WHITEPAPER

Smart Software Modernisation: Choices, Approach and Business Case

TABLE OF CONTENTS

INTRODUCTION

Standing still is the fastest way to move backwards. Your software may work fine now, but the pace of the market is accelerating and needs are constantly changing – both in terms of functionality as well as user experience. And so there will come a day when your existing technology platform can no longer keep up.

Then it's time to make some hard choices: Should you start over or try to modernise your existing software?

In this whitepaper, we take you step by step through the entire process of modernising your software.

1. Towards more maintainable software

The demands that users place on software are completely different from 5 to 10 years ago. Today's organisation works with cloud infrastructure, integrates data and systems from dozens of sources and from different vendors and works in increasingly shorter innovation cycles.

If you developed your software years ago, you may have a package that can't keep up with new developments. Maybe your system isn't cloud ready and you can't offer it as SaaS? Perhaps it's unprepared for multi-tenancy and is difficult to integrate with commonly used solutions in your market?

Or are you lagging behind in terms of user experience and performance, now that users have been spoiled by the smooth-running mobile apps of big tech companies?

These are challenges that many software vendors are currently facing. Added to this is the increased pressure on the privacy and security aspects of software - developments which are now also difficult to keep up with on older platforms.

In the meantime, sales may seem to be going quite well, because the market has a huge need for good, specialist software. Sadly, due to technical challenges, subscriptions don't start until months after the sale or, because of the long waiting list, customers drop out. This results in a substantial loss of turnover.

2. Start with thorough research

You know where you want to go with your software and your business, but do you actually know where you are now? In order to make choices about rebuilding or modernising your software, you need a detailed picture of the status of your software now.

> **“SO, START WITH SOME THOROUGH RESEARCH. WE CALL SUCH AN INVESTIGATION A TECHNICAL DUE DILIGENCE (TDD).”**

Here are some of the most important points that we look at in such an investigation:

SYSTEM LANDSCAPE

We look at your system landscape as a whole and evaluate its scalability, performance and security. We also examine how the software package is integrated with the rest of your IT. Any dependencies on external suppliers and vendors are also included in our analysis.

CODE QUALITY

Code quality is very important. The better the code is, the better we can see exactly what it does and the easier it is to reuse. Poor code quality is a key reason to build (part of) an application from scratch.

MAINTAINABILITY

Not only the quality, but also the future-proofing of the code is important. Programming languages evolve and old ones sometimes need to be updated to run in a modern environment.

DATA QUALITY

We also look at the data in your database. Contaminated data has to be cleaned up, whether you choose to rebuild or to modernise.

DATABASE STRUCTURE

A database with a good structure gives you an important advantage, because if we can leave the database intact, we avoid the time-consuming and error-prone data migration process. It's possible we can then, partially or incrementally, modernise the application.

COMPONENTS AND LINKS

Most systems start out as an orderly collection of clearly defined components. But, over the years, the cracks start to appear as development teams are sometimes forced to take shortcuts to meet user requirements. Often, such a landscape starts to look like a plate of spaghetti, with hardly any structure left. We review all the components, map their functions and determine which, if any, are suitable for reuse. In this way we untangle the spaghetti and get a clear overview.

TECHNOLOGICAL TRENDS

We also compare your existing software with general developments in the software world. This allows us to see whether existing technology can still meet market needs. But we also look at whether there are now better ways to realise your functionality and what opportunities new technologies have to offer to make your software faster, more scalable, more maintainable and more user-friendly.

3. New build or smart modernisation?

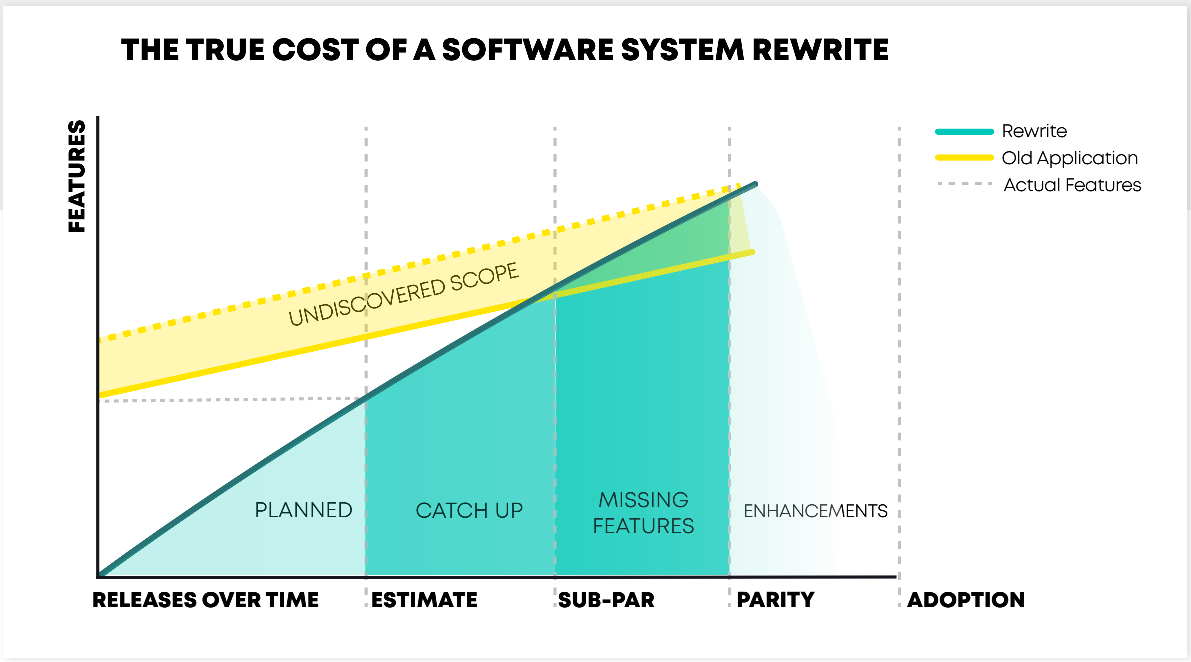
With the insights gained, you can arrive at an honest analysis of the condition and reusability of your software. Based on that analysis, you can choose to either renovate, rebuild or a combination of both. Not every piece of software is suitable for renovation. Is the technology too far behind and the quality too low? Then it is probably better to start from scratch.

Think about this carefully, because a complete rebuild is an expensive project, which is also very difficult to budget for. We see rebuilding projects that become three times more expensive than planned.

The main reasons for this are:

> While the project is running, you also have to continue development on your old platform. That means you need double the number of developers during your rebuild project.

> Your software contains 'hidden functionality', which you didn't include in the scope of your original plan. But many of those features are still mission critical and do need to be included in your new software.



But a rebuild also has important advantages over modernisation:

> You can choose a different scope. For example, maybe your application supports manual operations that can now be done by an algorithm.

> You can develop faster. An existing codebase full of legacy code is always a slow-down factor. If you start with a 'blank piece of paper', you can move forward quickly with high code quality.

Ultimately, you make your choice by applying the '7R model' for each component:

1. RETIRE

Has a component become redundant and can it therefore be removed?

2. REPLACE

Can we replace a component with third party software?

3. RELOCATE

Is it enough to move the existing code to the cloud?

4. RE-PLATFORM

Can we modify the existing code to take advantage of modern capabilities, like those of Microsoft Azure?

5. REBUILD

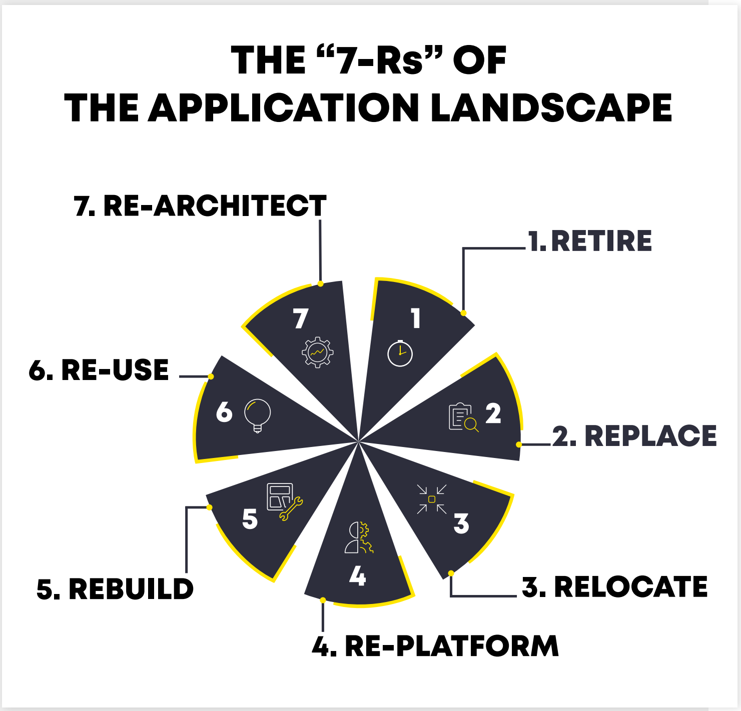
Or do we have to start from scratch?

6. RE-USE

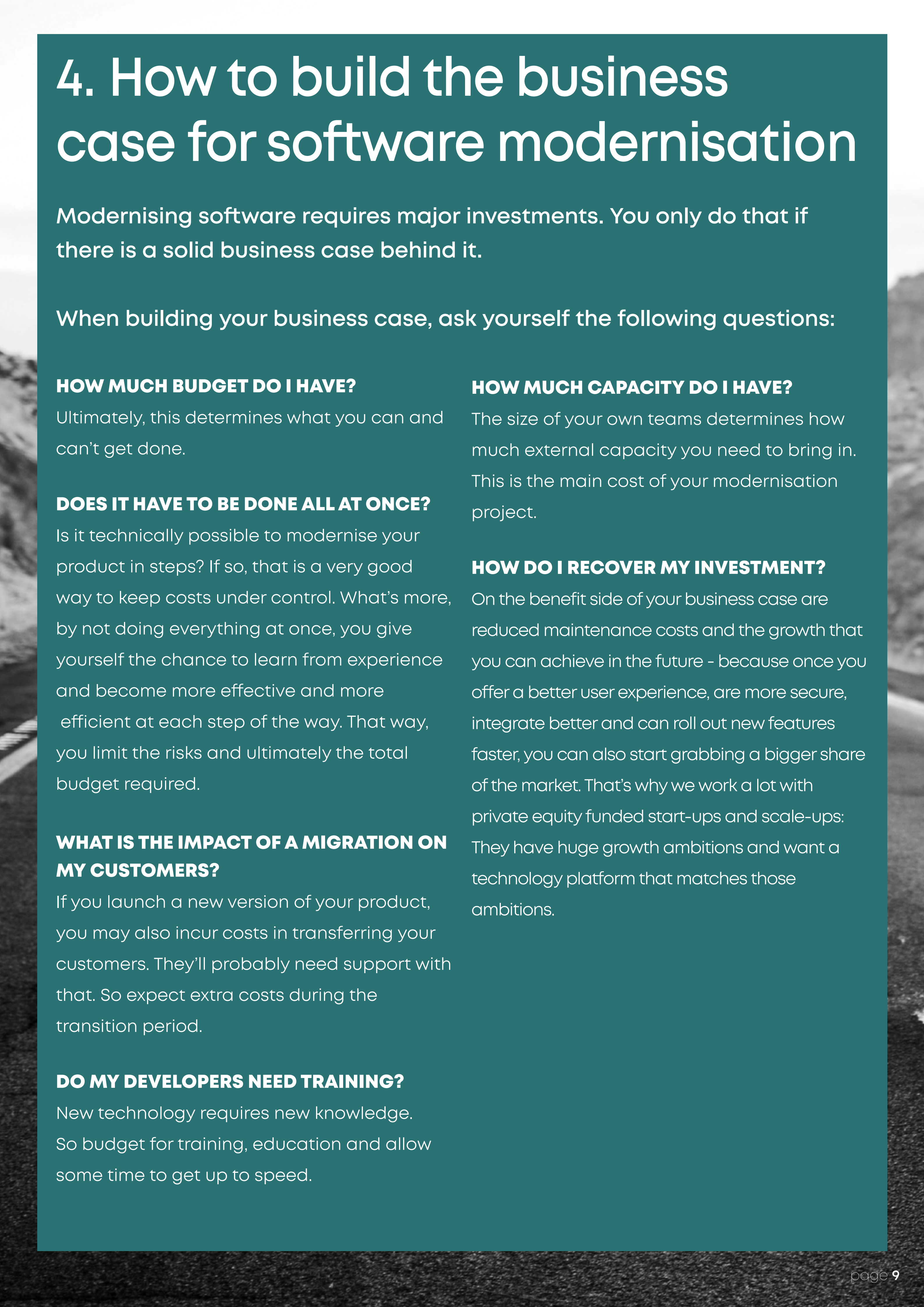
A rare scenario in which a component can be used in the new situation without modifications.

7. RE-ARCHITECT

Is the component actually still a component in the new situation? Or should we split it up and re-implement the parts separately?



The answers to all these questions will give you a technical roadmap.
Once you have that clear, it's time to start looking at your business case.



4. How to build the business case for software modernisation

Modernising software requires major investments. You only do that if there is a solid business case behind it.

When building your business case, ask yourself the following questions:

HOW MUCH BUDGET DO I HAVE?
Ultimately, this determines what you can and can't get done.

DOES IT HAVE TO BE DONE ALL AT ONCE?
Is it technically possible to modernise your product in steps? If so, that is a very good way to keep costs under control. What's more, by not doing everything at once, you give yourself the chance to learn from experience and become more effective and more efficient at each step of the way. That way, you limit the risks and ultimately the total budget required.

WHAT IS THE IMPACT OF A MIGRATION ON MY CUSTOMERS?
If you launch a new version of your product, you may also incur costs in transferring your customers. They'll probably need support with that. So expect extra costs during the transition period.

DO MY DEVELOPERS NEED TRAINING?
New technology requires new knowledge. So budget for training, education and allow some time to get up to speed.

HOW MUCH CAPACITY DO I HAVE?
The size of your own teams determines how much external capacity you need to bring in. This is the main cost of your modernisation project.

HOW DO I RECOVER MY INVESTMENT?
On the benefit side of your business case are reduced maintenance costs and the growth that you can achieve in the future - because once you offer a better user experience, are more secure, integrate better and can roll out new features faster, you can also start grabbing a bigger share of the market. That's why we work a lot with private equity funded start-ups and scale-ups: They have huge growth ambitions and want a technology platform that matches those ambitions.

5. UX Blueprint

Improving the user experience is an important factor in software modernisation.

So before we start such a project, we always make a 'UX blueprint'. In it we plot the course for the experience we want to offer the user with the new software.

We aim for a look that not only perfectly supports the user in their daily work, but that's also in tune with the times. This will make your product stand out from the competition. We base the UX blueprint on user data, an evaluation of your current software and interviews with users.

The result of this process is a complete overview of the flows of your new software and a clickable prototype that shows all stakeholders what the future looks like - all before a single line of code has been written.

6. Monitoring, maintenance and further development

Development does not stop after the modernisation of your software. The entire project must then focus on monitoring, maintaining and further developing the new application.

So it's important to pay attention to the following points from day 1:

Intensive cooperation and knowledge transfer.
Work in multidisciplinary teams that include your people and those of your partners. This ensures a seamless transfer to your development department and your own developers gain the necessary knowledge to continue working independently on your application.

Data Collection.
You want to continuously optimise your application, so you need to make sure that every component is monitored. Performance and usability are particularly important variables when optimising your user experience, but security is also largely dependent on data.

Maintainability.
The whole idea of your modernisation project is that it enables you to move forward faster and with lower maintenance costs. That should be an important motivation for every technical decision you make. Clean code, good architecture and the implementation of clear working agreements and processes are essential parts of the process.

7. Partner selection is crucial

When it comes to your product, you are the specialist. But you probably have less knowledge within your company about software research, redevelopment, modernisation and the state of the technology market in general. To successfully modernise your software, you need to bring that knowledge in from outside.

Here at Blis Digital, we are busy modernising or rebuilding business-critical applications every day. Maybe it's the first time for you. So, together we are stronger. We know the modernisation process and you know all about your product and your market. Another very practical factor is capacity. As we saw in the previous chapter, software modernisation temporarily requires double development capacity. There are few organisations that can just 'crack open a can of developers' for this, especially in the current labour market. For that reason alone, you need outside help.

But modernising your software is not just another development project. It is the project on which the future of your product, and perhaps your company, depends.

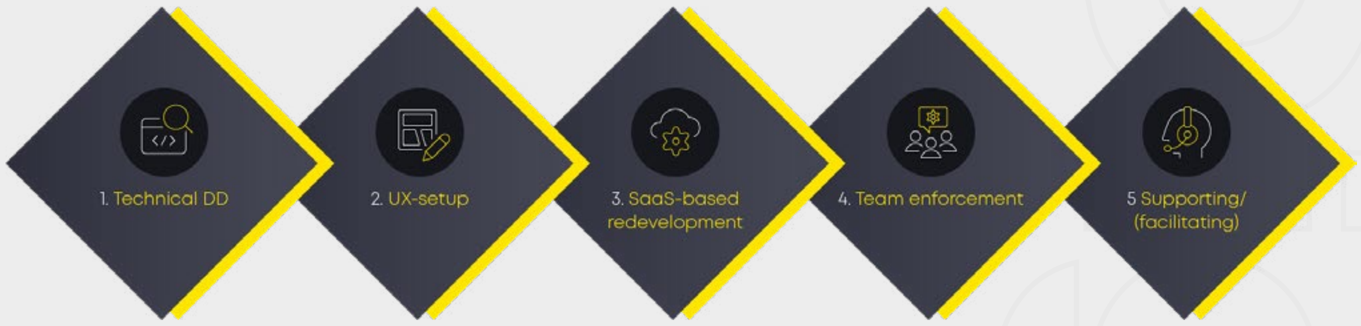
So you don't want to work with just any partner. It is not only our knowledge and experience that enables us to bring these types of projects to a successful conclusion, but also the clear processes we use, ensuring everyone involved always knows which steps we are taking and why. We also help you get all stakeholders around the table and create support for this radical change.

> **"THERE ARE FEW ORGANISATIONS THAT JUST 'CRACK OPEN A CAN OF DEVELOPERS' FOR THIS, ESPECIALLY IN THE CURRENT LABOUR MARKET."**

8. The Blis Digital NextGen Programme

When creating mission-critical and business-critical software with a long-term perspective, nothing can be left to chance.

That's why at Blis Digital we have designed all the necessary processes in the Blis Digital NextGen Programme. In it, our years of experience and global best practices come together in a proven framework to make your software product SaaS-ready and future-proof.



WANT TO KNOW MORE?

Do you want to accelerate modernisation and get more revenue from the market with better software? We can help you.

Contact ⟶

Or read more about the NextGen programme here

About Blis Digital

Transformation, renewal, fundamental change, digitalisation: they sometimes feel like magic formulas, concepts that are difficult to apply to your concrete situation. But if you have the right knowledge, technology, processes and people, the fog disappears and you see clearly which steps to take.

Modernising your software shouldn't be a leap of faith - because you don't need more risk, you need less. We always work in clear, safe steps. We take responsibility for the technology on which your entire business runs. That requires trust from you. We earn that trust by committing to long-term partnerships and mutual success. We always choose the relationship over margin or quick turnover. Your success is our success.

So, let's get to work.