Managing Projects That Never End

Alexey Miroshnikov, InfoStroy Ltd.

Dyalog 2017, Elsinore, September 10-14

The Subject. The Project.

A particular complexity of that kind of projects is that it does not have a clear finish . At a glance it might looks as like we are in an infinite loop of patches like that donkey walking around the water pump.

For the last 25 years, we've had to deal with at least one of that. It is a portfolio management software solution.

Financial market is evolving so fast that a software company entering the market actually start a project that never end.

Many traditional values and parameters does not work for endless projects. Most painful are the project goal and the team motivation. ⌘

The Subject. The Project.

⌘ The team motivation is a real problem as the motivation is closely related to an achievement, and what would be achieved at the finish line that doesn't exist??

When we started our endless project in the beginning of 90s the motivation at the top. ⌘ Then the things started to change…
We've entered the race: version 1, 2, 3, … 7, 8,… 13, 14, …

Perfect Strangers

- Orthodox top to bottom

Each developer reported straight to me and in a way did not see the entire project. All control was concentrated in one hands…

In a way that looked like …

Perfect Strangers

- We did collaborate but that was mostly on the purely technical issues.
- In a way no one knew what his colleagues are working on.
- And everyone was so busy…

Victims of MS Project

- On Microsoft Project for many years. ⌘
- The problem was that any new high priority task destroyed the initial version. Diseases, days off, delays made the planning hopeless.
- Automatic scheduling could not count particular person skills, area of his best expertise. At the end there were always manual adjustments… and that kept going on, over and over again.
- At some moment Microsoft added the MS Project Server and new pricing scheme. After that a good MS Project installation became way too expensive.

Victims of MS Project

We’ve moved to MS Project 365, a light cloud based version of the “big” MS Project. Here the disaster came… ⌘

And what is the most important that planning did not make much sense. Due to constant changes, after 3 months, the plan had very little to do with the initial one.

At the end our plan has become just a long backlog of our tasks, and the project management “downgraded” to manual appointment of a person to a particular task based on his/her experience and a common sense. ⌘

Perfect Strangers II

⌘ Would you be able to compare performance of 10 tennis players placed on 10 separated tennis courts to play against a wall? ⌘

It was exactly our case. Usually one person worked on one task. As all tasks were different your estimation of the developer performance should be very subjective.

In a situation without real collaboration and healthy competition a manager has a good chance to find him in a situation like that:

.

Perfect Strangers II

It actually affects many things: task deployment, performance assessment, bonus calculations, sometimes even personal relationships…

Sometimes it looked like as the tail wags the dog.

By the beginning of the year 2016 we became 12 APL developers plus other staff, number one in the portfolio management software in Russia owning some visible part of the Russian financial software market.

We realized a need for growth…

But the things started to go out of control…

We may not continue to work as before.

Face The Reality

It had just happened. Suddenly, one day, late spring of 2016, I'd realized that the fundamental problem was …

me (!)

Maybe not me personally, but the structure of the development and my role in the structure, which I was very much responsible for creating.

⌘ That all must be changed and the sooner the better!

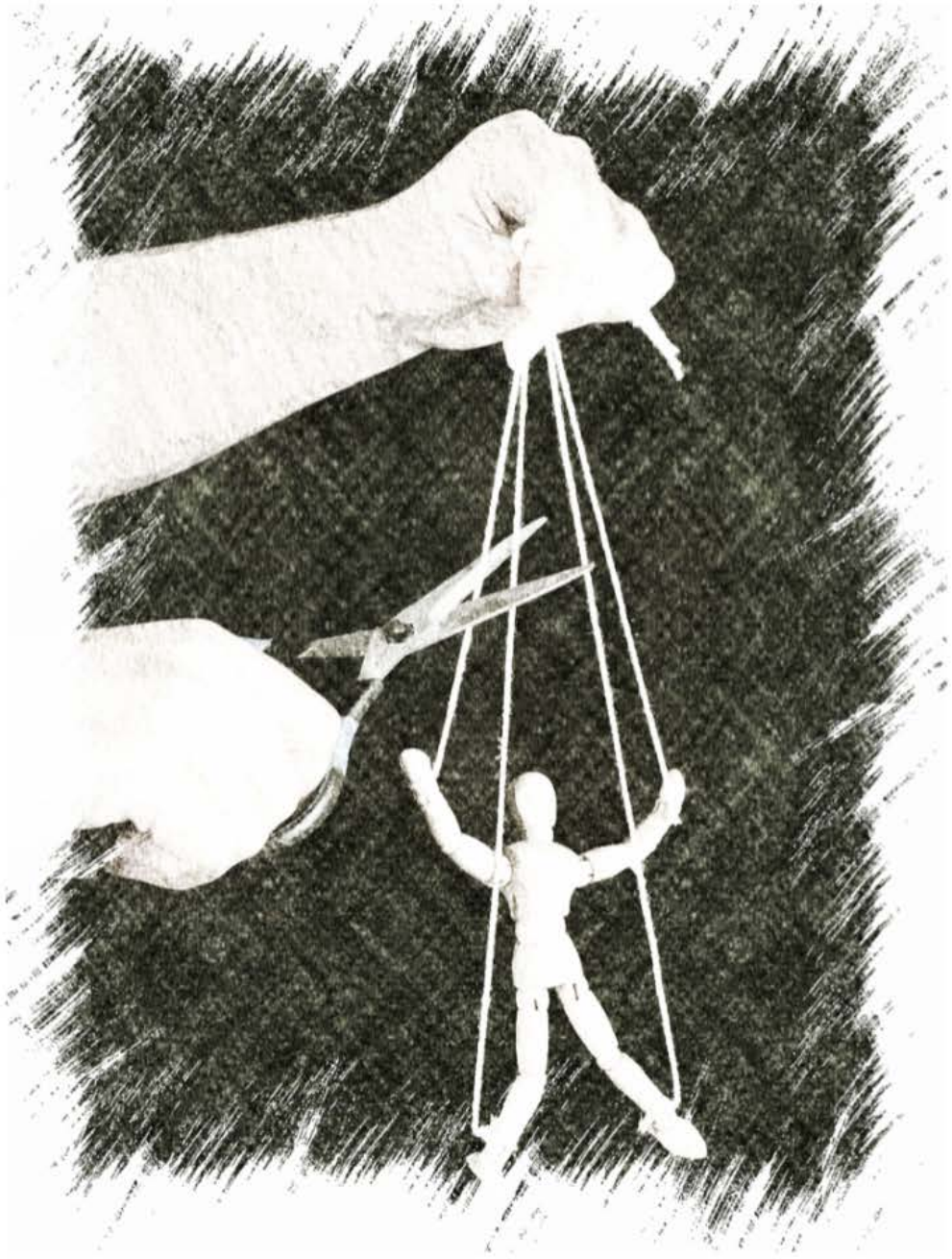
The change

Though it was not very clear
what the change should look like...

Cutting Ropes

The first move was the most
challenging – "…let you people go…".

The orthodox top to bottom
management does not sit well with a
free initiative. ⌘



A Chain of Synchronicities. Project Agile.

It started with a book Scrum written by Jeff Sutherland… ⌘

We've never tried to implement a classic scram stuff as is. We simply borrowed everything we found applicable and useful.

The centralized planning and management has been buried. What was "the plan" before has become the scrum backlog.

Today every release cycle, the sprit in the scrum terminology, starts with developers planning meeting. They select tasks to be included into the sprint. The tasks are picked from the top of the backlog. I don't participate in this kind of meetings.

Project Agile

Business meeting before each developers sprint meeting. I represent the product owners being in charge of the overall value of the project, our marketing director represents the market, and our head of the Support departments represent our customers.

Our most experienced developer has become the sprint master. ⌘ Various short meetings weekly… ⌘

We switched to 1-1.5 month releases, and a major version once a year. So we usually have 4 weeks of development and 1 week of tests.

Project Agile

Later it was updated to 1st sprint planning meeting, 1 week of working with specs and implementation ideas, 2nd sprint re-planning meeting, 3 weeks of development, 1 week of tests.

Finishing a task a developer “leaves a trace” – a chapter in the technical documentation and a chapter in the user manual.

Our tests split to 2 parts. The first is performed by developers themselves provided that the author never tests his own code. The second part of test is performed by the Support department. Their goal is to check if the developed stuff really solves the problem it intended to. They also check docs. ⌘

Project Agile

What we've actually done is we broke down our endless project into multiple small projects with quite definable goals and clear deadlines.
That was a big change...

Project JIRA

- Why JIRA is good for the never ending projects? Initially it was a bug tracker! Bugs fixing is the perfect never-ending project!

Atlassian has another product – the JIRA Service Desk. The support is indeed the ultimate never-ending project!

Both JIRA Software and Service Desk are integrated. Third major product - Confluence – a wiki , which is also well integrated into the family.

Implementation of the new software (Software+Service Desk+Wiki) took us 1 month including migration of the MS Project to JIRA.

Planning

In our business, the time estimation is possible when the job is 80% complete.

In our case, the time estimation for a task was made like that:
(1) how long would it take for a reference developer to complete the task, roughly;

(2) what is the performance difference between the reference developer and the one in question (from 2 to 10 times!);

(3) multiply – got the estimate!

“The plan” has been replaced by the monthly backlog revision meetings.

Traps: Roles Hell.

After years of working with the same small team on the same projects every person got a “preferable specialization”.

One person always rides the donkey, another one always looks after the water pump, another one always…

To build a perfect scram team either you should, in a way, sacrifice the personal skills for increase the team level or you doomed to switch back to a “manual control”.

Traps: Specification Hell

No people dedicated to work on specs.

It is the developer himself who should understand the problem, suggest an approach to implementation, provide the complexity and time estimation.

That means that when the team meet to make a sprint no one really knows what are those tasks in the backlog are about – the backlog has been reshuffled a day before!

We appointed a person who's job was to collect all related information and help with the task understanding… Did not work! ⌘

Traps

Tradition Hell

In any stress situation the team tends to fall into the mode they used to.

Ambitions Hell

Relatively short release cycle and a rise of productivity made possible some things not imaginable before. We decided to start sending our customers an announcement with what is coming in the next release and when… Don’t do that !!!

Problems

No complexity/time estimation

It is the biggest one. So far we have not been able to find a reasonable method to measure complexity and duration of a task. Without that any reward upon effort is difficult to achieve.

Problems

Client Support Department Crisis

One version/release per year scheme assumed that quite a few patches ordered by customers as well as and bug fixes on the course of a year. That means it was constant flaw of software updates and tens variations of the same product…

Switching to 1-1.5 months we releases we decided to kill that chaotic procedure.
1 month term looked as short enough for almost any customer to wait for almost any patch (but the critical bug fixes, of course).

Problems

We completely missed the point that while any single customer had had fewer updates during a year he had his own patch much faster than 1-1.5 month. We effectively broke a traditional problem solving service cycle of our QA…

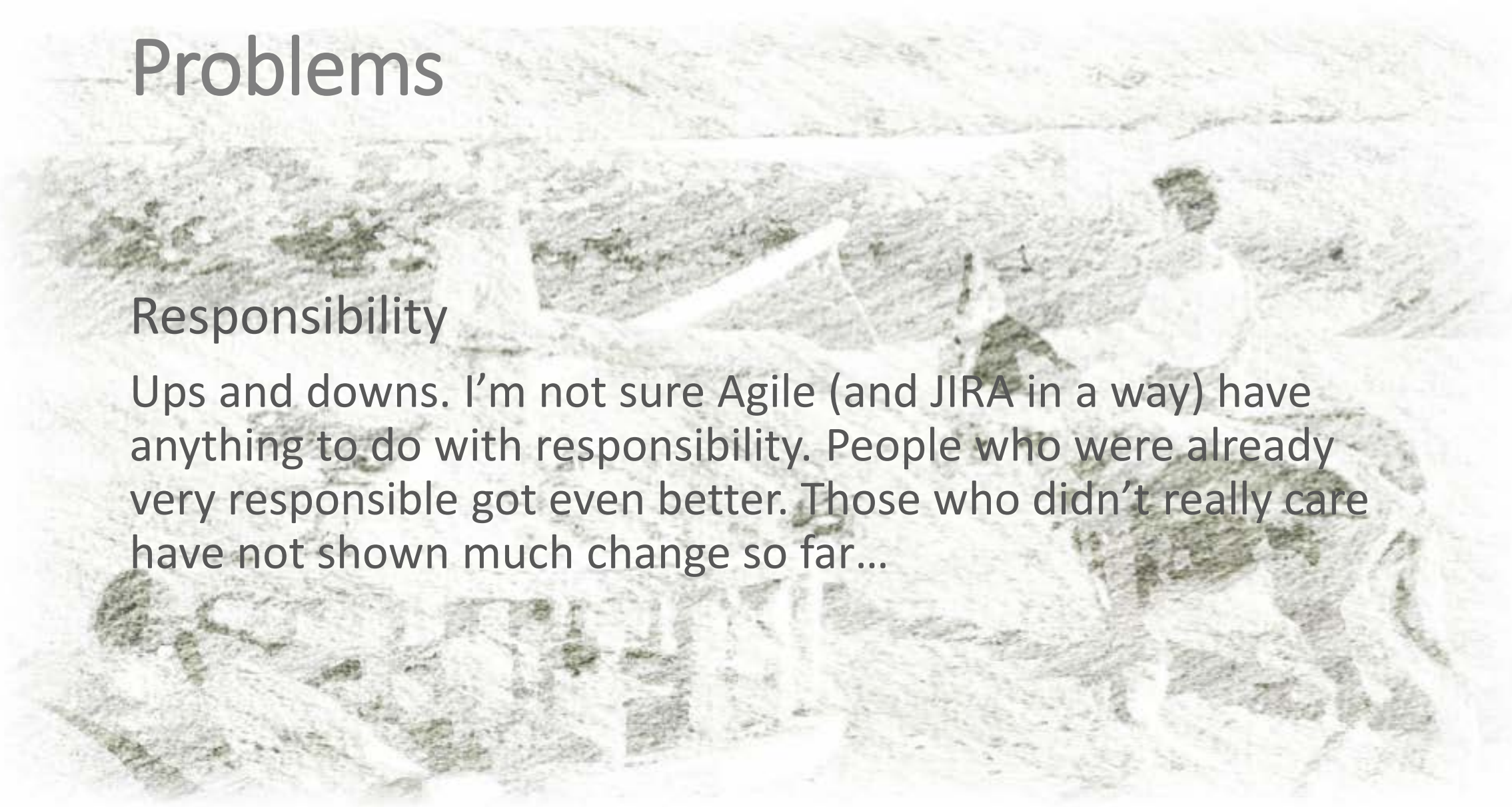
So we got an opposition in our Support. An effect of that was very depressing – not only they became too slow, softly speaking, adjusting to the new situation, they also in a way “sabotaged” implementation of the JIRA Service Desk software, wrongly assuming that it would make their live even harder.

Problems

The situation was finally resolved when we parted with the head of the Support team, and I de facto had to become a crisis administrator for it. It was a time for a change there, too! It started a month ago and now we have a much more positive team, the JIRA Service Desks running full scale, slightly relaxed customers, measurable statistics of the team activity… that's another story.

BTW I have not been able to find any reasonable publications on the "Agile products" maintenance and support so far.

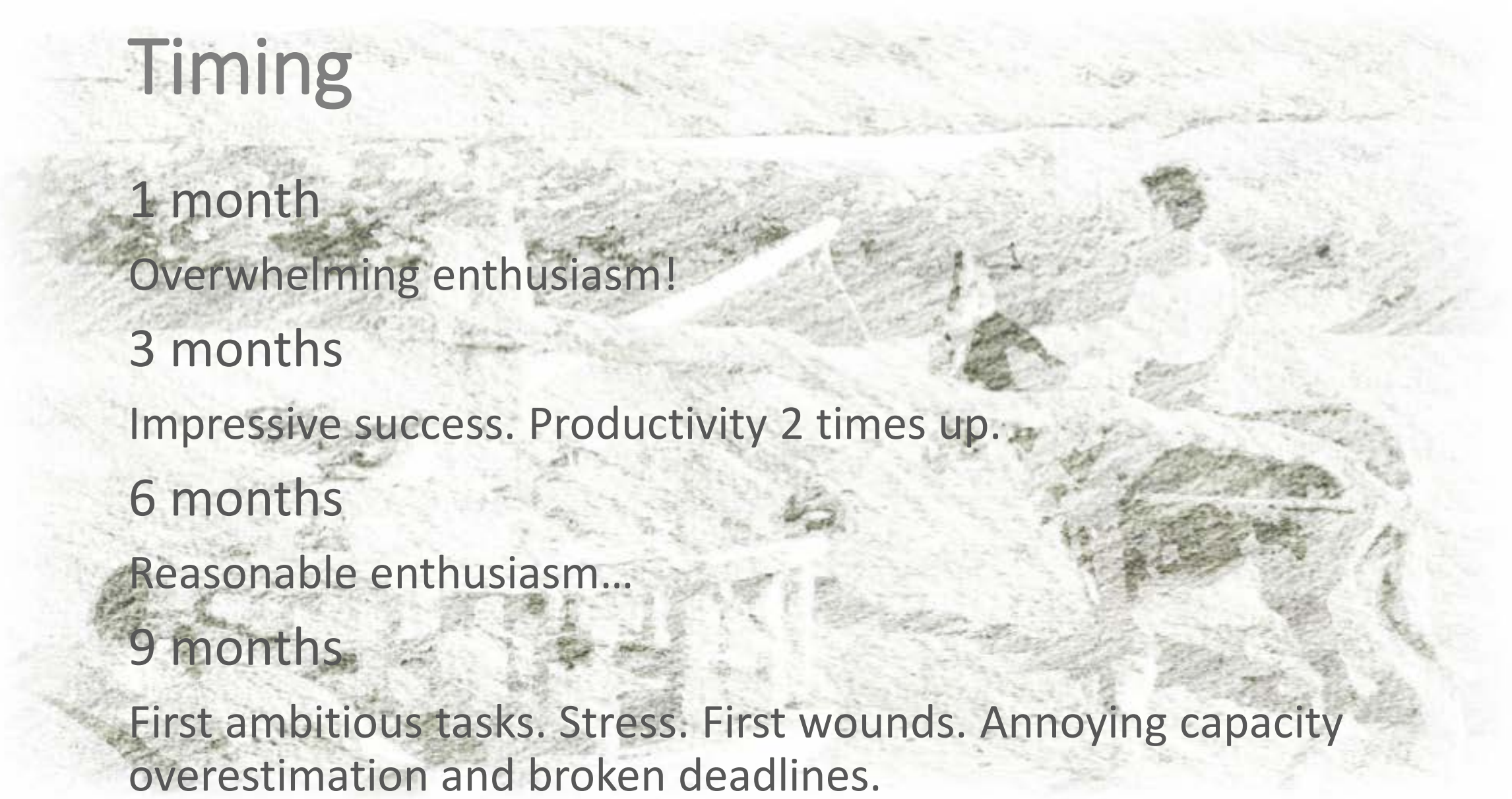
Problems



Responsibility

Ups and downs. I’m not sure Agile (and JIRA in a way) have anything to do with responsibility. People who were already very responsible got even better. Those who didn’t really care have not shown much change so far…

Timing



1 month

Overwhelming enthusiasm!

3 months

Impressive success. Productivity 2 times up.

6 months

Reasonable enthusiasm…

9 months

First ambitious tasks. Stress. First wounds. Annoying capacity overestimation and broken deadlines.

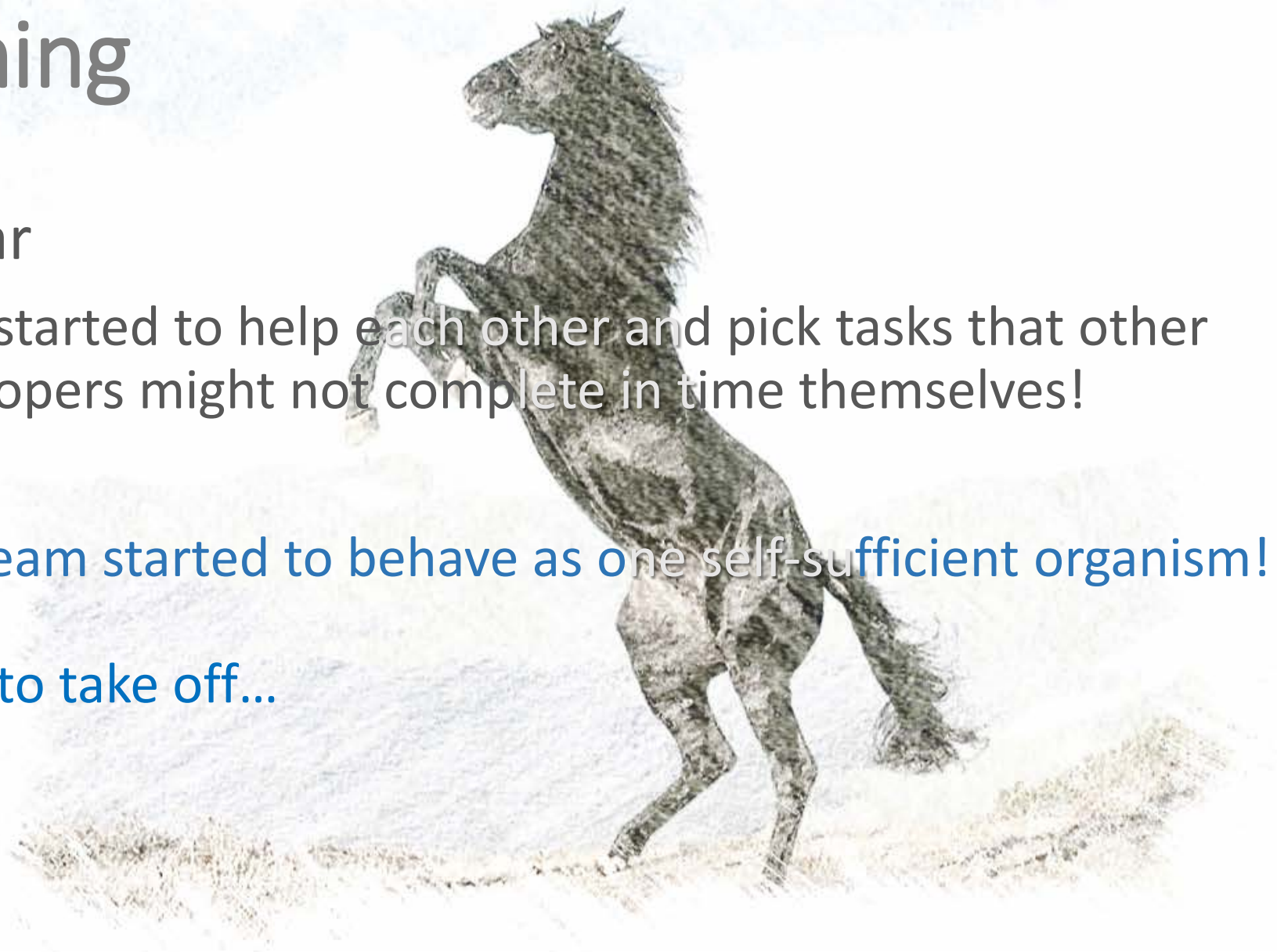
Timing

1 year

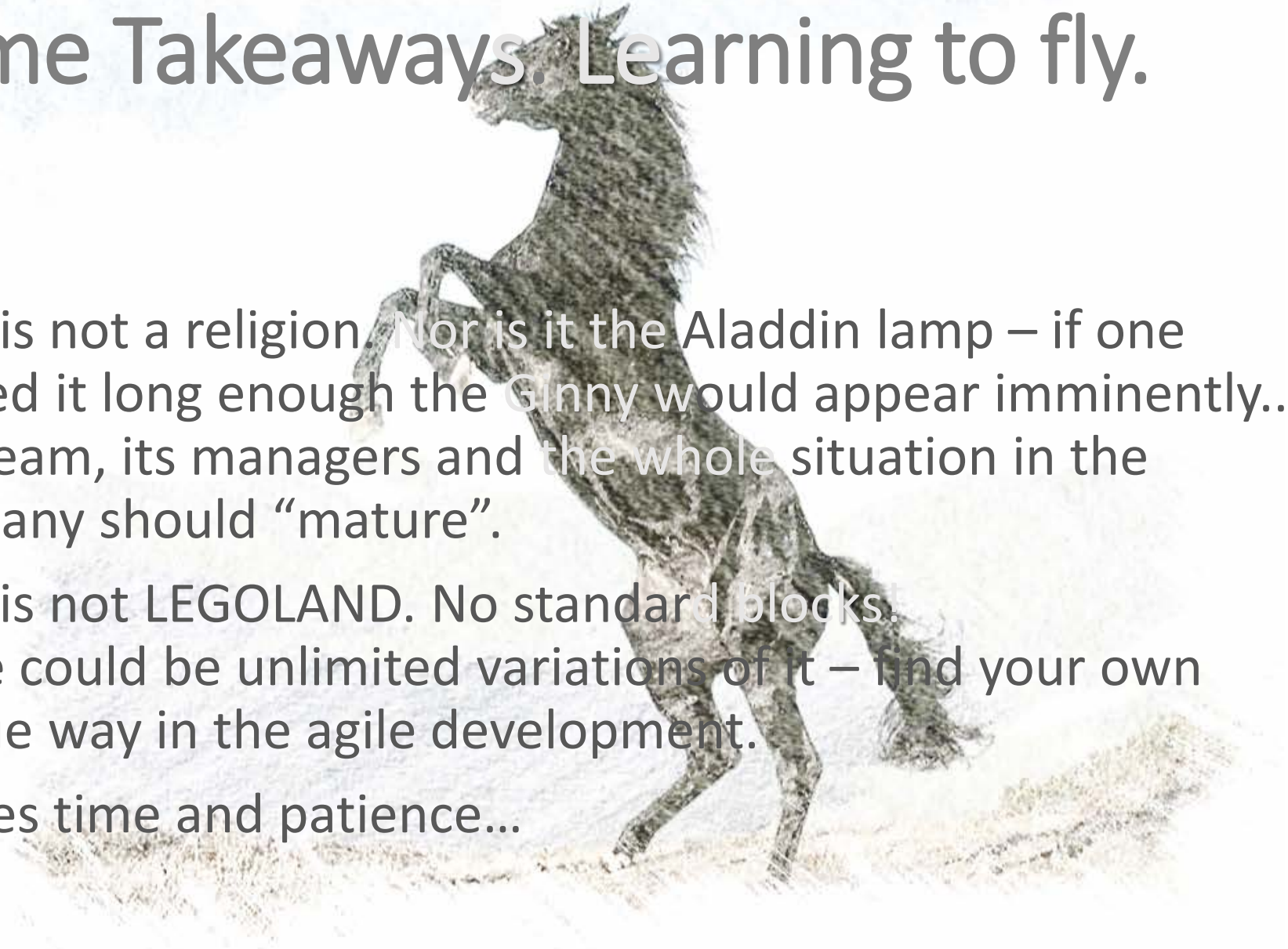
They started to help each other and pick tasks that other developers might not complete in time themselves!

The team started to behave as one self-sufficient organism!

Time to take off…



Some Takeaways. Learning to fly.



Agile is not a religion. Nor is it the Aladdin lamp – if one rubbed it long enough the Ginny would appear imminently... The team, its managers and the whole situation in the company should “mature”.

Agile is not LEGOLAND. No standard blocks. There could be unlimited variations of it – find your own unique way in the agile development.

It takes time and patience…

Thank you