Key Components in Custom Software Agreements

Prevent misunderstandings by covering your bases

By:
Avonelle Lovhaug
Avonelle@CodePoetrySoftware.com
www.CodePoetrySoftware.com

Last revised on: 6/26/2010

code poetry

custom software for your business

astute

If you've never been involved in a custom software development project, then hammering out an agreement with a software development vendor can be challenging. There are some pitfalls specific to software development that you'll want to be aware of, or else the results may go awry.

A good software vendor will know most of these pitfalls and will come to the bargaining table with these things in mind. Still, they'll do it with their own best interests, and you shouldn't rely on them to remember everything.

Disclaimer: I'm not a lawyer, and this document should not be considered legal advice. It is merely an attempt to describe items that should probably be covered in software agreements that may not be covered in other agreements. It is not an exhaustive list.

Here are some items that belong in most custom software agreements:

1 **Project objectives and / or requirements.** The best project objectives focus on the business value that the project will provide, not the implementation details. Here are two example objectives:

> Mediocre:
> *Create a quotation system for our sales reps.*
>
> Better:
> *Reduce data entry and calculation mistakes in the quotation process that cost us sales and money.*

The second example is much better, because it focuses us on *why* we are building the application.

If possible, it is also good to include any requirements that you know of. Requirements can be functionality specific, such as "must capture property name/address information". However, there can be a lot more to requirements than just the data you need to store. Here are some other requirement areas you may not have considered:

- ✓ Maximum # of total users
- ✓ Maximum # of simultaneous users
- ✓ Typical transaction load (how many records do you think will be created/updated every day or week or some other time factor)
- ✓ Maximum transaction load
- ✓ Security
  - Will users need a user name/password?
  - Will some of the data captured require special security handling (like social security numbers or credit card numbers)? A good developer will help you to identify these things, but it doesn't hurt to give this some thought.
- ✓ Response time (for example, a web application response requirement might be: typical response time for end users should be no greater than 10 seconds for users with a DSL connection)
- ✓ For web applications, the browsers types/versions that will be supported.

- ✓ The operating systems (and versions) that will be supported.
- ✓ Installation process needs (for non web applications). For example, you might require that a previous version of the system must be automatically uninstalled when the new version is installed.

Many of these numbers don't need to be exact. For example, for a web-based application it probably doesn't matter whether you will allow 50 simultaneous users or 55. But there is a big difference between 50 and 50,000. And some system tools like databases may license on a per user basis.

2 **Assumptions.** Assumptions differ from requirements in that they typically cover things that are NOT required or included. Here are some assumption examples:

- This application will not attempt to calculate the tax rates.
- Database client license fees are outside the scope of this agreement, and will be purchased separately by the client.

Assumptions are useful because they can help to make sure that both you and the developer are on the same page. Assumptions can also help keep costs down by helping the developer to understand the items that don't need to be included in the price.

3 **Tentative delivery date(s).** If there are critical dates for delivery, you'll want to include those in your agreement. But keep in mind that a lot of things can happen during the course of a development project, and so there may be legitimate reasons why you and the developer will mutually agree that a delivery date can slip.

4 **Communication guidelines and expectations.** It is important to determine when you expect the developer to respond to questions and requests. Typically, two business days is a reasonable response time but you may have different expectations. Also, you should agree to a response time as well. Early on, the developer will be very dependent on the information you and your team can provide to them. If you don't respond in a timely fashion, you can bottleneck the entire project. Also, you may want to specify the desired communication approach. I prefer to use email with my clients for most day-to-day communications, but everyone is different.

Issue Tracking System: Software that allows users to enter application problem information, and then track those problems until they are resolved.

5 **Feedback criteria.** Nothing is perfect, and the initial delivery of your software won't be, either. As you discover bugs and other issues, you will want to identify the problems you have encountered to the developer. This information can be communicated via an email, a document or spreadsheet, or through the use of an issue tracking system. Personally, I prefer to use an issue tracking system, as other methods can become unwieldy over time to track progress. In addition, a good issue tracking system will help you to determine the version that fixes/changes were applied in. Also, your agreement should spell out how quickly the developer can expect feedback, and how quickly

they will be expected to provide changes/corrections. Otherwise, the developer may move on to other projects.

6 **Acceptance criteria definition.** One of the challenges with any project is determining when the project is "done". One approach is to decide that "done" is when the project is being used by its target user group for "real", production work. That will work for internal applications, but what about external applications where the developer may not control if users are taking advantage of the new web application. In this case, you may determine some other criteria, such as X days after delivery.

7 **Cost and payment criteria.** As a business person, you need to know how much your project will cost. On the other hand, many developers are skittish about providing software development on a fixed fee basis, because project scope seems to grow over time. There are several reasons:

- ✓ Users change their minds about features after they are implemented.
- ✓ Requirements are often missed.
- ✓ Requirements are often misunderstood.

Generally speaking, the larger the project, the more likely the scope will change. Here are my suggestions for mitigating risk so that developers are willing to fix bid your project:

- ✓ *Start small.* If you have multiple projects in mind, try starting with one that is smaller in scope. You'll get a good introduction to how the process works, and the developer will get an opportunity to understand you better.
- ✓ *Chunk it.* Break your project into smaller deliverables instead of one big deliverable at the end. Then spend the time to review each piece as it is completed. The sooner you discover disconnects between what you meant vs. what they heard, the quicker they can adjust the other pieces to your needs. That minimizes the risk to them and also gives you peace of mind about the state of your project. A good delivery frequency is every two to four weeks.
- ✓ *Create a separate contingency budget.* Set aside some additional funds for unknown requirements that crop up during development. This is less necessary on smaller efforts, but as projects become larger this is smart planning.

7 Work product ownership. Typically the party who is paying for the software owns the resulting work product. However, you will want to spell this out in your contract – otherwise it is possible that the developer could turn around and sell the work you paid for to your competition. Yikes! Also, there are some muddy aspects of this that you will want to consider. For example, developers often use libraries of common routines that use for multiple customers. This allows them to develop applications faster and in a more consistent fashion. Does this mean that they won't be able to reuse those routines, and will have to start from scratch? This might affect their bid. Also, sometimes developers will use libraries or components from third parties – you'll need to make sure that those they plan on using don't have licensing agreements that will interfere with your own plans.

Keep these items in mind as you develop your custom software agreement, and you'll increase the chances of having a successful project.