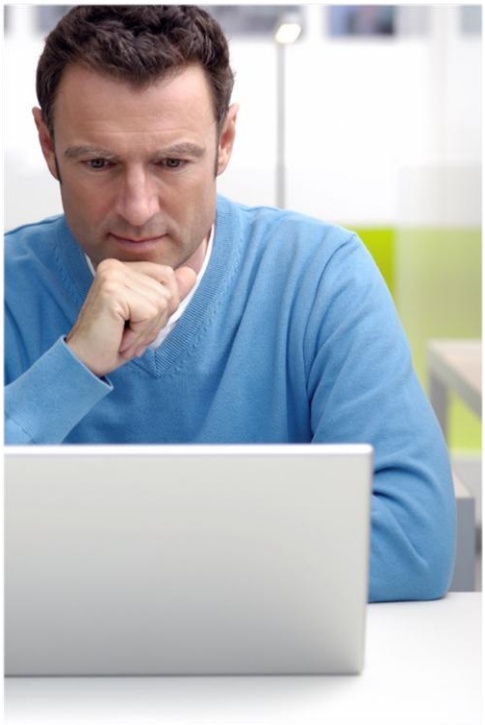
Best Practices:
Team Development with Sitecore



Too many cooks spoil the broth

That proverb has been with us for over 500 years, yet it has probably never been more truthfully invoked than in the case of Team Development. Without careful management and strict adherence to carefully agreed-upon methodology it becomes very easy for a team of cooks to spoil the broth of code.

Most developers learn to code working on projects individually. They do all the planning, the conceptualizing, the design, and the development in a self-contained development environment. As such, they become accustomed to having complete control over all assets, where they're stored, naming conventions, and the entire development process from end-to-end. It is only when they join a development team that they first realize the need to align themself with other talented coders and designers.

What Changes in Team Development?

When coding alone on a project developers can do what most people do with any kind of project - have their own controlled "mess" with all the components of their project wherever they want them to be. Most developers, in fact most people, can innately keep track of their own mess and bring everything together eventually over the time it takes to get the project done.

But nobody can keep track of somebody else's mess, so when you set out to develop in team you're going to have to make some decisions and set some standards.

What Do We All Need to Keep Track Of?

"A place for everything, and everything in its place." Another old axiom, but totally applicable and valuable to the team that wants to develop together effectively and efficiently. So what are the key elements in the development process that everyone on the team is going to need to be aware of?

Documentation

Begin with the documentation generated before your team ever gets assembled or called in to develop. The fundamental concept behind the project, the design and comps generated during that process, and the functional specifications developed for you to work from need to be somewhere that everyone on the team can quickly and easily access. You'll ideally want to put them in a collaborative shared storage environment that will lend itself to easy consultation by any number of team members at any time.

Later, notes developed at team meetings, structures developed, and other documentation created during the lifecycle of the project will need a single repository that is convenient for all team members to access.

… when you set out to develop in team you're going to have to make some decisions and set some standards.

Data

While at the outset this may seem like a simple, obvious concept, today's content management structures have redefined "data" for developers. Now, many of the content elements you develop will be stored and managed as data. As such, this data must be readily and commonly available to all developers, and protocols must be in place to assure that any changes to the data structures are agreed upon by all team members and incorporated into their development work consistently.

Since a major portion of the development being done today ends up residing in a Content Managed environment, we must also consider best practices surrounding where and how we will store the various schema, scenarios, and other elements the selected Content Management System will introduce.

By utilizing Team Development for Sitecore, the manual steps involved in packaging are completely removed from the process, saving significant time and effort.

Code

Another obvious necessity, but the concerns here branch out in many directions as the complexity of the project rises. As library routines are developed how will all developers be made aware of the changes so they can call these routines, pass parameters to and from them, and otherwise interact with them properly? Where will they be stored so all team members can access them and use them? How will you avoid having one developer clobber another's changes?

Since this is team development, how will you keep track of how different modules interact with each other? How will you assure they do so in the most efficient manner possible?

Builds

At a given point we will want to draw together all the modules the team has developed and create a "build" of the entire application for testing and development purposes. We will need to be able to confirm that all stored modules are the most current and are consistent with one another. We will also need to be able to store each build so we can refer back to it later in the development process if we need to troubleshoot any errors or other weaknesses.

.

Who's Doing What?

Now that you're part of a team you have the opportunity to take advantage of multiple talents and skill sets. An important key to success is to very effectively distribute responsibilities within the team. You have two basic ways to distribute responsibility and work assignments in a team development environment.

The first is to give specific segments to each individual developer who becomes the only developer to work on those segments. This approach has several inherent weaknesses not the least of which is that each module will not benefit from the aggregated experience and talent of the team. Only one person will be familiar with each module which could delay the process should one of the team members need to be removed. The project will become a collection of individual efforts rather than the product of a team.

With a properly structured team development environment, each developer can be dynamically assigned and reassigned to work on various modules based on their strengths and experience. Every team member can have a hand in every module as required, which means that everyone on the team is familiar with most or all of the code making it far easier to remove and replace members if necessary. And with proper "check-in/check-out" procedures you'll assure that nobody "steps on" anybody else's code, inadvertently eliminating changes made previously by someone else on the team.

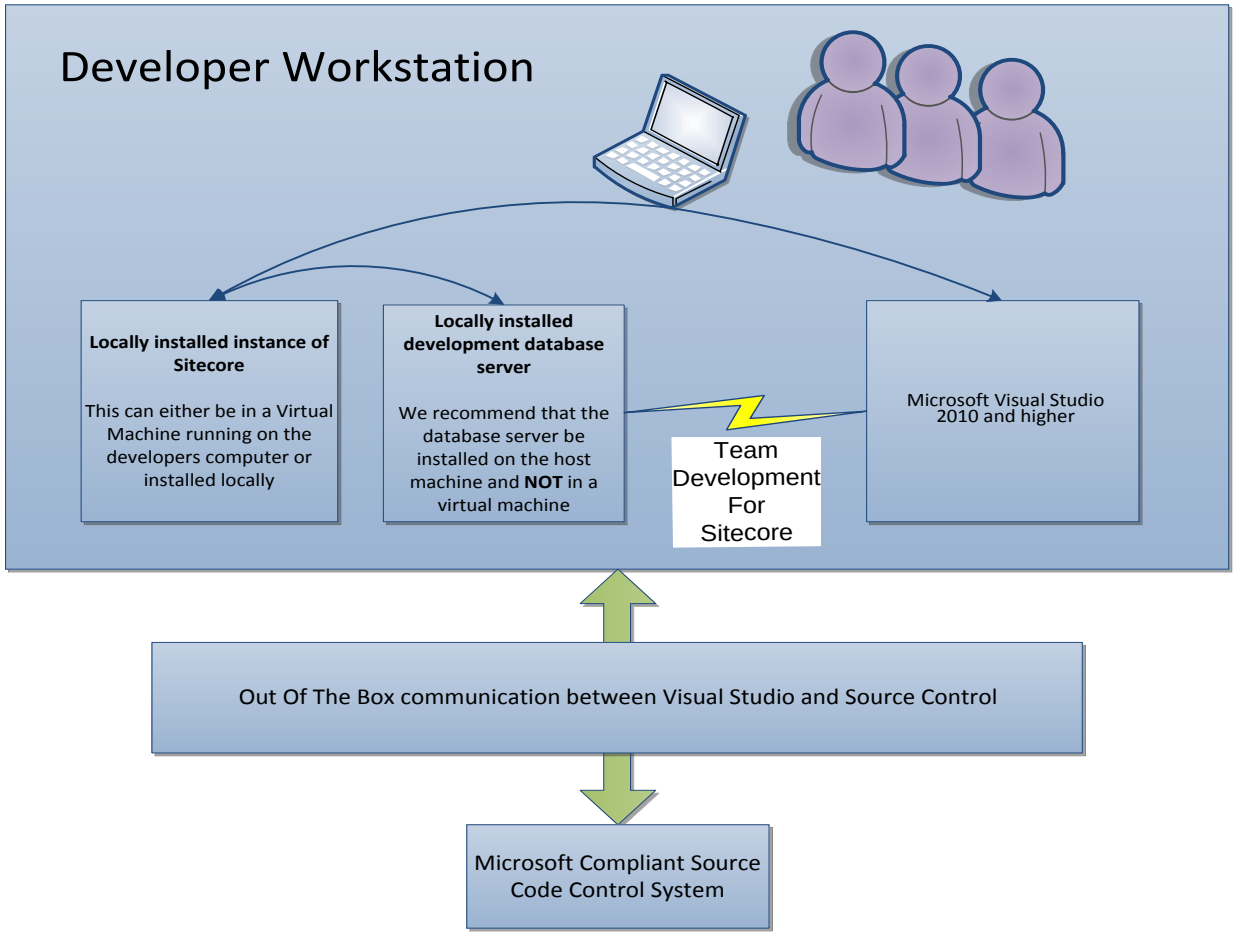
Recommendations for Best Practices in Sitecore Development

When developing on your own you probably had a server for the database and application engine, and your own personal computer for your development tools. In a team development environment you can look at 100 different projects and may see 100 completely different configurations for the development environment, but here are some "best practice guidelines" that will help you create the optimal development environment for each specific project.

The Developer's Work Environment

To avoid all the issues with a shared database, we recommend that developers work locally with all their own resources. On the developer's workstation there will be an instance of Sitecore installed, plus a development database. The Sitecore instance can be either installed directly on the developer's workstation or in a Virtual Machine hosted on the developer's workstation. We recommend that the development database be installed on the developer's workstation and not in a Virtual Machine. The Sitecore development environment can be enabled to talk to Microsoft Visual Studio (2010 and greater) which is also installed on the developers work station, via the Hedgehog Development Sitecore Connector©, which allows Team Development for Sitecore to directly manage Sitecore Items, Templates, Layouts, Sub layouts and Renderings.

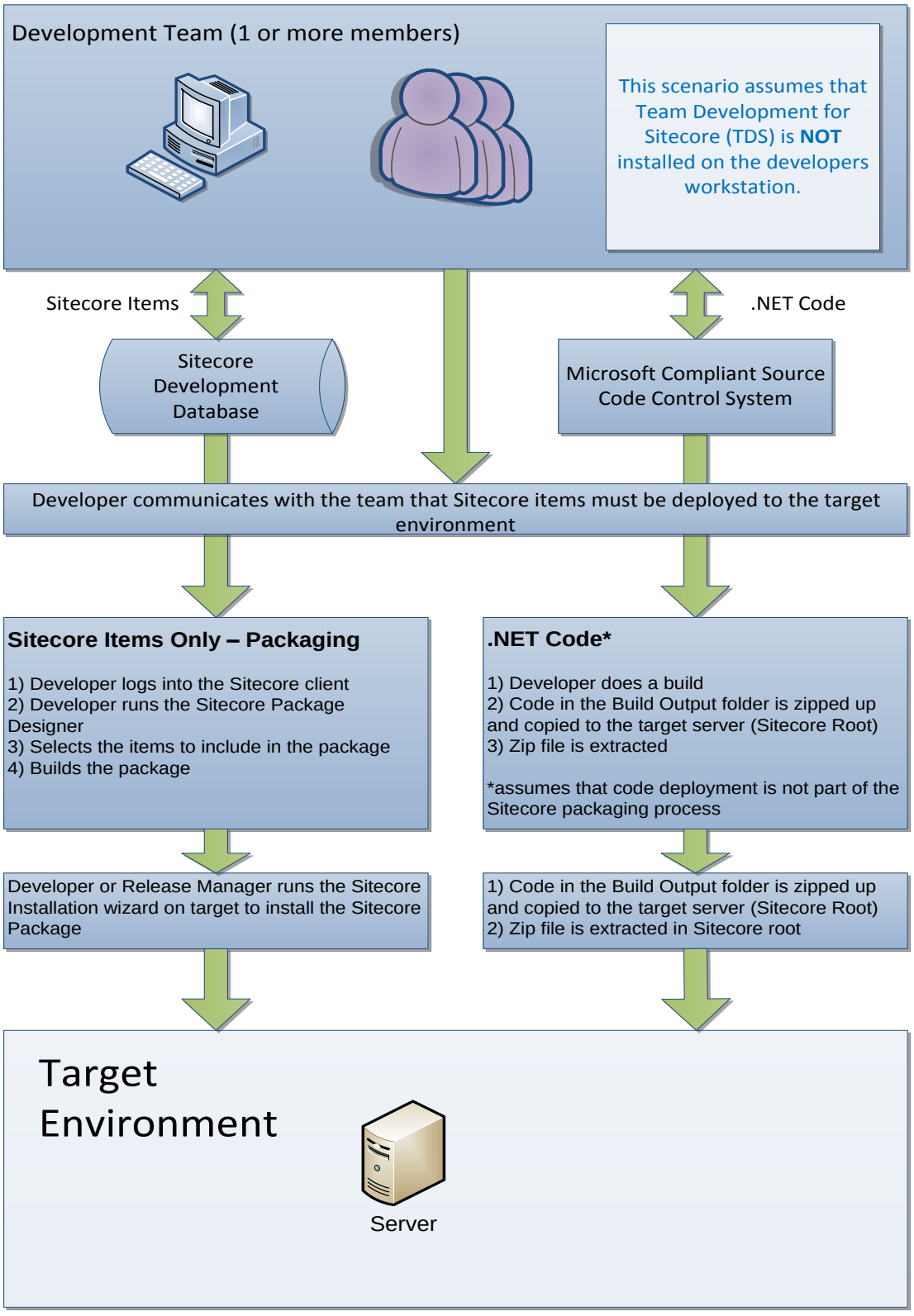
By bringing all of your Sitecore items into your Source Code Control System (SCCS), you have an authoritative source for your project. We like to say that your SCCS is "the single source of truth". If it has not been tested, and checked in, you don't want it working its way upstream to your production environment.



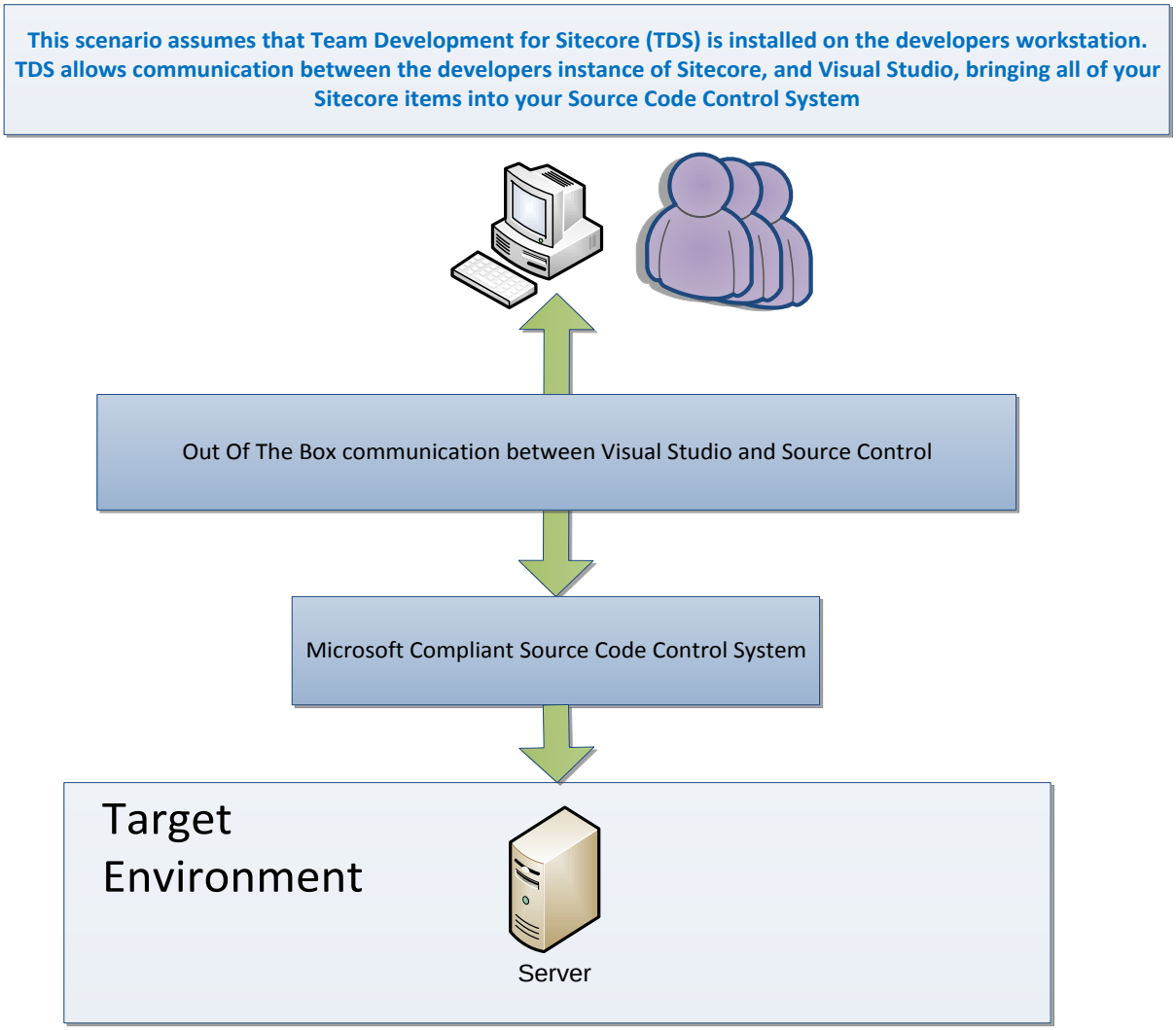
From Development to the Target Environment With and Without Team Development for Sitecore

Much of the significant complexity added when developing in a team is introduced when source components must be combined and a complete application built. The following is an illustration of how Team Development for Sitecore (TDS) mitigates much of that complexity restoring much of the consistency and reliability often lost in the process.

From Development to the Target Environment



From Development to the Target Environment with Team Development for Sitecore (TDS)



- By bringing all of your Sitecore items into your Source Code Control System (SCCS), you have an authoritative source for your project. We like to say that your SCCS is "the single source of truth". If it has not been tested, and checked in, you don't want it working its way upstream to your production environment
- With the TDS Deployment Property Manager you have complete control over what Sitecore items are included in the builds. For example if you have test content that the development team uses, you can simply flag that content to be not included, and it will be excluded from all builds and update packages
- With TDS you can have continuous integration to your target environment using any automated build utility that is compatible with MSBuild
- TDS can deploy directly to a running instance of Sitecore or create a Sitecore Update Package that can be manually moved to an instance, and installed using the Sitecore Update Package Wizard

About Team Development for Sitecore

Team Development for Sitecore (TDS) brings all of your Sitecore items into Visual Studio. Imagine having all the benefits of a source control management system fully integrated with your Sitecore development environment. Team Development for Sitecore (TDS) makes that a reality. Now you are able to take advantage of the value Visual Studio Team System (VSTS), Subversion, or many other Source Code Systems offer to you on your Sitecore based projects.

With Team Development for Sitecore you can stop worrying about dependencies and allow your developers to work with complete freedom, focusing on what they need to accomplish rather than worrying about how well it will play with changes others may be making.

Microsoft's Visual Studio Team System delivers some of the most highly regarded and highly productive development tools available. Team Development for Sitecore not only expands upon these tightly integrated, extensible lifecycle tools, it also facilitates the incorporation of Visual Studio Team System or Subversion with Sitecore delivering an unparalleled development environment for you and your team.

- Full integration with Visual Studio Team System work item tracking and reporting
- Project managers and testers can easily track and assign work items
- Testers can report bugs that are assigned to the developers via workflow

About Hedgehog Development

Hedgehog is a leading global provider of quality digital experiences. By combining our years of experience delivering outstanding Sitecore solutions with the impact of a cutting edge marketing innovation team, Hedgehog brings companies and their audiences together resulting in real business impact. Learn more about Hedgehog by visiting www.hhogdev.com.

For more information about Team Development for Sitecore
contact Hedgehog Development today at (631) 676-2186 ext. 450 or email sales@hhogdev.com.