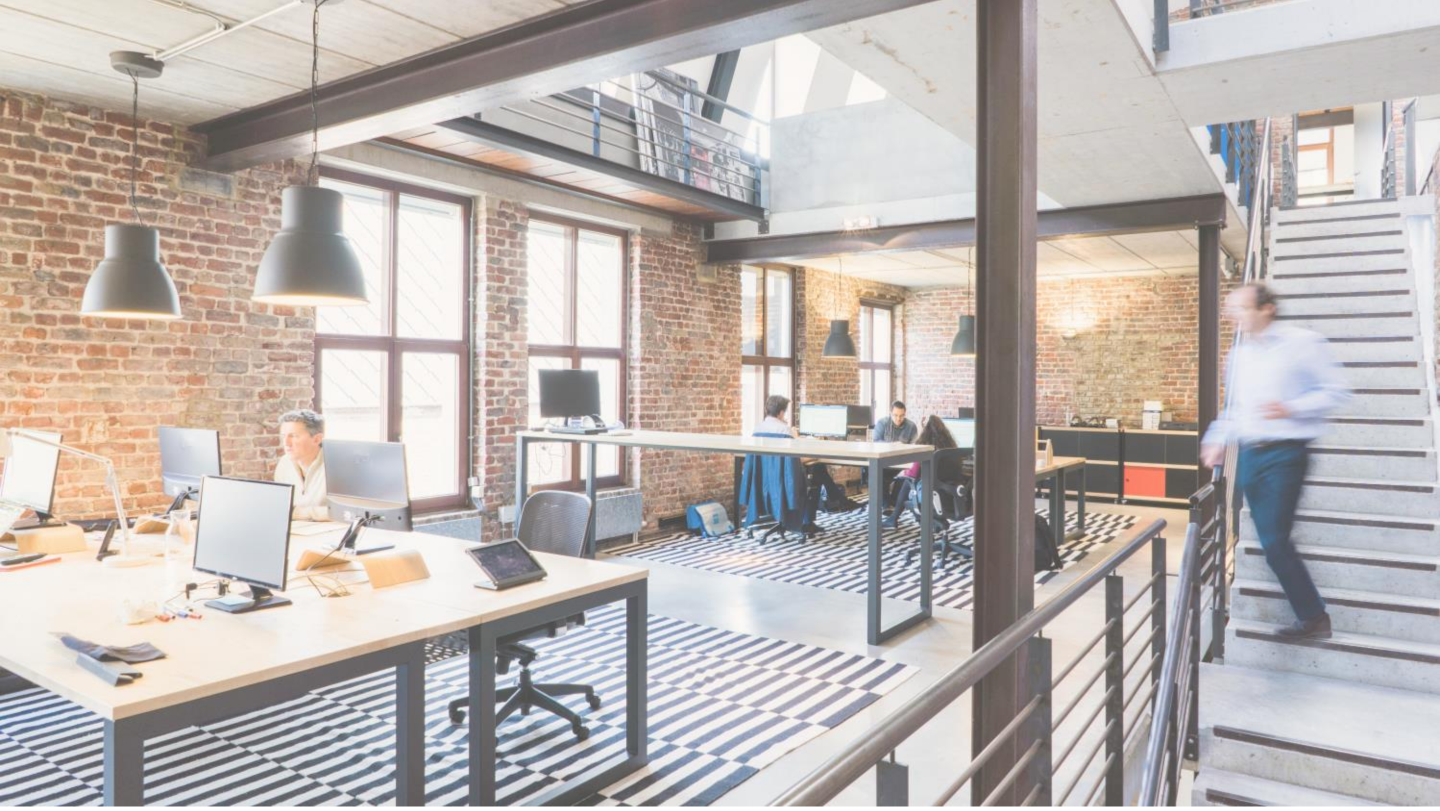
Is your software project

Doomed From The Start?

5 research findings & 10 strategies for success

Geneca



Contents

The Introduction

A software project is a balance between business and IT. Each team needs the other in order to deliver an on-task, on-time, on-budget product. From the very beginning, the teams need to have shared ownership with each one understanding they are essential to success.

Although it is true that the IT team will log in more hours in the project, their success is measured by the business. They need the business team to participate fully. A common complaint of IT teams is never having enough of the business leaders' time.

On the other hand, the business team members have full time jobs outside of helping with the software project. Whether they are in sales, marketing, customer service, or another division, they have plenty of work to do every day. A common complaint of business teams is not having enough time to help IT do their jobs appropriately.

How can these two different viewpoints come together for success? By building on knowledge gained from research into projects that failed and best practices utilized by companies who have built large numbers of software products, leaders can take practical steps toward success.

Read on for 5 key research findings and 10 recommendations to help you achieve success in your next software project.

The Research

Geneca surveyed approximately 600 U.S. business and IT executives and practitioners as part of ongoing research on why teams struggle to meet the business expectations for their projects. Responses revealed 5 key findings that lead to failed software projects.

Participants represented a range of industries with 33% in finance/insurance/healthcare, 29% in manufacturing/industrial goods, and 23% in technology/professional services. 34% of respondents work at companied with less than 500 employees, 50% with 500-5,000 employees, and 16% over 5,000 employees. Most of the participants work for companies with revenues between $100 million and $1 billion (67%).

All research was directed by an independent research firm. Data reports are available upon request. Please contact operations@Geneca.com.

The Overview

“75% believe projects are doomed from the start

Although research on the high percentage of software project failures is not new, this study focused on discovering when and why individuals believe the failures occur.

Many responses reflect a positive attitude to projects while articulating key pain points related to project execution. Interestingly, responses from IT professionals and business counterpoints are fairly similar throughout, revealing many of the same issues and concerns with regard to their projects.

The overall perception is that challenges start at the very beginning of a project and increase as the project progresses.

By understanding the root causes of the uncertainty about success, leaders can implement a few simple, but powerful, changes to help teams anticipate and overcome roadblocks.

Finding

1

Out of Sync Stakeholders

Out of Sync Stakeholders

Software projects need to begin with a shared vision between the business team and the IT team. The business understands **why** the software needs to be built and the IT team understands **how** it should be built. If the two teams are out of sync, the when and how will be greatly impacted and result in project failures.

78%	State that the business and IT teams are always or usually out of sync with each other
43%	Responded that there is confusion around what the business is asking for in the project
31%	Identify lack of a common vision on project success criteria as the greatest barrier to success in project completion

Perhaps the most concerning response is the huge percentage who believe the teams are rarely in sync. If the project becomes us vs them, failure truly is inevitable. These responses indicate that projects begin to fail before they have even started; however, you can take several simple steps to start for success.

In order to forge a project team where business and IT work collaboratively and in alignment, you will need to clearly articulate goals and examine team language choices.

1

Clearly Articulate The Goals

1

To prevent feeling of out of sync throughout your software project, take a few minutes at the start of your project to create a common vision through clearly articulated goals. This does not mean you need to have every detail explained, but a quarterback needs to know where the receiver will be before they get there.

However, you don’t need to distribute your business plan as required reading for all team members. Often, that level of information would lose meaning with people that aren’t performing that part of the plan. Show them what they need to own and the pieces they are connected to prior to the start of the project and craft a list of a few topline goals for why what they are doing is important. They can start with “we are building this software to . . .” or “we are adding 3 features to our product in order to x, y, and z.”

Keep the wording clear and simple. Write out the major goals and post them in a central location. As leaders of the project, you need to use these goals both to maintain a common vision and to reduce confusion. Constructive debate will happen throughout the project as team members make suggestions and work toward those goals.

When team members make individual contributions to those goals, they will use these goals to verify they are in line with the bigger vision. Taking an extra few minutes to ask, “Does this help us reach our goal?” can save rework time later, help keep the project on plan, and get the best thinking and efforts of everyone involved.

2 Language Shapes Our World

An old adage states we should "say what we mean and mean what we say". This holds true for software projects, but it is not always easy. Language is not very good at explaining concepts.

Life experience shapes how we perceive the world and that perception alters our interpretation of various words. When two people look at a color and say its name, that answer will depend on how deep their past is related to colors as well as how their eyes and brain function at color perception. Your project has many of the same challenges.

1. **Shared Words.** Because words can have various meanings, make sure you take nothing for granted. Your version of red may be maroon to me. Consider creating a glossary of words and their definitions that include both commonly used business and technology words as they relate to this project.
2. **Templates.** Commitments in a software project need to be measurable and actionable. Saying you are "working on it today" does not produce an outcome but rather informs the receiver of how you are spending your time. In a project, what is important is when you will have that complete. Create wording templates for more exact phrasing like, "I will have that complete by stand-up tomorrow morning." Help your team provide the data that is necessary, not merely informational.

Given the heavily collaborative nature of software development, failures in communication are most often the cause of derailment of a project. It is faster and more cost effective to ask twice and code once rather than let failures in communication result in code rework.

Finding

2

Unclaimed Accountability

Unclaimed Accountability

Who is responsible for the software project? The answer needs to be "all of us." The IT team should never build software **for** the business, but instead **with** the business. With a variety of cooks in the kitchen, setting up roles, responsibilities, expectations, and accountability becomes a necessity, not a luxury.

38%	Identify confusion around team roles and responsibilities to be the greatest barrier to delivering successful software
30%	Identify a lack of clarity around team roles and accountabilities to be the greatest frustration during a software project
24%	Believe that stakeholders do not align in staffing, budget, time, and progress tracking

By identifying responsibility both as a barrier and a frustration, responses indicate the unsettling nature of confusion. People want to do good work and, to do so, need to have clarity in roles and responsibility.

To create a project team with clear accountability, you need to create a cohesive plan and foster a culture of ownership among every member of the team.

Create a Master Plan

The nature of successful business requires planning. In fact, it is often said that the difference between a wish and a goal is a plan. Usually, the problem within a software project is not that there is no plan, but that there are too many plans. Marketing, sales, finance, customer service, and IT will all craft plans related to the software project.

Each plan will have intersection points and dependencies on the others but, all too often, these are not shared and communicated. Handoffs from one team to another will fail and the project will be negatively impacted. The bigger the build or the bigger the business, the more dependencies and handoffs there will be. In order to create and maintain order with so many moving parts, you will need a master plan.

You need two things to make this work.

1. A single person responsible for everything coming together. This is the person that pushes obstacles out of the way and makes sure the work meets at the right place down the road.
2. While every group needs a plan for the work they are doing, there needs to be visibility into how all the pieces fit together. Use this to let each group know who is the consumer or their work and who they need work from. Focus time on meeting the commitments to produce those connection points.

The master plan will need to be reviewed, updated, and shared regularly throughout the lifecycle of the project.

Foster Ownership

Most often people are given a project but never really make it their own. It is so tough to make things happen when the people doing the work don't care. Team members need to feel the purpose, the goal, and the pride of success. They need a common enemy to rally against whether that is a completion date or a technical challenge. In all cases, if the people involved don't care about the project, it will fail.

How do you foster that ownership? The kind of passion and focus you need never comes from someone just doing their job, it comes from people caring about what they are doing and believing in why they are doing it. It is the difference between a job and a career, a supervisor and a leader, a failure and a success.

The challenge here is getting the team to gel together and rally against the goal. Avoid making yourself as the leader the bad guy to encourage the team to band against as that will not bode well for the long term. Instead, be in the game with them.

Take the project goals and the project deadline and say, “we need to get from here to there by then” and make the plan together. You will need to let them see you hard at work to make it happen. If they perceive they are putting in much more effort than you are, they won't follow you into battle, they will just watch you go.

Finding

3

Unclear Requirements

Unclear Requirements

When asked to define requirements, less than 20% of team members—business or IT—selected “articulation of business need” as the purpose of the requirements process. If the needs of the business are not at the center of the requirements process, how will the final product be what the business needs?

70% Of respondents believe that requirement issues will result in a project that is over budget or fails to deliver the desired capabilities

57% State issues with requirements will result in a project not being considered an overall success

61% Relate poor requirements to the project taking longer than the estimated time to complete

Of all the findings, unclear requirements can be most easily tied to project failure. Requirements most often take the blame for where the project went off track, providing strong insight into problems lurking in the software development process.

For a software project to be successful, the requirements need to be broad enough to road map the business need but detailed enough to allow the development team to code.

REPORTS / REPORTING
THE REPORT WRITING IS HARD WORK IF YOUR NOT ACADEMIC
I SPENT MORE TIME NEGOTIATING FORMAT THAN BEING CREATIVE
COLLATING ALL THE INFORMATION AT THE END WAS HARD DISTILLING FRAMING
GOT HELP FROM OTHERS TO 'GOVERNMENT IFY' REPORT
I WOULD HAVE MADE IT PRACTICAL IF IT WAS FOR EVERYONE

5

Bite-Sized Requirements

5

In the current agile world of software, gone are the days of creating all the requirements before a project starts. Gone, too, should be the extensive requirement documents with hundreds of pages of text. Why?

Because the purpose of requirement documents is to clearly state what the business needs the software to do and how it should do it in a way that the development team can build it. Long, tedious documents filled with technical jargon tend to be signed off by business folks who do not read the document as it is too difficult to comprehend. They default to believing it must be correct since they told IT what they wanted, and the document should be the translation of that need.

On the flip side, the developers don't want to read all that either so most will flip through them, look at a few pictures, and then start coding. The problem gets worse with more experienced developers who will fill in much more of the information without reading it. Unfortunately, their attempt at efficiency can result in wrong functionality.

What does that leave us with? Two teams—business and development—with one thing in common: unread requirements. Instead, create bite-sized requirements that are easy to consume by both business and IT. Combine functionality into a scenario or story that makes sense to all and write the requirements using clear language shared by both tech and business. Using the established common language will allow you to spend your time on building, instead of rationalizing or defending.

6

Create Real Visuals

6

Project visuals are an important part of the requirements process. They start with sketches on whiteboards or with pencil and paper. These sketches become the wireframes the IT team will need to begin writing the code.

Unfortunately, while IT teams are used to working with low-fidelity mockups or wireframes, most people are not. Without significant experience creating software products, it is difficult to look at a sketch and be able to visualize the end product. In order to see what their product will look like, business members need a better visual. Whether you choose to create a few high-fidelity images or stub out the actual application, you will want to showcase several strong visuals to your business team early.

By presenting stronger visuals in the beginning of the project, you have the opportunity to work together to make any changes before you build, and then need to rebuild, your pages. You will save considerable time and frustration by taking extra care at the start of the design.

Otherwise, you run the risk of a system that doesn't match what was intended and isn't discovered until the business begins user acceptance testing and can look at the results of what they asked for. You do not want the real design and requirements to begin at testing when the business is forced to choose between what they thought they asked for and what they received and must decide whether they pay for more work or subjugate their vision. Even if the team must deliver for no additional cost, you will still lose either on time or on expectation.

Finding

4

Inconsistent Involvement

Inconsistent Involvement

Many organizations consider technology projects to be exclusively an IT responsibility rather than a joint responsibility with business. As a result, business involvement in the project often decreases with time. As their participation decreases, project failure increases and the predictability of success diminishes.

25% State the business does not remain engaged in the project or leaves the process to IT

42% Responded that IT does not build what the business asks for

70% Believe their CEO would rate the ability to deliver software projects without surprises as most important

Surprises occur when people are not involved throughout the project. By participating as the project evolves, the business team maintains the visibility needed to avoid being surprised. Instead, they provide feedback and make informed choices contributing to project success.

Consistent involvement prevents making reactive decisions through false urgency and provides proactive ways to make the plan become a successful reality.

GENECA

The Urgency of Organization

Urgency can be thought of as artificial importance. It is important for my health (both physically and mentally) for people to eat.   Delaying eating or downgrading the importance of eating lunch to below the importance of finishing a task happens. However, at some point, the body will insist eating become urgent and will no longer allow it to be downgraded.

Unfortunately, that urgency doesn't usually cause the best decisions or the best results. When urgently hungry, people make poor choices as to what to eat. Grabbing a candy bar to reduce urgent hunger back to important hunger is not the healthiest option for the person's system.

Projects are exactly like that. When important items are allowed to become urgent, poor decisions are made in the name of immediacy that will lead to project delays and failures. Urgency is ultimately a function of organization, prioritization, and discipline more than anything.

Knowing all the things we need to do and how they relate to the things around them allows us to rank their importance, determine priority, and schedule when they should be performed. Having the discipline to complete items in the agreed upon order is a challenge for people. All too often, we will choose to complete easier or more fun items before returning to more difficult tasks. This can lead to the most challenging tasks being connected to the poor urgently made decisions. Create visibility and commitment around items to ensure they never advance to artificially urgent.

8 Make the Plan Come True 8

There is a difference between the people that have success consistently and those who do not. The difference is leadership. Watching a project does not make it happen. Taking action makes it happen.

"The best laid plans of mice and men often go awry."

That quote from Robert Burns is so true in the world of software. We work so hard to decompose all the work that needs to be done. We estimate it out by getting input from multiple people and tuning our estimates based on past results. We put together a team that has exceptional skills. We give them all the support we can rally. We use the Agile tenants to drive our project. Yet, projects still go off-the-rails.

The reason? You cannot watch things to happen. What I mean here is that the act of watching does not make the outcome true. While many things do get better simply by measuring them and creating visibility, that is a temporary condition. Systems (and teams) always revert to their innate behavior.

To combat this we need to "make the plan come true." What I mean here is that a plan is just that, a PLAN. I would offer that many project "plans" are more the wishes of the people creating it. A real plan knows what it is trying to accomplish every step of the way. It is not a dream or an end state. It is something we can measure against and most importantly, make adjustments when we aren't seeing the results we expect.

Finding

5

Rework & Scope Creep

Rework & Scope Creep

The resigned acceptance by developers that a majority of their time will be spent in rework could be the biggest indicator that projects are not set up to succeed from the start. Without a clear and shared vision, it is inevitable that developers will need to build and rebuild which increases the project budget and extends the timeline.

80%	Admit that at least half of their time is consumed by rework
46%	State they are unsure of the details the business needs them to achieve with their project
23%	Feel business and IT always agree on when a project is fully completed

For a project to end on-time, on-task, and on-budget, rework needs to be minimized. To prevent rework, all team members need to understand the main causes of scope creep and rework and have strategies to avoid causing the extra work.

Strategies include providing access and openness for the types of communication that will prevent members from making individual decisions that conflict with the best choices for the success of the project.

Ask Before You Act

What is scope creep? When decisions made by any member of the project team add extra work to the project, the scope increases or creeps. Sometimes the simplest decision can increase scope and derail a project.

Let's look at an example. On a project, a developer thought having icons wiggle side to side when the mouse hovers over them would be a good addition. It was not in the requirements to have a wiggle, but it was also not in the requirements to not wiggle. The developer took 15 minutes to add the wiggle functionality. It worked, so the quality analyst sent it on to the business for user acceptance testing. When the business team member tested the page, they were surprised to find a wiggle and did not like it. The next meeting included a discussion of the wiggle and it was decided to remove the wiggle. The code went back to the developer and it was removed. It was then retested by quality and business.

The original 15 minutes of work ultimately added 4 hours of work to the project. If the developer had taken 5 minutes to ask if the business would like the wiggle added, the scope would not have increased and the developer would not have ended up recoding the page. During a project, hundreds of decisions are made opening the potential for good intentioned choices to increase scope and cause rework.

Teams with a strong culture of communication and consistent business involvement will have less "wiggles" in the project. Create a team norm of ask before you act to keep your project on the road to success.

Done. Done. Done.

The majority of people believe that IT and business both begin and end out of sync. Much of the inability to align on completion surrounds the understanding of bugs. A bug is any flaw in a software system, but who decides if it is a true flaw or a choice?

Most IT teams define bugs as critical, high, medium, and low with the level tied to the coding effort it will take to fix the bug. An item that will impact the entire system stopping it from working correctly will be a critical where a typo will be a low.

Business teams define issues by how they impact the customer or user. Agreement that an item that stops the system from working is critical is easy to accept; however, typos are rarely low to a business member. They may be easy to fix, but they are embarrassing to the business.

There will be a set of bugs that are a matter of choice. For example, if the user hits save and the page saves and takes the user to a dashboard page, this was a business choice made in requirements. If during testing, business decides they would rather have the user directed to another page, it would be logged in as a bug that could be low to high depending on the amount of occurrences and the effort.

Bugs that do not stop functionality need to be decided on very intentionally. Where the business may want the page change if it is 15 minutes of effort, they may not want to spend 4 hours on the change at this time. Teams with agreement on bug classifications will be likely to agree a project is truly done.

The Conclusion

Although most projects are considered by those surveyed to be **doomed from the start** that does not need to be the reality. By examining the 5 key findings closely looking for interconnected issues and opportunities for change, leaders can set up software projects with a collaborative view of success moving forward.

Take a hard look at your software development process and determine how you will facilitate changes to:

- Align your stakeholders from the start
- Articulate accountability, roles, and responsibilities
- Define requirements that are readable and actionable
- Establish practices for easy and continuous involvement
- Increase visibility and communication to reduce rework and scope creep

Acknowledge and validate the uncertainty felt by team members at the start of the project and openly commit to using these 10 success strategies to avoid repeating the problems of past projects. You can get your team to feel optimism and confidence about the next project.

Utilizing these 10 strategies for success, Geneca's success rate for software projects delivered on-target, on time, and on-budget is 97%. We continue to investigate issues on every project, determine cause and effect, and advance our practices. We can deliver your next project successfully or can help arm your team with the best practices to make project success a standard in your organization.

If you would like help ensuring success on your next software project, we can help. Contact Geneca for a free consultation today. We would love to discuss your company's needs and answer any questions you have so we can get you on the path to completing your project successfully.

CONTACT US