TheCodingMachine
TCM://

Project Rescue

—

There can be many reasons why a project fails, is unpopular with users, or has gone wildly off-schedule: design faults, incompetent service providers, or unclear specifications can all play a role.

Nicolas Peguin, Partner
Jean-Guillaume Dujardin, President

April 2016

Introduction

TheCodingMachine has helped many clients to rescue their projects. These critical situations enabled us to understand how these slips happened, to find solutions and to implement them.

"Project rescue" is an exciting subject in several ways:

- It requires a wide range of skills:

Human skills to objectively distinguish real problems from imagined ones, technical skills to get to the core of the applications, not to mention the creativity to imagine and find the right solutions.

- This is a challenging activity:

Client expectations are high. These kinds of assignments require investing maximum energy to yield results in record time in order to convince the relevant parties and implement the corrective measures.

This white paper provides a summary of our experiences, pitfalls to avoid and the methods we have applied.

If you currently have a project you feel is in danger or rife with problems, or if you simply want to deepen your understanding of project risks, this white paper is for you.

Nota bene: This white paper is the result of our experience. You may find it flawed or limited. Furthermore, the solutions presented do not apply to all situations, and therefore their efficacy cannot be guaranteed.

1 A project evolves in a hostile environment

The difference between "risk" and "failure"

Risks can be managed by taking corrective measures. Failure occurs following an accumulation of unmanaged risks that often lead to the project coming to a general halt.

If you're unsure whether your project has totally crashed or if you're simply managing a risk, then you're at a crucial stage in your project's development.

Generally speaking, the later you realise the project has crashed, the more serious the problem is. In other words, the more we accumulate risks throughout the project, the trickier the rescue will be!

The dangers that lie ahead!

Every stage of a project involves some risk. For the most part, we are aware of these risks well in advance, although some reveal themselves in specific situations because they are related to a specific problem.

Nevertheless, structuring the response to meet the users' needs (specifications) and choosing the right service provider are key steps to limiting any overspill into future projects. You should pay close attention to this

Project stages	Risks
Requirements, Specifications	Interviews with users or clients (especially for web business models) are essential for understanding needs. There is a risk of writing a complex but non-comprehensive document. If you organise a consultation but do not possess technical skills yourself, allow your service provider to design the most appropriate technical solution and give them the opportunity to respond with the technologies in which they excel. However, do not disconnect completely from the technical process. Failure to oversee this aspect is very dangerous. Wanting too many features is sometimes risky. Start small, with the essentials: the more you expand the scope, the more difficult the project will be to manage. The distribution of the solution must be logical and understandable. The aim is not to build with non-functional bricks.
Functional and technical design	Do not let a project start (except for a very small portion) without having validated - and therefore perfectly understood - the functional specifications. The specifications guarantee a good understanding of the project, which can only be developed from these documents.
Developments	Progress reports must be made to ensure the project's performance. A delay may represent a technical risk in terms of the architects' / developers' skills. Information should be communicated on a regular basis; this includes bad news as well as good. These reports should detail the risks and how to manage them. The integration should enable the service provider to technically validate the solution. Developing for any length of time without user involvement is problematic. It is important to counter this risk by providing for intermediate testing.
Implementing the solution	Get involved in your project's testing. Log any tests, and create datasets to test the key elements of your project. The dimensions and environments should allow for optimal use of the application (server management, network infrastructure, etc.). It is better to separate the development from the production (setting up the servers, the environments, and managing the backups etc.).

The risks in detail

- **The fear of "not enough", or the scope getting out of control.**

The fear of "not enough" is common. From the outset of the project, the client defines a broad scope or expresses an increasing number of complex needs as the project progresses.

This is common when a client starts a new project. By offering a wide range of features, they feel like they are giving a reassuring impression i.e. providing a more comprehensive service to the end client. In reality, the opposite is true. You should define the smallest possible scope, test quickly and then redefine the needs through user feedback or beta testers ("Pave the cow path").

Clients also believe, wrongly, that defining a broad scope at the outset is more cost effective than multi-step development. This is a false economy. Ultimately, if numerous features are not useful, the cost of developing them, and the effort required to produce them, can lead the project to failure.

When the scope of a project slides, it can also be due to poor project management. A service provider that finds themselves in a tricky position due to major delays or lack of resources may well agree to take on extra developments in the hope of maintaining good relations with their client. To do so is to do a great disservice. By accumulating delays and adding new development work, the project will take even longer to complete

- **The hook price**

The service provider's price is deliberately low in order to "hook" the client. Service providers often deliberately reduce the price of their services in order to win a contract.

Then, as the project progresses, two situations will arise

- The development continues because you accept the adjustments to the original quote
- The project is blocked because you do not want to (or cannot) pay for the adjustments

This "fake price" can seriously degrade the relationship you have with your service provider as the project progresses.

However, it is important to distinguish between a deliberate action (a service provider who "hooks" a client) and a scope that evolves and grows excessively along the way (in this case, some of the fault lies with the client).

- **Measure rather than assume**

From the outset, it is vital you implement a mechanism to measure progress, manage risks, and advise or make decisions about any additional developments.

These measurements: time consumed, progress etc. should enable you to provide the entire team with an objective indication of your project's status. They also make it possible to implement action plans to manage certain risks or to reschedule certain parts of the project.

- **The tunnel effect**

The tunnel effect consists of developing for a long time without ever involving users.

The developed solution may ultimately not be suitable for users' needs / wishes. The testing phase - long and therefore tedious - may then be chaotic; users won't necessarily have the time to see it through to completion.

- **The number of anomalies**

This is a question that often arises: Does a large number of anomalies during the testing phase indicate that a project is going to fail?

In most cases, no! The greater the number of anomalies, the more your users test the solution. It is when the solution is insufficiently tested that a project can run aground. So, in a counter-intuitive way, this indicator is in fact more of a quality factor.

The first technical test should have solved most of the basic anomalies. If this is not the case, you may need to reframe to avoid exhausting users during testing.

This also makes it possible to determine if the corrections are effective. Performance issues that cannot be corrected may indicate significant design issues.

2 Rescue methods

The verdict is unanimous: your project has crashed. Do not panic, TheCodingMachine has the solution! Getting the right "diagnosis" solves much of the problem, because once the problems have been identified, an effective action plan can be implemented.

To guarantee a reliable and honest result, it is best to seek advice outside the company. The very first step is to shed any peripheral problems that inevitably ride alongside an errant project and blur our vision. Appealing to an independent third party can help you to see more clearly.

The importance of context

A successful project rescue means getting to the core of the problem in order to obtain an objective vision of the project's status. What are the problems? Can they be resolved quickly? If so, by whom and how?

The solutions depend closely on the project's status: is the project still in development or in production? They also depend on the nature of the problems: Is this a technical problem or a problem concerning the scope? These solutions obviously depend on the size of the project. The smaller a project is, the easier it is to start from scratch.

Méthode 1 - Ne pas avoir peur … de supprimer le problème

If the problems are of a technical nature, it is unlikely you can "rescue" a developed application. Correcting serious architectural design mistakes is often very expensive. For the same budget, it is better to resume technical developments from scratch.

Therefore, it is vital to perform a rigorous analysis of the existing product to avoid throwing away an application that could potentially be rescued and equally, to avoid trying to rescue a non-operational application and in doing so, prolonging its failure.

Method 2 – Manage peripheral problems: Solutions for managing human issues.

You must first establish that the project is in failure. This will be obvious to the project managers whose service provider has abandoned the project, but for those whose service provider is still reassuring them, promising success and attempting to maintain a good relationship, establishing failure can prove to be difficult.

If the application is worth saving, you will need to overcome the human aspects of the problem. This is tricky because both parties tend to feel defensive. It is always painful to be wrong or to feel that you have been tricked by an unscrupulous service provider. However, complaining won't solve anything.

Unfortunately, there is no universal solution. We recommend common sense and pragmatism: are you a victim of your service provider or of yourself? Will changing the project management enable you to see the project in a new light? To bring new motivation to the team? Can you create an 'electroshock' to trigger awareness among the teams?

Here, creativity knows no bounds! Here is a list of the most obvious and effective courses of action:

- **Changing and replacing the service provider often makes it possible to see the project in a new light and to remove any recurring problems:** Is this option desirable and/or possible? What additional costs will it generate? What benefits will it bring? How do you envisage the transition phase?
- **Take a break, gather your thoughts and get organised**: Is this break desirable and/ or possible? How long (too much could cause the project's failure)? Who should plan the rest of the project, and how?
- **Changing teams can give a new lease of life to a project.** In this case, be careful not to speak ill of the project, so as not to discourage the new team: Is this change desirable and/or possible? How will you restructure the team, and who will you keep?

- **Relaunch the development, step by step. Breaking down the problems can help you to see the light, which is motivating:** Is it possible to split the project into subsets? Is the cost of is this reorganisation manageable? Are the teams ready for this change?

Method 3 – Managing technical problems

If the project can be rescued, then you need to act fast! However, acting methodically and rigorously is key to the rescue operation.

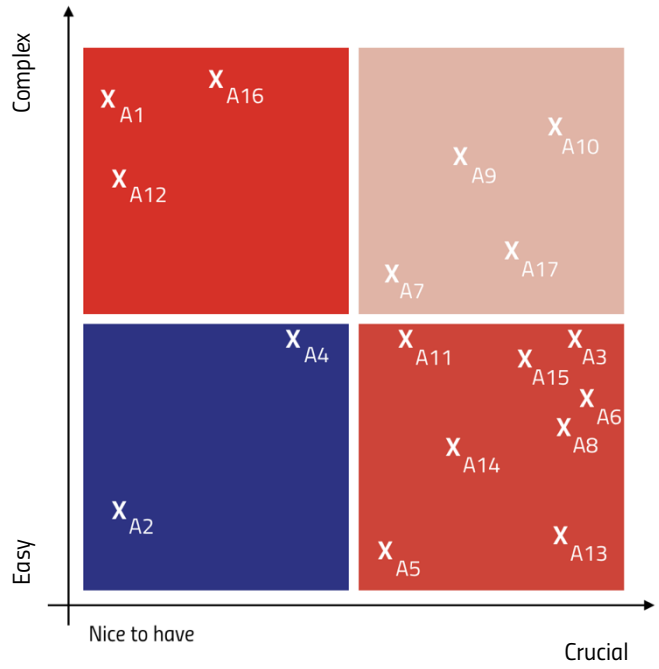
Several techniques can enable you to identify the risks and actions associated with the recovery of your project. We offer one that has the merit of being both simple to share and easy to apply:

Ranking risks and actions:

- Impact: the cost associated with the occurrence of the risk. For example, it costs more if the main database crashes than if the mail server crashes.
- Probability: the probability of the problem occurring;
- Efforts to correct: The cost associated with setting up a patch;
- Category: the category associated with the risk (architecture, code security, etc.)

Description of the risk and associated actions	
	Category
Impact	Low/Average/High: description of the impact
Probability	Low/Average/High: probability of the event happening
Efforts to correct	Low/Average/High: description of the tasks necessary to correct the problem

The actions are then placed on a quadrant, with impact*probability on the abscissa, and the complexity of setting up the parch on the ordinate



The quadrant's actions on the bottom right will be priority actions.

Specific cases – Performance issues

Performance issues are often overlooked in projects. They often do not appear until the very end of the assignment, unless they were foreseen, and are difficult to detect until the project goes into production.

3 TheCodingMachine's Best Cases

Case 1 – A badly-designed architecture

- **The project:**

As part of its paper-free correspondence policy, the subsidiary of a key player in bank-insurance asked a service provider to create an electronic document management system (or EDMS). The project included outsourcing the opening and scanning of mail, and then distributing mail in electronic form through an application that managed client request workflows.

- **Our intervention:**

  Given the delays that accumulated during the development of this project and the dysfunctions noted, TheCodingMachine was hired to conduct an audit, to analyse the application in close detail and define the scope of the rescue plan.

- **The problems detected:**

  The solution's architecture had been very poorly designed. The data was distributed in two separate databases. The service provider had worked with an open-source tool and technology with which he was unfamiliar. The response times were unbearable for users and sometimes even rendered the application unusable (exceeding 5 minutes). The application was unstable, and there were sporadic production shutdowns lasting several days.

The solutions provided:

The build process and the systematic use of caches and sessions made it possible to consolidate the application without calling into question the tool's foundations.

The industrialisation of developments, the implementation of good practices and the launch of automated tests to ensure no regression occurred, ensured the application's stability over time.

Case 2 – An ill-adapted technique

- **The project:**

The client, working in the field of real estate, wanted to offer its employees an Intranet: to present information, manage training, report back etc.

- **Our intervention:**

Having fired their former service provider, the client had to establish that their project was in failure. Part of the design was taken over to maintain the database schema for user information.

- **The problems detected:**

The technology was totally unsuited to the needs, because of the service provider's inexperience in this type of project (more accustomed to showcases sites than Intranets).

The solutions provided:

The project was redesigned in record time and with a reduced budget. Indeed, the client had already spent 50K leaving only 30K available for the rest of the project. Thanks to an extension on the budget, we were able to review the whole structure of the code: MVC framework, Open Source library, etc.

Case 3 – A service provider that abandons a project

- **The project:**

A renowned beauty franchise wanted to sell its services online. A marketing agency won the project tender and then sub-contracted the technical part to a company called 55II in Tunisia.

But profound changes brought about by the Revolution, and due to a tight budget, the service provider was forced to abandon the project.

- **Our intervention:**

In order to meet the deadline, which was very important for the client, our rescue operation was conducted in stages:

- we scaled back the scope to ensure a pre-production version could be tested by the client and to provide them with some visibility of the project's progress
- in parallel, we finalised the development of the initial scope.

- **The problems detected:**

The technology was totally incompatible with the client's needs due to the service provider's inexperience on this type of project, having been more accustomed to developing showcase sites than intranets.

The solutions provided:

The initial architecture - use of Magento - was preserved.

However, we redeveloped and reconfigured all the specific features: multi-store, themes, add-ons etc.

OUR OPINION

It is impossible to conduct a project rescue operation on behalf of a third party, rather than the end client. Communication and client feedback are key to this type of rescue.

You need to liaise with the client to reassure hem as much as possible - et ne pas partir avec une agence qui dénigre la technique !

Case 4 – A sliding scope

- **The project:**

A mail-order and door-to-door sales company decided to overhaul its information system (stocks, sales, billing, etc.). The choice of an off-shore service provider with a complicated structure (no direct communication but via a third party, etc.) meant the client ended up with a half-finished application.

- **Our intervention:**

We started with a technical audit of the existing "solution" (although few features were complete). Then, along with the client, we restructured the project.

- **The problems detected:**

Two main problems were detected:

1. There was no precise specification of the functions, the development had been carried out in bits, guided only by a broad description.

2. When relations became strained between the client and the service provider, the development was accelerated to the detriment of technical quality and coding rules (use of installed tools).

The solutions provided:

We started from the most advanced point of development, strictly adhering to rules and standards in order to recover and reuse the functions developed.

We then restructured the development by composing the missing specifications, we then evaluated and prioritised each task accordingly.

OUR OPINION

You must never abandon the basic technical rules for the sole purpose of saving time.

This often proves counterproductive. The project would be, at best, too complicated to maintain, and at worst, impossible to bring to run.

Conclusion

No one method should be applied in isolation, but rather used as part of an offensive.

It is simple enough to estimate whether an application is worth saving by establishing the quadrant for managing the technical problems. This quadrant can also be useful in making a rational - and therefore more readily accepted - analysis to ascertain whether or not the project is, in fact, in failure.

Our experience has shown us that there are technical solutions to most problems – though they can be expensive. Often, the hardest part is managing client to service provider relations. It can be tricky to defuse a conflict caused by an accumulation of failures during a project. However, it is important to maintain calm communication in order for rational decisions to be made with the project's success in mind.

While there is no one-size-fits-all solution for these types of problem, we hope that this white paper will have provided some hope if your project has come to a standstill, crashed or failed altogether.

We acknowledge that a number of elements exposed here can be explored in greater detail. Please feel free to contact us if you wish to discuss any of these matters further.

FOLLOW US ON LINKEDIN

Find all our **news**, **articles**,
white papers, **innovations**
and more!

hongking@thecodingmachine.com

Tower 535
535 Jaffe Road, Causeway Bay

TheCodingMachine
TCM://