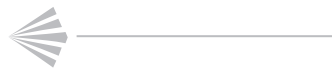
2 Requirements Value



Whenever we make an investment—some stocks, a new car—or do some work, we like to think we get something in return for the money or effort expended. In this chapter, we look at your investment in requirements and what you can expect in return. And though all men may be considered equal, all requirements are not. Some requirements are crucial to the product, while others are gold-plated luxuries. We discuss how business people and requirements analysts can determine how much to invest in requirements, by focusing on the value of requirements to their organization.

An Investment Story

Amelia's parents have decided move to Australia. A week before they leave, Amelia's parents call Amelia over and tell her the camera shop they have been running for the past 20 years is all hers. "It's not the greatest camera shop in the world," they tell her, "it has been losing a little bit of money for the last few years, but we kept it going for your sake. We are sure you can turn it around. Here are the keys and the lease for the shop. We're off now. Be sure to write."

Amelia looks at *her* shop. "A little run down," she thinks, "but the location is good. The main problem is Mom and Pop have not kept up with the times. Perhaps if I update things a little it will work out for me."

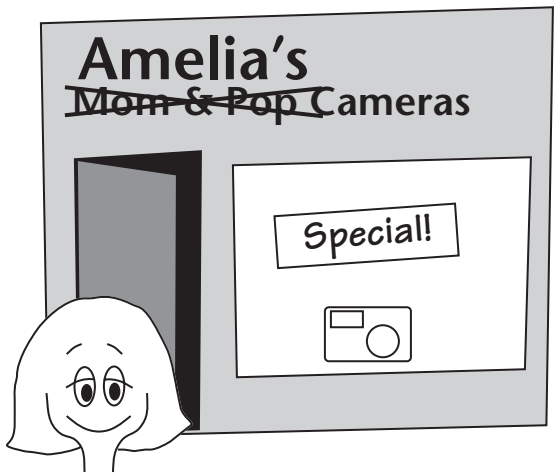


Figure 2.1
Amelia has to decide whether or not to invest in the camera store. Her thinking is not unlike deciding whether to invest in requirements.

Amelia is now faced with several investment choices:

1. Take over the store and continue to run it exactly as is.
2. Take over the store and invest in it. This means updating the image, putting emphasis on digital photography, and buying equipment to give digital customers full service.
3. Make minor changes hoping the changes will attract enough customers to keep her head above water.
4. Walk away.

The first option gives Amelia no apparent risk in that she invests no money of her own, and thus will probably steadily lose money over time. So "business as usual" appears not to be a good strategy for her.

If she chooses option 2, she has to borrow heavily and invest in the store and particularly in digital photography. The track record of other stores doing this seems good, and the market for a good, modern camera store in this neighbourhood is buoyant. Choosing this option, the next question for Amelia is how much to invest. She might plunge in heavily and turn the store into a complete digital imaging center, in which case the return on investment (ROI) will be longer term. Or invest just enough to make this a popular store for people looking for the latest piece of cool photographic technology. The lighter investment will see returns earlier, but the long-term gain will be less than if she makes the deeper investment.

Option 3 is bland. Amelia knows she will not make a lot of money from the store if she follows this route, but the investment amount is low and she feels good about being able to repay the loan.

Option 4 has a complete lack of risk. She invests nothing and gets nothing in return.

This is not a book about camera stores. It is about requirements and project success. However, Amelia's choices are concerned with investing in something to make it more valuable. Requirements are the same: you are investing in the requirements part of product development to end up with a more valuable product *and* get it less expensively. We aim to show you how investing in the requirements part of your project should be seen and treated the same as if you were investing in property, bonds, stock shares, or anything else that has the potential to give you a positive return on your investment.

Investing in Requirements

Let's return to the allegory of Amelia and her camera shop. She has several investment choices. The first is business as usual. The parallel for you is continuing to do whatever you are doing in the requirements field today. Nothing ventured, and nothing will be gained.

Her second choice is to invest in the store, bring it up to date, install modern equipment and sell modern cameras. This might cost her some money but is probably the best way for her to make money in the long run. For your project environment, the analogy is to invest time and effort in improving the way you discover and communicate requirements and reaping the reward of better products and cheaper delivery.

You are investing time and effort in improving the way you discover and communicate requirements, and reaping the reward of better products and cheaper delivery.

Amelia has two other options. She can tinker with the store. A minimal "band-aid" approach might be acceptable if she thinks the store is incapable of producing more value. However, it will probably not yield much. Nor can you expect a "band-aid" approach in project development work to yield substantial returns.

Her final choice is to walk away. For Amelia, walking away may be the right choice depending on the business climate for camera stores. In your case, however, given the increasing evidence of the positive effect of good requirements on project success, this is the equivalent of management's head in the sand.

In this chapter we give you some ways of justifying, determining, and managing your investment in requirements. You can, and should, look at your investment in requirements as you would look at any other investment:

- How much are you willing to invest?
- What is the ROI?
- What is the time for ROI?

- What are the risks?
- Should you invest at all?

The last question is perhaps the most pertinent for requirements. In our experience, several reasons exist for investing in requirements:

1. So your project builds the right product.
. To shorten the delivery time.
3. To find the right product for the long-term operational or sales success.
4. To take advantage of the products of requirements engineering to help guide the project by making more informed decisions.
5. To discover a better product.

We address each of these reasons in this chapter, but first let's consider the least-cited reason for investing in requirements:

Discover a better product.

Investing in requirements means investing the time, effort, and skills to discover the requirements correctly. The careful gathering of requirements always results in an inspirational leap forward for the product. The collective effort of the interested stakeholders produces the "killer requirement," something considerably more than routine automation of an existing task.

Killer requirements turn out to be wonderful business assets and produce a huge return on their investment.

These killer requirements turn out to be wonderful business assets and produce a huge return on their investment. For example, requirements that enabled customers at Amazon.com to write their own reviews gave customers a feeling of participation and belonging to the site; people kept coming back for more. Another great Amazon requirement was to make ordering as simple as possible. When requirements analysts had defined "as simple as possible," Amazon invented 1-Click and customers loved it.

"On FedEx.com we're averaging more than 2.4 million tracks per day and each one of those transactions averages just under a nickel. We're saving about $25 million a month."

—Robert Carter, *CIO of Federal Express*[1]

Federal Express, UPS, DHL, and others invested in a requirement they discovered from their customers. Customers wanted to track their shipments online. The couriers spent millions to satisfy this requirement, and so far it is proving to be a very good investment. FedEx calculates each telephoned tracking request costs FedEx about $2.30. FedEx receives about 100,000 calls per day. The Internet alternative is far cheaper and has produced some stunning savings in ongoing operational costs.

As another example, FedEx has recently added InSight to its services, which enables customers to track inbound shipments without needing to know who is sending them. Nor do they need to know the tracking

1. *Fast Company,* April 2003

number. So far, InSight is proving popular with clients who wish to plan their production for incoming shipments of materials.

Building the Right Product

Building the right product means finding what is really needed, and not just what people say they want. It also means getting it right first time, and not having to drag through the long repair activity that many projects suffer. Requirements gathering is about finding this right product and specifying it unambiguously to the developers.

Shorten Delivery Time

Shortening delivery time might seem at first glance to be a surprising reason to invest in requirements. However, our clients consistently find that by including a thorough requirements activity in their development cycles, they deliver the right product sooner. Requirements have value because they enable the builders to concentrate on developing the product instead of guessing about missing or ambiguous requirements, then having to retrace their steps and undo incorrect interpretations. When the builders do not receive the correct specification, programmers spend at least half the project budget in post-implementation correction. This is not only expensive but very frustrating to the client and business users waiting to deploy the finished product.

Requirements have value because they enable the builders to concentrate on developing the product instead of guessing about missing or ambiguous requirements, then having to retrace their steps and undo incorrect interpretations.

Linking Requirements to Managing

Requirements activity includes several deliverables we have found invaluable in the management of projects. For example, it includes building a context model. From the context model, we count function points to accurately measure the size of the work to be studied. We also advocate you use the context model to identify business use cases. Then use these as a collection unit for gathering requirements.

We know plenty of managers who use the status of business use cases as their way of measuring progress. The identification of stakeholders gives you a way of assessing the willingness of the people whose participation you need. This alone can serve as a reliable indicator of whether or not to proceed.

We will discuss this area more fully in Chapter 9—Managing the Requirements. For more on context models and business events see Chapter 4—Learning what People Really Need.

Return on Investment

The simple way to look at the ROI is the net gain from the requirement divided by the cost of discovering, verifying, building, and operating the requirement:

*% ROI = {(benefits – costs) / costs} * 100*

> *"Improved software practices provide returns ranging from 300% to 1900% and average about 500% . . . The reason for these exceptionally high returns is . . . improved practices [that] have been available for decades, but most organizations aren't using. Risk of adopting these practices is low; payoff is high."*
>
> —Steve McConnell, *Professional Software Development*

Suppose you discovered and implemented a requirement for your Web site to recognize long-term customers and award them loyalty discounts. Then you find out orders from the Web site are up 5% because of these loyalty discounts. Let's say the cost of finding the requirement is $1,000, the cost of implementing it is $100,000, and the profit to the business because of the increased orders from that requirement is $250,000 in the first year. Assuming the correct requirements are implemented as specified, the ROI of the requirement (147% in the first year) is pretty good.

It would be very difficult, and time consuming, to establish the cost of a single requirement. As we progress, we will look at "units of requirements" (product use cases, business use cases, tasks, products) we can measure. Similarly, the benefit can be a saving as a result of installing the requirement or income generated by the requirement's existence.

Investment Risks

Any reasonable investment advisor, and most investment advertising, cautions you about risk. The following are some of the risks related to investing in requirements:

> *"Our SAP project had built a budget for the development work being done this year for our SAP installation into Dublin, Ireland. We have only had to spend 53% of those dollars. Why? A more thorough job of requirements development was performed, resulting in the employment of less programming contractors."*
>
> —John H. Capron, *Worldwide Systems Technology Manager, IBM Enterprise Systems Group*

- Not knowing the requirements
- Putting too much effort into requirements
- Being too formal with the requirements and the specification
- Being too casual
- Becoming bored with the requirements process

Risk, as we use the word here, is a potential problem. In project management terms, it means you have to be aware of risks, the likelihood of a potential problem becoming an actual problem, and the impact if it does. If the likelihood is slight, then the best course of action may be to proceed, monitor the risk, and act only if the problem manifests itself. On the other hand, if the risk is considerable and the consequences dire, then the obvious action is to implement preventive measures.

DeMarco and Lister[2] define risk as "a weighted pattern of possible outcomes and their associated consequences." The risks that you face in project work fall somewhere between "slight" and "considerable." Let's look at them and consider the appropriate response.

2. DeMarco, Tom & Tim Lister. *Waltzing With Bears: Managing Risk on Software Projects.* Dorset House, 2003—*Practical strategies on how to recognize and monitor risks. The writers focus on risks common to software projects but their techniques are applicable to any type of project.*

Risk of Not Knowing the Requirements

This risk must always be considered serious, as it is impossible to build the right product if requirements are unknown. Note the requirements need to be *known*. The builders can know them without having them in writing; undocumented requirements are acceptable in some circumstances. However, it is extremely unlikely the builders can know the requirements if they have not studied the work area into which the product is to be deployed.

About 60% of software errors are requirements errors.

From our experience, about 60% of software errors are requirements errors. Thus, not knowing the requirements is, by most people's definition, a considerable risk. As with any other risk, you can calculate the impact of poor or missing requirements. If the unknown-requirements risk becomes a problem you attribute the cost of rework, debugging, error correction, complaint handling, lost customers, and delays while the product is made right to lack of awareness of the requirements.

For strategic products, the impact can threaten the health of your organization. If the product is in any way medical, it can threaten the health of your customers. In any event, you need to assess the impact along with the likelihood of the risk manifesting itself as a problem.

Risk of Putting Too Much Effort into Requirements

This one might seem strange coming from people who make their living consulting on requirements projects. However, it has been our experience that some organizations specify requirements that have already been specified in some other form. The object of the requirements activity is to ensure the right product is built. However, it must only specify requirements that are not already known to the builder of the product.

This risk is usually greatest in organizations that have an inflexible development process. Analysts are forced to grind out the requirements specification and perform whatever else is ordained merely to satisfy the process itself. The risk manifests itself as a delay in building the product, as well as in the dissatisfaction of people involved in the project. Further, builders spend extra time wading through extraneous documentation generated by an inflexible development process.

A balance exists between not putting enough effort into the requirements activity, and putting in too much. You have to assess the skills and knowledge of the builders and determine if the requirements analysts are feeding them what they need or more than they need.

Risk of Being Too Formal with the Requirements and the Specification

You do not have to write down all requirements as atomic statements, provided you have some other way of communicating them and pro-

vided this other way of communicating serves as a record of the requirements. For example, if you have a set of process models supported by process specifications, a data dictionary, and a data model, then these models may be sufficient specification of the functional part of the product. If the models are rigorous, little is gained by writing out each functional requirement.

Alternatively, you may have fully specified a product use case. This specification includes the preconditions, the exit criterion, a description of the process, and the actors. Provided the use case is relatively simple, this specification alone may suffice as functional requirements.

Please bear in mind neither of the above alternatives specifies the equally important *non*functional requirements. These are the properties or qualities the product must have. The alternatives mentioned deal only with functional aspects of the product.

Factors that help to decide the necessary degree of requirements formality are:

- Degree of hierarchy and politics
- Fragmentation of knowledge
- Geographical distribution of stakeholders

If you have several levels of management and the requirements must be reviewed at all levels, you need consistent formality. If requirements knowledge about a particular subject is scattered among several people, you need formality; otherwise, you waste a lot of time dealing with inconsistent interpretations. When stakeholders are located in different offices, buildings, cities, or countries, then you need more formality to avoid misinterpretations and wasted time.

Risk of Boredom with the Requirements Process

This investment risk applies to projects in which you determine the requirements by communicating with business users. The business users are giving you their time in the expectation they will eventually get a suitable product. Business users generally feel the time they spend with analysts is time away from their own work. You need to give them something in return for their time, and give it quickly and often, so they do not become bored and discontent with the requirements process.

The boredom risk manifests itself in the behavior of business users participating in the requirements process. They believe they are contributing, but because they lack feedback or because the projected delivery of the product remains so far away, they enter into a robotic-like style of participation. Their answers sound good, but are often intended to get the analysts out of their office.

Practical ways of avoiding the boredom risk are to give feedback on the requirements the user has contributed and to implement early and frequent releases of the product or simulations of the product. Only when the business users (and for that matter the project team) can see the results of their requirements effort do they fully understand the benefits of spending time with the business analysts. By firmly and continuously establishing the requirements-to-product connection in the business persons' minds, you address their reluctance to participate in your project.

Only when the business users (and for that matter the project team) can see the results of their requirements effort do they fully understand the benefits of spending time with the business analysts.

What to Invest In?

There are always more possible investments than time or money available. We find some products make better investments depending on the time needed to realize a return. Figure 2.2 summarizes this idea.

A *strategic product* is one that adds value to your organization by providing a service or product not currently provided. Further, it provides some long-term market or operational advantage. For example, at the time of writing airlines are converting to e-tickets. This is a strategic product as it enables passengers to interact differently and may eventually significantly reduce the cost of check-in.

At the time of writing, Apple has become very successful with its online iTunes Music Store. The millions of songs sold through this site have added considerably to Apple's revenue stream. This product is strategic: though it is not part of Apple's core (no pun intended) computer and software business, it is a logical add-on to its digital lifestyle approach.

Infrastructure products on the other hand are changes more than additions and are intended to make some internal process run more smoothly

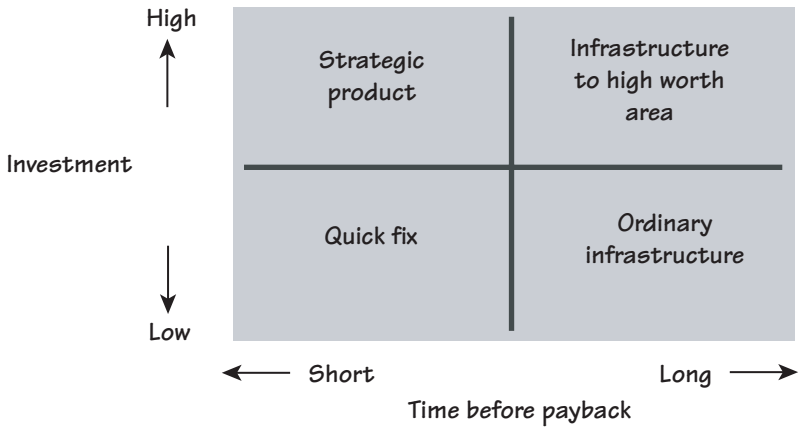


Figure 2.2
This model helps to manage expectations of ROI. High investments in a strategic product usually result in a quicker ROI than the same amount invested in an infrastructure product. This does not necessarily mean it is better to invest in strategic products, only that expectations for return on investment are different and should be recognized as such.

Strategic products usually add to or make a significant and valuable change to the organization's business.

or at a reduced cost. For example, when you changed to using barcode scanners in the warehouse to perform inventory, that was an infrastructure change. When you abandon barcodes in favor of radio-frequency identifiers, that will be another infrastructure change. You do not sell infrastructure products (unless you are a vendor of ERM or CRM or similar kinds of software), and infrastructure products do not generate revenue.

Think of an infrastructure product as something you build (or buy) to use.

Returning to Figure 2.2, suppose you are a bank. Let's say a strategic product for you is a new type of bank account intended to attract a type of customer you do not have at present. Your investment is probably high but you expect to realize a high return in a short time.

Continuing with the banking analogy, suppose you make enhancements to your Internet banking so your customers can pay any bill using the Web site. This shows on the diagram as *infrastructure to a high worth area*. Your investment is high (to enable paying *any* bill is expensive), it is likely your return will be longer term because you are providing a better service and it may take people a while to realize it. After all, paying bills is not uppermost on most people's wish lists. (As an aside, if you could produce a product whereby people did not have to pay their bills at all, that would be a strategic product, and the payback would be quick and spectacular.)

You can have a low investment and get a fast payback from doing a *quick fix* like making corrections to something that is not currently working. You might also make an enhancement to the *ordinary infrastructure* that bank employees use to order stationery. The payback is longer term: whilst the bank infrastructure works better for the employees, it is not a high-worth area that adds value to your organizational purpose.

To Invest or Not to Invest

John Favaro is a software consultant who works in Italy. His brothers work in management consultancy in New York and Chicago. Despite the geographical separation, they have managed to combine the well-established principles of strategic investment management with investment in software products. They have used the Market Economics and Competitive Position framework to explore two factors to consider when making an investment decision: *worth* and *competitiveness*.[3]

3. Favaro, John and Kenneth. *"Strategic Analysis of Application Framework Investments,"* Building Application Frameworks: Object-Oriented Foundations of Framework Design, edited by M. Fayad, R. Johnson. John Wiley & Sons, 1999—*Combines John's software knowledge with Kenneth's strategic investment knowledge and provides background on the principles of bringing together strategic investment principles with the investment in software.*

Worth

Worth is about identifying the value you place on your area of investment—the market in which you might invest. For example before Amelia decides whether to do anything about her inherited camera shop, she must decide if it has any worth to her. That is, does she really want to run a camera shop? Does she intend to become a writer, say, and a camera shop has no worth for her? On the other hand, if she dearly wants to carry on her parents' work, or she sees the shop as a great way to make her living and feels the digital camera market is lucrative, then she may place a high worth on it.

Next consider competitiveness. To attract customers, Amelia will have to give her shop features, or provide services that attract potential customers. If her shop sells a better range of cameras, or sells them more cheaply, or provides printing and uploading services unavailable in other shops, she increases her competitiveness.

We can ask the same two questions—worth and competitiveness—to assess investment in requirements.

Your product (regardless of whether it is a consumer product, a piece of software or a service) will be deployed in a work area. Bear in mind a work area is either a part of your own organization or an external market for which you are developing products. The work area may be accounting, research and development (R&D), marketing, engineering, or almost anything else.

The question is, what is the worth to the organization of that work area? For example, the pharmaceutical companies place a high worth on R&D. If they do not come up with a breakthrough drug every few years, sales decline as generic products start to encroach on sales. If you are an investment broker, the trading part of the business has a high worth to you as the source of your revenue stream. If you are in the retail business, the supply chain—having the right goods available for sale at the right time—is a high-worth work area.

Look at your organization and possibly speak with your upper management to find what they find worthy. The answer by the way is not "profitability"—profitability is a goal, not a work area. But it is a useful answer. Now ask what parts of the organization contribute most to the profitability of the company. Also ask what work areas have the most value in meeting company goals. This question is all about investing in good work areas or markets: where the money is.

Competitiveness

Next, competitiveness. Consider the differentiation and cost position of the product in question. Differentiation does not merely mean different.

(It's easy enough to have a different product—just paint it yellow.) Differentiation refers to the perceived improvement of the new product. In other words, do consumers believe the new product makes some tasks much faster, or makes the work cheaper or easier? Or does it enable external customers to look up their own accounts and save internal clerks from doing it? Does it provide some new quality customers value? Or can it provide some advantage—streamline operations, reduce costs, comply with the law, etc.—that has value to the project sponsor?

For software for sale, differentiation means the improvement or benefit offered by your product over your competitors. Think about the advantage to the consumers if they buy your product instead of someone else's.

Differentiation has all to do with consumers' perception of your product. If the consumer is willing to pay more (or alternatively the price stays the same but more people are willing to buy it), you have achieved differentiation.

The other aspect of competitiveness is economic cost position. Can you produce the same product with a better cost advantage? For instance, Amelia can go through an analysis to see how to run her shop to obtain an advantaged cost position. Maybe if she outsources the photo lab instead of having her own in-house printing, she can offer prints at lower cost than the competition.

We strongly suggest some combination of project sponsor, intended buyer, and operational user evaluate competitiveness. Unless you are a very unusual project manager, you tend to overestimate the differentiation and underestimate the cost of your product. After all, it's your project and you want it to be seen as valuable. You need a more objective assessment of competitiveness than you can normally provide by yourself.

Figure 2.3 looks at the desirability of investing. Note that if your organization is building products or services for sale, then the "worth of work

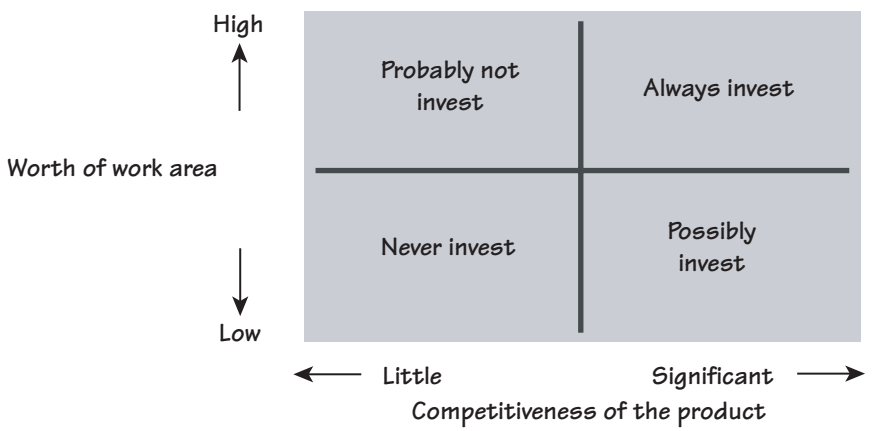


Figure 2.3

The vertical axis shows the worth of the work area relevant to the delivered product. The horizontal axis is the degree of competitiveness, improvement over the existing or competing product, the proposed product delivers. This graph is adapted from work by John and Ken Favaro.

area" scale should be replaced by "Attractiveness of the market." Is it a high-yield market with good margins and high demand—video games, software, DVDs? Or is the market soft and unattractive with low margins and unprofitable sales?

The obvious project that warrants investment is building a product likely to be deployed in a work area that has a high worth to the organization, and the proposed product is more competitive (produces increased profits) than its predecessor or competition.

When the product is to be used in a work area of high worth but the product has low competitiveness, investment is not advisable. From the point of view of investing in a product for your own organization, people working in the high-worth area are probably bright (look at your own organization to see where the best and brightest work) and you will not gain by giving them a product that provides little competitive advantage.

From the point of view of an external market, a company in a bad competitive position is likely to be unprofitable even in a market where the average participant is profitable. For example suppose that Amelia assesses that the Internet photo printing market is highly attractive. However, she is horribly disadvantaged competitively because she only has a 56Kb dialup telephone line and an old desktop printer for printing photos. Even in this profitable market, her competitive position makes it inadvisable for her to invest.

The product that offers significant competitiveness in the low-worth work area is a better investment, as it may make some noticeable difference to that area—perhaps enough for the area to become more valuable to the organization.

You can consider this from the point of view of an external market. Suppose Amelia wants to invest in running photography classes. Her analysis shows the average camera store does not make money in this market. However she has a huge competitive advantage because her uncle, who is a famous photographer, is the teacher. So everybody comes to her store's class and she's the only profitable participant in an otherwise unprofitable market.

Regardless of whether you are analyzing an investment in an internal work area or an external market, competitive position generally carries more weight than worth of the work area.

Investing in Requirements

Having determined it is advisable to invest in a product, you must now make the same decision about whether you will invest in gathering requirements for that product. Given that requirements accelerate the delivery of the product, the answer is generally yes, but there are degrees of

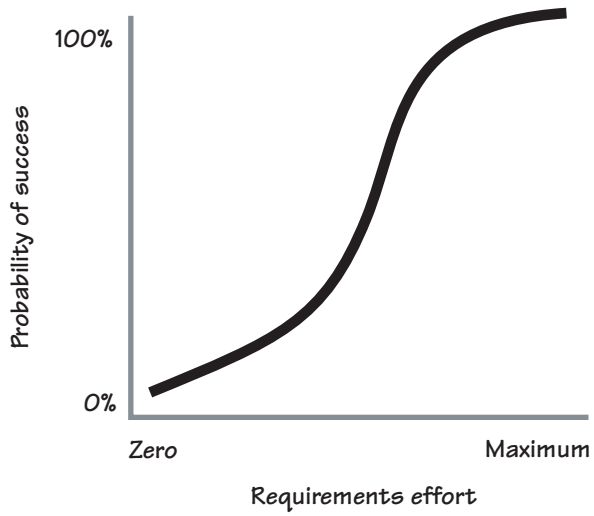


Figure 2.4
If you invest zero effort in requirements, your chance of success is also zero. Probability of success rises as you invest more and more in requirements. The maximum amount you can invest is close to half of your project budget.

No successful projects result from activity in which no effort was devoted to requirements . . .

requirements gathering. The effort you put into your requirements activity, and the formality of the requirements specification, should be varied to suit the project.

Figure 2.4 looks at the probability of success for your project measured against the effort put into requirements. No successful projects result from activity in which no effort was devoted to requirements, so we start at zero effort/zero chance of success. After that the probability of success rises as the requirements effort increases. However, at some point extra effort brings less incremental improvement in the success probability. The inference of this is that the requirements effort to be expended in the project is not automatically the maximum, but should be selected by considering the project's variables.

For large projects, the decision is always to invest in requirements and to budget for an extensive requirements activity. The magnitude of the requirements effort usually depresses project managers, but the ROI is huge. Without a proper requirements activity, likelihood of failure is high, and cost of failure is also very high. Large projects always make the headlines by canceling when it becomes obvious to all concerned the project cannot deliver. And almost always, at the time of cancellation, too much code exists and too few requirements.

Conversely, if your product is very small, a better chance exists for developers to successfully guess some of the requirements. The cost of failure here is not too large, as requirements errors can be corrected relatively cheaply.

Similarly, for an infrastructure product, the requirements are better known to the analysts and the developers, and subsequently a less for-

mal requirements effort is needed. The ROI for requirements for an infrastructure product is less than on strategic products, but still significant.

The worth of the work area also comes into consideration. For high-worth work areas, the cost of failure is high, usually because of lost opportunities to that work area. The cost of lost opportunities alone makes it worthwhile investing in requirements.

For strategic products, always invest in the requirements.

For strategic products, always invest in the requirements. The risks of a strategic product are far higher; in some cases organizations are betting their future on the success of the project. The cost of failure is thus extremely high, and the likelihood of failure without good requirements is also very high.

We always advise our clients, before deciding to invest, to take into consideration the value of the resources that the project will use. If the project uses valuable resources (assuming all resources are not equal) then failure is expensive as these resources can be used elsewhere.

The final consideration whether to invest in requirements is the inevitable question of how close is the deadline? Apart from trivial products, delivery is always fastest when developers have an intimate knowledge of the requirements. Always.

Size of Requirements

In Figure 2.2 we use "investment" as an axis of the diagram. This refers to the amount of the investment that you are about to make. The investment amount can be an amount of money or an amount of effort. Alternatively, you can also think of the size of the business area you are about to study.

When you think of an investment amount, an obvious correlation occurs between effort and money. But also a relationship exists between the size and complexity of the work area (or product) and the effort needed to determine the requirements. In Chapter 8—Measuring Requirements we talk about measuring the size or functionality of the work area.

The amount of investment follows the size of the task. But it can be reduced by factors such as:

- The experience of the analytical team
- Availability of the stakeholders
- Experience of the stakeholders
- Extent of the documentation of current work
- Existence of reusable requirements
- Skill and involvement of the testing team
- A template that has been customized for the organization
- And other factors unique to your organization

The point we wish to stress is you must know the size of your investment before proceeding. When we talk about the value of requirements we are really talking about ROI. Naturally, it is crucial you know with a fair degree of accuracy how much you have to spend before calculating any return.

The Value of Requirements

We have discussed the idea of investing in requirements, therefore the project. As a requirement is a demand for something to be built, we also discuss the idea of whether it is worthwhile investing in building a solution to a requirement. In *Mastering the Requirements Process,* the authors referred to collecting the requirements as "trawling."

Trawling is a peculiarly British word that refers to deep-sea commercial fishing. The boats, or trawlers as they are known, use large nets, sometimes incredibly large, to drag through the oceans for their catch. The idea of running a net through the ocean for fish is analogous to dragging a metaphorical net through the organization to gather up requirements. By using a net, a fisherman catches species of fish other than what he wants. Similarly, the diligent requirements analyst dredges up not only trivial or low-value requirements but also more requirements than can be built in the time allowed.

Not all requirements are of equal value to the organization. This makes it worthwhile to consider the value of each requirement before deciding whether or not to build it. We refer to this as *customer value,* with the inference that you ask your customer "how important is this requirement to you?" Keep in mind if you ask a customer to grade a requirement as high, medium, or low, the customer will tend to grade almost all the

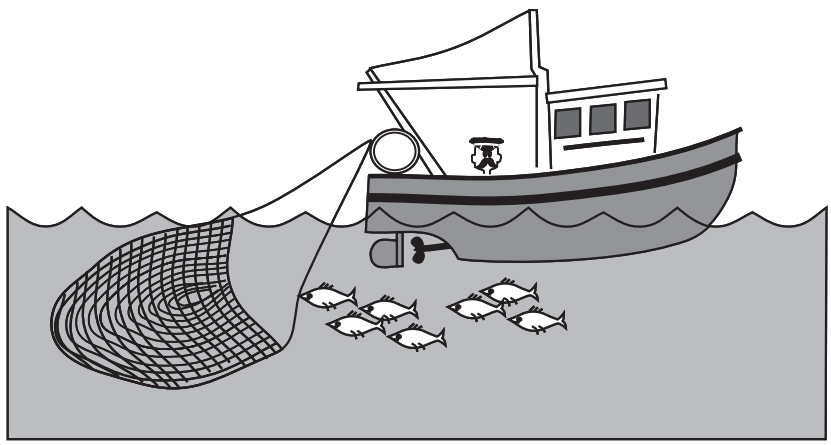


Figure 2.5
Trawling for fish has similarities to trawling for requirements.

requirements as high importance. Such is human nature—if I tell you that everything is of high importance then I believe I will get more of my requirements implemented regardless of their importance.

Another thing that fights against grading a requirement as high, medium, or low is the expectation embedded in the term "requirement." If I call something a requirement the implication is I definitely need you to give it to me. Within systems engineering we have a different meaning for the term—in fact, it would be more accurate to talk about *wishes*.

A requirement is something somebody wishes to have implemented but no guarantee exists it can be implemented until we know all the requirements and can determine which ones can be implemented within the constraints. Given this reality, it makes sense to make requirers aware they will need to make choices and to provide them with a mechanism for considering choices early.

A requirement is something that somebody wishes to have implemented but no guarantee exists it can be implemented until we know all the requirements and can determine which ones can be implemented within the constraints.

We have adapted William Pardee's idea of *Customer Satisfaction* and *Customer Dissatisfaction*[4] as a way of helping people to consider the relative importance of their requirements.

Pardee suggests you ask two questions about any requirement:

"On a scale from 1 to 5 how satisfied will you be if I implement this requirement?"—where 1 means you don't particularly care whether the requirement is implemented and 5 means you will be extremely pleased if it is. (The scale of satisfaction.)

"On a scale from 1 to 5 how dissatisfied will you be if I do not implement this requirement?"—where 1 means you are unconcerned if the requirement is not part of the product and 5 means you will be extremely unhappy. (The scale of *dis*satisfaction.)

The idea behind Pardee's thinking is that if your customer has a large-enough continuous scale, he can give you a more accurate indication of importance. The notion of having two scales is even more helpful. For example, the customer considers some requirements to be natural or part of the existing system, and thus the customer sees no reason why they would not be contained in any future product.

At www.volere.co.uk you can download the Volere Atomic Requirement Template, which shows requirements attributes including customer satisfaction and customer dissatisfaction.

In the situation of "natural requirements" the satisfaction rating is likely to be low: the customer is not going to get excited if you deliver something already there. However, if you do *not* deliver the requirement, dissatisfaction is likely to be high. Conversely, if a customer considers a requirement to be a trivial piece of gold plating, it will garner a low dissatisfaction rating.

4. Pardee, William. *To Satisfy & Delight Your Customer*. Dorset House, New York. 1996—*Contains many good ideas in the area of managing customer expectations. The notion of customer satisfaction and customer dissatisfaction is a useful tool for prioritizing requirements.*

We have found Pardee's way of measuring satisfaction and dissatisfaction superior to the commonly used scale of "must have, nice to have, fit it in if you get time."

A simple way of determining the future of a requirement is to add the satisfaction and dissatisfaction scales. Any requirement that has an aggregate score of four or less should not be implemented. Then, starting with the highest aggregate scores, implement as many requirements as you can in the allowed time. However, to make your implementation decision, weigh the customer value against the cost of implementing the requirement. In other words, just because a requirement has high customer value does not necessarily mean it will be possible to implement it. In Chapter 9—Managing The Requirements, we discuss the subject of progressive prioritization and prioritization techniques.

Reusing Requirements

You are considering investing in requirements with the objective of a payback on investment. A final consideration is whether the requirements have a value after they have been used on your current project. To put that another way, can you think of the requirements you gather as business assets? They deliver a further payback on your investment when you recycle them on subsequent projects.

Think of the requirements you gather as business assets.

You are probably not reusing only one or two requirements—no justification exists for doing so. However, if we look at the idea of clusters of requirements, then reuse becomes much more viable. Earlier in this chapter we mentioned the idea of clusters, or units of requirements (work context, product context, business use cases, product use cases, functional requirements, nonfunctional requirements, constraints) as an investment vehicle. Figure 2.6 identifies these units of potentially reusable requirements and the associations or relationships among them.

Requirements reuse depends first on the ability to identify reusable requirements, and second on having requirements reuse as part of your requirements process. Our experience shows many projects have significantly overlapping requirements. By abstracting and looking at the functionality, and for the moment ignoring its subject matter, we have been able to identify whole clusters of requirements in which simple subject-name changes have yielded significant portions of a functionally correct specification.

We have also learned many projects can save considerable time by intentionally looking at existing requirements documents. The analyst trawls through existing specifications looking for requirements that can be reused. It helps to ignore the names of the objects being manipulated, and concentrate instead on the abstract functionality.

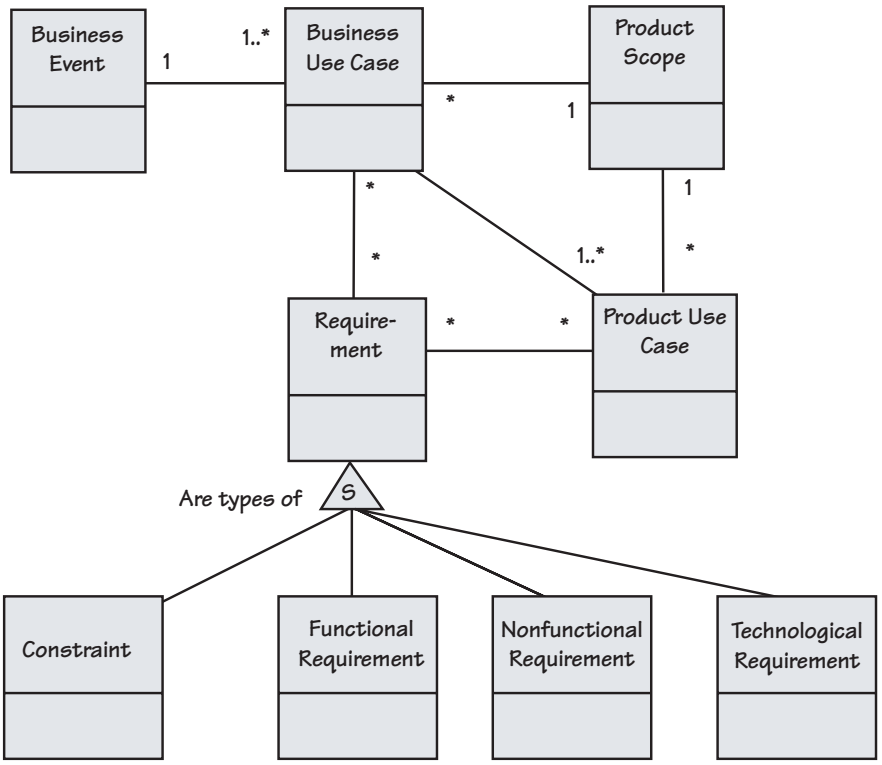


Figure 2.6

This class model shows units of requirements that are potentially reusable. The * indicates that many instances exist of the entity at that end of the association. For example, a *Product Scope* may have many *Product Use Cases* but the latter belongs to a single *Product Scope*. The triangle shows that functional requirement, nonfunctional requirement, and constraint are all types of requirements.

Identifying Reusable Requirements

Even if you have not been consciously producing reusable requirements you are sure to have many potentially reusable requirements already existing in your organization. The chart shown in Figure 2.7 suggests some of the places you might find them. Use it as a checklist to help take advantage of reuse opportunities.

For example, the checklist identifies a dozen potential sources of reusable functional requirements and six potential sources of constraints. There will be others within the documents and models produced by your organization. The chart can help you get started. Your organization should update the list when discoveries take place, for the benefit of future requirements projects. The more consistent your requirements deliverables, the more likely you can reuse them as requirements on another project.

Your Process for Reusing Requirements

The best advice we can give you is to start each of your projects with a stock-take and use potentially reusable components as a starting point.

Figure 2.7 This chart summarizes potentially reusable requirements components along with suggestions for where to find them. Its intended use is as a checklist of reuse opportunities. The more progress you make toward formalizing reuse, the more consistent items in the *Where to Look* column will become. You should update the chart with places from your own environment.

	Reusable Requirements Components							
Where to Look	**Work Context**	**Business Event Response**	**Product Context**	**Product Use Case**	**Functional Requirement**	**Non Functional Requirement**	**Constraint**	**Design Requirement**
Project Goal Descriptions					×	×	×	
Work Context Model	×	×			×			
Business Task Descriptions	×	×			×	×	×	
Business Process Models	×	×			×			
Business Scenarios	×	×			×	×		
Business Data Models	×				×			
Requirements Process Patterns	×	×			×			
Requirements Data Patterns	×	×			×			
Product Context Models			×					
Product Use Case Models			×	×				
Product Scenarios			×	×	×	×		
Product Sequence Diagrams			×	×	×	×	×	×
Product Class/Data Models					×			×
Atomic Requirements Definitions					×	×	×	×
Technical Architecture Models							×	×
Design Patterns							×	×

This does not have to turn into a bureaucratic process. It simply means spending some time (maybe a few hours) looking around before starting detailed work. We suggest drawing a quick work context diagram of your project. Then within this scope look for potentially reusable requirements components using the chart in Figure 2.7 as a guide. The questions to ask are:

- Does the component contain any of the same data bounded by the interfaces on my project's work context diagram?
- Does the component refer to any of the adjacent systems mentioned on my project's context diagram?
- Does the component contain any business rules that contribute to my project's goals?

For each of the potentially reusable components you identify, consider whether it will save you time by giving it the *essence test*. This test asks what percentage of the component exists because of essential business rules versus what percentage exists because of a particular solution to meeting the business rules.

For example, one of us, Suzanne, undertook a project with an insurance company. The project team wanted to reuse the business data model. However when we applied the essence test we discovered that it was a business data model in name only. It had been partitioned according to a particular database implementation technology and contained many attributes such as keys and pointers that existed because of the implementation.

Because of that unique implementation, the model was very difficult to reuse: business requirements data (claims, premiums, policies, bonuses, rates) were extremely fragmented and confused as a consequence of the number of implementation attributes. It was quicker to start from scratch—this annoyed the business experts because we had to ask them questions the other project group had already asked. If the

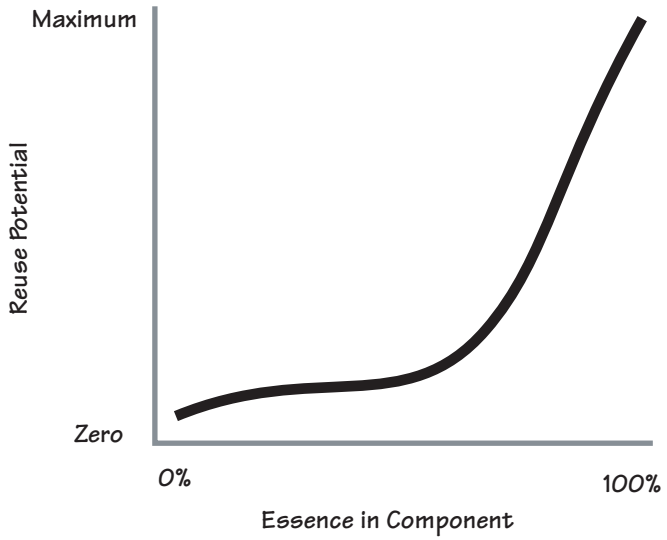


Figure 2.8
The essence test evaluates the percentage of the content of a component related to the essential business within the domain. The higher percentage of essence, the more likely the component is reusable.

business data model had truly been a specification of business requirements rather than a specific implementation, we could have reused the business knowledge it contained.

Components containing less than 60% of essence are probably going to be too implementation-specific for reuse. You would spend too much time removing the implementation details and reorganizing them so you could figure out their meaning in terms of the business requirements. For more on how to discover the essence of a system, please read *Essential Systems Analysis* by Steve McMenamin and John Palmer[5]. Also, please refer to Chapter 4—Learning What People Need, which contains more on essence.

The approach of considering requirements reuse at the start of a project leads you toward treating requirements as continuing business assets rather than things that only apply to one project.

What Do I Do Right Now?

A number of the project success indicators (see Chapter 1) are affected by requirements value:

- No excessive schedule pressure
- Correctly sized product
- Product adds value to the organization

Determine the value your project adds to your organization and the value requirements add to your project. According to studies carried out by the Standish Group, about three-quarters of software projects deliver late or not at all. The main three reasons are lack of user input, incomplete requirements, and constantly changing requirements. These handicaps are eliminated, or significantly reduced, by a competent requirements process.

So long as your project is not trivial, you are going to invest in requirements. However, the adroit project manager takes into account the uniqueness of the project and the degree of fragmentation of information and people, then determines how much effort needs to be expended on requirements.

Estimate the effort needed by measuring the work area to be studied. We suggest function points as an effective way of doing this. From the

5. McMenamin, Steve and Palmer, John. *Essential Systems Analysis*. Yourdon Press, New York. 1984—*This classic work explains the difference between the implementation and the real problem (referred to as the essence). Full of examples on how to arrive at the essential view of a system and thereby understand the real requirements.*

size of the work area—or product if that is what you measured—determine the amount of requirements analysis needed. Capers Jones of Software Productivity Research puts the US average cost of installing a function point at $1,000. Requirements activities should consume between 30 and 50% of that cost. We discuss function point counting in Chapter 8.

Adjust this number by the nature of the project—infrastructure, strategic, large, small, and so on—to determine whether your requirements activity is going all out to produce an absolutely complete requirements description of the product, or whether the risks involved in doing less than the maximum can be tolerated.

What's the Least I Can Get Away With?

A little bit of honesty. After you have read this chapter, ask yourself:

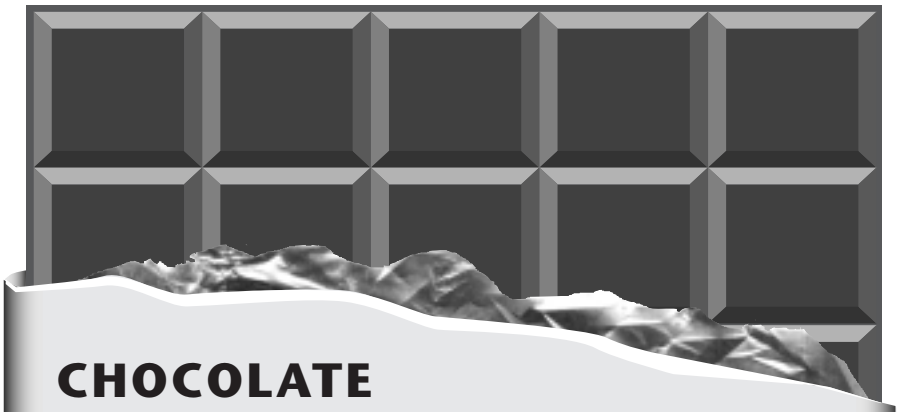
- Does the product I am about to build add value to the organization?
- How do I know it will add value?
- Does my project contribute to the overall goals of the organization?
- Is the project worth investing in? Does it create a product that has significant competitiveness for a high-worth work area?
- Do I know what it costs to discover the requirements?
- How do I know the costs? (If you are guessing then you are doing less than the "least you can get away with.")
- What is my break-even point—when the cost of reworking the product is less than the cost of investing more in the requirements?
- What return do we expect on the investment, and when?
- Have I honestly considered the risks associated with this project?
- Are there any risks that require some preventive actions?
- Do we already have some requirements that are reusable by this project?

These questions may seem obvious, but to start a project without having satisfactory answers for all of them is flirting with alligators.

Additional References for Requirements Value

The following sources, in addition to those already referenced, have useful information on requirements value:

- Favaro, John. *Managing Requirements for Business Value*. IEEE Software, March 2002—*Discussion of the value of reusing requirements and the advisability of involving business strategists in the process*.



CHOCOLATE

Perhaps the most intriguing investment story is of J.K. Rowling, renowned author of the Harry Potter stories. Her investment was the time, and not-especially encouraging time, she spent writing the first of her Harry Potter books. Unemployed at the time, she was living in a mouse-infested flat in Edinburgh when she started writing. It is a tribute to her belief that her investment in writing was not going to be wasted, as reports suggest that 14 publishers rejected the manuscript before Bloomsbury Publishing paid her the equivalent of about $4,000 US to publish *Harry Potter and the Philosopher's Stone* in June of 1997.

Today the five Harry Potter books have sold 250 million copies, and on the day we write this, J.K. makes her debut on Forbes magazine's billionaire list.

A good investment of effort.

- McConnell, Steve. *Professional Software Development*. Addison-Wesley, 2004—*McConnell's words on return on investment are alone worth the price of this excellent book.*
- Thomsett, Robb. *Radical Project Management*. Dorset House, 2002—*Packed with tools and hints for making project management decisions in a changing environment.*