management?

The frameworks which have been made for IT Service Management, Quality Management or even Corporate Governance are *best practice* frameworks. According to the model, best practice is only fit for purpose in *Simple* organizational environment situations.

To my experience, there is a huge chance, that:

- you will try to implement best practice frameworks in premature organizational stages just to do it like the big guys

- you will try to apply best practice to non-ready product stages just because "best practice" is "best"

- you will deliberately ignore if somebody tells you you shouldn't consume what the grown-ups do

Don't become childish, we know our business

Perfect. This is the key to do proper IT Service Management. Being able to realistically judge the maturity of your organization in terms of product, market, production and information technology environment dynamics.

All you need is to do an inventory of your organization. Then implement Agile IT Service Management methods for *Chaotic* and *Complex* product(ion) situations (the left side of The Threshold), best practice frameworks on the right. Gradually blend the two into each other from *Chaotic* to *Simple* to avoid organizational and cultural strain.

You're all set. There's nothing more I can tell you now. Please go away.

1.6 A collection of common sense truths which usually are not so common

(The Agility Self-Test)

Growth hurts

Acquiring any new skill is connected to collecting bruises. I don't know why people, who act in business situations, can no longer imagine stumbling and falling on the floor.

Is it because:

☐ they consider them as grown ups who should not collect any bruises anymore?
☐ they are operating with foreign money, and thus just want to be careful?
☐ they think the most profound competitor is the one who collected the least bruises?
☐ falling provides them with a feeling of inferiority?
☐ not falling has been symbolized as success?

Stress kills productivity

If our body anwers to stress, it puts all energy to the muscles, reduces thinking to a minimum and prepares to runaway.

Did you ever encounter:

☐ a situation where you lost the thread just because a sudden fear of losing it seemed to shut your brain down?
☐ entrepreneurs talking to employees in a way so they react similar to the above statement?
☐ a project leader exerting more pressure the closer people were desperately bustling towards deadlines?
☐ superiors deliberately straining their people just to give them less time to think?
☐ people with burnouts resulting from internal competition?
☐ your boss rant about furiously just when you're trying to fix this major crash and get servers back up?

Results arise from not being available

This one may be a little bit tougher. I'll be more figurative. Imagine the following situations, and draw your own conclusions. Which of the following statements do you think is appropriate:

☐ When calling your child at a date, this will contribute, because you have a ton more experience with dates and need to make sure there will be results.
☐ If you know your significant other will repeatedly walk in, push through the rows, ask you a couple of questions and leave, this is the best prerequisite for really enjoying a movie at a theater with your friends.
☐ Your instructor calling you while performing on stage at a local concert hall will ensure a great performance.
☐ Practicing math is best done at a local coffee shop. The audience keeps your focus on avoiding mistakes.
☐ You consider it a decent strategy to place people in open offices and then discipline them to by all means be quiet.

☐ Your boss not being available is usually the reason why you cannot get anything done.

☐ You manage servers best while being observed by at least two people.

☐ You never offered somebody to be available just to distract from not being able to deliver.

Lazy ways of doing things are superior

Now, what would you say:

☐ Cooking dinner is superior to ordering dinner if all you have to do is getting your stomache filled.

☐ Placing toilet rooms behind the house provides better hygiene.

☐ Visiting the local radio station provides better information than lying in your bed and listening to their newscast.

☐ Working at an open office desk provides better results than laptopping on your couch.

☐ People walking about checking their watch will reach their destination earlier than people who are sitting lazily in the train, gazing out the window.

☐ From watching people sit in front of computer screens, you can judge what they are just busy with.

☐ You never clad yourself in an air of bustling activity to relieve the fact that you were behind schedule.

☐ You never inserted blank lines, increased font size or added complicated graphs just to make the same result look like more effort.

☐ You think the figure in this illustration is just a lazy bum who doesn't get any work done.

If something breaks, throw it away

This is the one, big exception to the above rule.

Would you say, that:

☐ If your TV set breaks, it is intelligent to go buy the old model, just because the new one comes with new features and has a completely revamped remote control?

☐ Driving your overhauled, old car will bring you more safely to a destination than a new one, just because you are used to it?

☐ If one of your servers fail big time, it is superior to reinstall the old version from backup than importing the database to a readily available, newer version?

☐ Your administrators are less capable of getting the new version to work than fixing the old?

Ok, next ...

Meaningful things may be simple

Compare the following two pictures. Which one ...

... conveys more information about my work attitude?	☐	☐
... gives you a better impression of my superb drawing skills?	☐	☐
... required more work to be finished?	☐	☐
... would you call more professional?	☐	☐

Did you ever spend 3 hours on a slide for something which could have been drawn in five minutes on some piece of paper, just to look more professional?

Personal chaos is more productive than central organization

Would you say ...

☐ ... people keep their files in disorder just because they are too lazy to structure them properly.
☐ ... people do things in weird ways just because they don't know how to do it right.
☐ ... people keep their desks or desktops in disorder just because they have no work discipline.

Do you ...
☐ ... put your files where you best see fit?
☐ ... do things how you think is best and learn from experience?
☐ ... give a dang about your desk when there really are more important tasks to do?

Would you think ...

☐ ... telling teams how to organize themselves will improve their productivity?
☐ ... telling teams how to organize themselves will make you be liked?
☐ ... centrally maintaining file structures will reduce backup times?
☐ ... cataloguing data is more efficient than searching via google?

Having two people think is superior to just one

In your company or elsewhere, did you ever observe ...

☐ ... people arguing on who is entitled to speak about a particular subject based on position rather than knowledge?
☐ ... people think one is lazy or has serious deficits if two people are working at the same screen for longer periods of time?
☐ ... several departments working on the same subject?
☐ ... these departments sharing their experience on a regular basis and focussing on particular aspects rather than complaining to an abstract force about their lost pride?

- [ ] ... people dive alone?
- [ ] ... yourself thinking your projects do not have as much at stake?

2 The Prime Directive

The Prime Directive of agility in Service Operations (PDASO)

(spoken [pidas…] err … wait …)

Optimise everything for speed of use.

This could further be defined as:

The right of each administrative component to be used at optimum operational velocity is considered sacred, no Operations personnel may interfere with the optimum operational velocity development of administrative components or tools. Such interference includes introducing deferring tools, processes, complexity or any other technology or procedures to an environment whose computing environment or operating personnel is incapable of handling such improvements proficiently.

Operations personel may not violate this Prime Directive, even to save their behinds and/or the ones of their colleagues or superiors, unless they are acting to right an earlier violation or an accidental contamination of said components. This directive takes precedence over any and all other considerations, and carries with it the highest moral obligation.

Never STUbbornly violate PDASO!

3 Making (fr)agility work

3.1 Making the Service Owner work

Finding a proper pendant to the Product Owner you can find in Scrum is maybe the most difficult task to do. There are — of course — many roles which IT Service Management has to offer which could qualify. None of them will do the job without adjustments. Rumours say it also is not easy to get Service Owners "work" and "do the job".

Before defining what the Service Owner actually could do, I would first like to exclude a couple of roles which definitely will not fit, to reduce the sheer number.

Couldn't the Service <xyz> Managers do this?

They shouldn't. Roles like Incident Managers, Availability Managers, Capacity Managers, Service Catalogue Managers, Service Managers or even IT Planners are cross-service roles. If there has been a reason to introduce this role, then there is a need to share this role among other services, as well as to provide it separately from other roles. We need a role which is acting on units of service instead of cross-service functionality. Anything else will create conflicts, which … slow things down.

What about the process owners?

No. Same cross-service issues. Also, process owners are responsible for the design, documentation, policying and assessment of processes themselves rather than Service Items.

The Service Manager maybe?

May be. A service manager, by definition of best practice, is *"a manager who is responsible for managing the end-to-end lifecycle of one or more IT services."*. Now this depends on your organization. If you're an IT Service Organization selling services to external customers, the Service Manager resembles much of what a Product Manager may be, just that your products are Services. This would make your Service Manager a decent Product Manager. It works fine in Situations, where you provide large scale application or hosting services.

The situation I want to discuss is different. I'd like to fix IT Service Management to Scrum. In this case we usually have dedicated Research & Development departments, as well as Product Management in place as internal units or working with outsourcing partners. What we are searching for is more like a sparring partner for a classic Product Owner within the IT department who is as proficient with Operations of particular services in terms of technical issues as she is with their impact on customers, users and business requirements which arise thereof. Instead of a Product Manger, who translates customer needs to producable functionality and cares for their implementation, a Service Owner translates business and product requirements to IT components and cares for their provision and operation.

What about the Service Owner?

This is so far the best match I could find. The role of a Service Owner has been defined in Continual Service Improvement and thus represents a service across the organization, understands the service components, and according to standard definition participates in negotiating or is a stakeholder of many underlying processes like Service Level Management, Asset and Configuration Management, Change Management, Release and Deployment Management, Problem Management, etc. Therefore I'm using this role in my work.

However I take the liberty to make a couple of adjustments:

- Gathering Service Level Requirements from the customer or negotiating and maintaining SLAs with the Customer is according to best practice a secondary role of the Service Owner. I can understand this in best practice situations, where the Service Level Manager does this. In agile environments I would assign this as a primary role to the Service Owner, since levels of service will have to evolve. This also applies to SLA reviews. ITIL proposes yearly SLA review meetings with customers, which shows us, that the cycles which these roles are based on differ from what we need for agility. Our agile Service Owner must operate at least in tactical spans, be present in Tac Sessions or even Ops Sessions.

- The Service Owner will most likely also be responsible for ensuring or negotiating underpinning contracts and Operational Level Agreements.

- At initial stages, the Service Owner will do a lot of the work which an availability, capacity, configuration or change manager usually does. The further a service evolves towards best practice — let alone if you throw it

over The Threshold — the more the Service Owner will only use the figures arising thereof, not generate them or do the actual work.

- The Service Owner in many situations also decides as ultima ratio upon Change, Asset and Configuration Management, or is at least responsible for making or obtaining necessary decisions. A good tip is achieving consensus with Product Owners on such issues.

So that's it?

The Service Owner is in many aspects a match of what the Product Owner does, in terms of IT requirements which are concerned with operating and delivering the service to market, with a focus on technology infrastructure and its operation procedures. She has to function between service unit, business unit and product management, with a focus of translating business and product requirements into actions to take within the IT Service Organization.

The Service Owner can either be a dedicated person, or even an experienced administrator with proper knowledge about the product and outstanding people and business skills. Further information is available in literature about product owners.

3.2 Making the Coach work

Would it surprise you if I said "Don't!"?

... ?!

My shortest definition of a good IT Coach would be "a good leader who does neither manage nor administrate nor do any work except in cases of emergency". Somebody like **House**.

House? He's a psychotic addict with a severe attention deficit!

Right. And he also:

... does not talk about coaching or who and when should be allowed to do what, but just inspires on the job.

And freaks the living crap out of his team!

... develops his team's confidence and personal skills while saving lives.

His team does save the lives. He´s a lazy retard!

... never produces his vast knowledge to his employees but uses his skills to find out what's still missing and offers the right problems to the right people.

That´s the most hilarious definition of clueless I have ever heard!

... does never get in the way of his team doing any patient work except in situations where life depends on it.

He is lacking professional attitude, that´s it!

... still is more proficient on the general subject as any of his team members.

Which is silly! Managers should manage!

... waits for people to come to him other than staging himself (Ok. I'll withdraw this one ...)

Now this really was uncalled for!

He also:

... always coaches his people to do and learn on the job anyway.

Trying to avoid getting his hands dirty!

... speaks up to Cuddy, his boss, whenever he encounters a decision which he deems useless or requires one for the well being of his patients, no matter how unusual or "uncomfortable" this may feel for some people within the organization.

He´s a pathetic troublemaker who has a problem with authority!

... does only do so if the situation really requires and he's absolutely sure.

Placing his judgement above anybody else!

... openly appreciates if people outperform him in specialist affairs (at least ... in terms of his definition of appreciation ...)

To avoid having to admit they are actually better than him!

... secretly enjoys if his team members are more clever than him (prove the opposite ...)

This is hilarious ...

... puts any of the the above over his own position.

Which is only a consequence of his inability to accept decisions!

He also:

... hates administrative paperwork or chores and consequently tries to avoid them, so he can concentrate on what's important.

Flees from anything which could involve work!

... rejects requests from his team which only require own brain work and always accepts personal decisions.

He´s playing with people, can´t make up his mind, and is little assertive!

... is accepted by his team even when he's acting completely unusual because they are always required to make their own judgements.

Knows zip about good manners and leaves them alone with his mess!

... is a generalist with both medicine and people.

Can't make up his mind!

... is passionate about the essence of his work and cares little about bias.

Is a complete bastard when it comes to handle patients!

... likes people so much as not having to please them all the time.

This is your interpretation. Where do they say that? I call him misanthropic!

He also:

... talks about things on the way.

Doesn't respect people or take the time to answer.

... tries to get everybody out of the meeting room as quickly as possible.

Just wants to be left alone!

... can forget about everything else when working on a problem.

Has neither work-life balance nor any discipline or routine!

... ensures his motivation by caring for things which really interest him.

Deliberately neglects everything he doesn´t like!

He also:

... has a couple of very reliable friends, or well, at least one.

He´s a complete loner who is unable to maintain mutual relationships!

... is able to think cyclic, interdisciplinary and making vast use of it.

Is wasting his time in dreams and drug hallucinations!

... doesn't care about conventional behaviour, rules or roles more than is really worth.

Is a complete social retard!

... doesn't put himself above everything but sees himself as an instrument of getting to the point.

Who would respect somebody like him. This is only what he deserves and gets!

What on earth does House have to do with an IT Coach?

If you manage to qualify for all of the above we (and a couple of other people I guess) may be willing to be easy on you in terms of psychotic behavior, deficits ... and maybe even pain killers. Maybe even find you sympathetic.

If — on top of this — your area of generalistic expertise is some kind of mixture between Information Technology, People Dynamics and at least one extra-professional Geek Domain, you will probably be a decent IT coach, if you practice.

Who would need such a retard.

Every IT organization would, really. Two or more, dependent on size.

And what for?!

To solve the usual problems:

- ☐ "We're doing this Scrum thing, but it just won't work with Service Management."
- ☐ "We are really brilliant but our Ops department is slowing things down."
- ☐ "IT and R&D departments are fighting over who is right!"
- ☐ "We didn´t get any new feature delivered during the last six weeks and people are arguing over who needs to decide on administrative accounts!"
- ☐ "Sometimes I get the feeling that they're taking drugs in server administration!"
- ☐ Our tech department is asking us for 12 month plans where we want to switch to rapid development cycles!"
- ☐ "Our Ops people constantly complain about lack of documentation and too little training!"

Or maybe:

- ☐ "Our people are not skilled enough and there seem to be none available at the markets!"
- ☐ "Why are our executives the only ones who seem to think or decide!"
- ☐ "People are sitting in meetings all day arguing about who is entitled to do what!"
- ☐ "Nobody seems to speak up for what he needs to get his job done!"

- [ ] "Everything would be fine if people would just take responsibility."
- [ ] "Product Management only needs to learn how to structure their work!"
- [ ] "I should really be doing this but they won´t let me!"

How about:

To make agile and best practice methods work togehter?

3.3 Making the Team work

I am a hobby musician. All attempts at denying this have yet failed. No matter how hard I try, people seem to notice this even on web pages which I strictly build for business purposes. Teamwork sometimes reminds me of concert situations. I envision being on stage. I'm playing rock music. I prefer playing rock music, because it is simple, but effective in terms of fame. Thus, stage situations are easy:

First, replace in-ear-monitoring with stage monitor speakers so you get a better feel for the atmo. If you then should ever happen not to hear yourself on stage because of the enormous volume, simply play louder. As a consequence, everybody else will also have to play louder too which will contribute to this particular style of music. We will continue on this until our ears enter natural saturation, everything will appear equally loud and thus everything will be equally well balanced. Our audience will like this sound, because it is definitive, simple and they can bang their heads on it. To counter ear deterioration we can simply play louder at the next concert.

The Team is always at the center of attention

This is I believe the standard opener when agile methods come to speak of teams. Running good teams means enabling an atmosphere of collaboration, where people can find their way on their own, learn on the job and communication is targeted at obtaining solutions. The rest will be done by the people. Teams are where work is done. Work is, what produces results, no matter how much work is involved. Thus, teams are, where we need to focus our

attention on, when we want to produce results. Not technology. Not procedures. They only assist.

For the team to do this job, they need to be the best informed people throughout the organization. A good team leader makes sure that they are. This means, not only putting the team as a unit, but also the team's members at the center of attention. So whenever there is a chance, the team members will go and present their results, negotiate with team members of other teams or units or similar. I strongly discourage meta-communication via so-called superiors which place themselves in most meetings and filter the information for their hatchlings. Of course, any leader should avoid being underinformed. But information is shareable. Just because team members are informed, that does not mean, a team leader may not be.

Here´s a box of candies:

- You may find it useful to create an atmosphere of continuous learning, which will be time consuming enough to do in typical, "modern" organizations.
- Team members are usually able to decide on their own whenever possible, they're intelligent human beings. Some organizations are told to train them not to be. You might not want to belong to such organizations. Both can usually be corrected on the spot.
- Delegating results often leads to superior achievements than delegating methods. You may find it difficult to know how somebody else does things best.
- Candies given out for finding solutions or asking intelligent questions taste a lot better than candies for business results.
- Reprimanding mistakes usually is a mistake.
- Your mood may influence people more than their notion of achievements. Using this deliberately belongs to the dark side of the force, except if you're in a very good mood.

- Stability boosts results, especially in highly volatile surroundings. You are probably in an excellent position to create stability.
- Discouraging any behavior which places status or achievements over can-do is vital. Achievements are relics. Only can-do can do.
- Simple sketches are possibly completely sufficient for the basis of discussions, and presentations may be used as little as possible where simpler means are sufficient. Color Paint will do really well when advertising your team.

- It is very relieving to accept the fact that everybody, who does anything for the first time, has no clue about it, and this is perfectly ok.
- If one step after the other is done, this usually results in permanent motion.
- When appreciating effort, you may find it an interesting intellectual challenge to really mean it.

- If teams have expiration dates, changes every once in a while will avoid Statusitis. Otherwise clinging to empty roles or position may ruin the joyful experience of working on one´s own skills. Sitting does not count as a skill.
- If the team will predictably almost never have time to clean up, we can be glad, that digital systems can be recycled way easier than be tidied out. For obvious reasons the collection of slag is discouraged.
- When acquiring tools or changing organization, it saves money also to think in terms of expiration dates.
- Concentrating on not killing motivation is more Zen than trying to motivate.
- Nobody appreciates on the very first attempt that simple, lazy solutions are in fact smart.
- Encouraging courage is encouraged.

3.4 Making Strategy work

In his famous book "Mintzberg on Management" Henry Mintzberg states *"Strategies need not be deliberate — they can also emerge, more or less. […] To manage strategy, then, is to craft thought and action, control and learning, stability and change. […] To manage strategy is in the first place mostly to manage **stability**, not change. (pp. 29-39)"*. This has been written already in 1989. In 2004 John Roberts argues, that for volatile environment situations, the famous Chandler's Dictum, that "structure follows strategy", has to be reversed ("The modern Firm", pp. 27-31).

This is important for us to understand how IT startegy emerges, and what is involved:

Thinking in cycles: Understanding agility in IT strategy

In volatile environment situations, strategy will have to be frequently adjusted as the organization reacts to environmental factors like market conditions, user acceptance, trends and similar. These adjustments, which also include decisions

on IT procedures and infrastructure, are the result of decisions and actions which stem from within the organization. These decisions and actions depend on the structure of an organization, which has already been put into place, as its members are subject to it. And thus, reversing Chandler's Dictum, strategy follows structure.

Sounds freaky. But what are the consequences?

Here's a list of thoughts. This is only valid for environment situations wich involve rapid change, where you usually want to apply agile techniques:

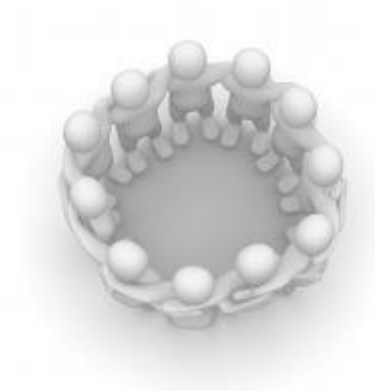
- It does not make sense to plan and structure your Service architecture or operation platform too far in advance. This is even one of the **most dangerous things to do**. If you're doing so, you will influence your perception of the services' performance, user acceptance, because you're viewing it through the goggles which you have created by your administative platform. This picture may have nothing to do with what the user or customer perceives, even though figures may show you that you are "right". There is no right or wrong in perception. On neither side.

- It also tells us that there is some sort of blurry threshold (I'll call it "the Threshold"). Strategy seems to emerge as a program on one side, and it seems to be structurable in plans on the other side. This threshold marks the transition from agility to best practice. **Deciding upon whether a component should be driven on the agile or the best practice side and setting a constraining corridor for orientation is, what a Strategy Session should be about.**

- It also means, you can start out with just any practice, as long as you're willing to review it on a regular basis. Only if you can and will make adjustments based on unbiased observations, prioritising customer and user feedback over any other opinion. Even though some people will not like to hear this.

Can you give examples for the dangers of overdoing structure?

Sure. Here are a couple:



- If you measure your freshly created service in terms of an availability management suite which has already been in place, you are more likely to estimate its performance as poor, the more

components you already have which are running fine under best practice. For your mind will always unconsciously compare them.

- If you are running products from agile development through change boards you will very likely reduce the release cycles far below what market requires. Yet you will perceive the release requirements as chaotic rush.

- If your network infrastructure in place requires a lot of changes to be executed before a new server or connection can be taken online, you will probably cause more side cost than a feature is actually worth at this point in time. So you're likely to estimate the return of investment of this service low, where it would possibly be decent if you'd have done it with properly scaled procedures and technology. This again will influence your future investments.

  This is also the reason when sizing the hardware or procedures of a service for too many users at the beginning, you will have one disastrous period of blame where you will get the impression, that the service is nothing but a waste of money. It wouldn't be, if you's have begun smaller. Our subconsciousness cannot eliminate these investments, as we perceive the money spent in terms of machine sizing, the number of team members, and many more factors, even if we present calculations where we eliminate those costs, likeEBITDA.

- If your decisions are based on a balanced scorecard system which is already in place, then you are most likely to underestimate anything which didn't reach a state which you can properly measure it yet. However nowadays almost all champs stem from such conditions. However your organizational decision structure may already prefer cash cows.

- In complex decision situations you are likely to compare any new feature with experiences you felt comfortable with from the past of your organization rather than how the market is perceiving them. This is owing to the fact that your decisions reflect your organization's memory, which is completely natural. Cyclic development with rapid time to market however needs to have its bubbles, where you can decide independently. Experience is for creating proficiency, not some source of esoteric knowledge.

How can I decide on agile procedures or best practice?

There are several perspectives you can look upon your Service Universe which help you form your opinion:

- *vertically in terms of Services* — This means you review your service in Terms of user functionality (or as a whole) if it has reached a stage where you could apply best practice. Indicators may be availabilty figures, user base, ease of applying changes, service lifetime, revenue streams and many more.
- *horizontally in terms of shared capabilities* — There can also be layers or components of your Service Universe transitioning to best practice, which are shared among or used by many services. This is typically the case for hosting platforms, operating systems, standardised hardware, company wide shared libraries, communication infrastructure components, your company or hosting network, etc. These components possess high affinity to shift towards best practice. This will help ensure your overall quality of service, but can slow you down if you're not careful.

Other perspectives can include: software release cycles, achieved service level definitions, revenue streams and many more.

What's the implication on moving a Service to best practice?

This means that the prerequisites will have to be met for the service to be run under best practice. This will have an impact on release cycles, which will most likely be less frequent. There will be more overhead involved in deciding upon changes. I.e. to gain stability you trade in flexibility.

To put it blank: **Whatever you throw over the threshold of best practice will have a high chance to clash with agile development methods**.

What are techniques you usually use in agile practice, which you avoid in best practice?

In agile Service Management environments:

- programmers will usually have administrative access to live servers
- roll-outs can be decided upon without change advisory boards on an informal basis

- service levels are being defined in terms of service aspects rather than quality figures
- Service Owners will do many jobs which are usually done by Service Level Managers, Availability Managers, Capacity Managers and the like
- comprehensive documentation is almost never available and thus a lot of knowledge will have to be in the minds of people.

How do Strategy Sessions interface with agile development methods?

Part of this is resembled by the review and retrospective meetings, so they are welcome occasions to link them to Strategy Sessions. I recommend to have them monthly at the introduction of a new service, and gradually reduce the number to quarterly sessions as your service becomes more stable.

What are important issues to talk about in Strategy Session?

- The services' **stability** in terms of availability, maintainability, utility, warranty and release cycles
- decisions on components to transition to best practice.
- prospective funds
- upcoming major releases or changes to software or hardware and their impact on architecture or any other issue discussed above.
- risk management, security and service continuity issues
- 360 degree feed-back upon the mutual business relationship

A word on people?

I've seldom met people who work well on both sides of the Threshold. In fact, I know none. Either you're hot for agility, or you're hot for stability. When placing your personnel, make your pick wisely. Believe me, I know what I'm talking about.

3.5 Making Tactics work

This is a stage Agile development methods usually do not really focus on, but which we need for agility in Service Management. Half of what you do at the end of a Scrum Sprint could be considered tactical, for example during your review or retrospective session, or as part of sprint planning. But I'd call the other half part of an evolving strategy, which we can do at a different time scale within IT Service Management. We'll call this the **tactical** level within (fr)agility (which is also common to do in Service Management).

To drive tactics I recommend a per Service (or group of Services with common stakeholders) **Tac Session**.

What do we do in Tac Session?

Tac Session is there to make sure that Services keep running as they are, under the assumption that there are no major changes to its current state or configuration. As a consequence, you discuss in Tac Session what has to be done if the Service is just driven the way it is right now.

Adjustments which arise thereof have to be communicated (and best be decided upon) during Tac Session.

What could happen, so I have to make adjustments?

There are a couple of factors which influence regular operations of a service, even if there are no upgrades to its technology, software or configuration:

- *changes in user base* — could cause changes in CPU, file space or bandwith usage.
- *changes in user behavior* — could do the same.
- *changes in general infrastructure components* — could change constraints for operating a service.
- *changes in administrative workforce* — could effect availability, utulity and warranty dependent on the volatility of the service.

- *changes in maintainability* — regularly occurring bugs caused by user behavior or (maybe unknown) system states could have an impact on availability.

If you keep thinking on this, you will probably come up with a couple more, however I think those are the most important aspects of a service to be discussed.

Who should attend this meeting?

Every stakeholder of a Service should, really. As I have already recommended for fixing Service Level Agreements, it makes little sense to define quality of service in terms of abstract quality figures when doing iterative development cycles or when operating in volatile environments. Therefore providing for utility and warranty depends a lot on mutual trust. Tac Session is your key to organizing stakeholder communication, which can create this mutual feeling of confidence.

Which items should be discussed?

Anything which would impact the quality of service, if no action were taken. What I've written above:

- *Capacity* trends, in terms of CPU, file space, network bandwith or memory usage
- *Availability* figures (min, max, average)
- *perceived Utility and Warranty* by your users and customers

Connecting stakeholder feedback with your trends helps translate gut feelings and service perception gradually into measurable terms. They will be able to tell you if they found acceptable what they got, or not. But don't expect too much in the beginning.

At a minimum, Team, Service Owner, your corresponding Product owner, if applicable, and a member of the development team. and a stakeholder representative which can decide on funds need to attend this meeting, since any capacity or availability issues will usually involve expenditure decisions, architecture or ownership changes.

In stable environments, where you have defined quality, you don't need to talk on availability and capacity trends this often, if proper monitoring is in place. In volatile environments, especially at early release stages: The more often you do this dialogue, the better.

Wouldn't we need to base this on underpinning contracts?

ITIL usualy tell you to discuss availability and capacity issues "as defined in Service Level Agreements and Operational Level Agreements". These of course need to take any underpinning contracts into account. Alas, in volatile environments we're at a loss there. Like strategy is evolving programmatically,

we also have to "learn" the SLA by increasing mutual understanding of our actions and their observed consequences. If there are underpinning contracts, you could consider inviting a representative of your outsourcing partner to Tac Session.

We better start communicating our experience before our stakeholders will tell us "Our experience tells us that we'd rather see availability reports on a weekly basis from you."

How often should we conduct this meeting?

At initial stages I recommend doing this weekly, then gradually change to biweekly and expand as you see fit. This depends on the amount of trust you can achieve between your stakeholders, the quality of your software, hardware platform, administrators, the volatility of your environment, and many more. I really cannot make any better recommendation. You will have to find out on your own.

If the pace works for you, you could connect this to your Sprint review or Sprint planning meetings. But this is not a must.

What do I need to prepare for this meeting?

Meaningful graphs on the availability and capacity issues discussed above. Without them, this is a waste, since you will end up in endless discussions on stakeholder perception which you cannot translate to facts and derive actions from them. Then, this meeting usually ends with any technical personnel promising to do everything better in the future, business people promising goodwill if it happens, and every body is leaving with a feeling of having been completely misunderstood and severe doubts the other part will ever get their share done.

What additional questions could you ask?

You could ask:

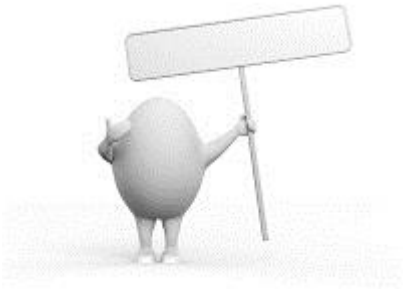
- *your administrator*, whether it felt like a pain running this service.
- *your customer*, if there is anything else you could do for him

- *your user*, if there is anything you could do better, if you could grant her three wishes right away.

- *your product owner*, what was getting on her nerves most.

Foul Eggs

To obtain realistic judgements on perceptions of maintainability and quality of service, I recommend handing each attendant of Tac Session a set of rubber "foul eggs" (or paper equivalents thereof). Upon request, everybody may assign foul eggs to the service by placing them in the middle of the table.



Tac Session should not be finished before any issue related to foul eggs has been treated in a way, that there have been traceable tasks assigned to responsive drivers to resolve the issue, along with a mutual understanding that this solution is the best the team can due in terms of their current understanding of the Service.

3.6 Making Ops Work

This article only descibes the problems with dynamics in administrative operations, focusing on**daily routine within volatile production environments**. As a prerequisite, we recommend the intense, self-aware contemplation of:

- The Prime Directive
- The Second Law of Administrative Dynamics
- The rest of the site above this level.
- Dilbert

IT Operations can be conducted with agility, however there are some important differences to methods like Scrum, or software engineering as a whole:

- In IT Service Management you usually have no pre-planned stories, but services which produce a certain number of expectable (change, service request) and random events (incident, problem). Therefore your stories are more like "permanent issues" which produce tasks on an irregular basis

- You can usually forget about estimating size. Your scale differs from huge to tiny, and there's an imbalance towards the huge and tiny issues.

- Your job usually feels more like juggling balls in the air than burning down charts.

- Burn down charts are of little use. You'll be glad if your watermark won't rise above unhealthy levels.

What about this juggling? What and how do you juggle?

You're juggling standard IT Service Management items:

- *incidents* — will occur randomly. Fixing them consumes a certain amount of daily resources, the more volatile your environment and the hotter off the presses your software, the bigger this amount will be. Usually they're being resolved within the day. They're usually are not a big issue in terms of organization and **don't need to be discussed** in a daily Ops session, **unless** they're **major** incidents.

- *sticky incidents* — could either not be resolved within the same day or will return. They consume more time. You should keep track of those and discuss them at any case in your daily Ops Session, as long as they're not known problems and resolved via change.

- *problems* — will arise from incidents. You should track them in Ops Session, maybe this would be something to estimate. Take care that people who work on problems do not also work on incidents, for this will create conflicts which cost time and quality.

- *changes, service requests* — can be planned just as story items and tasks.

- You can organize them with any means you see fit and scale, if it can order, prioritize respectively tag them and be used by a team. It's not the software, it's the collaborative attitude.

Dependent on the number of Services, when you match agile development methods with Service Management procedures, you may start by replacing "User Stories" with Services, or major releases thereof, and "tasks" with the items described thereof. However there may be a time, where the number of services you manage becomes to large for this to be efficient. You may then either split up your teams, bundle services or convert parts of your service portfolio from agility to best practice operations. But this really depends on your very situation and placing any recommendations therefore on this website wouldn't be too serious.

There will also be a couple of internal projects for you to do to build your administrative infrastructure. Just develop them using regular Scrum, XP or Crystal Clear. That's not something I will discuss here.

How often do you juggle, and what is this Ops Session?

You're juggling at least once, if not twice daily. **Ops Session** is a meeting for operational, short term coordination for issues of daily chores. You typically time box it to 15 minutes at most.

There are three types of issues:

- *operational issues* — which you will discuss daily,
- *tactical issues* — which you will discuss only on a mid-term basis, e.g. every one, two or three weeks, like capacities and availabilities, and
- *strategy issues* — which you will only discuss manually countable times per year.

Ops Session is a daily session for the team to point each other at the most important stuff which is on the way, and telling the others what they don't have to care about because you are going to do it. It's also a good chance for the team to communicate any issues to the IT Coach, which they do not want or don't have the the time or capability to resolve on their own. This meeting can ideally be placed at the overlapping phase at the end or start of shifts, if applicable.

How do you assign roles?

You don't. Whoever is best proficient to do a certain job and free to do, does it. The team can decide on its own upon this issue. Therefore, the team should share its (physical or at least virtual) workspace. However, everybody in your team should have proficient knowledge about the typical roles of standard IT service management, i.e. what an incident manager does, what a problem manager does, how second and third levels work, what availiability as well as capacity management do, and your way of doing configuration management, just to name the most important of them for daily operations. Flexibility arises from proficiency, so this is your obligation.

This has to be done by your people. From an outside perspective, you are not entitled to plan who will be doing what. There is no way for an observer to judge

who will be best in doing a certain job, because this decision depends on the current state of team operations, and how this state will change upon the perception of any alteration. This means, you do not know how the team will perceive whatever you want to impose on them, what this does to their motivation, neither do you have sufficient knowledge to judge it. Even if you try to measure it, this will change the results. I've been discussing this in detail in my introduction on organizational entanglement.

So all you can really do is set a goal for the team (or better let the team set a goal for itself), and provide the opportunities for the team to self-organize towards it. If you want to konw more about how the team works, please continue here.

Who should attend the meeting?

The team should attend. Service Owners and IT Coaches are considered to be part of the team. However, the IT Coach concentrates on issues and people work, and the Service Owner is responsible for providing decisions or answering any questions the team comes up with.

Anybody else might attend, if the team so desires. The IT Coach or Service Owner should make sure they attend. The IT Coach usually also moderates this meeting to stay tuned and collects impediments.

For the rest of the day … just let people do their work!

What should I prepare for this session?

The usual you prepare for agile daily sessions:

- What did I do yesterday?
- What am I going to do today?
- What's hindering me from doing my work?

You should be able to do this session without preparation. It helps having some sort of task board. What it does look like varies, dependent on the size of your organization.

Isn't an Ops Session twice a day time consuming?

Not really, if you avoid discussing content, and rather quickly point each other to the most pressing issues.

Besides, the motto*"Nobody leaves the ship unless unresolved issues have been decided upon"* has proven to be very helpful.

*You should **not** care about capacity and availability **in this session** unless there has been an incident connected to them. Capacity and availability are tactical issues, they're not worth discussing here, since you will not be able to do anything about them, since they usually involve larger scale decisions or involve additional funds. There's a Tac Session to do this. If you avoid these issues, your Ops Session should be short.*

Your primary focus are items of daily business which you will work on at the very same day. Otherwise the usual rule applies: the most eligble person does the job. They can decide for themselves.

What would be typical performance measurement I could do?

If you insist on measuring, you could do the following:

- the number of incidents or problems per week (which gives you an overview of the fire which hits your service organization)
- the number of times people needed to switch their work between two items without finishing one of the both (which is a critical success factor for the efficiency and sanity on your people)
- the percentage of slack the team had to work on problems (which is an alert figure, if it drops, since tech people do like to work on problems)

- the number of pizzas you provided on the house (only if the team tells you they like pizza)

You shouldn't put up any number which the team doesn't deem necessary to achieve a set target. If you do, only use it to work on your own personal performance. If it's not good for working on your own personal performance, drop it.

4 Fixing Things

4.1 Fix Repositories

I have seen many airports in many countries. I have seen many travellers. I am a very open minded person and in principle can imagine quite a lot. However I've seldom seen somebody at the airport studying a detailed construction plan of the building when trying to find their way to the gate. I also cannot imagine this to be a superior strategy, especially when you're in a hurry. — However this is about the level of configuration documentation being asked for in IT Service Departments to give people a feeling of security when administering services.

Remind you, as a direct consequence from the Prime Directive:

It must work when you're really really in a hurry and have no clue how you are able to still catch this flight.

We've already dealt with the documentation part in Fixing Documentation. So let's concentrate on what's important when we think of flights: **Route information**.

In the Late 90s switching started to be a superior protocol to routing, because using static tables enabled faster routing decisions than resolving the routing tables with every packet. Eventually routing and switching were combined to exploit the advantage of both. I still consider this a decent solution.

But what does that mean for repositories?

Let's try to dismantle routing and switching at airports. In this example, the detailed map of the building consists of our complete network topology. It should be obvious that reanalysing topology is not suitable for fast passenger switching. At first attempt, we need something like routing tables. Do we find routing tables at airports? I think yes. That's what the signs with directories do, which read "Gwennair, Terminal A", "(Fr)agility Wings, Terminal D", etc. Wherever we arrive at the airport, we these routing tables and can find the right direction to dash in.

Those directories do not only lead us the way, they also keep us from getting lost (and thus losing time) by hiding detail. But from doing this arises the obligation to follow up. If we trust in these directories and hurry up, further information must follow in time. This information may vary: "Gwennair passengers use

checkins D104-125.", "(Fr)agility Air passengers please proceed to Gate D45", "Mr. President, …"

So Directories list instances which are related to a certain kind of service, and connect them to destinations, so we can make routing decisions when we are going our path trying to find this instance. Having this directory of instances with their destinations at a central place with up to date information is one of the most vital points in operating IT services. Otherwise, you may end up having to explain that you cannot bring this machine back up because you don't know which hosting center (maybe town …) to go to.

Detailed, highly meshed representations of configuration items may work for developing projects, implementing new servers and the like. They are of little use for Ops. **Movies** may create the impression that they do. You've probably seen it. Somebody who has to sneak into building. A team operates in some rundown industrial park backoffice, eats incredibly complex and complicated maps, and digests them into clues for the poor fellow outside who has to find his way. I can find a couple of illusions there:

- The effect is only effective, because those maps create an air of technically insane achievement, where all they really consist of are meaningless projections on some fancy, shiny plastic. Sorry to be the disillusionist there.

- People in there are really cool because it is so incredible, that one can grasp all that information and translate it to clues in next to no time. Despite, most of the time they also really do look cool. This effect only works, because reality usually differs, otherwise there wouldn't be anything special about it. Ops is about getting servers to work, not about being special. I'm not commenting on the looks.

- Movies are timeless. The contents of the screens in Star Trek (2009) will not seem very outdated to us by 2015. The screens of The Original Series may by 2010, however not judging by its contents.

- The poor guy outside is only getting along with this information, because there's this team sitting in his head via radio, or maybe even retina projection. Alas, we do not have retina projection readily available. In most

cases your colleagues will be so tied up with mundane work, that they just can't find the time to coach you your way.

So you're alone out there in this complex world of hosting centre. And you still need to catch that plane.

So if technocracy won't help, what will?

Signposts. No matter whether analog or digital technology. If you know your destination and you can find decent signs on the way, you're all set. This is obvious for buildings, hosting center rooms, racks, stacks, machines. Where this is not as obvious are digital signs, because there is a much bigger variety and they blend a lot more with their background. People find this technique less intrusive. We only need to remain aware of the fact that the signs are there.

You can build your digital signpost pathways, for example like this:

- Hyperlinks from your Service Dashboard to the very next level of complexity
- Standardized installation locations for services and file system links to those things you need to find quick, when you're in a hurry and do not have time to look stuff up
- Consistent, hierarchical typing and naming of files. For example Service.conf can contain the global configuration data for a service. Service.log its global log. Service.Process.log the log of a particular component named "Process".
- In the above example Service.conf could again be a directory, which enables you to find further instances of this particular service.

- Any other, mutually proficient way of doing this will work.

The Second Law of Administrative Dynamics

From this, we can postulate the **Second Law of Administrative Dynamics**, based on the General Directive:

Within an isolated IT Service Universe, the entropy of administrative proficiency will tend to increase over time, approaching a maxium at equilibrial deadlock.

In easier words: Within any company, as only time passes and nothing else is done about it, the can-do of Operations (and a ton of other) people has a tendency to decrease, until it culminates in a mutual deadlock where nothing can be achieved anymore.

As a direct consequence, we get **Administrative (Di-)Lemma #1**:

To retain momentum, an administrator, knowing any destination within his or her IT Service Universe, must be able to find the way through technology as a self driven system.

This means:

Once an administrator encounters any comprehensive name of an instance of a particular Service managed in his domain, he or she needs to be able to locate, pursue and reach it starting from his/her administrative interface without any further lookups than those in directories and signposts, analogue or digital. Whatever knowledge is required to accomplish this task must be located in the administrator's working memory (brain), not any form of swap space (paper, file or database).

It does not make any difference if the information required is meta information on how things are done at your site, keys files which aren't in place, accounts which have been locked out, passphrases, or similar. Administrators who are not capable of following this directive with no exception are subject to report this to their colleagues, superiors and pets immediately. It is not considered heroic to try to "wing it".

Isn't this an illusion?

It is highly effective, and frankly, the only way. As a result, your demand for detailed documentation will drop drastically. On the other hand, if you cannot handle things like this, then I really recommend considering the following:

- ☐ Are you running too many things for you to proficiently comprehend when "time to <*something*>" is a relevant concept?
- ☐ Are you using your administrative infrastructure too seldom, because you are kept up with different things? This may be the case and is perfectly ok. But it reduces your incident skills, and the team must know.
- ☐ Did you tell them?
- ☐ Is your underlying infrastructure necessarily this complex, for example for reasons of security (which I doubt)? Things will be slower, then, also in case of incidents. This will increase stress.
- ☐ Did you tell your organization?

☐ If there is no way around additional tools, they must be highly available, highly reliable, and you must be highly proficient, skilled and trained using them. They must be comprehensive, and if they're big, they need to be thoroughly structured by directories and signs. For example if you use password files. To maintain high velocity proficiency, usually multiple daily use is required. Are you keeping up with this prerequisite?

As an additional service, we recommend making sure your virtual signs have exact copies in the real world, so you can always get up and dash along the signposts whenever there should a sudden need arise to find the real machine in your hosting centre in case of hardware failure. This may become a problem if your administrative universe consists of some 1000 pizza boxes.

4.2 Fix Administrative Interfaces

Sometimes I think this part is so simple that it's not worth while mentioning. Sometimes I think it is so vital to make things function smoothly that it's one of the most important aspects of Service Management at all. It's of course the Prime Directive. But this time we instantiate it.

The Prime Directive of agility for administrative interfaces

Optimise administrative interfaces for speed of use.

This statement contains some vital conflicts which have caused havoc in many IT Service Organizations:

- *personal speed of use* — requires to use just the tools everybody is proficient with best. On large scale, this leads to severe Toolitis, which … slows things down.

- *general speed of use* — is a mild compromise and will, apart from power struggles, terminally lead to Gadget Miraculosis, as soon as all candidates for best compromise

have been considered and dumped. Which … slows things down.

So what then do you consider an optimum?

The optimum Administrative Interface shows the following traits:

- it is only one click away from your administrator's desk
- it has exactly one central entry point per service
- it has only one central entry point at all

This central entry point

- should be redundant (you wouldn't want your administrative interface to be taken down by incident, would you?)
- could, to reduce administrative overhead, be cloneable to individual administrator's machines

And in terms of functionality?

In terms of functionality it should:

- contain the basic performance data of a service, in terms of running state (up/down) with history graph, availability and capacity charts
- links to Baseline Documentation, Component Repositories and Service Notes
- possess an easy interface to write Service Notes
- provide information on the most important contacts for escalation
- provide information on the agreed Service Aspects

Avoid anything else which the team does not consider vitally necessary for rapidly bringing the service back online in case of failure.

4.3 Fix Documentation

Let me start with a story. Once upon a time …

… at WEB.DE we had a ton of administrators complaining about a decent lack of documentation. However, no matter how much docs we actually produced would improve anything. I remember one hilarious attempt at charting our services, a server chart almost filling a wall. The comment of Carsten upon introducing it to me, with a slight chuckle: "Well, it's already outdated …"

At the same time we were recruiting a lot of newbies. In the early 2000s good Linux people were not readily available all over Germany, especially if they should know their way through internet services which **scale**. So we introduced a *boot camp*. Our way of making use of probation. Boot camp meant, the newbies would have an introductory session every other day to one particular subject which was important for running our suite, which lasted about an hour and a half. The rest of the day was dedicated to self study with a couple of links to walk them through. The day after, our incident manager on duty piggy bagged them to get them into it.

The introductories were held by our administrators, each by the one who was suited best. Permanents, who wanted to get a refresher, were free to join at any time.

What happened?

A couple of months later nobody was complaining anymore about a lack of documentation. It vanished. What happened was this: The felt lack of documentation was really a feeling of insecurity. An insecurity arising from the imagination, in case of failure not to know which buttons to push. If you don't know it, you need a source to read up on. If you have no single entry point to get this source within your organization, you are at a loss and feel underinformed, or in chaos. Those introductory sessions gave everybody a refresher on the most important things, which was decently up to date. So everybody knew what to do.

What we really learned from this is: There wouldn't really have been a chance to write this "knowledge" down to documentation. In case of failure, where it was needed most, it would have taken way too long to look all this up, and panic contributed its own share to keep the minds from comprehension. This definitely was not a documentation issue.

Documentation, as we all envision it, seems to me like an illusion. Nobody will ever have the time to keep it up to date, and it can never be both simple enough for operative purposes and comprehensive for administrative reference. No matter how professional a project has been established. I have never encountered documentation on a service, which was not outdated at some point of time. This may work for R&D. If it doesn't work in your program, you ask. But this will be really hazardous for IT Operations, if you rely on such documentation during night shift where nobody else is available to ask.

So how would you organize documentation?

I would split documentation in three parts, always

- Baseline Documentation
- Instance Diretory and
- Service Notes

Baseline Documentation

Baseline documentation contains the basic description of what services are composed of, what they do, how they interact and how you can administrate them. The baseline documentation needs to contain everything which is proprietary, so a person who is new to the company, but a proficient computer scientist or administrator, will get the hang of what we're doing by reading this documentation. However, she still cannot find any servers to administer, since there is **no information of instances or current configurations** in this documentation.

So there are entity relationship-diagrams, class diagrams, component diagrams, process chains for all they're worth, interface specifications and the like.

The advantages if this documentation are:

- Its change rate is very low. Therefore reading it provides knowledge which will last.

- You usually only need to update this with major releases. So it is little work. Little work means changes are good it will be kept up to date.

- Reading this documentation will provide you with knowledge which is worth memorizing. It bridges your mind from theory to practical application without stuffing you with useless details which are useless after your next server upgrade.

Instance Directory

Baseline documentation provides you with server classes. To administer real machines you need to know which instances thereof exist in your hosting centers. For doing this you consult an instance directory. This directory should give you (among other things) also an overview of all instances for any given component class of a particular service.

I will discuss this directory in detail when showing "How to fix repositories", so this should be enough for here.

Service Notes

Whenever you administer a service, for incident or upgrade — given that you studied the baseline documentation, thus you know what the service is about, and you know the instances, thus you know what you fidget with — the most imminent question you will have is: What has been done to this server or service last. Or even last -1, last -2 or last -3.

You don't want to redo anything which has been already done, and in case somebody changed the configuration, you want to know it instead of desperately searching for errors which are just obvious. Apparently, writing this to baseline documentation is of little use. How would you find it. On the other hand, this has to be updated whenever somebody does a change to a server. So this has to be really easy to update.

So what's your solution?

This is what we introduced *Service Notes* for. Service notes are very simple. It is just some sort of change log, mangled with other useful information. It doesn't only comprehend configuration changes, but everything an administrator found worth while noting. You could implement it on a blog, svn history with one file per server, mailbox or whatever. Anything which can have time stamp, message and creator with the ability to sort by time. So if you administer a service, it is very easy for you to look up what's been done to this service during the last two or three weeks. Usually this list is well below 5 items, so the time scanning through it is really not worth while mentioning. If the list is a lot bigger, you are right to be sceptic, draw your consequences, and talk to those people you can identify from the notes. Together with Baseline Documentation and Instance Directory you know everything you must for proper administration of this service.

One final note: The most important thing is that Service Notes are easy to write. Otherwise they will not be comprehensive, which is as fatal as outdated documentation. At the same time, limit the length, so they have content rather than a ton of words to find your needle in. In any case I recommend a central place where whoever does any administrative action can just drop a couple of words into an input field about what they changed, and pull the trigger. This central place should be a one-click from their desktops.

What would I implement this with?

If you acquired some tool which comprehends this functionality, cool, go use it. If you acquired a couple of tools, which together provide this functionality, merge them via <*something*>. If you bought a ton of tools, you may consider reviewing your practice.

Whatever I'm recommending here, my experience is that most administrators will tend to disagree. Do this with whatever you see fit best. This could, at an initial stage, be done via simple web pages. One page per service, neatly designed, with the most important information linked in. Some form of LAMP system with secure shell infrastructure underneath for example. This still works pretty reliable and has enough flexibility. If you're skilled in using links with custom protocol designators, people could even configure their own tools to do the job, if they really must, since the link will only yield the target, and one standard way to do things.

This does sound a bit like home cooking

Home cooked food tastes good, fills you up, is less expensive than restaurants, you know the ingredients and can guarantee healthy components if you prefer, plus the process of cooking provides you with proficiency on operating the kitchen platform. You usually know best how to operate something if you have built some components of it yourself. This does really become a problem once your platform gets so big that adding new functionality creates resource problems or you need to scale this platform for performance reasons which always adds complexity. But homemade solutions for this purpose hold far longer than some vendors and researchers want to make us believe, and the effort of building it has way more to it than being a waste.

But what am I talking. You're the administrators, and know best how to do this stuff. All I'm asking is to put the above requirements over personal preferences and never STUpidly violate the Prime Directive for agility in Service Operations.

4.4 Fix Service Advisory Boards

To fix service …visory, it appe… best … me … abolis… … al… … …

I feel a sudden dizziness. My eyes are getting heavy. Monotonous voices from my telephone receiver gently talk me into dreamland …

…ooOO(I see people involved in email discussions, sitting in front of their screens and getting all worked up in next to no time. They get upset, because they don't see the smiles of other people, as they write their thoughts into digitally bottled messages. I see them reframing their lack of non-verbal **attitude exchange**straight into maximum malevolence. I envision Godwin sitting on a cloud, watching. And I think to myself: "Geez, get up, meet for just the sake of five minutes' digital silence, and get things straight instead of steadily "inviting" more carbon copy spectators to your virtual showdown!"**)**

…ooOO(My vision blurs, and I find myself in a completely different scenery. I see the same people sitting at a huge desk. No, wait. Half of them are sitting at their office desks with headphones on their ears and microphones under their nose. Apparently they are involved in some sort of **information** phishing. They trade facts for figures to do some sort of collection for an upcoming event. To have a proper **decision** base for some kind of action they call "release". I have never seen so many important people at the same time finding slots to actually conduct a meeting. I had no clue collections were the reason for it. I see their secretaries lurk with little understanding pondering to engage their self destruction sequence. It appears to me there is more to this meeting than just information exchange. I have a sudden feeling this is wrong and something strange is going on. Look out, here … I wake up. **)**

Isn't not arguing a good thing?

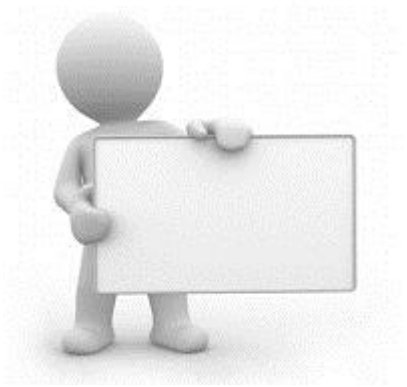
I could see the point that in person communication can help resolve conflicts, and brings a feeling of mutual responsibility. If it were not for the fact, that exactly this is, what the same people usually look down upon as useless child play. But to speak in responsibility, each service has its unique service owner. The service owner has to decide upon what is being rolled out and what not. It is her obligation to collect whatever information she needs to substantiate her decision, in any situative way she sees fit.

I think service advisory boards are really about the chairs. The fact who is entitled to be asked before a decision is made. Being asked as an expert before decisions provides people with an air of being important. Having to be asked as a must before something happens is the ultimate. And exactly this feeling of ultimacy makes me want to abolish service advisory boards altogether. A notion of importance on a bad day will result in a delusions of grandeur. And communicating with deluded people on intellectual traits such as decision making only leads to useless power struggles which create tedious breathing noises especially during large telco sessions.

Doesn't this improve collaboration?

So it appears to me, that being member of a service advisory board is rather a social asset which makes people feel empowered instead of a proper vehicle to empower decisions. I do not say this is the case everywhere. In your company, this will most likely not be the case. I just say there is a tendency, that these struggles arise, and if there is chance they will, probably exactly then when you least need them. At least my risk management would want to get rid of advisory boards under these circumstances.

These discussions lead nowhere, host heavy power struggles, waste a ton of time and money, provide the service owner with exact knowledge about the personal preferences and attitudes of the attendants, and leaves everybody with the feeling: "Well, this was one hell of a tough day, but we **really**managed to get this done." Only nobody knows what "this" really was.



Then the Service Owner has to make her decision, hopefully not without a mailbox full of non-emotional facts to reconsider.

4.5 Fix Service Level Agreements

As I took my first glance at ITIL V3, I noticed two "upgrades": from simple "Service Catalogue" to "Service Portfolio" and from simple "Service Level Agreement" to "Multi Layer Service Level Agreement."

I was seriously asking myself who the recipient for these books is and came the conclusion: it must be "the big guys". The ones to rule them all after small and medium businesses made their maker. Just kidding. Not. While it probably is a decent strategy for encyclopaedic works to enumerate all possible use cases for techniques comprehensively, this showed me that there is no built-in scaling guide to the approach.

In agile Situations, we are dealing with completely different problems, for example: Understanding Service Level. One of the most vital points in successfully launching services to market is to timely grasp appropriate utility, warranty and their translation into terms of administrative action **at all**. Neither administrator, nor product owners, nor financing executives could say what availability the service would need during early product stages using rapid application development techniques. A mutual understanding will have to be developed which can culminate in translating all those gut feelings to something more substantial which we can work with.

Quality is always a result

Therefore, before we discuss this, we could postulate a Law of Quality in agile Service Management which amends the Prime Directive:

Quality is always a result, never a definition.

What do you mean, quality is a result. What are the consequences?

Postulating quality figures as performance targets does only make sense if we are able to use them as guidelines to achieve those targets. Or if we can pressure anybody to pay big bucks if they don't achieve them, and replace our TV set with a big brother cam. This means for quality figures like 99.99% to make any sense we need to know what to do to achieve 99.99%. In product stages which we are talking of in volatile market situations, we most of the time do not have that knowledge, or at least can only provide this for very basic, shared components of our service portfolio.

This has several reasons:

- Quality targets are usually there to enforce penalties upon violations, or to create incentives for avoiding those penalties. This attitude is about as much anti-agility as you can get. It isn't that one big surprise that for environments where agile development methods work nicely, numbered quality targets don't.

- Product managers have no clue yet about the user acceptance. Since the Mid-90s up to 2010 we were concentrating everything on quality. Nowadays we see that time to market and cheap prices can be way more important for user acceptance than quality. But this is only one vague assumption which is far from ready to be put into numbers.

- Without experience with operating the products you cannot say what you will be willing to endure on a long term basis, and what will drive you nuts. Functionality is also subject to change, so there are no stable trends yet on feature (or more exactly transaction) scale performance averages.

- Administrative procedures as well as software systems are far from being stable. This means you cannot translate your administrative procedures, i.e. the concrete actions you take, into prospective quality figures. You simply don't know if what you do is the right thing, and chance just may beat you hard.

- Every one of those .9s behind the comma create an impression of being expensive. Each missing .9 behind the comma creates an impression of trashy performance. In an environment where market acceptance is key but budget is small, this creates a conflict which immobilises people into thinking "I just don´t know …!"

What can be done about this?

If defining Service Levels in terms of quality targets does not work, simply define *Service Maintenance Aspects* in terms of maintenance procedure, i.e. write down during which times there cannot be restarts or updates, even if it's a silent restart, because chance at this software maturity level may be that it breaks anyway. Or write down whether administrators will have to get up at night time if this service should break. Or write down how long they will be tolerated

to leave this incident untouched and who to inform. This will have the following effects:

- These Service Maintenance Aspects give orientation for the administrators what to do which results in intrinsic continuous improvement.

- Your customer knows what your people will do in case the baby should break. If you follow up with phone calls on this impression you're in a good position to build up trust. Trust does not increase availability figures. But it increases perceived warranty by a ton, which can lead to customer satisfaction even in times where quality is not a killer feature anymore.

- If your customers or users are the open public there has never been any other way than doing it like this until you are stable enough to measure customer satisfaction via marketing research. Even if you recruit responsible representatives within your company they will only be stubs.

Administrators will be motivated to bring the service back online as fast as possible anyway if something should go wrong, especially if its maturity is low. Saving the mess provides them with a feeling of being relevant. But be warned: This only works if administrators and software developers for the respective services join themselves for a couple of beers on a regular basis so they know and respect the capabilities of each other and know they're both "the good guys". Providing the beer will help facilitate the experience.

No! This stinks!

I respect that.

Some of them want to use this ...

...

Some of them are stricken with fear of strain ...

Yes, but this will not work for us!
Yes, but our management will not tolerate that!
Yes, but our business does work differently!
Yes, but we cannot change things like this!

Some of them want to evite this ...

Yes, but we cannot afford the time to!
Yes, but our daily business requires different procedures!
Yes, but we will not manage to do that!
Yes, but we cannot pay for this!

Some of them are structuring huge excuses ...

Yes, but that does look like one huge patchwork!
Yes, but we do need a well structured approach!
Yes, but we've been dealing with chaos long enough!
Yes, but we need to advance and become a big player!

Some of them fear of losses ...

Yes, but that is below our requirements!
Yes, but we are moving towards a completely different direction!
Yes, but our orders differ!
Yes, but we need complete control over our systems!
Yes, but our business is way more complex!

Some of them are dreading the winds of change ...

Yes, but you do things like best practice.
Yes, but we already do some sort of ITIL.
Yes, but we really already do things agile.
Yes, but we use agile methods implicitly.

Yes, but this is nothing new.
Yes, but we were doing this all the time.

Some of them are fearing their future …

Yes, but if …

Yes, but when …

Yes, but there could be …

Yes, but chances are that …

So what?!

They thought it was cool ☺

Here's what a couple of others said. Some about my thoughts, some about my work in general. However, most of it in German ...

"One of the best people leaders I know."

Kris Köhntopp — IT Expert, Database Architect, booking.com

"Bei Dana Stoll mischen sich umfassende theoretische Kompetenz mit erheblicher Praxiserfahrung und seltener Selbstreflexion. Das Ergebnis ist mindestens inspirierend — und eröffnet in der Regel neue Wege der Problemlösung."

Dirk Fox — Geschäftsführer, Secorvo Security Consulting GmbH

"I've been impressed by Dana, she has been highly energetic to our company. I am convinced that bringing her method to IT service managment will guarantee the service we dearly need for agile software development already today."

Boris Gloger — Scrum Expert & First Certified Scrum Trainer in Europe

"Die einzige IT-Expertin, die ich kenne, die Menschen führen und Kopfschmerzen beseitigen kann."

Nikolaus Zirwes — CTO, Family One. Ex-Entwicklungsleiter, WEB.DE.

"Vor meinem ersten 2tägigen Training hatte ich zwei Probleme: komplett neue Trainingsunterlagen - und einen Ansatz, wie ich dabei meine eigene Wirkung optimal einsetzen kann. Dana hat es tatsächlich durch ihre Betreuung während (!) eines Trainings geschafft, dass ich mir die "Ähs" abgewöhne und meine Stärken ausnutze. Mein Training wurde so zu einem vollen Erfolg."

Andreas Schliep — Scrum Trainer & Team Coach für Software-Entwicklung

"Seit 1/2000 bis zum Ausscheiden im Jahre 2008 haben Frau Stoll und ich in vielen Projekten gut zusammengearbeitet. Frau Stoll verfügt, neben Fachkenntnissen in der vollen Breite moderner IT-Technologien, über die außerordentliche Fähigkeit, sich schnell und lautlos in komplexe Sachverhalte einzuarbeiten und permanent auf hohem Niveau Output zu produzieren. In besonderem Maße kam diese Kompetenz mir und dem Unternehmen beim Verkauf mit nachfolgender Ausgliederung des WEB.DE Portals mit fast 500 Mitarbeitern zu gute."

Matthias Hornberger — CFO, Kizoo AG (vormals WEB.DE AG)

"Dana Stoll ist mir seit fast 10 Jahren bekannt. Ich habe Frau Stoll als sehr überlegten, redlichen Menschen kennen gelernt, der mit ironischer Gelassenheit die Dinge intelligent, also konzentriert, Wesentliches von Unwesentlichem unterscheidend und Regelmäßigkeiten erkennend, anpackt."

Ulf D. Posé — Präsident des Ethikverbandes der dt. Wirtschaft e.V. & freier Management-Trainer