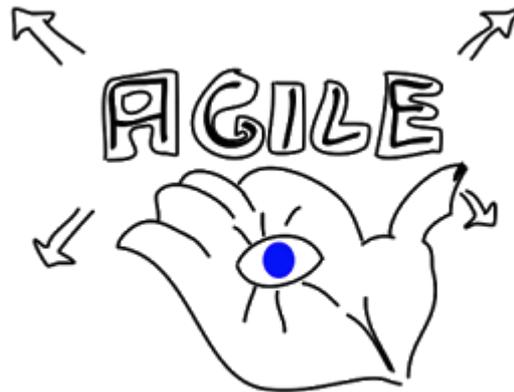


Migrating legacy applications – 7 points for doing it the agile way (and avoid BIG BANG, intransparent approaches)

 ontheagilepath.net/2015/11/migrating-legacy-applications-7-points-for-doing-it-the-agile-way-and-avoid-big-bang-intransparent-approaches.html

By Sebastian Radics



After spending some time rebuilding the main page for OnTheAgilePath (your feedback is highly appreciated) this weeks update is a short **learning recap regarding migration projects**.

I just share some insights that might help you in your current project or upcoming ones where you aim to migrate a legacy application to a new technology stack.

Special thanks to [Johannes](#) for collecting and sharing your insights and raising transparency and iterative thinking to the next level.

#1 Separate migration and new business functionality

Avoid a mixing of building new business functionality (new customer wishes) and the migration. As soon as you start changing existing business functionality or adding new business functionality on top you increase complexity for your migration (it is not longer just a migration under your control but you're adding new dependencies).

Very likely you have to check and re-test far more than it would be necessary when “just” doing the migration step.

The amount of people and stakeholders to talk too and systems to understand (that either use your application or provide input for your system) **increase complexity**.

By mixing both approaches one shifts the project nature. As customers can just clearly express what they really want, when they see and can play with the feature, you need a fast and highly iterative approach.

Although executing the migration in an iterative manner should be the way to be used in this kind of project it is very likely still a big difference to building a product.

#2 Focus on doing the migration

Aim for having a focus on the migration and **avoid working in parallel on (too many) other topics**.

We already know the terrible side effect of multitasking but still migration topics tend to be not priority one or being worked on in an extra lane. **WHY?**

By using the unfocussed, parallel way you can be sure that drive and energy for migration will be consumed by the ever ongoing always most important daily business tasks.

Commit to the importance of the migration and work solely on it. This way you create the necessary sense of urgency, provide transparency and everyone knows that it is important and has to be done ... now.

#3 Separate business feature migration from non functional requirements

Try to **slice the migration into smallest possible steps** with a live delivery after every step. Avoid big bang at all costs. It will not work this way.

A good approach is to tackle a **small business requirement migration first**. Learn from the system, build your new environment including delivery pipeline and new production environment. And **ship it**.

It's a great feeling, supports motivation and builds trust, when migration steps are really getting used.

As a next step, check for scaling opportunities based on the business feature and scale the system properly. E.g. we learned that after migrating one business feature affecting one customer (focus for iteration one), we found many more customers that needed the same feature too.

This way we are able to tune the system already in iteration two and therewith address scaling risks. Is our new architecture approach able to serve the amount of requests?

What do we have to consider when scaling?

Good to learn it early on and validate that our chosen migration approach is going to work.

#4 Production live deployment as early as possible

Like mentioned in the previous point – aim for a **rapid first live deployment of a migration step** (and continue with the following steps).

Even if it takes (much) more effort – ship it and learn from the system. This way you can **validate early and apply course correction if necessary**. (On paper you're fast with planning the big bang – but as soon as you move to your real live environment, that big bang plane will crash.)

Planning for a sliced migration approach has to be integral part of your migration strategy. **NO BIG BANG** – I'm quite sure that rethinking your approach will uncover ways to slice it so smaller and shippable units. Using a solution focused mindset towards slicing and working with experts in this area is important.

#5 Small deliverable migration units

I guess small deliverables naturally follow by aiming for early live deployments – think in small deliverable migration units. **Create learning opportunities** (include mistakes and corrections) as early as possible.

This way lightweight course corrections can be done in time and don't accumulate to a mountain of code fragments no one really has under control (like it gets created when building a migration unit 1/2 a year++ and trying to ship it afterwards).

#6 Define migration goals and simple migration step goals

Everyone working in the migration project (or being affected by it) needs to understand the purpose of it. Some guiding questions:

- What are we replacing?
- Why is it necessary?
- Why now?
- What is the approach for the migration?
- What is our next step? How can we check that we're done with that step?

By providing the migration step goal everyone can check what is really necessary to achieve it. Gold plating and not (yet) necessary parts can be highlighted and either moved to a next step or discarded.

If in doubt ask: **Is it really necessary NOW to achieve the goal of this migration step?**

#7 Transparency, transparency, transparency

Work with your customers and explain that you work on an important technology migration. This way they are prepared and maybe even offer help.

Explain the migration approach to everyone involved and start using the combined solution focus and groups intelligence to solve the important slicing topic.

Visualize your goals, provided appropriate measurements and progress check ins (real usable ones and not just fake progress reports in shiny power point presentations 😊)

There will be mistakes – for sure. Welcome them as learning opportunities, adjust your next steps and build a flexible and agile environment.

All part of thinking and working the agile way, sure! But still often forgotten when working in legacy migration projects.

I hope you got inspired? What will you use? Do you have more recommendations? Thx for sharing your insights via your comment 😊