devoteam

2020

Software **Modernization**

7 considerations before starting

Creative tech for Better Change

TABLE OF CONTENTS

INTRODUCTION

WHY SOFTWARE MODERNIZATION?

Complexity and increasing needs.

As the world changes, businesses need to change with it. So do their IT landscapes, and as they do, deadlines, new regulations, and business demands take their toll.

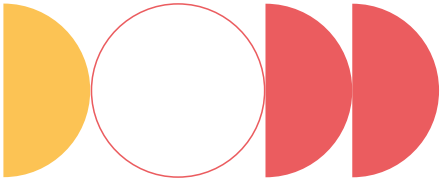
The constant stress of this change will strain an IT landscape, as it seems there is never enough time, money, or people available to tackle an ever growing amount of technical debt.

Critical problems might be patched over, increasing complexity and reducing flexibility. The time it takes engineers to tackle problems increases, and implementing new features becomes time- consuming and increasingly expensive. Companies start to feel constrained by their IT systems.

Instead of enabling, IT is holding business back, preventing those companies from moving forward.

Once in such a situation, getting out of it can be a difficult, and often costly task. It may be tempting to scrap existing software and start from scratch. However, the results of this approach may vary, with cost overruns and missed deadlines frequently cited.

Software modernization is one way of dealing with legacy IT issues. It is defined as the changing, porting, rewriting or replacing of legacy software to modern standards. Before starting on a software modernization journey, we have compiled a list of considerations that can help guide you on the right course.



CONSIDERATION 1

START WITH WHY

Software modernization is not a goal, rather it is a way of achieving a certain goal. The first step of any software modernization programme then, is finding out why you are considering it. What are the reasons software modernization sounds like an appealing proposition for a company?

A clear view of the reasons for starting this process can help in determining what actions to take, and how to measure success.

We have identified three broad categories of business problems; technical limitations, operational challenges and digitization challenges.

3 CATEGORIES OF BUSINESS PROBLEMS

TECHNICAL LIMITATIONS

Description: IT systems are preventing the company from moving forward. Modernizing business processes, entering new markets, developing new products, these activities are hindered by IT rather than empowered. Time-to-market for new products and services is too high, because of IT. Scaling business is troublesome.

Example question(s): Do IT systems have a negative effect on TTM?

Solution focus: Solutions for technical limitations need to focus on enabling new business opportunities. For short-term goals, often upgrading existing systems can relieve some of the pain experienced.

OPERATIONAL CHALLENGES

Description: Keeping the company going "as-is" is becoming more difficult, and often more costly. The maintenance of existing systems is getting too time-consuming. Old hardware or software stacks might lose vendor support, or qualified personnel cannot be found.

Example question(s): How much of their time is IT spending on keeping things running? And how much on developing new capabilities?

Solution focus: Solutions for operational challenges are mostly focused on delivering the same level of IT services with less effort or cost. Out-of-support hardware and software will often need to be replaced.Reducing complexity and reducing maintenance costs are important.

DIGITALIZATION CHALLENGES

Description: The digitalization and automation of for example SCM processes cannot be achieved with current systems. Integration with stakeholders is troublesome.

Example question(s): How integrated is your IT landscape with those of your stakeholders?

Solution focus: Solutions for Digitalization challenges will often focus on integration and automation.

CONSIDERATION 2

NO PRESCRIPTION WITHOUT A DIAGNOSIS

An answer to the question of why to start a software modernization journey, can be seen as an indication of the symptoms of outdated IT. The next step is to find out what the underlying causes are; to diagnose the problem.

FOR THIS DIAGNOSIS, TWO QUESTIONS NEED TO BE ANSWERED.

1) What requirement gaps can we identify?

Some IT landscapes are not unlike a medieval cathedral. It took vast amounts of effort, material and money to complete these often complex structures. Over great amounts of time, additions might have been built, and subtle changes introduced. The church however, has been able to

fulfill its intended purpose for centuries, and, as long as this raison d'etre still holds, will continue to do so. It makes no sense to tear down the church in order to modernize it. However, by modern standards, the church might be cold and dark. So in order to meet new requirements, a central heating system is added, electric lights are installed, and a WiFi-network is set-up.

Other IT landscapes are more like houses. Most houses do not last as long as the cathedral might, and the owner's requirements for the house will change at a faster pace. A new kitchen or bathroom, solar

panels on the roof, an extension or a new conservatory, all changes that might profoundly affect the house. However, the foundations and load-bearing walls will probably remain the same. If the owners do not have requirements that are too different from the starting requirements, it does not make sense to tear down the house.

As with the cathedral and the house, in order to determine whether to replace or upgrade an IT landscape, and how to upgrade, depends on the gaps between current (and future) requirements, and those the system is able to fulfill. In our diagnosis, the first important step is finding out where those gaps are, and how big they are. In terms of our example, the church might be too cold. In terms of an application landscape, developing certain new products might be hindered. Where does the existing landscape not meet current or future business requirements?

2) What IT problems cause these gaps?

Once we have a clear view of what the gaps between the capabilities of our current IT landscape and the desired capabilities are, the second stage of our diagnosis is finding out what underlying technical issues are to blame. In this stage of the process, we look at the technical side of things.

What applications are in the landscape, how are they connected, how have business processes been implemented in IT systems.

For each problem identified in the first stage of the diagnosis, a root cause analysis will be performed on the IT landscape, with the goal of finding out exactly what technical issues underpin the problems felt by business. Is a certain application not-performing? Are there no(t enough) people to maintain and expand a system? Is automation of certain tasks insufficient? Do people need to work around inflexible systems?

CONSIDERATION 3

WASTE NOT, WANT NOT

One of the key tenets of Software Modernization is that the **value of legacy systems should not be discarded**, but should be retained and optimized. This is not to say that phasing-out of legacy systems should never happen, but rather that the opposite approach of throwing out everything and starting anew often does not deliver the best value.

While it seems enticing to just "rip-out-and-replace" an old system with something new and fresh, the hidden value of legacy systems is often forgotten. Over the lifetime of a legacy system, many larger and smaller changes will probably have been made to it, in order to reflect the business environment it operated in.

Specific knowledge of people who might long ago have left the company might be encoded in a bit of software, a macro or an IT process.

People know how the system works, and are accustomed to getting their job done with it, even if they have to use it outside of the intended use.

In complete replacement of systems, the effort and cost expenditures necessary to capture and integrate all these small bits of knowledge into the new system is unfortunately forgotten routinely.

For a successful modernization, the hidden value of legacy systems should not be discarded, but either re-used or remodeled. Sometimes keeping a legacy system that is not entirely optimal, can still be a sound decision when **the cost of capturing and integrating the knowledge within that system outweighs the cost of preserving it.**

If however, the decision is made to replace a legacy system, steps should be taken to avoid the trap of designing the new system against just the known requirements. Missing the requirements set by the knowledge embedded in the old system can lead to costly rework later on.

CONSIDERATION 4

DECISIONS, DECISIONS

There are multiple ways to modernize an IT system, ranging from encapsulating a system in API's to replacing it. Gartner has defined the 7 most-cited options as being the following.

1. **Encapsulate.** To leverage and extend an application's features and value, encapsulate data and functions in the application and make them available as services via an application programming interface (API). Implementation specifics and knowledge are hidden behind the interface.
2. **Rehost.** Redeploy an application component to another physical, virtual or cloud infrastructure without recompiling, altering the application code, or modifying features and functions.
3. **Replatform.** Migrate an application component to a new runtime platform. Make minimal changes to code to adapt to the new platform, but don't change the code structure or the features and functions it provides.
4. **Refactor.** Restructure and optimize existing code without changing its external behavior to remove technical debt and to improve the component's features and structure.
5. **Rearchitect.** Materially alter the application code so you can shift it to a new application architecture and fully exploit new and better capabilities of the application platform.
6. **Rebuild.** Rebuild or rewrite the application component from scratch while preserving its scope and specifications.
7. **Replace.** Eliminate the former application component altogether and replace it, taking new requirements and needs into account.

Source: Gartner

All these options have their respective up- and downsides and associated costs. As vendors of different solutions sometimes hail the strategy they have chosen as the "cure for all things", it might be tempting to select just one and go all-in on that. However, we believe that no single one of these strategies is the best approach, but that every case requires a different mix of these strategies. While the options for modernizing are quite well- documented, the question of selecting the best option in any given situation remains.

WE HAVE IDENTIFIED SEVERAL FACTORS THAT OFTEN INFLUENCE THE CHOICE OF MODERNIZATION SOLUTION.

Dependencies

a. Support
If a system is wholly or partially composed of software licensed from a particular vendor, support by that vendor can play a big part in deciding which modernization strategy to follow. When licenses can no longer be renewed, or support for a solution ends, the possibility of getting new features implemented in said product no longer exists. There is also an increased possibility of critical security vulnerabilities not being resolved. When no other supporting party can be found, replacement will often be the only long-term viable option.

b. Hardware
If a system requires certain hardware, there is a risk of the hardware going out of production. When this happens, and parts can no longer be sourced, rehosting and/or replatforming can be options to consider if the software is still meeting requirements. In this case, migrating to cloud infrastructure also deserves consideration. There is also the possibility of hardware limiting interoperability of the software running on it. In this case, the preferred solution really depends on the specific hardware limits.

c. Skilled engineers
The skills needed to run and maintain an IT system are often a reflection of the time that system was developed. Over time, the most popular IT skills change, and the pool of engineers capable of working with any technology may decrease. If it does, finding qualified personnel to maintain a system might become difficult and costly, with fewer people carrying an increased responsibility because "no-one else can fix it".

If a system is running well, and no changes are needed, a lack of qualified engineers is no problem. However, for a system that is often changed, or requires lots of attention, shifting to a more common technology stack can be a way to reduce maintenance cost and increase development agility. Rearchitecting or rebuilding might be suitable solutions in these cases.

d. External stakeholders

Interoperability

Interoperability of IT systems is becoming more and more important, as automatic integrations and sharing data with suppliers and customers become more common. A warehousing system not suitable for internet communication for example, might disrupt company plans to automatically restock with vendors if stock gets low. In these cases, where a system by itself might function in a satisfactory way, but its communication abilities are problematic, encapsulation might be a good solution. By, for example, creating a layer of web-accessible API's around the system, interoperability can be enhanced.

IT department capabilities

One often overlooked factor in determining the best approach to modernization, is the IT department itself. Does IT have the ability to work with a new system? Can they upgrade a legacy system themselves or is outside help needed? If IT has a thorough understanding of the current system, but no experience with a more modern replacement, does the cost of keeping a perhaps sub-optimal legacy system outweigh the cost of retraining IT and hiring new staff?

Upgradeability

Looking towards the future, if new requirements are likely to appear in the future, can the existing system be upgraded to meet them? In modernization, it is important not to look at only the present, but also to consider the future. If a system can be upgraded or expanded to meet current needs, but is then highly unlikely to meet anticipated future requirements, the cost of first upgrading the existing system and then replacing it can be higher than the cost of replacing immediately. While anticipating future requirements is often difficult, and we recommend not to consider unlikely or vague ideas to reduce feature creep, the inability of a system to meet future needs can make (partial) replacement a good option.

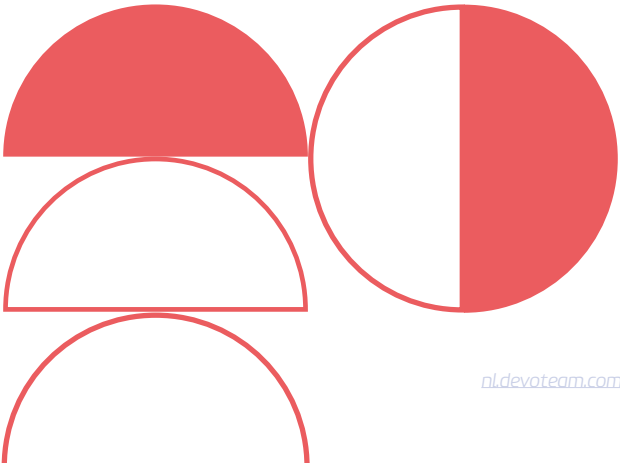
Time

Different modernization approaches require different time-paths. If meeting some requirements quickly is very important, modernization options that require the least amount of effort in the short term might be chosen. For instance, a combination of extending and refactoring in the short term to meet deadlines, combined with a rebuild on the long term to allow more expansion of functionality. When speed is essential, replacement will often not be the best choice for the short term, as implementation, training and testing might take too long.

Cost

As with any project, cost is an important factor in deciding what modernization strategy to follow. Sometimes keeping a system running will be prohibitively costly, while other times the added business value of an extensive rebuild can be lower than its cost

As situations between IT landscapes can differ greatly, and the above factors all interconnect, a thorough assessment is the best option to find out what fits best for a particular situation.

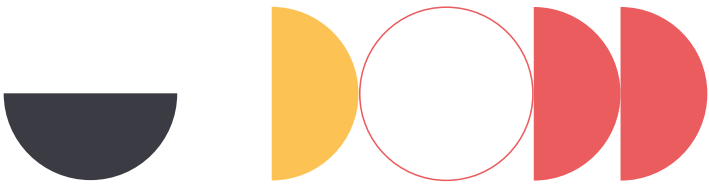


CONSIDERATION 5

BE AGILE

What any large change in an IT landscape needs, is an architectural overview or blueprint, and an execution plan. Armed with knowledge of the existing landscape, a list of shortcomings, problems, and new requirements, we can make choices on what to change where, and how to perform these changes. But bear in mind, software modernization can be a long process, and it is really never done. Requirements from business change all the time, and external factors can shift rapidly. It is therefore **important to be able to respond rapidly, all while not threatening business continuity**. Architecture and realization should go hand in hand, with both being allowed to evolve during the modernization journey.

This is why an agile approach to software modernization is a must. Small iterations allow the business to profit from software modernization quickly, while enabling the IT department to **rapidly respond to changing priorities and requirements.** Modern project management tools like Atlassian JIRA or Azure DevOps can help in keeping track of work, and dividing work into manageable sprints. After each iteration, it is important to measure the effectiveness of that which has been realized during said iteration. Have the issues been resolved that were meant to be resolved by the tasks performed during this iteration? Has there been improvement in other areas, or have new issues been discovered? Using a capable work-tracking system can also help business stakeholders keep track of progress being made, while enabling them to raise issues quickly and to monitor resolution quality and speed.



CONSIDERATION 6

GAIN INSIGHT

As mentioned earlier, software modernization is not a one-time thing, rather we could almost apply the term “continuous modernization”. In order to find out where modernization might be needed in the future, we recommend incorporating technical and functional monitoring into any software that is modernized.

Better safe than sorry

Fundamentally, monitoring provides overview. More specifically, it ensures fast insights and solutions to incidents, which support the business even more. It also allows an organisation to become more proactive and gain increased control over their suppliers. Monitoring provides you with a proactive approach towards incidents, enabling you to avoid impact on your day-to-day business. Causes of incidents are found and solved fast, and troubleshooting takes less time.

As the subtitle explains, monitoring your new or modernized application is a must. The next level in monitoring, however, brings you much more. Imagine having one centralized monitoring tool with just a couple of dashboards, showing you the state of your entire IT landscape, from front-end to middleware and back-end.

- **Central storage of logging data**: Clear central place where logging is collected and secured.
- **Single source of truth**: Everyone uses the same source data for research and monitoring, «single source of truth».
- **Advanced Alerts and Machine Learning**: Can be used from many sources and central storage.

CONSIDERATION 7

BE PREPARED FOR CHANGE

Modernization of IT cannot be seen as purely a number of technical changes. As systems change, the people and processes supporting that system, and the **people using it, also need to change**.

Do not expect IT modernization to be just an IT thing, instead, consider changes to the business side as well. Users will need to be trained on using new systems, and support needs to be able to answer their questions.

Depending on the company, users might embrace change, or reject it as **they have "always worked" a certain way**.

Likewise, the IT department might need to train its people in new programming languages, certify them to use new products, or change internal processes to allow technical changes to bear fruit. This is **a process that takes time and effort, both from the company and the individual**, which both must be willing to invest.

Perhaps certain skill sets will become superfluous; can the people with this skill set be re-trained and given new tasks? If new skills are needed, is the company able to hire people with those skills?

CONCLUSION

THERE'S NO ONE-SIZE-FITS-ALL FOR SOFTWARE MODERNIZATION

Modernizing an application landscape can be a daunting task. Finding out where to start, and what to do in order to get your IT back on track is complex and takes time. **Vendors push certain strategies, such as cloud- migration, as the solution for every problem**. Determining the right approach to take requires extensive expertise.

In order to overcome these challenges, a thorough assessment of current IT capabilities and shortcomings with regards to business needs is of crucial importance. This assessment needs to be done without prejudice and **without steering towards certain solutions**.

Once the initial assessment has shown where the biggest problems lie, and what approaches may be taken to solve them, we recommend using an agile approach to realizing the chosen solutions.

Jasper Baljeu

Software Engineer at Devoteam

e: jasper.baljeu@devoteam.com

l: view Jasper's LinkedIn

SERVICES

WHAT DEVOTEAM CAN DO TO HELP YOU ON THIS JOURNEY

Rapid Assessment

Modernization Strategy

Unlock Legacy IT

Software Development

Monitoring & Observability

Managed Services

Devoteam Netherlands, based in Amsterdam and The Hague, consists of +/-300 technology consultants who guide clients through their digital transformations from day to day. From integration, API Management, Cloud, and Microservices to DevOps, Automation (CICD), Software Engineering, and Data Integration.

Working together with the world's leading technologies such as Google Cloud, Red Hat, Microsoft, Mulesoft, Atlassian, and Elastic, Devoteam functions as a consulting and implementation partner for clients such as Liberty Global, Vodafone Ziggo, Eneco, Rabobank, various municipalities/ government institutions, and many more.

#TechforPeople

info@devoteam.nl +31 (0)20 630 4750