Best Practices for SharePoint Development and Customization

Written By:

Mario Fulan,
MCM, Account Technology Strategist, Microsoft

Ricardo Wilkins,
SharePoint Practice Lead, Improving Enterprises

QUEST SOFTWARE®
Simplicity At Work™

Contents

Abstract

SharePoint enables developers to quickly deliver prototypes and solutions to users. However, just as quickly, users respond with enhancement requests and new requirements, often overwhelming the development team. This white paper explores the challenges that SharePoint developers face and details a set of best practices for responding to those challenges, so that you can make the most of your SharePoint investment.

Introduction

SharePoint adoption increases when users come to appreciate the many sophisticated business applications that their development teams can implement with the platform. As business needs inevitably change, developers are faced with the challenges of deciding when to advance past creatively using out-of-box features and modules, to incorporating the functionality of third-party web parts and custom development efforts in order to meet requirements.

These SharePoint development efforts are often faster than other development projects. Traditional platforms typically require a great deal of work building the infrastructure in which the business solution will reside: 'plumbing' components like security, user interfaces, and authorization systems must be created before the actual solution feature set can be presented. One of SharePoint's main strengths is that much of this 'plumbing' is already done, so developers can focus immediately on the business requirements at hand. There's no need to create a new security model, for instance, when the People and Groups model already exists in SharePoint and is available to the end users through an existing and familiar user interface.

But SharePoint's ability to deliver rapid prototypes and composites introduces its own challenges. Users quickly get the first version of their solution, and, just as quickly, they proceed to give feedback. This feedback usually includes requests for additional features, with expectations of similar turnaround time from the development team. So, while SharePoint is continuing to make a positive impact on the business, the development team can struggle to keep up with the seemingly incessant demands of its user base.

The success of a SharePoint development team depends on their effective use of best practice methodologies, as well as their ability and foresight to understand when to take advantage of third-party add-ons and services to limit the effort required while accelerating the quality when developing fully customized solutions. While no amount of tooling or methodology can meet the needs of every group of demanding users, leveraging the proper toolset, be it out-of-box, third-party or custom, can help ensure that the development process runs smoothly.

Although it's not accurate to call SharePoint's native development challenges "limitations" (as being built on the .NET web application platform, just about any custom solution can be created with enough effort), it's also not practical to always expect to have the luxury, budget, or timeline needed to build fully-customized solutions in the timeframe that quick-to-market business users often demand. A SharePoint developer must understand these challenges, and be prepared to address them, as well as plan for them accordingly as it relates to the long-term success and longevity of the solutions that are built.

This paper discusses many of the decision pains that developers face when put in this unique position and provides guidance about how to best address them. We'll examine the native development options in SharePoint 2010, highlight both the platform's strengths and its limitations, and show how these limitations can be mitigated through best practices and use of third-party add-ons and expertise.

A Developer's Daily Pains

Introduction

SharePoint's features and services enable developers to create solutions faster and easier. But quick delivery of useful applications introduces challenges for developers, as the user community begins to clamor for increasingly powerful solutions that often require heavy customizations.

Consider this scenario: Your users have a general idea about a workflow solution they need for their line of business. As a developer, you realize that there are several native SharePoint features that could be used to prototype the solution. You quickly deliver a mock-up of the application, which is well-received. As the users begin to better understand their own needs and start to see the power of SharePoint to meet those needs, more requests are made. Soon the functionality needed for the solution exceeds the capabilities of out-of-the-box SharePoint features, and custom-coded solutions must be created. Suddenly the effort of keeping up with the demands and enthusiasm of the user base begins to strain the developer's ability to maintain and update their custom solutions.

It is this type of scenario that produces a developer's day-to-day pains. Even the best developer will find it challenging to manage the growing needs of the user community while at the same time trying to produce quality solutions within acceptable timeframes. Let's look more closely at the specific challenges developers face.

Extremely Tight Deadlines

One issue that plagues every developer at some time in the development cycle is the competing demands of functionality, schedule, and resources. Sometimes the issue shows up immediately, when the customer tells you they really must have the solution "no later than tomorrow." At other times, the tension shows itself after you are done with the prototype: new enhancement requests start to take on a life of their own, and the functionality of the prototype pales in comparison to the "next development cycle." And sometimes the issue raises its head when you finish developing a fairly functional solution but the final result has enough spaghetti to feed a family of six; your senses tell you to re-factor and clean up the mess, but there "just isn't time."

Failure to Separate User Interface and Logic or Data

As professional developers we should know better, but often we use the wrong design approach and get into trouble right at the start. Let me give you a real-life example: Developers working on a new public-facing internet site were looking for a "quick way" to put events in the right-hand rail of several pages of their site. They wanted these views to provide a way for a user to click through to a registration site that would offer a link to learn more, which would open another page with a more complete description of the event. The initial solution put forth by the developer was to create reusable content and place it on a content editor web part, and guide the end user in filling in the appropriate information.

However, this solution would have required two dozen direct authors of content, and the page would have to be modified and approved for each and every event. In addition, the editor of each page would have to meticulously review each change to verify that the author had followed the template, since the system did not prevent content authors from changing the initial event template design while filling out the specific details of the event. Ultimately, the

page design wasn't really changing, so why was the page editor getting involved at all? There had to be a better way, and indeed we present one below.

Configure Versus Custom Coding

Another pain is created by the sheer depth and breadth of the solution space covered by a product as comprehensive as SharePoint. From InfoPath to workflow; from search to business data connectivity; from Web Content Management to Enterprise Content Management – it is staggering the variety and richness of the out-of-the-box capabilities of SharePoint. Add to that the additional capabilities beyond SharePoint Foundation 2010 to those provided by the Standard and Enterprise Client Access Licenses (CALs), and a SharePoint developer must be well versed in hundreds of out-of-the-box features, dozens of APIs, and numerous tools such as web parts and configuration options. SharePoint 2010 now adds additional capabilities for sandbox solutions and three new client coding models. No one can possibly know everything there is to know about every aspect of SharePoint development. Due to this fact, most developers tend to specialize.

Creating solutions involves the decision of whether to configure SharePoint or rely on custom coding. Best practices dictate that leveraging out-of-the-box features of SharePoint is the best approach whenever possible, since the solution will be faster to develop and easier to maintain. What if the solution calls for experience or expertise in an area that the developer has had little or no exposure to? The out-of-the-box solutions may be unknown or pose higher risk in the developer's mind simply because they are unfamiliar. Third-party tools or solutions may be unknown or only offer a partial solution to the requested functionality. In these cases the developer may choose to create new solutions from scratch, rather than take the time to incorporate existing solutions. They may be hesitant to expose the project to the risk of learning some unfamiliar solution, even if that solution may be an adequate or even better solution in the long run. For example, while one could certainly build a custom web service for extracting data from an external SQL Server database, often a better approach is to use something native like Business Connectivity Services to extract the data. However, a developer unfamiliar with BCS may choose to fall back on code that is more familiar. This issue is even more pronounced for developers who are new to SharePoint but may have a wealth of experience with creating custom-coded solutions in .NET.

A Flood of Change Requests

One of the most significant pain points for developers is the need to respond to change requests from end users. After enough use of any solution, end users inevitably identify enhancements or fixes that they would like. Simply collecting and filtering these requests can be challenging, depending on the size of the user base. Then comes the challenge of finding enough time to meet each request and deploy the changes responsibly. This challenge is compounded in situations where policy dictates a multi-environment deployment process.

Solutions to Developer Pains

SharePoint 2010 provides a platform and feature set that can enable developers to manage these pains successfully. Below you will find key best practices that will help.

Empower Users with Reusable Solutions

To reduce the pressure on the development staff, empower your users to be more self-sufficient by enabling them to develop creative solutions to fit their own needs with no or little support. Empowered users will try out a variety of solutions on their own, often succeeding without requiring time from developers. And when empowered users do request help, their requests tend to be better thought out and more realistic for the SharePoint platform. The users also tend to be more directly engaged in the give and take of the development process because they are more aware of what is provided by the platform.

Web parts are one good way to delivering reusable solutions that empower users. SharePoint 2010 comes with several web parts that meet many basic needs, including the List View, List Form and Content Editor web parts, as well as more advanced web parts like the XSLT List View or Silverlight web parts. Each provides a mechanism for the delivery and personalization of applications without the need for full customization and coding. Some of the out-of-the-box web parts can be extended by the developer to provide new solutions to users. For example, providing users with new group and item styles for List View or Content Query web parts can provide a customized UI without requiring the complexity of overhead of a completely custom-coded solution. Even the more advanced web parts can still spare the developer from the task of writing and deploying custom code. Silverlight solutions in SharePoint 2010, for example, provide a way for developers to create powerful custom solutions which are easily deployed and used by end users.

When out-of-the-box web parts are insufficient, you can create your own custom web parts, based on the ASP.NET web part infrastructure that SharePoint builds on. This is where the value of SharePoint as a development platform begins to truly pay off for the developer. SharePoint 2010 has included many enhancements that make custom development a more mature and feature-rich experience. An example of this in the case of web part development is the inclusion of the Visual Web Part template. Not only has the web part development experience been enhanced, but deployment and troubleshooting of web parts, or any custom code, has been greatly improved as well.

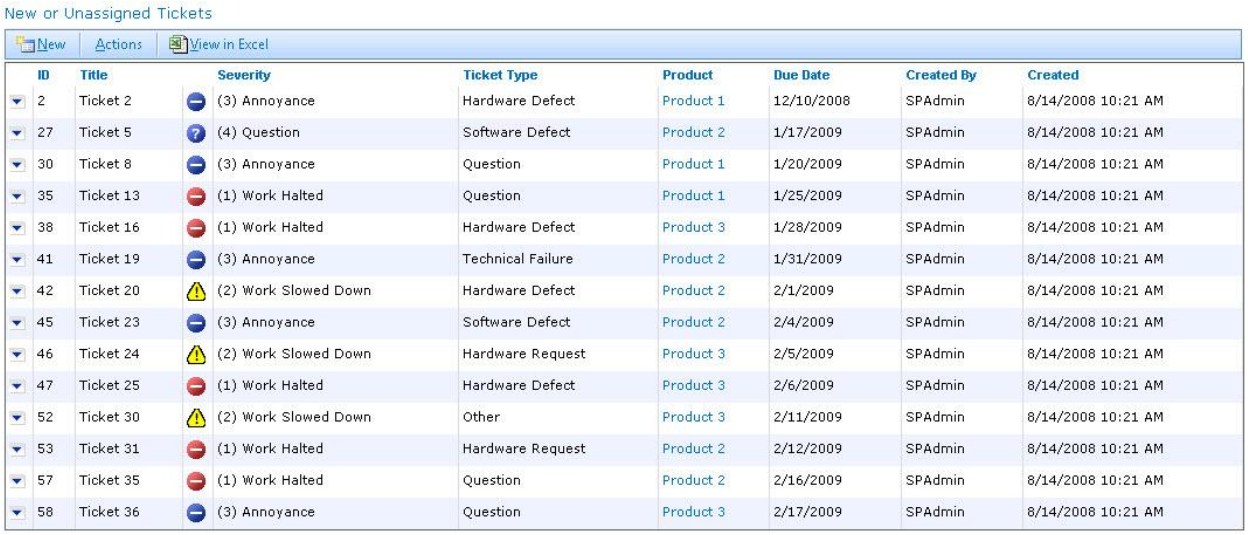
However, as was mentioned earlier, due to the enormity of the platform there will certainly be times when the experience of the developer is insufficient. In these cases, expertise from local consulting service providers should be considered. In particular, Microsoft-certified managed partner organizations can be particularly useful, as they may have access to SharePoint product team resources that other organizations may not. But again, whether the development is done in-house or outsourced, the pros and cons of pursuing a fully-customized code solution should be fully explored before proceeding down this path.

In addition to these options, you could also minimize your investment in development or consulting time for building highly customized reusable solutions by looking to third-party components, like Quest Web Parts for SharePoint, which provide additional functionality beyond SharePoint's native web parts.

For example, a native List View web part has several limitations that many users need to work around, including:

- Inability to group on more than two columns
- Inability to rename or resize columns
- Lack of robust filter or search capabilities within the web part

The Quest qListView web part addresses these limitations and more with an easy-to-use configuration interface that lets end users add this important functionality without code.


New or Unassigned Tickets

New | Actions | View in Excel

ID	Title	Severity	Ticket Type	Product	Due Date	Created By	Created
2	Ticket 2	(3) Annoyance	Hardware Defect	Product 1	12/10/2008	SPAdmin	8/14/2008 10:21 AM
27	Ticket 5	(4) Question	Software Defect	Product 2	1/17/2009	SPAdmin	8/14/2008 10:21 AM
30	Ticket 8	(3) Annoyance	Question	Product 1	1/20/2009	SPAdmin	8/14/2008 10:21 AM
35	Ticket 13	(1) Work Halted	Question	Product 1	1/25/2009	SPAdmin	8/14/2008 10:21 AM
38	Ticket 16	(1) Work Halted	Hardware Defect	Product 3	1/28/2009	SPAdmin	8/14/2008 10:21 AM
41	Ticket 19	(3) Annoyance	Technical Failure	Product 2	1/31/2009	SPAdmin	8/14/2008 10:21 AM
42	Ticket 20	(2) Work Slowed Down	Hardware Defect	Product 2	2/1/2009	SPAdmin	8/14/2008 10:21 AM
45	Ticket 23	(3) Annoyance	Software Defect	Product 2	2/4/2009	SPAdmin	8/14/2008 10:21 AM
46	Ticket 24	(2) Work Slowed Down	Hardware Request	Product 3	2/5/2009	SPAdmin	8/14/2008 10:21 AM
47	Ticket 25	(1) Work Halted	Hardware Defect	Product 3	2/6/2009	SPAdmin	8/14/2008 10:21 AM
52	Ticket 30	(2) Work Slowed Down	Other	Product 3	2/11/2009	SPAdmin	8/14/2008 10:21 AM
53	Ticket 31	(1) Work Halted	Hardware Request	Product 2	2/12/2009	SPAdmin	8/14/2008 10:21 AM
57	Ticket 35	(1) Work Halted	Question	Product 2	2/16/2009	SPAdmin	8/14/2008 10:21 AM
58	Ticket 36	(3) Annoyance	Question	Product 3	2/17/2009	SPAdmin	8/14/2008 10:21 AM

Figure 1. Using Quest Web Parts, anyone from developers to end users can dynamically combine and group SharePoint lists without custom code.

Moreover, Web Parts also enables users to combine or filter information, create compelling 3-D charts, and improve application navigation. While this functionality could be achieved through custom coding, there are considerable time savings in not having to code and maintain this functionality. The developer gets a large percentage of the capability needed to fulfill the requirements from Web Parts, and they can then focus their attention and expertise on extending Web Parts through custom actions or by coding the additional functionality that custom actions can't do.

Another huge advantage of third-party components like those from Quest is that the vendor is responsible for bug fixes and upgrading their component when new versions of SharePoint are released, relieving the developer of this duty.

As a result, end users are enabled to customize their own sites and applications, so they get what they need from SharePoint, when they need it. Meanwhile, developers benefit by reducing their backlog of tedious customization requests – while still delivering value to the users – and can focus on more value-added coding.

Separate User Interface from Logic or Data

Good software design always tries to separate the user interface from the business logic and the data. This separation makes the solution more reusable, more testable, and more likely to hold up to requirements changes that will inevitably come in the future. However, many developers lean way too heavily on very generic solutions, such as the

Content Editor web part or some variation, which requires the page author to add content directly to pages for every data element.

In the example given earlier in this paper, events targeted to a specific page would require the page author to copy content to a page for each event using the "reusable content" library in SharePoint. Although this had "reusable content" in the solution, this was far from a reusable solution for the user. After a bit of mentoring and coaching, the developer in this case created a new custom content type for a custom event. A quality UI web part (e.g., XSLT List Viewer, Content Query web part, or Quest qListView data rollup web part) was then used to display the data from this list. For this solution, the developer used customization to extend an existing web part rather than create something completely from scratch, and the page designer would configure and customize the web part properties and parameters to match the data to be displayed on each page. Content authors would be directed to one or more content lists containing the custom event data and then simply fill out data forms for data about the event. A combination of properties (such as the event category, the approval status and event start and end dates) would be used to drive data visibility on each page.

The main lesson here is to separate page design from content creation. Allow users to generate content in lists or libraries and use web parts to surface that data to each page in a properly formatted way. If data must be targeted for each page then use parameters and configuration data to target the data appropriately rather than have content authors modifying each page of your site directly.

This approach offers several benefits:

- Content authors can edit event content but have no access to page design, which prevents them from inadvertently causing issues to the overall page design.
- Rules in the web part logic or in the XSLT transforms can govern data inclusion and exclusion from the display. Therefore, page editors no longer have to edit content on a daily basis to remove stale content.
- This same approach can be used to display a huge variety of data, including calendar events, product listings from catalogs, and periodicals in a library, making this solution a good pattern to follow for UI development.

The main point here is to be sure to separate elements of development, design, and content. Ensure that user content in SharePoint is produced as items in a SharePoint list or library (unless you are using a free-form entry method such as a wiki, where users are directed to input content directly into a page). In most practical solutions, when content authors create content in a list or library, the designer can then lay out the page(s) for the solution using custom or out-of-the-box web parts to display the data in a meaningful way. The developer is called upon to create or modify these web parts to suit the needs of specific applications. If you find your SharePoint solutions rely heavily on content authors to directly input content on pages, then you have a fairly easy way to remove at least one of your self-inflicted developer pains. Change the design just a bit and your users will be coming to you much less often, so you can concentrate your development effort on better things.

Become a Student of SharePoint

SharePoint developers generally need educating, as their skills aren't in-born. In the beginning, developers should be put through a general curriculum including .NET, javascript, debugging, and design skills, then continuing that training with several weeks of SharePoint-specific training. In this time, developers should learn the ins and outs of SharePoint's out-of-the-box capabilities, as well as other major components of the platform. Historically, developers with

little SharePoint knowledge tend to move too quickly to creating custom code for solutions when there are often solutions in out-of-the-box tools or web parts, whether that is directly or with proper configuration and minimal customization. Developers should learn to determine if out-of-the-box features are inadequate and how to proceed from there before immediately jumping to custom coding as a solution. Instead, research and consider third-party add-ons, like Quest Web Parts for SharePoint, which extend the functionality of SharePoint yet don't require custom coding. Note that some third-party web parts can be extended with custom actions, which represent custom programming but with significantly fewer hassles than a pure custom-coded approach.

Nevertheless, sometimes project requirements are so specific that custom-developed code is required. When true custom programming is required, developers should tread carefully to ensure that the custom approach meets best practices and proper design patterns, and is both maintainable and robust enough to meet changing needs. The effects of a bad decision about whether to configure or use custom code can often be felt throughout the life of a project.

Manage Change Requests Effectively

There are many tools designed to help development teams manage change requests. The most recommended tool for application lifecycle management in SharePoint is Team Foundation Server (TFS). But whatever the tool you choose, when trying to respond to the demanding change requests of your end users, it's important to have the following processes in place:

- **Request tracking** – You need something similar to a help desk ticketing system to enable you to document each request and monitor the progress toward its completion. Ideally, this system will also enable requestors to monitor the progress of their requests. The ability to add notes, attachments, and status information is also helpful. TFS achieves this through Work Items, whose type and parameters can be customized and configured.

- **Source control** – Source control is important to maintaining a well-documented and repeatable development lifecycle. This not only serves as a way to securely store source code for custom solutions, but also provides an audit trail for each part of the process, including things like who worked on each part of the solution and date stamps for each part of the process. If a user asks a follow-up question about a particular feature, source control documentation can help the development team find the answer. TFS enhanced features, such as Annotation and Changesets, can add additional functionality to source control as well.

- **Build automation** – Implementing a repeatable script for compiling, testing, and/or deploying a solution has benefits for the development team, IT administrators, and end users. For developers, it automates many mundane but necessary tasks for each new build of the solution, freeing their time to focus on the problem domain. For IT administrators, it adds reliability to the process by offering consistent repeatability and reducing the chance of error. And it gives end users a certain level of confidence, since the results of nightly builds, for instance, can be examined by all. The TFS Build Server provides all of these features, and can be configured to address SharePoint-specific deployment options such as the drop location of a resulting WSP file.

Manage Deployments Carefully

After development is complete, new solutions must be deployed to the proper environments. While deployment falls largely to IT administrators and engineers, the developer also plays an important role in this process, by adhering to corporate governance as it relates to deployments and by properly packaging the solutions.

Multiple Environments

Many organizations require at least three SharePoint Farm environments: development, test (or staging), and production. Solutions, particularly custom-coded applications, need to move from the development environment in which they were created to the production environment where they will ultimately reside.

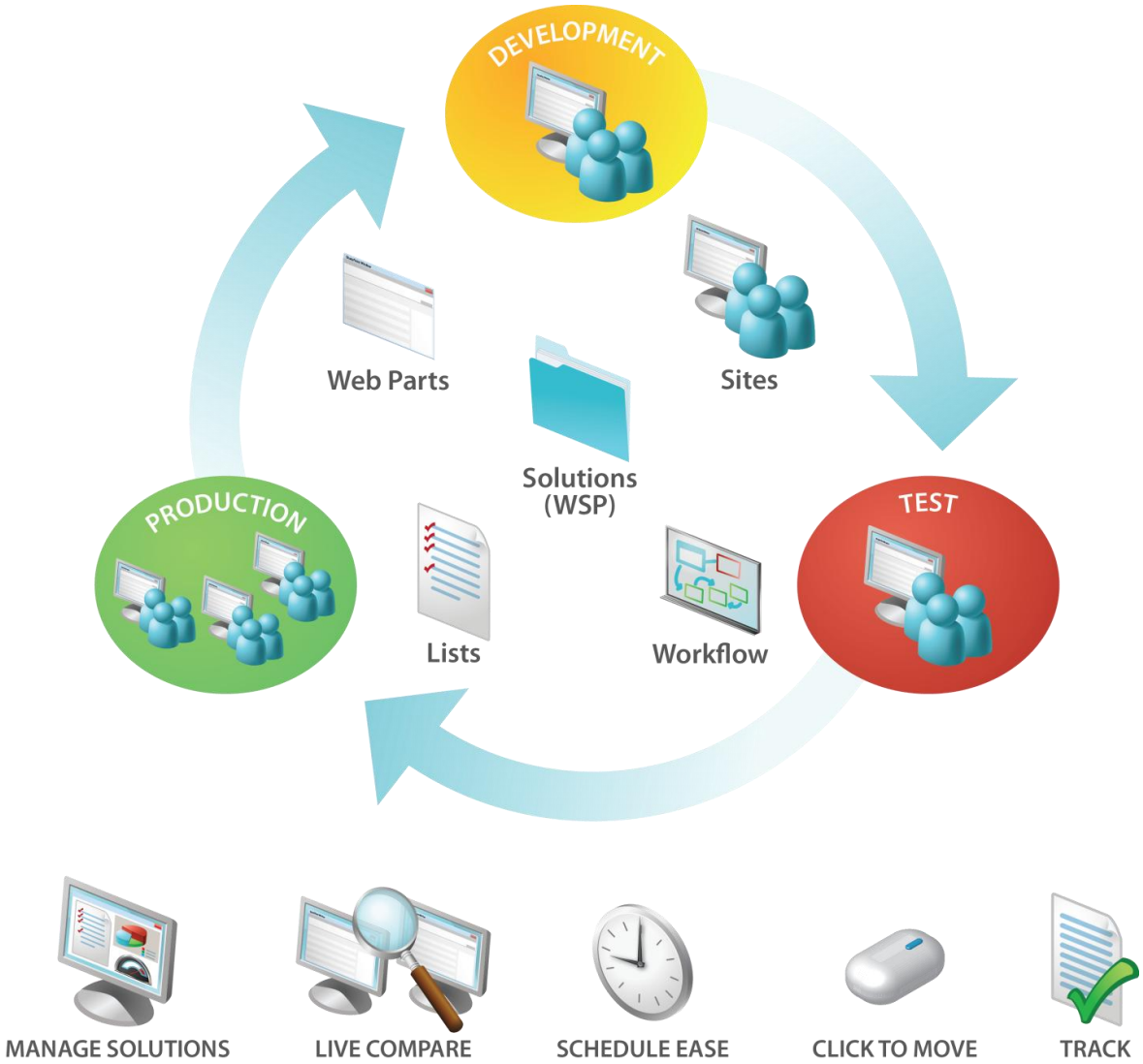


Figure 2. With Quest Deployment Manager for SharePoint, you can easily control which, how and when applications and customizations get moved across SharePoint environments.

A development or test environment is most useful when it accurately reflects the artifacts and configurations of the production environment that it represents. However, there are not many native SharePoint tools to help keep these environments in sync, particularly since many of the nuances involved, like, for instance, the specifics of using a hybrid physical/virtual server environment, are unique to each organization.

Another option is to do full backups and restores from one environment to the other, but this is not always beneficial and is, often, a symptom of inability to manage discrete changes among the environments. With complete backups and restores comes unrelated source material that takes up space in the new location. Instead, it is best to identify the specific differences between environments and make case-by-case decisions about what needs to be copied or changed. Out of the box, SharePoint 2010 doesn't offer any compare tools. Quest Deployment Manager for SharePoint, however, provides a graphical comparison tool enabling you to easily determine if a component from one environment needs to be deployed in another. Quest Deployment Manager then gives administrators a "one-click" ability to move artifacts from development to testing, and then later to production.

Governance

It's important to establish, early on, policies about development and maintenance of SharePoint solutions. Many organizations that are new to SharePoint opt to institute a no-code policy, in which no custom code solutions are allowed in any environment. Organizations using SharePoint 2010 might make an exception for deploying sandboxed solutions across environments. There may even be limitations on what third-party tools can be installed. These are all factors that development teams must consider when determining how they will create and deliver their solutions to end users.

Packaging

The recommended best practice for moving solution artifacts across environments is to create a solution package called a WSP. This package can be deployed and activated in a SharePoint farm by an administrator, and can also be deactivated and retracted if needed. For many organizations, this is the only way code may be deployed to any non-development environment, especially production. This practice not only benefits SharePoint administration engineers; it also benefits the development team, by allowing them to deliver solutions in a repeatable and measurable way.

For organizations whose governance does not demand WSP packages in every situation, there are alternative ways of delivering solutions. For instance, a new custom list can be saved as a template and moved to the new environment in such a way that users can create new lists based on the template. Many third-party add-ons provide export/import features for their artifacts.

Conclusion

Your SharePoint development team does not have to be overwhelmed by custom development and enhancement requests. By providing your users with the right native tools and third-party add-ons like Quest Web Parts for SharePoint, you can empower them to customize their own sites with less assistance. By employing good design that separates the user interface from the business logic and the data, you can create solutions that are more flexible and reusable. With effective request tracking, source control, and build automation, you can stay on top of the queue of change requests. And by establishing effective deployment processes, including the use of third-party tools like Quest Deployment Manager for SharePoint, you will get solutions and enhancements to your users faster and more reliably. Together, these best practices will help you gain the most value from your SharePoint investment.

About the Authors

Mario Fulan

Mario Fulan is an account technology strategist at Microsoft for state and local government customers in Ohio and Kentucky. He is a Microsoft Certified Master (MCM) in SharePoint with more than 20 years of professional experience. He is one of fewer than 10 individuals with Microsoft Certified Masters for both SharePoint 2007 and 2010 in North America. Mario has a strong depth and breadth of technical expertise with several Microsoft-related technologies. Specific technologies include SharePoint 2010, Microsoft Office SharePoint Server (MOSS) 2007, Windows SharePoint Services (WSS) 3.0/2.0, MS SQL Server 2008, Active Directory Federation Services (ADFS) V2, Windows Identity Framework (WIF), MS Team Foundation Server (TFS), .NET Framework, C# and VB.NET. Additionally, Mario holds several certifications above and beyond his MCM in SharePoint including Microsoft Certified Solution Developer, Microsoft Certified DBA, and Microsoft Certified System Engineer.

Ricardo Wilkins

Ricardo Wilkins, SharePoint Practice Lead for Improving Enterprises, is a seasoned software developer and technology consultant. Ricardo has delivered solutions for the public, non-profit, and private sectors, and has helped clients maximize their investment in Microsoft technologies by leveraging SharePoint as a development platform for business solutions and integrated systems. He specializes in SharePoint development, combining out-of-box feature sets with custom development tools like Visual Studio, workflow for business automation, and Team Foundation Server for application lifecycle management. Ricardo is very active within the SharePoint community, and speaks across the nation at conferences, user groups, and SharePoint Saturday events. He also maintains several widely-read blogs, and his articles have been featured on multiple SharePoint and IT-related websites.

.

Quest Software World Headquarters
LEGAL Dept
5 Polaris Way
Aliso Viejo, CA 92656
www.quest.com
email: legal@quest.com

Updated—January 2012

About Quest Software, Inc.

Quest Software (Nasdaq: QSFT) simplifies and reduces the cost of managing IT for more than 100,000 customers worldwide. Our innovative solutions make solving the toughest IT management problems easier, enabling customers to save time and money across physical, virtual and cloud environments. For more information about Quest solutions for application management, database management, Windows management, virtualization management and IT management, go to www.quest.com.

Contacting Quest Software

PHONE 800.306.9329 (United States and Canada)
If you are located outside North America, you can find your local office information on our Web site.

EMAIL sales@quest.com

MAIL Quest Software, Inc.
World Headquarters
5 Polaris Way
Aliso Viejo, CA 92656
USA

Contacting Quest Support

Quest Support is available to customers who have a trial version of a Quest product or who have purchased a commercial version and have a valid maintenance contract.

Quest Support provides around-the-clock coverage with SupportLink, our Web self-service.
Visit SupportLink at https://support.quest.com.

SupportLink gives users of Quest Software products the ability to:

- Search Quest's online Knowledgebase
- Download the latest releases, documentation and patches for Quest products
- Log support cases
- Manage existing support cases

View the Global Support Guide for a detailed explanation of support programs, online services, contact information and policies and procedures.

WPW-BestPrac4SharePointDevAndCust-US-SW-01102012