

# Developing a Vision: The First Step in a Project Lifecycle

# Developing a Vision

## Before Developing Requirements...

Even at the best of times, software development can be a risky business. Projects seem to be striving to prove

Murphy's Law; anything that can go wrong will go wrong.

When a software project begins to go wrong, the first response is usually to demand extra effort from the development team to pick up the slack. Many of these teams manage to work wonders, working extremely long hours to deliver software to meet critical business needs. But the worst kind of problem, one that can cause a project to fail completely, is one that no amount of heroics can possibly fix. The worst problems occur when a project team is building the wrong product in the first place.

The only way to avoid that problem is to make sure that everyone involved is in full agreement as to what is needed and what the project team is trying to build. The process of developing a vision for a software application brings everyone into agreement as to what constitutes success for the project.

## Defining the Vision

Developing a shared vision is key to successful software construction. As defined in the Microsoft Solutions Framework

(<http://www.microsoft.com/technet/itsolutions/techguide/msf/msfovrw.mspx>):

When all participants understand the shared vision and are working toward it, they can align their own decisions and priorities (representing the perspectives of their roles) with the broader team purpose represented by that vision. Without this vision, the business value of a solution will lean toward mediocrity.

A shared vision for the project is fundamental to the work of the team. The process of creating that vision helps to clarify goals and bring conflicts and mistaken assumptions to light so they can be resolved. Once agreed upon, the vision motivates the team and helps to ensure that all efforts are aligned in service of the project goal. It also provides a way to measure success. Clarifying and getting commitment to a shared vision is so important that it is the primary objective of the first phase of any project.

## Reasons to Launch an IT Project

The very first thing that needs to be done is to determine who the project fits into the overall strategic direction of your organization. A software project can help bring the following benefits:

- **Improved Efficiency**—this means using IT to reduce the cost of doing business, generally by automating processes that used to be done manually. These reduced costs often allow businesses to do significantly more work with the same staff than they could before the project was completed.
- **Better Customer Service**—enables you to better meet the needs of your customers. These projects are usually targeted at a specific customer or closely related group of customers rather than customers in general.
- **Customizability**—to allow the organization to more easily adapt to change, either in the marketplace or internally. These kinds of projects make it easier to release new products into the marketplace, to change internal processes, and the like.
- **Knowledge and Insight**—allow better access to the information you have, so that senior management can make better decisions
- **External Requirements**—this kind of project is triggered because forces outside the organization require it. This might happen through regulatory legislation, or because a competitor has developed new capabilities that you have to match.

You need to identify which of these reasons are the key drivers behind your project. Many software development projects are initiated for several or all of these reasons in combination, but it's important to decide which goal takes priority. For example, a project begun to bring you in compliance with new regulations probably must meet a fixed deadline, and it's necessary to subordinate other goals to that objective.

## The Planning Process

### Identify Your Key Stakeholders

The people who matter to you are both inside and outside your business. It's critical to the planning process that all of the affected stakeholders have their interests considered during the early planning stages of the application development lifecycle.

#### Internal Stakeholders

These are the people within your organization who will be affected by the new system. Successful development requires the input of all of these groups and the recognition that each of them is likely to have different concerns and desires. For instance, line staff are primarily interested in the system being easy to use and are frequently resistant to change for fear that their jobs will be negatively impacted. Management is likely to be interested in

how the software will enable them to better meet their strategic objectives and in improved reporting on the work done by their departments. Internal Audit wants to be sure that the system ensures that users follow the correct procedures and that any actions taken are recorded and can be reconstructed. And of course, IT wants the system to be as straightforward and simple to develop as possible. Finding an appropriate balance between these goals is the biggest challenge you will face during this stage.

### External Stakeholders

External stakeholders include customers, industry regulators, and other groups who may not directly use the system but will be affected by the changes it brings to the way an organization operates. It is usually not practical to directly involve them in the planning process, so internal people familiar with their needs should be asked to represent them. The people responsible for representing external stakeholders need to be careful not to project their own desires for the system onto the planning session.

### The Business Case

Once the objectives for the project are determined, it's necessary to build the business case. A properly developed business case is necessary to allow intelligent decisions to be made when running the project. The business case shows the benefits that the project will deliver to you (and the costs you may incur if you decide not to go ahead with the project). The purpose of the business case is to define what will constitute a successful project.

Most people think that it'll be obvious whether a project has succeeded or failed. The reality is quite different. Most software projects are partial successes, with the various stakeholders getting just enough to make them feel that they've benefited from the project in some way. But it's rarely clear whether the project paid its own way. The business case allows you to answer that question.

However, a properly constructed business case can do a lot more than that. Most projects are not equally valuable no matter when they are delivered. If it's important for the project to be delivered quickly, a well-structured business case can demonstrate whether or not it's worth accepting additional cost to accelerate the schedule. It can also aid the project team by indicating the comparative priority of features and in some cases may even show that a project cannot produce enough benefit to the organization to justify its existence.

### Risk Assessment

The goal of the risk assessment is to identify potential problems that could cause the project to not meet the goals set out for it in the business case. Risk assessments should be carried out and updated over the lifecycle of the project, of course, but the goal of the initial assessment is to understand the uncertainty associated with the effort.

The risk assessment must look at risks that are outside the scope of the project itself. Potential sources of uncertainty include:

- New government regulations or legislation affecting your industry;

- Competitors introducing a product into the marketplace that alters the competitive landscape;
- Changes to operating systems or other architecture components that force an upgrade;
- The release of a commercial software project with similar functionality to the software under development;
- The organization acquires or is acquired by another organization;
- And, of course, risks internal to the project as well.

Any project worth doing involves some degree of risk. The purpose of risk assessment is to allow you to spot risks as they show signs of materializing and reduce or eliminate their impact on the project. A proper risk assessment ensures that you don't undertake projects that only make financial sense if everything goes perfectly—because it won't.

### **Business and Infrastructure Assessment**

A common step in the development of a vision is to conduct an assessment of your current business and IT infrastructure. How much you need to do this depends on the project, of course; if the goal is to develop an entirely new product that has little connection to existing processes then you don't need to know much. In most cases, though, you will be replacing systems and processes that already exist. Your project team will need to be closely familiar with these processes and systems in order to successfully integrate the completed project into the organization.

The purpose of the assessment is to ensure that no major requirements are overlooked. You need to identify things like:

- Unusual or exceptional circumstances that your line staff have to regularly deal with. These exceptions often require more analysis and development effort than the normal course of business, and identifying them early can help avoid scope changes later on.
- Data retention and migration requirements.
- Interoperability and system interaction requirements.

## **Structuring the Project**

### **Determining Key Features**

Once the opportunity is understood, it's necessary to put some structure around the project. First and foremost, that means developing a listing of the key features the new software is required to support.

Often, this will require an assessment of the organization's current IT infrastructure. Users are generally reluctant to give up any current functionality they possess, and so the system must offer clear advantages over the existing application.

This list of functions will be the core on which detailed requirements are built during later phases of the project. It is usually considered to define most of the scope of the project and limits the responsibility of the project team. Changes to this scope over the project lifetime will occur, of course.

The feature list must be sufficient to deliver the benefits to the business as defined in the business case. If they don't, then it becomes clear that a requirements gap exists and it must be addressed. It is likely that as the feature list is developed, stakeholders will request additional functions to fix problems they perceive with the status quo. These features should be justified as part of the business plan.

Once a feature list is developed, it must also be prioritized. All else being equal, the features that deliver the most value should be developed first. However, in reality, other factors will have to be considered as well. Once the features are defined, the project team will have to examine the features and identify which of them carry the highest risk to the project, usually because the team does not (yet) know how to implement them.

### **Define a Release Plan**

Iterative development has become the generally preferred method of building software. In the past, a software solution would be built over a period of years, with all of the functionality of the project bundled into one release. These large releases often caused significant problems for the business. Often, key processes would have changed since the requirements were defined and agreed to, meaning that the application would have to be expensively retrofitted to match the changed environment. In addition, there was a real risk of analysis paralysis, as the project team had to nail down—and get agreement on—a feature set larger than any one person could comfortably understand.

Software development is much easier when tackled in small pieces. However, that represents what's easiest for the developers, not what's easiest for the business. From the development and project management point of view, small and frequent releases are best, with the planning for each release done once detailed requirements are complete.

However, the business may prefer larger and less frequent releases. Often, managers are concerned about the potential disruption to the operation of their departments that can be caused by frequent upgrades to key applications, particularly if those upgrades happen during especially busy times of year. There is, as previously mentioned, the need to avoid taking functionality away from end users. And there is often pressure to add new features to each release to meet changing market conditions.

The release plan is designed to address all of these concerns and find an appropriate balance between them. Early releases should usually try and tackle the highest-risk components of the system. The reason for that is that if the project is going to fail, it should fail early on, allowing the project to be cancelled if it will not work.

This is not always appropriate, however. Sometimes, the project includes lower-risk components that are valuable to the business even if the rest of the system is never delivered. In other cases, infrastructure must be developed before it's possible to implement the high-risk technology.

### **Select the Technology**

The final step in this process is to determine the basic technology and architecture that the project will require. The decisions taken in this step are likely to have more impact on the long-term success of the project than any other.

### ***Build or Buy?***

Buying pre-packaged software can result in significant cost savings if your situation allows for it. Purchased software is best at supporting standardized functions that are not central to your business.

The main problem with these programs is that they come with many assumptions about how to do business built in. If you want to take advantage of the cost savings they offer, you have to be prepared to use the application pretty much as-is. Changes to pre-built software are extremely expensive to make, as the program has already completed its development cycle.

It's generally unwise to pour a lot of effort into customizing an already-built application. Even seemingly small changes can require extensive reconstruction of the application's infrastructure. The costs of extensively customizing an application are likely to be even higher than the cost of building a similar application from scratch. If you choose to purchase and implement an existing application, you should seriously consider accepting any limitations it has and adjusting your business processes to work around them. If you can do so, however, you may realize significant savings and will certainly be able to implement the application much more quickly.

If no existing application meets the needs of the organization, then it may make sense to develop an application from scratch. Of course, few applications are entirely custom-built. Most custom development initiatives integrate several pre-built general purpose applications and make use of toolkits to provide needed common functionality. The development work will focus on the user interface and other portions of the application that are specific to the needs of your organization.

The selection of these components will depend on your existing IT infrastructure (another reason to do the survey mentioned earlier). It's unlikely that all your existing applications will be replaced at once, meaning that the new applications will have to be able to communicate with the old. The choice of vendor is important; you will want to be certain that the vendor will remain in business and be able to support the products for the projected lifetime of the application (and preferably beyond; many software applications remain in use well past the point their original designers assumed they would be replaced).

## Conclusion

When the decision is made to begin a software project, there's often a strong temptation to just get on with it. People often feel that until coding starts, nothing useful is being accomplished. However, that is not the case at all. Before the project is begun, the business must verify that it is worth doing at all, set clear goals for the project, define the trade-offs that can be made during the project, and determine the features and architecture of the system.

This information, and the decisions made in this initial phase of the project, are critical to its success or failure. Organizations that fail to work through these questions at the beginning of a project may well feel that they have a track record of successful projects—but that's mostly because they defined success as whatever they ended up doing.