

Couldn't Wait For Code... So Went Ahead Without It

Why? Because it makes fiscal sense!

Code is a productivity thief. Once an organization introduces code it introduces risk into a project. There is no longer a need to take a risk with custom coding when an organization can *assemble* their software solution.

The Enterprise Content Management (ECM), Business Process Management (BPM) and Business Intelligence (BI) space has evolved to the point where it is now possible to deliver custom software solutions without introducing custom code at the application layer. In other words, organizations can get exactly what they need in a business solution without the compromise and risk typically associated with custom software solutions. And, this can be effectively and efficiently achieved while the organization remains fiscally responsible.

Unleash the Power in Your ECM, BPM and BI Products

Once an organization has done its due diligence, has selected the best ECM, BPM, BI and other infrastructure products, why should it add risk to the process by having programmers write custom code on top of those products? It shouldn't. Hard coding solutions on top of an organization's ECM, BPM and BI infrastructure can defeat the purpose of carefully choosing its products and strategy. In addition to the introduction of risk, you will only succeed in masking and marginalizing the underlying power of what you purchased.

Rather, the organization should turn to a core capability of these products – assembling and configuring. The organization can assemble within, on top of and around these products.

What Do We Know about Custom Code?

We all know that code is necessary. We also know that it must be eliminated or marginalized to reduce the risk and cost on any software development project. We also know that alarm bells go off as custom code becomes a more significant portion of the so called “drop-in”, “customizable”, or “packaged” ECM, BPM and BI solution. However, in many cases these alarm bells either come too late or are muffled by the sweet sounds of the marketing buzz.

When code arrives neatly packaged inside of a box or in a link and can be installed quickly to begin solving business problems, it seems that it is always a welcome guest to the party. Code is also useful and highly valuable if it arrives as a 100 % reusable component or service or can help organizations solve business problems when no other options are available. But, code does not play fair. It usually arrives to the party late and

is usually cleverly disguised as simple customization, a few tweaks, a small change order or even more inventively and destructively as a framework.

Using these schemes as a ruse, code can easily penetrate projects and become not only a way to derail productivity, but also a way to mask, marginalize and minimize the capabilities of the underlying ECM, BPM and BI infrastructure products.

Code commonly increases risk, rarely travels alone, and typically arrives with its travel companions – Cost and Time. Regardless of whether or not these travel companions arrive together or separately, they all lead to the same result: significantly decreased productivity. It is extremely difficult for organizations to achieve their business objectives of implementing business solutions faster and at a lower cost if they are relying on custom code to create end user business applications.

Effectively using assembly and productivity tools

Today, software productivity tools, products and technology are mature enough that organizations can assemble without code, virtually every piece of their end business solution (*i.e.*, the applications that sit on top of ECM, BPM, BI and other infrastructure products.) Organizations can now profit by using assembly and software productivity tools not just for their infrastructure, database, and reporting products – but all the way out to the UI and everything in between.

Most organizations are familiar with and use various software productivity tools that can assemble end-user functionality, such as SAP Business Objects Crystal Reports, IBM ECM FileNet Process Designer, Adobe LiveCycle, and ObjectBuilders LiveApp Player Suite. All of these products combine a simple player or engine strategy that leverages three key items:

- **Documents** - These control the Player. As an example; a PPT PowerPoint file controls what a PowerPoint viewer displays as a slide presentation. An SWF file controls what a Flash Player displays. LiveXML files control the details of the business application that is rendered by the rendering engine etc.
- **Assembly Tools** – Software productivity tools for assembling and maintaining the application and drag and drop configuration by business users, non-developers, and non-programmers
- **A Player or Rendering Engine** – Read the XML (or proprietary) documents and render the application to any deployment style, Rich Client, HTML/Ajax, Disconnected, PDA, etc.

This approach is not risky or new as the industry has been moving in this direction for years and, it makes sense from both a business and a technical perspective as it combines the following:

- Tried, tested, and true
 - Limited risk
- Open and flexible

- Easily support and move into the future
- Widely supported
 - Lower total cost of ownership

Where do frameworks fit in eliminating or reducing code?

When properly utilized, frameworks can provide great value if they can truly do what they claim. However, most frameworks typically:

- Are limited to their features, much like a packaged solution
- Are risky and proprietary
- Cannot expand horizontally or vertically without custom code
- Eventually lead to custom development
- Tie you to an approach or product

Therefore, unless the framework or packaged solution demonstrates *in a live demonstration* that it provides at least 90% to 95% of the capabilities needed straight out of the box and the ability to configure (without code) the remaining 5% to 10% of the required functionality, organizations will most likely need to add a significant amount of custom code on top of the framework to meet their overall needs. If you can't see what was promised in a *live demonstration*, why would you expect to see it in your solution? In other words, trust your eyes, not the words in the presentation.

Using an Assembly Strategy

Accessing, leveraging and harnessing what an organization knows and owns is a simple objective that can be accelerated by using an assembly strategy. Organizations can easily accomplish this objective by adding a layer of abstraction in the form of a virtual business object model (VBOM) and allowing their solutions to be loosely coupled to their back-end data sources and infrastructure products, and then assembling their solution on top of this layer instead of hard coding it. By doing this, organizations can move closer to their SOA initiatives and, at the same time, offer a level of future-proofing, vendor, product, and technology independence that hard coded applications do not provide. It will also help drive and promote reuse, which in turn reduces cost, time and risk.

Data as the Application

Dr. Charles Goldfarb, the Father of XML Technology, said that "The next big thing for XML is representing entire applications." And, since we all know that data is the lifeblood of decisions, that content is data and data is best represented as XML. XML is the most open, flexible and accessible standard available today, it makes sense for organizations to leverage existing standards. XML is doing for software applications and business solutions what it did for data. Why not move in that direction? If smarter data is the goal, and data is the application, how much smarter can the data be?

XML is quickly becoming the recognized standard for representing business applications, and is doing for business applications what it did for data. Examples for using XML to represent components, services or business solutions abound in virtually every

organization. Industry leaders like IBM, Microsoft, Oracle, ObjectBuilders, and many others have now standardized on XML.

Organically achieve Service Oriented Architecture (SOA) objectives

Code is not going away; it is just being put to better use. Code is most useful in creating new services and components that can be consumed into applications through assembly and reused in other solutions. This not only helps organizations *organically* achieve their SOA objectives, it reduces overall costs and helps their teams become more productive.

The fact is that if everything is represented in XML and bound at run-time into an “assemble once, deploy any style strategy”, hasn’t everything been turned into a service and hasn’t the organization *organically* achieved its SOA objectives?

Organizations Have Choices

It is much more efficient for organizations to have their high priced developers and programmers create new services, reusable components, and widgets than to utilize their talents creating the same screens, rules, workflows, reports, etc. over and over again. Now, those tasks can be left to less skilled resources. Why? Because by simply using software productivity tools, an organization’s less skilled resources can quickly deliver new functionality for end users.

Using software productivity tools means organizations are assembling instead of custom coding. This affords organizations the ability to move towards software manufacturing and achieve the same benefits in software development that the Industrial Revolution provided. As once the automotive and other industries moved to manufacturing rather than relying on skilled craftsman to deliver consumer products, productivity exploded and revenues increased.

Enter a Software Factory

If an organization can assemble it can also manufacture. With manufacturing, the next logical step is a **Software Factory**.

Simply put, a **Software Factory** is a facility that utilizes less skilled resources to deliver quality solutions with a level of predictability and scalability that enables organizations to realize significant savings and productivity gains that dramatically affect both the top and bottom lines. Because, if an organization plans to assemble rather than code and utilizes software productivity tools, they can effectively and efficiently achieve their business objectives as the tools, technology, and best practices exist.

Organizations are now safe and they can securely and effectively proceed toward their business objectives without waiting for custom code.

ObjectBuilders, Inc
jbrophy@objectbuilders.com
www.objectbuilders.com
610-783-7748 x103