

PSC White Paper: Rules, Tools and Frameworks

In today's rapidly changing business environment, it is becoming increasingly hard to keep up with customers and consumers. The problem for many of us is that we can't get our systems to work any faster, much less change them quickly enough to keep pace with the demanding marketplace. Adding more "code" is not the answer. Working harder is not the answer. This leaves working smarter, which is what this white paper is all about -- using tools and frameworks to work smarter.

Shrinking Cycle Times

The 21st century market forces are relentless. The increasing need to capture, distribute, and integrate information in real time through Portal and Web 2.0 technology is the latest driver. The pressure to pick-up-the-pace and reduce cycle times is two-fold. First is the time to access and display the information itself, and second is the requirement to re-configure the systems just as quickly. There is also the requirement to share information with a large audience, both inside and outside the organization. Everybody wants it, raw material suppliers, component vendors, contract manufacturers, distributors, retailers, and aftermarket support providers, and they want it now.

Survival depends on how well you can streamline and integrate your systems and on how quickly you can get your departments in sync internally and externally with your business partners. Better management of workflow and collaboration are more important than ever. Head-count and inventory reduction as well as more efficient distribution are also on everyone's mind. Providing customers and suppliers with direct access through the Internet is becoming 'the way' of life. The list goes on. So much opportunity, so little time. Time is the problem. Adaptability is the problem. Change is the challenge. We need to find a whole new way to work.

The Technology Paradox

The systems that we have built to better run our business have now become the problem. The applications that were once our friend are now the enemy. This seems to be the nature of things and logically so. Here is why...

Most software applications are based on logic embedded in their source code. Programmers combine business logic with computer code as they design programs and create the processes that solve business problems. This does not become a problem until it is time to make changes. Since the original logic is imbedded in hard code, making those changes kicks-off the repeat process of programming, coding, compiling, testing, reprogramming, recompiling and retesting. Making changes is time consuming and expensive. Therefore, they are only made when it is absolutely necessary.

Tools and Frameworks to the Rescue

Today, creative software vendors are responding to the greater need for change with rules-based tools that can modify the business processes without changing the underlying source code. They have

been joined by equally creative systems integrators and business process consultants who have gone one step further and are using those tools to develop frameworks that are literally transforming the way business processes are built and implemented. The result is a new level of configurability not possible with code-based systems.

The benefit from tools and frameworks is significant -- very significant. The general rule of thumb is that for each \$1 spent on a packaged application, \$5 to \$9 is spent on the labor for implementation, integration, and customization. With tools and frameworks, the time and cost is cut at least in half. To appreciate how these savings are made possible is to understand the underlying difference between hard code, and tools and frameworks -- an understanding that is made easier by comparing them with the differing operations of trains and trucks.

Let's start with the basic relationship between humans and computers. Humans do their work with hosted applications through interface devices such as 'dumb' terminals, web browsers, and wireless appliances. When entering a customer order, for example, the order entry clerk fills in the prescribed fields on the order screen that 'find their way' to the application's database. (See Figure 1)

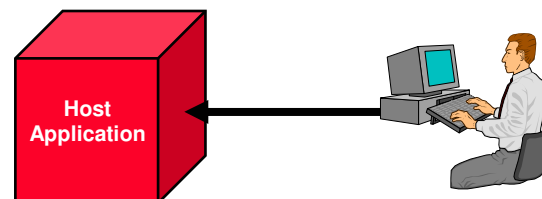


Figure 1

In the early days, this connection was very direct. The fields on data entry terminals were rigidly prescribed and controlled by the code-based programming logic of the host application. As each piece of data was entered at one end of the connection, it followed a prescribed path to its final destination -- it could only go to one place and could not be changed. It was like the travel of a train. The train engineer can only start or stop the train. He has no control over the destination -- that is the job of the tracks and switches. Because railroad track is expensive to lay and even harder to move, the destination (logic) must be known well in advance.

The same logic applies to code-based applications (Figure 2). The design logic has to be in place before the programming can begin. Consequently, once the application becomes operational -- once the 'track' has been laid -- making programming changes becomes a major event. Changes require new code, which means laying more track -- expensive and time consuming.

Direct, code-based connections, like trains, are the most efficient way to do repetitive tasks in a constant environment. The destination is known in advance and the path taken is the same every time. But in today's business environment there is very little that is constant. With the increasing rate of change and the decreasing time available

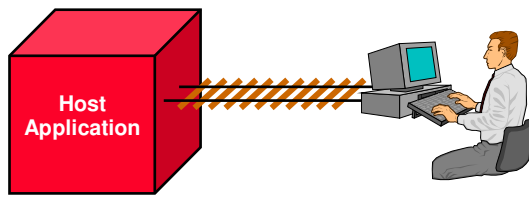


Figure 2

to respond, adaptability has become much more important. Code is not adaptable, at least not easily. We have to find a better way. We have to move from 'trains' to 'trucks.'

Trucks don't carry nearly as much as trains. Their cost-per-mile is more than double. In that way they are not as 'efficient,' but they are more adaptable. They can go anywhere they want and change direction at a moment's notice. All that is needed are instructions from the customer or dispatcher. The same logic applies to tools, and frameworks. Since they are not code-based, they can be used to make systems more adaptable. Given the right instructions, they can be used to make changes in near-real time. (Figure 3)

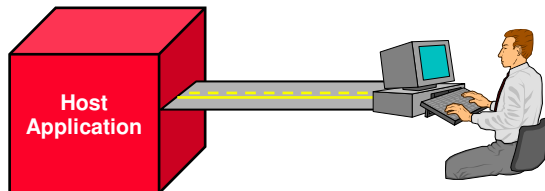


Figure 3

Because they can be reused in a variety of situations; tools and frameworks reduce both time and costs. This is why hard-coded software is giving way to rules, tools and frameworks – just as trains have given way to trucks. Adaptability matters. Time matters. Total cost matters.

One last comment about application code. Moving to tools and frameworks does not mean that code-based applications can be ignored altogether. Not at all. Application code will always be the foundation of any business process. The trick is to use it as such and leave the day-to-day functionality to more adaptable alternatives.

The Architecture Factor

The use of rules, tools and frameworks need not be limited to integrating with applications. They can also be applied in the form of Service Oriented Architecture (SOA) at the infrastructure level where the applications themselves are installed and implemented. Through the use of SOA, applications can be quickly integrated with the other components of the business process. Conceptually, Frameworks and SOA are one in the same.

Working Smarter

The best way to increase productivity is to be more adaptable – to be able to respond to changes when they occur, as they occur. This is why rules-based tools and frameworks are so important – especially frameworks. Frameworks and SOA are organized 'thought processes' or rules that can be used for all or part of a process or

software system. They are best practices that are expressed as reusable business objects, application & solution templates, models and component-ware. They provide a context for the assembly of components in a library of reusable objects for the purpose of solving business problems in a more efficient manner. Like tools, they speed development and reduce cost.

There are several advantages to using frameworks. At the top of the list is the saving of time and money, which is followed by the increase in quality and the creation of a more flexible architecture.

Time: Frameworks fit well within the confines of the '80-20' rule. Eighty percent of the work in most business projects of a given type is common to all projects of the same type. Using frameworks for the eighty percent can reduce project design time by nearly the same amount. This leaves only the remaining twenty percent for the heavy lifting needed to meet the unique requirements of a particular business.

Cost: Less time equates to less cost. This means that frameworks can reduce costs three ways. Because they are reusable, their development costs are spread over multiple projects. They can also be applied one at a time to solve specific problems – no more, no less – thus keeping development and maintenance costs to a minimum. And most importantly, because they are "on the shelf and ready to go", they can reduce total project cost by more than one half.

Quality: The re-usable aspect of frameworks means that most of the work is also 'pre-tested.' This feature translates into less time testing and correcting, which further drives down time and cost.

Flexibility: Most applications, with their logic in the code, have to be programmed in advance making them slow and inflexible. Frameworks, on the other hand, are developed using rules-based procedures and functionality that 'sit' on top of more widely used, code-based foundations. Freed from the constraints of code, they are inherently more flexible and can be used and re-used just about anywhere. They also leverage the built-in security, scalability, reliability, and availability that are inherent with broad-based foundations. They are adaptable.

Running Your Business Your Way

Rules, tools, and frameworks not only let you work smarter, but also help you keep doing what you do best, only better. In an era where there is already too much change, this is very important. Having the ability to make changes as needed, one at a time, while holding your course is the better strategy. Just ask the tortoise and the hare. The obvious benefits are less disruption, less training, and the ability to make smaller and more frequent adjustments along the way. Less obvious, but maybe more important, is the consistent architecture and less dependence on outside software.

For more information on this subject, please contact PSC Group, LLC at (800) 592 8003 or send an e-mail to info@psclistens.com