ESSENSYS
SOFTWARE SOLUTIONS

A guide on

How to evaluate a software company

"It is not the answer that enlightens, but the question."

Eugen Ionescu (Eugène Ionesco), Romanian Writer

39% is the success rate of software development projects in 2012 (according to the 2013 CHAOS report published by the Standish Group). It is an alarming figure if you are involved in the software industry, and it is even more alarming if you are faced with decisions regarding software development projects. I believe this challenge that can be overcome by using the right due diligence process and thus finding the right software provider for a given job.

Software development projects fall in two main categories: in-house development and outsourced development. When outsourcing software projects, one of the first and probably most important decisions you will be faced with is deciding the software company that will build your software. This can be a daunting task if you are non technical person.

The purpose of this guide is to help you in the process of making this important decision. It is mainly addressed to non technical people involved in the decision making process. The guide is focused on evaluating outsourcing software providers.

The resources provided with this guide are:

- **An evaluation method** based on key evaluation criteria.
  This document explains the criteria used for evaluation and provides guidance on how to use this and adapt this evaluation method to your requirements. Once you read this guide you can use the evaluation tools provided to jump start the evaluation process.

- **Free tools** you can use during the evaluation process:
  - Evaluation form (Microsoft Excel file)
    You will use this form during the evaluation process to configure evaluation parameters and score the different providers.
  - Due diligence questionnaire (Microsoft Word file)
    An initial due diligence questionnaire that you can send to selected providers as a first step in the evaluation process

Download address:
www.essensys.ro/Downloads/Due-Diligence-Questionaire.doc
www.essensys.ro/Downloads/Evaluation-form.xls

In "Chapter 11 – Evaluation tools" of this guide you will find a guidance on how to use and customize these tols.

About the authors

The author of this guide is Mihai Matei, Business Development Manager at Essensys Software. Mihai has a background of almost 14 years in the software industry and he is also a Microsoft Certified Solution Developer (MCSD) and a Microsoft Certified Trainer (MCT).

Essensys is a software development company with a technical focus on Microsoft technologies, based in Bucharest – Romania. Essensys is also a Microsoft Gold Certified Partner. Among the Essensys customers are: Microsoft, Thomson Reuters, ING, Xerox, Erste Group. Additional details on the last page of this document and on the Essensys web site: www.essensys.ro.

Table of contents

Part I - Introduction

1. Realities of the software industry

"Success consists of going from failure to failure without loss of enthusiasm"

Winston Churchill

Here's where the software development is … in figures …

Year	Failed projects	Challenged projects	Successful projects
2012	18%	43%	39%
2010	21%	42%	37%
2008	24%	44%	32%
2006	19%	46%	35%
2004	15%	52%	34%
1994	31%	53%	16%

* 2004: According to the CHAOS report from Standish Group quoted in Software Magazine: http://www.softwaremag.com/L.cfm?Doc=newsletter/2004-01-15/Standish
* 2006: According to the CHAOS report from Standish Group quoted in SDTimes: http://www.sdtimes.com/content/article.aspx?ArticleID=30247
*2008-2012: According to the CHAOS report from Standish Group: http://versionone.com/4EF780A5-85FF-459B-B427-CBFADDE17692/FinalDownload/DownloadId-68F16CC2975BA9FF841FA6319C9514B2/4EF780A5-85FF-459B-B427-CBFADDE17692/assets/img/files/ChaosManifesto2013.pdf
* Standish Group Store: https://secure.standishgroup.com/reports/reports.php#reports

- **Failed projects** = projects that were stopped before completion;
- **Challenged projects** = projects that were delivered either over budget, over time or lacking functionality;
- **Successful projects** = projects delivered on time and budget.

These figures make it quite clear that the software industry has made some progress over a period of twelve years. However, we still have a long way to go. While the 1994 figures were alarming with a considerable number of failed projects, today's challenge is to decrease the number of challenged projects and increase the success rate to a level that would qualify the software world as the software industry. There is a comparison with the aviation industry that is used in many

industry speeches: would you, as software engineers, board a plane knowing that the plane has 35% chances for a perfect landing?

Success chances can be increased if proper due diligence procedures are used for selecting appropriate providers for a given project, and this is where this guide comes into play.

This paper is not intending to analyze or explain the causes of project failures, nor is it to provide guidance regarding techniques of increasing the success rate of software development projects. It is not the magical recipe for successful software projects. The intended purpose of this guide is to help you as a software customer increase your success chances by selecting an appropriate software provider. The guide is focused on evaluating custom software development providers as opposed to ISVs (independent software vendors = software companies that provide "off-the-shelf" products and/or customization services), though most of the evaluation points are also applicable to ISVs as well.

The question this guide is aiming to answer is:

"Which software company has the best chances of successfully delivering a software project, under certain given constraints?"

2. Evaluation method

Each software project has specific functional, technical or resourcing requirements. In addition each organization has its own standards and organizational requirements. So we need an evaluation method that is flexible enough to accommodate this diversity and yet simple to use as you probably do not want a huge overhead only for the provider evaluation process.

In the second part of this guide several evaluation criteria are defined and discussed. These are grouped into high level categories like: Overall Distance, Process or Quality. The importance or impact of these criteria to the success of a project is different from one project to another. For example low complexity/risk projects may not require the usage of a defined development process, while long-term complex projects will certainly need an established formal development process. In the first case any company with the right set of technical skills will probably do the job so the existence or nonexistence of a development process has no impact or limited impact on the success of your project. In the second case, however, the usage of a development process is crucial to the success of your project, as you will want a structured, transparent and controlled approach to the project. So in this case the "process impact" is high.

The suggested evaluation method assumes that at the beginning of the evaluation process, based on the project specific requirements, an impact coefficient (weight) will be selected for each evaluation criterion. Once the impact levels are decided you will grade each provider on each evaluation criteria. The overall score for a high level category (ex: Quality) is computed using a weighted average of the individual grades weighted with the impact. We are suggesting 4 impact levels: No impact (0); Low impact (1); Medium impact (2); High impact (3). For each criterion I tried to provide brief guidance on when to apply each impact level.

In the end you will have a score between 1 and 5 for each high level evaluation criteria and you can use your own method of computing the overall score for each provider (for example simple average). The high level evaluation criteria are:

- Overall Distance
- Software Development Process
- Quality
- Technology
- Company Information
- Costs

This evaluation method will work for medium to complex projects without special constraints. For projects with special constraints you should adapt the method to suit your needs. For example mission critical systems may require 7/24 technical support and on-site presence. In such cases the geographical distance may be a constraint and should be treated as mandatory/blocking.

In the third part of this document you will find details about the tools that are provided with this guide. Guidance on how to use the tools is also provided.

3. In-house vs. outsourced?

The options you have when you plan to develop or implement a software solution fall mainly in two categories: in-house development or outsourcing. There are mixtures of these approaches and the typical software development project is usually a mixture of the two approaches. This guide is focused on the latter option, of outsourcing the development work.

Outsourcing software development is an important decision, that creates multiple implications. You should carefully consider the advantages and disadvantages of outsourcing. Once you decide that you are going to outsource your software development work it is likely you will start an evaluation process which is where this guide will be helpful.

In the remaining sections we will be talking about plenty of details you should consider when outsourcing. Assuming you have decided you will develop everything in-house, here are a few thoughts:

- This is a long-term decision and you should consider all aspects and have a long term plan. It is the same as having a small software company within your organization. It will probably be a department, but in order to work you should make it work like a real software company.
- Take a look at the top ten priorities list and the additional reading section in the third part of this document. It might be useful.
- You probably already have a development team or plan to hire new people. Consider the long-term costs. You should have a constant work load to cover the costs.
- Make sure you have all the required skills not only the development skills: planning, analysis, architecture, design, testing, etc. Developing software is not only about writing code.
- Invest in tools. Development tools, third party components, source control, planning tools, design tools, code generators and a lot of other tools should be considered which will make your life easier.
- Define your development process and follow it. Constantly improve your process.
- Define your technical standards: technology, coding standards, architecture & design standards, etc.
- Invest in people and think of a long-term technical development plan. Do not assume that the technology you are using today will do the job a couple of years later.

Part II - Evaluation criteria

4. The overall distance: inshore, near shore, offshore?

"A shortcut is the longest distance between two points."

Murphy's Technology Laws

The outsourcing location of your development work is a decision that should be based on the **overall distance** between the two organizations. As in mathematics the distance can be computed based on multiple dimensions. I am suggesting the same: consider several components when you "measure" this distance and not only the obvious geographical distance. The geographical distance is not the only one that matters, and in fact these days, especially in the software industry, this is the distance that matters the least (usually). Depending on the specifics of your project some components of the overall distance may be very important while the others could be ignored.

Several components should be considered when evaluating a software provider and we have identified the following:

- Language distance
- Cultural distance
- Time distance
- Geographical distance

4.1 The language distance

In any software related project communication is important. I would say that this is one of the few parameters that always have an impact on a project, so its impact level should almost never be adjusted to zero (no impact). Depending on the complexity and clarity of the requirements adjust the impact of the language distance.

For example, short-term small projects with well defined requirements may only require that the provider speaks an international language in order to understand the requirements, so in this case you may afford, with minimal risks, to outsource your project to any provider that can understand the requirements, assuming communication will be minimal. In this case you may adjust the impact to low.

On the other hand complex projects always require extensive communication between the customer's team and the development team. Speaking the same language is critical and also speaking the same business language is important in such cases. You should ensure that your partner of choice has at least a basic understanding of the business problems you are facing and understands your language. Only natural language understanding may not be enough for complex projects.

How to evaluate the language distance?

This is probably one of the easiest tasks. Communicate and look for feedback from the other side. Try to use multiple forms of communication (email, phone conversations, direct meetings, etc.).

Software projects always involve questions and issues. The first step to success is communicating your goals. In my opinion one of the signs of good communication is when questions are asked and issues and opinions are raised. No feedback or only minimal positive feedback (the "YesYes" answer) is not a good sign of communication. You should determine if your potential partner is trying to assess the situation.

To evaluate the level of business understanding, approach your partner with some real problems, watch the reaction and evaluate the solutions they provide. Your partner should have a good understanding of the problems you are facing and should be able to offer and discuss solutions.

Impact levels	When applicable
0 – None	Never … almost never
1 – Low	Small size projects, very well defined requirements
2 – Medium	Clear scope, stable and defined requirements
3 – High	**Medium & High risk projects, unstable scope & requirements, medium to complex project**
Grades	**When applicable**
1 – Very Distant	Low level of communication.
2 – Distant	Basic communication is possible
3 – Medium distance	Good communication is possible
4 – Close	Good communication is possible and basic understanding of the business language and problems
5 – Very Close	Good communication is possible and good understanding of the business language and problems

4.2 Cultural distance

By cultural distance I mean the level to which two organizations share the same cultural and organizational values. This distance is influenced by a multitude of factors like: geographic location, organizational culture, religion, history, etc.

It is important to understand that a long cultural distance is not necessarily good or bad for a software project. It only depends on the specific requirements of the project. For example multi-cultural teams may be required for public projects that require globalization and localization; in such cases a long distance provider might be required. In other cases a provider with a similar culture will be desirable.

How to evaluate the cultural distance?

This is not straight forward and it is a subjective call. In the end you should be confident that the cultural differences between people and organizations will not create risks for your project.

Based on the specifics of your project you should decide if you need a similar or different culture provider or if this is an aspect without a real impact to your project. Rearrange the grades to fit your case.

Impact levels	When applicable
0 – None	Purely technical projects. No cultural impact.
1 – Low	Internal use projects. Very limited cultural audience.
2 – Medium	**Medium exposure projects. Controlled audience.**
3 – High	Public projects. Multiple culture exposure.
Grades	**When applicable**
1 – Very Distant/Close	Very different organizational and cultural values. Example: formal vs. informal organizations. Use 1 for projects that require cultural similarities, otherwise reverse the scores/grades
2 – Distant	Somewhat similar organizational and cultural values
3 – Medium distance	Similar organizational and cultural values.
4 – Close	Small organizational and cultural differences.
5 – Very Close/Distant	Very similar organizational and cultural values

4.3 Time distance

The time distance (time zone differences) has minimal or no impact for non mission critical systems. For mission critical systems the time distance has a high impact, in such cases you will probably require a provider that works on a similar or the same time zone with you and this constraint will narrow the geographic search.

In order to determine if your system is mission critical you can ask yourself the following question: "How much time this system can be unavailable before I start losing money?" If your answer is anything higher than 30 minutes, most probably your system is not mission-critical. Core banking systems are a good example of mission critical systems. Amazon.com web site is another example of a critical mission web site where as the Essensys.ro web site is not.

Time differences are important due to support and communication issues. In most of the cases the time difference gap can be accommodated by using appropriate communication tools (collaboration portals, forums, wikis, blogs, emails, etc.).

How to evaluate the time distance?

This is the easiest issue to overcome. Look at the time zone difference between your location and your provider location. Even if there is no or minimal impact related to time differences try to evaluate whether or not the provider has previous experiences working in similar conditions and also evaluate the communication tools they are using.

Impact levels	When applicable
0 – None	Delays of more than one day in communication are acceptable.
1 – Low	**Next business day communication is required.**
2 – Medium	Same business day communication is required
3 – High	Immediate communication is required. No alternative solution is available.
Grades	**When applicable**
1 – Very Distant	10+ hours
2 – Distant	8-10 hours, no collaboration platform in place, or 10+ hours, collaboration platform in place and previous experience in similar conditions
3 – Medium distance	6-8 hours, no collaboration platform in place, or 8-10 hours, collaboration platform in place and previous experience in similar conditions
4 - Close	4-6 hours, no collaboration platform in place, or 6-8 hours, collaboration platform in place and previous experience in similar conditions
5 – Very Close	0-4 hours

4.4 Geographical distance

The last factor that influences the overall distance is the geographical location. I would say that in most cases this is the least important factor these days as there are multiple ways to overcome the distance. The geographical distance influences the way you will communicate and has an impact on support and deployment. The communication issues can be accommodated using appropriate tools. The deployment and support issues are also controllable to some degree as physical access is rarely required. Unless your project requires on-site physical presence of the provider, the impact of the geographical distance is low or there is no impact.

The geographical distance has a high impact in projects where provider on-site presence is required. For example projects with multiple locations deployment where hardware is delivered and needs special setup. Another example would be projects that require on-site support. When evaluating the impact of the geographical distance impact keep in mind that there are plenty of technical solutions to overcome the disadvantages.

How to evaluate the geographical distance?

It is as easy as with the time distance. Choose the impact level based on the specific requirements of your project then grade the geographical distance according to the criteria below or adapt it to your needs.

Impact levels	When applicable
0 – None	Project does not need on-site presence. All work can be done remotely. Project communication can be handled remotely. Examples: hardware deployment can be outsourced to local partner or handled by local staff.
1 – Low	**On-site presence would be preferable but there are ways to overcome the distance. Examples: provider has an appropriate communication platform (online collaboration platform, conferencing solutions, etc.). Remote access is possible.**
2 – Medium	On-site presence will most probably be required. Examples: multi-site deployments
3 – High	On-site presence is mandatory Examples: mission critical systems.
Grades	**When applicable**
1 – Very Distant	Distant location (12+ hours of traveling). Provider unavailable for traveling. No technical solutions to overcome the distance.
2 – Distant	Distant location (12+ hours of traveling). Provider unavailable for traveling. Technical solutions to overcome the distance are available
3 – Medium distance	Medium distance (within 8-12 hours traveling distance). Provider available for traveling. Technical solutions available to overcome the distance
4 - Close	Small distance (within 4-8 hours traveling distance). Technical solutions available to overcome the distance
5 – Very Close	Local provider. Immediate availability.

4.5 The typical distribution of impact levels

The table below shows a typical distribution of impact levels.

Distance	Impact level	When applicable
Language distance	High (3)	In most of the projects good communication is critical. Speaking the same language is a must and basic understanding of the same business language is important.
Cultural distance	Medium (2)	Most of the projects typically require either multi-cultural teams or similar culture teams. In rare cases the cultural distance is not important.
Time distance	Low/None (1)	The time distance usually has a low impact on the success of the project, assuming that the provider has previous experiences and has a collaboration platform in place.
Geographical distance	Low/None (1)	Geographical distance also has a low impact usually as there are plenty of technical solutions available to overcome the potential disadvantages.

In the above configuration the overall distance would be composed of: 42.8% (3/7) Language, 28.5% Cultural distance (2/7), 14.2% Time distance and 14.2% Geographical distance.

5. The software development process

"I know you believe you understood what you think I said, but I am not sure you realize what you heard is not what I meant"

Roger S. Pressman, American Author

The Software Development Process refers to a structured approach of software development, as opposed to an ad-hoc approach. Other common terms for the software development process are: Software Development Life Cycle (SDLC), Software Life Cycle (SLC), or Software Process (SP).

The existence of a well defined software development process ensures that projects are handled in a structured, transparent and predictable way in order to minimize risks and handle inherent complexities related to software development. To extrapolate and use the comparison to the aviation industry used in the first part of this document, we can safely assume that you probably would not board a plane knowing that the plane was not carefully designed or thoroughly tested.

It is true that in most cases it is not your life on the line, but usually software development involves considerable amounts of money and time, and failure can inevitably have negative effects on the business.

Software development is inherently complex and requires a structured approach. A well defined process is required; however a process by itself will not save the day. There are numerous models, processes and frameworks each one with advantages and disadvantages, some of them:

- RUP (Rational Unified Process – IBM),
- CMM (Capability & Maturity Model),
- MSF (Microsoft Solutions Framework),
- XP (Extreme programming),
- FDD (Feature driven development),
- TDD (Test driven development),
- Waterfall & Spiral models,
- etc.

The detailed description of these models and processes is out of the scope of this guide. There are numerous resources on the market that deal with process theory. Instead I tried to identify the key characteristics of a process that you should try to evaluate.

Your main objectives when evaluating the software development process of your potential partner are primarily to determine if they have a well defined and implemented process and secondly to determine if that process accommodates the requirements of your project. No organization is perfect so you should also identify the process weak points in order to identify and quantify the risks for the project

Several characteristics of a well defined process:

- Clear team roles, authorities and responsibilities
- Well defined milestones with exit criteria, phases and activities
- Clear project deliverables
- Appropriate QA activities for each project deliverable (not only code)
- Traceability of deliverables

When evaluating a provider you should define an impact level for each of the process characteristics listed below. For projects that are entirely outsourced (all project related activities are outsourced) the impact level is the same for all criteria. For partially outsourced projects you should adjust each impact level and measure the provider capacity to integrate with your company's processes. For example if the testing activities will be performed in-house or outsourced to another provider, you should determine if integration between the two teams is possible and quantify the risks.

A well defined process is usually documented so the presence of process maps, process procedures, document templates, and process tools is usually a good sign. Again remember that only the presence of such documents or tools is not enough. You should confirm that the process is also actually used within the organization and embraced by the employees otherwise its definition is useless.

5.1 Team model

A process should define clear team roles, each role with its own authority and responsibilities. The provider team members should clearly understand their role within the project.

How to evaluate Team model?

Question the provider about their team model, and how teams are organized. Ask them how they work and who is doing what. You can also ask for interviews with people from the organization.

Their answer should be concise and to the point. If they have a well defined process they should know who is doing what. Remember that there is no right answer, each organization structures its team differently, so you should also evaluate if the provider's team model seems appropriate for your project. Also remember that one person can have multiple roles as long as there are no conflicts of responsibilities.

Organizations are made of people. This is often forgotten. Passionate and involved people make great companies. So just talk to the people.

Grades	When applicable
1 – Very Poor definition	No roles are defined. Provider answers are unclear.
2 – Poor definition	Fuzzy definition of roles. Ad-hoc responsibilities
3 – Medium definition	Informal definition of roles. No documentation available
4 – Good definition	Good definition of roles. Semi documented.
5 – Very good definition	Well documented roles and responsibilities.

5.2 Process deliverables

By deliverable (artifact) I understand the output of a software project. Software development involves more than code writing, so the deliverables of a software project are not only the source code and the executables.

A well defined process should clearly define its outputs: the deliverables. These artifacts are progress indicators for you as a customer, and in the end it is what you are paying for. Some examples of software project deliverables: scope document, specification document, usage scenarios, user interface prototypes, deployment procedures, test plans and many others.

How to evaluate the process deliverables?

Ask the provider how they work and what the deliverables of each phase of the process are. Enquire as to what the purpose and content of each deliverable are. Ask for document templates and moreover ask for previous project deliverables and samples. The existence of previous deliverables is a clear sign that the process is well implemented. Remember that only the process definition will not help at all.

The provider will probably be reticent in disclosing previous project artifacts due to existing non-disclosure agreements and this is a good sign. In such cases you should at least try to obtain the structure (table of contents) of some deliverables without the actual content. The provider may also be reticent in delivering templates or samples in electronic format, which is also normal. These artifacts are part of their know-how.

Grades	When applicable
1 – Very Poor definition	Provider proposes none or not enough deliverables
2 – Poor definition	Only major deliverables exist (specification, code, binaries)
3 – Medium definition	Most of the deliverables are defined but no template or guidance is in place. Ad-hoc definition process.
4 – Good definition	Most of the deliverables are defined based on templates.
5 – Very good definition	Clear definition of deliverables. Existing templates. Existing well documented samples.

5.3 Scope analysis

Scope analysis refers to the process of determining the scope of a system in terms of functional and non-functional requirements. Scoping involves determining what will be included in the system (in-scope) and what will not be included or will be included in a later version (out-of-scope). Functional requirements determine what the system will actually do or enable the user to accomplish (examples: “the system will allow users to manage customers and technical support incidents”). Non-functional requirements determine other not so obvious characteristics of a software system such as: performance, availability, scalability, localization, globalization, security, etc. A good definition of both functional and non-functional requirements is equally important for the success of a software project.

Scope analysis is one of the first activities in the project life cycle and its primary target is to determine the high-level objectives and business requirements of the project. The output of this activity is usually a document that describes the main functional and non-functional requirements of the system. This document can have many names and several examples are: Scope document, Vision & Scope document, Offer.

One of the first things a software provider should do is to try and understand your business problem and requirements without going into very much detail. Business people tend to focus mainly on functional requirements; this is why you should pay attention if your provider also tries to determine non-functional requirements which are equally important, by asking questions such as: How many users do you have? How many users are you expecting in the future? Do you need a multi language interface? How many customers do you have? Etc.

How to evaluate the scope analysis activity?

Ask your provider for the following information:

- What is the first step in a project
- Scope document or offer template
- A previous project scope document or offer
- Ask about what aspects are discussed during scoping and how discussions are conducted
- Is the provider trying to determine non-functional requirement through direct or indirect questions

Grades	When applicable
1 – Very Poor definition	No scope analysis in place. No activity defined. No templates available.
2 – Poor definition	Ad-hoc scope analysis. No templates available.
3 – Medium definition	Scope analysis in place. Accent mainly on functional requirements. No well defined templates.
4 – Good definition	Scope analysis. Existing templates. Structured samples.
5 – Very good definition	Well defined activity or phase. Existing templates and well documented samples. Accent on both functional and non-functional requirements.

5.4 Specification & planning

With a clear and agreed scope the provider should uncover all the details of the project. Usually this activity involves discussions between the customer team and the provider team that are meant to clarify every aspect of the future system. The main output of this activity is the functional specification of the software system that clarifies and documents the requirements and the functions of the system. Depending on the process your provider is using, this activity may occur as a distinct phase of the software development process or it could be an iterative activity that is repeated constantly during the project's lifecycle (Agile processes). In either case the output should be well documented and planned.

Several common names for the functional specification are: Software Requirements Specification (SRS), Functional Specification Document (FSD), Program Specification (PS), etc. In most of the cases the Functional Specification is a package of documents such as Use Cases, Usage Scenarios, Business Object Models, Data Models, etc.

Once requirements are gathered and analyzed the provider should plan development activities. The output of this activity is a Plan document, also known as: Master Plan Document, Staged Delivery Plan, Planning Document, etc. This should document at least the phases, milestones and deadlines of the project and could also detail all the project related activities.

The functional specification document formalizes what you will receive, and the plan document formalizes when you will receive. These documents act as the agreement on the deliverables between you and your provider, and are of critical importance to the success of the project. Errors in these documents usually have huge impact on the project. It is a good practice to make these documents part of your legal contracts, to ensure that you are also legally protected. During evaluation you should ensure that the provider understands the importance of these documents.

How to evaluate the specification and planning activities?

Ask your provider for the following information:

- How are they gathering and documenting requirements
- Functional Specification templates
- Specification output samples from previous projects
- Typical topics that will be discussed
- Plan templates and previous sample
- Estimation procedures
- Activities included in planning documents

In the end you should evaluate the provider's capacity of understanding and clarifying your requirements and correctly estimating the level of effort your project requires.

Grades	When applicable
1 – Very Poor definition	No specific activities. No templates available. Ad-hoc process
2 – Poor definition	Some activities are defined. Some existing samples.
3 – Medium definition	Minimum activities and templates. Existing samples. Samples not very similar. Author dependent.
4 – Good definition	Well defined activities. Well documented templates and samples. No tailoring is made.
5 – Very good definition	Well defined activities. Existing templates and previous samples. All typical deliverables exist. Tailoring according to project complexity.

5.5 Architecture & design

You can think of architecture and design as the blueprints of a plane or a building. You would not start building a plane or a building without having a well thought technical plan. It is the same with the software development discipline; it is an engineering discipline and requires proper planning and design. Although at the first sight this activity does not deliver anything useful it is of great importance to the success of a software project. You should ensure that the provider will first dedicate some time considering what it has to do instead of rushing to building it. Typical outputs of this activity are: software architecture document, database diagrams, class diagrams, user interface prototypes, proof of concept solutions, etc.

How to evaluate the architecture and design activities?

Ask your provider for the following information:

- Architecture & design activities
- Design tools they are using
- Architectural validation
- Architectural & design patterns they are using
- Templates & samples from previous projects
- How are these items used during development

These activities usually involve the usage of specialized tools (software tools), it is not enough that the provider spends some time with architecture & design activities, the output of these activities should be well documented. For example a database diagram could be done with a word processor (such as Microsoft Word) but it would not be the happiest choice of tools. For this case a specialized database designer tool would be indicated (for example: Microsoft Visio or any other tool available on the market).

Grades	When applicable
1 – Very Poor definition	No distinct architecture & design activities. Ad-hoc process. No tools are used.
2 – Poor definition	Minimum documentation & tools.
3 – Medium definition	Design tools are used. Existing samples. No architectural framework, architecture or code reviews.
4 – Good definition	Distinct activities. Well documented structures and samples. Design tools. No architectural framework. Architecture or code reviews are conducted.
5 – Very good definition	Distinct activities. Existing templates and previous samples. Existing design tools. Existing architectural framework. Architectural & design reviews.

5.6 Development

The system is built through development. There is a common understanding that the development activity should be iterative instead of one big development phase. Iterating through development ensures the existence of several development milestones so progress can be tracked. This way delays and roadblocks will be immediately identified and intervention will be possible. Another very important activity during development is testing. You should ensure that your provider will test the system and its components during development. This helps to identify and fix problems during development rather than at the end of the project. The existence of coding standards or rules for writing code is also important.

Personal opinion: It would seem natural that during development the customer has no active role. While this is possible it is not recommended. It is preferable to have an iterative development process and involve the customer in the process. One successful practice is for the development team to release public intermediary builds of the system at regular intervals (2-6 weeks). Customer involvement does not mean that the customer team will actually write the code, it means it will have access to each intermediary build which will be reviewed and feedback will be provided. This practice has several advantages: it helps detecting problems early in the process; involves the customer and helps managing expectations; the development process is transparent for the customer who can easily track progress.

How to evaluate the development activity?

Ask your provider for the following information:

- What are the activities of the development phase(s). Testing should definitely be one.
- Ask how are tasks organized and how is the team interacting during development. Every member of the team should clearly know its role.
- Ask about the common problems they are encountering during development and how are these solved.
- How is progress assessed (status meetings, time sheets, time tracking tools)
- Are they conducting code reviews during development
- Source control tools (Source code should be controlled: TFS, VSS, CVS)
- Development tools (code generators, refactoring tools, standard development tools)
- Testing environment (ideally separated from the development environment)
- Are there any coding standard that every team uses

Minimum requirements: testing during development, source control and standard development tools.

Grades	When applicable
1 – Very Poor definition	No intermediary builds. No customer interaction during development. No testing during development. No source control.
2 – Poor definition	No intermediary builds. Testing during development. Source control. No code reviews or time tracking.
3 – Medium definition	Iterative development process. Testing during development. Source control. No code reviews. No time tracking tools.
4 – Good definition	Iterative. Testing during development. Source control. Code reviews. Coding standards.
5 – Very good definition	Iterative development process. Testing. Source control. Code reviews before each milestone. Time tracking tools. Coding standards.

5.7 Testing

In order to fix errors, the errors should be known. Testing is the only way of knowing what the errors are. The process your provider uses should ensure that faults are discovered as early as possible. This is the first priority of the testing process. The reason for this is simple: errors are cheaper to fix at the beginning of a project as opposed to at any other stage of the process. In short, errors in the early phases of a project have a huge impact on the success of the project, while errors at the end have a smaller impact. Errors in the scope definition or requirements specification have a great impact: the provider might risk developing something different to the customer's requirements. Deployment errors on the other hand, in most case, have a smaller impact.

In well defined processes all project deliverables are tested (not only the code), including functional specification, scope document, plan document, etc. Typical testing procedures for these artifacts (documents) are the peer review and the team reviews.

For the actual system testing there is a considerable set of tests that can be performed, including: unit testing, functional testing, integration testing, regression testing, guerilla testing, performance testing, alpha testing, beta testing, code reviews, user acceptance tests (UAT), etc.

Testing procedures are usually the provider's responsibility, unless otherwise agreed, so you should ensure that the providers completely understands this and does not rely on your team or users for testing.

All the testing procedures should occur during the project lifecycle rather than at the end of the project (functional testing for example could occur after each intermediary public build). It is also very important that the testing results are recorded (electronically) and there is a standard procedure for testing and fixing defects. The provider should also have a test plan that describes the testing activities.

Personal opinion: there is never too much testing.

How to evaluate the testing activity?

Ask your provider for the following information:

- How do they ensure that requirements and scope are clear (team reviews, customer reviews, formal agreement on the specs)
- How do they ensure that the technical architecture and technology choices are correct (reviews, proof of concept solutions, prototypes)
- What is their testing procedure
- What tools are they using to record testing results
- What kind of reviews are they conducting
- What tools are they using for testing
- Templates and samples from previous projects (bugs, issues, test plans)

Grades	When applicable
1 – Very Poor definition	Testing at the end. No reviews. No tools for recording results or testing.
2 – Poor definition	Ad-hoc testing during development.
3 – Medium definition	Testing during the project lifecycle. Recording tools. Only system testing. No reviews.
4 – Good definition	Iterative testing. Peer reviews. Occasional automated testing.
5 – Very good definition	Iterative testing phase. All deliverable are reviewed or tested. Testing and recording tools. Automated testing.

5.8 Deployment

Deployment is basically the activity that delivers the final results of a project to the customer. It usually involves deploying or installing the system into a production or staging environment. Deployment should be also performed in a controlled manner instead of using ad-hoc procedures. Deployment results should also be tested before the final delivery. The impact of the deployment procedures increases with the complexity of the deployment. Projects with multiple sites deployments, public projects, projects that require data migration, or core business projects require carefully planned deployments.

Deployment tools and procedures should be developed as early as possible during development, exercised and tested with each public build. Deployment procedures should be documented. Deployment tools and procedures should be tested in environments as similar as possible to the production environment.

How to evaluate the deployment activity?

Ask your provider for the following information:

- What will be the deployment procedure
- When are the deployment procedures and executables developed
- What tools are used for deployment
- How are deployment artifacts tested

Grades	When applicable
1 – Very Poor definition	Ad-hoc deployments (“copy-paste” deployments)
2 – Poor definition	Documented deployment procedures. No automated deployments.
3 – Medium definition	Documented deployment procedures. Automated deployments only at the end of a project.
4 – Good definition	Documented & automated. Deployment testing only at the end of the project.
5 – Very good definition	Well planned deployments. Deployment procedures. Automated deployments. Testing.

5.9 Support & maintenance

If you plan to also outsource maintenance & support activities you should ensure that your provider has the capacity of delivering these services. These services should adhere to several constraints that are typically formally agreed in Service Level Agreements (SLAs). Maintenance and support services require dedicated resources and tools.

How to evaluate the support & maintenance processes?

Ask your provider for the following information:

- Tools used for recording support incidents
- Maintenance procedures
- Typical SLA
- Previous experiences

Grades	When applicable
1 – Very Poor definition	No previous experience. No existing tools. Ad-hoc process. Unacceptable SLA parameters.
2 – Poor definition	Ad-hoc procedure. Some previous experience. Basic tools only.
3 – Medium definition	Ad-hoc procedure. Some previous experience. Available tools.
4 – Good definition	Defined procedure. Available tools. Acceptable SLA. No dedicated personnel.
5 – Very good definition	Well defined procedures. Good previous experience. Tools. Acceptable SLA parameters (guaranteed response times, guaranteed availability, etc.). Dedicated personnel.

5.10 Change management

Change is natural and should be expected. It took some time and a considerable number of failures until the software industry understood this. This was the time when the classic waterfall model was declared officially dead.

Usually at the beginning of the project requirements are gathered, analyzed, documented and formally agreed. This acts as the agreement between you and the provider on what, when and how things will be done. It would be great if requirements would stay stable during the project lifecycle, however, this almost never happens. Businesses are continuously changing and today's requirements are not tomorrow's requirements. In addition businesses gain a better understanding of the value software systems can provide after several development iterations. Most often changes are referring to changes of the initially agreed requirements. Typical examples are adding new features or modifying or eliminating existing features.

You should ensure that your provider fully understands this reality and is capable of adapting. You should also understand that change always has an impact. The provider should be able to quantify the impact of each change in terms of costs, duration, technical impact and/or resourcing, so you can decide whether the change should be implemented or not.

The change management process ensures that changes are possible and controllable. While you might be happy in the first phase with a provider that silently accepts any change you request, you should fully understand that this attitude creates considerable risks for the project's success. A well defined and implemented change management process protects you and the provider from the so called "spaghetti" software solutions.

It is very important to understand that the existence of a change management process does not mean that requirements should not be clear and well documented. This process only ensures that changes (in requirements for example) are controlled and not applied using some chaotic informal procedure.

How to evaluate the change management process?

Ask your provider for the following information:

- How are they accommodating changes during the project lifecycle
- What is the procedure for discussing changes
- What tools are they using to record change requests
- Samples of change requests from previous projects

Grades	When applicable
1 – Very Poor definition	Changes are not possible during the project lifecycle.
2 – Poor definition	Ad-hoc change management process.
3 – Medium definition	Informal change management process. Informal agreement.
4 – Good definition	Well defined process. Formal agreement of changes.
5 – Very good definition	Well defined change management process. Change request are recorded, estimated and integrated. Project deliverables are synchronized once a change is accepted (specs, plans, etc.).

5.11 Project management

Project Management is the discipline of planning, organizing, and managing resources to bring about the successful completion of specific project goals and objectives (Wikipedia: Project Management). Project Management is the application of knowledge, skills, tools and techniques to a broad range of activities in order to meet the requirements of a particular project. (PMI definition)

The Project Management discipline has considerable impact on software projects. Good planning is of critical importance to the success of a software project. PM activities ensure that all project related tasks are planned, organized and tracked. PM also ensures that everyone in the team knows what it has to do and when it should be delivered (deadlines).

How to evaluate the project management discipline?

Your objective in the evaluation process is to ensure that your provider has the knowledge to plan, organize and track all project related activities. You should also ensure that progress can be determined and communicated at any time.

Ask your provider for the following information:

- What are the project management activities
- How is progress reported
- How are issues reported
- Status communication
- Plan documents samples and templates
- Tools used for project management
- How is work time tracked

Grades	When applicable
1 – Very Poor definition	Informal project management. No documented plans. No tools.
2 – Poor definition	Informal. Minimum documentation. No standard tools.
3 – Medium definition	Formal project management. Documented plans. No standard tools. Ad-hoc status meetings.
4 – Good definition	Formal project management. Basic tools. Status meeting and reports.
5 – Very good definition	Formal project management. Extensive documentation. Documented plans. Frequent status meetings and reports. PM standard tools (ex: MS Project, MS Project Server, other pm tools). Dedicated role.

5.12 Risk management

Risk management is the discipline that ensures that project related risks are identified, assessed, managed and mitigated. This discipline objective is to reduce project related risks to acceptable levels and provide a strategy to handle potential problems. Another reality is that risks always exist. To handle this reality, communication is of critical importance. You should ensure that your provider perfectly understands project risks and will communicate instead of hiding problems under the carpet.
Risk management is particularly important for long-tem complex project that are exposed to a considerable number of risks.

How to evaluate risk management activities?

Ask your provider for the following information:

- Ask if there is a standard risk management approach
- Ask about common software project risks (ex: scope creep, requirements risks, deployment risks, user risks, technology risks, etc.)
- How are risks recorded and communicated (ex: accessible risks lists, status meetings, etc.)
- What is the frequency of risk management activities.
- Samples of risks from previous projects
- How are risks identified and quantified

Grades	When applicable
1 – Very Poor definition	No risk management activities.
2 – Poor definition	Ad-hoc risk management.
3 – Medium definition	Top 10 risks list. Frequent updates.
4 – Good definition	Top 10+ risks list. Periodic risk assessment. Informal communication.
5 – Very good definition	Formal & detailed risk management process. Frequent risk assessment. Formal communication and escalation.

6. Quality

"Quality is never an accident; it is always the result of intelligent effort."

John Ruskin, English writer

Quality is not negotiable. In very rare cases quality is not important and you will never sacrifice it. I have insisted in the previous part on the fact that having a process in place is not a guarantee for the success of a project. However a well defined process is a prerequisite and a major influencer for software quality. Quality also depends on company culture and values, employees' motivation, work habits and a lot of other factors.

Quality is also influenced by the customer. Customer involvement in a software project is crucial to the success of the project. Projects in which the objectives or the requirements are unstable have considerable chances of failure. This is where a well defined process comes into play. If the process is well implemented the provider should have the ability to detect project risks early in the project lifecycle.

So the question is how can you know which provider will certainly deliver quality? The short answer is you cannot know for certain. The long answer is that there are some indicators you can evaluate in order to increase your chances. Keep in mind that most software providers are marketing quality, though not everyone is delivering quality. To avoid risks you should go for the details that could confirm a quality mindset.

For all the criteria defined in this chapter you should adjust the impact levels according to the complexity and duration of your project and the particulars of your project. As a rule of thumb the same impact level for all evaluation points will probably work for most of the projects.

Impact level	When applicable
- None	Almost never…
- Low	Simple projects. Short-term. Clear scope & requirements.
- Medium	Medium projects
- High	Highly complex projects.

6.1 People

Quality is about people. In an environment that encourages quality, people are delivering it. Try to talk to key people in the provider's organization and evaluate their attitude towards quality.

How to evaluate people?

This is a subjective evaluation. Ask your provider for the following information:

- Key people CV's
- Interviews with key people
- Look for people certifications, courses taken, experience
- Company human resources policies (trainings, coaching)

Grades	When applicable
1 – Very poor	Junior people. Inexperienced. No technical authorities.
2 – Poor	Mostly junior people.
3 – Medium	Mixed team. Existing technical authorities.
4 – Good	Mixed team with enough senior people. Existing technical authorities & leaders.
5 – Very good	Experienced people. Balanced team. Enthusiastic people. Quality mindset.

6.2 Customer references

What previous customers have to say about previous project says a lot about a company. Make sure you check the customer references you are given.

How to evaluate customer references

Ask your provider for the following information:

- Ask for customer surveys
- Look for customer testimonials
- Previous customer contacts that can confirm the references
- Check the references and ask about the overall opinion
- Ask previous customers about the company weak points
- Ask previous customers about what went wrong in their projects and what can be improved

Grades	When applicable
1 – Very poor	References are not confirmed or negative feedback
2 – Poor	Confirmed references. Mixed feedback. No testimonials or periodic surveys.
3 – Medium	Confirmed references. Mixed feedback. Existing testimonials. No periodic surveys.
4 – Good	Confirmed references. Mostly positive feedback. No periodic surveys.
5 – Very good	Confirmed references. Positive feedback. Customer surveys. Existing testimonials.

6.3 Responsiveness

The reaction speed of the provider will influence the progress of your project. Ensure that the provider is responsive and provides timely answers to your requests. Make sure you find a provider that has a real interest in doing business with you.

How to evaluate customer references

Measure the time of responses (emails, faxes, letters, etc.). It is possible that the provider's team is not in a position to give a complete answer immediately (the answer might require additional effort). At the very least they should ask clarifying questions or provide a time estimate when a complete answer will be given. Responsiveness shows professionalism, politeness and most important a clear interest in doing business.

Grades	When applicable
1 – Very poor	Constantly delayed answers.
2 – Poor	Somewhat delayed answers. Not always on topic.
3 – Medium	Reasonably timed answers. Small delays. On topic.
4 – Good	Timely answers. Only occasional delays. On topic.
5 – Very good	Timely answers. On topic answers.

6.4 Architectural & design quality

Clean architecture and design have a considerable impact on the overall quality of a system. Stay out of unnecessary complicated architectures; look for the right balance between complexity and simplicity. Your provider should clearly understand its role: to solve business problems, not create technical problems. A clean, simple, orthogonal architecture that solves business problems will make development, maintenance and additional extensions easier. Over complicating things is not what you want. Stick to the KIS principle: **Keep It Simple**.

How to evaluate architecture & design quality

To evaluate this you will need a senior technical person.
Ask your provider for the following information:

- Architectural & design patterns used within the company
- Common architectures within the company (frameworks)
- Sample architecture documents from previous projects
- Look for "spaghetti" architectures and stay out
- Database diagrams

Grades	When applicable
1 – Very poor	Spaghetti architecture & design. No company standards. No code or components reuse.
2 – Poor	Reasonable architectures.
3 – Medium	Well defined architectures. No existing patterns or framework. No code or components reuse.
4 – Good	Well defined architectures. Code or components reuse.
5 – Very good	Orthogonal, simple, clean architectures & designs. Existing architectural patterns and framework. Reuse mindset.

6.5 Code quality

Code can be well written or poorly written. The way code is written has an indirect influence on overall system quality. In order to evaluate code quality you will also need a senior technical person.

How to evaluate code quality?

One very efficient way of doing this is by conducting code reviews. Ask your provider for code samples and review those.
Also ask the provider for the following information:

- How and when are code reviews conducted during development
- Are there any written coding standards used
- Are code reviews formal or informal
- Are there any source code check-in policies
- Are they using any automated tools for software measuring code indicators (cyclomatic complexity, comments per lines of code, cohesion, coupling, lines of code, etc.)
- Ask for sample code review results samples from previous projects.

Grades	When applicable
1 – Very poor	No code reviews. No coding standards.
2 – Poor	Occasional code reviews. No coding standards.
3 – Medium	Informal code reviews. Ad-hoc measuring.
4 – Good	Formal & frequent code reviews. Basic coding standards. No follow up reviews.
5 – Very good	Formal & frequent code reviews. Coding standards. Well written code. Follow up reviews.

6.6 Work habits

Another factor that influences the quality of a software project is the way people are working. In order to achieve quality, companies should have a quality mindset by encouraging a quality attitude within the company. Otherwise quality is just a random effect.

Personal opinion 1: I personally believe that for long-term effects people should keep a balance between professional life and personal life. In the short-term it is very probable that workaholics will probably produce better results; however, in the long-term encouraging and enforcing extra hours is not the right strategy in my opinion.

Personal opinion 2: software engineers are creative people. Creativity does not work well with constraints and strict rules. So in my opinion software organizations should be informal. If you are looking at the giant success software companies (Microsoft, Google, Apple, Adobe and many others) and how they are working you will see they are keeping or kept rules to a minimum, they are informal and they are fostering creativity. Asking a software engineer to wear a suit, for example, is not quite motivating.

How to evaluate work habits?

Ask the provider for the following information:

- Are people usually working extra hours? Are these billed :)?
- How are they encouraging quality
- Retention rate
- How many people left the company in the last 6 months and why? In the last year?
- What rules are applicable within the company
- On what grounds are people evaluated?

Grades	When applicable
1 – Very poor	No quality mindset. Poor retention rate. Tight rules. Regular extra hours. Bad environment.
2 – Poor	Reasonable retention rate. Occasional extra hours. Reasonable environment.
3 – Medium	Medium retention rate. Rare cases of extra hours. Good environment.
4 – Good	Good retention rate. Good environment. Quality and communication is encouraged. Regular working schedule.
5 – Very good	Informal organizations. Pleasant environment. Quality is encouraged. Communication is encouraged. Good retention rates.

7. Technology

"One day soon the Gillette Company will announce the development of a razor that, thanks to a computer microchip, can actually travel ahead in time and shave beard hairs that don't even exist yet"

Dave Beery, American writer & humorist

This scenario is science fiction today. Tomorrow who knows what will happen. It is hilarious but it is just a glimpse of the changing speed of software technology. Speaking of technology, in today's world you are either always on the wave or you are left behind. It is a very thin line between "state-of-the-art" technology companies and "legacy" companies.

Technology has two roles in software development projects: development and support. The development technology is the one that gets the job done. Support technology is the one that gets the job done better and faster. Both of them are important.

Examples of development technologies: programming languages, frameworks, third-party components, etc. Examples of support technologies: integrated development environments (IDE), modeling tools, code generators, reengineering tools, communication platforms, stress testing tools, bug tracking tools, database tools, planning tools, source control tools, etc.

When evaluating your provider you should consider both aspects of technology. It is important that your provider masters the development technologies and those technologies are suitable for your requirements. It is of equal importance that support tools are used within the provider's company.

Adjust the impact levels according to your project requirements.

7.1 Standard development technologies

As mentioned earlier, software engineers are creative people. There is a downside to that, we are sometimes too creative. You should ensure that technology is under control. The safest path is to go for previously tested, industry standard technologies. In some cases, considering advantages of brand new technologies, you will accept some risks and accept using some brand new technologies.

One of the things you should evaluate is the technological fit between the two companies. For example if your company favors Java as the development technology you should make sure that the provider masters that technology. Most software development companies have a technology orientation, or they have dedicated technical divisions (.Net, Java, etc.). No technical focus is not what you are looking for.

How to evaluate development technologies?

Ask the provider for the following information:

- Technical competencies. This should be a well defined list of technologies
- Make sure that you are technically compatible with your provider
- Ask about exotic technologies on the list and applicability
- Ask about future technical development plans

Grades	When applicable
1 – Very low	No technical focus. Technical competencies dependent only on people competencies. No development plans.
2 – Low	Very limited choice of technologies. Development based only on short term needs.
3 – Medium	Reasonable choice of technologies. Short term development plans.
4 – High	Good technical focus. Reasonable choice of technologies. Medium term development plans.
5 – Very high	Clear technical focus. Wide choice of technologies. Existing technology strategy. Long term technical development plans.

7.2 Support technologies

Support technologies are the other technologies and tools the provider is using to simplify or automate work. The list can be long. At the very least the list should include an integrated development environment, a bug tracking tool, planning tools, source control, customer communication tools.

How to evaluate support technologies?

Ask the provider for the following information:

- Tools used within the company.
- Infrastructure technologies (email, ftp, web, virtual environments, etc.)

Grades	When applicable
1 – Very low	Poor support technologies. Minimum tools.
2 – Low	Basic tools only.
3 – Medium	Existing support tools for the critical development related activities (code, testing). No integration between tools.
4 – High	Most software development related tasks have tools support.
5 – Very high	Comprehensive list of tools. Most software development related tasks have tools support. Integrated tools.

7.3 People

People are creating and empowering technology. People's technological knowledge is critical to the success of a project. The composition of the technical teams is also important. Senior teams or senior lead teams are desirable. Provider should adapt and suggest team composition according to the project's complexity. If you consider your project is highly complex you should strive for senior people only and be prepared to support increased costs.

How to evaluate people?

Use the same evaluation process as for Quality, but this time with a focus on technology. Look for experience on previous similar projects, technical certifications and ask technical questions if possible.

Grades	When applicable
1 – Very low	Junior people. No experience.
2 – Low	Mixed teams. Mostly junior.
3 – Medium	Balanced teams. Individual training plans. Training is encouraged.
4 – High	Mixed team with enough senior people. Existing training plans.
5 – Very high	Experienced and knowledgeable people. Existing training plans.

7.4 Certifications

Company technical certifications say something about technical competencies. Examples of company certifications or partner programs: Microsoft Certified Partner, Sun Partner Advantage, IBM Partner, ISO standards, SEI CMMI, etc. Each of these certification or partner programs has different certification requirements (confirmed experience, people requirements, technical requirements, etc). It is a good idea to compile your initial providers list based on certified competencies as most of the certification programs provide online catalogues that you can use.

For example, Essensys Software is Microsoft Certified Gold Partner with recognized competencies in Custom Development Solutions and Data Management Solutions. Some other competencies that are certified in the Microsoft Certified Partner program are: Advanced Infrastructure Solutions, Business Process and Integration, Information Worker Solutions, etc.

How to evaluate certifications?

Ask the provider for the following information:

- Company certifications & partnerships
- Certified competences

Grades	When applicable
1 – Very low	No applicable company certifications.
2 – Low	Minimal level company certifications.
3 – Medium	Medium level company certifications.
4 – High	High level company certification.
5 – Very high	Multiple high or medium level company certifications or partnerships.

7.5 Projects portfolio

Previous projects will say a lot about the provider's experience and technical capabilities. Ideally you should identify in the provider's portfolio projects similar with yours to ensure a possible match.

How to evaluate projects portfolio?

Ask the provider for the following information:

- Brief descriptions of previous projects
- Technologies used in previous projects
- Ask for demos if possible

Grades	When applicable
1 – Very low	No previous projects.
2 – Low	At least 3 previous projects – startups
3 – Medium	Existing similar previous projects (at least one)
4 – High	At least two similar previous projects
5 – Very high	Existing very similar previous projects (at least one)

8. Company information

"Why join the navy if you can be a pirate?"

Steve Jobs

Besides the technical evaluation of potential providers you should also evaluate company general information like history, customers' portfolio, financial information, size, products & services offering and as many information you can get. These details will give you information about the match between your company and the partner company. Your first objective is to find partners that are interested in doing business with you.

8.1 Customers

The configuration of the customers' portfolio can hide some clues regarding the company's business history. The portfolio is also a health indicator. The easiest things to look at are: number, customers' profile, recurrence. How you analyze the portfolio depends a lot on the size of the company. Overall I would try to evaluate in context the business history of the company, stability and risk exposure.

A high number of one-time customers might not be a good sign, but it depends a lot on the services offering. For example web development agencies typically have a considerable number of customers and in their case this is a good sign. A small amount of customers on the other hand may indicate high dependency on customer business. So you should look for balanced portfolios, and what balanced means depends on the specifics of the provider's business and it is for you to determine judging the context.

Usually recurring/returning customers indicate a good business history. High profile customers are also a good sign.

How to evaluate the portfolio?

Ask the provider for the following information:

- Customer portfolio
- Customers profile
- Long-term partnerships/customers
- Project that did not work optimally, ask what went wrong
- The average number of projects / customer
- Customer testimonials
- Biggest customers (turnover), look for balance

Grades	When applicable
1 – Very low	Unbalanced portfolio. Customer dependencies. No long-term partnerships.
2 – Low	Somewhat unbalanced portfolio. Existing dependencies.
3 – Medium	Balanced portfolio. Medium customer dependencies/risks. Existing returning customers.
4 – High	Balanced portfolio. Medium customer dependencies/risks. Returning customers. Long term partnerships.
5 – Very high	Well balanced portfolio. High profile customers. Returning satisfied customers. Long-term partnerships.

8.2 Company size

Personal opinion: There is a common misconception that going for the big providers is the way. My opinion is that you should find the right size fit between companies, and big is not always the right answer. For example if you are a startup and you decide to outsource your first software projects to some major player you will end up spending lots of money, you will probably receive good quality but most probably you will receive more than you need at the time and you will pay a lot for it. It is like going on family shopping with a big truck to carry the goods. On the other hand if you are one of the big players, outsourcing your work to a startup software company is probably not the right answer, as you will probably do not want to outsource the work to an unstable company without a confirmed history.

In my opinion the key is to find the right size fit. Let us split companies into five categories: startups (1-10 people), small businesses (11-50 people), medium businesses (51-250 people), big businesses (251-1000 people), and large enterprises (1001+). As a general rule I would say that you will find the size fit at your level, one level upper or one level lower. Adjust the five categories according to your requirements, for example use the turnover as the criteria. Also adjust the grades according to your needs, for example if there are corporate rules you must adhere to you will probably need to change the scoring method.

How to evaluate the company size?

Ask your provider for the following information:

- Number of people
- Number of locations
- Turnover (last year, current year estimated)
- Business plans

Grades	When applicable
1 – Very low	Three or more levels difference
2 – Low	Two levels difference
3 – Medium	One level upper
4 – High	One level lower
5 – Very high	Same level

8.3 Certifications & partnerships

Company certifications, partnerships and participation in partner programs are also important. These certifications are supporting the company image, but are also important as a confirmation of company skills. Usually certification processes have strict rules that must be followed by the certified organizations.

For example the Microsoft Certified Partner has strict rules regarding technical people certifications, projects history and technical skills.

How to evaluate certifications & partnerships?

Look for company certifications. Score according to your project requirements. For example if you are looking for a partner with Oracle technical skills, the Microsoft Certified Partner status might not be important.

Grades	When applicable
1 – Very low	No company certifications
2 – Low	No applicable certifications
3 – Medium	Entry level certifications
4 – High	Medium level applicable certifications
5 – Very high	High level applicable & useful certification

8.4 Financial information & history

From the business perspective this is very relevant. Turnover could also be used to indicate size. Most probably you will be looking for stable companies, as you will probably want your provider to live your project. Analyze this information in the context of the company. An interesting indicator you can easily compute is the productivity: divide the turnover with the number of technical employees and you will get the productivity.

How to evaluate certifications & partnerships?

Ask the provider for the following information:

- Turnover history
- Development & business plans
- Debts or legal litigations

Grades	When applicable
1 – Very low	No history. Poor results. No plans.
2 – Low	Medium results. Short term plans.
3 – Medium	Reasonable results. Medium-term plans.
4 – High	Good results. Medium term business plans.
5 – Very high	Constant growth. Good history. No legal issues. Long-term business plans.

8.5 Services & products

The provider's offering should be also evaluated. Structured diversity could imply stability. You might also find other interesting opportunities.

How to evaluate provider's offering?

Ask the provider for the following information:

- Services or products offered
- Development plans

Grades	When applicable
1 – Very low	Limited low value offer
2 – Low	Limited offer
3 – Medium	Medium offer
4 – High	Extensive offer
5 – Very high	Extensive offer. Interesting future opportunities.

8.6 Typical distribution of impact levels

Below is a suggested list of impact levels that will probably cover a good range of projects.

Distance	Low complexity	Medium complexity	High complexity
Customers	Medium (3)	High (3)	High (3)
Company size	Low (2)	Medium (2)	Medium (2)
Certification & partnerships	Low (2)	Medium (2)	High (2)
Financial information	Low (2)	Medium (2)	Medium (2)
Services & products	None (0)	Low (1)	Low (1)

9. Overall cost

"Hope costs nothing"

Sidonie-Gabrielle Colette, French novelist

Last but not least: costs … the trickiest part of all. This is the trickiest part because software cost estimation is not straightforward. When evaluating costs try to evaluate the overall cost not just the development costs.

There are a lot of factors that are influencing the overall cost; here are a few of them:

- Estimation complexity is directly related to systems complexity. The overall estimation effort required for highly complex projects is considerable. Could be weeks or months.
- Development costs are only part of the overall cost. The percentage of development cost very much depends on the project's requirements. In figures the effort distribution could look like this: analysis (15-20%), architecture & design (10%), development (45%), testing (15%), project management (10-5%), deployment & others (5%). This would be a typical effort distribution for medium complexity projects. So development costs are only 45% of the overall costs. The difference of 55% is composed of activities that have at least the same importance. So make sure you are counting everything not only development costs.
- As a general rule I would say that development costs can vary in the 30%-80% range. 30% or even less for mission critical systems, 80% for short term, low complexity projects.
- The productivity difference between two programmers has considerable effects on overall costs. An excellent software engineer can solve things up to ten times faster than a junior engineer.
- The right choice of technologies can have a beneficial effect on costs. For example if you are developing a community web site you might need a public forum. In most cases the right choice would be to use an existing third-party solution and integrate it, instead of writing your own.
- Poor quality is very expensive. You might think that poor quality should come cheap. My opinion is that in the long-term that is not true.
- Maintenance/operating costs are also important and you should count those too. Estimate the operating costs for the entire period you are planning to use the system.
- Warranty will protect you from defects that will appear after the software is released (post-production)
- Hardware and software requirements. Software that is designed to work on open source operating systems (ex: Linux) or open source databases (ex: MySql) will have no costs related to software licensing.
- Try also to quantify your team's effort as it counts for the overall cost.

The common pitfall when evaluating multiple providers is to compare just the man-day rates and if one rate is higher than another to assume that the high rate provider is expensive. The man-day rate is less relevant so you should try to estimate and compare the overall cost. The problem with this approach is that at some point you will need a cost estimate from the provider in order to estimate the overall cost. For low complexity project or projects with very clear documented requirements the estimation process may be straightforward, and the provider may be in the position of giving an accurate estimation rapidly with a reasonable error margin. If requirements

are unclear, in order to have a cost estimate with a reasonable error (maximum +/- 15% error), at least the scope should be clarified first: what the system should do, what is in scope, what is out of scope.

A good, tried & tested approach when scope and requirements are unclear is to split the project in two phases. In the first phase scope and requirements will be clarified (for example: 20% of the effort), and in the second phase the actual development and all the other activities will take place (80% of the effort). The first phase can be contracted on a time & material basis and the second phase can be fixed price or also time & material.

This approach has considerable advantages:

- Minimizes provider related risks for the customer. You will have the chance of evaluating the provider during the first phase. Quite useful if it is the first time you are working with that provider. If something goes wrong you will be able to walk out.
- Will motivate provider's performance. Their interest is to win the second phase contract. The first phase requires senior resources and has a lower margin.
- Minimizes project risks. If something goes wrong you will have the option of canceling in early stages.

If the above is not acceptable try spending some effort to clarify the scope in order to request a reasonably accurate estimation. If it is not possible to clarify the scope one way or the other, before contracting a provider, do not use the cost criteria in the evaluation process, it is not relevant.

How to evaluate the overall cost?

The suggested evaluation method for the overall cost is slightly different. Try to estimate the costs for each component in the table below. Ask the provider for additional costs not only development costs (extended warranty, maintenance, support, hardware, licensing costs, etc.). After you have finished evaluating all providers you will have a set of overall costs (one for each provider). You will have a minimum cost (N) and a maximum cost (M). Divide the distance between M-N into 5 ranges:

- between N and N + (M-N)/5;
- between N + (M-N)/5 and N + 2 x (M-N)/5;
- etc.

Providers with the overall cost in the lowest range will have the highest score (5), and providers in the highest range will have the lowest score (1).

Cost	Value	How to estimate
Project development	?$	Estimate all project related activities costs (not only development)
Maintenance/operating	?$	Estimate the operating costs for the entire lifetime of the solution.
Warranty	?$	A warranty period will save some costs. No warranty will create costs.
Hardware acquisition	?$	Hardware acquisition costs. Hardware maintenance costs.
Software acquisition	?$	Software acquisition costs. Infrastructure maintenance.
Own effort	?$	Estimate the costs of your team's effort
Any other costs…	?$	Add any other costs that are influencing the overall costs according to your project's requirements. Examples: localization costs, training costs, deployment costs, etc.
TOTAL	$$$	

Part III – Additional information

10. Top 10 priorities

1. **Tradeoff**

There are three main variables that are influencing a software project: Features, Schedule and Costs. These are directly related: a change of one variable will affect the others. There are no fairytales in software, so you are allowed only one wish instead of three … a constrained wish. You cannot just have the greatest software in the world, for free and available tomorrow. So be prepared to agree on a tradeoff as early as possible and define your priorities. Ideally the tradeoff should be agreed at the beginning of the project and should be clear for everyone involved in the project.

Tradeoff example: Given a fixed schedule, features will be chosen and costs will be determined. So your wish is: "I want the system to be delivered by this date, let us decide on the feature set that can be delivered within the time frame and let us know what the overall costs are".

2. **Customer involvement & Communication**

Involve yourself. Involve your customer and users. Carefully review specifications and builds. Communicate efficiently as often as possible. Do not ignore or hide problems under the carpet.

3. **Define a clear scope**

Define the scope of the project. Define what is in scope and what is out of scope. Avoid gold plating especially for first versions of a system. Prioritize features: Core features, important features, nice to have features. Follow the tradeoff and be prepared to cut.

4. **Stable documented requirements**

Clarify, document, review and formally agree on the requirements. Do not start anything if scope or requirements are unclear. Avoid gold plating. Keep it simple. If the scope is too big and reaching an agreement takes a long time break the project scope into multiple versions.

5. **Prototype**

If your system will have a user interface ask your provider to develop a user interface prototype. The behavior of the future system will be easier to understand this way. The user interface prototype is the visual representation of requirements specification. This will help visualizing the system before it is developed. Develop prototypes if necessary to confirm technical feasibility. This is the cheapest way to ensure that things are possible.

6. **Plan and track everything**

Measure everything ten times and cut only once. Spend time in planning and reviewing plans. Do not spend too much time planning. Ensure that all activities are planned. Think of B plans. Plan defensively, code aggressively; you may end up just in time. Update the plans when something changes. Track progress. Ask for status reports. Discuss problems. "When something can go wrong, it will".

7. **Test everything**

There is never too much testing. Review every document especially in the initial phases of the project. Ensure the provider is performing appropriate testing. Watch the bug lists.

8. **Iterative development**

Do not develop everything at once. Do it iteratively. Review, test and move to the next iteration. You will certainly have your share of surprises, but at least this will happen during the project not at the end when you might be overwhelmed.

9. **Keep it simple**

Solve business problems, do not create technical problems. Do not overcomplicate things. Frequent operations should be easily accessible, complicated operations should be possible with a reasonable effort.

10. **Control changes**

Plan and architect for changes. Changes will happen. Control, clarify, estimate, assume and agree changes. Update plans and artifacts when changes occur. Changes always have an impact.

11. Evaluation tools

There are two evaluation tools provided with this guide: a due diligence questionnaire and an evaluation form preconfigured spreadsheet. Both of them and this document can be downloaded from the following address: http://www.essensys.ro/en/HowToEvaluate.aspx. At this address you will also find an online version of the guide.

Due Diligence Questionnaire

This is a standard template that you can distribute to your candidate providers as a first step in the evaluation process. It contains the following sections:

- General Company Information
- Staff
- Customers & projects
- Software Development Process
- Technologies

The template also contains brief guidance and what details are required for each section. The template can also be used as the main evaluation tool if the process is conducted remotely. However in this case I recommend customizing the template according to your specific requirements.

Evaluation Form

This is a Microsoft Excel spreadsheet file that you will use during the evaluation process to store and analyze the evaluation results. It contains several sheets:

- Settings
- Results
- Provider 1 - 5
- Lists

The **Settings sheet** contains the evaluation process settings. This is where you can configure how scores are calculated. The sheet has two sections: "1. Overall scoring settings" and "2. Detailed scoring settings".

The overall scoring settings section determines how the overall score for each provider is calculated. In this section you should assign a weight for each high level evaluation criterion. The sum of all weights should be 100%. The overall provider score is calculated as the sum of each high level criterion score multiplied with the given weight.

Overall Score = $\sum High\ level\ criterion\ score\ *\ Criterion\ weight$

The detailed scoring settings section determines how the high level criteria scores are computed. In this section you will adjust the impact levels (0 - None, 1 - Low, 2 - Medium, 3 - High) for the detailed criteria. The high level score for a given criterion is calculated as the sum of each detailed criterion multiplied with the given weight. The weight for each detailed criterion is calculated as the impact level (0-3) divided by the sum of all impact levels associated with a high level criterion.

Criteria	Weigth
Overall distance	5,0%
Software development process	15,0%
Quality	20,0%
Technology	15,0%
Company information	15,0%
Costs	20,0%
Subjective score	10,0%
	100,0%

Detailed Criterion Weight = $\frac{Detailed\ criterion\ impact\ level}{\sum Detailed\ criteria\ impact\ levels}$

High Level Criterion Score = $\sum Detailed\ criterion\ score\ *\ Detailed\ criterion\ weight$

Example:

Overall distance	Impact level	Impact level value	Weight
Language distance	Low	1	=1/6 = **16.66%**
Cultural distance	Medium	2	=2/6 = **33.33%**
Time distance	None	0	=0/6 = **0%**
Geographical distance	High	3	=3/6 = **50%**

Assuming the language distance score is 5, the cultural distance is 4, the time distance is 1 and the geographical distance score is 2, and considering the settings above, the Overall distance score will be:

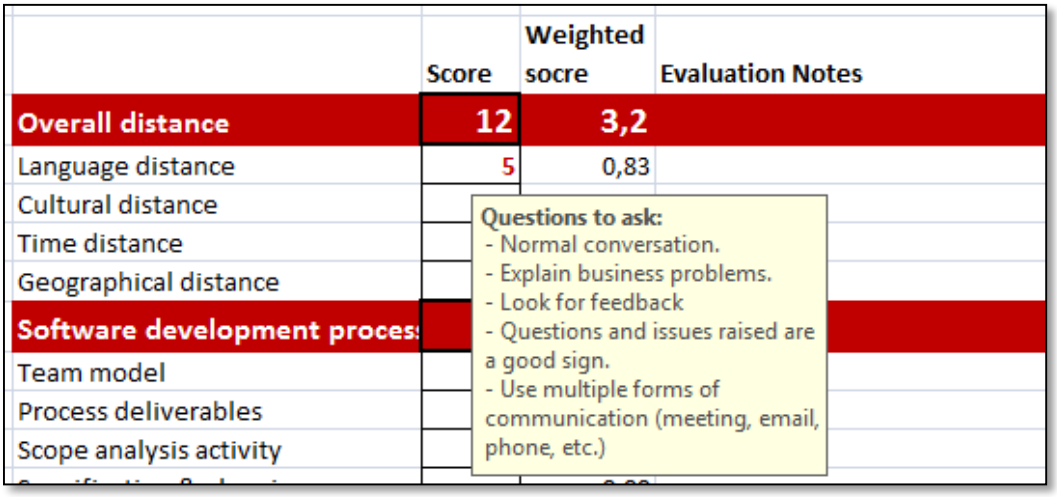
5 x 0.1666 + **4** x 0.3333 + **1** x 0 + **2** x 0.5 = 3.1662

	Impact level	Weight
Overall distance		
Language distance	Low	16,7%
Cultural distance	Medium	33,3%
Time distance	None	0,0%
Geographical distance	High	50,0%
Software development process		
Team model	High	18,8%
Process deliverables	High	18,8%
Scope analysis activity	Medium	12,5%
Specification & planning	High	18,8%
Design & architecture	High	18,8%
Development	Medium	12,5%

The **Results sheet** contains a list with all the providers. For each provider it displays the high level criteria scores, the total number of points and the overall score. These are automatically calculated using the settings you have specified in the Settings sheet. You can use this sheet to see the overall aggregated results of the evaluation process.

Provider	Overall distance	Software development process	Quality	Technology	Company information	Costs	Subjective evaluation	Totals points	OVERALL SCORE
ABC Company	3,2	4,2	3,8	0,0	0,0	5,0	5,0	105	3,04
XYZ Company	4,7	5,0	3,8	0,0	0,0	5,0	5,0	111	3,23
Another company	3,2	4,2	3,8	0,0	0,0	5,0	5,0	105	3,04
4th Company	4,7	5,0	3,8	0,0	0,0	5,0	5,0	111	3,23
Last Company	3,2	4,2	3,8	0,0	0,0	5,0	5,0	105	3,04

Provider sheets (1-5) contain the individual provider evaluation forms. This is where you will store the detailed scores for each provider. During your discussion with the provider it would be useful to have the workbook open on your laptop and grade the provider. When you select a given cell in the Score column a tip box will be displayed containing useful tips that you can use during the conversation. You can also use the evaluation notes column to annotate relevant information. This is how a provider sheet looks like.



The workbook contains 5 provider sheets. If you are evaluating more than 5 providers you can just clone one of the sheets and adapt the Results sheet by adding additional rows.

12. Further Reading

Books

Software Project Survival Guide (Steve McConnell, Microsoft Press)
http://www.stevemcconnell.com/sg.htm
More about software requirements: thorny issues and practical advice (Karl E. Wiegers)
https://www.microsoft.com/MSPress/books/9347.aspx
Peopleware: Productive Projects and Teams (Tom DeMarco, Timothy Lister)
http://www.dorsethouse.com/books/pw.html
The Mythical Man-Month: Essays on Software Engineering
http://www.pearsonhighered.com/academic/product/0,,0201835959,00%2Ben-USS_01DBC.html
Joel on Software (Joel Spolsky)
http://www.apress.com/book/view/9781590593899
Software estimation: demystifying the black art (Steve McConnell, Microsoft Press)
http://www.stevemcconnell.com/est.htm
TECH: Practices of an agile developer: working in the real world (Venkat Subramaniam, Andy Hunt)
http://pragprog.com/titles/pad/practices-of-an-agile-developer
TECH: The Pragmatic programmer: from Journeyman to Master (Andy Hunt, David Thomas)
http://pragprog.com/the-pragmatic-programmer
TECH: Rapid Development: Taming Wild Software Schedules (Steve McConnel)
http://stevemcconnell.com/rd.htm

Online resources

Microsoft Solutions Framework
http://www.microsoft.com/technet/solutionaccelerators/msf/default.mspx
Rational (Unified Process & Tools)
http://www-306.ibm.com/software/rational/
http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf
Agile
http://www.agilealliance.org/
http://www.agilemanifesto.org/
Capability and Maturity Model Integration
http://www.sei.cmu.edu/cmmi/

13. About this guide

In time several of our customers, partners, friends or colleagues looking to outsource software work, asked us how to evaluate software companies and what to look for. The question kept popping up, and every time the answer was the same: "There is no short answer to that". Fortunately, in time, Essensys has also been the subject of being evaluated by potential customers, so we have had our share of experience in this area. Sometimes the process was very formal and thorough, on other occasions, it was very informal which might work for small size projects but it is not a practice we would recommend. For this reason we considered that such a guide would be a useful resource for people with a non technical background who are in the position of evaluating software providers.

About Essensys

What we do?
Quality Software on Time. Every Time!

Who we are?
We are a software solutions provider with offices in Bucharest (Romania) and Rotterdam (Alblasserdam, Netherlands).
We are a dynamic and professional team of software engineers, an organization based on trust, verticality and determination.

What makes us different?

WE DELIVER

No excuses, no blame game. We communicate, we understand and we deliver quality software on time! Everytime! No matter how difficult it may be.

WE BUILD LOYALTY

We have learned that trust is built on results and we always aim for long-term partnerships. Over 80% of our customers are recurrent with multiple projects delivered.

WE CARE ABOUT OUR WORK

It is not just a job. We take pride in our work. We enjoy building software that matters. Software for people that solves real problems. We always enjoy a challenge and we take pride in our results.

Please visit our website for additional details: **www.essensys.ro.**

About the author

My name is Mihai MATEI. I am the General Manager of Essensys Software, a development software company based in Romania. I have a background of more than 14 years in the software industry and I am a Microsoft Certified Solution Developer (MCSD.Net) and a Microsoft Certified Trainer (MCT).

I hope you found this guide useful and I will welcome your thoughts, comments or inquires. Feel free to drop me a line anytime at Mihai.Matei@Essensys.ro

Copyright information