JOIN OVER 100,000 PROS! Join Log In | Member Benefits

STICKYMINDS

Agile Testing DevOps Security AI

Private Developer Workspaces: Where the Development Process Meets SCM Process

ARTICLE

By Steve Berczuk , Brad Appleton Robert Cowham | June 15, 2006

SUMMARY

Software configuration management supports the delivery of application code in a reliable, repeatable manner. Having a CM process in place does nothing for the success of your organization unless you have mechanisms in place to develop application code reliably. Proper private workspace are a key element linking your SCM and your Development processes. In this article we discuss why they are important and how you can set up private code workspaces to help your team to be more effective.

"I regard it a basic programmer's right that I should be able to, at any time, check out all the source code and build it."

-Caroline Foster on the Extreme Programming Yahoo Groups list, February 2003.

Introduction

Being able to reliably create a version of your application that is consistent, regarless of where you are creating it is a key success factor for a project. All code starts in a developer workspace, and the more attention you pay to setting up the workspace, the better you will be in the long run. Private Workspace is one of the central patterns in the SCM Pattern Language. This article also follows up on our promise to fill in some of the details from our article The Importance of Building Earnestly.

Effective private workspaces give you the following advantages:

- Repeatability: Developers are building, testing and deploying many times a day. This gives you amble opportunity to find problems with the code and the deployment and install process before you unleash your code on customers (or Quality Assurance Teams). This repeatability leads to reproducibility
- Productivity: By providing a way for developers to control how and when they start

We use cookies to ensure you get the best experience on our website. Accept

Read More

integration issues before they make it into the repository, and improve the productivity of the team
- Progress: By enabling frequent integration and testing, you give developers more confidence in the codebase. This will enable them to feel more comfortable about integrating frequently, and moving forward in a regular fashion
- Accountability: Developing build and integration processes at the beginning of the process allows you to get the application to a point where you can demonstrate it to stakeholders earlier

While it may not be trivial to create private workspaces, the benefits allow you gain make it well worth the effort.

What Is a Workspace?
Before we go forward, let's define what a workspace is. A workspace is:

- Code and other files that you are working on
- Any locally built components, which you create from the source code (or retrieved from a repository in a controlled fashion)
- Third party components at the correct version for the project
- Resources that you need to run and test the application. For example, if your application updates a database, you should have a private database schema, or if your application uses more than one schema, a private copy of the database
- Any tools that you need. If the tools are the same for all versions of your app, they can be centrally installed, or they are be associated with a specific component

The key theme here is that you should have (in effect) a private copy of:

- Any resource that you write to. This includes both "soft" entities such as source code and database space, as well as "hard entities" such as network devices or application servers
- Any resource that might change over the course of the project so that you can control when to accept the change. This includes, among other things, code, libraries, and databases

You should be able to create a Private Workspace with a simple process not too different from:

- Install a version management client
- Check out the files for the appropriate version of the application
- Execute a script to build, configure and deploy your application

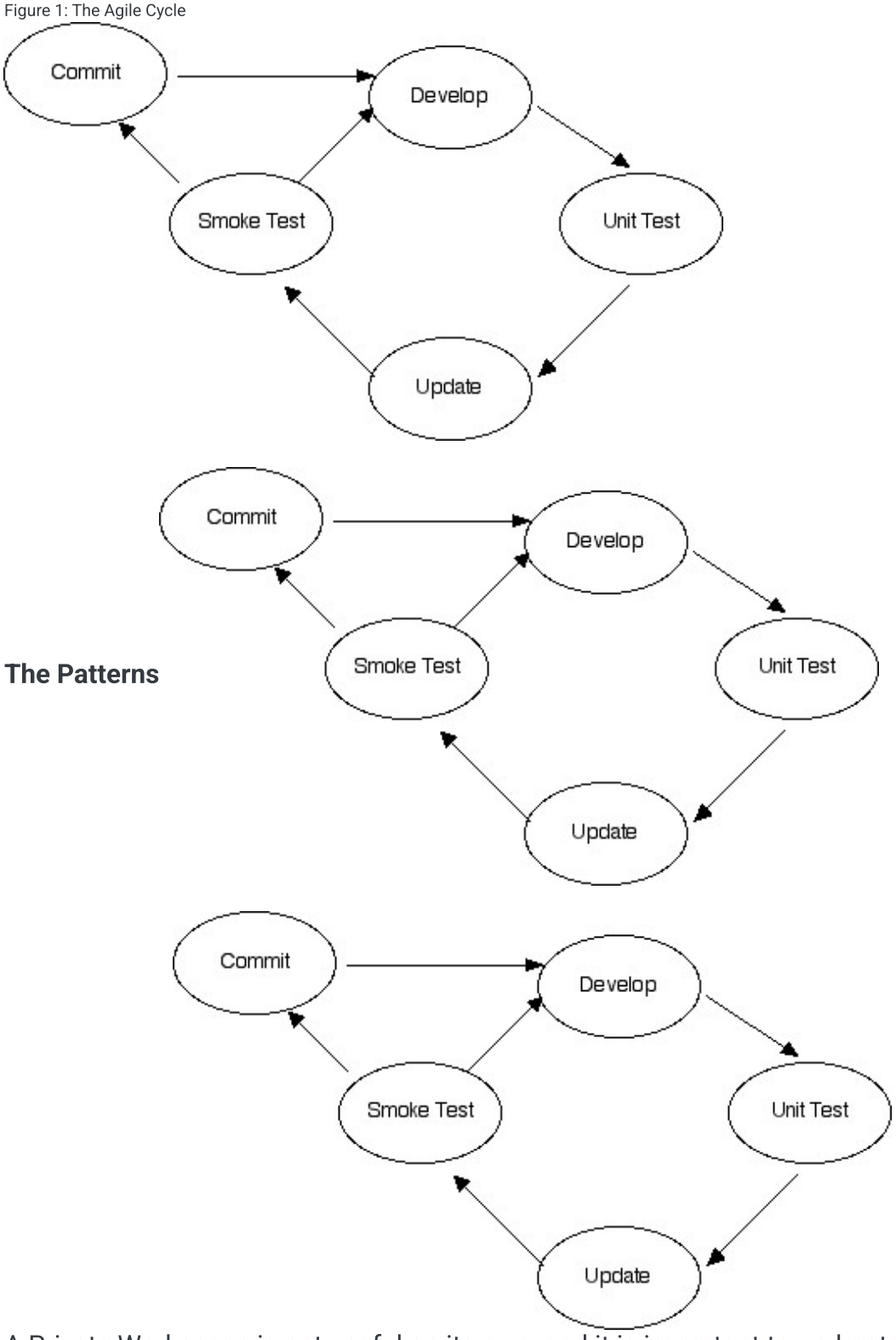
There may be some extra steps, to be sure, but you want to automate as many steps as possible to ensure that the process is reproducible. Allowing developers to create Private workspaces easily is a core element of the Agile Cycle.
The Agile Cycle
Agile software development relies on feedback. The simplest form of feedback is how tests execute. Figure 1 shows the steps that we repeat the following steps frequently:

- Code to develop the specified feature
- Unit Test as we Code

We use cookies to ensure you get the best experience on our website.

- If the Smoke test passes, and we want to commit our changes we do so, otherwise we repeat the cycle until we have a good build. In most cases you will want to commit changes frequently

Figure 1: The Agile Cycle



The Patterns

A Private Workspace is not useful on its own, and it is important to understand the patterns that define its environment. Figure 2 shows how the Private Workspace pattern

We use cookies to ensure you get the best experience on our website.

- Private Build, which allows you to verify that the system works
- Repository, which provides the tools that allow you to create the workspace
- Smoke Test, which gives you a way to check that your changes have not broken the build
- Unit Test, which allows you to test your specific changes in detail
- Integration Build, which provides a way to verify that all of the changes that were coded and tested in the private workspace really do integrate together
- Codeline Policy, which is how you communicate how the development process works

Private Workspace enables Active Development Line and relies on Repository, Integration Build and Private System build to work.

Arguments Against Private Workspaces

One item that seems to raise issues from time to time is the requirement for copies of the database schema. You can apply these rationales to any other resource issue. The common issues that arise are computing resources (machines to run the databases on), complexity of setup, and that databases are the province of a different team (the "Database Team"). While there may well be exceptions, in most cases these rationales are often not valid reasons to deny this important productivity improvement tool. Let's address these in turn.

The Cost of Resources

Cost of resources is often overstated. The first thing to consider is the cost of not having these resources in place. Consider the amount of effort you expend working around not having these resources. One major one is the coordination cost of having to time private changes to a shared schema with the work of others. You may have to spend time waiting for the resources, your test cases may disrupt the work of others (by design if you are working on an issue that could break the whole application, or by accident if you make a mistake). Developer and tester time is relatively expensive compared to hardware, and delays also can mean slipped schedules and missed market opportunities. Also, the hardware you need for developer testing need not be at the same price point as Production hardware. A desktop machine could host a number of developer databases, or developers could host the database on their machine.**Setup**

Another common argument is that it's just too hard to set up multiple databases. Like any other aspects of your application, you should be able to create a database in a reproducible manner. If it is costly in terms of time and resources to create a new database at a certain version, you need to fix your database deployment process. Having these private workspaces give you many opportunities to debug the database deployment process through practices.

Ownership

I know a fair amount about database design, but I am by no means an expert, and I

We use cookies to ensure you get the best experience on our website.

table change, I can write the DDL to create new columns (or new tables). I certainly want the people with database expertise to review my work (that's how we can learn), but I don't need them to spend time on all of my dead-ends. A key aspect of agile approaches is that there is a team with common of goals, but while a certain amount of division of labor is inevitable, there are may valid reasons for the team to cross boundaries.

Cautions
When your team has good Private workspace environments, you need to be sure that team members keep their workspaces in synch with the state of the codeline. The goal of Private Workspaces is to give team members small windows of time (and space) to work on tasks, not to isolate themselves for the long term.

Conclusion
Private Workspaces are a simple way to increase productivity. Setting up workspaces for your team may require a bit up-front work, but the increased rate of progress will more than compensate for the initial outlay of work. References & Further Reading

- **The Illusion of Control in Software Configuration Management**
- [SCM Patterns], Berczuk, S.P. and B. Appleton, *Software Configuration Management Patterns: Effective Teamwork, Practical Integration*. 2003, Boston, MA: Addison-Wesley.
- [Flow] Csikszentmihalyi, M., *Flow : The Psychology of Optimal Experience*. 1st ed. 1990, New York: Harper & Row. xii, 303 p.

Topics: configuration management,

We use cookies to ensure you get the best experience on our website.

ABOUT THE AUTHOR

Steve Berczuk

Steve Berczuk is a Principal Software Engineer with experience as a manager, Scrum Master and technical lead in Boston, MA. The author of *Software Configuration Management Patterns: Effective Teamwork, Practical Integration*, he is a recognized expert in software configuration management and agile software development. Steve is passionate about helping teams work effectively to produce quality software. He has an M.S. in operations research from Stanford University and an S.B. in Electrical Engineering from MIT, and is a Certified ScrumMaster. Contact Steve at steve@berczuk.com or visit berczuk.com and follow his blog at blog.berczuk.com.

Brad Appleton

Brad Appleton is an Enterprise Agile+DevOps leader, coach & manager, and seasoned DevOps/ALM/CM solution architect at a large Fortune 100 company. Currently he helps organizations and teams scale, adopt and apply lean/agile development methods and DevOps/ALM/CM practices and tools. He is co-author of *Software Configuration Management Patterns*, the Streamed-Lines branching patterns, and a former section editor for *The C++ Report*. You can read Brad's blog at blog.bradapp.net.

Robert Cowham

Robert Cowham has long been interested in software configuration management while retaining the attitude of a generalist with experience and skills in many aspects of software development. A regular presenter at conferences, he authored the Agile SCM column within the CM Journal together with Brad Appleton and Steve Berczuk. His day job is as Services Director for Square Mile Systems whose main focus is on skills and techniques for infrastructure configuration management and DCIM (Data Center Infrastructure Management) - applying configuration management principles to hardware documentation and implementation as well as mapping ITIL services to the underlying layers.

UPCOMING WEBINAR

We use cookies to ensure you get the best experience on our website.

LETS HANG!

FEATURED RESOURCES

SECURITY

Shifting Security Left: DevSecOps eGuide

White Paper | StickyMinds

Log In or Register to post comments

USER COMMENTS

0 comments

YOU MAY ALSO LIKE

SECURITY

Agentic AI Opens Up New Security Flaws

Nahla Davies | August 27, 2025

We use cookies to ensure you get the best experience on our website.

SECURITY

Countering the Threat of Ransomware-as-a-Service (RaaS)

Nahla Davies | August 22, 2025

SECURITY

Building Trust in Blockchain: The Role of Smart Contract Testing

Ekaterina Egorova | November 18, 2024

Join the crew for FREE! Access exclusive content, contribute, and more.

See All Member Benefits

We use cookies to ensure you get the best experience on our website.