FivePaths

Customizing Software and Services

Navigating the slippery slope of custom software design

By Eric Leland & Jason Salter
FivePaths, LLC

Summary

Custom software configurations and programming can be a significant cost component in implementing many projects, leading to long-term support problems if undertaken incorrectly. Here, we highlight key issues and suggest a straightforward approach when tailoring software to your needs.

The Landscape: Configuring and Customizing Software

Before they can be used at your organization, most feature-rich software and software services require careful setup, and may benefit from some (to substantial) customization. The extent of this customization will depend on a combination of your project requirements, the product's features, and your organization's capacity to plan, implement, and sustain your project.

The Benefits to Customizing Software

When looking at various software options, organizations may find that less configurable packages include features that address many project goals but do not thoroughly satisfy all of their requirements. CRM software packages, for example, often advertise "custom fields" that allow users to track constituent data beyond out-of-the-box functionality. However, this feature is often of limited scope, offering an insufficient number of fields, restrictions on the type of fields or field placement, as well as constraints on how fields relate to each other, and whether or how they can be used in reporting tools.

The advantage of more configurable software is that it allows the product to more easily adapt to project requirements. For users and administrators, highly configurable software provides more opportunities to customize the way data is stored, displayed, imported, and exported. Customization can also allow users to more easily generate visual reports and infographics (as opposed to simply code). For developers, configurable software provides comprehensive access to controlling stored data using custom code, as well as (ideally) a strong, logical, and consistent framework that can be easily adopted by other developers. This combination of configurability can help ensure that the software or service will meet your exact specifications, as well as a smoother adoption process.

The Downsides to Customizing Software

Customizations, being highly technical and specialized, require skilled technical support that is often difficult to find and costly to retain. In addition, the final product can require a high level of skill to operate. While most CRM systems offer methods for importing constituent information, for example, these methods demand varying levels of expertise. Some systems offer graphical, step-by-step tutorial wizards, others allow Excel-file uploads, while still others demand complicated API (programming interface) access requiring advanced knowledge.

Moreover, the greater the level of customization, the higher the risk of support, update, and migration problems down the road. The less supported a feature is, the more likely it will be ignored or unsupported by developers in the future. One (more skeptical) way to think about customizations is that they are features with a user base of one.

Software and Users: The Negotiation

While it can be tempting to customize your software or services to meet a particular problem, this can ultimately lead to inefficient solutions and unnecessary costs. For this reason, it is advisable to use tools as-is as much as possible, and customize only for the most critical needs. (Note that if you have to customize a whole lot, it could be a sign that the software is not a good match, in which case you may be inviting substantial cost and maintenance issues.)

When deciding whether or not to opt for a custom software solution, look closely at your user workflows and the data your organization tracks; identify those elements absolutely critical for success. Next, carefully examine the framework and rules of the software packages you are considering; adhering to these built-in guidelines will be critical in ensuring that the package you select serves your organization effectively. Taking the time to weigh these factors will help you better pinpoint smart trade-offs that capitalize on the strengths of your software and the workflows of your users, and help you decide when customization is truly warranted. Those elements not critical for success may not demand any changes, or modifications may be postponed to a future phase.

Software Customization: Important Elements

Projects involving customization should pay special attention to definition and detail to ensure success. Elements to remember during software development include:

✔ Explaining What and Why

In talking to developers, focus on what feature should be built, and why these are important from the organization’s business perspective. Leaving out the why can lead developers to make incorrect assumptions about how features should work, resulting in costly overruns. Take care to clarify any informal or undocumented processes that might affect the development of the system, and clearly explain these to the developer.

✔ Participating in the Process

Organizations should review the work of the developer at regular intervals to ensure satisfactory outcomes. This is also true for features in development, even if other parts of the project are not yet ready. Even features defined precisely on paper can surprise reviewers on implementation. You can help avoid small mistakes from ballooning into costly budget items or missed timelines by conducting feedback rounds early and often.

✔ Tracking Accomplishments

Ask the vendor or consultant to document the features they have worked on, and what they accomplished in each case. This helps organizations understand the costs of specific features, and what it will cost to continue working on them or build similar features down the road. This also helps confirm that the vendor or consultant is on schedule.

✔ Clarifying Change Requests and Bugs

Be sure to identify issues that appear to be "bugs" — something broken the vendor or consultant should fix within the budget — as well as "changes" required that may cost the organization extra. Agree on a framework for addressing bugs and changes before proceeding to work on these.

The Algorithm: Bending, Configuring, Customizing

In both implementation and support, custom development is generally a time-consuming, costly process. When a requirement is not met with the core software or service, your options should be explored in this order:

1. Explore a "bending" (i.e. reexamining and redefining) of the requirement, which generally saves a lot of time and money, such that no customization is required.
2. If the requirement cannot be changed to match the core software or service, then attempt to customize through supplied wizards, settings screens, and utilities. These are built to provide a safer and more time effective approach to changing the system and are generally more future-proof.
3. Lastly, if a key goals cannot be satisfied through vendor provided widgets, custom coding can be used to accomplish your goals.

Wherever possible, work with your project team to meet your project goals without additional customization, using standard tools provided by the system or service, and as a final resort build your own custom solution. While this advice may cause you to go back to the drawing board a few times, the final outcome of your project will be stronger and significantly easier to support in the future.