Level: Intermediate
Works with: Lotus Workplace
Updated: 01-Dec-2003

by Flemming Christensen
with Joel Abbott and Greg Pflaum

IBM Lotus Software division successfully migrated all source code for IBM Lotus Workplace to the Rational ClearCase UCM solution in June 2003. The key challenges to overcome included providing the source control services to a globally dispersed team around the clock, while ensuring minimal impact to the ongoing Lotus Workplace project by the migration effort and crafting the foundation for an efficient, iterative, and cyclical development process. For the benefit of customers as well as internal colleagues, who may be considering a similar migration, this article summarizes our goals, plans, migration activities, and lessons learned.

What is IBM Lotus Workplace?

IBM Lotus Workplace is the next generation collaborative platform from IBM Software Group. It was created by rewriting most of the Lotus collaborative functionality—email, instant messaging, Web meetings, e-learning, search, and so on—on the J2EE platform. It serves this functionality through WebSphere Portal. In March 2003, all Lotus next generation projects merged into a single platform to become Lotus Workplace. Earlier this year, Lotus launched Lotus Workplace Messaging 1.0, which showcases the platform's messaging capability. The development and test teams proceeded immediately with version 1.1, the first full-featured release. Version 1.1 contains more than 40 subcomponents grouped into eleven major components, including installation, messaging, calendar and scheduling, on-line meetings, instant messaging, team spaces, and more.

Developing the Lotus Workplace family of Version 1.1 products involved over 500 employees, working at eight IBM sites distributed globally across the US, Europe, and Asia, plus a small number of remotely connected developers from sites mostly in the US and Europe. The next version of Lotus Workplace is already under way with 11 IBM sites participating in the development process.

As of late October 2003, the Lotus Workplace source code consists of roughly 4 million lines of code distributed over 43,000 files. The aggregated source code is about 1 GB in size, and the binary executable is of the same order of magnitude. These numbers include "dead code" that will not be activated until a later release. Furthermore, the code has not yet been through the final clean-up prior to release, which will reduce both source files and executables in size. Nevertheless, these numbers give an idea of the magnitude of the source being managed in our Rational ClearCase UCM solution.

Why migrate and why to ClearCase UCM?

Merging 11 components into a single platform left us with no alternative but to implement a common source control solution. Without a common solution, the effort to manage code interdependencies between components, whose source code was stored in different solutions, would quickly grow unmanageable, and it would slow down the pace of the development project. The pre-existing component projects had implemented

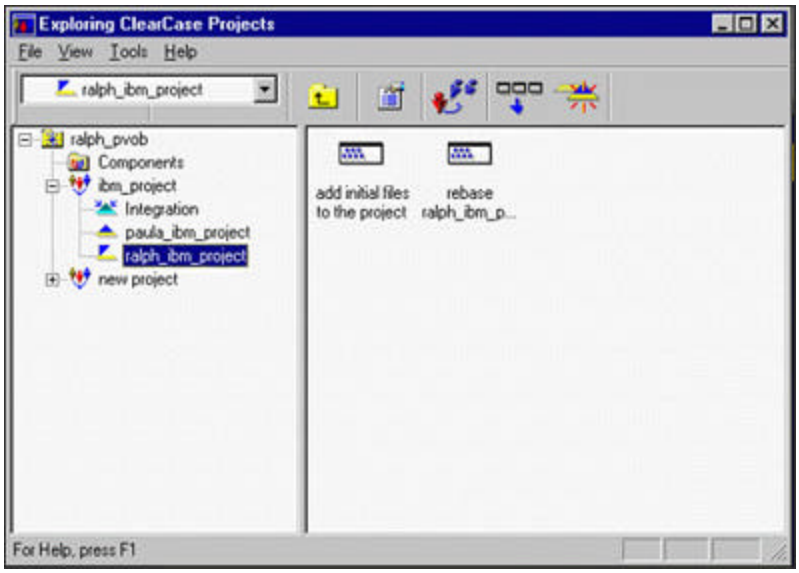
© Copyright IBM

different source control solutions: Some were using CVS, some ClearCase, some Visual Source Safe, and some PVCS. In addition, Lotus Workplace has interdependencies with code components developed by the WebSphere team in Raleigh, North Carolina and by the WebSphere/DB2 team in Toronto, Ontario, Canada. Both the Raleigh and Toronto teams were using IBM's CMVC solution.

Migration was thus dictated by circumstances. When selecting a new tool in software development, it is natural to first search for internally available tools as well as colleagues who have had experience with those tools. Because of the recent acquisition by IBM of Rational Software, it was obvious that we needed to look at the ClearCase solution and to determine whether or not it would meet our needs. The development team for Lotus's largest platform and product area, Notes/Domino, as well as the Sametime development team, have used ClearCase for source control for years, so we were confident that ClearCase could meet our needs, and better yet: We had colleagues, who had worked with ClearCase for several years.

What is Rational ClearCase UCM?

ClearCase UCM is Rational's source control tool or configuration management solution. UCM stands for Unified Change Management. It is a powerful, professional, GUI driven multi-user tool offering parallel development capabilities and a set of associated tools for merging and working with code. For the Lotus Workplace project, we implemented the 2002 version of Rational ClearCase UCM. The following screen shows a ClearCase project.



So why the UCM version? We chose the UCM version of ClearCase to take advantage of the Unified Change Management (UCM) functionality to help us manage change. The configuration management concepts offered in UCM were identified as necessary to produce a large product. These concepts are already in use by Domino/Notes and Sametime, where they were designed using ClearCase with homemade tools and processes. UCM offered these concepts out-of-the-box giving us the ability to deploy quickly, rather than having to customize the home grown tools for the Lotus Workplace project. In addition, UCM better integrates with other Rational tools providing a full suite of development tools to enhance the development process.

Two very key differences between ClearCase and some of the solutions initially used by the component teams are its support for parallel development and for multi-siting. These two aspects of ClearCase make it particularly suitable for large development teams, such as Lotus Workplace. Parallel development is the ability for multiple developers to simultaneously modify copies of the same file. Merge tools are used later to resolve merge conflicts when multiple sets of changes to the same file need to be delivered to the source. This is a very different environment from purely sequential development, where only one developer can check out a given file from source control at a time. With over 500 engineers working on 11 interdependent major components, parallel development must be enabled; otherwise, overall productivity would suffer considerably.

© Copyright IBM

www.lotus.com/ldd/today.nsf

Multi-siting is the ability to place local replicas of the source code at each participating site to minimize the need for WAN communication by individual developers. ClearCase is an input/output intensive solution that is also latency sensitive due to a number of RPC calls, and it would not work well to ask our European and Asia-Pacific sites to work directly in a US-based replica. Performance is relatively slower over the WAN, where latencies are normally higher than on a local LAN.

Goals for the migration

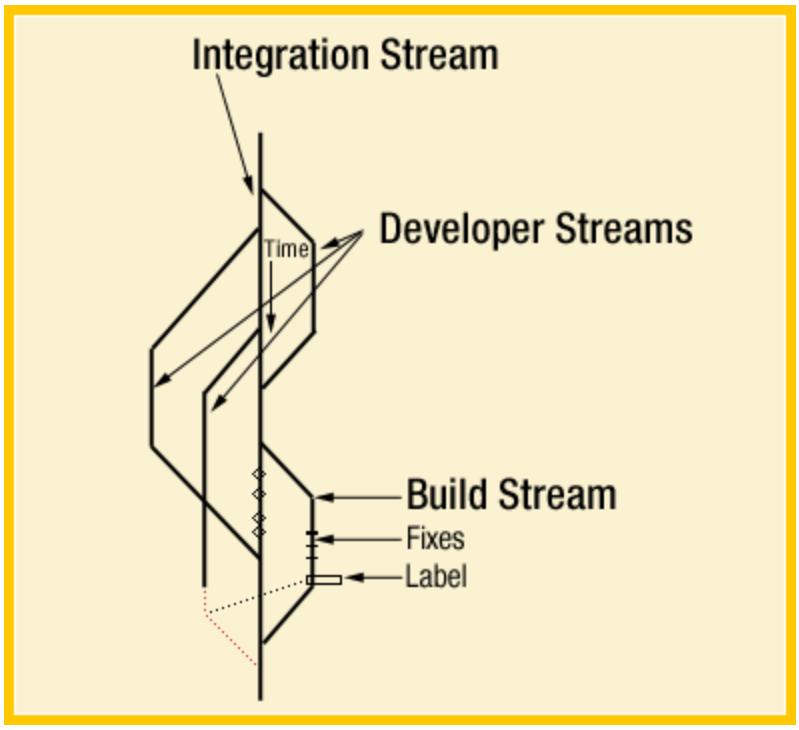
The goals for this migration were fairly straightforward, yet challenging:

- To provide a common source control solution for all components of the Lotus Workplace offering
- To make the solution available to all participating development sites around the world
- To provide a mechanism for remotely connected developers to deliver source code, such as from home offices
- To provide the ability for all Lotus Workplace teams to share code around the clock (that is, to minimize any lock-out periods)
- To migrate existing source code from a variety of source control solutions to the common solution
- To group ongoing code changes into manageable subcomponents to enable functional integration efforts to take place prior to system integration
- To provide a common code level for developers to adopt at regular intervals, reducing developer time spent tracking common issues and paving the way for an iterative, cyclical development process
- To complete the migration project by the planned Design, Code, & Unit Test (DCUT) completion date
- To minimize system downtime during migration to no more than one night
- To minimize any temporary negative impact to developer productivity during migration
- To provide training to ClearCase administrators and to users
- To provide a framework for working on multiple releases in parallel (that is, merging code between releases)

Usage model

As mentioned earlier, Notes/Domino and Sametime were already using ClearCase, so our initial plan was based on the usage model applied by those teams. In the following illustration, time progresses along a downward axis. The integration stream, or integration branch, is the central vertical line. Developers create developer branches off the integration branch, modify files, and deliver the developer branch back to the integration branch. ClearCase works with branches, while ClearCase UCM uses streams. The two terms are used interchangeably here.

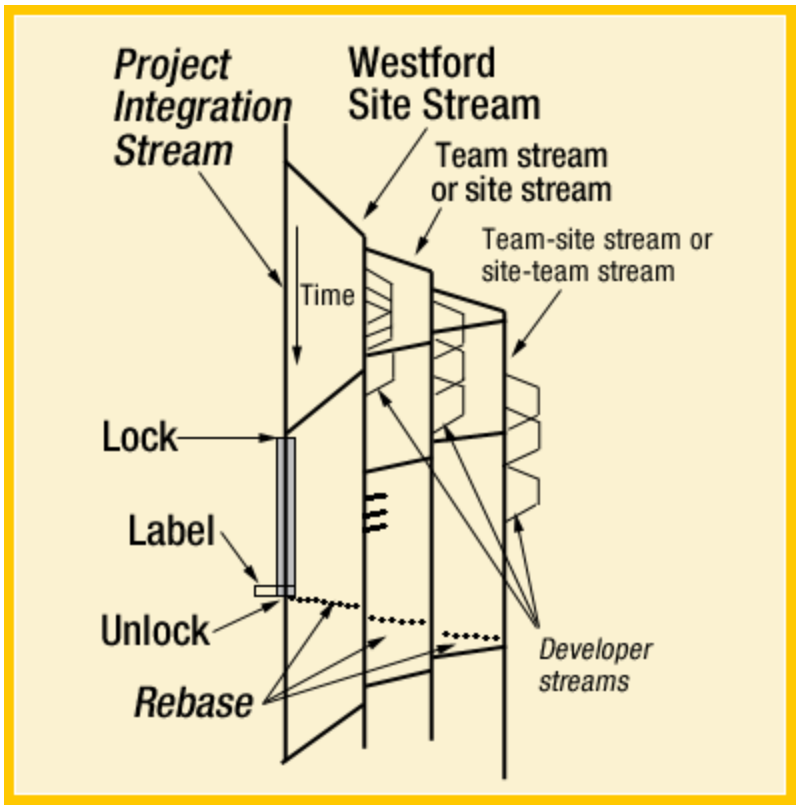
© Copyright IBM



At some point, typically daily, a build branch is created. Code on this branch is used to build from, so developers can continue to check their new contributions into the integration branch, while the build team is working on the day's build. Fixes may be needed on the build branch, but once the build is accepted, the build team publishes a label to designate a required baseline on the build branch and to force all other branches to rebase to it before delivering code to the integration branch. The rebasing operation provides a way for developers on a project to update their work areas with code that has been integrated, tested, and approved for general use. This code is represented by baselines. The ClearCase administrator organizes delivered activities into baselines. Usually baselines go through a cycle of testing and bug fixing until they reach a satisfactory level of stability. When a baseline reaches this level, the ClearCase administrator designates it as a recommended baseline. To work with the set of versions in the recommended baseline, you rebase your work area, which replicates the baseline code to your work area or stream. This is represented by the dotted line crossing over the center in the graph.

This Notes/Domino and Sametime usage model for ClearCase was originally created to allow developers unrestricted availability of the integration branch for merges, so they can deliver their code any time without a need to lock the branch while building. This usage model would have worked for Lotus Workplace as well if it had not been for a significant yet small difference between ClearCase and ClearCase UCM. In the UCM version of the product, a rebase cannot be forced to a label anywhere but on the integration stream. ClearCase UCM cannot find the label to rebase to, unless it is located on the integration stream. This forced us to rethink the usage model for ClearCase UCM on the Lotus Workplace project. We came up with a modified usage model as shown in the following illustration covering a 24-hour cycle.

© Copyright IBM



Because of the requirement that the label be on the integration stream, we immediately created a Westford site stream to be the only child of the integration stream, providing seven day, 24 hour availability of a mechanism to exchange code between any set of teams. Westford is the main development site for Lotus Workplace.

While the Westford site stream is an only child, it has multiple children. To maintain flexibility and control disk space usage, we set up the remaining stream hierarchy to meet the needs of each individual component team rather than enforce a rigid model across all teams:

- Some teams are entirely Westford based, and for those teams, we created team streams as children of the Westford site stream. The team stream facilitates developers sharing code within their team prior to delivering to all teams via the Westford site stream.
- Some teams have a handful of team members at an overseas site, such as Dublin, Ireland. For those teams, we created a team-site stream as a child of their team stream. Dublin-based developers can share code between them within the team-site stream prior to delivering the code to their team stream in the US.
- Dublin is a mixed site owning an entire component (the installation code) suitable for a team stream, but also having small teams of two to three developers, who are part of the US-based component teams. These developers have a team-site stream, which delivers to a team stream in the US, which in turn delivers to the Westford site stream, and ultimately to the integration stream.
- Some sites outside Westford own an entire component, such as Lexington, Kentucky and Rehovot, Israel. For those, we set up site streams as children of the Westford site stream. This enables developers at those sites to share code prior to delivering to the Westford site stream.
- Developers can create developer streams off any stream within the hierarchy, whether it is the Westford site stream, a team stream, a site stream, a team-site stream, a site-team stream, or even the integration stream itself. However, they can't deliver to the integration stream; only the Westford site stream can do that.

Some, though not all, of these streams are in place today, and the multi-site replication is fully functioning. An additional construct easily adapted into this flexible usage model is the idea of a relatively short-lived feature stream, which is being planned for the next version of Lotus Workplace. Two or more component teams may cooperate to develop a new feature for the product. They can easily set up a feature stream for their two to three component teams to share code related to this feature and test that it works before delivering it to the overall

© Copyright IBM

product and all the other teams via the Westford site stream. The ability to test a new feature in isolation before delivering it to the common code stream greatly stabilizes the overall code package enabling other teams to make solid progress in their own areas without being hampered by problems introduced by the new feature code.

The important attributes of this usage model are:

- Most merge conflicts are resolved locally—by merging code at overseas locations prior to delivering to the Westford site stream, merge conflicts are resolved by the developers who wrote the code rather than by a Westford-based team attempting to complete the deliveries several hours after the developers in Europe or Asia-Pacific have gone home.
- The build happens from the integration stream, so a rebasing of the code can be mandated.
- Locking of the integration stream for building is worked around by providing the Westford site stream.
- Delivery to the Westford site stream is facilitated around the clock.
- Merge conflicts have been eliminated for the project integration stream because it has only a single child.
- Feature streams allow separation of various product activities by their level of maturity.
- The power of this usage model lies in the separation of merge conflicts from the project integration stream and the flexibility in stream hierarchy.

Multi-site solution

The configuration management solution must be usable by teams at all participating sites. This is easily achieved with Rational's multi-site functionality. We replicate source code changes to all participating sites every 10 minutes around the clock. Because of the frequent replication, bandwidth and latency to each site is not a major issue. Our biggest difficulty with the multi-site functionality has been the need to provide a solution for remote completion of team and site stream deliveries in Westford. Those code deliveries are always made to the Westford replica. Overseas lead developers post, or initiate, a delivery of their team's or site's common stream, and someone in Westford has to accept or complete the delivery. It is the latter part of the delivery which includes the resolution of merge conflicts, which is better done by the developers themselves. That's why it is best to merge all developer streams from a team, or at a given site, into a common stream before delivering the common stream to the Westford site stream. To help with the overseas deliveries, we decided to provide a small number of workstations located in Westford for remote control operation by overseas team leads to complete their deliveries in the Westford replica.

Wanting to ensure we had a solution for most conditions, we defined three solutions for remote control: Windows Terminal Server, Windows XP Remote Desktop, and Virtual Network Computing (VNC) used in conjunction with Secure Shell (SSH) to address vulnerabilities. SSH is a communications protocol for connecting to a remote machine similar to Telnet. It provides strong authentication and encryption and has the ability to tunnel other communications between the SSH client and server. We also configured some remote access machines with IBM Desktop On-Call. Windows Terminal Server is the only solution among these that provides concurrent multi-user capability. The team is currently experimenting with a ClearCase Web server to help remotely connected developers, typically people working from home via a phone line, perform important tasks without running into the latency issues they have when connecting the client over a dial-up connection. However, the ClearCase Web server has limited functionality and does not appear to offer what we need at this time, for instance, there is no integration with WebSphere Studio Application Developer. Also, it only allows creation of child streams from the top level—the project integration stream—and it lacks a version tree browser.

Migration

The main steps we took to complete the migration can be summarized as follows:

1. Decide on hardware and network location for the ClearCase UCM servers. See the sidebar "Hardware planning and specifications" for more information.
2. Design usage model. See the earlier "Usage model" section for more information.
3. Dedicate a person to identify all users and IDs and to manage the creation of needed domain accounts, ensuring everybody has the right connectivity.
4. Create project-specific documentation explaining how to set up and how to migrate to ClearCase UCM and distribute them to all developers. Also provide project-specific use instructions including naming conventions for streams and so on.
5. Identify one to two ClearCase mentors per team.

© Copyright IBM

6. Find an experienced deployment resource to teach a train-the-trainers course on Rational ClearCase. Include a hands-on demo of exercising ClearCase from within WebSphere Studio Application Developer or Eclipse.
7. Designate the ClearCase mentors as level 1 support for users during transition, designate specific people for level 2 support, and specific ClearCase administrators as level 3 support.
8. Set up a support database to enable the teams to ask questions and to have them resolved off-line to reduce the workload on key resources.
9. Provide a test environment for users.
10. Ask each user to edit a certain file in the test environment's versioned object base (VOB). A VOB is Rational ClearCase terminology for a source code database associated with a single project. By letting users edit a selected file, the migration team can get positive confirmation via the log that users are up and running in the environment and able to check out, edit, and check in material.
11. Some developers have work in progress in the old solution, when it comes time to migrate. Provide a way to enable them to deliver to the new solution after they're done modifying the files they had checked out from the old solution. It is not required that all work in progress be checked into the old source control solution the night before the migration.
12. Copy all source from the old solution to the new, build from the new solution, and verify that you can create a working build at least as good as what you built from the old solution. After this proof, lock the old solution, copy all the source code to the new solution again, build, and recommend a baseline, then unlock the new solution and have developers use that from then onward.

By the very nature of the migration, we forced people to work in the new ClearCase environment overnight because there was no way for them to deliver code in the old system. Our greatest concern was to anticipate the magnitude of the user support task during migration. Because we had a limited number of skilled ClearCase administrators, we could not afford to spread their effort across user support in addition to administration because their skills were needed for the code migration. Steps 3 to 10 above helped us mitigate that workload up front and likely contributed to the success of the migration. User support bottlenecks were not a problem, and the productivity of the 500 person Lotus Workplace development team was minimally impacted by the migration. We migrated the component teams one at a time, so the actual migration excluding preparation work took about two weeks. We could have migrated faster, but again we were concerned about a potential overload of user support requests. The build team was tied closely into this effort, so the build scripts were updated each time a component was moved to a new location.

Lessons learned

Key lessons we learned from this migration effort include:

- In spite of the very short planning time for this migration, the effort was very successful, proving that ClearCase UCM is a sufficiently robust tool for a 500 person development team dispersed across eight sites on three continents. With more time for detailed planning, migration to ClearCase UCM can be accomplished with minimal risks.
- Our migration benefitted greatly from having two to three skilled, experienced ClearCase administrators available. Teams without prior experience in ClearCase should consider borrowing a skilled administrator for a short-term assignment and should ensure their own administrators receive sufficient education before embarking on the migration. If there are no skilled ClearCase administrators in the organization, hire or contract one.
- With a large development team, there are all levels of understanding of source control solutions. Comprehensive communication is necessary to reach all developers.
- User support *during migration* required one ClearCase skilled developer acting as a team mentor per 20 developers, a level 2 supporter per 100 users, and a ClearCase administrator per 100 users. This level of effort was needed for about a month. Less is needed for steady-state operation after the migration and initial learning is completed.
- Too many users did not install the ClearCase client and use the test environment prior to the day of their component's migration to ClearCase UCM, resulting in an increased volume of user support requests during critical parts of the migration. Although we handled the increased support volume, better tracking of user readiness and follow-up to ensure users had exercised the test environment and were ready would have helped further smooth the migration.
- Remotely connected developers, typically working via phone line, DSL, or broadband, generally experience performance slow-down of Rational ClearCase UCM. Different workarounds may or may not satisfy the

© Copyright IBM

individual developer: Use a remotely controlled workstation located on the same LAN as the ClearCase server, or use snapshot views for hands-on work and accept the long delivery and rebase durations by running them during off-hours.

- Rebase performance can degrade in some cases, especially if code is received from many UCM activities, which can be the case for a developer who has chosen not to rebase for a number of consecutive days. We are looking into ways of alleviating this problem. In the meantime, daily rebasing of the code is recommended.
- Unless you're based on a LAN with a replica of the ClearCase VOBs you're working on, ClearCase performance is slower due to its use of the RPC protocol, which also makes it latency sensitive. As a general rule of thumb, latencies above 10 milliseconds cause degradation of ClearCase performance to an extent that is not acceptable. Workarounds include remote control of a system on the appropriate LAN or setting up a ClearCase Web server.
- Wireless cards did not work well for us with ClearCase because they occasionally disconnect from the network, and when they are connected they are too slow, at least in some facilities. Under ideal conditions the maximum bandwidth with 802.11b is 11 Mbit/s, and you are sharing the available bandwidth with anyone else using the same wireless access point. The result is lower bandwidth and higher latency. However, the problems in this area were caused by low reliability of our wireless network in a few areas, rather than by the ClearCase product.
- If your stream is very long lived, you may have many elements in your stream, in which case UCM operations, such as a rebase, can take longer because many more elements need to be searched and possibly merged. If this becomes a concern, simply obsolete the long lived stream and begin development again fresh with a new stream.
- You need a clear naming strategy to avoid running into the 260 character length limitation for ClearCase path names, which includes stream names in the version-extended path.

Conclusion

Rational ClearCase UCM proved to be a robust solution, even if there are areas where the tooling can be improved to boost productivity—for instance, a version tree browser for the ClearCase Web server. Overall, the migration was successful, so the experience collected, such as the user support volume, training approaches, and so on will hopefully be helpful to other teams planning to migrate to Rational ClearCase UCM.

Acknowledgements

A team of people accomplished this source control migration. The authors thank a team of migration project contributors including Lewis J White, Issam Hachem, Stuart Clemons, Anke Vorbau, Erik Anderson, and all the designated ClearCase mentors from each component development team. Special thanks are extended to Malcolm "Dudley" Thomas from Rational Software, who on short notice visited to deliver an accelerated T3 (Train The Trainers) class on Rational ClearCase UCM to the ClearCase mentors on the IBM Lotus Workplace project. The success of this project also relied on a world class build team managed by Ann Innis and including Kar Chung and Al Mello drawing from multiple sources during the migration. Thank you all!

ABOUT THE AUTHORS

Flemming T Christensen earned a doctorate degree in engineering from the Technical University of Denmark and a business degree specializing in international economics and management of innovation from the Copenhagen School of Business. He has professional experience in consulting, customer support, and software test. He is currently managing a fledgling Test Technology Group at Lotus focused on building an integrated, shared tools environment for a global set of software test teams, as well as the business processes to manage test efforts across multiple sites.

Joel Abbottt is a Senior Software Engineer working on IBM Lotus Sametime and IBM Lotus Workplace products where he leads Configuration Management and Release Engineering teams distributed around the world. He has been in the software industry since 1989 where he has had numerous technical and managerial roles. Joel is a graduate from the University of Kentucky and splits his time between Lexington, KY and Dublin, Ireland.

Greg Pflaum is the configuration management project lead for Lotus Workplace and Notes/Domino. He joined Notes/Domino development in 1990 and was responsible for much of the Notes networking code, including the addition of support for TCP/IP and other network protocols. He has held a variety of developer and management positions and has worked with ClearCase since 1999.

© Copyright IBM