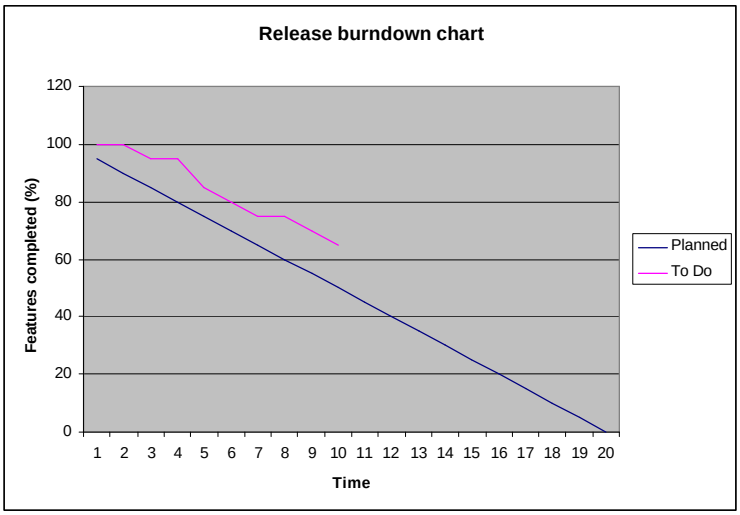
# Bob the project manager thinks about systems

## *The story: opportunity knocks*

This was Bob's first project as a project manager. This was the one where he could show that he had what it takes, the "right stuff". He could finally apply all he'd learned in manager school and from all those books. This would be a great project. Jinnie, an experienced project manager, was to be his "mentor". Whenever he wanted, he could ask for her help. Not that he thought he would need any help, but it was reassuring to know there was someone he could call on.

## *The story: the first iteration*

The team started to work on the first release. About halfway through, Bob noticed that things were not going as planned. Bob had charted progress in a "burndown chart". Half of the time had been spent, but they had implemented less than half of the planned features. It looked as if his team would not be able to deliver all the features planned for the next release.




Bob didn't panic. This was the moment he'd been waiting for; this was the moment to apply his knowledge and skill. Something had to be done.

## *The first system*

Bob decided to apply "Systems Thinking" [Senge 1990] [Weinberg 1997] and build a systems diagram that described his team. With this diagram he could figure out how to make changes. But where should he begin? He could ask Jinnie.
"Jinnie, can you help me to make a Diagram Of Effects (DOE) to better understand this situation?" Bob asked.
"Sure Bob," Jinnie replied, "What do you want to know?"
"Where do I start?"
"First, tell me about your situation".

| |
|---|
| Tell a short story to give an overview of the situation. Go into more detail as we ask further questions of the storyteller. |

Bob told her what was happening.
"Then," Jinnie added, "try to find some observable variables that seem important in your situation. What are the important variables here?"

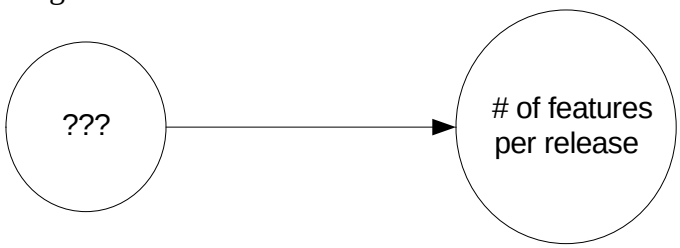
"The most important thing right now is how many features will be implemented this release", Bob said, "that's obvious!"

| Look for a few observable variables that seem important to you. |
|---|

"Okay, we'll start with this variable". Jinnie started drawing on the whiteboard. "This is the number of features your team implements per release. This is a variable we can measure."


| You don't have to be able to measure variables exactly, but you should be able to see clearly when they change. |
|---|

"We'll find more variables as we go along", Jinnie continued. "What are you looking for?"
"We need to deliver more features than the team is implementing right now", Bob said.
"So, you're looking for something that has a positive effect upon the number of features your team delivers each release." Jinnie started drawing again. "Are you looking for something like this?"

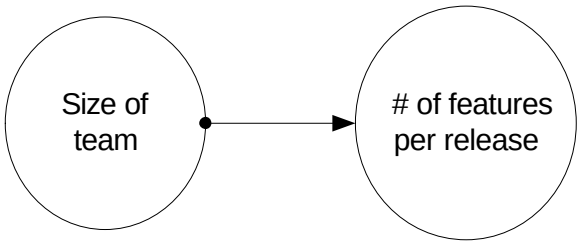



| Look for links between variables.<br><br>The arrow indicates that the variable on the left has a positive effect on the variable to the right. If the first variable gets larger, the dependent variable gets larger; if the first variable decreases, the second one will decrease too.<br><br>We don't need to know the exact sizes of the variables, nor the mathematical relationship between the values. |
|---|

"Yes!" exclaimed Bob, "Tell me how I can get my team to produce more features per release. What's the mystery variable?"
"I don't know", Jinnie replied. "Think about your team, your situation. What variable could have this effect?".

## *Not Brooks' law*

Bob had read his classics: he knew from [Brooks 1995] that "Adding manpower to a late software project makes it later." He quickly drew a "Diagram Of Effects" of this famous law:



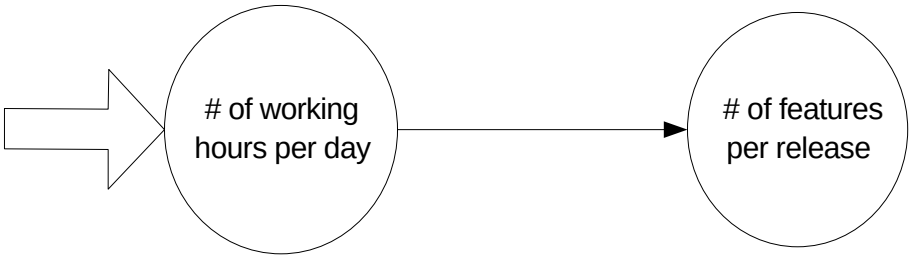

The circle at the base of the link indicates this is a negative effect: if the variable on the left gets bigger, the variable on the right gets smaller.

"Brooks' law is a bit more complex than that," Jinnie smiled. "We'll draw a DOE of that law later, when we've had more practice."
"And anyway, the variable depicting the size of the team isn't very useful in this diagram, as we can't change the team", Jinnie continued. "Team size will be constant during this release, so it won't have any effect. We might as well remove it from the diagram."
Bob erased the "Size of team" variable.

Always try to simplify diagrams.

## *Finally! An intervention point.*

Bob was thinking: "What if we all worked a bit longer? We could get more done every day, we could catch up with the plan!" Bob drew this system on the whiteboard:




"Here's a variable I can control!" exclaimed Bob, "I can ask my team to work a bit longer, until we catch up with the plan."

"Number of working hours" is an "intervention point": a variable you can influence to effect changes in the system the diagram describes.

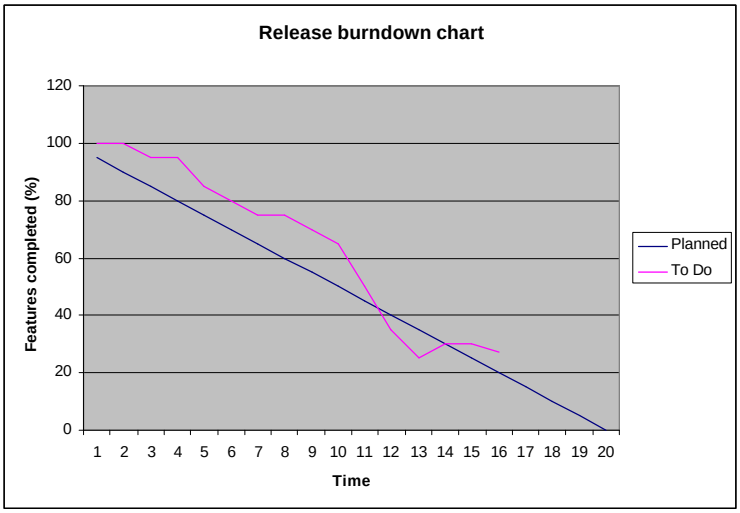
"Be careful with this intervention!" Jinnie warned.

## *The story: It works! Or does it?*

Bob didn't hear her warning; he had already left her office. He asked his team to work a bit longer. The people in the team, eager to make this project a success, agreed to work one hour extra a day, until they were back on target. And so they did.

Bob was happy: he hadn't panicked; he had thought it all out and acted decisively. His team were diligently plugging away. He could see that they were getting more work done. They would be back on schedule soon. His system acted as he thought it would. This Systems Thinking stuff was really working! He was really in control.

But after a while, Bob noticed that the team started to deliver fewer features. If they slowed down liked that, they wouldn't be able to deliver as planned. Something had gone wrong.



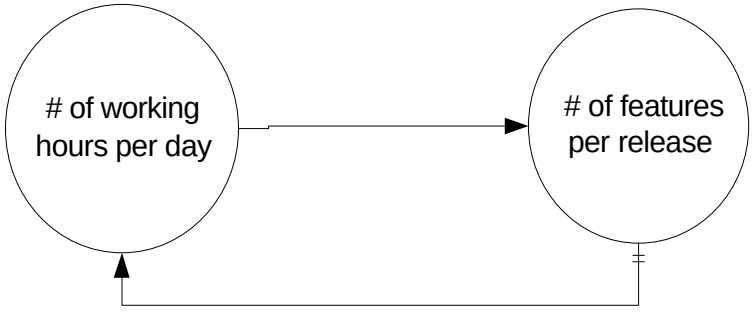

## *The story: More overtime and a party at the end.*

Last time the team went too slow, he had asked them to work longer and it increased the number of features they got done per day. He would have to ask them again to work longer hours. Two hours overtime per day and they would be back on schedule in no time! It had worked before. It would work again. The system told him so.

The team agreed to work two hours overtime a day, until they got back on schedule. And they began to deliver features faster again. Bob owed them. He would throw a big party at the end of the project to thank them for their effort and dedication.

## *An addictive system*

Bob was worried. What would he do if the team slowed down again? Ask them for more overtime? There are only so many hours in a day. The Systems Thinking instructors had talked about feedback loops and "addiction" archetypes [Senge 1990]. Was he getting addicted to driving his team harder? He looked again at his system.




The two lines at the base of the link indicate that this is a long-term effect: the effect isn't immediate.

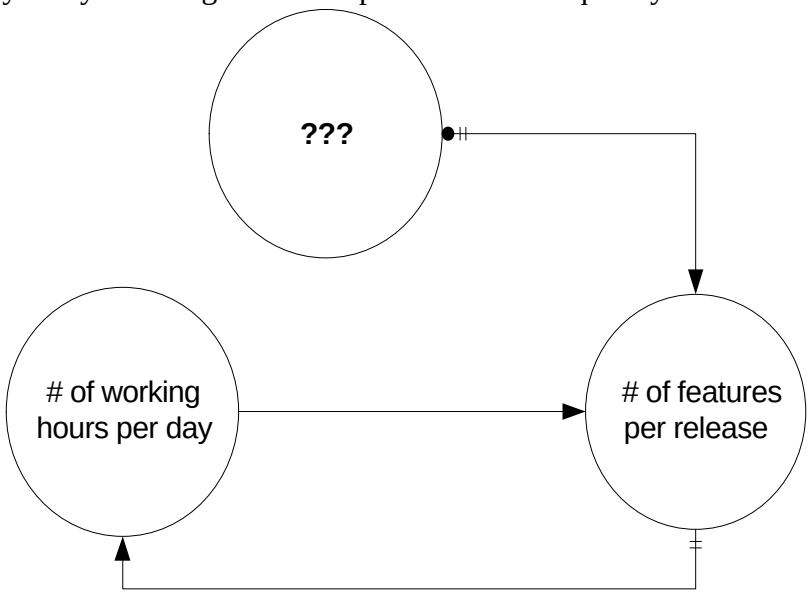
How long is "long-term", how soon is "immediate"? It depends on the system. In this system, short-term effects are visible within days; long-term effects are visible within weeks.

There's a reinforcing feedback loop between the amount of overtime and the number of features. The more we use overtime, the more we're likely to use overtime, following the motto "If it works, do more of it".

Bob saw a "reinforcing loop" appear: when he increased working hours, the team delivered more features. Because this intervention had worked, he became more likely to increase working hours again. He was getting "addicted" to increasing working hours!

## *Something's missing*

Bob felt uneasy: he had missed something, but he didn't know what. Time to ask for help. He went to Jinnie's office.
"Jinnie, can you help me to understand why my intervention seems to backfire?" he asked.
"Sure, Bob", Jinnie replied. "Tell me what happened".
Bob told his story.
"There's something in your story that worries me", Jinnie told him "A**fter a while, you noticed that the team started to deliver fewer features**. There's nothing in your system diagram that explains this". She quickly sketched this system:




"Something has a delayed, negative effect on the number of features the team delivers." Jinnie explained, "You could go on raising the number of working hours (actually, you can't), but this mysterious effect would undo the positive effects each time. Now, what could this variable be? Tell me more about your team."

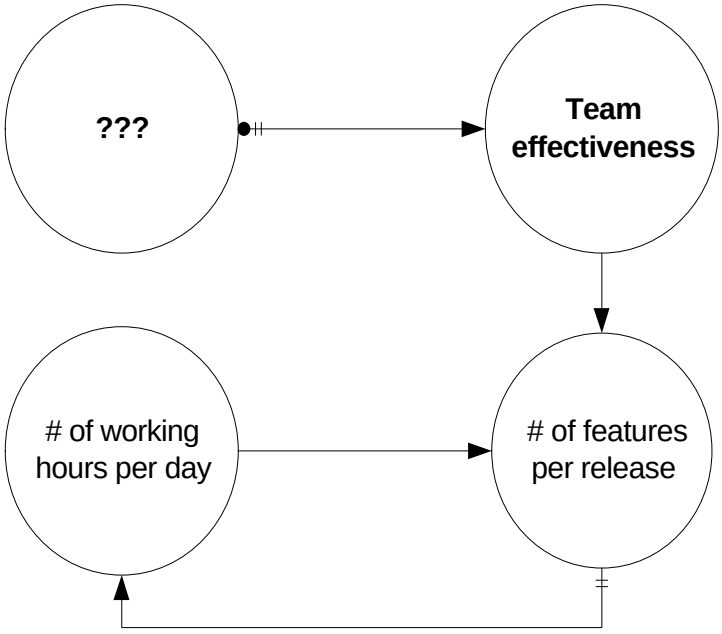
## *The story: It's THEIR fault!*

"I've noticed that the people are working slower than they used to.", he told her, "They are making more mistakes: the bug count is rising. They are getting short-tempered: yesterday there was a huge row over architecture. They have trouble concentrating: meetings drag on and on. Maybe they don't have the right stuff; maybe they're not motivated enough. They don't seem to understand how important this project is."

“If only I had a better team, I would have been able to meet the deadline!” Bob said angrily. “That’s a fine mess they’ve created! And, as usual, I will have to solve the problem. And I have to solve it now! You MUST help me!”
“I’m sorry”, she answered calmly, “but I won’t be able to help you any more today. I’m just too tired and I can’t concentrate well. Let’s complete this diagram tomorrow, after I’ve had a good night’s rest.”
Bob grudgingly agreed. One extra day behind schedule wouldn’t make the difference.

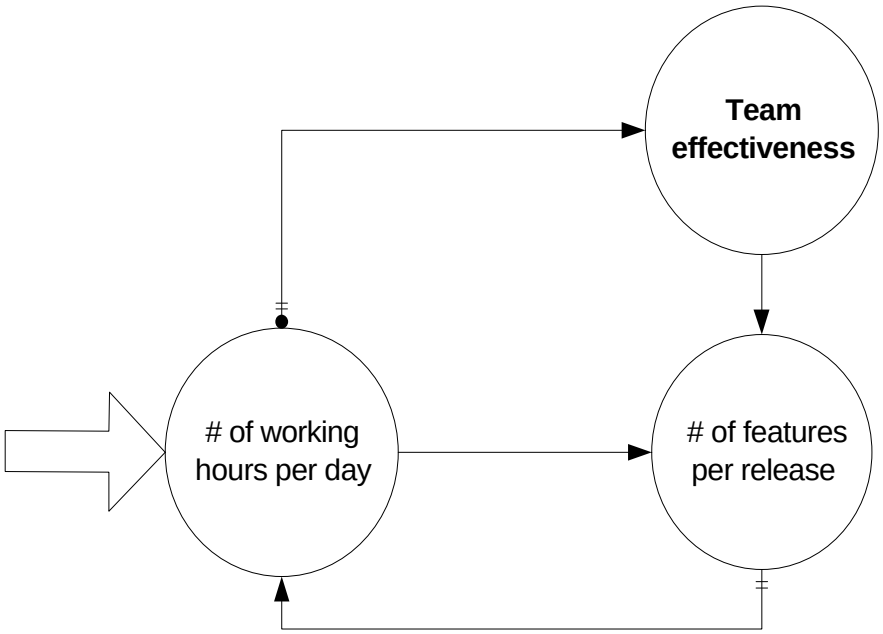
## *Adding “effectiveness”*

The next morning Bob and Jinnie started to extend the diagram: they added an “effectiveness” variable. This variable indicated how well the team worked, how fast they could implement features.
Could Bob influence this variable? Could he make his team more effective? He looked at his team. It was clear that they weren’t as effective as before: features were taking longer to complete, bug counts were rising, and discussions and debates were less productive. What was causing this?




## *I have seen the enemy. And it is me.*

“Now, Bob,” Jinnie asked, “what could cause the effectiveness of your team to drop? What would make them work slower, make more mistakes and hamper communication?”
And then it dawned on him: he was.
All of this overtime was tiring the team. They were showing the signs of the strain. Even Bob was feeling it: how could he not have seen this? His use of overtime as a “quick fix” was backfiring and was making things worse. He was caught in a “vicious loop”. He quickly drew this diagram to explain his insight to Jinnie:




There are two loops in the system: a short term reinforcing loop (below) and a long-term negative "balancing" loop (above). Each time we try to improve the system, it works for a while and then the system "fights back" and undoes the effects of our improvement.

This is a common "archetype" (a recurring pattern) when we try to solve a problem by working on the symptoms instead of the causes of the problem.

He knew what he had to do:

- Stop the overtime. Break the loop.
- Let the team relax a bit to remove some of the stress. That would take some courage: he would deliberately let his team slip further behind the schedule. This sounded like professional suicide. It was madness. But it had to be done.
- He would have to talk to the customer and his manager about reworking the planning. The original plan had clearly overestimated the speed of his team. Maybe "Number of features planned per release" was another variable that could be controlled?
- He would start thinking about ways to improve the effectiveness of his team. What could he do to achieve a **lasting** increase in effectiveness? No quick fixes, this time! He would try to find more important parameters and effects. Each time, he would devise some experiments to verify if his new system corresponded with reality and yielded some useful insight.

## *The story: a party!*

"This project management and systems thinking stuff is not as easy as it looks", Bob said. "But I'm learning."
And then he had an idea: "Why don't I throw that party now? That'll relax the team. Will you come to the party, Jinnie? We could start looking into ways to affect team effectiveness."
"I'd be glad to come to the party", Jinnie replied. "But just to have some fun and get to know the people in your team. Tomorrow we can continue to examine this system. Maybe some people in your team might be able to help us."

## What did Bob learn?

Making a useful systems diagram takes time and effort. The diagram gets improved in several iterations. Each new version is verified against the situation.
What system diagram you end up with, which variables and effects are included and which are excluded depends on what you want to use the diagram for. Here, we were looking for important results (number of features) and parameters we could influence (overtime and effectiveness).
Beware of short-term improvements, because they may cause long-term problems. It's important to search for these nonobvious delayed effects.
Overtime is only acceptable in very small doses, so that we don't suffer the long-term negative side effects of it.

## References

[Brooks 1995] "The Mythical Man-month. Twentieth Anniversary edition", Frederick P. Brooks – Addison Wesley 1995
[Senge 1990] "The Fifth Discipline", Peter Senge – Random House 1990
[Weinberg 1997] "  Quality Software Management: Systems Thinking", Gerald Weinberg – Dorset House 1997

## About the author

Pascal Van Cauwenberghe
pvc at nayima.be
http://www.nayima.be/