

Managing SMB Tool Synchronization

*A White Paper Highlighting Overhead Cost
Reduction and Improved Developer Productivity
enabled by Genuitec's Pulse "Freelance Team
Edition" product*

July 2009



Managing SMB Tool Synchronization

EXECUTIVE SUMMARY

Developers who utilize Eclipse-based tools can spend anywhere from one to upwards of ten hours each month managing and synchronizing their development environments with those of their team members. While this coordination effort is certainly critical to enable team collaboration, the organization's cost for each developer can easily reach thousands of dollars per year for this tool maintenance overhead. The additional cost of inadvertent mistakes due to inconsistent tool stacks can have these costs grow even higher.

These untracked expenses are particularly problematic for small and mid-size businesses as they traditionally have smaller revenue streams, thinner margins and less room for operational inefficiencies than their larger competitors. As a result, minimizing wasted time and the accompanying overhead expenses becomes particularly critical to achieving overall profitability.

Eclipse developers typically spend hours each month manually managing and sharing their development environments.

This white paper reviews how the manual and tedious process of team tool synchronization can be replaced by a fully automated solution that enhances development efficiency while reducing the expense to the organization by an order of magnitude.

WHAT DOES IT MEAN TO SYNCHRONIZE A DEVELOPMENT TEAM?

Simply stated, a synchronized team comes to agreement about the fundamental requirements of their development environment and ensures that each environment is conformant for all team members on every machine they utilize. Specifically, for developers using Eclipse, this commonality includes the following:

- The Eclipse-based tool stack
 - The base IDE (Eclipse distribution, MyEclipse, ...)
 - All additional Eclipse add-ons (Subversive, FindBugs, Mylyn, ...)
 - All prerequisites of selected add-ons
- Relevant projects in the workspace
 - Including versions, branches and working sets

- Workspace preferences
 - Source code formatting and style preferences
 - Boilerplate copyright notice comments
 - Other company standards

It's widely recognized that keeping a team in “tool sync” reduces the number of bad builds, broken environments, and enhances collaboration efficiency, but it's seldom done in practice.

Why don't teams stay in sync, or do they?

To accomplish team synchronization manually, a variation of the following process is traditionally undertaken by development teams:

1. The team appoints one developer to be the Technical Environment Architect (TEA) who will be responsible for defining and verifying the development tool stack.
2. TEA initially spends many hours (or days) manually downloading, installing, configuring, validating, testing, and demonstrating a variety of Eclipse IDEs and plug-ins.
3. At some point TEA finally completes his research and creates a development environment that he then needs to share with his team.
4. TEA creates a wiki page containing detailed instructions on how each developer can replicate the creation of the development environment on his local machine.
5. TEA emails the other development team members a link to the wiki page and requests they all go through the installation / configuration process exactly as described.
6. TEA then spends hours helping one or more team members for whom the wiki instructions somehow did not “work” for one reason or another.
7. At the conclusion of the entire configuration / reconfiguration exercise, the team becomes synchronized - at least for a short while.

Initially synchronizing a team and keeping everyone up to date is a time-consuming and error-prone process.

The problem with remaining synchronized over time is that during the stress of the next development / delivery cycle, most developers will prioritize keeping their environments in sync as “low priority.” So, as TEA modifies the tool chain with additions or updates, dutifully updates the wiki and emails everyone so the changes can be manually applied, his emails soon begin to fall on deaf ears. After all, developers are busy people, so if their own environment works and all the code still compiles, why would they take time to manually apply all these changes?

But TEA isn't the only one making changes. To further complicate the issue, several team members are adding projects to the workspace, deleting them, or changing their source control branches. Of course they all comply with the established policy, so wiki updates are made and emails are sent to notify the team of each change, hopefully without accidentally stepping on changes made by a peer.

Unfortunately, once a developer has fallen a little behind with the incremental changes, it becomes increasingly more difficult to recover and update his environment. So, since nothing tragic has happened yet, the tendency is simply to ignore all the emails and focus on development tasks instead. Then one day, when he's updating his projects from the version control system to begin work on something critical, his workspace simply won't compile cleanly anymore. He's now so far out of sync with the team, by virtue of his inaction, that his environment is simply "broken".

When forced to synchronize their environments manually, most developers eventually fail to keep pace. Unpredictable, expensive and time-consuming "environment breakage" is to inevitable outcome.

Once a developer's environment is broken, there is no choice but to resynchronize with the current team settings to restore functionality. Typically this environmental diagnosis / fix phase takes multiple hours of effort from the broken developer as well as considerable time from TEA and possibly other teammates that have been working on the project configuration changes.

Unfortunately, by the time the broken environment is rebuilt and the developer is productive it's not uncommon to waste close to a man-day on a large product when the combined efforts of all participants are considered. As so much time goes into manually recreating an environment from scratch, the first approach is to attempt to "patch" the existing setup which leads to further wasted effort before the decision is made to just "start over."

So, this cycle repeats with the next team member and the next. Always at different times and under different time lines, resynchronization for each team member does eventually occur, even if it's forced by breakage and only as a response to the ensuing calamity.

In the end, the choice isn't whether or not your team will periodically need to synchronize their environments. They will. The choice is simply how they will do it and whether they will choose the time or the time will choose them.

MAKING TEAM SYNCHRONIZATION WORK

As the section above illustrates, a manual team synchronization strategy, no matter how well intentioned, streamlined or documented, is simply going to fail over time. For any team synchronization solution to be truly effective and beneficial, it must satisfy the following requirements:

- Accelerate the creation and maintenance of the development tool stack by the Technical Environment Architect (TEA)
 - Provide a large, annotated, descriptive catalog of both FOSS and commercial tools to facilitate search and discovery of valuable tools
 - Allow custom tool additions from providers that are not in the catalog, including proprietary internal tools only available on a company's intranet¹
 - Enable the rapid creation of custom software stacks in an intuitive and efficient manner
 - Enable a easy way for TEA to experiment with changes to a configuration without impacting others, or by trailing the changes with a small subset of peers
 - Ensure that all installation prerequisites are automatically satisfied for the software stacks
- Easily allow sharing of a completely configured development environment
 - Complete, custom Eclipse-based tool stack
 - Workspace preferences
 - Relevant projects in the workspace
 - “One-click install” on any machine; no wiki necessary
 - Sharing with peers all over the world
- Automatically synchronize the entire development environment
 - Synchronization checks performed automatically or on demand
 - Synchronization for local as well as remote workers and offline usage of the tool stacks
 - Immediate or delayed update, as desired

Agreeing on a common tool chain isn't enough. Teams need a mechanism to ensure their entire development environment remains synchronized as well.

¹ Exposing the details of the proprietary tools outside the company's intranet is prohibited.

Genuitec's *Pulse Freelance Team Edition* product provides these features for only \$60/year (or \$6/month) per developer² as well as these additional benefits:

- The fastest installations available
 - Multi-threaded downloads
 - Dynamic mirror selection and optimization from both the Internet and the corporate intranet
- The smallest installation footprint
 - 5MB download to get started
 - Common plug-in cache across profiles to maximize reuse and minimize disk requirements
- No additional infrastructure required
 - Configuration, installation and synchronization services are provided in a SAAS (Software as a Service) model³
 - Fully redundant, scalable, and reliable
 - All provisioned applications are installed fully on the client machines so Internet connectivity is not required to use the development environments
- Development profiles are fully portable and reproducible
 - Easy for a developer to replicate their environment and work on multiple machines or OS's, if needed
- Full technical support is included in the subscription price
- Fully cross-platform
 - Available for Windows, Linux, and Mac OS/X

*Automatically
synchronizing a
development team through
Pulse Freelance enhances
their collaboration
efficiency, eliminates
drudgery and reduces
costs.*

Simply stated, Pulse Freelance Team Edition synchronizes teams effectively, automatically and inexpensively. And, now that you realize how much your lack of team synchronization is costing your company in lost time, slipped deadlines and decreased productivity, can you really choose to ignore the issue? Or will you do take the next step and transition from being “informed” to becoming an “informed consumer” of Pulse Freelance Team Edition?

CONCLUSION

The ability for the team members to keep their complete development environments in sync is critical, and it facilitates the development of the team's productivity. Historically, this has been done manually either at incremental cost on an ongoing basis or great cost when a breakage occurred. Going forward, teams can choose an innovative, inexpensive and completely automated solution to this age-old problem: Pulse Freelance Team Edition.

² Volume discounts apply for yearly license orders over 50 users

³ Local server installations with custom catalogs are available as part of Genuitec's Pulse Private Label product



*Pulse for SMB
July 2009*

*Genuitec, LLC
2221 Justin Road #119-340
Flower Mound, TX 75028
USA*

*<http://www.genuitec.com>
<http://www.poweredbypulse.com>*

*Copyright ©2009, Genuitec,
LLC All rights reserved.*

This document is provided for informational purposes only and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document.