

Build It or Buy It ?

Deciding whether to purchase a tool or develop it in-house

▶▶ QUICK LOOK

- Dispelling the myths surrounding both approaches
- Weighing your options

by Elisabeth Hendrickson

You've discovered that you need a new tool to support your work. Perhaps you're looking to replace an existing defect tracking system, or you want to have a system for tracking test results. Maybe you're looking for a test automation harness. You might even be wrestling with creating a library to support part of the programming effort for your primary product.

In any of these cases, the inevitable question that arises is: "Do we build it, or do we buy it?"

This seemingly simple question is actually quite complex—and your answer may have deep and lasting implications. Whether you build or buy, you will ultimately invest a large amount of time, money, and resources in acquiring the tool. The build-buy decision you make today will continue to affect your organization for years to come: the tools that enable us to do our jobs shape the way we do them, and we don't replace them lightly.

If the general thinking in your company is that this tool will be a temporary solution, then you may feel a little less pressure. The plan may be to replace the tool in a year or so when the organization is larger and more mature, or when you have more time. In that scenario, you might think that the decision of whether to build or buy is easier, because the consequences of making a bad decision would be minimal. In my experience, however, no organization is eager to toss away everything that has been invested to date in favor of a new tool, even if that new tool is demonstrably *better* than the old one.

Think about it: Deep in your heart, you know that this tool won't be replaced any time soon. Such tools become the keystones of the organization, supporting much of the development effort, and are difficult to replace without serious ramifications. So don't make the decision casually, thinking you can inexpensively remake the decision if it doesn't work out as you had hoped.

Whether building or buying, the important thing is to make the right decision for the right *reasons*. After weighing your options, you want your choice to be a conscious decision—not a default course of action propelled by assumptions. Let's take a look at the buy and build approaches, and examine some of the more common myths surrounding them.

Debunking the "We Can Build That" Myths

It's often tempting to decide to "just throw something together quickly" in-house. I've worked with more than one defect tracking system that was the

legacy from just such thinking. In each case, the defect tracking system probably did meet the organization's bare minimum needs when it was initially created...but three years down the road, the tool was wholly inadequate. At that point, however, so much effort had been put into maintaining it that it seemed counter-productive to replace it—and therefore hard to justify purchasing an off-the-shelf tool. Further, the developers of the tool were firmly attached to their creation, and very resistant to abandoning it.

Often arguments for building are more a result of the *Not Invented Here* syndrome than any real analysis. Let me debunk some myths about building that I have heard at a variety of companies:

MYTH: Building is cheaper than buying.

FACT: Building is usually just as expensive, or more so, than buying. Beware false economy.

When calculating the cost of the tool, remember the hours that will be spent in creating it, maintaining it, documenting it, and supporting it. It's true that creation costs, in terms of out-of-pocket expenses, might well be less than the sticker price of a commercial tool—especially if someone is willing to work nights and weekends to create the tool.

On the other hand, maintenance costs are likely to be much higher. You might think that users would demand less from an internal tool, but in reality users often demand *more*. The tool developer is right there in the next cube, making it easy to demand bug fixes or enhancements. And users expect that developer to react quickly to their complaints. It isn't unusual for an organization to discover that they have to assign a person to maintain the tool at least half-time.

Keep in mind that some costs don't show up in budgets. There is a cost associated with someone working nights and weekends that may not translate easily to dollars. There is also a cost of opportunity: What else could the person building the tool do that might have more potential payback for the company? What additional capabilities might a commercial tool have and how might those capabilities benefit the business?

One company decided to save money by developing their defect tracking system in-house using a low-end desktop database. They spent about \$1,500 on the database software for all their users, and a team member developed the initial defect tracking system over a series of nights and weekends. The company then hired an outside consultant—and ultimately a full-time employee—to maintain and improve the system. In the end, they spent at least \$100,000 on custom programming and maintenance in an attempt to make the system meet their needs before replacing it with a commercial defect tracking system.

Many companies have experienced the same problem. If the manager of the department responsible for the tool had realized how much money would be spent on enhancements and maintenance of the tool, she probably would have decided to adopt a moderately priced commercial system (available at the time for around \$50,000).

Don't make the decision of whether to build or buy based solely on which option is least expensive. Comparing the benefits of each of your choices rather than the cost is more likely to result in a good decision. It is very difficult, if not impossible, to accurately estimate the real cost of build-

ing and maintaining a new tool.

MYTH: We don't need anything fancy, so it should be easy to build.

FACT: No tool is as easy to build as it appears at first glance—so if you think your tool will be a trivial building project, you probably don't understand how hard the problem is. Before you rate the difficulty of building the tool, spend significant time examining the requirements. You may discover that while most of the tool is straightforward, that last 10% is so difficult that it might never get implemented if built in-house.

A company I once worked with built a simple test tracking tool. It allowed testers to indicate test results easily enough, but did not support test *maintenance* very well at all—and after several releases the tests documented by the tool were obsolete. The next generation of tests was documented using a word processor because it was so difficult to use the test tracking tool for test design.

Simple tools may also not scale to larger groups or more complex processes. The simple test tracking tool this company built was more usable for a small group testing a relatively simple product. The testers could keep all the areas to test in their heads, so the tool didn't need to display a high-level view of the organization of the tests. As the product evolved to be more complex and more testers were added to make sure it was fully tested, the simple test tracking tool worked less and less well.

MYTH: We can build a better tool than we can find on the market.

FACT: Unless your company's or department's core competency is building tools, chances are you can't actually build it better than you can buy it. Remember that the companies that offer commercial tools have invested a tremendous amount of time and effort in developing their product. It is pure arrogance to think that your department can create a better tool in just a few weeks.

One company seeking to replace a faltering home-grown defect tracking tool considered having the development group build a Web-based tool to replace it. The development group spent time gathering the requirements for the tool and began designing it, confident that they could have a full solution available in a few weeks. As it became apparent that it would take much longer than anticipated to build the tool, the development group began scaling back on the features to be implemented. Eventually, the development group proposed implementing a tool that would have been significantly less functional than the *existing* home-grown solution.

The developers also suggested that because they were running short on time, the testers should take on maintenance of the new tool—and that the testers should also implement the remaining features themselves. Ultimately the test group decided to pursue commercial defect tracking tools, to avoid being stuck implementing the most important features in an unfamiliar development environment.

MYTH: We don't have a firm process in place yet. Building will give us the flexibility to change the process on the fly.

FACT: If you don't know what process the tool is

intended to support, acquiring a tool won't help. You're better off implementing the process as a paper-based system initially—so you can work the defects out of the process before writing them into the code. Once you know (at least to some level of detail) what your process is, you can revisit the decision of whether to build or buy a tool to support it.

So does this mean that you should never build a tool in-house? Certainly not. There are often very legitimate reasons to build tools instead of buying them. At the same time, “We should just buy something” is another trap with its *own* set of myths that need debunking.

Debunking the “Just Go Buy Something” Myths

MYTH: Buying a tool is easier than building one.

FACT: Whether you build or buy, introducing a new tool means introducing change. Most people naturally resist change. Just because you aren't programming the tool doesn't mean that rolling it out to the organization will be easier.

In addition, most commercial tools allow you to customize the tool to meet your process more closely. Someone will have to specify the customizations to be done and implement them. You might have someone on staff do the customizing, or you might hire a consultant from the tool vendor. In either case, customizing a tool requires planning and effort not unlike building a tool from scratch.

MYTH: Buying a tool means we can deploy it more quickly.

FACT: Although buying will save you time in designing and programming, other activities will take a significant amount of time. You still have to select a tool, set it up, migrate your data to it, deploy it to all the users, and train the staff to use it. Depending on the complexity of the tool you choose, how much customization is required, and the resistance to change in your organization, it may take just as long, if not longer, to deploy a commercial tool.

A company I knew decided to migrate their defect tracking system from a homegrown solution to a commercial tool. The homegrown solution had taken a month or so to implement and roll out to the organization some years earlier. It took *three* months to roll the commercial tool out to the whole organization once the tool was selected—and it had taken six months to select the tool! Ultimately it took significantly more time than expected to customize the tool, migrate the data, and get the organization to accept the new tool.

One of the stumbling blocks to full-scale implementation of the new tool was the fact that some members of the staff strongly preferred the old tool. Not because it was better—but because it had been created in-house and because they were familiar with it. This was a case where choosing a commercial tool actually slowed down its adoption in the organization; some users preferred having a tool created and maintained by their co-workers.

MYTH: If we buy something, we won't have to maintain it.

FACT: Whether you build or buy, you will need to expend some effort on maintenance and administration of the tool. And if you customize the tool, installing upgrades from the vendor may be

more complicated and may require you to redo some or all of your customizations.

After considering everything that can go wrong in building or buying a tool, you might decide that you don't really need that tool in the first place. You might be right—lots of tools that sound like really good ideas at the time become shelf-ware soon after their adoption, for a variety of reasons. Tools that are too cumbersome, that don't fit the organization's way of doing business, or that don't offer users a compelling reason for adopting them are very likely to fail.

First Steps: Understanding What the Tool Needs to Do

The key to successfully building *or* buying is understanding what exactly it is that you need the tool to accomplish—what the requirements are for the tool. If you work in an organization that is moving very quickly, you may think that you don't have time to write a requirements document for your tool. It takes a lot less time, however, to find out *up front* what people expect the tool to do; once you've deployed the tool, it's too late to discover that it meets only a small portion of the organization's needs.

You may be reluctant to spend time on a requirements document if you think the tool is too simple to necessitate one. After all, everyone knows what a defect tracking system is supposed to do, right? But you will probably be surprised to discover how many different definitions the other people in your organization have for this tool. Writing a requirements document is an excellent way to bring differing expectations to light early in the process.

To write the requirements document, bring the stakeholders for the tool together in a room with a whiteboard and discuss these basic questions:

- Who is going to use this tool, and for what purpose?
- What problem(s) will this tool help us solve?
- What kind of process does the tool need to support?
- Can the process change to fit the tool, or does the tool have to change to fit the process?
- What features will the tool need?
- What reporting capability does the tool need to have?
- Who should have access to the tool?
- Do you need the tool to have a concept of an administrator who has special permissions?
- What kind of user interface does the tool need—desktop application, Web-based, programmatic, command line?
- What kind of platform(s) will the tool need to run on?

- Will you need to integrate the tool with any other applications, such as email, a reporting package, or other tools already in use by your organization?
- How much is the organization willing to budget for acquisition and maintenance in terms of both time and money?

In the meeting, make sure the participants distinguish between *nice-to-have* features and absolute requirements. Note that one stakeholder's *nice-to-have* feature may be another stakeholder's *must-have*. For example, if you are discussing a test tracking system, Management needs to be able to get summary reports of the percentage of testing complete by area. This might be only a *nice-to-have* feature for a tester, but it's an absolute must for a manager.

Record the group's answers to these questions and circulate the resulting document. Nothing fancy or formal—this will serve as your tool requirements document. You'll want to have everyone review it before proceeding. To make sure that everyone is on the same page, schedule another meeting in which the group will review the requirements document in detail. This is an opportunity for each stakeholder to express concerns or raise additional requirements that didn't come out in the first meeting.

If a disagreement about the requirements surfaces in the meetings there may not be time to resolve it right away. Make sure to resolve it later. Before making a decision about building or buying, the group should achieve consensus about what the tool needs to do.

This is a quick, low-ceremony way of gathering requirements. If the stakeholders for the tool are willing to make the time to attend the meetings and review the resulting document, the entire process can be completed in less than a week.

Deciding Whether to Build or Buy

Now that you know what the tool is expected to do, you can decide whether to build it or buy it (see Table 1). The decision criteria for building versus buying involve asking yourself a series of questions as you weigh your options.

Are there commercial tools available? Before you make a final decision on whether to build the tool or buy it, you owe it to yourself to find out what's on the market. The availability of a commercial tool that meets your requirements is the ultimate determining factor in whether you should build or buy.

One company where I worked decided to automate their tests. Their application was GUI-intensive, and they wanted to exercise the application through the user interface. In this situation, the choice was easy: they decided to buy a GUI-based test automation tool rather than trying to add automation hooks into the application under test. Happily, there were several test automation tools on the market that would meet the organization's needs.

By contrast, another company had an application with a minimal user interface. Most of their product's functionality was hidden from the end user. The product

had support for command line flags and API's that made it possible to write test harnesses using batch scripts, *Visual Basic*, and *C++*. Although some of the testing could have been done with a commercial tool, most of the testing could be more easily accomplished by writing custom test harnesses. For this company, buying a test automation tool didn't make sense.

In evaluating commercial tools, be sure to do a hands-on test of the tool to see if it meets your needs. You cannot tell from the marketing literature, demo, or documentation alone whether the tool will perform as you expect. (See "Evaluating Tools" by this author in *STQE* Jan/Feb 1999, page 38, for more information.)

Can we get budget approval for the tool? Sometimes building a tool is the only option. Many years ago, in the early days of my career, I wrote a call tracking tool for a UNIX development organization. The tool I wrote had a number of features that the organization needed: automatic emailing of issues when assigned, reports that showed peak call times and problem resolution times, and automatic problem escalation for issues remaining open past the committed resolution date. At the time, the selection of commercial tools that would run on our platform of choice—and that had all the features required—was minimal.

More importantly, even if we had found the perfect commercial tool we would not have been able to acquire it; there was no budget allocated for a new call tracking tool. This was a case where it was significantly easier for Management to approve my spending time on the tool than it would have been to get budget money approved for it. Many organizations are in the same situation: It's difficult to get authorization to spend money, but it's easy to get authorization to spend time. Logic tells you that in both cases you're really spending money, but one shows up on a balance sheet and the other does not.

Do we gain a competitive or organizational advantage from building or buying? Another company produced client-server software designed to work with almost any database. The development organization created a library of function calls to work with Sybase, Oracle, and other databases. At the time there weren't any similar libraries on the market; but a few years later commercial libraries that would do the same thing became available.

By that time, however, the development group had developed significant expertise in programming for any database. The company promoted the product as having an "open architecture." Customers responded very favorably to the open architecture, making it a competitive advantage. Changing over to a third-party library would have forced the company to work within the restrictions of that library. Because the open architecture was a competitive advantage, building and continuing to maintain the proprietary library was the right decision—it gave the company complete control over one of their key competitive advantages. In the long run, adopting a third-party library might have been less expensive, but it also would have reduced Development's ability to innovate within the open architecture.

Do we stand to learn something important by building the tool that we cannot learn any other way? It may also make sense to build a tool if your organization needs to develop expertise in an area. For example, if your organization is about to venture into Web-based development for the first time, building a Web-based tool for internal use might be a good way to learn how to build Web-based applications. This is one way of “doing one to throw away” without having to throw away the results. If you choose to do this, just remember that the resulting tool may be a little clunky. And when the tool is all put together and the organization has learned its lessons, the organization will have to learn another important lesson: how to maintain it.

Other Options

Asking the question in terms of “build versus buy” implies that these are the only possible ways to acquire a new tool—but there *are* other options. You could hire another firm to build a custom tool for you, or you could use a free- or share-ware solution.

If you choose to outsource the building of a custom tool, remember that you will have to either continue to pay that organization to maintain the tool for you, or take over the maintenance of the tool yourself. When you are outsourcing development, it is particularly important that the development contract clearly specify:

- The requirements of the tool to be built
- Whether the outsourcing company is expected to provide documentation, training, technical support, and maintenance services

- Who owns the source code once the project is finished

When outsourcing development, you cannot assume that the outsourcing firm will provide any of these things as a matter of course.

If you choose to use a free- or share-ware tool, remember that you will probably have to provide your own training, support, and (at least to some degree) maintenance. In particular, if you choose to adapt an open-source tool to your needs, you will end up essentially building your tools—just with the advantage of having the first version pre-built.

Conclusion

The key decision criteria to apply when determining whether to build or buy are deceptively simple:

- Can we get the tool elsewhere?
- Do we gain significant organizational benefits from choosing one course over the other?
- How does the decision affect the organization's flexibility?

But building versus buying is a complex issue. The results of the decision will affect the organization for a long time to come. For that reason, it's a good idea to test the decision before you implement it. If you decide to build, ask yourself, “What value do we get from building that we don't get if we buy a pre-packaged tool?” Similarly, if you decide to buy a tool, ask, “What value do we get from buying this tool that we don't get if we build our own?”

Finally, the ultimate test is to ask yourself, “Five years from now, when we look back on this point in our history, which decision will we wish we had made?” Few people can claim that kind of clairvoyance, and you might not be able to completely answer the question—but just asking it will improve the likelihood of making the right decision. **STQE**

Elisabeth Hendrickson (ehendrickson@aveo.com) is the Director of Quality Engineering at Aveo Inc., and she has bought—and built—numerous tools. More of her articles are available at www.qualitytree.com.

Factors Favoring Building

- You need a tool custom-tailored to your organization's process or business practices.
- The organization will learn necessary lessons in the process of building and maintaining the tool.
- Management will not allocate a budget for the tool, but will permit time to be spent on development and maintenance.
- The tool is an aspect or simple extension of your organization's core competency.

Factors Favoring Buying

- There is a selection of tools on the market that will meet your organization's needs.
- The organization can adapt its process or business practices to the tool.
- The organization's needs will grow over time.
- The organization does not have the time or expertise to build the tool.

TABLE 1 Contrasting factors for building vs. buying