TECHNICAL MEMORANDUM

To: All Personnel

Project: General Information

From:

CC:

Date: 12 Jun 11 Last Modified: 06 Jul 11

Subject: A Brief Guide to Successful Development Outsourcing

---

1. INTRODUCTION

This document is intended to provide a roadmap to the new or existing development organization that is contemplating outsourcing some or all development to external organizations or consultants. It is, by its brief nature, not comprehensive in detail, but does describe the major requirements and considerations that are necessary to a successful outsourced operation.

2. OVERVIEW

It is assumed that the target audience for this document:

Either has already developed software products, or is contemplating development of new products

Either doesn't want to actually carry out the development or needs to produce more deliverables than can be accomplished in the desired timeframe with existing staff.

Is comfortable with the concept of outsourcing software development work to individuals or organizations external to the existing organization.

This document addresses this topic in the following major divisions:

- **Definition of Terms and Assumptions.** What we're talking about, and what we're thinking..
- **Project Management and Client Involvement.** Describing the role of the Project Manager, the necessary involvement of the existing organization, and management of the outside entities.
- **New or Existing Development Organizations.** Considerations when there is no existing development organization, and when outsourcing some or all of existing or new development from an extant organization.
- **Definition and Documentation of the Environment.** Requirements before actually beginning the outsourcing effort.

- **Identification of Candidate Developers.** Criteria for interviewing and acceptance of development organizations or individuals.
- **Closing the Deal.** Once agreement has been reached, and a developer is identified, establishing the legal relationship.
- **Requirements, Project Plan, and Milestones.** Defining expectations and measuring progress from design through final acceptance.
- **Ending the Project.** Even in ongoing relationships, there must be final closure for any project. In cases where the relationship terminates at the end of the development effort, additional considerations are in order.

3. DEFINITION OF TERMS AND ASSUMPTIONS

3.1 DEFINITIONS

Client The organization or individual specifying the product and utilizing the external developer.

Contractor The external developer—whether a group, or an individual.

Development Environment All aspects of the production of software—including, but not limited to:

Development platform hardware and software

Software development and diagnostic tools

Document preparation tools

Project management tools

Any and all other software or manual procedures or processes necessary to the development and production cycle

Execution Environment All aspects of the target hardware and software environment on which the work product must execute.

Project Manager The individual identified as the responsible party for auditing all aspects of the development effort, carrying out the business directives of the Project Sponsor, and facilitating communications between the Client and the Contractor. Ideally, the Project Manager (or **PM**) should be involved from the inception of the outsourcing effort.

Project Sponsor The individual or group within the Client organization that "owns" the project. The Project Sponsor defines the requirements and final acceptance of the Work Product, and is responsible for all decisions affecting the initial and ongoing development effort.

Version Control A means of monitoring and controlling all aspects of a project during and after development by defining a collection of source, binary, and documentation that comprise a known state of the system—usually a "checkpoint" or "release" version. This is almost always accomplished through a software control system.

Work Product The desired deliverable software system, including all supporting documentation.

3.2 ASSUMPTIONS

3.2.1 Legal

It is assumed that any project will require the Client to divulge information to the Contractor. This may or may not involve sensitive or proprietary information; in any case, the Legal Department, or an attorney versed in contract law (especially as it pertains to software contracts) should be consulted to produce the appropriate Non-Disclosure agreements, Statements of Work, contracts, or other legal documents.

3.2.2 Client Involvement

Close contact between the Client and Contractor is assumed throughout the outsourcing effort. Simply put, if you expect to be able to find someone to go off into a corner and produce software without requiring time or effort on your part, you are guaranteed failure. Most likely, expensive failure, with recriminations and other unpleasantness.

Ideally, the time required will be far less than would be necessary if you were to design and develop the software. You may be able to forego hiring developers (or more developers) with specialized (and expensive) skill sets.

But you *will* have to be involved and committed to get the desired results.

3.2.3 Offshore Development: A Caveat

I want to state up front that, while all of the issues discussed in this document could apply equally to on-shore/local outsourcing and offshore outsourcing, I'm simply not a big fan—or, rather, a fan at all—of offshore outsourcing. Most development organizations I've encountered have problems properly managing software development when everyone involved is from the same culture, speaks the same language, and are in the same physical location. The effort becomes immeasurably more difficult when trying to manage an effort that may be twelve time-zones away, using management and staff for whom English is a second language, face-to-face meetings are difficult or impossible, and even cultural assumptions can cause miscommunication. Studies have shown that upwards of 60% of offshore projects have not ended to the satisfaction of the Client; the decision to offshore development should be undertaken only after exceedingly careful examination of the anticipated benefits and potential difficulties.

That said, if you simply *must* go off-shore, please consider the following:[1]

> Off-shore developments that I've heard of that have worked often include:
>
> - A period of time in which the offshore developer(s) visits the client site and learns the client business culture. OR:
> - A member of the client staff relocates to the contractor site for the duration of the contract to oversee/manage the project.
>
> A potential difficulty in option 2 is "Stockholm Syndrome": the employee at the contractor site may begin to identify more strongly with the contractors than the home office. Frequent visits to the home office helps to alleviate this.
>
> Short form: Include a lot of travel in the budget.

[1] Thanks to Clif Flynt, who provided this almost verbatim.

4. PROJECT MANAGEMENT AND CLIENT INVOLVEMENT

It can't be emphasized enough that *somebody* has to be the Project Manager. This may be an individual from the Client organization; it may be an external consultant or contractor hired for this specific task; or it might even be provided by the Contractor, if it is an organization with the skills and personnel to carry out the task.

In any case, the Project Manager has to be an advocate for the Client—if that isn't clear and accepted, choose another PM.

The PM must keep a close finger on the pulse of the project. He or she must communicate with both the Client and Contractor on a regular basis. Compliance with predicted milestones and deliverables, and remediation of deviations are a primary responsibility.

5. NEW OR EXISTING DEVELOPMENT ORGANIZATIONS

There are advantages and disadvantages to either having an existing development environment and organization, or starting from a clean slate. However, in both cases the key requirement is fully defining the tools and processes necessary to produce and maintain software.

5.1 NEW ORGANIZATIONS

The advantage of a new development organization is the ability to design the design process and specify tools and resources available for developers. If all development is outsourced, it's possible to avoid having to acquire, install and maintain the actual development platforms and support tools.

The disadvantage, of course, is that it is necessary to address and define all aspects of the development and support environment—and get it right the first time.

In any case, the minimum requirements to define a development environment should be considered to be:

- **Acceptable hardware development platforms**. Minimum hardware requirements, development operating, storage and backup systems.
- **Acceptable software development platforms**. Required and acceptable languages, development and testing tools.
- **Version control environment**. This is a critical resource—it must not be skipped or skimped, and there's really no reason to not make it an integral part of the development environment. There are truly excellent, full-featured FOSS (Free Open Source Software) packages available, such as Subversion, Fossil, and Git.
- **Test environment**. This isn't optional, either. You need to be able to test the software—and not on your production environment, but on an environment that you can afford to have fail if something goes horribly pear-shaped. You needn't necessarily have exactly the hardware it's going to run on—again, there are very good, functional and free virtual environments, such as VMWare or VirtualBox.
- **Development Process Documentation**. How a software project is proposed and evaluated; how it moves from approval to development status. What stages it goes through in the cycle of design, implementation, testing and acceptance. If you're going to use a standardized methodology (e.g., classic waterfall, Agile, etc.), document how fully

you adhere to the methodology, and where you've deliberately decided to deviate.

If you use a custom methodology, it must be defined and documented. Also consider if your in-house methodology is suitable for an off-site development effort; for instance, most Agile approaches emphasize close communication with the client and developer—this may not be possible.

- **Development Documentation Requirements**. What is necessary to define the business requirements for software; functional specifications (function without consideration of the implementation language or platform); and technical design specifications (the nitty-gritty of what language, libraries, etc. are to be used). Define testing requirements, acceptance criteria, and modification proposal, evaluation, and acceptance.
- **Support and Production Documentation Requirements**. What must be documented, and how to actually install, configure, and use the software as an administrator; required end user documentation content and structure.

5.2 EXISTING ORGANIZATIONS

The major advantage of an existing development environment is that you presumably have a functional development process and environment. The major disadvantage of an existing environment is, usually, that it's not documented fully. Before you can expect someone else to design and develop software for you, you have to have enough information to provide to them to allow them to meet your requirements and expectations.

Before ever contacting someone for an outsourcing effort, you need to review and evaluate your existing environment. Use all of the categories enumerated in the previous section (5.1) to evaluate your existing documentation, procedures and practices. If any area is inadequate, make sure it's corrected before you try to bring someone on-board to work with you.

6. DEFINITION AND DOCUMENTATION OF THE ENVIRONMENT

You will need, in either case, to carry out a definition and documentation phase of your new or existing environment. In the best case, you have a fully documented and functional environment, with everything up-to-date. Need I say how exceedingly rare that situation is in practice?

Make sure, if you do this in-house, that whoever is given this thankless task has enough time committed to the effort to actually accomplish the task. It can't be a secondary or tertiary responsibility.

This is also the time to remediate any deficiencies. Make sure the authority exists to actually implement changes; this requires "buy-in" from management.

This process may, in itself, be the subject of a contract; you can bring in a consultant or contractor to help you clean up and document your existing environment and process. However, this should be someone with enough experience in project and software development management to effectively dig into your existing documentation, or produce new documentation. And you *will* be spending time with this person or team—they'll almost certainly have to interview management and developers to pin down information that is missing or inadequate in the current documentation. If you don't get annoyed at how often this person is bothering you during the research phase of the definition/documentation project, they're probably not really doing the job you need.

And in the same manner as if done by an in-house resource, they must have the authority to effect recommended changes, and the support—"buy-in"—from management to get the job done. Otherwise, it's just an expensive way to dispose of funds allocated to your budget.

7. IDENTIFICATION OF CANDIDATE DEVELOPERS

Once you're certain you can actually tell someone how to develop software and documentation that will meet your needs, and in a manner that allows you to determine that it does, you actually need to find someone to do the job.

Be picky; this person or group is going to be taking your money, and the result of their effort will presumably be something you're going to have to live with for a long time. This doesn't mean you have to be ridiculous in either the specificity of your requirements, or the level of expertise you demand—if the level of complexity only requires a mid-level developer, that's all to the good (provided your Project Manager can do his or her job and actually manage the developer!)

You've a plethora of sources to tap for this—individual consultants or developers; consulting firms (providing a range from "body shops" to true consultants); former employees (who, it should go without saying, left on good terms) who know how to work with your environment; and so on.

In all cases, be sure to determine if they have the skill set(s) needed to work with the tools and business requirements of your project.

When dealing with individuals or small groups, interview as if you were going to hire them. If you don't know them personally, ask for references—*and follow up on those references*. Make sure your Project Manager is closely involved in the interview and winnowing process—he or she presumably has the experience to recognize good candidates and evaluate them, and will have to work closely with them during the project.

If dealing with consulting firms, ask for references from current and former clients. Again, actually contact these references. Moreover, if experience is claimed in your area of business and/or development environment, make sure it's with staff who are still with the firm; too often these days, firms are hiring off the street to meet contracts and there's no long-term staff or expertise in the firm.

In addition, make sure they identify the individuals who are going to be assigned to your project *by name*, and "accept no substitutes"—bait-and-switch is, unfortunately, not uncommon in the industry. I know from personal experience that there are consultants whose primary purpose in the firm is to interview and help get the contract—but when the deal is signed, move on to the next interview while another staffer is assigned to the job.

8. CLOSING THE DEAL

When you finally identify a candidate and reach an agreement in principle, you have to nail down the actual agreement. Make sure you have a Non-Disclosure Agreement (NDA) acceptable to the Contractor—or, if they provide one, make sure it's acceptable to *you*. Protect intellectual property, and make certain it's spelled out what happens to software, hardware, and documentation provided to the Contractor during and after the project.

Make sure the project is developed to your ownership and licensing requirements—e.g., will it be a Work For Hire that you totally own at the end of the effort? Does the Contractor want to retain any ownership of the work product—and is that acceptable to you? Find out if the Contractor intends to use any proprietary or licensed components in the project. You really don't want to find

out, at the end, that he/she has used proprietary libraries or components that don't belong to you, or aren't licensed to you. Require that any software not explicitly developed by the Contractor be fully documented as to its provenance and legal usability.

There's nothing wrong with using licensed components—as long as you're aware of the licensing terms and requirements, and they're acceptable to you.

Open source software—commonly called FOSS, or Free Open Source Software--is prevalent, and can be an excellent way to avoid costly development. But, again, make sure of the licensing restrictions and agreements attendant with its use—is it released under the GPL, LPGL, Creative License, Berkeley License, etc., and will your use be acceptable under the applicable terms? Require the Contractor to do the legwork on this, and your Project Manager to validate and verify. The Project Manager especially must make the effort to be aware of the differences between the different licenses—drawing on the expertise of your lawyer or legal staff as needed. It's also important to keep up on trade news—there have been legal cases regarding challenges to the open-source status of some software packages and systems. This doesn't necessarily mean you can't or shouldn't use such software—many, if not most, of these lawsuits have been clear cases of attempted intimidation, and to date have generally failed—but that's a decision that must be consciously made.

A note here—you should never, ever let your lawyer or legal department "drive" decisions--they'll park the car and lock the garage door to avoid any and all risk. We're not blaming them, that's their job—but they can't make business/technical decisions for you. They can and should advise you what risks you may be taking, and work to mitigate them under direction.

Decide if the Contractor is to be a W2 or 1099 employee, if it's not a company. There are advantages to you if they're 1099 or incorporated—you don't have to deal with taxes and unemployment, for one thing.

Make sure the contract covers payment, deliverable requirements, and acceptance criteria. It must define conflict resolution—when is it acceptable to withhold payment, how do you resolve differences of opinion as to acceptability of a deliverable milestone or failure to deliver in a timely manner, and ultimately what is the procedure for either you or the Contractor to disengage from the relationship. Make sure it's clear to both parties—"good contracts make good business partners."

"Straight T&M" (Time & Materials) is the Holy Grail to Contractors/consultants, and the bane of Clients. Similarly, "Fixed Bid" is the Holy Grail of Clients, and the bane of the Contractor/consultant. Most successful contracts are a compromise, based on T&M tied to deliverable milestones and "not-to-exceed" agreements. This doesn't mean the Contractor never gets more than the not-to-exceed amount; just that if that amount is approached, it's time to evaluate *why* things are close to or over the edge, and what can be done to contain the problem.

Another concept is in order here—"Individual Accountability". Essentially, *someone* must "own" and be responsible for any decision or action—if a deliverable milestone is signed off as accepted, who accepted it? If the Contractor is supposed to return resources, did someone from the Client actually accept it?

9. REQUIREMENTS, PROJECT PLAN AND MILESTONES

Once you've got all the legal agreements and contracts in place, it's time for the Project Manager to earn his/her keep.

9.1 REQUIREMENTS

No matter what methodology you use (except maybe Extreme Programming—of which I am *not* a proponent), you need the following general classes of requirements:

- *Business Requirements*. This defines just exactly *what* the software/system must do. It should describe function and general form, but not what technology is used to accomplish this, unless it's absolutely critical to the business function. As an example, you don't specify development language, execution platform, or which database should be used. However, if it's important that the user interface be a web browser, that can be stated at this level. Essentially, "What does it do; how do they use it; what goes into it; and what comes out of it."

  This is also the point that you must address any outside requirements—for instance, if you must comply with Federal or other regulations (HIPAA, FERPA, Red Flag rules, etc.) make that explicit in the business requirements. In fact, if you *don't* have any such compliance requirements, state that explicitly, too.

- *Functional Requirements*. Still avoiding the specificity of development tools, hardware, etc., this defines in detail all operational aspects of the project, such as display/user interface requirements and operational flow; the existence of any persistent data stores (database, flat files, etc.); graphic display design from the storyboard level. Effectively, if complete, from this set of requirements you would know just how the work product looks, acts, and what it does, down to the specific controls, buttons, etc.; and whether there is a database, and if so, what it stores. You wouldn't know what database is used (e.g., PostGreSQL, MySQL, or SQL Server), or if the project is written in HTML5, C++, etc.

  Architectural details are only defined at this level if they're of specific importance to the function—e.g., is it acceptable for a database to reside on the webserver, or must all data be passed back to a secure database?

- *Detailed Design Requirements*. Now you get down to development language, database package, support libraries, etc. Generally accepted industry practices need not be documented—e.g., I don't need to know how you intend to use PHP to open and retrieve data from that MySQL database—unless there's an aspect specific to this project.

All of these will, of course, not be complete at the beginning of the project, save the Business Requirements; they will evolve over the course of development. But everything must be quantifiable—you can't know when you're done if you don't know where you're going.

It's OK to change requirements—but determine beforehand just how much change is acceptable, and what costs in terms of time, money, and development effort are acceptable. Include these in the contract so the criteria and costs are clear to all parties. Again, managing this is the job of the Project Manager. "Scope Creep"—adding new requirements and features during development—has killed or, at best, ruined the budget for many a project. Evaluate all change requests carefully—if they're significant, you'd be better off to document and save them for a following release.

9.2 MILESTONES AND THE PROJECT PLAN

The Project Manager should have a project plan early on—effectively, a "10,000-foot-view" should be available as soon as he/she knows what the project is supposed to accomplish. By the time development actually starts, there should be clear milestones—significant deliverables—with

required and testable operational features. It should go without saying that payment for development should be tied to such milestones, as mentioned before. Don't make milestones too far apart, either—you want clear and continuous evidence of progress.

Another concern that lies squarely with the Project Manager: Delivery of all project materials. It's an unfortunate fact of life that not all Contractors are going to make it to the end of the contract; some may not even let you know they're not hanging around. The first indication you get is when they don't show up and can't be contacted. It is critical that at least at milestones, every element of the project-to-date be delivered to the Client—source code, documentation, the works. At least, if the Contractor disappears, you've got everything needed to pick up from the last milestone. It's even better if you can host the version control database for the project, and require the Contractor to check code and documentation into it on an ongoing basis.

The Project Manager has yet another critical role—guaranteeing that both the Client and the Contractor actually understand and agree on what the requirements mean. This sounds simple—believe me, it's not. Such misunderstandings may make for entertaining sitcoms on TV, but they only make life unpleasant in a development project.

Milestones need testing for acceptance; in this manner, by project end you'll have a mature set of regression tests and procedures in place for maintenance and future development. It's OK to have the Contractor do the testing, but make sure someone from your organization is actively involved—ideally, all through the process; at the least, in the final acceptance test for each milestone and the final product.

10. ENDING THE PROJECT

Hopefully, after final acceptance of the project, the Client and Contractor are still best of friends and business partners, and will go on to future work together. However, even in this case—and especially if not—when the project is done and accepted it's time to wrap up. All of this, of course, should be spelled out in the contract—and (as usual) the Project Manager is responsible for making sure everything is carried out according to the contract.

Account for all deliverables. Make sure that any resources in the possession of the Contractor that are supposed to be returned actually are, including but not limited to proprietary documentation, software, and hardware. Have signoff documentation that all of this was actually accomplished, and make sure that it's actually signed by the owner of that responsibility.

11. CONCLUSION

Much of the material discussed in this document is "common sense"—although we could wish that it was actually common. The goal is "no surprises"; and, in practice, most of this apparently smothering welter of requirements and checks can go surprisingly smoothly with the right mix of experience and skills.

12. CONTRIBUTIONS

I've received reviews and suggestions from numerous colleagues; I'd like to thank Michael Ciagala, Clif Flynt, Richard Jensen, Janet Plato, and Jenn Ridley, all of whom made suggestions that have been incorporated into the document.