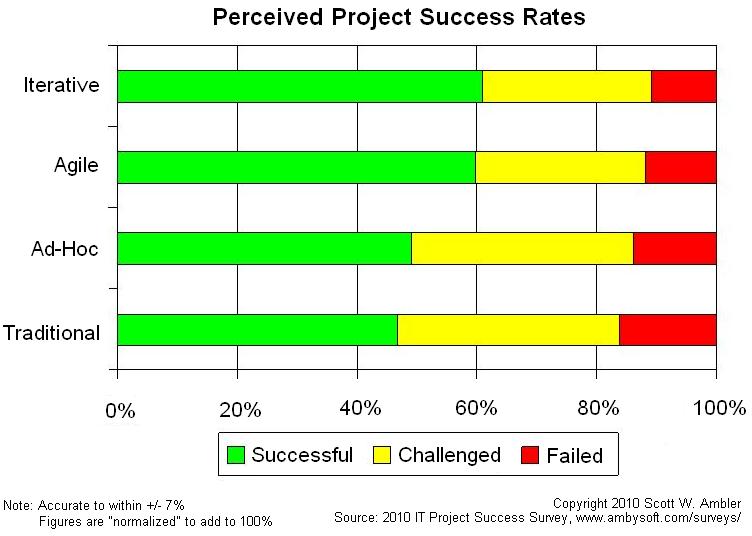
|  |  |
| --- | --- |
| **Justifying a Software Development Project** (Scott Ambler + Associates, 2006-2014) | [[Scott W. Ambler + Associates](http://scottambler.com/)](http://ScottAmbler.com/) |
|  | |
| Top of Form    Bottom of Form | |

|  |
| --- |
| Top of Form  [Home](http://www.ambysoft.com) | [Articles](http://www.ambysoft.com/onlineWritings.html) | [Books](http://www.ambysoft.com/books/) | [IT Surveys](http://www.ambysoft.com/surveys/) | [Podcasts](http://www.ambysoft.com/podcasts/) | [Contact Us](http://ScottAmbler.com/) | [Announcements](http://www.ambysoft.com/mailingList.html) | [Site Map](http://www.ambysoft.com/siteMap.html)  Bottom of Form |

|  |  |  |
| --- | --- | --- |
| [Recently reviewed](http://www.ambysoft.com/books/reviews/current.html) | Part of initiating a software development project is to do a reality check to determine whether or not the project even makes sense. Because of the [success rate on IT projects isn't 100%](http://www.ambysoft.com/surveys/success2010.html) (see [Figure 1](http://www.ambysoft.com/essays/projectJustification.html#Figure1)), the implication is that some projects should end at this stage, long before a large investment has been made (and then lost) attempting to build them. Your main goal should be to define the best implementation solution for your project, if any, and justify why it is best.  Many projects tend to skip this effort, either because they are too rushed for time or because competition within your market segment places demands on your organization to create the application. Nevertheless, even if you know that you are going to do the project it is often still worth your while to attempt to justify it. If the project makes sense then you have the ammunition that you need to argue your case when your project is called into question. If your project does not make sense, i.e., it looks like it is going to fail, then it is a very good indication that you should consider other alternatives (you do not have to automate everything). |  |

**Figure 1. Project success rates.**

[](http://www.ambysoft.com/surveys/success2010.html)

Project justification will either be one of the first steps of the actual project itself, or it will be a significant part of your [system portfolio management](http://www.enterpriseunifiedprocess.com/essays/portfolioManagement.html) efforts. Regardless, a key deliverable of this activity is a feasibility study. A feasibility study is important because it drives the development of your project proposal, which can be presented to senior management to gain their commitment to the project and to obtain project funding. Furthermore, during the creation of the feasibility study you will often identify risks associated with the project, providing valuable input into your risk management efforts. Your feasibility study should indicate:

1. [The implementation alternatives](http://www.ambysoft.com/essays/projectJustification.html#ImplementationAlternatives)
2. [The economic feasibility](http://www.ambysoft.com/essays/projectJustification.html#EconomicFeasibility)
3. [The technical feasibility](http://www.ambysoft.com/essays/projectJustification.html#TechnicalFeasibility)
4. [The operational feasibility](http://www.ambysoft.com/essays/projectJustification.html#OperationalFeasibility)
5. [The political feasibility](http://www.ambysoft.com/essays/projectJustification.html#PoliticalFeasibility)

**1. Identifying** **Implementation Alternatives**

The first step of performing a feasibility study is to identify potential implementation alternatives for your project. Contrary to popular opinion, there are always several options for implementing an application. Potential alternatives include:

|  |  |
| --- | --- |
| 1. **Do nothing**. A valid option is to remain with the status quo and not implement an application at all. Remember, you do not have to automate everything. 2. **Build it in one or more languages**. There are many implementation languages that you can choose from, including Java, C#, and COBOL. Furthermore, you can choose to work with several languages on a single project. For example, many organizations choose to use Java on client machines because of its portability and ease of distribution and COBOL for shared logic implemented on your mainframe. 3. **Build it using one of several technical architectures**. You can usually choose from several architectural strategies, including service-oriented architecture (SOA), mainframe-based with thin clients, fat-client, or application server based (e.g. J2EE). 4. **Buy a package and integrate**. Perhaps your best alternative is to choose one or more commercial-off-the-shelf (COTS) packages developed by a software company that specializes in the problem domain that you are attempting to automate. Excellent packages for human resources, manufacturing, distribution, insurance, and banking exist, to name a few.   The important thing is to identify several viable alternatives for your project so that you may assess and then compare them to select the best one for your organization. | [**[Managing Agile Projects](http://www.ambysoft.com/books/managingAgileProjects.html)**](http://www.ambysoft.com/books/managingAgileProjects.html) |

**2. Determining** **Economic Feasibility**

When assessing the economic feasibility of an implementation alternative the basic question that you are trying to answer is “does the project make financial sense?” You do this by performing a cost/benefit analysis, which as its name suggests compares the full/real costs of the application to its full/real financial benefits. The alternatives should be evaluated on the basis of their contribution to net cash flow, the amount by which the benefits exceed the costs, because the primary objective of all investments is to improve overall organizational performance.

[Table 1](http://www.ambysoft.com/essays/projectJustification.html#Table1) lists some of the potential costs and benefits that may be accrued by a software project. Although the list is not comprehensive, it does provide an indication of the range of factors that you should take into consideration when assessing the economic feasibility of an application. The table includes both qualitative factors, costs or benefits that are subjective in nature, and quantitative factors, costs or benefits for which monetary values can easily be identified. I will discuss the need to take both kinds of factors into account when performing a cost/benefit analysis.

**Table 1. Potential costs and benefits of a software project.**

|  |  |  |
| --- | --- | --- |
| **Type** | **Potential Costs** | **Potential Benefits** |
| Quantitative | * Hardware/software upgrades * Fully-burdened cost of labor (salary + benefits) * Support costs for the application * Expected operational costs * Training costs for users to learn the application * Training costs to train developers in new/updated technologies | * Reduced operating costs * Reduced personnel costs from a reduction in staff * Increased revenue from additional sales of your organizations products/services |
| Qualitative | * Increased employee dissatisfaction from fear of change * Negative public perception from layoffs as the result of automation | * Improved decisions as the result of access to accurate and timely information * Raising of existing, or introduction of a new, barrier to entry within your industry to keep competition out of your market * Positive public perception that your organization is an innovator |

**2.1 Quantative Cost/Benefit Analysis**

To perform a quantitative cost/benefit analysis you need to identify the initial monetary costs of development, the expected monetary costs of operating and supporting the application, and the expected future monetary benefits of using the application. Because these costs and benefits are accrued at different times, some immediately and some in the future, you need to convert the costs to present-day values so that you can compare them fairly.

A [present-day value](http://en.wikipedia.org/wiki/Net_present_value) is an amount of money where inflation has been taken into account to determine its value in today’s terms. For example, at an inflation rate of 5% per annum (per year) the value of $1 four years from today is worth $0.71 ($1/(1.05)4) in today’s terms. Another way to look at it, if I had $0.71 today and invested it at 5% per annum I would have $1 four years from now. By converting all figures to their present-day value you are able to fairly compare them.

It is not enough to look at the net benefit of an alternative, you must also look at its [internal rate of return (IRR)](http://en.wikipedia.org/wiki/Internal_rate_of_return). The IRR is an indication of the payback of the alternative as a percentage of its cost. Not only does a system have to pay for itself, it also has to provide a certain level of profitability to make it worthwhile for your organization. IRRs are important because your organization can often invest its money in several projects at any given time, therefore they want to choose the ones that provide the best payback for their investment (i.e., the ones with the greatest IRRs).

For example, if alternative A and B both have a net annual benefit of $50,000 in present-day terms, which is a better project? With this information they both alternatives appear equal. Knowing that I invested $1,000,000 in alternative A and $100,000 in alternative B which is the better one now? Alternative B is superior because it has an IRR of 50% whereas alternative A only has an IRR of 5%. IRR is an important economic measure because it allows you to compare investments of different scope and size, and therefore is a critical deciding factor in determining whether or not a project should be undertaken.

IRR is also important from a risk management point of view – high-risk alternatives should have a greater IRR than low-risk alternatives. For example, if alternatives C and D both have IRRs of 10%, which is a better one to undertake? You cannot tell. However, if C has a lower risk than D, then it is the better one as its risk-to-return ratio is superior. A small minority of organizations have defined desired IRRs for various levels of risk, perhaps low-risk projects must have an IRR of 20% whereas high-risk projects must have an IRR of 50%, that must be exceeded in order for a project to be considered feasible. You should look into whether or not your organization has defined expected IRRs, and if it hasn’t then you should consider suggesting that it should.

**2.2 Qualitative Cost/Benefit Analysis**

There are two lines of thought regarding qualitative cost/benefit analysis. The first strategy is to keep things simple and go with your instincts: if the project seems like a good idea, then it most likely is. The second line of though is that you can quantify the qualitative aspects of a project and thereby compare them fairly. My experience is that it is so easy to make the results come out any way that you want them too that you're effectively doing busy work to explain your "gut feel" decision. If this is the case, stop wasting everybody's time and simply admit that you're making a judgment-based decision.

So, how would you quantify qualitative factors? To do so, follow these steps:

1. Identify the qualitative factors. Brainstorming with several people is usually a good option.
2. Quantify the importance of each factor to your organization. For example, give each factor a rating of one to five, where five is the most important.
3. Numerically rate each alternative against each qualitative factor. For example, rate each alternative on a scale of zero to ten where ten is the highest possible rating.
4. Multiply the importance weighting by the rating for each alternative.
5. Calculate the overall score for each alternative by summing the individual scores.

**2.3 Best Practices**

When determining the economic feasibility of an alternative there are several important things that you want to do:

1. **Allocate costs fairly**. If your project requires hardware/software upgrades that other applications also need then you should not have to bear the full cost of the upgrade. You will need to negotiate your portion of the upgrade with senior management.
2. **Allocate benefits fairly**. Many benefits can be achieved through the improvement of business processes without the need for additional automation. Because they still could have been achieved without writing a single line of code you cannot fairly attribute them to your project. The only benefits that you can claim are those that are the direct result of automation.
3. **Work with developer salaries accurately**. There are two fundamental realities of software development that makes estimating the cost of a software project unique: 80% of the cost is developer salaries and developer salaries grow at a rate greater than inflation. The implication is that although a junior programmer earns $60,000 today the same position next year may pay $66,000, a 10% increase, even though the rate of inflation was only 2%. The implication is that the net present value (NPV) of a developer tomorrow is greater than that of one today, something that rarely happens in any other industry.
4. **Look for the benefits first**. You should first look for the benefits of a software project and then determine if you can afford the required IT spending to obtain them. It is an issue of perspective: if you know the benefits first then you are able to quickly decide whether or not the project even has a chance of making sense whereas if you start by looking at the costs you are more likely to buy into the project, therefore more motivated to falsely justify the project, as a result of the planning that you need to do to estimate those costs. The short story is that if the project offers very little in the way of benefits you can stop it right there.

**3. Determining** **Technical Feasibility**

Fundamentally, you're trying to answer the question "Can it actually be built?" To do this you must investigate the technologies to be used on the project. The problem with technology is that everything works perfectly on marketing slides, but when you get the technology in-house it is often a very different story. As a result, alternatives for each technology, if any, should be identified. Note that to do a reasonable assessment that you may need to do a mini-project and build a proof-of-concept prototype that verifies that the technologies work together . This effort may take several weeks but will pay for itself because it verifies how well your technology choices work.

For each technology alternative that you assess you should identify the advantages and disadvantages of it. Because technologies evolve quickly, if you discount a technology today you should document what needs to occur for it to be reconsidered at a later date. For example, a database may fail your technical assessment because the current version does not scale large enough for your organization, but you might indicate that if it was able to process 5,000 transactions per second (or more) then it should be looked at again.

[Table 2](http://www.ambysoft.com/essays/projectJustification.html#Table2) describes two basic categories of issues that should be addressed by a technical assessment. The first category addresses hard-core technology issues such as the scalability of the technology whereas the second category addresses market issues with the technology such as the viability of the vendor. Both categories are important to adequately assess the technical feasibility of a project.

**Table 2. Some issues to consider when determining technical feasibility.**

|  |  |
| --- | --- |
| **Technology Issues** | **Market Issues** |
| * Performance * Ease of learning * Ease of deployment * Ease of support * Operational characteristics (i.e. can it run 7 days a week, 24 hours a day?) * Interoperability with your other key technologies * Scalability | * Vendor viability (i.e. is it likely that they will be in business in two years? In five?) * Alternate sources for the technology, if any * Third-party support for related products and services * Level of support provided by the vendor * Industry mindshare of the product (i.e. is the market gravitating toward or away from this technology?) |

|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | A common mistake, one that is constantly being repeated, within the information industry is the identification of a technical solution to a non-technical problem. The lesson to be learned is that technical solutions are only appropriate when used to solve technical problems. | |

**4. Determining** **Operational Feasibility**

Not only must an application make economic and technical sense, it must also make operational sense. The basic question that you are trying to answer is “Is it possible to maintain and support this application once it is in production?” Building an application is decidedly different than operating it, therefore you need to determine whether or not you can effectively operate and support it. [Table 3](http://www.ambysoft.com/essays/projectJustification.html#Table3) summarizes critical issues which you should consider.

**Table 3. Issues to consider when determining the operational feasibility of a project.**

|  |  |
| --- | --- |
| **Operations Issues** | **Support Issues** |
| * What tools are needed to support operations? * What skills will operators need to be trained in? * What processes need to be created and/or updated? * What documentation does operations need? | * What documentation will users be given? * What training will users be given? * How will change requests be managed? |

Very often you will need to improve the existing operations, maintenance, and support infrastructure to support the operation of the new application that you intend to develop. To determine what the impact will be you will need to understand both the current operations and support infrastructure of your organization and the operations and support characteristics of your new application. If the existing operations and support infrastructure can handle your application, albeit with the appropriate training of the affected staff, then your project is operationally feasible. You can read more about operations and support at [Extending the RUP with an Operations and Support Discipline](http://www.enterpriseunifiedprocess.com/essays/operationsAndSupport.html).

**5. Determining** **Political Feasibility**

The fundamental question that you need to ask yourself is "Will you be allowed to succeed?" Your project can make total sense, yet still not be accepted by senior management for political, instead of logical, reasons. A good project manager will be cognizant of the political landscape and present their feasibility study appropriately.

Furthermore, many project teams which are trying new technologies, or following new techniques, may discover that there is significant political resistance by other people within your organization. For example, if your organization is a "J2EE shop", then your .NET project, no matter how much it makes sense, may be torpedoed by other people in your IT department who don't want .NET to be adopted. Or, perhaps your development team is following an evolutionary process such as the [Rational Unified Process (RUP)](http://www.ambysoft.com/unifiedprocess/rupIntroduction.html) or an agile one such as [Extreme Programming (XP)](http://www.agilemodeling.com/essays/agileModelingXP.htm). However, your data management group still hasn't adopted any of the [evolutionary/agile techniques](http://www.agiledata.org/essays/bestPractices.html) which would allow them to work effectively with your team. Instead, they insist on following their largely [disproved serial approaches to development](http://www.agilemodeling.com/essays/examiningBRUF.htm) all in the name of risk reduction (serial development is actually much riskier), and will fight it out with you politically instead of improving the way that they work.

|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | To paraphrase the pro-gun community in the United States:  Technology doesn't kill projects, people kill projects. | |

**6. Some Final Words of Advice**

Here is some food for thought:

1. **Actually do it**. It's important to invest the time to justify a software development project, particularly considering that the [success rate](http://www.ambysoft.com/surveys/success2007.html) [for IT projects](http://www.ambysoft.com/surveys/success2010.html) could be better.
2. **Keep it as simple as possible**. A very common mistake which organizations make is to over do this effort. You need to spend some time doing this, but remember that every dollar you spend justifying the project is one dollar less that you have to spend to [build high-quality, working software which meets the changing needs of your stakeholders](http://www.ambysoft.com/essays/agileManifesto.html). Keep your [documentation](http://www.agilemodeling.com/essays/agileDocumentation.htm), if any, as simple as possible because frankly once the effort is complete you may never look at the feasibility study again. It doesn't need to be pretty, it [just needs to be good enough](http://www.agilemodeling.com/essays/barelyGoodEnough.html). For example, don't write prose when point-form is good enough, don't spend hours making a fancy financial payback period graph when a [whiteboard sketch](http://www.agilemodeling.com/essays/whiteboardModeling.htm) will do.
3. **Document your assumptions**. If you make any assumptions during the feasibility study then consider documenting them because this is important information that senior management needs to judge the validity of your work. If an alternative hinges on the database from a given vendor to be used consistently on all database servers then they need to know this – perhaps they are considering changing vendors.

**Suggested Reading**

|  |  |
| --- | --- |
| [Disciplined Agile Delivery](http://www.ambysoft.com/books/dad.html) | This book, [Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise](http://www.ambysoft.com/books/dad.html) describes the [Disciplined Agile Delivery](http://DisciplinedAgileDelivery.com) (DAD) process decision framework. The DAD framework is a people-first, learning-oriented hybrid agile approach to IT solution delivery. It has a risk-value delivery lifecycle, is goal-driven, is enterprise aware, and provides the foundation for [scaling agile](http://DisciplinedAgileDelivery.com). This book is particularly important for anyone who wants to understand how agile works from end-to-end within an enterprise setting. Data professionals will find it interesting because it shows how agile modeling and agile database techniques fit into the overall solution delivery process. Enterprise professionals will find it interesting beause it explicitly promotes the idea that disciplined agile teams should be enterprise aware and therefore work closely with enterprise teams. Existing agile developers will find it interesting because it shows how to extend Scrum-based and Kanban-based strategies to provide a coherent, end-to-end streamlined delivery process. |

|  |  |
| --- | --- |
| **[Managing Agile Projects](http://www.ambysoft.com/books/managingAgileProjects.html)** | Are you being asked to manage a project with unclear requirements, high levels of change, and/or a team using Extreme Programming or other Agile Methods? If you are a project manager or team leader who is interested in learning the secrets of successfully controlling and delivering agile projects, then this is the book for you. From learning how agile projects are different from traditional projects, to detailed guidance on a number of agile management techniques and how to introduce them onto your own projects, we have the insider secrets from some of the industry experts – the visionaries who developed the agile methodologies in the first place. Managing Agile Projects is edited by [Kevin Aguanno](http://www.agilesecrets.com/), a noted speaker and educator on agile project management, and includes contributions from many noted figures in the agile movement. |
| **[Order now!](http://www.ambysoft.com/books/enterpriseUnifiedProcess.html)** | [The Enterprise Unified Process: Extending the Rational Unified Process](http://www.ambysoft.com/books/enterpriseUnifiedProcess.html) by [Scott W. Ambler](http://www.ambysoft.com/scottAmbler.html), John Nalbone, and [Michael Vizdos](http://www.michaelvizdos.com?eup_book), describes how to extend the [RUP](http://www.ambysoft.com/unifiedprocess/rupIntroduction.html) to address the entire information technology (IT) lifecycle. The extensions include two new phases, [Production](http://www.enterpriseunifiedprocess.com/essays/productionPhase.html) and [Retirement](http://www.enterpriseunifiedprocess.com/essays/retirementPhase.html), and several new disciplines: [Operations and Support](http://www.enterpriseunifiedprocess.com/essays/operationsAndSupport.html) and the seven enterprise disciplines ([Enterprise Business Modeling](http://www.enterpriseunifiedprocess.com/essays/enterpriseBusinessModeling.html), [Portfolio Management](http://www.enterpriseunifiedprocess.com/essays/portfolioManagement.html), [Enterprise Architecture](http://www.enterpriseunifiedprocess.com/essays/enterpriseArchitecture.html), [Strategic Reuse](http://www.enterpriseunifiedprocess.com/essays/strategicReuse.html), [People Management](http://www.enterpriseunifiedprocess.com/essays/peopleManagement.html), [Enterprise Administration](http://www.enterpriseunifiedprocess.com/essays/enterpriseAdministration.html), and [Software Process Improvement](http://www.enterpriseunifiedprocess.com/essays/softwareProcessImprovement.html)). |