



BY CHRISTIAN R. BURGER & ANGUS H. FROST

The Perils of the Contractor as Software Developer

Introduction

Just as many misunderstand the intricacies of construction financial management, contractors generally underestimate the complexities of software development. While we do not want to discourage anyone from choosing to build in-house software, it's important to know that software development often requires substantial time and resources. Whether your goal is to develop a product for outside sale or simply to build an application for internal use, this article describes the challenges of software development.

Strategic Analysis: Why Should Your Company Develop Software?

Here are several observations about construction IT that could influence a contractor's decision to build an application:

- Many contractors struggle to maintain their current IT infrastructure.
- Some contractors have a history of building custom applications and have a "build vs. buy" culture.
- Occasionally, a staff member has a penchant for writing computer code and wants to build a "killer" application for the company. These individuals tend to be hobbyists rather than professional developers.
- Many contractors have partially or poorly implemented software that requires an organizational commitment to complete installation and promote company-wide use.
- While software developers appear to move slowly, keep in mind that they have to design, code, test, and document the next version of their product before its release. This takes time to do well.
- There are contractors who are on the leading edge and consider technology use a competitive advantage.

Only one of these factors provides sufficient reason to move beyond the "what if" stage. *Specifically, unless your company has an idea that will give it a competitive advantage, there is little reason to develop your own software.*

Application development is difficult, but it can be done. However, it can only be done well with proper resources and an adequate plan that begins with a thorough assessment of your company's application needs.

Needs Assessment: Does Your Company Need to Develop Software?

After your company determines that a new application will provide a competitive advantage, it must identify if: 1) it has a compelling need, 2) adequate resources, and 3) a user group that will actually benefit from the new software. The following questions will help you further refine your needs assessment:

- Does the business function that will use the new software add considerable value? For example, a Heavy & Highway contractor with large inventories of rock, gravel, and asphalt could lower overhead costs significantly with customized inventory management software.
- However, a residential HVAC contractor who orders parts for each job would probably not need a highly-customized solution – especially if the solution costs more than the short-term savings recovered from better inventory management.
- Does your company need the new software because its current system is inadequate? If you have to build around a weak system, it might be better to simply replace it. Chances are that you will soon do so anyway.
- Does your company implement and use systems well? If not, fix that problem first. There is little point to building new software that no one uses.
- Will everyone who performs certain functions be expected to use the new application?
- Will new software address a need identified throughout the company or is it solely a response to a need perceived by a single user or set of users? Unless the user community drives software functionality with development support from IT, the project is not likely to succeed.
- For what IT function (input, processing, or output) will the

continued on page 80

company develop this program? Only consider company-built software when fundamental processing software is not readily available or is too expensive for the number of users. With today's report writing and business intelligence tools, it's often unnecessary to build applications to run customized reports or inquiries.

- What resources are available to maintain and support the new software? Once started, application development never really ends. There are always user requests for updates and changes (and from vendor communities if your application integrates with a vendor application in some way). You must also provide on-going training and support for the new application.

The answers to these questions should clarify whether to move on to a feasibility study or not.

The Feasibility Study

Developing an Application for Market Release

If you are considering an existing application for market development or release, objectively assess whether or not the application has broad appeal.

Some companies build an application around a given process based on older conventions, antiquated systems, or a lack of adequate thinking about what the overall outcome really should be. In those cases, the company often discovers that few, if any, other companies have a similar process. "Unique" does not necessarily translate to valuable or marketable.

Even if an idea has merit, not all companies will want to use the application in the same way. To increase market appeal, the product should contain a degree of flexibility (for example, designing the application to use data from a variety of databases). This means additional design, programming, and testing time, but it's probably worth it.

Developing an Application for In-House Use

Before your company builds an application for internal use, scour the market for an off-the-shelf alternative. Make sure that the people you assign to do this do not have a strong desire to build an in-house application; otherwise, they may be easily convinced that there's no comparable product.

Can Your Company Actually Build Software?

If you decide to build an entire application, particularly one for large-scale use within your company or for outside release, the proper approach requires several stages:

Requirements – First and foremost, understand and document the project requirements. This should include several meetings with people from the potential user community, as well as others who may have valuable insight. In design, too many people can become frustrating and distracting, but not enough input and perspective can be almost as bad.

Planning – Once the requirements are complete (or nearly so), developers can begin to decide how to build the program. This is an important step before anyone touches a keyboard. This is where questions like scope, budget, staffing, interfaces, and development platforms should be considered objectively. At this stage, a realistic schedule and testing approach should also be outlined.

Design I – Once the planning finishes, developers can begin to organize the application functions, outputs, and data structure. Now such items as security, menus, reports, and user options can be considered.

Design II – This is when programmers begin organizing fields, files, and table structure. They also think about the application's posting routines and processing controls. This second design phase is far more detailed and rather time-consuming. If this seems like a lot of planning, it is. Contractors certainly know the costs of "designing as you build," and the same holds true for programming software.

Coding – When coding, you must decide how many developers should be involved. Many companies attempt to build an application with one or two people as the designers, coders, testers, trainers, and documenters. However, there are different skill sets and aptitudes for each of these functions, and a great designer is not necessarily a great coder.

Testing – Once the application is substantially complete, testing can begin. To find and eliminate unexpected errors, it's critical to test the entire application under a variety of conditions. Too often, individuals test software for a "perfect world" and do not anticipate user mistakes.

Testing can vary from discreet testing of one function to integrated testing of the entire application. If the application is large or will be used by many people simultaneously, it's important to test the application's performance under heavy use.

Documentation – This is a critical function for a couple of reasons. First, system documentation ensures that others have a source of knowledge about the original programmer's design and conventions. Without system documentation, a

continued on page 83



new developer or outside development company would have to read the code and table structures, a task comparable to reading a map without street names.

Second, documentation is required for end users. Because this documentation tends to be less technical and more process-oriented, it can be very helpful during training.

Piloting – Once the application has been thoroughly tested and the documentation is ready, it's time to train your end users, so discuss your rollout strategy next.

Time permitting, the company may decide to train a few users to "pilot" or try out the application first. Set a trial period, and then select users based on their patience and understanding of the process and application. Whenever possible, these people should use the application during their day-to-day duties, so they can document issues and communicate frequently with the development team.

Rollout – Once the piloting stage is complete, you are ready to introduce the application to the rest of the company, which can be done in phases or all at once.

Post-Rollout Training – Ask some of your pilot users to help with the post-rollout training. Software developers understand the code they have written and how the application is supposed to perform, but they do not always make the most patient tutors.

Also, if there are going to be many application users, consider creating a help-desk position. Users typically want a quick response to their questions, particularly when an application is new and may still have a few "undocumented features." The help-desk person can also train new users.

Post-Rollout Review – Once the application has been rolled out and has been in use for a pre-determined amount of time, invite session developers, testers, trainers, and end users to a post-rollout review. The workshop's goal is three-fold: 1) review and document issues that have arisen since the initial rollout, 2) identify issues that require immediate attention, and 3) decide which issues to address in the application's next version.

Keeping the application fresh and functional helps ensure adoption and productivity. So, after the application is stabilized and widely used, review issues reserved for the application's next version, consider any additional features and functions, and combine these ideas with input from the user community on how to improve the application. Now the development team

is ready to repeat most of the development process and release a new and improved version of the application.

Conclusion

Applications and support programs are developed in-house all the time. But, too many companies begin the process with poorly conceived plans, unrealistic expectations, and/or inadequate resources. Depending on the scale of the application, you may not require everyone previously identified or every stage of the process. However, a well-planned process based on these recommendations will help ensure success. **BP**

CHRISTIAN R. BURGER is Principal of Burger Consulting Group, a consulting firm in Chicago, IL dedicated to the information system needs of the construction industry.

For more than 20 years, Christian has worked with contractors on software selection, implementation management, IT strategy and planning, and evaluation. He is a frequent speaker at industry events and contributes articles on technology in the construction industry to many publications, including *CFMA Building Profits*.

As a member of CFMA's Chicago Chapter, Christian also participates on the national Heavy & Highway Newsletter Task Force.

Phone: 630-942-1875

E-Mail: crburger@burgerconsulting.com

Web Site: www.burgerconsulting.com

ANGUS H. FROST is a Senior IT Consultant for Burger Consulting Group; he concentrates on IT organizational and infrastructural planning and management. Angus also provides technical support on software selection and implementation projects.

Prior to joining the Burger Consulting Group, Angus was the Chief Technology Officer for management and technology consulting firm Bridger, a subsidiary of Wight and Company. Before joining Wight and Company, Angus was an Information Specialist with Spencer Stuart, a worldwide executive search firm.

Angus holds a BS in Information Management from Thames Valley University in the UK and a BA in Business and Finance from Kutztown University in Kutztown, PA and Brunel University in the UK.

Phone: 630-942-1875

E-Mail: ahfrost@burgerconsulting.com

Web Site: www.burgerconsulting.com