The Deadly Disease of the Focus Factor

By

Henrik Berglund, Agile Coach ProAgile AB

www.proagile.se

henrik.berglund@proagile.se, @henrikber

There is this small germ that keeps thriving year after year. Untreated it will bring the organizations it infects to an unheroic death. To check your organizational health, answer these two questions:

1) Do you estimate work in "ideal" hours?
2) Do you follow up on your estimates, comparing it to how many "real" hours work it actually took to get something done?

If so, you may be in big trouble. You are exhibiting symptoms of the lethal disease of the "focus factor". This is how the illness progresses:

"There is this small germ that keeps thriving year after year. Untreated it will bring the organizations it infects to an unheroic death."

Speed of development will keep dropping together with quality. Predictability will suffer. Unexpected last moment problems and delays in projects are common. Morale will deteriorate. People will do as they are told, but little more. The best people will quit. If anything gets released it is meager, boring and not meeting customer expectations. As changes in the business environment accelerate, the organization will be having trouble keeping up. Competitors will take away the market and eventually the end is unavoidable.

This article will show you how to further self-diagnose and treat this disease.

The start of it all

Here is what might happen: As a team we might try to estimate some work. But whose hours are we going to estimate? People may have different skills and knowledge related to the work. Perhaps even several people will help out with the task as needed. How will we take that into account? And how much interruptions should we plan for? This might vary unpredictably. And what if we really don't have much experience in this area at all, should we pad our guesses then? This is getting hard, lots of questions…

But hey, let's sweep this under the rug. Let's estimate in "ideal hours"! It will be a metric stating how long it would take a non-existing person to do the work under non-existing conditions. That is, being able to really focus on doing it without "disturbances" or "surprises" (the latter a.k.a "learnings").

"It will be a metric stating how long it would take a non-existing person to do the work under non-existing conditions"

Ignoring all these problems will let us get on with it. Great! Perhaps we will come up with a best guess of 55 "ideal hours". Actually these 55 ideal hours would be better referred to as 55 "smurf-points". That name better reflects that any perceived correlation between these "ideal hours"/"smurf points" and the actual time used to complete the work is likely due to dropping quality and cheating. That is, not from any information present in the metric.

Anyway, let's say that we later together logged 100 actual hours to complete this work. For sure, someone would then come up with the idea to compare estimates and actuals. In this case this would give a "focus factor" of 0.55.

So now the problems start to accelerate, because once metrics like these are available, people will start to use them…

It gets worse

With this new metric in place, it is only a matter of time before a manager will notice that 0.55 does not look very "efficient". Seems people are spending half their time surfing the web, having too many meetings or being unproductive in general!

Interpreting the metric like this will quite naturally lead to some actions to make things more efficient!

Trying to make things efficient though, is not such a good idea as one initially might think. Even in a factory environment, trying to be efficient is a well-known way to create sub optimizations and inefficiencies. Check any book on lean thinking if that does not sound familiar. (Focusing on cycle times and flow is what you may want to do in that environment.)

More seriously, software development is not a factory activity. It is not about pushing materials, such as "requirements" through a production line. It is not about always getting predictable results with low variation as output at the end of the line.

"Software development is not a factory activity. It is not about pushing materials, such as requirements through a production line"

Software development is about delivering value to customers. The main activities in this is learning and problem solving. We need to learn about what would bring value to customers. We also need to learn about the solutions that do that and we need to learn about how to work together to do it. To stay competitive, it is likely that a higher level of creativity and innovation is needed than before.

To make it easier for you to self-diagnose and treat attempts to "increase efficiency", the next sections lists some common examples.

Failure strategy –Efficient Experts

Here is an idea often found in infected organizations: Only assign each work item to the expert most qualified to do the job. They will then get better and better at that type of work and can perform it with very high efficiency! They should also sit in their own offices so that they would not be disturbed by questions. Time spent silently hacking away at the keyboard surely would count as time 100% efficiently used?

But, learning will suffer when people do not communicate. Queues of work will accumulate before experts. Others will have to wait for them. Lead times will go up. Feedback from the market will be slower. Less value will be delivered. Bottlenecks will not get resolved because nobody else gets the option to learn. People will get de-motivated by not delivering any real value.

"To focus on efficient experts will cause inefficiencies and eventually organizational death"

To focus on efficient experts will cause inefficiencies and eventually organizational death.

Failure Strategies – Efficient Teams

From team point of view, with this new metric in place, it is now clear that getting to a "focus factor" of 1.0 is an important goal for the organization. So, how can a team achieve that?

For sure teams will swiftly figure out that if they complete all work at the exact estimated time this will be the ideal! I.e. 100% efficiency! Below are some techniques that teams frequently apply to do that.

Failure Strategy – Play it safe

To hit your estimates as a team, add buffers to all your estimates. Take on less work to make sure you will not be running over estimates if unexpected problems occur.

Also, do not try to innovate or step outside what is already known. Do not question requirements. Stick to the most well-known way to implement them.

But, if teams play it safe, taking on less work than they think will be possible, less work will get done! At least in an infected company, where any free time will probably not be used to add value. Also, estimates that have been padded will be even less useful for making longer term forecasts.

Also, to not question requirements creates severe problems. This is since most "requirements" are actually just bad solution proposals. To optimize value created it is vital to keep asking why. We do this to figure out what the actual problem is that the software is trying to mitigate/solve.

To do this we should work closely with users and various experts to collectively come up with an effective solution.

"The work needed to implement something has a very low correlation to the value delivered.

This is something completely different than focusing on "implementing requirements" and hitting your "estimates". The work needed to implement something has a very low correlation to the value delivered.

Failure Strategy – Just Mark it as Done

To hit your estimate: When you have used up the estimated time for an item, just mark it as done! Don't bother if the current version has a few bugs, no automated tests of if it is rather hard to understand.

But, lowering quality to hit estimates leads to deteriorating code bases. The code will be harder to understand and it will break more easily. Work will slow down more and more and it will cost more and more to add new features.

Moreover, half done features will cause defects and make development more and more unpredictable. That is, trying to improve predictability by following up on estimates will cause less predictability.

"Trying to improve predictability by following up on estimates will cause less predictability"

Failure Strategy – Focus on Backlog

To be efficient: Remove all time for experiments and learning. Focus on working down the backlog.

But, many great products have been developed as "black ops" or "skunk works" not sponsored by anyone. Previously this was possible since people had a lot of slack to fiddle with stuff on the side without anyone noticing.

"It is the larger leaps that have defined history so far. Trying to be more efficient clearly is the opposite of this"

Nowadays, with frameworks like Scrum and Kanban, there is no place to hide. It is transparent what everyone is doing. So, even to sustain the innovation level of yesterday, explicit slack has to be introduced in the process. If not, the next generation products to save our companies may not emerge.

Incremental innovation is all good, but it is the larger leaps that have defined history so far. Trying to be more efficient clearly is the opposite of this.

Failure Strategy – Mathematical Planning

Many Scrum Teams also quickly figure out how to use the focus factor while sprint planning. Just add the available time planned for all "resources", multiply with the focus factor and compare to the estimates for upcoming work. Voilá! Sprint planning in 5 minutes. Just a mathematical calculation and no need for tiring interpersonal communication and collaboration.

But, the idea with sprint planning is that a team works out how to creatively and productively work together towards a specific, goal. A goal that is consequential and related to value.

"A real sprint planning session is a very different type of session than a mechanistic, mathematical handling of hours, factors and resources"

After the session, the team should have a first version of a plan on how to do that. A plan to optimizing value and learning. The plan would then keep evolving as they update it each day during the daily scrum. Always reflecting their best current understanding on how to reach the goal as a team.

Thus, a real sprint planning session is a very different type of session than a mechanistic, mathematical handling of hours, factors and resources.

Symbiotic viruses

There are some related viruses that often co-exist with the “focus factor” disease. One of them is the habit of referring to people as resources, or as full time man equivalents.

So, does it really matter if you refer to a person as a “resource”? Here is an experiment for you to find out: Tonight, tell your spouse, or someone else that is close to you, that they are merely a resource that you will utilize for some purposes. Also, tell them to relax. As long as you see them as efficient, you will not swap them out for a newer version.

See what they think about this. Then ponder if this is the view that will stimulate value creating, learning and innovation in your organization.

“Being thought of as a small cog in a large machine that will run efficiently may not be the most uplifting view to some people”

A related virus is that of the “Scrum machine” or “delivery machine”.

Infected people see the organization as a delivery machine that processes requirements and that should be run with high efficiency. Being thought of as a small cog in a large machine that will run efficiently may not be the most uplifting view to some people. For me it also demonstrates complete lack of understanding of the current challenges and likely solutions in our industry.

Summary of the organizational decay

By using ideal hours and “focus factor”, you can achieve all this:

- Reinforce an ineffective view of persons and organizations as machines
- Reduce productivity
- Reduce predictability
- Reduce learning and value created
- Kill the spirit of teams and individuals
- Kill creativity and innovation

Quite an impressive impact from such an innocent looking little bacteria as the “focus factor”!

Treatments

So, what then are some alternatives to “ideal hours” and “focus factors”? Here are some patterns used in healthy organizations:

For sprint planning, just facilitate the planning process. As the team keeps adding things to the sprint, ask them from time to time if they have about the right amount of work now. Also ask if they have a plan on how to get started doing it.

For release planning. Just ask the team to lay out the work over some columns with weeks/months on a board. Then ask them if they think their plan looks realistic.

For bigger estimates, check out blink estimation.

For follow up, just count your stories/features and draw a burndown.

For essential work not directly related to features from some backlog, set aside 10-20% of the time for people to work on this in self-organized communities of interest.

One final crazy idea

If you think that your organization could benefit from delivering some highly innovative and valuable solutions, you might also toy with the idea of actually giving up on predicting the future and focus on optimizing value and learning instead.

If that sounds too extreme, perhaps just try some of the treatments from the previous section for now ;-)

"You might also toy with the idea of actually giving up on predicting the future and focus on optimizing value and learning instead."

Get well soon!