cegeka

Why complex application development projects fail

– and six tips to help you succeed

The number of IT project failures is astonishing. 1/3 of all application implementation projects run late or over budget *.

And Yet, IT had never been more important than it is in today's dynamic digital era.

So, why is it that so many application development projects are put on the rails – often to be paused later – or deliver disappointing outcomes?

Is there a way to beat those odds? There certainly is. In this e-book, you will find six tips to help you keep your complex application development project on time and on budget while ensuring that it meets the needs of everyone involved.

* Source: Use These 10 Contracting Steps to Dramatically Reduce Your Application Implementation Project Overruns (Gartner, 2017)

TABLE OF CONTENTS

1

Why do complex it projects fail?

Why do complex it projects fail?

Large-scale, complex application development projects are more prone to go wrong by their very nature: they take several years to complete, which means that the requirements and scope are likely to change over time. Even more, they involve a lot of people – some of whom may come and go. That makes it harder to retain the same focus, ensure a consistent approach and be successful.

IT project failure can take many different shapes and forms, from exceeding deadlines and running out of funds to falling short of user expectations. **The factors that put IT projects at risk of failure** include:

- poor initial scope definition;
- shifting priorities and scope;
- lack of focus on the budget;
- poor technical expertise or inadequate software development techniques.

Read on to discover **six tips to ensure application development success.**

2

Six tips to ensure application development success

Six tips to ensure application development success

Tip 1: know your users and the business drivers

One of the biggest factors that puts software development at risk of failure has nothing to do with technology. It's failing to clearly understand the user and the business needs behind the initiative. An in-depth analysis will help you understand the business drivers and goals of the project to ensure you get your project scope right.

Your users need software tailored to their specific business needs and goals. So, no matter how great your application looks and how well every function works, it won't be successful if it doesn't do what the users expect it to do.

That's why successful IT projects start with a thorough analysis. This means asking questions, not only about the application needed, but also about the vision, strategy, way of working and the future plans of the company and/or team. By **getting to know your users and the business context really well**, you'll be able to exactly identify the business drivers and, as such, the solution the users need – and take that into account at every step of the software development process.

> *No matter how great it looks and how well it works, your application will fail if it doesn't do what users expect it to do.*

ORGANIZE WORKSHOPS TO IDENTIFY THE TRUE BUSINESS DRIVERS

What exactly is the scope of a new IT project? Does every stakeholder agree on its objectives and the requirements? To make sure that everybody is heading in the right direction, a **thorough pre-analysis** is key.

Organize a series of **workshops with the business** to understand the organization and learn what challenges they want to solve, talking to as many stakeholders as possible. Based on that input, you can then determine and visualize the **business drivers** (goal, actors, impact and deliverables) and the **high-level project scope** (application context, business processes, conceptual model and story mapping). Gauge the feedback on your proposal during another workshop. Last but not least, you'll need to **identify the technology needs**, including functional as well as non-functional requirements, **risks and budget** – again, in close cooperation with every stakeholder.

TIP 1

Take time to **analyze business and user needs** and be patient if your application development partner keeps asking questions. A thorough business analysis at the beginning of a project is the best way to ensure success at the end!

Tip 2: take it one step at a time

No matter how meticulously you've identified your initial scope, requirements and priorities will change in the course of an application development project. By working iteratively and holding regular feedback sessions, you'll stay on top of user needs and respond to change quickly and efficiently.

In software development, change is the only constant. New legislation, a new CEO, changing customer demands: plenty of factors can impact the scope and priorities of your project. The ability to adapt to that change – and communicate about it – is key to the success of your development project.

Current software development approaches, such as agile and DevSecOps methodologies, help teams respond to changing circumstances and trigger constant feedback. By developing the application in **small incremental steps** with **short feedback loops** between development and either the **business** (agile) or **operations** (DevSecOps), you'll spot inconsistencies or issues quickly and can adapt on the fly, solving them before you move on to the next chunk.

In addition, it makes sense to **schedule high-risk development early on in the project's life cycle.** After all, it is easier to identify an alternative approach in the beginning than at later stages.

AGILE AND DEVSECOPS: WHAT'S IN A NAME?

Both Agile and DevSecOps aim to promote collaboration and teamwork and accelerate delivery. Both approaches, however, address different aspects of the delivery process:

- **Agile** encourages short feedback loops between the **development teams and the business**. Agile techniques allow for the quick evaluation and adaptation of the application under construction to make sure it meets business – and, as such, end-user – requirements.

- **DevSecOps** promotes better collaboration between **development, operations and security teams.** It focuses primarily on the frequency of the deliveries, pushing past divisional silos to enable rapid integration, testing and deployment of software changes to a production environment – in order to improve planning, design and release processes.

PROOFPOINT: THE VENTOURIS CASE

"As there is a new release of the Ventouris e-platform every two weeks, we have the flexibility to quickly implement changes. More importantly, every release requires minimal testing. If you have one big update every four months, you need a huge user base to test whether there is software regression. By constantly releasing small updates and implementing minor features, testing is minimal and we can assure users that there is no software regression."

Johan Lybaert, vice president of Social & Government at Cegeka

> *"As there is a new release of the Ventouris e-platform every two weeks, we have the flexibility needed to quickly implement changes."*

TIP 2

Develop the application step by step (incrementally) and get rapid feedback so that you can change course when needed – without big negative impacts.

Tip 3: keep cost top of mind

1/3* of complex software projects exceed their budgets. That's not really surprising, as budgets are set before the project kicks off, when there are still a lot of uncertainties. Moreover, the scope of your project is bound to change overtime. To avoid budget overruns, you'll have to keep costs top of mind at every step – from scope definition to project delivery.

Defining a **clear project scope** that accurately takes into account the project requirements is, of course, a must to set the budget right and, consequently, to avoid the risk of running out of funds. **Prepare for surprises** when you estimate your costs, as large projects typically come with unforeseen circumstances that impact your budget.

When kicking off your project, think big, but start small. Keep the project requirements in mind and build a **minimal viable product** first that **focuses on the must-haves**, i.e. the features that the business needs the most and that will add real business value. Nice-to-haves can then be added at a later stage, if you still have budget available.

To be able to stick to your budget, you have to **keep track of it and communicate about it** with everyone involved, at all times. Iterative, step-by-step development and regular feedback sessions are a big help in this. During biweekly meetings, for example, you can review not only the application and its progress, but also the actual spend versus budget and reset priorities if needed to prevent things from getting too far out of hand.

Prepare for surprises when you estimate your costs, as large projects typically come with unforeseen circumstances that impact your budget.

TIP 3

Think big but start small' should be the adage when developing an application: focus on the must-haves and only add nice-to-haves when you're sure you have enough funding left.

* Source: Use These 10 Contracting Steps to Dramatically Reduce Your Application Implementation Project Overruns (Gartner, 2017)

Tip 4: develop software like a pro

Elaborate IT projects that involve large applications are complex and, consequently, more prone to failure due to technical issues. That's why they require clear software architecture, hands-on development methodologies and software engineering best practices that every developer strictly adheres to.

Smart software development starts with the creation of the right software development team: a group of **dedicated experts** capable of achieving the objectives of the project and that understand the project's vision, goals and timeline.

Successful software developers write **clean, reusable code** that is easy to test and read. There is a wide range of **best practices** that contribute to building high-quality, low-complexity code. If the entire software development team follows a **consistent approach** to design and coding standards, code will be easy to share among colleagues – and their future colleagues – to ensure the continuity of your project. Moreover, this software craftsmanship leads to built-in quality and first-time-right applications and software projects.

Last but not least, make sure your software engineers are **constantly learning about new trends and technology** – including new languages, frameworks, methodologies, etc. – and that they are not afraid to apply their learnings directly to their work.

A SELECTION OF SOFTWARE DEVELOPMENT BEST PRACTICES

Every best practice is important for building high-quality, future-proof software. In practice, however, software teams often choose which best practices to use according to the context and needs of each project. Examples include:

Readable code
Readable code is key to ensure high-quality software. Developers have to write code that is easily understood by colleagues on the team and will be self-explanatory for those maintaining the code in the future.

Pair programming
To avoid quality issues, you can have two developers work together simultaneously on a task, one in the role of the 'driver' and the other as the 'navigator'. The driver types the code while explaining what they are doing and why. The navigator thinks ahead to the next steps and potential pitfalls, anticipating issues before they occur. By combining their experience, the developers often come up with solutions that might not have occurred to them alone.

Collective code ownership
Collective code ownership means that the code belongs to the whole team. That means that any developer can edit any piece of code and start working on the next iteration. In addition, people are encouraged and expected to make any changes in the code needed.

Domain-driven design
If the words used in the software do not precisely match those used by the business, it can lead to all kinds of problems. Domain-driven design connects the implementation to a model of the domain the software will be used in, using a language that is shared by the team and the customer/ users.

Refactoring
Continued effort to keep the internal code structure clean and in line with the evolving architecture and frameworks helps avoid costly reworks at later stages of the project.

THREE KEYS TO KEEPING YOUR SOFTWARE DEVELOPERS MOTIVATED

More than ensuring that your development team has great development skills, how can you keep them engaged, satisfied and productive? Three keys to keeping developers happy (spoiler alert: an outrageous salary isn't one of them):

1. Give them a purpose
Employees – especially millennials – increasingly look for jobs that are useful to society or that give them the chance to help others. So, if you want to bring out the best in your people, give them purpose.

2. Ensure access to the latest technologies
Software engineers generally enjoy trying out the newest innovations. Let them use the latest and greatest tools and test new technology. Even more, make sure that developers who update or maintain existing applications can combine that with exciting new projects.

3. Provide ample opportunities to learn and grow

Tech workers are always yearning for opportunities to grow and advance. Satisfy that desire and challenge them to prove themselves on a bigger and bigger scale.

PROOFPOINT: THE VENTOURIS CASE

"We continue to add young talent to the 40-person Ventouris team. Investing in continuous training and keeping in-house knowledge up to date is a priority at Cegeka. Challenging work is the key to maintaining their engagement. Our way of working ensures that the people on the team develop state-of-the-art software even after 20 years, and at the same time, we offer the best quality to our clients."

Natalie Vanderhasselt, sales manager at Cegeka

> *"Investing in continuous training and keeping the in-house knowledge up to date is a priority at Cegeka. Challenging work is the key to maintaining the engagement of our people."*

TIP 4

Make sure your entire software development team adheres to best practices and follows a consistent approach to design and coding standards. That's simply a must to ensure the quality and continuity of complex application development projects.

Tip 5: test, test and test again

While software engineering best practices are a great start to developing high-quality software, properly managed testing is just as important. Consistent, repeatable and automated tests will help you detect bugs from the early stages of development onwards to prevent your application from failing once it's up and running.

Over the past few years, there's been a lot of buzz around the use of automatic testing as a way to speed up the software development process – and rightly so. However, the best way to ensure complex applications of the highest quality is to c**ombine manual and automated testing** – as both have their pros and cons.

When determining the design or the usability of your application, for example, human insight is critical. Hence, **manual testing** – by developers as well as users – is your best option. PEN testing, for example, requires human creativity to assess whether your applications and infrastructure meet the security standards needed to protect them against outside threats. **Automated tests**, for their part, are ideal for testing large applications with many variables and many platforms, and useful when tests have to be quick and accurate. They can be used to verify if an application still works correctly and if non-functional requirements, such as performance needs, are met.

> *Automation is ideal when tests have to be quick and accurate and for testing large applications with many variables and many platforms.*

PROOFPOINT: THE VENTOURIS CASE

"Automated testing ensures a high quality standard and offers us the flexibility needed to release a new version on a biweekly basis."

Johan Lybaert, Vice President
Social Impact Division at Cegeka

TIP 5

Testing automation is definitely worth the investment, especially in large application development projects. Although building and maintaining an automated test suite does take time up front, it is sure to save time later on.

Tip 6: collaborate and communicate nonstop

If there is one golden tip in our series, it's this one: never forget the human factor!

We've highlighted it throughout this e-book: communication is king. Talk to the business and the users and question them extensively to **determine your scope**. Ask for their **feedback on a regular basis** to make sure you stay on track, avoid misunderstandings and can swiftly adapt your application whenever needed without losing too much time. **Talk openly about the budget.** Collaborate closely with peer developers and the business to develop high-quality software and test it.

> *An IT project is doomed to fail if collaboration and communication fall short.*

TIP 6

Software projects are about people. Strong collaboration and communication may mean the difference between project failure and success.

In conclusion

Large, complex software development projects are not doomed to fail by definition. There are sure ways to avoid crashing and burning. If you take our six tips into account, you're well on your way to success:

1. know your customers
2. take it one step at a time
3. keep costs top of mind
4. develop software like a pro
5. test, test and test again
6. collaborate and communicate nonstop

Looking for more tips to make your large, complex application development project a success? Are you looking for a partner that has proven its worth in countless complex IT projects, or interested in seeing the Cegeka team in action?

PROOFPOINT: THE VENTOURIS CASE

"There are not that many firms with the same level of discipline when it comes to software best practices. All this leads to high-quality software that is still up to date today. For Ventouris, we can count the number of bugs we're fixing on two hands at any given time. Since we started building Ventouris over 15 years ago, we have never had any serious incidents. Moreover, the speed and functionality have never been questioned by users. These are quality parameters that prove the Ventouris platform's reliability."

Natalie Vanderhasselt,
Sales Manager at Cegeka

Check out full-blown Ventouris Story

> **www.cegeka.com/ventouris**

WWW.CEGEKA.COM