The New ROI

2

If software development is to be treated as a value creation exercise, a solid understanding of the financial metrics used to evaluate and track value creation activities is necessary. In this chapter we define these metrics and show how they are impacted by the introduction of MMFs and incremental delivery concepts.■

Applications and ROIs

In the world of commercial application software development, good ideas, technology, and design are rarely sufficient to elicit approval for a project. The reality of the commercial world is that software development is an investment. As with any investment, it involves certain risks and is made with the objective of achieving a return on that investment.

Of course, the return on that investment does not have to be in quantifiable financial terms. But usually, unless there is a measurable financial benefit, it is hard to justify an application software development activity, and especially hard to obtain approval from a finance department for that activity. Troy Zierden, business-intelligence capability manager at electronics retailer Best Buy, says it's very difficult to get project approval when returns on investment are intangible, since there's a team of accountants at his company who watch the numbers very closely[1].

Today's software methodologies tend to begin at the point where the development project has been approved, that is, after the ROI discussions have taken place and conventional methods of measuring ROI have been applied. The software designer, or the software methodologist, is usually not involved in the ROI discussions. As a result, the reasons for funding a project are largely opaque to the designer. Sadly, this means that the designer rarely has any input on the ROI discussions. Our approach changes this, by breaking the black box of application software development, and engaging architects and designers in the investment that we call software development. The result is a radical transformation of the traditional ROI model.

Why ROIs Matter

Let's imagine that you, or your client, want to construct an electronic bill payment system for a retail bank. You estimate that it will permit customers to pay bills two to three times faster than via an ATM and ten times faster than standing in line to present a check payment to a bank teller. You intuitively know that this system will make your customers happy and more loyal to the bank. You're excited about writing it and you can certainly justify why you should do so in terms of customer loyalty and ease-of-use benefits.

Unfortunately, none of these reasons or motivations will cut it with the finance department! Although the benefits sound interesting, and perhaps even compelling, they are not in a measurable form. What the finance people are looking for is a number, a dollar figure, to which those benefits translate for the company's bottom line. Finding that figure isn't as easy as it might seem. But it may be one of the most important decisions to be made in the lifecycle of this software development project; one so crucial that it could actually halt the project completely or cause it to be stillborn. This is neither the time nor the place for the developer to fade into the background. It's often here that the success or failure of an application software development project is ultimately determined. And it's not as though financial approval is going away or becoming less important. Over 80% of IT managers surveyed in 2001 reported that the importance of ROI has increased compared to the previous year[2].

Dataquest reports that in these "trying times . . . IT investments are being looked into more closely than ever before." They suggest that the best strategy involves finding a business sponsor to back the investments, and that without this type of sponsorship it becomes very difficult to justify the costs of implementing the project[3].

The Business Case

Ultimately the calculation of ROI compares the financial impact of different options over time. The time context is essential. Without it, an ROI is meaningless. Typically it starts with a question of the form,

> "Should we spend $1,000,000 to develop part of the system over two years or $1,500,000 to develop all of the system over three years?"

The answer to a question such as this emerges from the construction of a business case. A business case is best described as a financial story based on facts, structured assumptions, and logic. It provides a vehicle by which the financial impact of the options can be examined and conclusions drawn.

Cash Flow Projections

At the heart of a business case is a cash flow projection. It is a sequence of calculations of a specific financial position over time. To construct a cash flow projection, we need to look at the net financial situation at each of the time intervals. Cash flow projections are typically calculated on a monthly basis, though for longer projects they may be calculated quarterly. The key thing about cash flow projections is that for each calculation point, all the costs and all the benefits have to be factored in as they occur.

Let's look at an example. Imagine that we're considering an application development project to create a customer relationship management (CRM) system that is intended ultimately to eliminate a call center. How do we quantify the impact in cash flow terms?

Clearly the major benefit is the removal of the costs associated with the call center. But for a cash flow analysis we need to know when those costs are taken out of the equation. It's also important to understand what the components of the savings are. There are, for example, the operational costs of the center, the possible capital gain from the sales of the land, buildings, or lease, and perhaps other savings associated with facilities management.

On the benefits side, we need to quantify factors such as the increased customer satisfaction created as a result of a more responsive CRM experience. Other factors include the additional revenue achieved by attracting new customers because of our reputation for processing customer calls faster.

In the cash flow analysis, these positive and negative factors are computed for each period in the analysis. By summing these factors, we can determine

whether the project is cash-positive or cash-negative over its development lifecycle.

Payback Time

The lifecycle cash position isn't the only information we can derive from a cash flow analysis. A cash flow picture also indicates the extent to which the business is investing in the project at each period in the analysis. It quantifies the "investment flow" if you will. Normally, as a project starts to return revenue, this investment flow is reduced and will likely become zero prior to the end of the project. At this point the project is said to have become "self-funding" or to have reached "self-funding status." It no longer needs cash injections from the business to sustain it.

This does not mean the project has reached a breakeven point, however, as there may still be a debt to repay to the business (see discussion of breakeven time below). We will call the period between self-funding status and breakeven status the "repayback period." Figure 2.1 illustrates these points over the timespan of a successful application development project.

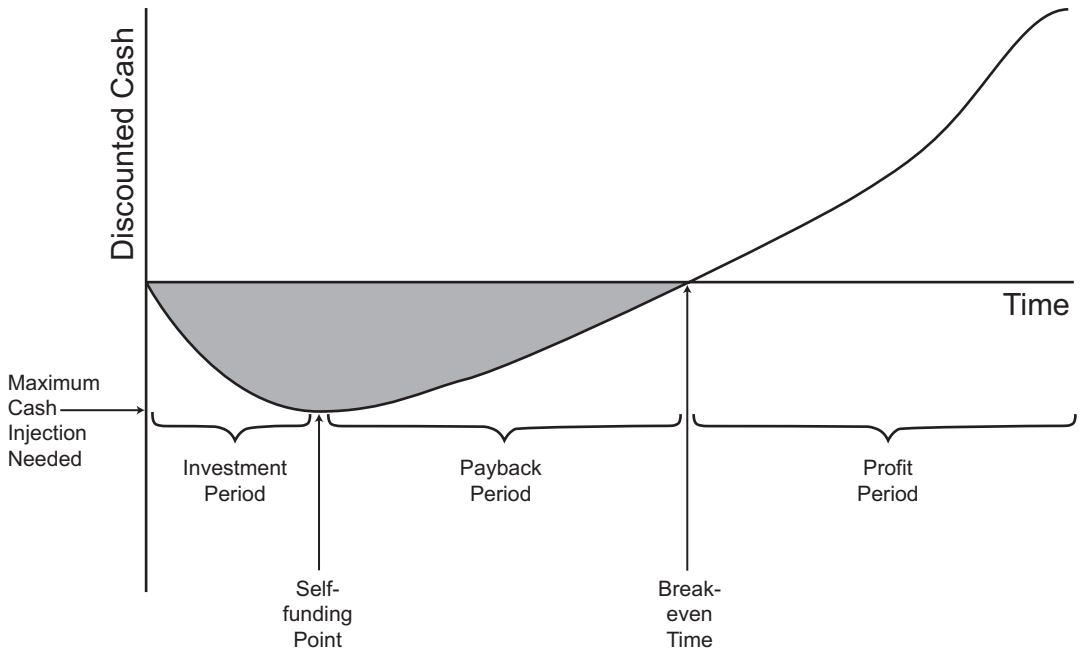

Figure 2.1
A Successful Application Development Project

Present Value of Future Money

To recap, conventional ROI analysis is about measuring the amount of the payback and the time in which is it achieved.

For example, a project that becomes profitable in 18 months is intuitively more compelling than a project that takes five years to become profitable. However, it may not be financially more compelling over the long term. It's possible that the return from the five-year project is significantly larger than the one from the 18-month project. Under these circumstances, how is it possible to decide if the five-year project is to be preferred over the 18-month project?

Clearly money has a time value. A piece of software that delivers $1 million in savings in one year is more interesting than a piece of software that delivers $1 million in savings in 20 years. So how do we compare the value of $1 million next year with $1 million in 20 years?

To some extent the value of future cash can be measured by discounting it against an assumed interest rate. This calculates the present value (PV) of the future cash. This approach is clearly simplistic because it fails to take into account risk factors associated with the predicted future cash, but more on this later. For now, we'll assume the future cash is certain. As an illustration, imagine that the interest rate is i%. The present value of $x in *n* years' time is defined as follows:

$$PV = \$x \,/\, (1 + i/100)^n$$

In other words, if we assume an interest rate of 5% per year, receiving $1 million in 20 years is equivalent to gaining $1/(1 + 0.05)^{20}$ = approximately $377,000 now. On the other hand, receiving $1 million next year is equivalent to gaining approximately $952,000 now.

Net Present Value

We can create an overall figure for the net cash position of the project by calculating the cash position for each time period in the development cycle (month, quarters, or years) and then summing the PV corrections. The set of PV-corrected cash positions is known as the discounted cash flow (DCF). The sum of these positions is known as the net present value (NPV). An example of an NPV calculation is given shortly.

Because the NPV allows us to measure the overall value of the software development, even if its returns are spread over a period of time, DCF tends

to be a more useful measurement of a software development project's projected costs and returns than either net cash or ROI. In general, when we refer to the cash position we'll be implying DCF. It is for this reason that Figure 2.1 reflects DCF rather than just a straight cash flow position. This will become especially important as we look at the effect of iterative software development approaches and the order in which MMFs are executed.

Breakeven Time

An important metric that emerges from a DCF calculation is the "breakeven time," or the point in the lifecycle at which the project reaches "breakeven status." This is the number of periods (e.g., months, quarters, or years) before the return from the new software, corrected for time, matches the costs expended to create it. In other words, it is the point at which the rolling NPV transitions from a negative value to a positive value. A project that has reached breakeven status is making real money for the business. We call this the "profit period." The breakeven time thus marks the end of the *payback period* and the start of the profit period.

Before the dot-bomb era, breakeven times of five years were not atypical. In today's market however, software developments with a breakeven time of more than 12 to 18 months are rarely approved.

Internal Rate of Return

Another factor in conventional ROI analysis is the internal rate of return (IRR). This is the interest rate, or cost of capital, at which the NPV for the project becomes zero.

The usefulness of IRR is a matter of some debate, so it is not a term we deal with extensively in this book. However, it is an interesting concept, for the following reason.

NPV alone does not give us enough information to answer the question "is this project worth doing?" If a particular development project has an NPV greater than zero, undertaking the project is certainly a better investment than doing nothing at all. However, it doesn't tell us anything about whether this is the best thing to be doing in comparison with other possible money-generating activities.

For example, if the application's NPV is positive only when interest rates are 6% per year or less, but we know we can get an 8% return from low-risk

bank deposits, the application development may not be worth doing. We would make more money just by depositing the money in a bank. The IRR, or the interest rate at which the NPV is zero, is thus an interesting metric in conventional ROI analysis and provides us with a useful yardstick for evaluating a positive NPV.

Summary of the Terms

Briefly, the terms introduced so far can be summarized as follows:

- Return on investment: The amount of undiscounted cash (profit) returned by the project over the lifecycle, divided by the undiscounted cash (investment) used to fund the project over the lifecycle, expressed as a percentage.
- Self-funding point: The number of time periods until the business no longer needs to inject cash into the project to sustain it.
- Present value: The value of future cash converted to the present time.
- Discounted cash flow: The cash flow with PV corrections applied to each period.
- Net present value: The sum of the DCF.
- Breakeven time: The time until the NPV of the project becomes positive (i.e., the point at which the project is making real money).
- Internal rate of return: The interest rate at which the NPV becomes zero.

Now let's see how this all works in practice.

An Example

In this example project we analyze a software development effort over five one-year periods.

An early release of the software allows revenue to start flowing in year 4, though in practice it's not until year 5 that the real revenue flow starts. On the cost side there are some early capital outlays associated with buying the hardware and software needed to do the development work. A technology refresh cycle in year 4 updates the development hardware and some of the data center facilities. In addition to the capital costs, there are operational costs related to personnel, support, data center facilities fees, and marketing.

In addition some savings arise from using the software internally within the organization. Figure 2.2 captures these figures. The numbers are $US in thousands.

Figure 2.2 shows that we have a five-year project that requires $2,760,000 to fund. The project generates cash in years 4 and 5. After five years, the total cash returned is $1,288,000. This represents a ROI of 1,288,000/2,760,000 = ~47% over five years.

We now turn to the PV analysis to examine the NPV of the project. The table in Figure 2.3 shows the calculations for three annual discount rates: 5%, 10%, and (for reasons that will be clear in a moment) 12.8%. We have used the standard NPV convention of assuming the cash is generated, or the cost incurred, at the end of each period. Values have been rounded to the nearest integer.

The discount rate at which the NPV is zero turns out to be 12.8%. By reference to Figures 2.2 and 2.3, we can draw the following conclusions about this application development project:

Description			Periods 1	2	3	4	5	Total
Income								
	Revenue					1,000	3,800	**4,800**
	Savings					200	600	**800**
Total Income			0	0	0	1,200	4,400	**5,600**
Expenditure								
	Capital							
		Hardware	700	20	20	200	20	**960**
		Software	300					**300**
	Total Capital		1,000	20	20	200	20	**1,260**
	Operating							
		Headcount	240	360	550	360	112	**1,622**
		Data Center	30	30	30	30	30	**150**
		Support	140	120	120	150	150	**680**
		Marketing	0	0	100	200	300	**600**
	Total Operating		410	510	800	740	592	**3,052**
Total Expenditure			1,410	530	820	940	612	**4,312**
Cash			-1,410	-530	-820	260	3,788	**1,288**
Investment			-1,410	-530	-820			**-2,760**
ROI								**47%**

Figure 2.2
Classic ROI Analysis ($US in Thousands)

Period		1	2	3	4	5	
Cash		-1,410	-530	-820	260	3,788	
Self-funding Status					X		
							NPV
DCF @	5.0%	-1,343	-481	-708	214	2,968	650
	Rolling NPV	-1,343	-1,824	-2,532	-2,318	650	
	Breakeven Status					X	
	10.0%	-1,282	-438	-616	178	2,352	194
	Rolling NPV	-1,282	-1,720	-2,336	-2,158	194	
	Breakeven Status					X	
	12.8%	-1,251	-417	-572	161	2,079	0
	Rolling NPV	-1,251	-1,667	-2,239	-2,079	0	

Figure 2.3
NPV Analysis ($US in Thousands)

- The project generates $5.6 million over five years, at a total cost of $4.3 million.
- The business invests a total of $2.76 million at various points over that period to fund it.
- The project pays back that investment and returns an additional $1.288 million to the business after five years.
- The resulting ROI over that period is 47%.
- The project reaches self-funding status in year 4.
- Breakeven status is achieved somewhere in year 5.
- The NPV of the project, assuming an annual discount rate of 5%, is $650,000.
- The NPV of the project, assuming an annual discount rate of 10%, is $194,000.
- The IRR of the project is 12.8%.

Despite the apparent attractiveness of the ROI, the IRR indicates that it's only worth undertaking this development project if we're unable to find other uses of investment capital yielding better than 12.8% over five years.

If the project is perceived to be risky, this rate of return is probably insufficient to persuade a CIO to proceed. An alternative but lower-risk use of the money with just (say) a 10% rate of return may be thought more attractive.

The bottom line is that at this IRR and in today's market, the project will probably not get approved.

Incorporating MMFs into the Financial Case

Now let's take a look at that funding model again. As it stands currently, it's a classic ROI scenario in which we invest the majority of the capital up front and achieve a return only at the very end of the lifecycle. Intuitively, this is unsurprising. The application clearly has to be designed, written, integrated, tested, and packaged before sales of the software result in a revenue flow to offset the development costs.

But suppose it were possible to generate revenue earlier? How would this impact the ROI model?

At first, these may seem like theoretical questions. After all, how can revenue be released before the development work has been completed?

However, imagine we were able to separate the application into groups of features that, although they represented just a subset of the overall application feature set, were still inherently marketable. These MMFs would by definition be capable of creating revenue when released independently and incrementally.

We explore the concept of an MMF and discuss how use cases or user stories can be composed into an MMF in Chapter 3.

For now, let's assume that the total functionality of this application can be partitioned into four distinct MMFs. We'll imagine that we can develop the MMFs sequentially, aiming to bring one to market each year in years 2, 3, 4, and 5 of the project. For the purposes of this example we'll assume each MMF is equally valuable.

In year 2 we get a small revenue flow from MMF 1. In year 3 we get revenue from MMFs 1 and 2, and so on.

There are clearly additional costs associated with packaging and releasing these early MMFs, so this needs to be taken into account in the financial model. There are also additional headcount costs, because of the additional testing at the MMF level.

Taking these into account, and again using broad-brush estimates only, the returns of our modified project are shown in Figure 2.4.

The PV analysis, using the same conventions as previously, reveals the modified NPV of the project as shown in Figure 2.5.

The incremental delivery approach produces a development project that now looks like this:

- The project generates $7.8 million (vs. $5.6 million) over five years, at a total cost of $4.712 million.

Description			Periods 1	2	3	4	5	Total
Income								
	Revenue			700	1,400	2,100	2,800	**7,000**
	Savings					200	600	**800**
Total Income			0	700	1,400	2,300	3,400	**7,800**
Expenditure								
	Capital							
		Hardware	700	20	20	200	20	**960**
		Software	300					**300**
	Total Capital		1,000	20	20	200	20	**1,260**
	Operating							
		Headcount	240	360	550	360	112	**1,622**
		Data Center	30	30	30	30	30	**150**
		Support	140	120	120	150	150	**680**
		Marketing	200	200	200	200	200	**1,000**
	Total Operating		610	710	900	740	492	**3,452**
Total Expenditure			1,610	730	920	940	512	**4,712**
Cash			-1,610	-30	480	1,360	2,888	**3,088**
Investment			-1,610	-30				**-1,640**
ROI								**188%**

Figure 2.4
ROI Analysis Using MMFs ($US in Thousands)

Period		1	2	3	4	5	
Cash		-1,610	-30	480	1,360	2,888	
Self-funding Status				X			
							NPV
DCF @	5.0%	-1,533	-27	415	1,119	2,263	**2,236**
	Rolling NPV	-1,533	-1,561	-1,146	-27	2,236	
	Breakeven Status					X	
	10.0%	-1,464	-25	361	929	1,793	**1,594**
	Rolling NPV	-1,464	-1,488	-1,128	-199	1,594	
	Breakeven Status					X	
	36.3%	-1,181	-16	190	394	614	**0**
	Rolling NPV	-1,181	-1,197	-1,008	-614	0	

Figure 2.5
NPV Analysis Using MMFs ($US in Thousands)

- The business invests a total of $1.64 million to fund it (vs. $2.76 million).
- The project returns that investment and pays back an additional $3.088 million at the end of five years.
- The resulting ROI over that period is 188% (vs. 47%).

- The project reaches self-funding status in year 3 (vs. year 4).
- Breakeven status is achieved during year 5.
- The NPV of the project, assuming an annual discount rate of 5%, is $2.236 million (vs. $650,000).
- The NPV of the project, assuming an annual discount rate of 10%, is $1.594 million (vs. $194,000).
- The IRR of the project is 36.3%.

Comparing the MMF-based ROI with the Classic ROI

Clearly, the impact of using incremental delivery of MMFs in this example transforms the project. It costs just a little more to undertake the project, but the impact in terms of overall revenue, five-year ROI, and the IRR is attractive. The revenue over five years is higher primarily because it starts earlier. However, for the purposes of this simplified analysis we have assumed nothing about changes in marketability over the five years. With an IRR of ~36%, this now appears to be an excellent way to deploy the investment capital over five years.

The self-funding time is particularly significant. Now we have a project that is self-sufficient in cash terms as early as year 3. We will expand on this idea of self-funding development projects in later sections.

Taking the Risks into Account

This financial analysis has by necessity been oversimplified. Several variables were not taken into account. One of the most important of these is risk.

In general, incremental delivery has the benefit of reducing overall project risk. But for this to be both demonstrated and measurable, it's necessary to implement a mechanism for taking risk into account in an MMF context.

The quantification of risk has an immediate impact on the perceived value of an MMF and must therefore be factored into the financial equations. We may decide, for example that an MMF yielding $1 million in two years' time at 90% risk is less valuable than one yielding $500,000 in two years' time at 10% risk. Of course, this raises the important question of what is meant by 90% risk and 10% risk.

There are many approaches to handling risk assessment in complex projects, and in fact much has already been written on the topic. The approach that we adopt is more fully explored in Chapter 3.

The Impact of MMF Ordering

In the previous simplified example, there were only a few MMFs, and they were easy to identify, were all equally valuable in revenue terms, and all took the same amount of time to develop. In this idealized scenario the question of which MMF to develop first does not arise.

In the real world, things are usually quite different. The task of grouping use cases to create MMFs is to some extent both an art and a methodological process. Furthermore, the order in which MMFs are developed can radically affect the financial model of the project. At the very least, it can determine whether the project is self-funding or not.

This subject is explored in the next chapter.

Summary

- Analysis of the financial context for a software development project requires an understanding of breakeven time, net present value, and internal rate of return.
- Introducing MMFs impacts these financial metrics, usually for the better.
- MMF ordering, MMF risk evaluation, and the parallelization of MMF sequencing all determine the extent to which those financial metrics are impacted.

References

1. Sandra Swanson, "Business Intelligence—Can't Live Without It," *Information Week*, March 13, 2002.

2. Mary Hayes, "Payback Time: Making Sure ROI Measures Up," *Information Week*, August 6, 2001.

3. "It's All About ROI," *Dataquest CIO,* January 10, 2002.