



# CBAT

## Codeless Business Application Technology

Arlen Feldman

Copyright © 2009 Cherwell Software, Inc.

### Contents

Overview.....	2
The Problem.....	3
Issues with custom-developed applications.....	3
Issues with generic tools.....	3
Issues with off-the-shelf products.....	3
The CBAT approach.....	4
CBAT tools.....	4
Characteristics of a good CBAT system.....	5
Conclusion.....	6

## Overview

CBAT (Codeless Business Application Technology) is an approach to software that allows managers and business domain experts to quickly customize applications to precisely meet business needs *without* also requiring software expertise or the hiring of expensive consultants.

This goes far beyond just providing some basic options within an existing application. A good CBAT application is layered. It starts with a core of business-appropriate functionality that can be customized in terms that are meaningful to someone familiar with the business domain. It should also allow for deeper, more specific customizations and integrations that can be accomplished by someone with some basic training.

That doesn't mean that there isn't a place for consultants or local development with CBAT applications. Ideally, consultants could be used for extremely advanced customizations, and APIs should be available for specialized integrations. However, these should not be the starting point for a CBAT application, and even if external resources are used for implementing advanced functionality, local resources should be able to make any minor changes and tweaks without having to call for help.

Advanced CBAT systems even allow for different business functions to be created and seamlessly integrated with other users' systems. If a business expert in one company understands a particular problem, they can customize the system to address that problem, and then provide that solution to other companies—all without having to become programmers or database gurus.

The ideas behind CBAT have been around for many years, but it is only recently that vendors have truly embraced the approaches that make CBAT systems effective. At a time when company resources are stretched, while being challenged to solve ever more complex business problems, CBAT may well be the only approach that can address the needs of the business without breaking the bank.

## The Problem

For many years, IT departments have been called upon to provide tools to manage all sorts of data and processes. This might be for entering simple timesheets, or for handling the entire company lifecycle for bidding, development, sales and marketing. Depending on budget, time and skill levels, the solution provided might be a custom developed application, an off-the-shelf product or even a lightly adapted generic tool, such as Excel. There are issues with each of these approaches.

### Issues with custom-developed applications

Building custom applications to meet business needs often seems like the simplest and most direct approach—particularly if there is in-house development talent, or consultants on contract. One reason for this is that in-house development typically does not directly add to budgets. However, there are many reasons to be wary about this route. For one thing, between 62% and 80% of IT projects fail in one way or another.<sup>1</sup>

Even for projects that do succeed, or partially succeed, the results often do not meet expectations. In-house developers rarely have the resources to add all of the various “bells & whistles” that are considered to be standard for modern software. They may also not be able to take into account scalability or security issues. And they may not have the time to revisit the tools to address bugs or changes to requirements.

### Issues with generic tools

Just about every office has someone who is an expert in Excel or MS Access, and who can whip up a quick spreadsheet or database to handle some data. While this is often (but not always) better than nothing, they cannot be seriously considered to be permanent solutions to business-critical functionality. Just using the tool requires specialized skills that are unlikely to be reliably available, and the solutions tend to be tricky to use and suffer some of the same issues as custom-developed applications, as far as usability and changes to requirements are concerned.

### Issues with off-the-shelf products

There are now applications available for just about every business application you can think of. These tools tend to be extremely expensive, and either lock you into a fairly rigid set of procedures that don't match up with your business needs, or require expensive consultants to configure the product. Even minor changes or upgrades also tend to require bringing back the consultants, or having a dedicated trained staff on budget.

---

<sup>1</sup> Original study, Standish Group, 1994. Dynamic Markets Limited, 2007 study showed similar results.

For this reason, many expensively purchased packages never get implemented, or get replaced by a package that addresses one particular new need, and then re-replaced when another new need presents itself.

Obviously, none of these solutions are tenable in the long run.

## The CBAT approach

CBAT has grown out of the limitations of all of these different approaches. Vendors of off-the-shelf products were already addressing the basic business needs, and the software issues such as scalability, security, and integration. They are now retrofitting (or completely re-architecting) their products to also provide support for codelessly changing application behavior to match the needs of their customers. Newer companies are simply building applications with the assumption that CBAT (under various names) is not an option.

While CBAT is technologically agnostic, the heart of most successful CBAT systems is *meta-data*. In a classic business system, first someone designs a database, then the programmers write code to access the data in that database, and build in the rules of the system. In a meta-data driven system, the system has a meta-base that defines what data needs to be stored and the rules of that data. The beauty of a meta-base is that it is generally much easier and quicker to update its content and, once changed, the system just starts reacting to the new structures and rules, as defined, without any code needing to be changed.

Meta-data driven systems have been around for a long time. In fact, many systems that appear to be hard-coded use meta-data behind the scenes to make it quicker for the developers to respond to changes. What makes CBAT's use of meta-data stand out are the tools for working with it, and the amount of meta-data—the more powerful the system, the more complex and detailed the meta-data that drives it.

## CBAT tools

If the only thing that differentiates a system that requires arcane knowledge of a programming language from one that requires arcane knowledge of the internal structures of the tool, you haven't gained much. In fact, you have lost, because there tend to be a lot of programmers familiar with a particular language, while experts in one particular system are a lot rarer and more expensive.

Tools can be broken down into two general categories—low-level and high-level. Low-level tools are the ones that would be more familiar to anyone who has done RAD (Rapid Application Development) or built simple databases. These would be for doing things like adding fields and

relationships, and designing forms. With CBAT, though, these tend to go much further, as they include the ability to define the rules of the system. High-level tools are more business-level.

The low-level tools for customizing a good CBAT system should be usable by any reasonably intelligent manager with an idea of what they want to accomplish and some basic training. In fact, simple changes should be possible without special training. The tools should guide the user, and warn them when there are unexpected side-effects to their changes (such as to performance).

Of course, more complex customization might require more training, or even the assistance of a consultant—but then again, they might not. It depends on the nature of the changes and the comfort level of the customizer. If a consultant *is* needed, then ideally he or she will be able to design the new functionality and simply send it to the customer to apply to the system.

The high-level tools for customizing CBAT systems are business-level tools. Whereas a low-level tool might ask whether you want to add a new field, and what you want to call it, a high-level tool will ask whether you want to implement a particular business function, and ask you a few expert-generated questions on how you want it to work. For example, if you want to add bill-back to your system, you might be prompted about the conditions where you would/would not bill-back, rates and minimums, etc. The system will then configure the underlying meta-data to provide the desired functionality. You might then use the low-level tools to tweak the implementation if required.

The important point about the high-level tools is that they represent expertise. This might come from the vendor or, as likely, from partners of the vendors or even other customers. A quality CBAT system will provide tools for encapsulating expertise in a way that can be transferred.

There are some other types of tools, including some that fall in between high and low-level, such as when dealing with integrating with other systems, and tools for providing an overview of how the system has been configured, but these are not really at the heart of CBAT.

### Characteristics of a good CBAT system

It is easy to get confused when shopping for systems. Many business applications these days advertise that they are 100% customizable (which is a mostly meaningless statement) and that they have one or more characteristics of CBAT. However, there are certain things that make a system a *good* CBAT system:

1. CBAT systems are business applications, not *toolkits*. You don't want to be given a pile of metal and some tools, when what you want is a car. You just want to be able to customize the car to your liking. Likewise, a CBAT system should address your basic business need out-of-the-box (or after applying some specific pre-packaged customizations). The tools

are to make the system fit your needs precisely, not for you to build something from scratch.

2. The tools should be *layered*. There should be high-level business tools, and lower-level tools, but even the low-level tools should be straightforward to use, at least to do straightforward things. In the software world, this is known as *progressive disclosure*—you are only exposed to as much complexity as you need.
3. Although one of the major features of CBAT is that it doesn't require *code* to do most things, there are always extremely specialized tasks that just can't be done any other way. A good CBAT system should have a thorough API for interacting with it. It should also have a lot of connection points for working with other systems—many of which should not require code (for example, for pulling data from another database system) but the option should be available.
4. It should be possible to wrap business functionality and transfer it between different implementations. This might be from a test to a live system, from a consultant to a customer, or even between customers. Ideally, it should also be possible to build in knowledge from a domain expert into the functionality so that the person applying the changes can be quizzed, and then get a more exact fit immediately.
5. Customizations *must* be upgradeable. This sounds mundane, but there are a number of systems available that allow complex customization, but then can't be upgraded to a new version without the customizations being rebuilt or manually upgraded by experts.
6. The pricing model of the software should not penalize the customer for using the tools. For example, if the customer creates an entirely new use for the product, the vendor shouldn't charge the customer for their creativity (although they can benefit from selling additional licenses as more users now use the solution).
7. Of course, the system must also meet the standard requirements for any other software package as far as stability, security, scalability, performance, integrations, availability, reporting, etc. are concerned, and it must address your business needs. The vendor should also meet requirements for support and availability of consultants when needed—and, critically, they must support the custom changes made by the customer—some vendors do not.

## Conclusion

Providing business software to your company can be an extremely frustrating activity. This has a lot to do with the nature of software—while hardware performance has been doubling roughly

every 18 months for the past 60 years, the techniques for building software have only slowly evolved. It is only recently that the process of building software has begun to get out of the way of the business people who understand the purpose of the software.

CBAT is a major step in that direction. It makes it possible for the software developers to focus on their area of expertise (building software) and the business people to focus on theirs. The vendors who are providing CBAT solutions are not simply solving problems for their customers—they are providing tools for their customers to solve their own problems, even those they have yet to encounter.