

Innovative Projects & Software Development: Is Failure a Risk You Have to Take?

Pre-analysis lays the foundation for a successful software project.

Tony Karrer, Ph.D.
CEO/CTO TechEmpower, Inc.

Abstract

The surprising fact is that more than 50% of software projects fail, and the number is even higher when an innovative concept or idea is being implemented. Yet, most of these failures could have been avoided had the software developers done the right things at the outset. In this paper I present the approach that TechEmpower uses to lay the foundation for a successful project. During the stage that I call pre-analysis, we define a project completely, using a “target process” that iterates until we understand what results we want and how we are going to get there. The keys to successful pre-analysis involve:

- Identify the client’s real motivation and goals
- Identify and attack risks
- Break the project into pieces
- Involve the right people
- Establish the right process for the project

While there is nothing mysterious about our approach, software developers often ignore these things or do them poorly, at great peril to their projects.

Introduction

Have you ever been exhorted to “fail faster?” I first encountered this idea some years ago, when Dr. George, then a well-known Los Angeles weatherman, suggested that success requires failure. To really succeed, he said, you must “increase your failures.” The concept was reiterated in a recent Fast Company article (issue 54, Jan. 2002, page 68): “So the only thing you can do is try to fail faster in order to move on to the next idea.”

That may be sound advice in some arenas, but software development isn’t one of them—not if you care about your career. While business philosophers may talk a good game about failure, the hard reality is that software failure is not rewarded. It’s quite the opposite.

Look up “failure” on Dictionary.com and here are the first two definitions you’ll find:

fail·ure, *n.*

1. The condition or fact of not achieving the desired end or ends: *the failure of an experiment.*
2. One that fails: *a failure at one’s career.*

What a perfect point. When it comes to translating an innovative idea into a working application, “failing faster” suggests a quicker demise to your job situation—if not to your career or even your company. But the surprising fact remains that *more than 50 percent of software projects fail*. And the more innovative the project, the more likely it is to fail. As long as you believe that failure is unavoidable, you are doomed to sometimes be part of projects that fail.

How can you turn the odds to your favor, especially when you’re looking to transform an innovative idea into a system that no one has built before? How can you become part of the favorable minority of projects that succeed?

There’s no silver bullet here, but doing the right things *before you start* is the best way to ensure success. In other words, before you’ve officially received approval for the project, you must establish what the project must accomplish and how you’re going to attack the work. The main purpose of this stage, which I call pre-analysis, is to define the project correctly. Once you have a project definition then the client decides whether to go forward with the project. Thus the basic goal of pre-analysis is to define the project correctly in order to help decision-makers decide if the project should go forward and to ensure downstream success.

In this paper, I discuss how to attack pre-analysis in order to achieve this goal. In particular, I look at where pre-analysis fits in to a project, how to define a project using a “target process,” and five keys of a successful pre-analysis:

- Identifying the real motivation and goals
- Identifying and attacking risks
- Breaking the project into pieces
- Involving the right people
- Establishing the right process

In many projects that have failed, you'll find that these keys were ignored or done poorly. This is especially true for innovative projects—projects where few, if any, systems exist to serve as precedents. Because such projects have been conceived independent of existing or well-known domains, they often are very challenging, and a rigorous pre-analysis is critical.

How Pre-Analysis Fits In

Pre-analysis occurs before the project has really begun. If you look at a classic waterfall lifecycle for software development, you will see that it has the following stages:

- Requirements and Analysis
- Design
- Implementation
- Unit test
- Integration test
- Delivery

Pre-analysis is a stage that occurs before official requirements gathering has started. Usually, in fact, it is only once we have completed the pre-analysis and the client has approved the project definition that we get the directive to go forward with the project. At that point the project really starts and we begin our development process.

Some software development organizations have different names for the pre-analysis stage – although there are also many terms out there that sound like pre-analysis that actually turn out to be Requirements and Analysis. Other terms that I've heard used to describe this same stage are “conception” or “concept exploration.” However, be careful, many vendors try to make their process sound unique and it can be hard to determine if they have a pre-analysis stage or if they are really talking about requirements gathering and analysis.

In the world of software development vendors and in-house development teams, pre-analysis is the first look at the project. Most clients, whether external or internal, come to you with a rough idea of the project that they want to implement. It's both a wonderful time and a horrible time. You try to spend enough time to figure out what the project really is—without spending too much time on a project that may never go anywhere. And when the idea is truly innovative, your job becomes that much harder.

What happens when a development team shows that a project concept has flaws or at least needs more homework before it's ready to go forward? The client is often unhappy. And there is no project for the development team to work on. No one seems to be happy. So, if anything, development organizations tend to avoid pointing out problems. And, of course, that defeats much of the purpose of pre-analysis—to help decision-makers decide if the project should go forward. Thus, pre-analysis is an exciting but challenging stage of any project. That said, let's consider how to define a project.

How to Define a Project

Again, the first part of a successful pre-analysis is to define the project correctly. To do this, you must understand what you are shooting for and how to get there. In this section I look at what a project definition is and at the target process that TechEmpower uses to arrive at the definition.

What constitutes a project definition?

When I say “define the project,” I mean define the following elements:

Motivation	Why is this project being initiated? Is it driven by a particular need or opportunity? At first the motivation is often something such as, “The business practice we currently use is horrible” or “We can’t get the information we need.” Later, the motivation is refined into business objectives such as saving time, reducing costs, or capturing new revenue.
Constraints	What are the limitations on time, budget, people, technology, systems, and environment? Are there hard end dates that must be met? What is driving these dates? What happens if those dates are missed? Are certain people required for the project? Are there limits on their availability? Does the software need to work with particular systems? What is the state of those systems?
Scope	What will the system do? At first we use a basic description of the system. Then we increasingly define the specific functions of the system. Who are the users? What will they be able to do? What business logic does that functionality imply? What data is stored? Keep in mind that we’re not yet at the requirements or analysis stage; we are looking for high-level answers.
Goals	How will we know if we have succeeded?
Risks	What are the potential pitfalls? What is being done to avoid them?
Process	How will we attack building the system and reduce risk?
Schedule and budget	Obvious what I mean, but hard to get right!

When the project revolves around an innovative idea, defining these elements becomes even more important. Here’s why. If the project is of a well-known type, e.g., a call center application, then you can pull from a wide range of examples and base technology. Defining this kind of project has its own challenges, but you are choosing from among known types of software systems and kinds of functionality. Few innovative projects, on the other hand, have an established base from which to start. Beginning with a blank sheet is nice in some ways, but it’s challenging. The picture can turn out to be many different things. This initial uncertainty makes having the right process for pre-analysis critical. So, let’s take a look at TechEmpower’s pre-analysis process.

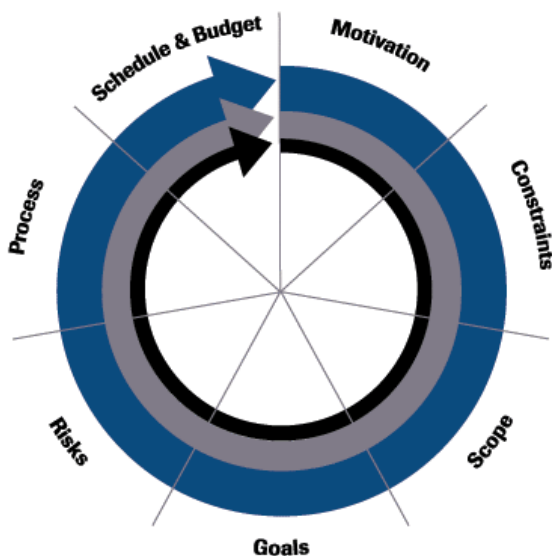
How is a project definition reached?

When you’re working on an innovative project, the process used to define the project is anything but straightforward. Classically, as you work to define a given element, you may pinpoint risks that force you to circle back and redefine the other elements of the project. To avoid those risks, you find yourself defining many subprojects, each with its own definition. Because you’re exploring different ways to define the project in an effort to find one that works, pre-analysis should not be treated as a simple linear task. You can’t just step through, define each element in turn, and end up with a correctly defined project.

Instead, at TechEmpower, we use what we call the “target process.” We start with a brief, simplified “core project definition” that addresses all seven elements at a fundamental level. Then we work outward, developing more detailed definitions for these elements.

We keep working outward until we have found a project definition that works or, failing that, we go back in to the core and redefine the project at a fundamental level. That risk—having to go back—is why we keep definitions simple and brief at first.

The following diagram depicts this iterative process.



Here's an example that shows how this process works. We suggest a simple core project definition, then proceed with pre-analysis. Well in to the pre-analysis, we discover that if we can meet a key release date, we'll capitalize on a particular market opportunity and win big in the eyes of key stakeholders. This fact causes us to redefine the scope to get to something that can be achieved in that timeframe. Redefining the scope probably also causes us to refine our definitions of motivation, constraints, goals, risks, process, schedule and budget. Whoops, that was all of them! This is why we keep things brief at first.

Because pre-analysis is often conducted before a project has been approved, one of the goals of pre-analysis is to balance two needs: the need to define the project correctly with the need not to spend too much effort on something that may not go forward. While you will still have a requirements and analysis stage to define the project more precisely; if you fail to take the time during pre-analysis, the expectations you set can head you right into failure.

This balance is even harder to achieve on innovative projects. Because this kind of project is not well defined, the natural urge is to define the project in greater detail. On the other hand, innovative projects are also most likely to be shot down because of the "No one else is doing that" factor.

At TechEmpower, we struggle constantly to find this balance, and we normally conduct a limited but intense form of pre-analysis. We spend a few hours with the client, asking and answering many questions. In fact, clients often feel like they are being bombarded by questions and challenged by the very people they came to for help: "Why are they questioning my definition?" Reassuring the client requires a gentle hand and an explanation of why we're asking so many tough questions. Usually, once the client sees that we're clarifying the project, the hard questions are accepted.

Five Keys of a Successful Pre-Analysis

Earlier I said that TechEmpower's project definition addresses seven elements of a project: motivation, constraints, scope, goals, risks, process, and schedule and budget. Because these elements are not entirely discrete (for example, the availability of resources can affect scope, risks, process, and so forth), I will not elaborate on each in turn. Instead I'll look at five keys we've found at TechEmpower to ensure that we define our projects correctly:

- Identify real motivation and goals.
- Identify and attack risks.
- Break the project into pieces to establish appropriate goals and minimize risk (really a subset of the first two, but important enough to call out!).
- Involve the right people.
- Establish the right process.

Let's take a look at each of these keys to pre-analysis success.

Identify real motivation and goals

Looking beyond the obvious

It is astonishing how often projects fail because the real motivation and real goals were not defined. What do I mean by that? At the start of a project, the internal or external client usually states that he or she "needs a system to do X." The first question that an in-house development team or a good vendor should ask is, "Why build the system and what do you hope to achieve?"

In drilling down to answer these questions, you often discover that it's not X that's needed after all. And, in fact, I can remember only a couple of cases where we ended up building exactly X. When you uncover the client's real need or rationale for the project and determine what constitutes success, the project almost always becomes something different—maybe a little, maybe a lot—and always becomes something better.

Case in point

A client came to TechEmpower with an innovative idea—a workflow system that tracks the responsibilities for marketing a product from the time the product has been approved until it's in the retail channel. That's a great start. Then we asked: Why build the system and what do you hope to achieve? The answer was that "everyone will use the system, it will reduce the time it takes to track products, and will save lots of money."

There are a couple of red flags in this definition, but I'll let them pass for a bit. Instead, let's delve further into the client's situation. As we worked with the client to identify the key stakeholders and determine how they would define success, it became clear that we had a bigger issue. There was one key stakeholder who would influence the rest of the stakeholders, but he had "serious concerns" about the project: Would the intended audience use the system? And would the company really reap benefits from the system?

The good news is that we had found an easy way to define motivation and success. What the project really needed to do was to convince this stakeholder that people would use the system and the company would realize a couple of specific benefits. This revelation caused

us to redefine the project at its fundamental level. Rather than build a system to do X, we would build a more limited system to test user buy-in and test benefits. In this case, we also redefined benefits: the original emphasis on saving time and hence money. We shifted to focus on reducing errors that had in the past resulted in heavy costs (missed product shipments in tandem with marketing expenditures).

From the standpoint of project scope, this project now turned into finding some low-hanging fruit. What were some inefficiencies and risk areas in the current process that we could address easily and that would provide immediate, obvious value? By defining a limited first version of the system, we recast the project under the “prove the proposition” umbrella and eliminated the risk of perceived failure.

What constitutes success

The case study I just discussed illustrates what I see as the two objectives of this part of the pre-analysis: (1) *to identify key stakeholders and evaluate how they will define success*, and (2) *to define goals that are achievable*.

Often the list of stakeholders is short. It can be as short as “my boss.” But keep in mind that many people will influence the stakeholders’ perceptions. Success is really “measured” by the perceptions not just of the key stakeholders but also of the project team, the parent organization, any partner organizations, and the customers or customer organization. It is important to understand how these influencers define success. When it comes to success metrics, perception is reality.

Knowing what constitutes success is especially difficult on innovative projects. Often there is more innate resistance among decision makers than on more typical projects. I call this the “no one else is doing it” factor. Also, because there are few precedents, it is harder to explain what the project is to decision makers and stakeholders. And they are more likely to have false expectations—an especially tough hurdle for project owners because they must get decision makers excited about their idea without overselling the project.

So how do you balance these challenges and set yourself up to succeed? I suggest that you:

- Sell each project based only on goals that are achievable.

The rule here is: Build only what is needed to achieve the desired result. Often, the first decision point or first unknown defines what you should build.

- Remember that every successful project is delivered:
 - Within the allocated period,
 - Within the budgeted cost,
 - At the proper performance or specification level,
 - With acceptance by customer/user,
 - With minimal or mutually agreed upon scope changes, and
 - With enough goodwill that you can use the client as a reference .

Red flags

Back to the red flags in the case study above. Two common goals—bad ones—are “everyone will use the system” and “it will save time and money.” Ensuring system usage is a tough goal for a project team. Testing usability, testing usage, and other such goals are

achievable, but rarely can you do more than design a system that you think people will use and that they say they will use. And no matter how good your system is, people may not adopt it. Always remember the Dvorak keyboard. It is far better than the Qwerty keyboard. Which one are you using?

The second red flag is the old “saved time equates to saved dollars” fallacy. I very often buy into “saves time,” but I am much harder to convince that a given system saves money. Saving time saves money only if people are let go—if you have ten people doing a job today, you will need only eight after the system is in place, and you plan to let go or reassign two of them.

The problem rears its ugly head when you start to hear this kind of calculation. Let’s say that the system, if adopted, will save everyone 20 minutes per day. That’s 16 hours a week for the ten users. Multiply that by some cost per hour and, poof, you have some big time savings. The problem is that if you still have ten people working, there is no actual cost savings, unless that extra time is turned into something valuable. Can the users service additional requests? That’s actually new revenue – not cost savings. Are they happier? Maybe we can make a case based on retention. Can you reassign one person to another job half time and lay off someone doing that other task? Then maybe we are back in the “saves dollars” realm. But the cold fact is that unless you’re talking headcount, we aren’t talking cost savings.

My point is that a project lives and dies with the expectations that are set for it. To set reasonable expectations, you *must* define realistic goals. The things you say you are going to achieve, you absolutely must achieve. To do this, you must define success based on the known outcomes. As the case study shows, it is acceptable to define success as proving or disproving usefulness, marketability, or return. And once you have defined the project this way, don’t let the conversation change to “building a runaway success.” If you define success based on things you aren’t sure of or can’t control, you lose your focus on the real goal, expectations get all out of proportion, and failure is inevitable.

Identity and attack risks

Areas of risk

On any project, you must do everything you can do up front to identify the real challenges, or “risk.” Risk is a measure of the probability and consequences of not achieving a defined project goal. In general, “risk management” means identifying likely sources of risk and developing, selecting, and applying appropriate options for handling risks. When a project is based on an innovative idea, managing risk is especially challenging because of the lack of precedents. Precedents reduce risks because some aspect of the technology has been proven, certain risks of adoption are known, you can point to examples, and so forth. For innovative projects, the best you can do is to point out similarities in software development in general and in innovative projects in particular. Thus, it is especially important to be diligent and thorough in identifying and analyzing risks.

Risks can take a wide variety of forms. At TechEmpower, we look for risks in these major areas:

- Schedule
- Organization and management
- Internal environment

- End-users
- Customer
- Third-party
- Requirements
- Product
- External environment
- Personnel
- Design and implementation
- Process
- Transitions

We created a detailed checklist of common risks in each area, and we use this tool during pre-analysis. Every software development organization should have something similar. (I'll take a closer look at ours in a future paper. If you're interested, please send an e-mail to info@techempower.com, and we'll send you a copy of the paper.)

While this process lets us find most risks, every project has a few unique risks. These we pinpoint through a gut sense and experience.

Case in point

A client approaches us to build an e-commerce system that ultimately will point customers to channel partners for fulfillment. Building this system requires integration with several other software systems, including one under development. Interfacing with external software systems is an obvious area of risk. Who owns these systems? Are they stable or in development? How well are they specified? Do they have established interoperability protocols? Is performance specified? Will they change during the project? How critical are they to the project? Can we define the project so that it can succeed even if this integration fails? Once these kinds of questions have been answered, we will have a pretty good assessment of the risk involved.

Assuming that this external system proves critical to our system's success, we must make sure that we don't fail because of deficiencies in the external system. Many software engineers seem to look at this kind of system and hope for the best. To be a happy developer, you have to be an optimist. You must feel that the very next change to your code will really fix that bug. That's why software developers are such lousy testers—they think the software works. They also have faith that this external software system will work and that their interface to it will work.

That's wishful thinking. The appropriate approach is to attack risks related to this external system right away. Maybe we need to define an initial, more limited project to test our planned interface with this system. For this project, success will be defined as proving or disproving that the external system can work with our future system—before we go and build our system.

Of course, no development team is ever that lucky. Usually, the external system is being developed in parallel with our system. Its specs are unstable. We doubt that the external system development team will make their delivery dates. Yet our system depends on its results. So we look for alternatives. Does our system really need real-time access to the data? Could we operate over an intermediate source that is refreshed less often? Are there other kinds of techniques to ensure hand-off such as message queues?

Again, the point of this is that to succeed, you *absolutely must* attack this risk early in the project. Assuming that the risks will take care of themselves may lead you down the path of the more than 50 percent of projects that fail.

Break projects into pieces

In both cases I've discussed, we took a similar approach: We started with a general, core project definition, but during the pre-analysis we found ourselves breaking the project into pieces. That's common at TechEmpower. I am a huge fan of this approach. Tackling smaller projects lets us achieve appropriate goals while reducing risk. In fact, I believe that the failure to do this is the most common reason that software projects go awry.

How do you break a project into the right pieces? There is no easy answer. But the rule I follow is this: Look for the unknowns (risks). In practice, I look for something that we aren't sure about. Then I define a smaller project that will give us insight into that issue.

These issues that you tackle may come in many shapes and sizes. If our goal were to gain user acceptance, the first project might focus on creating static mock-ups and getting users' feedback. If by doing this we proved the idea, we would then proceed to the next piece of the project. As I said earlier, that makes a lot more sense than building a system on the assumption that users will adopt it.

What if our goal were to try a new technology? The first project might focus on proving that technology. We might create one slice of functionality that might or might not represent a useful system. Either way, it would let us tackle the most onerous technical challenges. Basically, it's a technical proof of concept that is an inch wide and a mile deep.

Once you have identified the unknown, the basic definition of the project usually is obvious. Of course, you still must refine the definition using the iterative cycles I talked about earlier. Even if your project is a small piece of the original, it is a project in its own right and must be defined just as diligently.

Involve the right people

Involving the right people seems like an obvious step, but I'm surprised by how often it's missed. Who should participate when you define the project during pre-analysis? The normal cast includes the client project manager, a business analyst, and a project/account manager. We might also have key client stakeholders and key managers from the development organization. Who's missing?

In my experience, the most important person to have in the room at the start is the person who is going to drive the project's technical success. At TechEmpower, we call this person a systems architect. This person is the one who will be responsible for mapping an innovative idea from a specification into a living, breathing software system.

Why do I put the systems architect in the room from the start? Well, if he or she is going to be asked to build it, I want the architect to see the project from the start. I want to hear the person's concerns. I know that I must get the developers and the client in sync on the project definition. Of course, at the end, I'm going to ask the systems architect to define the schedule and budget. And he or she is going to have to live with it. The margin for error is pretty small. If the schedule or budget is off by 15%, the project will be pretty hellish. So, systems architects are highly motivated to be there and to get to a project definition that works.

Why do some people keep technical experts out of the room? I often hear that there's a certain fear that technical folks will say no. And sometimes they do. But they aren't saying no just to say no. They are identifying risks. They are telling you how budget and schedule are affected by scope changes. They are giving you a lot of information that you need but may not want to hear. I suspect that leaving out the systems architect lets people define projects in the way they want to—even if the projects are destined to fail. Of course, I may be biased: I used to be a systems architect who may not always have been invited!

Establish the right process

In defining the project, you must determine how you're going to attack the system. What will your process look like? While it is counterintuitive, innovative projects should not use innovative software development processes. Rather, innovative projects are best served by well known, proven processes.¹

Go to vendor Web sites and look at their branded process descriptions. They use obtuse language to describe something that has been researched, discussed, and used in software engineering for more than 30 years. The only major notable process innovation in the last few years has been extreme programming. And even there, besides the addition of programming teams and extensive peer review, the basic process is based on the same fundamentals – which is a good thing. Read between the lines and you'll discover that most of these vendors are describing a basic waterfall process. This process “flows” from stage to stage, going from requirements through to delivery. Except for a few variations, there's nothing new.

At TechEmpower, we distinguish ourselves not through our process per se but by how we *apply* our process. Because each project has subtle twists that require a slightly different approach, no single process is right. That's why we use what I call a “configurable” process.

Our configurable process, Empower™, is based on Barry Boehm's spiral lifecycle model (which itself is based on the waterfall model). In essence, we've defined a variety of elements that may be used in different ways or excluded entirely. The key is to decide what's called for given the situation, and to ensure that everyone understands the process that will be used.

Here's a simple example. Depending on the project, we may or may not put a prototype into the process. If we do, we make sure we've specified the right kind. There are many kinds of things that people call prototypes and they serve quite different purposes, for example, proving out technology, getting user buy-in, or evaluating usability.

The size of the project also affects how we define our process. We must follow the same basic steps, but for smaller projects, many steps may be abbreviated and/or combined.

The bottom line is that every project definition must include an accepted process. This process must be understood by the client and by the implementation team.

¹ I believe that projects based on precedents are often also best served by standard development processes. However, in some application domains where the problems are well understood, you can gain efficiencies. Thus, in such domains, there is a tendency to use processes that are more specialized.

Conclusion

Nobody sets out to fail; yet so many projects are doomed from the outset. Why? I think some projects fail because people fall in love with their concept and don't want to question it. Some fail because vendors or IT staff want the work and are willing to sign up for anything—including a leap into the uncharted waters of an ill-conceived innovative project—even if failure is pretty certain. And some projects, probably more than half, are lost before they start, because there was little or no pre-analysis.

Think about it: 25 percent of failed projects could have been saved with just a bit more effort up front. What a waste.

Your innovative projects can succeed, but you can't skimp on, or skip, the pre-analysis. If you are going to transform innovative ideas into outstanding software systems, you absolutely must make a rigorous pre-analysis part of your process.

About TechEmpower

We are the people to come to when you want to bring your innovative ideas to the Internet and can't afford to fail. Our 100 percent success rate means we're utterly reliable. We have deep technical skills, an efficient but flexible development process, and a straightforward style that makes us easy to work with. Our deep commitment to our clients shows in our ongoing relationships. Eighty percent of our revenue comes from companies we've already worked with. The other 20 percent comes from their referrals and recommendations.

To learn more about TechEmpower, contact us at (310) 524-1700 or via email at info@techempower.com.



TechEmpower, Inc.
898 N. Sepulveda Blvd.
Suite 300
El Segundo, CA 90245-2702

Phone: (310) 524-1700
Fax: (310) 524-1701

www.techempower.com