A Practical Guide to Outsourcing Your Software Development
The challenge
Why do firms outsource, and when may it be right for you?

How to meet the challenge?
Project type impact on scope, process and team management

WHAT - Scope Management
  Why scope is crucial
  Scoping successfully - Requirements and Tech Specs
  Working without an upfront scope definition
  Scope change management
  Scope creep and lack of control
  Scope division
  Expectations
  Paying: fixed price or by the hour?

HOW - Process Management
  Team structure
  Development methodology: Waterfall or Agile?
  Tools and metrics
  Quality Assurance
  Interaction and communication

WHO - The Customer / User and Project Team Management
  Your customer / user
  Your firm
  Your vendor
    Notes on vendor selection criteria
    Technical matters
    Manpower
    Do you need a team or a freelancer?
    Team selection tips
    Location
    Due diligence and final selection
    A fast vendor checklist
Software development is no easy task. Outsourcing software development is an even more challenging endeavour. Still, many find good reasons to try and meet the challenge.

**THE CHALLENGE**

The Standish Group’s reports on the development of software have shown both the steady rate of improvement in the field of developing software and the lingering spectre of disappointment and failure surrounding development projects.¹ Software development is no easy task. On top of that, for many reasons to be touched on in short order, many firms, both software and non-software producing, have turned to outsourcing for their software development, thus making it an even more challenging endeavour. Outsourcing has not proven to be a magic wand, however; it has failed to deliver the expected results over fifty percent of the time.² But despite this fact, outsourcing continues to be popular and widely used. A report by Gartner predicted that worldwide spending on IT outsourcing would rapidly increase from $268 billion in 2009 to $325 billion by 2013.³ Why is that? Research has suggested that IT outsourcing can provide firms with various economic, technological, and strategic benefits, such as reducing IT operation costs, improving technical competence, and even providing firms with competitive advantages. The outsourcing benefits, however, come together with additional challenges in software development projects, which need to be proactively addressed if the benefits are ever to be reaped by the outsourcing company.

What are the ingredients of a successful outsourced software development project? How can you ensure your outsourced project joins that lucky upper half of successful projects?

Before we begin to answer those questions, we should take a fast look at the reasons firms outsource in the first place.
WHY DO FIRMS OUTSOURCE, AND WHEN MAY IT BE RIGHT FOR YOU?

Cost savings in staffing and real estate are always the most obvious reasons for outsourcing but are not the only or – in some cases – the most important ones. Outsourcing may, additionally, allow a business to:

1. **Focus on core activities**, rather than non-value added work
2. **Add staffing flexibility** for special projects, including **skills not presently available** in your own organization
3. **Reduce development time** due to faster ramp up
4. **Improve quality** by working with more experienced developers
5. **Improve management** by taking advantage of the vendor’s experience and knowledge.

Whatever the reason(s), be absolutely clear about WHY you want to outsource. With a clear purpose in mind, you will be much better equipped to make reasonable sourcing decisions.⁴
How to meet the challenge?
Software product development is a challenge. Outsourcing software product development is an even bigger challenge, as evidenced by the number of publications on the topic. Under such circumstances, the software development process is spread across more than one organization and becomes a complex management endeavour with multiple issues to address. Outsourcing entrepreneurs and firms need to effectively manage:

1. the **scope** of their software development projects,
2. the **process** through which deliverables are implemented and
3. the **actors** involved, i.e., their customers/users at the front-end, their in-house staff, as well as their vendor(s) at the back-end.

The management task is made no easier by the fact that **there is no one universal approach with which to handle all projects**. The project type and size determine how to best manage within the three areas mentioned. **All in all, you need to tailor your approach to match your specific project’s characteristics.**

In addressing the multiple challenges above, you need to:

- **identify the project type** you want to outsource and be fully aware of how it will impact the scope, process and the people involved in the venture
- **manage the WHAT**, i.e., specify the **scope** and/or adopt a way to manage scope if it cannot be specified fully upfront
- **manage the HOW**, i.e., adopt an appropriate **team structure** and design the **process** that fit your project type
- **manage the WHO**, i.e., get your **customers/users** sufficiently involved in the process, gain support from your **organization** as well as select and manage a **vendor** that fits the project scope and process requirements.

On the whole, your success will depend on how well you plan, organize, execute and control the areas specified. Failing to understand these factors and their relation to each other increases the chances of failure dramatically. At the very least, disappointment in the process and final product is highly probable.
**PROJECT TYPE IMPACT ON SCOPE, PROCESS AND TEAM MANAGEMENT**

In general the first thing to consider is the type of your project. All in all, the type of project will determine to what extent you are in a position to define scope upfront, and thus, the way you will manage scope, the process you should adopt to implement the project and also the degree of interaction with customers/users and the vendor.

*In short, different project types require different ways of going about them.*

A successful project requires the proper attitude toward the scope and its management, the selection of a suitable development methodology, and the formation of the right team. It is the project type (and size) that drives the scoping of the project. Project types run the gamut in difficulty of scoping from very easy to very difficult. On the easy extreme may be the porting of an existing application, i.e., adapting an existing, well-documented system to a different technology platform. On the most difficult extreme would be a new application with a high market risk. Such a project runs the greatest risk of cost and time overruns and even outright failure and requires substantial collaboration between the parties involved.
WHAT?
Scope Management
WHY SCOPE IS CRUCIAL?

Assuming you want to make profits rather than just products, view scope as the ultimate driver of your economic results. It ultimately determines the value you will deliver to your customers and, consequently, the value you will be able to generate from your product or software-enabled business venture. While implementing a tool to support a business process, its value will be expressed in terms of cost savings realised with the solution. On the other hand, your project scope will drive the costs of the whole product development life cycle; not only the expense of product development but also the cost of support and maintenance. Scope also impacts your project schedule, so do not overlook the cost of delay, i.e., the revenue you may lose as a result of deferring the launch of your product. Get it to the market too late and you may discover your target customers are already locked into competitors’ solutions. The cost of delay may be hard to see, yet it may be much more important than the tangible costs you experience when parting with cash. Make scope decisions with economic trade-offs in mind.

Failure to manage scope will be expounded on in a subsequent section.
SCOPING SUCCESSFULLY - REQUIREMENTS AND TECH SPECS

The scope of your project directly determines the development cost and time of the project. When the scope can be properly defined but isn’t, the usual problems which arise are unnecessary cost overruns and time delays. **If it is possible then, define your project scope upfront** to avoid such unpleasant surprises. In cases where you can define the scope of the project, the technical specification should include, at the **high level**:

1. An overview of the desired application, i.e., its **business purpose** and problem(s) to solve as well as a definition of success.
2. A listing of the product’s **functionalities and features**, as well as who the **users** will be.
3. **Performance requirements** regarding speed, availability, volume, and reliability.
4. **Technology, environment, and integration requirements** such as the programming language, the operating system, and the software the product must work with.

For the **lower level**, a tech spec should present potential **users, user stories or use cases and wireframes or mockups** of what you are working toward.

Conversely, it is **unwise to overspecify**. It consumes your time and your vendor may have better ways of doing things.
WORKING WITHOUT AN UPFRONT SCOPE DEFINITION

If you are working on an innovative product and are thus confronted with high market risks, you will most probably work along the lines advocated by S.G. Blank\(^7\) (Customer Development), E. Ries\(^8\), A. Maurya\(^9\) (Lean Startup methodologies) or else develop your venture with some other similar framework in mind.

In such cases, detailed upfront scope definition is very likely to be a wasteful activity.

It can lead you to develop a solution with no / poor market fit. You may even lose time and money on loading the product with useless features.\(^{10}\) When you cannot or should not define the scope of your project fully upfront, develop it step by step concurrently with development. As you progress with development, you gather customer/user feedback, verify your initial assumptions and modify your product backlog accordingly. Your software development house should have the experience of scoping and implementing projects using an agile approach.

Whatever approach most fits your needs, you should prioritize the desired features and functions, and group them into deliverable milestones before you start the development work. Involve every relevant person in the scoping process. Most importantly, consider engaging customers/users and the outsourced team in the specification process. Their questions and insights can provide you with much better software.
SCOPE CHANGE MANAGEMENT

Scope changes must be carefully controlled. If they increase workload, they will raise the costs and may lengthen the implementation cycle if the development team’s capacity cannot be scaled up.

>All changes in scope should be weighed economically. Does the cost of the change in time and money significantly improve the value of the product to your customer?<\n
Will the customer pay for the enhancement you want to add? The number one goal when developing a new application is often to get the minimum marketable solution to market quickly, not take forever to develop the perfect product.
SCOPE CREEP AND LACK OF CONTROL

Perhaps the worst situations in a software development project are “scope creep” and ineffective change control. They can cause:

- delays
- cost overruns
- lower quality
- unnecessary features
- excessive rework
- loss of staff motivation

While the first five problems of scope creep and ineffective change control are intuitive and obvious, the loss of staff motivation can be the most pernicious, synergistically causing the first three problems and, worst of all, attrition and its accompanying knowledge loss.
To the extent possible, keep parts of the project independent from each other. Foster accountability and ownership.

SCOPE DIVISION

Last to consider is a usually unseen situation, until a problem arises. That of scope division. Scope division is all about responsibility and ownership when some work is to be done in-house and some at the vendor(s). Without clear responsibilities, delays and bad code may result in meaningless, non-productive finger pointing which cannot be easily resolved. It is important that each party know what their role in the project is. Proper scope division ensures this.

The scope of the project must be identified at two levels: the entire scope of the project and the scope for the vendor(s).  

It is difficult to coordinate two, usually physically separated, teams working on the same part of the same project. To the extent possible, keep parts of the project independent from each other so that each team can make progress without depending on other teams to get things not only done, but done the way they should be.
Many people (clients and vendors alike) come away from an outsourced project with a bitter taste in their mouth due to **unrealistic expectations**. Outsourcing can provide many economic benefits, but it still must follow basic economic rules, namely, that it saves on wages and real estate costs, but it cannot always significantly change the necessary amount of time to do a task in. Great things cannot be done overnight, nor do quality people work for nothing.

**Most importantly, workload and, consequently, cost and schedule cannot be accurately estimated as long as the scope is not sufficiently defined.**

The person responsible for the project should define the project scope and - as realistically as possible - estimate the workload that will ultimately drive the budget. Knowing more about the scope of the project can also ensure the selection of the team with the right skills. If you are unsure about your project’s demands, a few potential vendors should be able to help you in estimating the time and monetary costs of the job using the information you provide. You will, however, need to provide them with reliable information with which to establish the scope of your project. Otherwise, their estimations may be equally unreliable.
PAYING: FIXED PRICE OR BY THE HOUR?

Before going further it would be prudent to consider the methods of payment for the vendor. A *fast rule* would be that

> a fixed price is best for simple jobs with well defined scope, and time and material charges are best where the scope is large and/or poorly defined.

As projects run the gamut of complexity, there is no reason to exclude both models of payment on the same engagement or parts thereof. Many projects benefit from a compromise called a "milestone budget" approach. The milestone budget simply breaks a larger project into smaller, more manageable projects for which it is easier to establish time and costs for the work to be delivered.
HOW?
Process Management
Controlling your outsourced project is another key ingredient of success or failure, and the business process is just as important as the software development methodology. Your overall approach should naturally fit the project type and the associated scope management techniques.

The management process includes:

- **Definition of team structure.** The team architecture and responsibilities must be clearly defined.
- **Development methodology.** The development methodology (waterfall vs. iterative/agile approaches) should match the project type and size.
- **Tools to manage the software development, project progress and collaboration.** Ineffective or non-existent tools will compromise the project.
- **Quality assurance.** Lack of or improper quality assurance will certainly leave you and your customers with an inferior product.
- **Interaction and communication processes.** These must be mutually agreed upon and adhered to by both you and the vendor.
TEAM STRUCTURE

Problems ranging from late delivery to bad quality often arise from improper team formation. These problems often occur when responsibilities for each side are not properly agreed upon.

Knowing the scope of your project, clearly define the structure of the project team, accounting for both internal and vendor resources you will need.

Specify roles and the associated responsibilities, document them well and ensure they are well understood by all the people involved.

At the very least you need two key roles in managing software product development. On your side, there should be an in-house product manager who is responsible for day to day interaction with the development team and, therefore, making sure the outsourced team stays on track and, ultimately, that the product is delivered. The product manager also takes care of the customer/user contributions as needed during the development process. This individual may be the “product champion”, a person who believes in the product and makes it their goal to get it to market. If you split the role into Project Manager and Product Owner, specify clearly their areas of responsibility and the associated TDRs to ensure ownership and clear lines of communication.

Secondly, there should be a Technical Leader, whether in-house or from the outsourced team, who is responsible for technical advice and the programming. It is up to them to manage the development team. As you are likely to be heavily involved in customer development activities, you may find it impossible to effectively manage individual technical product development resources; your Technical Lead should ensure things run smoothly at the back-end of the process. If this person is from the vendor, they should be responsible for the delivery of the product on their side. A proper team takes into account the differences between technical people and market-oriented business people. The leadership must be properly balanced to ensure both technical and market success.
DEVELOPMENT METHODOLOGY: WATERFALL OR AGILE?

Development methodology must be matched to the project type and size. As alluded to in the scope section, a simple project with a low market risk and an easily defined scope is a fine candidate for the waterfall approach to development. After the project is thoroughly defined, it is coded and tested. Any deficiencies are corrected and the product is put into service. However, using the waterfall approach to develop innovative software with unclear market fit, where scope is hard to define fully upfront, is a recipe for either fielding a bad or useless product or the product never making it to market in time to beat a competitor’s product. Even if the product does find traction in the market, limited / delayed exposure to customer / user feedback may lead to the product being burdened with superfluous features and functionalities. Such features contribute no value to the user, often undermine user experience and increase the cost of product support, maintenance and further enhancements.

Thus larger poorly defined projects with high market risk benefit from an agile approach to development.

In short, the agile methodologies use an iterative approach and a high level of collaboration between the market-oriented customer development team and the technically-oriented product development team.

The approach is often connected with early and frequent customer / user engagement in the development process. In developing a version 1.0 as fast as possible, the most important features are developed first. Small batches of code are tested early and often to identify deficiencies. Additionally, agile development is often combined with a modular approach to development to ensure independent functionality.

In general, the development methodology must be worked out cooperatively with your vendor. Their experience can guide you as to which path to take.
TOOLS AND METRICS

To execute effectively, you need a set of tools which enable you to manage the software you are developing, control your project scope, schedule and cost as well as ensure effective collaboration / communication with and among the people involved. Accordingly, tools are generally of three types:

- **Software development**
- **Project management**
- **Collaboration**

**Software development** tools include those which manage the requirements, control the source code and track bugs and improvements.

**Project management** tools are used to keep tabs on the progress of the project and scope and budget. These tools should be able to manage changes in the project as well as projecting the costs in time and money of changes.

**Collaboration** tools may be things such as teleconferencing applications, secure networks, remote access, and enabling software. Collaboration tools allow the members of each team to communicate and work together.

Having these tools at your side is not an incitement to micromanage. Management at your level is for the big picture, i.e., milestones. Your vendor should know how to get the job done at his level.

The tools you use should, among other things, enable you to gather and monitor important **metrics** on your development project. They are the means of **tracking progress** and introducing process improvements. Naturally, the **metrics must be tailored to the job**. For example, a work unit for a large project might be a use case, a user story or an acceptance test scenario; those for a maintenance project possibly bugs fixed and enhancements added.
There is a plethora of both commercial and open source solutions available. Decide on the metrics you will track and select and set up your tool kit. You may have your own solutions, still a proper vendor should be able to offer a set of tools and metrics with which to manage a development project. If they cannot give a fluent demonstration of their management tools as well as their practical applications on projects, you should consider it a red flag and choose to continue your search for a more mature service provider.
QUALITY ASSURANCE

Quality assurance is the job that Everybody should do but Nobody does, as the old joke goes. Before work begins, like with scope division, it should be clear who does the QA work, or what part of it. Does the vendor do the acceptance tests, the unit tests, both or neither? What should be covered with specific tests? You and the vendor should jointly establish:

- coding and documentation standards and controlling procedures
- bug tracking, prioritization and triage
- release plans for users to validate market fit
- release criteria regarding bug severity and quantity
- the types of testing to be performed, i.e., usability, regression, etc.

Without a well executed QA plan you are likely to burden your product with costly technical debt which may cause significant delays, cost overruns or, in the worst case, bring development to an unmanageable halt. The bigger the system you build, the bigger the impact of QA measures – for better or worse.

From another perspective altogether, QA work can be an effective proving ground for trying out potential vendors.
INTERACTION AND COMMUNICATION

Communication is always difficult when people are physically separated, even within the same building. In a matter as important as developing your software product or software-based business venture, communication is key, especially when your scope is difficult to define. Communication is a multifaceted issue interrelated with such things as:

- language and culture
- time zone (to be further described in the vendor section)
- domain knowledge
- travel demands for both sides
- chain of command/approval

Generally speaking, poorly defined or definable projects require much close, verbal collaboration, especially in the beginning phases of the project, making fluently spoken English – or other language of your choice - a necessary attribute of at least one member of the outsourced team. The fluency of spoken English also has a cultural side. The same words may very well carry completely different meanings for people with different cultural backgrounds. Cultural proximity or mutual understanding of the partner’s cultural context are strong foundations on which to build and maintain a good relationship. The domain knowledge is important in that the necessary vocabulary must be known by both parties, and can cost considerable time if you have to teach your vendor a new domain. In general, it is much more efficient to develop your product with those who have previous experience within your industry and the type of solution you are trying to develop.
Establish and maintain clear lines of communication.

Physical co-location of the teams, or at least key people, at important junctures in the development has much to recommend it.

Consider the near-shore formula with an opportunity to periodically co-locate with the vendor and thus boost the information flow between all the people working on the project.

You may choose to visit the vendor to work with the whole team on their premises or have a key member of the offshore team work on-site at your offices. Such a solution is particularly useful at the fuzzy front-end of the process when collaborating on requirements specification and system design as well as at important moments in the project implementation, e.g. when dealing with a major pivot. Physical proximity fosters communication and helps get the job done better and faster.

Last, people on both sides must know who the boss is in both places. They must know who to go to for the many issues, from technical to interpersonal, that can, and will, sprout up during the project. They must also know who is responsible for approvals of changes and final renditions.
WHO?
The Customer / User and Project Team Management
Many Product Owners see themselves and their technical product development teams as the main agents to manage in the development process. In doing so, they often fail to recognize their target customers/users should actively participate in the process. Some may recognize the need; still they just pay lip service to the notion and, in practice, focus too much on the safe and psychologically easier back-office activities. The result is often a product without a market fit or a product complicated by extraneous features.

To avoid such risks, clearly specify the role of the customer and/or user in your software product development project. Make specific plans for their involvement in the development project.

In principle, the desired degree of customer/user engagement depends on the type of project - the greater the market risk, the greater the involvement needed.

If you are working on a new application burdened with high market risk, strive for intense collaboration with prospective customers/users throughout the process. Involve the customer in the specification process from the beginning before any decisions are firm. Test your assumptions about the functionalities you think your customers are seeking. The customers’ input on iterations and releases will be critical in prioritizing work on bug correction and the next features to be added.

If you are re-writing an application just to improve its performance, and customer buy-in or usability is not an issue, you may limit the customer/user involvement in the process.

All in all, you need to be involved in both customer development activities at the front end and technical product development activities and the back-end.

The proper balance will depend on the characteristics of the project you work on.
Note: Despite outsourcing the development of the project, the product is still yours and you are responsible for it. It is up to you to manage the customer development, not the vendor.
As mentioned above, **you do not outsource the whole software development process** - you need in-house skills to actively manage the front-end of the process focused on collaboration with your customers/users and ensure product management. As you determine your customer’s needs on the front end, it is up to you to communicate them to the vendor. While you may be hiring developing skills, it is your managerial skills that ultimately lead to success; you need to effectively manage the vendor delivering the back-end of the process and provide effective supervision to track progress.

**You or your manager needs to be a goal-driven type with good managerial and intercultural skills.**

Prior **outsourcing experience** may prove to be invaluable. They must also be **given the authority to do their job** for the company, including receiving **complete and unequivocal support from all levels** of the organization.\textsuperscript{21}

Leading your in-house people falls under the category of preparation, and failing to lead them will cost you. You must see to it that your team has the skills needed for their part of the operation and can be made to see the benefit of outsourcing, not just the “threat to their jobs.” The good manager knows that strong incentives, worthy challenges, and proper objectives are important in keeping your key people for the duration of the project. **Make sure your in-house team is both capable and willing to outsource.**
The importance of the outsourced team cannot be understated. The most easily-defined, simple project can still go wrong if the wrong team and wrong process are employed. Conversely, a great team, aware of how to employ the proper methodology, can turn an entrepreneur’s rough product concept into a solid software solution.

It is important to understand the process of outsourcing not as simply purchasing a service from a company, but rather, hiring specific people who you want to work for you. Choosing your vendor is similar to recruiting new employees, but as a group.

You must check not only the vendor’s credentials, but also those of the specific team and the individuals who are to work on your project.

This is because those people will have the greatest influence on the product development process and its result. Besides, if you are successful, you are likely to collaborate with the specific individuals chosen for a long period of time.

With the above firmly in mind, you can now search for, evaluate and select a vendor who is in a position to fit the scope of your project and the process you want to apply. Do not be haphazard or count on luck - there is too much at stake. Follow a systematic procedure to increase the odds of partnering with the right vendor:

- Compile a list of key criteria your vendor should meet; weigh the criteria which will help with selection.
- Create a list of outsourcing vendors - Internet search and word-of-mouth are the most common sources you may use.
- Use the weighted criteria matrix to shortlist the best candidates.
- Carefully scrutinize the shortlisted candidates, i.e., the company, the team and the individual team members.
- Select the best vendor.
VENDOR SELECTION CRITERIA

TECHNICAL MATTERS

The most important thing to know is if your vendor is able to do the job at the technical level. Which vendor has the proper technology for the job, both in programming language(s) / frameworks and domain knowledge? Have potential vendors describe a few of their projects in the necessary domain, especially the challenges they overcame. Confirm it by talking to their clients. Technical matters also relate to development processes and quality assurance techniques. Verify their level of process maturity and establish how it compares to your own. Be sure to confirm that your potential vendor aligns with your goals and requirements. It is up to you to specify your needs and expectations. It is up to them to prove they are in a position to live up to them.

MANPOWER

Does your project need three people to work on it or ten times that? Your answer may lead you to a smaller, less well known vendor instead of the big names. Does that small vendor have a back up plan for attrition and absenteeism? How is attrition at the vendor, and how quickly is it compensated for?

What are the experience levels, and balance thereof, within the team? What history do the people on the team have working together as a team?

It is worth remembering that you are not hiring just one employee, but a whole team. (See the section on freelancers below.) The size and quality of that team determines the amount of management required.
DO YOU NEED A TEAM OR A FREELANCER?

Assess carefully who you need. On relatively small projects or when you need to augment an existing team, it may be a better option to hire freelancers. In general, freelancers are best hired to bring a higher experience level or a missing skill to an in-house team on an in-house project or to independently do a very small project. A good rule to keep in mind is that the less staff you employ the more important their individual levels of expertise become. On top of that, if you collaborate with individual freelancers, rather than a team/vendor, you need to be prepared to operate at a low management level, e.g. to manage at the task level, which entails a greater amount of communication, coordination and monitoring.

TEAM SELECTION TIPS

When selecting a team, be sure that:

- The team is the right size for your project.
- The team has an appropriate balance of experienced and junior staff members.
- The team will be focused on developing your software rather than their own or extended over too many engagements.
- The Team Lead does a good job managing their people and projects.\textsuperscript{24}
- The team, or at least the experienced core, is stable and not subject to high turnover.
LOCATION

On a working level, the **time zone differences** have a definite impact in the consideration of the vendor’s location. Your employees have lives outside of work, and it will affect the process negatively if they are put upon to come to work early or stay late on a routine basis without proper incentives. The same may be true for your vendor’s staff.

*How often you foresee needing to truly speak to the vendor determines the location’s importance.*

How the **vendor’s workday overlaps** with your own is particularly important if the project requires much communication and/or direct management.

That said, workshift overlap may not be critical for your project. Outsourcing half a world away can give you the advantage of **round the clock work** on the project, for example, if the vendor builds, then the in-house team tests or vice versa.

The location can also determine the vendor’s ability to speak English or, for that matter, any other **language** you need to communicate in. Conference calling can offer a good way to confirm sufficient language fluency. **Culture** is also an important thing to remember when choosing a location.

*Do the two cultures have an affinity for each other? Be sure that your team and the vendor’s can get along.*

Performing a modicum of research on the culture in question should prepare you for the experience.
Lost time due to travel should also be taken into account if you consider face-to-face collaboration. Does it take a whole day by plane to get to the offshore location? Are there visa considerations to take into account? How often may you need to do this? If you are not the one doing the travelling, how willing are your employees to travel and stay offsite, especially in a different cultural environment? Similarly, are the vendor’s employees ready to travel and work on-site at your premises? Do you have the necessary space to accommodate the vendor if they come to your location? Last, have the travel costs been fully taken into account and properly budgeted for?

**DUE DILIGENCE AND FINAL SELECTION**

After using the parameters above to make a “short list” of two or three vendors, the time has come to finally select your vendor. **The final selection process is similar to recruitment of new employees.**

- Check references for the vendor, the team and the individuals to be involved.
- Check resumes and/or social network profiles of the individual people comprising the vendor’s team; if possible, check their publicly available code repositories, e.g. on Github.
- Conduct interviews as this is a sure way to confirm language / communication skills.
- Consider doing a short pilot project to verify in practice what you have established so far.
- Consider an on-site visit to find out even more about the candidates’ competencies and personalities.

With the information gathered, select the finalist who seems to be the best match for your project.
A FAST VENDOR CHECKLIST

1. Can they help you scope your project and develop the tech spec for your project, if needed?
2. Do they have experience in your technology stack, domain and project size?
3. Do they have experience in selecting and using the appropriate development methodology?
4. Can they ensure effective scope change management, esp. by using tools allowing you to track scope, its changes as well as their implications for cost & time of implementation?
5. Do they have tools to manage the project and communicate and collaborate with you?
6. What QA systems do they have in place?
7. Are they willing to use different payment methods and/or match the payment method to the type of job?
8. Does the location of the vendor and its implications coincide with your business needs?
9. Does the vendor have enough of the right people for your project and a plan to accommodate attrition?
10. Is their price competitive?
SURE WAYS TO FAIL

- Don’t identify the scope, manage it effectively or see its impact on schedule and costs.
- Don’t realize that the development methodology should match the project type.
- Don’t use a well thought out, thorough and effective QA system.
- Don’t use proper tools or metrics to manage your software and track project progress.
- Don’t involve your customers/users in the development process.
- Don’t be careful and diligent when selecting your vendor.
- Don’t establish a clear chain of command within both organizations.
- Don’t recognize that business managers and technical managers think differently and need to complement each other.
- Don’t communicate with your vendor.
This is the way we do it at Selleo.
BUSINESS CASE

We determine the business context of the software system to be developed, i.e., find out about the business model and strategy of the venture to be supported by the platform and/or the business processes to be served by the software tool. This stage helps to identify the business needs to be satisfied and business objectives to be achieved with the software solution. The goals and objectives established play a vital role in scope management further down the road. We can provide assistance in developing the business case.

INITIAL SCOPE, PRELIMINARY PROJECT PLAN AND GUESSTIMATES

Based on the business case as well as the requirements delivered by the client, we identify the initial project scope and develop the preliminary project plan—we prioritize product features and group them into milestones taking into account both market and technological constraints involved. In particular, we try to establish what constitutes the Minimal Viable Solution. With the plan at hand, we develop a rough estimate of time and cost needed to implement the project.

TERMS OF COLLABORATION

At this point we and the client settle the terms of collaboration, including process and methodology, the team architecture, team members’ roles and responsibilities, project technical environment, payment terms, intellectual property, etc.

PROJECT IMPLEMENTATION STRATEGY

We jointly adopt a Project Implementation Strategy consisting of guidelines used to make complex trade-off decisions regarding functionality, cost, time, and maintenance expense.
FIXED-PRICE ENGAGEMENTS

MILESTONE TECH SPEC

When we work on a fixed-price basis (milestone budgets), milestone requirements are converted into a detailed Tech Spec consisting of a complete set of wireframes (mock-ups) and a set of user stories. The latter are converted into acceptance test scenarios. Regarding a detailed Technical Specification, our Clients can either develop it themselves or commission Selleo to do the job. On small and well defined projects, tech specs can be developed for more than one milestone.

MILESTONE BUDGET AND IMPLEMENTATION SCHEDULE

Based on the detailed milestone tech spec, we verify the cost and deadlines for milestone implementation. In principle, specific work orders, deliverables and their costs as well as implementation schedules are agreed upon on a periodic basis for each successive project milestone. A milestone is usually structured so that it may be implemented within two weeks.

MILESTONE IMPLEMENTATION

When the Tech Spec for a milestone is finished, we start its implementation. After it has been completed, it is presented to the Client for acceptance.

While implementing a given milestone, we develop the Tech Spec for the successive stage, so that development can start immediately after the previous milestone has been completed.

If the Client requests changes during a specific milestone implementation, we assess the impact of the change and jointly decide whether to implement it within the milestone or move the changes to later iterations.
TIME & MATERIAL / HOURLY-RATE ENGAGEMENTS

On hourly-rate engagements the Tech Spec is developed progressively and concurrently with development. The Client is free to choose whether they prefer to collaborate on a daily basis or develop a larger Tech Spec batch to convene less frequently.

The Tech Spec items are guesstimated before they are implemented, still the Client is charged for hours actually worked on the project within a period of time agreed. They receive summary weekly workload reports to monitor progress and costs and are invoiced on a biweekly basis.

PROJECT REVIEWS

After each milestone the project scope, release plan, expected costs and timelines are reviewed and updated. In particular, the scope of the project is often modified to account for user feedback gathered, be it after an internal alpha or external beta release.

PROJECT COMPLETION

When the project has been completed, Selleo may be commissioned to deliver support and maintenance services under a Service Level Agreement.
SELLEO’S TOOL KIT

You may have your preferences when it comes to the tool set, in which case your vendor should be able to use the set of tools you intend to apply on your project to keep it on track. If you do not have a predetermined set, you may check what the vendor has to offer in that respect - many programs exist that may satisfy your requirements. What is important is that the vendor can both tell you what they use and that they can prove they actually use it on their projects.

We have used the tools presented below on projects we have developed with our Clients:

**Software Management Tools:**

- balsamiq
- justproto
- creately
- Cucumber
- steak.fm
- ENGINE YARD
- TeamCity

**Project Management Tools:**

- Cucumber
- steak.fm
- REDMINE
- DIAMOND MINE

**Collaboration Tools:**

- REDMINE
- Basecamp
- skype
- GoToMeeting
TWO BRIEF SELLEO CASE STUDIES

WATERFALL APPROACH

A network of pubs which had been using manual methods to track inventory and cashflow and make weekly and monthly reports for both individual locations and the whole network decided it wanted a web enabled BPO tool. The client wanted a system similar to the original tools, but also wanted to be able to either automatically generate reports or to do so on demand. The scope of this small project was easily defined, and we developed a budget for the whole project. The end product was implemented by dividing the project into milestones and delivering the most needed features first.

AGILE APPROACH

A client needed a multi-sided platform for gathering and selling business leads. The idea was to create a marketplace for business leads where agents can post their leads and customers (suppliers) may buy them. The customer provided a written explanation of the platform, describing its business goals and typical users. Working closely with the client, we refined their requirements and broke the work down into manageable pieces. Using the requirements, we formulated detailed specifications for each successive milestone. Selleo used a budget system and fixed timelines to develop successive iterations of the product. After the third milestone the product was being tested by the customer, after which they started to develop additional ideas. At the seventh milestone, the customer and select clients were using our solution internally with real data.
WHY SELLEO?

Selleo is an example of a firm that has a lot going for it. Our Ruby on Rails software development house was set up in 2005 and has served entrepreneurs and enterprise intrapreneurs from 9 different countries in Europe and America ever since. Our close-knit teams have gained expertise and developed solid design and programming skills as well as an efficient, result-driven, nearshore / offshore software development process.

The 30 projects or so we have implemented in diverse business domains so far include e-business and e-commerce systems, Web 2.0 solutions, SaaS / Cloud-based products, business process optimization tools and mobile solutions.

Located in the south of Poland, we can serve all of Europe as a near-shore provider, allowing fast visits from the client or on-site service with low travel costs. Our location even allows partial workshift overlap with the eastern U.S. and Canada.

Last, we are not only about technology. Our business is not only software development, but also web business consulting. Our focus on services industries has often allowed us to go beyond technology and provide valuable product and business insights to proactively co-create web solutions with our clients.

For guidance on your business’s software needs, feel free to contact us at Selleo. We may be reached by mail, phone, fax or email, and remember, if we cannot help you, we may know someone who can.

Selleo
ul. Kaminskiego 19
43-300 Bielsko-Biała
Poland

Phone: (48) 33 498 65 84
Fax: (48) 33 487 65 18
http://www.selleo.com
info@selleo.com
1 The Standish Group; *CHANG Summary 2009*; 2009


3 Harris, J. et al; Forecast: *Outsourcing worldwide forecast database, 2000–2013*; Gartner; Stamford, CT; 2009


5 Reinertsen, D.G.; *The Principles of Product Development Flow. Second Generation Lean Product Development*; Celeritas Publishing; Redondo Beach, California; 2009


7 Blank, S.G.; *The Four Steps to the Epiphany. Successful Strategies for Products that Win*; 2007


9 Maurya, A.; *Running Lean. A systematic process for iterating your web application from Plan A to a plan that works*; 2010
Cusumano, M.; *The Business of Software: What Every Manager, Programmer, and Entrepreneur Must Know to Thrive and Survive in Good Times and Bad*; The Free Press; New York; 2004

Denne, M. and Cleland-Huang, J.; *Software by Numbers: Low-Risk, High-Return Development*; Prentice Hall; New York; 2004

Sood, R.; IT, *Software and Services: Outsourcing and Offshoring. The Strategic Plan with a Practical Viewpoint*; AiAiYo Books, LLC; Austin, Texas; 2005


Poppendieck, M. and Poppendieck, T.; *Lean Software Development: An Agile Toolkit*; Addison-Wesley; Boston; 2003

Poppendieck, M. and Poppendieck, T.; *Implementing Lean Software Development: From Concept to Cash*; Addison-Wesley; Boston; 2006


21 Sood, R.; IT, Software and Services: Outsourcing and Offshoring. The Strategic Plan with a Practical Viewpoint; AiAiYo Books, LLC; Austin, Texas; 2005

22 Mezak, S.; Software without Borders: A Step-By-Step Guide to Outsourcing Your Software Development; Earthrise Press; Los Altos, California; 2006

Note! The book by S. Mezak contains an immense amount of invaluable tips for those who are involved with software development outsourcing.


24 Upadrista Venkatesh; Managing Offshore Development Projects: An Agile Approach; Multi-Media Publications Inc.; Oshawa, Ontario; 2008