Pro Visual Studio Team System Application Lifecycle Management

ISBN-13: 978-1-4302-1080-1

ISBN-13 (electronic): 978-1-4302-1079-5

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Lead Editor: Tony Campbell
Technical Reviewers: Norman Guadagno, Dan Massey
Editorial Board: Clay Andres, Steve Anglin, Ewan Buckingham, Tony Campbell, Gary Cornell, Jonathan Gennick, Matthew Moodie, Joseph Ottinger, Jeffrey Pepper, Frank Pohlmann, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh
Project Manager: Beth Christmas
Copy Editor: Sharon Wilkey
Associate Production Director: Kari Brooks-Copony
Production Editor: Laura Esterman
Compositor: Patrick Cunningham
Proofreader: Linda Seifert
Indexer: Becky Hornyak
Cover Designer: Kurt Krames
Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders-ny@springer-sbm.com, or visit http://www.springeronline.com.

For information on translations, please contact Apress directly at 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit http://www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at http://www.apress.com/info/bulksales.

CHAPTER 1

Why Do We Develop Business Software and Systems?

Information is available at our fingertips all the time if we want it these days. I remember the days back when I was a teenager. Music and movies were, and always will be, two of my top interests. This obsession started during my teens, and I chased rare records of my favorite artists and hard-to-find horror movies everywhere. When I found a rare vinyl pressing of a precious record from the United States, for instance, I was ecstatic. Not to mention my emotional turmoil when I managed to purchase a Japanese edition of the same record. Those days I wrote snail mail asking for record mail-order catalogs from all over the world, based on ads in magazines such as *Rolling Stone* or *Melody Maker*. After carefully considering what I wanted to purchase, I wrote down my purchase order, enclosed crisp bills, and sent a letter with my order inside. Then came the long wait for the package. And believe you me, this wait could be long indeed. These days I just go on the Internet, check some sites, and purchase what I want directly by using a credit card. The stock of many sites is so huge compared to what they were in my teens, and I can usually find what I want very quickly. In a few days the package comes, and I can start using the things I bought.

We communicate differently as well. Sites such as MySpace and Facebook have generated massive followers, not only by the early adopters of technology, but by our societies as a whole. MSN Messenger, mobile phones, Short Message Service (SMS), and other means of communication practically have exploded, at least in the parts of the world where the infrastructure for this is available.

Development teams in organizations also use new collaboration tools such as Visual Studio Team System, the focus of this book. *VSTS*, as it is generally called, is an Application Lifecycle Management (ALM) platform tying together a company's business side with its information technology (IT) side. *Application Lifecycle Management* itself is, briefly, the process an organization can use to care for an application or software system from its conception to its retirement. ALM is the glue that ties the development processes together and defines the efforts necessary to coordinate the process. You will see more about this in Chapter 2.

With the new opportunities organizations have to do business, much has changed in the world for us, including the reality for our companies. They now have to deal with a global environment presenting both opportunities and challenges. This means that business has changed and still is changing at a rapid pace. This is why we need to be clear on why we develop business systems and software.

I know that in my early days as a developer and system designer, I did not think about business value much, if at all. Those days I was much more interested in cool technology and clever ways of solving technical and logical problems. And I definitely was not alone in doing so.

During my university studies at the University of Gothenburg in Sweden, nothing was ever mentioned about this either. We learned C and C++, some Visual Basic for Applications (VBA), and perhaps Structured Query Language (SQL) and how to use our skills at writing code. Nobody said anything about *why* we should write that code or *why* we should learn SQL and all those other things. We never were taught the connection between the requirement of the software and the final value it should produce. This is an essential issue in development for an enterprise. We need to make sure that our development turns business requirements and business needs into business value. That is the whole point of business software.

I hope this attitude and focus has changed since then, but when I meet people fresh out of school, I often get the feeling that nothing much has happened. Sure, these recent graduates are very good at technology and programming, but they seem to lack a fundamental insight into why we write business software. So shouldn't the school put more emphasis on the reasons for the existence of business software? Wait! Don't answer yet. Read some more of this chapter before you do.

Understanding the Cornerstones of Business

First let's define the term *business*. What do we mean when we talk about this concept? After we agree on this, we can reach an understanding of what *business software* is so we don't talk about two different things here. When I discuss business in this book, I am talking about not only the commercial part of the company, but all the functions in the company. This means that business software is intended not only for e-commerce, but for all the things going on in an enterprise.

There are three cornerstones in business system development that are important:

- Processes
- Business rules
- Information

These three are dependent on each other. One of my coworkers makes an analogy with the human body. If the processes are the muscles of our company and the rules are the brain and nervous system, we can say that the information can be seen as the spine. Not one of them could be functioning without the other.

Processes

A company uses different *processes* to support its business. For developers, project managers, software designers, or people with other roles in a development project, it is so easy just to focus on the development process. We are often interested in development processes such as the Scrum process or the Extreme Programming (XP) process. The business people mostly focus on the business side of course, and have no interest in learning about the development process.

Processes in an enterprise are so much more than just development. A large company needs processes for procurement, sales, manufacturing and so on, and the development process is just one of them. The other processes are needed for the company to function and live on.

Obviously, business processes are valid not only for commercial companies but for all organizations, including those in the public sector. According to Sundblad & Sundblad, a typical company uses somewhere between 10 and 20 comprehensive and dominating processes to support its functions.[1]

SCRUM, XP, AND RUP

For those of you who don't have the full picture of what SCRUM, XP, or Rational Unified Process (RUP) are, I will cover them in Chapter 4. For now, it suffices to say that all three are development process models we can use for controlling our development efforts in projects.

Wikipedia (`www.wikipedia.com`) has the following short description of the three processes:

Scrum *is an iterative incremental process of software development commonly used with agile software development.*

Although Scrum was intended to be for management of software development projects, it can be used in running software maintenance teams, or as a program management approach.

Scrum is a process skeleton that includes a set of practices and predefined roles. The main roles in scrum are the scrum master, who maintains the processes and works similar to a project manager; the product owner, who represents the stakeholders; and the team, which includes the developers.

During each sprint, a 15–30 day period (length decided by the team), the team creates an increment of potential shippable (usable) software. The set of features that go into each sprint come from the product backlog, which is a prioritized set of high-level requirements of work to be done. What backlog items go into the sprint is determined during the sprint planning meeting. During this meeting, the product owner informs the team of the items in the product backlog that he wants completed. The team then determines how much of this they can commit to complete during the next sprint. During the sprint, no one is able to change the sprint backlog, which means that the requirements are frozen for a sprint.

Extreme Programming (XP) *is a software engineering methodology (and a form of agile software development) prescribing a set of daily stakeholder practices that embody and encourage particular XP values. Proponents believe that exercising these practices—traditional software engineering practices taken to so-called "extreme" levels—leads to a development process that is more responsive to customer needs ("agile") than traditional methods, while creating software of better quality.*

The Rational Unified Process (RUP) *is an iterative software development process framework created by the Rational Software Corporation, a division of IBM since 2003. RUP is not a single concrete prescriptive process, but rather an adaptable process framework, intended to be tailored by the development organizations and software project teams that will select the elements of the process that are appropriate for their needs.*

Business Rules

The second cornerstone is the *business rules* the organization needs in order for it to function well. The business rules tell us what we can and cannot do in the company. They also tell us what we *must* do. If we compare the processes to the muscles of our body, we can say the rules are equivalent to our brain and nervous system—that is, the things controlling our actions and deeds.

1. Sten Sundblad, "SOA and Business Processes," 2006, `http://academy.2xsundblad.com`.

Information

A third cornerstone of any company is its *information*, that is, information about the company and what is going on in it. For example, we can have all customer information, order information, product catalog, and so on here. Without access to relevant information at the correct time, the business will quite simply not function. Consider this example: it will be impossible for a company to sell any of its products if it has no information about which products it has or what price they sell for.

Understanding the Need for Business Software

So to get back to the question about business systems and software: the reason business software exists is to support the business. Business software should take the business needs and requirements and turn them into business value through the use of business software. Application Lifecycle Management is the process that can help us deliver this business value. And if we IT people do a poor job of building this kind of software or systems by having a broken ALM process, the business will obviously suffer.

This is the reason I think we need to have the question about why we develop business software and business systems in mind all the time (no, you do not have to think about it in your free time, even though your manager probably thinks so). We do not write software for an enterprise to fulfill our technological wishes alone; we write it to make the business run smoother and create more value (see Figure 1-1). This does not, however, make it less cool or interesting to learn new technology or write smarter code. These are important parts of any software or system, fortunately.

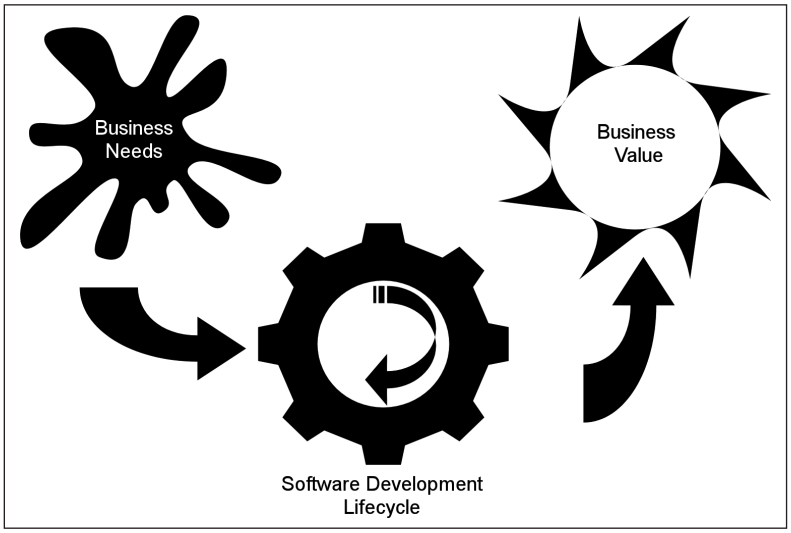

Figure 1-1. *The reason we write business software is to turn business needs and opportunities into business value.*

To furthermore stress the importance of the reason we write business software, we can take a look at a 2005 survey of department managers by IDC, a part of the International Data Group in Sweden.[2] In this survey, department managers were asked to indicate their most important IT priorities. Most of the managers—36.2 percent—wanted programs and services that were better integrated with the company's business processes. This was closely followed by a wish for better access to relevant information, at 35.2 percent. The third priority was better systems for cooperation and communication, with 31.6 percent.

These results definitely tell us that we IT people need to focus more on business value in the systems we develop. We will come back to other implications of this later in this chapter as well as in Chapter 3.

Today's Business Environment and the Problems We Face

With the new opportunities organizations have for business these days, much has changed in terms of the realities they face:

- Companies now have to deal with a global environment presenting both opportunities and challenges. A global way of doing business means competition can come from all sorts of places. Low-cost countries such as China and India can offer many of the same products and services as high-cost countries. This is a great challenge for development organizations all over the world. Consulting firms are opening development shops in low-cost countries, and other companies use these services they provide. An organization's internal development organization may also see their work move to these countries. So no matter where we work, this fact affects our jobs and us, and competition is fierce. In order for us to handle this situation, it is essential to have control over our ALM process. Through this process, we find the support for collaboration between the dispersed teams we see these days, which can give us the competitive edge we need to face competition from others. We need to automate and fine-tune the work process (read: ALM process) we use in our organizations so that we can face challenges, keep costs down, and win deals.

- This new reality has forced businesses to become more agile, ready to transform quickly to gain competitive advantages. This obviously affects the way we must architect and write business systems as well. In the ALM process, these topics are addressed and can help us achieve agility.

- Communication has also become more complex and different from before. Production of products and services is spread over the world, and gone are the days when an industrial plant supplied everything for a company. For us in IT, this means that software development has moved to countries such as India or China and we need to handle this somehow. This is quite a challenge. Just consider the potential communication problems in a company with offices or manufacturing spread across the globe—not to mention problems with time and cultural differences.

---

2. Sten Sundblad and Per Sundblad, "Business Improvements Through Better Software Architecture," January 2007, `http://msdn.microsoft.com/en-us/library/bb266336.aspx`.

As you can see, this means that business processes can (and do) change rapidly. Hence the supporting IT systems must also be ready for quick changes. If we do not have systems that allow this, business will suffer. This is one of the main reasons ALM tools such as Visual Studio Team System have emerged. Without an effective development process tied closely to the business side and supported by a set of tools, we will run into problems and risk being left behind by competitors already using such tools. And it is not only the ALM tools that are important; we need to consider the whole ALM process as well, including the way we run our development projects.

Project Health Today: Three Criteria for Success

What do we mean when we talk about project health? How can we measure this? Many surveys indicate the same criteria for success (or failure, if you are so inclined). Let's take a closer look. There is slight variation, but these three can be said to be the main criteria:

- Project delivered on time
- Project delivered on budget
- Project scope fulfilled

Let's discuss these three a bit. Is it reasonable to use these criteria to evaluate project success or failure? I am a bit skeptical and will explain why.

Projects Delivered on Time

In traditional project models, a lot of effort is put into time estimates based on the requirements specifications. This way of estimating was (and still is) great when estimating construction of buildings, roads, and other traditional engineering efforts. These are the projects that traditional project management wisdom comes from.

Such projects are far more static than most software development projects. The engineering discipline is also rigorous in its requirements management process, which helps a lot. You don't see as many changes to requirements during the process, and the ones that occur go through a comprehensive change request process. Many companies use CMMI to improve their process and thus be better at controlling projects. *CMMI*, which stands for *Capability Maturity Model Integration*, enables an organization to implement process improvement and show the level of maturity of a process.[3]

CMMI can be used to guide process improvement across a project, a division, or an entire organization. The model helps integrate traditionally separate organizational functions, set process improvement goals and priorities, provide guidance for quality processes, and provide a point of reference for appraising current processes.

Based on my experience at the Swedish Road Administration (SRA), where I have been for the past seven years, design risks when building a road, for instance, are pretty low, design costs are small, especially compared to building costs, and so on. Here you set the design (or architecture) early in the project based on pretty fixed requirements. From this, you can more easily divide the work into smaller pieces and spread them elegantly across your Gantt schema. This also means you can assign resources based on a pretty fixed schedule. Another

3. Software Engineering Institute, "What Is CMMI," `www.sei.cmu.edu/cmmi/general/index.html`.

benefit is that project governance will be easier because you can check off completed tasks against your Gantt schema and have better control over when tasks are completed and if there is a delay or lack of resource during the process. On the other hand, if you fail an engineering project, lots of money has been wasted, and in the worst case, somebody has lost their life because of poor control of the process.

When it comes to more-complex buildings, such as a new tunnel the SRA built in Gothenburg and which was opened in 2006, things are a bit different. A tunnel of this magnitude was not something that the construction companies built everyday. This made it harder for the team to estimate time and money for the tunnel. In this case, the opening of the tunnel was held at almost the date estimated from the beginning. It differed by a couple of months as I recall, which must be considered well done because the whole project took more than five years to complete. The reason for this was that everything from risk management to change requests, and all construction-related tasks, were handled with rigorous processes.

I think that one thing that greatly differs between construction processes and software development processes is that construction workers know that if they make a mistake, somebody might get hurt or die. We in the software development industry tend not see that connection clearly, as long as we don't work with software for hospitals, or other such areas. This could be one reason that we haven't implemented better processes before.

In his book *Software Engineering with Microsoft Visual Studio Team System* (Addison-Wesley Professional, 2006), Sam Guckenheimer calls this way of breaking down the project into work tasks a *work-down approach* because it is easy to see this as a way of burning down a list of tasks. This method of managing projects, he argues, is great for projects with low risk, low variance, and a well-understood design. In the IT world, you can see that implementations of standard products could benefit from this model. In such projects, you can do some minor customizations of the product, and the development effort is pretty small, especially compared to the effort put into business analysis, testing, and so on.

When it comes to IT projects with a lot of development efforts, things change. The uncertainty in the projects increases because so many things can occur that we cannot know about from the beginning. This inherent uncertainty in complex IT projects makes it hard to estimate tasks in a correct way early on. Things happen along the way that throw aside earlier estimates.

Considering this, is it then realistic to measure a complex IT project against planned time? To really know how projects are doing, we might want to consider whether this is just one of the measurements we can use.

Projects Delivered on Budget

Much of the same reasoning in estimating the time of a project applies to estimating costs, because so much of the cost is tied to the people doing the work. But cost involves other factors as well. We have software costs, office costs, and other costs, but these are often easier to estimate than development costs, because they are fixed for the office we use for development. We can put a price tag on a developer, for example, based on that person's salary, the cost of leasing of a computer, and the cost for licenses needed. This can be done in advance, and we then know that one developer costs a certain amount of money each day. Development cost, on the other hand, is harder to determine because it is harder to estimate the complexity of the system beforehand. The changes we encounter are hard to estimate in advance and hence the cost is hard to estimate as well.

Project Scope Fulfilled

This is also a tricky criterion, because what does *scope fulfillment* really mean? Does it mean that all requirements set at the beginning of a project are fulfilled? Or does it mean that the system, when delivered, contains the things the end user wants?

Most surveys seem to take the traditional approach: requirements are set early and never change. But what about the problems we saw with complex projects earlier? Can we really know all the requirements from the start? Something that I think everybody who has participated in a software project can agree on is that requirements change during the course of the project, period!

It might very well be that all the requirements that we knew about from the start have been implemented, but things have changed so much during the project that the users still do not think the project has delivered any value. The project could be seen as successful because it has fulfilled its scope, but is it really successful if the users do not get a system they are satisfied with? Have we really delivered business value to our customer? That is what we really should have as a goal.

All through the development process, we need to identify the business value that we will deliver and make sure we do deliver it. The business value might not be obvious from the start of the project but should be focused on during the process. A good development process and ALM process can help us achieve this.

Let's now take a look at what factors influence project success.

Factors Influencing Projects and Their Success

As I have said, today's enterprises face a lot of new challenges. Let's go through some of these in more detail, starting with the most important one based on the surveys presented earlier but also on my own experience.

The Gap Between Business and IT

Let's start with the biggest issue, which I mentioned before, as you may recall. IT managers' top priority was better integration between the company's business processes and the supporting IT systems. There seems to be quite a gap between the IT side and the business side, making it difficult to deliver software and systems that really do support the business. IT managers may focus on security, scalability, or availability instead of on supporting the business processes. These are of course important as well, but not the only issues IT should focus on. Business managers, on the other hand, may have trouble explaining what they want from the systems. This gap poses a great threat not only for projects but also for the entire company. We will discuss this later in Chapter 3 because I think this is an important topic that needs to be solved somehow.

The Development Process

Let's continue with the development process. Can this affect success? Obviously, it can. I have seen organizations that have spent lots of effort, time, and money on developing a good process. These organizations have trained both project managers and participants in RUP, XP, or any other development model they chose, and you would think all was dandy. Still, projects seem to suffer quite a lot. One reason for this might be that when a project starts, it is hard to follow the process. RUP, for instance, is often said to be too extensive, with many documents

to write and milestones to meet. Let's face it—even Ivar Jacobson himself seems to think this, considering his latest process development. If the process is seen as a problem or a burden, project members will find ways around it, and the money spent on training and planning will be wasted. The process may also be hard to implement because the tools have no way of supporting it. If we cannot integrate our development process into the tools we use to perform work, we most likely won't follow the process. Using the process must be easy, and the tools should make the process as transparent as it can be, so that we can focus on work but still follow the process.

When I travel around Sweden talking to organizations about VSTS and ALM, I usually ask what development process the organizations use. Often the answer is "the chaos model," or "the cowboy model," meaning they use a lot of ad hoc, often manual, efforts to keep it all going. Many say this is due to an inability to integrate their real development model into their tools, but others just had not given it a thought. These companies had hardly considered using any structure in work and if they had, the thoughts had often stayed in the heads of the developers (who quite often are the ones longing for a good process) or managers. Maybe a decision had been made to consider training staff in a model, but the company had never gotten around to it. No wonder these organizations experienced lots of failed or challenged projects.

Speaking of processes. I would say that not having a flexible development process (more on these processes in Chapter 4) most definitely will affect project success. Because business is sure to change during a development project, we need to be flexible in our process so that we can catch these changes and deliver business value in the end. I had a discussion with one of my customers about this some time ago. Most customers agree that there must be a way to make it easier to catch changes to requirements and make the system better reflect reality during the project. Otherwise, the perceived value of the project will suffer. But in this case, the IT manager was almost scared to even consider this. He argued that all requirements must be known at the start of the project and that they must remain static throughout the project. He thought the project would never reach an end otherwise. Not one single argument could break down his wall. He wanted to run his company's projects by using the Waterfall model as he always had. And still he kept wondering why projects so often ended badly.

Geographic Spread

With development spread across the globe and outsourced development, running projects can be very hard indeed. When development teams in a project are geographically separated, means of communication between them must exist and function seamlessly. For example, how can we share project status in an effective way, so that everybody can see how the project is going? How can we get a good, robust version control of source code and documents to function when we have long distances between teams? How can we catch changes to requirements when users, developers, and decision makers are separated?

The complexity in this takes its toll on both project managers and the projects themselves. Tools and processes must be in place supporting the project teams. Obviously, both time and cost can be severely negatively affected by this fact. If we do not catch requirements changes, fulfillment of project scope (or the perceived added value of the project) will most likely suffer as well, making the project one of the challenged, or in worst cases abandoned, in the statistics.

Synchronization of Tools

Numerous times I have seen situations where a developer (or other team member) must use several tools to get the job done. There is one tool for writing code, one for bug reporting, one for testing, one for version control, one for reports and statistics, and so on. I am sure you recognize this as well. The coordinating effort to keep all information in sync between these systems is immense. Not to mention the difficulties of implementing a development process in all of them, if this even is possible in all systems.

Resource Management

What about *Project Portfolio Management*, or *PPM*, as it is also known (see Figure 1-2)? Keeping track of all running projects and their resources can be a considerable problem for enterprises. The investments in applications and projects are enormous, both if you look at them from a financial perspective or from a human capital perspective. PPM helps organizations balance the costs and values of IT investments so they can achieve their business goals.[4]

Forrester says, "PPM provides a fact-based process for evaluating, prioritizing, and monitoring projects. PPM unites the process of strategic planning, resource and budget allocation, project selection and implementation, and post-project metrics."[5]

This basically says it all about what issues are covered by PPM.

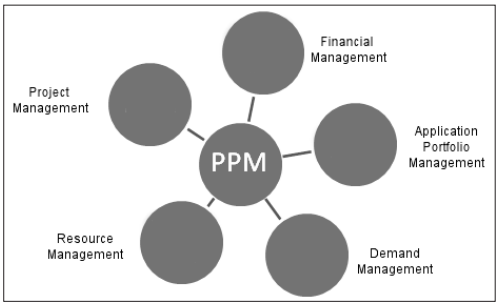

Figure 1-2. *ALM and PPM*

We can also see that a great portion of IT investments are focused on custom application development. If we cannot manage the resources we have at our disposal, the projects will most definitely suffer. We need to know, for example, that Steve will be available at the time he is needed in our project according to our project plan. If he is not, the schedule might have to change and the project most likely will be affected by both time and cost increases. To make matters worse, tasks depending on Steve's work might suffer as well. I would say this issue is one of my customers' top priorities now. A lot of the questions I get when speaking about VSTS implementations concern resource management integration with VSTS.

---

4. Kelly A. Shaw, "Application Lifecycle Management and PPM," June 2007, `www.serena.com`.
5. Craig Symons with Bobby Cameron, Laurie Orlov, Lauren Sessions, Forrester Research, "How IT Must Shape and Manage Demand," June 2006, `www.forrester.com/Research/Document/Excerpt/0,7211,39660,00.html`.

Project Size

Project size could also affect the outcome of the projects. This is perhaps no surprise, because complexity usually increases when project size increases. It is hard to manage a project that has many people involved or a long timeline. If you combine a large project size with geographically spread project teams or members, keeping it all from falling apart becomes harder, and it will be harder to foresee everything that can happen. This obviously does not mean that only the large size and geographic spread create these challenges; the combination of several risks in software development makes it hard to anticipate what might happen.

As you can see from all this, and everything you know for yourself, a lot of things can affect the outcome of projects. It is a complicated situation and because lots of money is involved, we have many considerations during the process.

What does research say about project success? When discussing this topic with coworkers, many have views and opinions, but not that many can reference research directly. They seem to argue on a gut feeling. So let's take a look at what the research indicates.

Project Success in Research

This section covers some of the research on project success over the years. You will see a well-known report from the Standish Group as well as some disagreement to this report. You will also see what the Swedish IDG has found and some research from ACM

The Standish Group

The Standish Group performs a survey on a regular basis on the performance of IT projects in the United States and Europe. The first report in 1994 was quite famous. It showed that many IT projects were cancelled or severely challenged. Since then, the Standish Group has performed the survey several times.

In 2006, the figures looked like this:[6]

- 46 percent of projects were challenged (53 percent In 1994).
- 19 percent of projects failed (31 percent in 1994).
- 35 percent were successful (16 percent in 1994).

Figure 1-3 presents these figures in graph form.

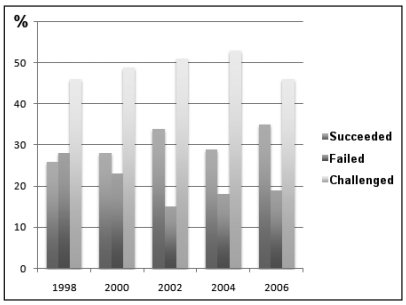

Figure 1-3. *The Standish report from 2008 shows figures from 1998 and forward.*

6. The Standish Group International, "Chaos Summary 2008."

The figures have improved a little over the years, but still 65 percent of all projects seem to be unsuccessful in some way. But to lump failed and challenged IT projects into the same bucket is not quite correct. Just because a project is challenged does not mean it has not added value to the company. A project might be late or overrun its budget but still deliver great value to the company, which might make it a well-received project anyway. Keep this in mind when you see figures like the preceding ones mentioned. A little perspective on the figures does not hurt.

Before we leave the Standish report, let's look at what it says about the reasons for project success. These are interesting no matter whether you believe in the actual success figures of projects or not. Here are the Standish Group's top ten reasons for project success:[7]

1. User involvement
2. Executive management support
3. Clear business objectives
4. Optimizing scope
5. Agile process
6. Project manager expertise
7. Financial management
8. Skilled resources
9. Formal methodology
10. Standard tools and infrastructure

Pretty interesting reasons, don't you think? Keep them in mind when you read about Visual Studio Team System later in the book (Chapter 6). We will also come back to some of them later in this chapter.

Robert Glass

The figures from the Standish Group have been challenged by other researchers. Robert C. Glass wrote an interesting article that questions where the data of the Standish report really comes from.[8] He also questions the methods used by the Standish Group.

Glass asks us to stand back and ask ourselves two things:

- Does the report represent reality?
- Is it supported by research findings?

Glass asks these questions because many other academic studies and guru reports in this area reference the Standish report from 1994. However, these other studies do not present

---

7. Deborah Hartmann, "Interview: Jim Johnson of the Standish Group," 2006, `www.infoq.com/articles/Interview-Johnson-Standish-CHAOS`.
8. Robert C. Glass, "The Standish Report: Does It Really Describe a Software Crisis?" August 2006, *Communications of the ACM*.

much new material to support the old findings, according to Glass. Another problem is that the Standish Group does not describe its research process or indicate where their data came from so that we can discuss its validity. This is of course a huge problem.

IDC

IDC recently published an article about the success (or perhaps failure) of Swedish IT projects. According to Sundblad & Sundblad, the article stated that as many as 82 percent of the IT projects were not finished to the customers' satisfaction.[9] This was the second time that this survey was carried out. The first time the survey was carried out in 2005, this figure was 72 percent, which means things have gotten worse. The survey has been carried out through the Swedish web site Projektplatsen.se on public as well as on private companies.

Thirty-four percent of the public IT managers stated that they did not get the features they wanted, and 20 percent of the private managers said the same thing. Forty percent of the IT projects in 2006 were delivered late, and the same number goes for projects running over budget. These figures were slightly lower than during the first survey.

Probable causes for these poor figures were believed to be the following:

- Internationalization of the business (more competition, to put it simply)
- The industry boom
- Higher demand for better efficiency in the public sector

The projects and customer satisfaction were evaluated on the following criteria (I will discuss the relevance of such criteria(s) later, so please bear with me a while more).

- Projects delivered on time
- Projects delivered on budget
- Competence of project provider
- Fulfillment of required functions
- Increase of efficiency
- Communication in the project

These figures are embarrassing and a poor grade for Swedish IT suppliers. What is even worse is that these figures seem to be similar internationally if we trust the Standish Group report.

Size and Volatility Survey

I want to address another survey before moving on. This survey claims that 67 percent of all projects are delivered close to budget, schedule, and scope expectations—quite the opposite of the preceding findings.[10] Of the 412 UK project managers in the study, they on average overshot budget by 13 percent, schedule by 20 percent, and underdelivered on scope by 7 percent. These figures are considerably lower than the Standish Group findings as well as the Swedish IDC findings.

---

9. Sundblad, "Business Improvement Through Better Software Architecture."
10. Chris Sauer, Andrew Gemino, and Blaize Horner Reich, " The Impact of Size and Volatility on IT Project Performance," November 2007, *Communications of the ACM*.

Table 1-1 shows the performance variance of the five types of projects defined in this study. Table 1-2 shows the size characteristics of these project types.

Table 1-1. *Performance Variance*

Performance Variance	Type 1: Abandoned Projects	Type 2: Budget Challenged	Type 3: Schedule Challenged	Type 4: Good Performers	Type 5: Star Performers
Schedule	n/a	+34%	+82%	+2%	+2%
Budget	n/a	+127%	+16%	+7%	+2%
Scope	n/a	–12%	–16%	–7%	+15%

Table 1-2. *Size Characteristics*

Size Characteristics	Type 1: Abandoned Projects	Type 2: Budget Challenged	Type 3: Schedule Challenged	Type 4: Good Performers	Type 5: Star Performers
Median budget	£1,000	£625	£500	£450	£2,000
Average budget	£24,232	£8,978	£12,513	£6,106	£12,119
Schedule	798	557	212	89	170
Budget	17.4	20.0	13.0	11.2	15.3
Scope	35.7	17.7	12.9	7.3	9.8

These figures are interesting, but the following findings are even more so. What the following three figures show is that size matters. Figure 1-4 shows that the size of the project directly affects the outcome. The more effort in terms of person-months we have in a project, the greater are the chances of failure or low performance.

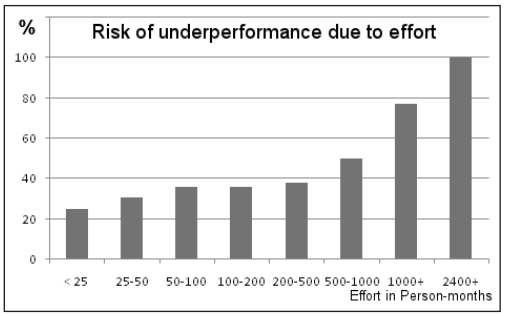

Figure 1-4. *Person-months*

Figure 1-5 shows that the duration of the project also impacts performance.

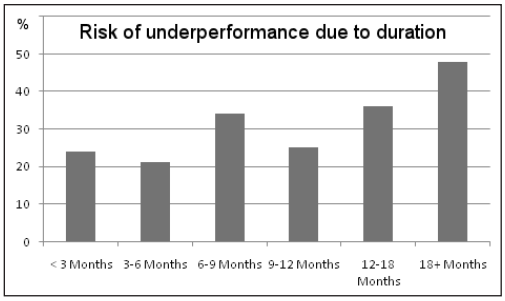

Figure 1-5. *Project length*

The longer the project, the greater the risk. Note that it seems like the greatest risk comes when the project duration reaches over 18 months.

Figure 1-6 illustrates the risk of project failure based on the size of the team.

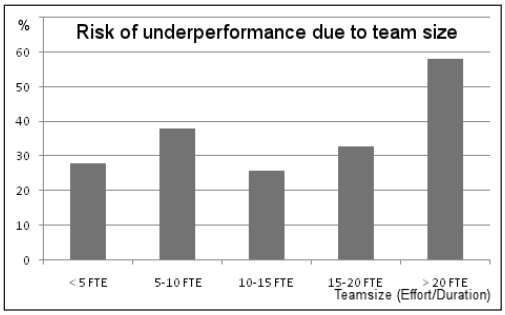

Figure 1-6. *Team size*

The risk seems to be the greatest when we have a team size of more than 20 people. Below that, we see that risk is pretty much the same.

To be counted as a low performer, the following three project categories applied:

- Abandoned
- Budget challenged
- Schedule challenged

These criteria are also found in the Swedish IDC survey presented earlier. The size and volatility survey further showed that changes to project targets affected the success rate as well. The more changes, the bigger the chance of being a low performer.

Conclusions

So what do all these figures tell us? Well, we can clearly see that projects are not carried out in the most optimal way. There are still too many challenged and abandoned projects in this day and age. No matter which survey we choose to believe, the figures are worrisome.

If our projects are not successful, the whole ALM process will suffer. The reason why we should take control of the ALM process is that we can deliver better projects; having an overall process helps us. And with this process comes a mindset focused on the application from its birth as a business need to the delivered business value.

Measuring project success or failure is complicated. We need to take this into consideration when we read surveys stating that this or that many projects succeed.

The importance of an overall ALM process is that it can help us control the project outcome better in the end, enabling us to deliver true business value.

The work-down paradigm mentioned earlier has a widely accepted *iron triangle* showing the relationship between time, resources, and functionality (Figure 1-7).[11]

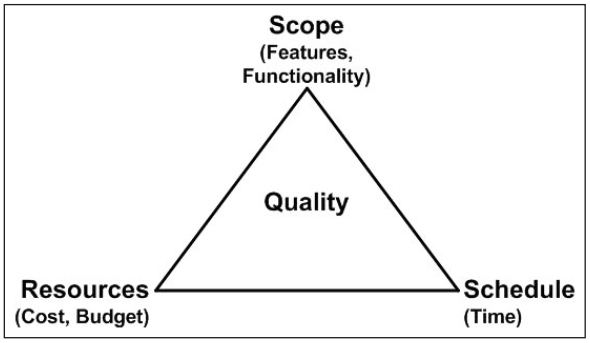

Figure 1-7. *The iron triangle*

In this we can see quality as a fourth dimension, giving us a tetrahedron. In this model, the relationship is fixed between the faces of the tetrahedron. If you stretch one, at least one other needs to be stretched. If more resources are needed, for example, you might need to stretch time as well, and so on. With complex projects, the scope usually changes unpredictably during the process and suddenly we might need to add more resources, which obviously affect the schedule.

We need to reflect on the results of surveys before we take them as the truth. This does not mean that the surveys I have referred to are without value. They definitely tell us something is wrong with the way we perform IT projects today. Why do so many IT projects fail? Have we not learned anything from the past years? The IT industry has been around for quite some time now, and we should have learned something along the way, shouldn't we?

Could it be because the problems we need to solve just keep getting harder and harder? Or is it so hard to estimate and plan for projects that we can only take an educated guess at the beginning of a project? If the latter is true, why do we still measure a project's success based on time, money, and requirements fulfillment? Maybe we should just shift our focus to business value instead? If the business value is greater than the cost of implementing the solution, time and money are usually of less importance.

11. Sam Guckenheimer and Juan Perez, *Software Engineering with Microsoft Visual Studio Team System* (Addison-Wesley Professional, 2006).

IT Budget Spending

In many companies that I have seen, the better part of the IT budget is usually spent on operations and maintenance and not on new development. The implications of this are that organizations have fewer possibilities to enhance their IT systems, and instead just spend the budget on keeping the systems alive. This could be a problem because the IT budget is needed to develop systems for meeting changes in the business processes.

Development vs. Operations

In 2006, the Corporate Executive Board (CEB) published the results of a survey concerning IT spending.[12] The survey found that of all IT spending in 2006, 30 percent related to IT system development. That might sound like a big part of the cake, but consider where the rest of the money is spent: 70 percent of an IT budget is spent on operations and maintenance (Figure 1-8).

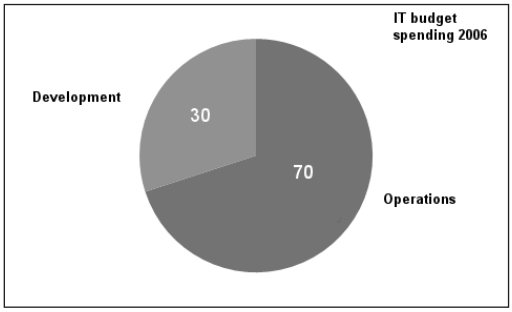

Figure 1-8. *IT spending in companies*

These figures are confirmed time after time in my discussions with customers. One of my coworkers told me that when he worked as an IT manager at a large Swedish car manufacturer, the figures applied as well.

Is it really rational to spend 70 percent on operations and maintenance? Wouldn't it be more interesting to try to switch these figures around? Imagine the possibilities of adding value to an organization with so much more money to spend on IT systems. And think about the cool new features we could try out when we might have a larger budget for testing new technologies. I say we can switch these figures around (see Figure 1-9) without increasing the total IT budget or lowering the quality of operations and maintenance. This is definitely not an impossible task, which I hope this book will show.

My friend from the car industry started this process in his part of the company. He never got to see the final result of his efforts before he left the company for a consulting business but he saw the figures turning—enough to make him even firmer in his opinion that it can be done. The key here is obviously to take control of your ALM process (see Chapter 2) by using a tool such as VSTS.

Before we get into a discussion about how to take control of our ALM process, let's look at some of the factors influencing this split of IT money.

12. Corporate Executive Board, "Application Budget, Staff, and Portfolio Benchmarks," 2006.

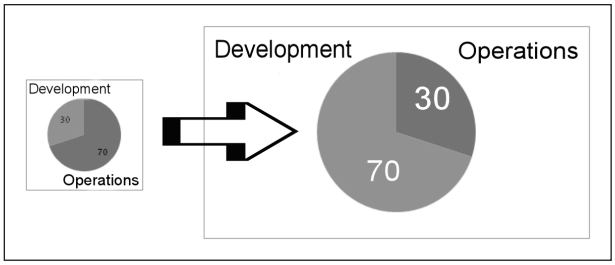

Figure 1-9. *Making the switch*

Factors Influencing IT Spending

I recently visited one of the biggest mail order companies in Northern Europe. They have managed to change their IT budget spending to 30 percent operations and 70 percent development. When I asked what was one of the most important solutions they had used to accomplish this, the IT manager said that they had just simply stopped making small changes or fixes to the existing systems. The cost involved in such changes was far too great, so only very important changes were allowed to be implemented. With greater traceability and automated unit testing, he said, they could have continued lowering operation costs.

One other cost driver for maintenance and operations is the retirement of an application or system. Raise your hand, everyone, who has actually planned for this event (okay, you in the back, you can take your hand down now). If this scenario is not planned for in the beginning of an application's lifecycle, great surprises can occur in its end. One example is from my friend, the car manufacturer. At times his business had to retire applications because the specific platform running the application became obsolete or a new version of the application was developed. Support for the platform might end because of newer platforms replacing it. On some occasions, my friend found that there was no way to migrate the historical data in the application(s). This scenario had not been planned for earlier and suddenly posed a great problem. The car company then had to negotiate an extended support contract with the platform vendor, which was way more expensive than it had been when the application was alive. It was quite ironic that they had to pay more for having access to historical data than they had when they actually used the system(s). One of the solutions to begin turning the figures around was to start planning for application retirement early in the lifecycle. The ALM process was changed so that, for example, data migration was planned for, making it unnecessary to keep applications slow cooking at a great cost.

I cannot cover all cost factors in this book, but I will mention some more key issues here. A big element causing increased operation costs is the way we have traditionally built our systems (and still do build them). Some years ago, client-server solutions dominated much of our IT environment. After that came multitier applications. These gave us great opportunities in writing business systems, with scalability and availability in mind, not only as standard Windows applications but also as web applications. My coworker Rickard Redler and I even wrote two books on this topic, *Designing Scalable .NET Applications* (Apress, 2003) and *Pro Scalable .NET 2.0 Application Designs* (Apress, 2005).

If we have only a few applications, this architecture works fine, but when new applications are added, it gets complex (see Figure 1-10).

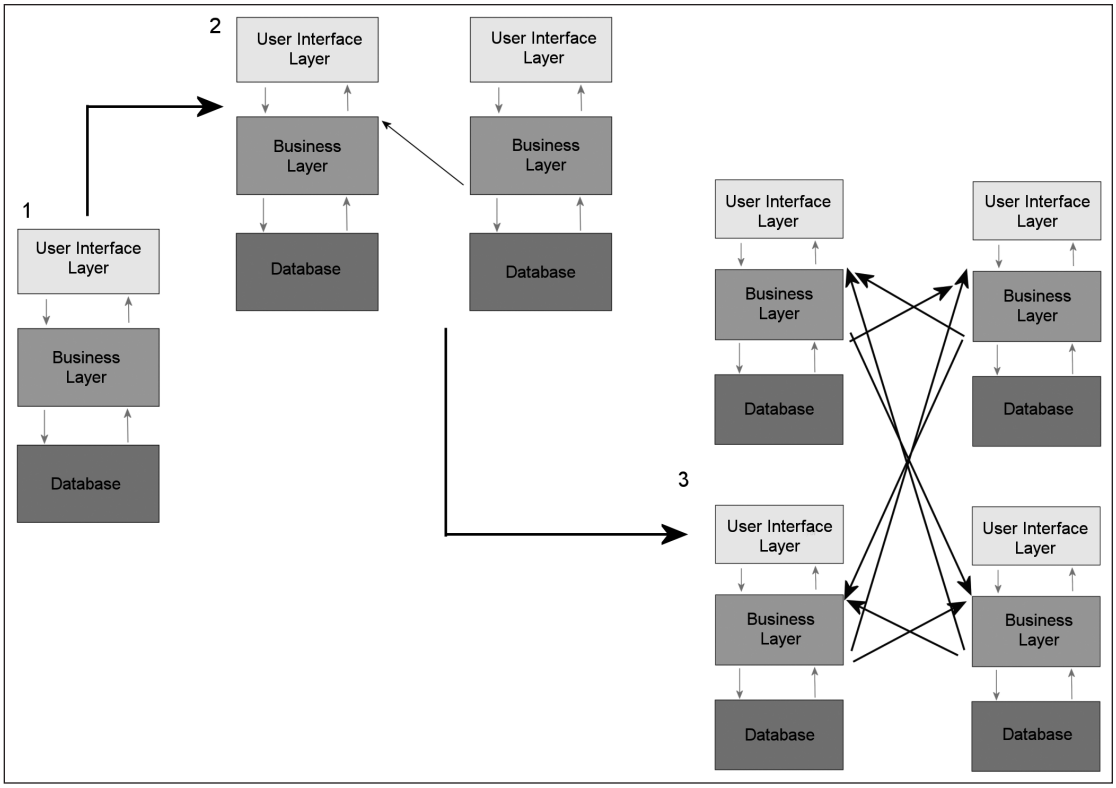

Figure 1-10. *A traditional multitier application design. When we have several applications trying to access each other's functions, problems may arise.*

Imagine what happens when a new application needs to access data from another. There are ways to solve this, as position 2 of this figure shows. We simply let the data layer of the new application access the business interface layer of the first. This way, we can reuse the logic that already existed and not spread database security and rights management around. If we have only a few applications, this works fine, but as you can see in position 3, things can get pretty complicated when only a few more new applications are introduced. Most large enterprises have perhaps hundreds of applications spread across the company's different locations. If there is no documentation task force in the company, there really is no way to have control over where the business logic is located.

In Chapter 4 we will discuss service-oriented architecture (SOA) a bit and see how we can be helped by that mindset. SOA has the potential, if implemented correctly, to help us align our business process with our business systems, making us adjust quickly to new competitive business strategies. SOA itself is a natural evolution from previous development architectures, where we have found that component-based development and traditional architecture just does not work for providing the flexibility we need these days.

Having an inflexible architecture is a great cost driver in maintenance (and of course in new projects). Imagine the nightmare of implementing a change request or bug fix in this environment. There really is no way to have control over where a change or fix will have its impact, so just a small change will need extensive testing on much more than just where the fix was implemented. This makes the cost of it much greater than it should need to be.

One way to avoid this is to have better ways of documenting traceability from the original requirement to accepted and delivered code. With the right tool(s) and the right work process, it would be much easier to find where a change or fix will have its effect, and testing could be minimized.

Another way to make the testing easier and hence less costly is to have good unit tests ready for the code (we will come back to this in Chapter 5) and ways of running them automatically. This way, it is easier to see if a change or a fix affects any of the code without manually testing everything else.

I have often experienced another problem at my clients. A company has a great (and costly) infrastructure in place with redundancy for most applications and databases. Availability is high, and everyone should be happy because it's really great to have such an environment in place. But think about it this way: is it really necessary for all applications to have such an infrastructure? Consider this: does a low-priority application not requiring 24/7 support or 99.999 percent availability need to run on such an expensive infrastructure? Isn't it more cost-effective to run those applications on a different platform and spend the money on the systems that really need it?

Furthermore, you could cut costs if you make considerations for the time the system is expected to live. Carefully consider the infrastructure needed for all new applications. Are the requirements the same for the system with a lifespan of two years as it is for one with ten? It could be, of course, based on its business impact, but a system should not be routinely implemented on a specific platform just because you always do it like that.

How about the company operations process? Is one in place or is ad hoc work done? I am talking Information Technology Infrastructure Library (ITIL) or Microsoft Operations Framework (MOF) here. It's just as important to have a good development process in place as it is to have an operations process. What happens when change requests come? When bug reports are entered? When new releases are introduced? Having a well-defined process in place could be a major factor in cutting operations costs. We will discuss this a bit more in Chapters 6 and 7 because it is an important topic.

With better traceability in our systems in place and tool(s) supporting it, I am certain that we could have a situation where we would not have to stop implementing small changes or features just to cut costs. If we can trace exactly where an update or bug fix will affect the system, we can make sure not to break anything else.

If we also consider our system architecture, we could create a better structured environment where we could more easily see how changes affect the system as well. Converting slowly but steadily to a SOA would be a good start.

Before we leave this chapter, I want to discuss the gap between IT and the business side again. If the business processes and the IT systems are not well aligned, there will be problems when processes change. This is a little bit like what we saw earlier with the application mess we can end up in. It is hard to say where we need to make changes to our systems when new processes need to be implemented or old ones change, if we do not know the structure of our systems. A lot of the costs involved in this process unavoidably end up in the operations budget.

Summary

An alarmingly large portion of IT projects delivered today are challenged or, in the worst case, abandoned. Many projects have overruns in terms of cost and time. Projects are also criticized for not delivering business value, and hence are not what the customer expects them to be in the end. One of the greatest risks is definitely the lack of integration between the business side and the IT side. This gap makes it harder to deliver what we should deliver in our project, which is business value. Having a development process that is ill-defined or not even used is another great risk. Furthermore, the lack of great ALM tools makes it harder to deliver as well, especially because we have more geographically dispersed development or project teams these days. Considering the amount of money spent on these projects, we can be certain lots of it is thrown away because of these challenges.

We can also be certain that much of our companies' IT budgets go into operations and maintenance, giving less money to develop better and more-efficient systems that give more value to the business. To the technology interested, this means less money and opportunity to try new cool techniques or tools, less testing of new technology, and so on. For the business, it means less opportunity to earn money and add value to the company. Either way you choose to see it, this model for IT budget spending is definitely a great problem.

The problems addressed in this chapter can be greatly improved by having control of our whole ALM process. This process, as you will see in the next chapter, focuses on taking the business needs and requirements, and turning them into business value for the organization. ALM does so by enforcing a process on how we work when developing software.

This book will show you how, with the use of a set of tools, Team Foundation Server (TFS) and Visual Studio Team System, we can take control of the ALM process. The result will be that we can reduce the impact of the problems presented in this chapter. We might not get all the way with the current versions of VSTS and TFS, but we can certainly come a long way. So follow me to the next chapter, where you will see what ALM really means.