Build and Deploy across multiple platforms

Written by Jeff Tickner
Monday, 20 April 2009

In today's enterprise environment, very few companies have the luxury of having a single platform to support. This means multiple development tools supporting multiple applications on multiple platforms. You may even be supporting ‘separate but equal' development teams charged with supporting specific platforms and applications.

Maybe 10 years ago this was the norm, with the web team coding the HTML in notepad while the enterprise developers worked on the business applications. If there was any data on the website it was relatively static, retrieved from the DB for display purposes. The enterprise applications were simple GUI applications with some command line interfaces.

When you made a substantial change, you coordinated the change as best you could to limit downtime and that meant a group of people spending the weekend at work shepherding objects around the systems. Hopefully, come Monday morning, at least the business critical apps all worked. The fact that the website was down or displayed cryptic error messages wasn't a big deal.

Today that is unacceptable performance. The web now accounts for a substantial portion of the user interface, whether it is exposed to customers on the internet, or it is for internal use by your employees. So the need is to design your build and deploy process to support all the diverse platforms that are impacted by a change. If you don't identify all the business applications that will be affected, then you are dooming yourself to the Monday morning call describing what percentage of users cannot complete their tasks because their interface no longer works properly.

A successful change is the result of planning and controlled execution. Impact analysis of a database change should provide a list of components that are affected by the change. The list should reflect ALL platforms in use by your environment unless the database in question is in isolated use.

Whatever the engine is behind your process – Hudson, Maven, Ant, MSBuild or a combination of these – it should be able to update components on multiple platforms and provide feedback on the results. Depending on the platforms involved, this can be a complex task.

To achieve these 2 goals requires both the tools and some kind of access to all platforms. Let's assume you have a truly mixed environment. Your users are on PCs running Windows. You have a Linux server with Apache serving up the external part of your website. You have several Windows servers providing normal network functions, and one has IIS secured with active directory that provides access to your intranet site. You have SQL server on one of the Windows boxes, and you have some of your core business applications running on an IBM i using DB2. Eclectic enough? Oh – the webmaster has a Mac for designing the websites.

What is the common factor here? Data. All of these platforms are sharing data to some extent. It may be something as simple as the user profiles or customers; it may be a list of products or raw materials that are shared between sales and manufacturing. Even if, as in our example, you have multiple databases, there is still the requirement of interaction to prevent chaos between sections of the company.

In our example, I'm putting sales on the x86 platform and manufacturing on the venerable IBM i (iSeries). Since this is fictional, I can do something crazy and expand the SKU used to track products from 5 digits to 10 alpha because I am going to run out of unique SKUs. This is fiction, right? I will not have real howls of disbelief from every developer telling me that they will have to change every single program and display.

First, you need to build a list of discrete components that use the field. That's easy, you say: "All of them!" Well, we know that field is used in both databases, so whatever means is used to connect the two databases will be part of the change. There are a number of cross-reference tools for the IBM i. Hopefully, you use one, and it will give you a list of components that use the SKU field. On the x86 side, there are also cross-reference tools, but they are platform centric. They will return the components on the same platform. There aren't many tools that can look at both databases and all components on all platforms and provide a complete list of both IBM i and x86 components that use a specific field, but they are out there (ARCAD Software's Observer, for example). Lacking such a tool, it will be a manual process to build this impact analysis.

With a complete list of components, you can now look at your build and deployment process to see how you will manage the changes, test them, and deploy them in a synchronized way to end users. There should be a host platform for your process to ensure synchronization; otherwise, you will have to manually synchronize multiple processes running on multiple platforms. This host platform might arise from where the most development is taking place and hence the highest level of experience among your staff.

On the x86 side, maybe you are currently using MSBuild, and you have Ant tasks plugged in to manage the web components on the Apache server. There is also support for Ant on the IBM i, but since there is no easy way to flag components on the i for processing, or an easy to access the repository, we can anticipate a separate process on this platform. Hopefully, you are using some kind of change management, and you can manually stage a process to be kicked off either manually or with interaction via Ant.

Maybe you are heading toward more open source and are using Hudson with Ant to manage your builds and deployment. Hudson provides the ability to design a consistent process for the build and deployment cycle. Its framework provides an interface for users to interact with as they move forward in the development cycle, and its plug-in architecture means that you can add the functionality you want to interact with the appropriate tool/platform. Ant's ubiquity comes in handy, as there are several Ant tasks that can interact with Visual Studio projects for the build process. NAnt can also be invoked from Hudson using the NAnt plug-in or Ant, as most Ant tasks are compatible with NAnt as well.

You have a high degree of flexibility because of the interactivity of all these tools. Communication between the tools can be simple TCP over your regular network or SSL to an external web server. The SSL connection can be used for both the actual movement of an object and any deployment commands needed to install the

object properly. You are designing the process and leveraging the functionality of these various tools to execute the commands required to accomplish the goal.

Unfortunately with this degree of flexibility comes complexity. While many functions are available freely, like the Ant tasks that interact with Visual Studio, there is still the requirement that you design and implement the process. If you have any experience available in house, that would be a strong motivator to use technology with which you are familiar. There is no best solution for any environment. You need to look at where most activity takes place and where most of your expertise lies. There are tools on platforms as disparate as the IBM i that can provide all this functionality. Plus, one possible benefit of going with a commercial tool is you may be able to pay for the configuration of your designed process. Additionally with a commercial package, you have the option of ongoing support.

All companies have a process in place for build and deploy. It may be very simple, and it may vary from platform to platform, but there is always a process currently used in some way. Document these processes and look at the workflow. Is it what you want, or can it provide additional functionality to make your work easier or your development cycle more reliable?

Sit down and lay out the ideal process, with whatever bells and whistles you desire. Validation of builds, approval of deployments, automated messaging, integration with helpdesk, error management, all or nothing deployment are all possible features.

Now re-examine the process and prioritize the features, as each feature will either increase the time to build such a process yourself, or push you closer to a purchased solution with that feature set. If all the features you have designed are desirable, then you need to budget accordingly.

The next step in deciding a course of action is considering what tools you use for change management or build and deploy processing.

Can these be extended to support additional platforms, or do they have that capability built in to the package?

Do they provide any kind of cross reference across the platforms involved to provide impact analysis of change?

Do they provide, or can they be extended, to provide the functionality in your design spec?

If you decide to take a look at the tools in the marketplace, it can be helpful to start with the most unusual platform on which you have any substantial development. You will find many choices for x86 support. However, the more exotic platforms will have limited support, and the tools available for those platforms might well have support for x86 objects.

Once you have your deploy and build process updating all platforms and making sure dependant changes go into production in the same time frame, you will find that Monday morning aren't quite so bad anymore.