**Title:** 8 Questions to ask about 3rd Party Software Components

**Key messages (this is not part of the blog):** using 3rd party software components from an established vendor helps decrease time-to-market, but open source components are scary

Without the reuse of blocks of existing design components, or “IP” (Intellectual Property), we would constantly be reinventing the wheel. On each new design project we’d have to start from scratch and design the entire system from the ground up. Using existing components lets us build added value with new functionality, while re-using the components that we have designed before.

Every design field does this, whether it’s a mechanical design for a new engine that reuses an existing turbine assembly, or an electrical hardware design that reuses an existing USB interface, or a software design that reuses a module that handles encryption. Productivity is enhanced, designs are advanced, and everyone wins. But there are a number of important factors to take into account when consider the use of someone else’s design component.

**1) Will it actually help?**

The first consideration you will probably want to explore is whether or not any particular function that you are looking for is something that you can actually use to differentiate your product. This might mean creating that component yourself and adding in features that no one else has, or it might just mean using the best available 3rd party component you can find. In some cases it might not be possible to create a better component than what is currently available, and in others it might not be worth the effort because that component is not part of your core functionality. Think carefully about what parts of your product makes sense to design yourself, and what parts are best outsourced.

**2) Where does it come from?**

The next consideration is the source of any 3rd party components. In the software world, probably more than any other design discipline, there are a lot of options available. In addition to actual open source, there are many alternatives that are provided free of charge, even if the source code is not actually available. Free is obviously good, and this fills many project’s requirements. However, not all the components you might want will be available for free. Even if they are, there are several reasons why you might still want to pay for them.

One key aspect to consider when looking at open source components is what organization is behind it? A component driven by a handful of individual users may be very different than something driven by a company (like Angular and Google) or well establish development project (like jQuery and the JS Foundation).

**3) Does the license meet your needs?**

Once you have decided to use an existing component, one key aspect to consider is exactly what type of license it comes with. There are licenses like the Apache 2 license, used by the Android operating system, that allows you to use a component in your product and still charge money for your product. The “MIT License”, used by jQuery and Ruby on Rails, is similar but requires a copyright notice to be included in the end product. The GNU General Public License (GPL) is based on the idea of a “copyleft”, which means that all derivative work must also adhere to the same licensing rules.

There are licenses that allow for educational use, personal use, non-commercial use… and even with regular paid licenses there are issued of royalty and non-royalty licensing (do you need to pay a royalty on each copy of your product sold). Another important part of the licensing discussion is the existence of any patents.

The point is that you have to be very careful when you look at any component source to make sure that the license lines up with your intended use. You’d be surprised how many times this gets overlooked, especially in start-ups and smaller companies.

**4) How much risk can you afford?**

Another key consideration is the level of quality you require for your application. Obviously we would all like perfect quality in everything, but quality usually comes with a cost, and higher levels of quality are not always required. If you are working on a classroom project, or a web site for your son’s Boy Scout troop, you can afford to take on a little risk. If there is a problem with your application in situations like those, the impact is minor.

If, however, you are creating an application that people will use to access their financial information or control a nuclear power plant, you will have very little tolerance for risk and will need to make sure the components have high quality. This is one area where software has an upper hand on physical designs like integrated circuits or mechanical parts. Software designs can usually be updated easily and quickly if needed, although significant damage can still occur before the bug is found and fixed.

**5) How do you judge the quality?**

Unfortunately there is no magic way to determine quality directly, so you will want to examine parameters like how long the providing organization has been in operation, how many other users are there, how much testing do they do, and what other users’ perception is. If you purchase a component from a company, then you have someone you can directly hold accountable for any quality issues, which you usually can’t do with open source components.

**6) Is there support?**

The follow-on issue to quality is support. Support ranges from having someone to fix any problems that might arise to helping you use the components properly. Even if the component itself is perfect, it usually has to interface to a variety of other code. Those modules might change either in functionality or even in their interface, and then you might need to quickly fix your code to work with the new environment. Most of us have had experiences in the past where we have received an OS patch and suddenly had one or more programs, that worked fine until that point, develop bugs or stop working altogether. It is usually incumbent on the application provider, not the OS vendor, to make the fix.

**7) What format do you need?**

You will want to consider whether or not you have access to the original source code. With most interpreted languages (primarily client-side web languages like HTML and JavaScript) you can obfuscate the code but not actually compile it. Most other languages (C++, Java, etc.) can be compiled to protect the source code. If you have high confidence in the provider of the component, source code is neither required or even wanted in most cases. If you are using open source components, however, you might end up having to patch the component yourself and access to the source code can be important. A compromise situation is to have escrowed versions of the source code, but this extra overhead is usually more applicable to full applications and not individual components.

**8) Does its roadmap match yours?**

Finally, is your application a one-time creation, or will it grow with you? If it is going to be an ongoing product for your company or organization, you will probably want the source of your components to update and enhance their components as well. How much control will you have over their roadmap? How much insight? Do they even have a roadmap? Open source components are often enhanced on an ongoing basis, but with less certainty. Often, you can make sure that a new feature is added to an open source component but you will have to be the one to develop it.

These are all just a few of the considerations you need to think about when using design components from a third party. Making the right choice can enhance your product, increase your product quality, and shorten your time-to-market. Making the wrong choice can decrease sales or even involve you in a lawsuit.