

Managing SharePoint Application Development Lifecycle

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in examples herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

© 2007 Microsoft Corporation. All rights reserved.

Microsoft, MS-DOS, Vista, Windows, Windows NT, Windows Server, ActiveX, Excel, FrontPage, InfoPath, IntelliSense, JScript, OneNote, Outlook, PivotChart, PivotTable, PowerPoint, SharePoint, ShapeSheet, Visual Basic, Visual C++, Visual C#, Visual Studio, Visual Web Developer, and Visio are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

All other trademarks are property of their respective owners.

[License Agreement](#license)

SharePoint Application Development Lifecycle Management

Mark Liu, Jonny Trees, Adam Cogan
SSW

Month 2008

Applies to: Microsoft Office SharePoint Server 2007, Visual Studio 2008, Visual Studio 2008 extensions for Windows SharePoint Services, SharePoint Designer 2007, Visual Studio 2008 Team Foundation Server.

Summary: This whitepaper describes how the best practices for Application Lifecycle Management were applied in the development of the SharePoint 2007 web content management project for WorleyParsons. It focuses on the team workflow, development environments, version control and deployment aspects of ALM.

# Contents

[Contents 3](#_Toc218522799)

[Introduction 4](#_Toc218522800)

[Team Roles 4](#_Toc218522801)

[Environment 5](#_Toc218522802)

[Stand-alone Development Environment 5](#_Toc218522803)

[Shared Development SharePoint Server 6](#_Toc218522804)

[Packaging Server 6](#_Toc218522805)

[Managing Environments with Virtualization 7](#_Toc218522806)

[Creating Template Image 8](#_Toc218522807)

[Creating a Template Image with Admin Console 11](#_Toc218522808)

[Deployment with solution packages 12](#_Toc218522809)

[Planning the Package 12](#_Toc218522810)

[Building the Package 12](#_Toc218522811)

[How to: Create SharePoint Projects 13](#_Toc218522812)

[How to: Extract Files from SharePoint Server 13](#_Toc218522813)

[How to: add Master Page & Page Layout 14](#_Toc218522814)

[How to: Add Style sheets and Images 15](#_Toc218522815)

[How to: Add Site Columns and Content Types 15](#_Toc218522816)

[How to: Generate Content Type Definition 16](#_Toc218522817)

[How to: add List Template 16](#_Toc218522818)

[How to: include custom assemblies 17](#_Toc218522819)

[Updating the Package 17](#_Toc218522820)

[Making a Build 18](#_Toc218522821)

[Deploying the Package 18](#_Toc218522822)

[Outcomes 19](#_Toc218522823)

[Conclusion 20](#_Toc218522824)

[Additional Resources 20](#_Toc218522825)

[About the Author 20](#_Toc218522826)

# Introduction

Microsoft Office SharePoint Server 2007 and Windows SharePoint Services 3.0 are feature rich application platforms that offer a suite of server capabilities such as content management, search and unified workflow. As an application platform, SharePoint application development should be regarded no different to traditional software development, and common methodologies should be applied.

A key discipline to any software development is Application Lifecycle Management (ALM), which is the coordination of all development activities that includes requirements gathering, design, modeling, development, testing, issue tracking and deployment.

This whitepaper describes how the best practices for Application Lifecycle Management were applied in the development of the SharePoint 2007 web content management project for WorleyParsons. It focuses on following aspects in ALM:

* Team workflow
* Development environments
* Version control
* Deployment strategies

# Team Roles

The development of a public facing web content management system for a multinational organization requires a variety of skills and resources. The following describes the roles in the team and the tools that were used by each role member:

* **Editors** are responsible for writing the content of the website. They are located around the world and use MOSS’s built-in rich text editor to compose articles, which then go through an approval workflow to get published to anonymous users.
* **Web Designers** are responsible for the creative design of the website and its implementation in HTML and CSS styling. They ensure the website is cross browser compatible by adhering to web standards and testing on multiple browsers. They use SharePoint Designer to modify Page Layouts and CSS, and to upload images.
* **Developers** are responsible for building custom web parts, creating site columns, content types, and base page layouts for the designers. They also create re-usable site definitions and deployment packages. They use Visual Studio 2008 Team System with Visual Studio extensions for Windows SharePoint Services.
* **Network Administrators** are responsible for the setup of SharePoint infrastructure – including farm, security and backup configuration. They are the owners of the production environment which cannot be accessed by other roles, and facilitate the deployment of versioned package to staging and production.

# Environment

In a development team, multiple developers and designers will be working concurrently on the same application. Decisions must be made to determine the levels of overhead and control required to keep SharePoint application environments stable and maintainable, while keeping the development teams productive.

The WorleyParsons project used the 'Patterns & Practices SharePoint Guidance' as the base for structuring the development environment with deviations in having a Shared Development SharePoint Server for the designers and a Packaging Server instead of Continuous Build Server or a Build Verification Test Server.

Figure 1 illustrates the overall environment and workflow:


Figure 1

The following describes the environments in figure 1:

* **Stand-alone development environment**. Each developer uses a stand-alone workstation. Each development workstation includes a Virtualized SharePoint Server.
* **Shared Development SharePoint Server.** Manages all designer artifacts using SharePoint’s built-in version control. It also provides developers a centralized server to configure site columns and content types.
* **Packaging Server.** Updates the solution package with designer artifacts extracted from the shared server. It commits any changed files into Team Foundation Server so it can be version controlled against the build. It builds a versioned solution package that gets deployed, in turn, to test, staging and production environment.
* **Source Control System: Team Foundation Server.** Manages and versions all source code. It also includes designer artifacts and XML definitions extracted from the Shared Development SharePoint Server so it can be tracked against a versioned solution package.
* **Test Environment.** Owned by the Developers, it is used for functional, performance and deployment testing.
* **Staging Environment.** Owned by the Network Administrators, it is used for User Acceptance Testing by the editors. Its content database is frequently restored from production so testing can be performed on relevant data.
* **Production Environment.** Owned by the Network Administrators, it includes two separate zones for editors and public (anonymous) access. The editors work in this environment and take advantage of SharePoint’s built-in version control and publish workflow. Alternatively content can be edited on staging, and 1-way ‘Content Deployment’ used to publish to production.

## Stand-alone Development Environment

Each developer should work in a stand-alone development environment to deploy and test their changes before committing to source control and sharing with the team. This prevents disruption to other team members when developing components such as custom web parts, event receivers, navigation providers, web control adapters, and HTTP modules.

The development of these components was done in Visual Studio 2008 with the project templates from Visual Studio extensions for Windows SharePoint Services (VSeWSS). This extension enables Visual Studio to build and to deploy solution package file onto the default SharePoint site on the local machine. The project is version controlled with Team Foundation Server.

## Shared Development SharePoint Server

A Shared Development SharePoint Server enables designers to quickly make and review their changes in SharePoint by using SharePoint Designer. This avoids the time consuming process of packaging and deployment. It also enables developers to create site columns and content types in a friendly built-in web interface as opposed to creating xml definitions in Visual Studio. These content types and site columns can also be used immediately in the page layout and web parts without deployment. Using this shared server, designer artifacts such as master pages, page layouts, XSL, CSS and images can be version controlled using SharePoint’s built-in functionality. These designer artifacts, site columns and content types can later be extracted and packaged in the packaging server.

## Packaging Server

This server is used to update the solution package with designer artifacts and content type definitions extracted from the Shared Development SharePoint Server. The updated files are then checked back into Team Foundation Server to allow all the files in the solution package to be labeled and version controlled.

Ideally, this package updating process is included in the build script. Then Microsoft Team Build Server can be installed on this server to automate the updating, build and packaging process. This build script can then be integrated with Continuous Integration to provide event-driven and daily builds. Unfortunately, in the WorleyParsons project, the package updating process includes steps that could not be automated due to the lack of tools available at the time.

# Managing Environments with Virtualization

The configuration of SharePoint Servers can be a complicated and time consuming process due to the number of server-side technologies SharePoint leverages. Virtualization of the development and test environment can provide the following benefits to the team and the development life cycle:

* Reduce setup time – New machines can be rapidly cloned so the team can be easily scaled in the development life cycle.
* Avoid conflicts – Isolated environments prevent conflict of server side technologies that may be used by other developments.
* Reduce cost – Existing hardware can be reused to host new SharePoint environments, workload can be distributed dynamically between development laptop, workstations and server.

In the WorleyParsons project, the following templates were used to streamline the setup of development and test environment:

* **Base Server Image.** This is used as base template for all virtual servers. It includes:
	+ Windows Server 2003 SP2 or 2008
	+ .NET Framework 3.5
* **SharePoint Server Image.** This is built from the Base Server Image and is used to build the Shared Development SharePoint Server and the test servers. Development tools should be excluded from this image because it needs to mimic the production servers. This image should include the following additional servers/services:
	+ SQL Server 2005 SP2 or 2008
	+ Windows SharePoint Services 3.0 SP1
	+ Microsoft Office SharePoint Server 2007 SP1
* **SharePoint Development Image.** This is built from the SharePoint Server Image and is used to build the Virtualized SharePoint servers used by the Stand-alone Development Workstations and the Packaging Server. This image should include the following development tools:
	+ Office SharePoint Designer 2007
	+ Visual Studio 2005 or 2008
		- Visual Studio 2005 will need Visual Studio 2005 Extension for .NET Framework 3.0 (Workflow Foundation) to work with workflow. This feature is built in to Visual Studio 2008.
		- [Download Workflow Foundation Extension for Visual Studio 2005](http://www.microsoft.com/downloads/details.aspx?familyid=5D61409E-1FA3-48CF-8023-E8F38E709BA6&displaylang=en)
	+ Visual Studio extensions for Windows SharePoint Services
		- [VSEWSS 1.1 for Visual Studio 2005](http://www.microsoft.com/downloads/details.aspx?FamilyID=3e1dcccd-1cca-433a-bb4d-97b96bf7ab63&DisplayLang=en)
		- [VSEWSS 1.2 for Visual Studio 2008](http://www.microsoft.com/downloads/details.aspx?FamilyID=7bf65b28-06e2-4e87-9bad-086e32185e68&displaylang=en)
	+ Internet Explorer Developer Toolbar
		- [Download the IE Developer Toolbar](http://www.microsoft.com/downloads/details.aspx?FamilyID=e59c3964-672d-4511-bb3e-2d5e1db91038&DisplayLang=en)
	+ Mozilla Firefox with Firebug add-on to test for cross browser compatibility
		- [Download Firebug for Firefox](https://addons.mozilla.org/en-US/firefox/addon/1843)

## Creating Template Image

A template image is a virtual hard disk (VHD) of an existing virtual machine with all the required components installed and the system prepared for cloning. The cloning process makes a binary duplicate of the template image for the new cloned machine. This enables rapid deployment of new environments as machines can be created with a simple copy instead of executing complex installers from various media.

A template image needs to be prepared with Microsoft sysprep tool before it can be cloned. This is because copying of the VHD also duplicates system specific information such as security identifier (SIDs) associated with the computer. This can cause conflicts if multiple cloned machines are hosted on the same network. Sysprep works by removing all machine specific settings from the template image, so it can be safely cloned.

Sysprep tool must be used with care when preparing a SharePoint template image, as SharePoint configuration contains machine specific settings. These settings are not supported by sysprep and running sysprep will leave the configuration invalid and the installation inoperable.

The recommended approach is to install all SharePoint components and sysprep the machine before configuring SharePoint. The image can then be used as template to safely clone new machines. SharePoint configuration can then be completed on the new machine. The configuration can be done either manually by running SharePoint Configuration Wizard (psconfigui) or automated by an installation script using psconfig.

Paul Horsfall has a blog that describes the scripts needed to automate the configuration of Microsoft Office SharePoint Server in conjunction with sysprep: <http://paulhorsfall.co.uk/archive/2007/05/21/SharePoint-Sysprep-and-SQL.aspx>

Overview of steps taken to create template images used in SharePoint development:

1. Create Virtual Machine with virtualization tool such as Hyper-V or Virtual PC
2. Install Windows Server 2003 or 2008 with the latest service pack and the following configuration:
	1. Leave the machine in workgroup, do not join domain
	2. Leave the administrator password as blank
3. Optional – the image can be cloned and sysprep-ed at this point to create the base server image
4. Add the following Server Roles:
	1. Application Server
	2. Web Server (IIS)
5. Install SQL Server 2005 or SQL Server 2008
	1. Note: full SQL Server installation improves performance and enables the development and testing of integrated reports
6. Install Microsoft Office SharePoint Server 2007 with SP1
	1. Use the Complete Installation option so it can be configured to use the full SQL Server 2005 or 2008 installation
	2. Do not run the SharePoint Configuration Wizard before sysprep
7. The image should be cloned and sysprep-ed to create the SharePoint Server Image
8. Install Office SharePoint Designer 2007
9. Install Visual Studio 2005 or 2008 Team System
10. Install Visual Studio extensions for Windows SharePoint Services
11. Install Mozilla Firefox with Firebug add-on to test for cross browser compatibility
12. Image should be cloned and sysprep-ed to create the SharePoint Development Image

#### How to sysprep for Windows Server 2003

The following describes the steps taken to sysprep a Windows Server 2003 image:

1. Download the deploy.cab for the sysprep tool
	* For Windows Server 2003, the cab file is located in the original media at:
	\Support\Tools\Deploy.cab
	* For Windows Server 2003 sp1, it can be downloaded from:
	<http://www.microsoft.com/downloads/details.aspx?FamilyId=A34EDCF2-EBFD-4F99-BBC4-E93154C332D6&displaylang=en>
	* For Windows Server 2003 sp2, it can be downloaded from: <http://www.microsoft.com/downloads/details.aspx?familyid=93F20BB1-97AA-4356-8B43-9584B7E72556&displaylang=en>
	* Note: the downloaded hot fix for SP1 and SP2 will install the cab file to: C:\windows\system32\deploy.cab
2. Copy the content of the deploy.cab into c:\sysprep folder
3. Run the c:\sysprep\setupmgr.exe with the following selection, the rest can be left on default:
	1. Select "Create new" for answer file
	2. Select "sysprep setup" for answer file type
	3. Select the correct Windows Server edition
	4. Select "Yes, fully automate the installation"
	5. Enter user name and organization
	6. Enter product key
	7. Select "Automatically generate computer name"
	8. Add any "Run Once Commands". These are commands that are executed the first time user logs in and are used to trigger manual configuration such as SharePoint Configuration Wizard (psconfigui.exe).
	9. Add any "Additional Commands"
	10. Save the answer file to: C:\sysprep\sysprep.inf
4. Run c:\sysprep\sysprep.exe with the following selection:
	1. Check "Don’t reset grace period for activation"
	2. Uncheck "Use Mini-Setup"
	3. Uncheck "Don’t regenerate security identifiers"
	4. Select "Shut down" for shutdown mode
	5. Click "Reseal"
	6. Click OK when asked to regenerate SIDs
5. System will be prepared and automatically shutdown. Do not restart the image, as it will undo the sysprep process.

#### How to sysprep for Windows Server 2008

For Windows 2008, the sysprep tool is included as part of the Windows installation. It is located at c:\windows\system32\sysprep\sysprep.exe. However, the setup manager used to create the answer file is replaced by Windows System Image Manager which is part of Windows Automation Kit. This can be downloaded from Microsoft Download Center at <http://www.microsoft.com/downloads/details.aspx?FamilyID=94bb6e34-d890-4932-81a5-5b50c657de08&DisplayLang=en>.

The following describes the steps taken to sysprep a Windows Server 2008 image:

1. Download Windows Automation Kit
2. Install and run Windows System Image Manager
3. Select the Window image or catalog file for your edition of Windows Server 2008
	1. Note: catalog file can be found in the Windows Server 2008 DVD under the Sources directory
4. Configure the Answer file tree with the following settings:
	1. Name, Organization Name, Product Key
	2. Enable automatic generation of Computer Name
	3. Add any Run once or additional commands
5. Save the answer file to:

c:\windows\system32\sysprep\unattended.xml

1. Run sysprep with the following command line:

sysprep /generalize /oobe /shutdown /unattend:sysprep.xml

For more information, see: <http://blogs.technet.com/askcore/archive/2008/10/31/automating-the-oobe-process-during-windows-server-2008-sysprep-mini-setup.aspx>

**Managing Virtual Machines with System Center**

System Center Virtual Machine Manager (SCVMM) is an enterprise level solution that provides centralized administration of virtual machine infrastructure. It benefits the SharePoint development life-cycle by enabling developers to provision their own Virtual Machines from a template in the virtual machine library.

A common problem with Hyper-V and Virtual PC is the lack of built-in support for managing template images. The cloning process in these solutions requires users to be disciplined in the management of the VHD files. Template VHD files are marked as read-only to prevent accidental changes. It is then copied to a new location and used to create the new Virtual Machine. This process is error prone and can be costly to the team if developers accidentally corrupt the template VHD.

SCVMM provides an Admin Console and a Self-Servicing Web Portal for Network Administrators and Developers to use respectively. The Admin Console is used to centrally manage all virtual hosts and virtual machines. It can also create a template image from an existing VHD using the Sysprep tool. The Self-Servicing Web Portal is used by the developers to provision new SharePoint machines for the development and test environments.

The following will describes the steps taken to create and manage template image with SCVMM.

## Creating a Template Image with Admin Console

SCVMM can create templates from existing virtual machines or VHD files, it wraps around the sysprep tool, so that the image does not need to be syspreped manually before adding to the template. However, all the sysprep limitations still apply so the image should be added before SharePoint is configured.

The following describes how a VHD created in Hyper-V can be added into SCVMM as a template:

1. Open SCVMM Admin Console
2. Open MSSCVMMLibrary with Windows Explorer and copy the VHD

3. Refresh the Library to show the VHD in the console
4. Go to Actions > Library Actions > New Template

5. Select Use an existing template or a virtual hard disk stored in the library

6. Enter template name, owner and description. Owner determines who can access this template in the self-servicing web portal. Setting it to the developer group will allow all developers to access this template.

7. Configure the hardware, allocate 1024 to 2048 GB of RAM

8. Configure the Guest Operating System with the following detail:
	1. Company name
	2. Admin password
	3. Product key for the Operating System
	4. Time zone
	5. Operating System type – this determine how the image is Sysprep
	Domain setting - leave the SharePoint Servers off the domain so it can be added if needed.
	6. Answer file – a Sysprep.inf file for Windows XP, Windows Server 2000 or Windows Server 2003 or a unattended.xml for Windows Vista and Windows Server 2008 can be supplied to provide additional configuration such as the automated configuration of SharePoint.
	7. Command line – additional commands that will be run the first time user login. To run the SharePoint Configuration Wizard the following command can be added:
	%COMMONPROGRAMFILES%\Microsoft Shared\Web Server Extensions\12\BIN\psconfigui.exe
9. Click OK
10. SCVMM will now prepare the VHD for the template process

#### Creating Virtual Machines with the Self-Servicing Web Portal

1. Login to the SCVMM web portal

2. Click New Computer
3. Select the SharePoint 2007 Template and enter the system properties:

4. Wait for the machine to get created:

5. Login to the new Virtual Machine to configure the SharePoint

# Deployment with solution packages

Solution packages were introduced in Windows SharePoint Services 3.0 to provide a way to bundle all the components for extending Windows SharePoint Services. The solution package uses a CAB based format with a WSP file extension and is a reusable package that can be deployed across web servers in a web farm. It can contain a set of features, site definitions and assemblies, and it can be enabled or disabled individually on the sites.

This section details how the WorleyParsons project was planned, built, updated and deployed using the solution package. This enables the application to be easily deployed to different environments in the application development life-cycle.

## Planning the Package

In the WorleyParsons project, two SharePoint projects were required, one for the custom features and another for the site definition. Ideally, both can be included in a single SharePoint project and deployed in one solution package. However, custom feature receivers interfere with the built-in features generated by VSeWSS for the site definition, so the site definition had to be separated from the main solution package.

An overview of the projects and features in the solution package is shown below:

* **WorleyParsons.Features** project contains the following features:
	+ **MasterPages**, **PageLayouts**, **Images** and **Styles** features contain customization for the look and feel of the site
	+ **SiteColumns** feature contains customization for additional fields and types that are required for the Page Layouts
	+ **WebParts** feature contains customized web parts in the gallery
	+ **ListTemplates** feature contains lists to be used by web parts
	+ **WebContentTypeBinding** feature contains bindings to associate libraries with content types
* **WorleyParsons.SiteDefinition** project contains a single feature:
	+ **SiteDefinition** contains the site templates for users to create new sites

Features were created for each logical component so they can be tested individually in test environment.

## Building the Package

The solution package can be created using Visual Studio extensions for Windows SharePoint Services. VSeWSS provides project templates that can be used to build the .wsp file in Visual Studio. It also provides a WSP view that can be used to configure solution package structure. Manifest and feature xml files that define the solution package are configured in the XML editor. The schema for WSS XML files can be added to Visual Studio to enable validation and intellisense.

Steps to enable schema validation and intellisense in Visual Studio 2005:

1. Copy: %ProgramFiles%\Common Files\Microsoft Shared\web server extensions\12\TEMPLATE\XML\wss.xsd
2. To: %ProgramFiles%\Microsoft Visual Studio 9.0\Xml\Schemas

And in Visual Studio 2008:

1. Open the manifest (or any xml) file to enable the XML menu
2. Click XML menu > Schemas…
3. Add %ProgramFiles%\Common Files\Microsoft Shared\web server extensions\12\TEMPLATE\XML\wss.xsd

The following topics will provide details on how the solution package is created, it includes:

1. Creating the SharePoint project
2. Extracting designer artifacts from Shared Development SharePoint Server
3. Add master page and page layout
4. Add styles and images
5. Add site columns and content types
6. Add list template
7. Add custom assemblies

### How to: Create SharePoint Projects

1. Create new project with the SharePoint > Empty template

2. Add new SharePoint Modules

3. Review the package with WSP View

4. Build and deploy to http://localhost for testing


### How to: Extract Files from SharePoint Server

Designer artifacts that were developed in SharePoint Designer are stored in the content database on the SharePoint Server. These files need to be extracted from the Shared Development SharePoint Server so they can be included in the solution package.

There are two ways to extract these files,

1. Use Web-based Distributed Authoring and Versioning (WebDAV) protocol to download the files with Windows Explorer.
2. Use or modify the STSADMExtractFiles sample from Michael Washam to download the files. This option allows the extraction process to be automated using a batch script.

During the development of the WorleyParsons project the first option was used since the stsadm extraction tool had not been released.

#### Using WebDAV

1. Browse to the Master Page Gallery
<http://servername/_catalogs/masterpage/Forms/AllItems.aspx>
2. Select Actions > Open with Windows Explorer

3. Copy files from Windows Explorer into the package project

#### Using STSADMExtractFiles

1. Download and install the tool from:
<http://blogs.msdn.com/mwasham/archive/2008/11/20/how-to-extract-aspx-files-out-of-a-sharepoint-content-database.aspx>
2. Run the following batch script:
stsadm -o savefromfolder -weburl <http://servername>
–localworkingdirectory C:\Projects\ProjectFolder\MasterPages
-starturl /\_catalogs/masterpage

### How to: add Master Page & Page Layout

1. Add new Module named as “MasterPages”
2. Extract master pages and page layouts from: /\_catalogs/masterpage
3. Add files to: %ProjectFolder%\MasterPages
4. Include files into the project
5. Add preview images to %ProjectFolder%\MasterPages\Previews folder. This allows administrators to preview the layout of the site in site settings. The recommended image size is 216 x 162 pixels.
6. Open %ProjectFolder%\MasterPages\Module.xml
7. Add the following attributes to the Module element:

<Module Name="MasterPages" List="116" Url="\_catalogs/masterpage">

1. Add the following File element for any new master pages:

<File
 Path="[MasterPageName].master"
 Url="[MasterPageName].master"
 IgnoreIfAlreadyExists="TRUE"
 Type="GhostableInLibrary" >
 <Property
 Name="ContentType"
 Value="$Resources:cmscore,contenttype\_masterpage\_name;" />
 <Property
 Name="MasterPageDescription"
 Value="[Your MasterPage Description]" />
 <Property
 Name="PublishingPreviewImage"
 Value= "~SiteCollection/\_catalogs/masterpage/$Resources:core,Culture;/Preview Images/[MasterPageName]Preview.gif, ~SiteCollection/\_catalogs/masterpage/$Resources:core,Culture;/Preview Images/[MasterPageName]Preview.gif"></Property>
</File>

**Note:** For more information on updating module definition, see: [How to: Provision a File](http://msdn.microsoft.com/en-us/library/ms441170.aspx).

### How to: Add Style sheets and Images

Designer artifacts can be found in either Style Library or the Image Library depending on its usage. Style library contains Cascading Style Sheets (CSS) and images that control the overall site layout, and Image library contains re-usable artworks that are used by the editors on individual pages.

#### To package files for the style library:

1. Add new Module named as “Styles"
2. Extract files from: /Style Library/en-us/ProjectName
3. Add files into: %ProjectFolder%\Styles
4. Open %ProjectFolder%\Styles\Module.xml
5. Add the following attribute to the Module element:
<Module

Name="Styles"

Url="Style Library/en-us/ProjectName"

RootWebOnly="TRUE">

1. Add the following File element for any new files:

<File

Url="[Your File Name]"

Path="[Your File Name]"

Type="GhostableInLibrary" />

#### To package files for the image library:

1. Add new Module named as “Images”
2. Extract files from: /Images
3. Add files to: %ProjectFolder%\Images
4. Open %ProjectFolder%\Images\Module.xml
5. Add the following attribute to the Module element

<Module

Name="Images"

Url="SiteCollectionImages"

RootWebOnly="TRUE">

1. Add the following File elements for any new files:

<File
 Url="[Your File Name]"
 Path="[Your File Name]"
 Type="GhostableInLibrary" />

### How to: Add Site Columns and Content Types

Site columns and content types can be deployed by including a content type definition file in the solution package. However, Windows SharePoint Services does not track all changes made to the content type definition. Therefore, there is no built-in method for pushing down all the changes made to site content types to the child content types used by existing lists. This is problem is briefly described in [Updating Content Types](http://msdn.microsoft.com/en-us/library/aa543504.aspx) on msdn and it poses a serious challenge to the redeployment scenario.

One solution to this problem is to use the SharePoint object model to force the propagation of the content type changes to existing lists. This solution, originally described by Søren Nielsen on his [blog](http://soerennielsen.wordpress.com/2007/09/11/propagate-site-content-types-to-list-content-types), iterates through all existing lists and creates, deletes, or updates fields based on the site content type. The solution was later extended by Gary Lapointe in his [blog](http://stsadm.blogspot.com/2008/05/propagate-content-type-changes.html) which allows the propagation to be called in feature receiver. In the WorleyParsons project, this code was further extended to support propagation by group name on content types. There are some limitations to the current implementation:

1. It only updates fields, it does not cover other aspects such as policies and document information panel (DIPs)
2. It does not handle read-only or sealed content type
3. It can be slow to deploy on a large SharePoint site

### How to: Generate Content Type Definition

Content type definition can be manually written in Visual Studio or it can be generated from Shared Development SharePoint Server using MossStsadmWcmCommand tool from Andrew Connell: <http://www.andrewconnell.com/blog/articles/MossStsadmWcmCommands.aspx>.

To generate site columns:

stsadm -o GenSiteColumnsXml -URL [http://localhost](http://servername) -outputfile "C:\SiteColumns.xml" -groupfilter "application name”

And to generate content types:

stsadm -o GenSiteContentTypesXml -URL http://localhost -outputfile "C:\ContentTypes.xml" -groupfilter "application name"

**Note**: These commands will generate XML based on its current properties on the server and may contain fields that are not included in the schema, fields such as Version, PITarget, PrimaryPITarget, PIAttribute, PrimaryPIAttribute and Aggregation.

**Note:** Group all site columns and content types to make this command work more effectively. The group filter can then be reused in the custom feature receiver to find content types to propagate.

### How to: add List Template

List definition can be exported with solution generator that comes as part of VSeWSS. This approach is safer than exporting list as STP because it handles related lists. The solution generator can be used to export site definition or list definition. The following describes how the solution generator can be used to export list definition for existing solution package project:

1. Launch ‘SharePoint Solution Generator 2008’ from the Start menu
2. Select List Definition

3. Select the site url:

4. Select the list to export:

5. Enter the project name and export location
6. Browse to the export directory

7. Copy folder with the list name into the existing SharePoint project


### How to: include custom assemblies

1. Add a project reference to the custom project in the solution package project
2. Include the DLL in the bin directory (otherwise it will not be packaged by VSeWSS): 

## Updating the Package

The solution package needs to be frequently updated with files from the source control and the Shared Development SharePoint Server. This update process consists of the following areas:

1. Designer artifact changes
2. Content type changes

#### Updating Designer Artifacts

The key to streamlining this update process is to keep the files, in the development environment, the way it should be deployed in production. This way automation can be set up around the Module.xml file to determine where the file should be downloaded from in the Shared Development SharePoint Server.

To determine the source URL of the file on the SharePoint Server, the URL on the File element can be concatenated with the URL on the Module element. The same can be applied to the Path of the file in the solution package by concatenating the Path between the two elements.

For example, with the following module.xml:

<?xml version="1.0" encoding="utf-8"?>

 <Elements
 Id="b4b87ed0-7556-42be-ad85-581edcdaa351"
 xmlns="http://schemas.microsoft.com/sharepoint/">
 <Module Name="Images" Url="SiteCollectionImages"
 Path="" RootWebOnly="TRUE">
 <File Url="logo.gif" Path="Images\logo.gif" Type="GhostableInLibrary" />
 </Module>
</Elements>

The relative URL of the file on the SharePoint server will be /SiteCollectionImages/logo.gif, and the local file path relative to the Module.xml will be Images\logo.gif.

In the WorleyParsons project, the stsadm extraction sample from Michael Washam was not available due to timing. The automation used to update package was setup to download files from Shared Development SharePoint Server based on the Module.xml. Downloading files over http means the process cannot check for modification and must update all files indiscriminately. It also means files such as master pages and page layout cannot be updated via automation.

Technically this is now fully automatable with the stsadm extraction sample from Michael Washam. One possible way is to use XSL to transform Module.xml into a batch script that calls the extraction sample to update the local files in the solution package.

#### Updating Content Type Changes

This is a manual process using the [MossStsadmWcmCommand](http://www.andrewconnell.com/blog/articles/MossStsadmWcmCommands.aspx) tool from Andrew Connell. Technically, this can be automated by transforming the output XML with an XSL to remove elements and attributes that do not belong to the Content Type Definition File schema. However, this update rarely occurs during the development life-cycle of the WorleyParsons project and it remains a manual process.

## Making a Release Build

Once the project is updated, the package can be build by running Deploy Solution in Visual Studio. However, developers should take care when making a release build for test and production environment. Process needs to be in place to ensure each build is correctly versioned so changes can be traced back the source code. To achieve this, the following steps should be taken when making a release build:

1. Update version and check-in changes
2. Get latest source code
3. Build the project
4. Label source code with the version
5. Rename the package to include the version
6. Copy the package to a shared location

Build automation can be used to minimize errors in these tedious and repetitive tasks. The following [article](http://msdn.microsoft.com/en-us/library/dd206936.aspx) in patterns & practices SharePoint Guidance describes how Microsoft Team Build can be configured to automate the build of a SharePoint solution package.

## Deploying the Package

Once the solution package is built it can be deployed and activated with Central Administration or stsadm command line. There are two options for upgrading solution package on the SharePoint server:

1. Upgrade solution – this will update any files in the file system. However, it will not reactivate any existing features. Which means upgrade code in the feature receiver will not be executed. These features can be force activated individually on the site. For other operations not supported by upgradesolution, see:
<http://msdn.microsoft.com/en-us/library/aa543659.aspx>.
2. Reinstall – this will take longer as the solution is uninstalled and then reinstalled. All features will be reactivated and upgrade code triggered.

In the WorleyParsons project, the reinstall option was used to trigger feature receiver to propagate site column’s and content type’s changes.

The deployment process can be simplified by using the setup batch file generated by VSeWSS in the output directory. The setup script calls stsadm to install or uninstall solution and activate or deactivate features (it does not support upgradesolution). It accepts a range of arguments to configure the target server, target site and the installation mode. By default the setup script will install the solution package and activate all features under <http://localhost:80>.

The following describe the command line usage and options of the setup script:

setup.bat [/install or /uninstall] [/weburl <url>] [/siteurl <url>] [/help]

|  |  |
| --- | --- |
| **/install or /uninstall** | Install Solution package (.wsp) to the SharePoint server or uninstall the solution package from the SharePoint server.Default value: install |
| **/weburl** | Specify a web url of the SharePoint server. Default value: http://localhost |
| **/siteurl** | Specify a site url of the SharePoint server. Default value: http://localhost |

Once the application is deployed and tested, the solution package is zipped up with the setup script and handed to a network administrator for deployment to staging and production environment.

# Outcomes

In the WorleyParsons project, the development and deployment aspects of ALM were implemented successfully, while the build and packaging aspects could be improved and streamlined through automation.

The key outcomes of this ALM process were:

* Streamlined development environment
	+ Allows designer to work directly with SharePoint using SharePoint Designer for a WYSIWYG experience rather than modifying package and going through deployment process
* Virtualization of development and test environment
	+ Isolated environment for development and testing
	+ Allows simple and fast scaling of development team
	+ Reduces network administration demands by using Self-Servicing Web Portals in System Center Virtual Machine Manager
* Deployment through solution packages
	+ Version control on these packages using Visual Studio Team System
	+ Simple deployment process can be easily repeated between different environments

Aspects that can be improved:

* Automating the build and the packaging
	+ Updating content types and page layouts remained as manual steps that can be automated
	+ This will allow the packaging process to be included in the Team Build
	+ This improvement would help reduce the time spent building packages
* Use of multiple solution packages
	+ This is because feature receiver have conflicts with the built-in feature
	+ Having a single solution package will simplify the deployment and version management

# Conclusion

A successful ALM implementation in SharePoint development can bring a host of benefits to the project by improving team productivity, product quality and on time delivery. This can easily be achieved with a bit of planning and by following Microsoft SharePoint Guidance.

# Additional Resources

* [The SharePoint Developer Introduction for .NET Developers](http://www.microsoft.com/click/SharePointDeveloper)
* [Patterns & practices: SharePoint Guidance](http://www.microsoft.com/spg)
* [Best Practices Resource Center for SharePoint Server 2007](http://technet.microsoft.com/en-us/office/sharepointserver/bb736746.aspx)
* [Overview of Team Foundation Build](http://msdn.microsoft.com/en-us/library/ms181710%28VS.80%29.aspx)
* [Automating Solution Package Creation for Windows SharePoint Services by Using MSBuild](http://msdn.microsoft.com/en-us/library/cc441431.aspx)
* [Using Team Foundation Server to Develop Custom SharePoint Products and Technologies Applications](http://msdn.microsoft.com/en-us/library/cc948982.aspx)
* <http://blogs.msdn.com/vesku/archive/2008/07/29/continuous-integration-in-moss-development-using-tfs.aspx>

# About the Author

Mark Liu is a Senior Software Architect at SSW, a Microsoft Certified Partner specializing in .NET Solutions. He is specialized in developing solutions with C#, VB.NET, ASP.NET, Web Services, SQL Server and Microsoft Office Integration.

Jonny Trees is a Senior Software Architect at SSW, a Microsoft Certified Gold Partner specializing in database driven .NET Solutions. Jonny specializes in building ASP.NET web applications with SQL Server backend, Microsoft Dynamics CRM solutions and SharePoint 2007.



The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering this document or the subject matter included in this document. The furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

© 2008 Microsoft Corporation. All rights reserved.