



Productivity Gains with AccuRev vs. ClearCase

Contact sales@accurev.com to get accompanying **Customized ROI Analysis** – [Click Here](#)

Written by:

Damon Poole, Chief Technology Officer, AccuRev, Inc.

About the Author

Damon Poole is CTO at AccuRev. He is the Architect of AccuRev, AccuWork, AccuReplica, and AccuWorkflow. Prior to AccuRev, Poole worked as an SCM toolsmith for IBM, Compaq, FTP, and others using ClearCase and a wide variety of other leading tools. Prior to consulting, Poole was responsible for the first rollout of ClearCase and ClearCase MultiSite at Fidelity and pioneered the use of a componentized stream-based layer on top of those tools which shielded users from the complexities of ClearCase and branch mastership. Poole was an invited speaker at the 1994 CCIUG.



AccuRev Advantages For ClearCase Users

This document looks at the differences between AccuRev and ClearCase from a productivity perspective. It covers a broad range of day-to-day development activities that depend on the SCM tool in use. For each activity it discusses how it works in ClearCase and AccuRev, why it is a problem in ClearCase, and how it is addressed in AccuRev. The impact on productivity is then estimated at the end of each section.

To be fair, there are things that ClearCase does that AccuRev does not do that positively impact productivity in ClearCase compared to AccuRev. Those activities are covered in a separate section at the end of the document.

This document does not attempt to cover all of the feature differences, advantages and disadvantages of the two tools. It only looks at features which can be directly connected to time gained or wasted where the amount of time is significant. In general the threshold is 10 minutes per developer per week, though there are a couple of exceptions. The average gain is thirty minutes per week. Thirty minutes per week may not seem like much time, but considering that there are over twenty differences in this document multiplied by the number of end users, the difference adds up fast.

The frequencies and productivity gains in this document are intended to be representative examples. The exact numbers for a given environment will vary depending on the number of users, the frequency of releases, the amount of changes per developer per day, the hardware and network environment that the systems are run in, and the usage patterns of your organization. However, the differences between ClearCase and AccuRev are architectural in nature and the explanations of where the productivity advantages come from will show that even if the numbers in your environment may be different, they will still be significant.

According to this document, the raw productivity difference between AccuRev and ClearCase can vary from 10% to 30%. An obvious response to that is that developers don't spend even 10% of their day using ClearCase. That's true, but it isn't because they aren't waiting on ClearCase, it is because they multitask while they are waiting. For instance, if they need to wait a day for a new baseline to be created by a ClearCase admin, they will work on another project during that day. But, developers can only effectively multitask if the wait is long enough to do so and they have other tasks to work on. Even if after multi-tasking the real hit to their productivity is 5%, that's still a significant cost, and there is a problem with forced multi-tasking.

If a developer is forced to multi-task, it may not affect their individual productivity, but it will affect the duration of their assigned tasks. The more that individual developers are blocked by their SCM system, the more likely it is that they will then block other developers that are dependent on their changes. Thus, the real impact of waiting on ClearCase is on the critical path of your projects.

Assumptions

ClearCase usage patterns vary from organization to organization. This document assumes that in a typical ClearCase shop, there is a clear delineation between "developer" and "admin" tasks. There is an admin, generally referred to as a release engineer, that is responsible for creating repositories, implementing branching strategies, writing and maintaining scripts, and troubleshooting tool problems. In some shops these responsibilities are decentralized and are the part-time responsibility of one of the members of each team. In other shops these responsibilities are handled by a central organization. Sometimes there is a toolsmith or two that maintains the scripts. In any case, this document assumes that all administrative and scripting duties are handled by one or more release engineers.

Private Branches Implemented with Regular Branches (A1)

ClearCase does allow users to create branches for private use, but the branches in ClearCase used for private use are the same branches used for shared use. Shared branches do not have the right behaviors for private use. To put it simply, private branches should use the same model as simple checkouts. When you check out elements in ClearCase, you can easily find out which elements are checked out. That makes it easy to find your work in progress. But if you create a branch for yourself, it becomes more difficult to determine your work in progress. You now need to do both a listing of checkouts as well as a branch comparison.

Making your work public is no longer a simple matter of doing a check-in. Instead, you have to manage two views: one for your private branch and one for the destination branch. You then have to run findmerge, specifying both the source and destination branch. Findmerge will do the merge to the destination for you, but you will then need to do the check-in yourself. Plus, from then on you need to do a merge to your branch to get any changes that other people have made because otherwise the version on your branch will eclipse it. Regardless of the number of steps, this also means that the best practice of merging to your work area is defeated. To make sure that you are not breaking anything via your merge, you will need to do a build and test cycle in the view based on the destination. That might require further changes, and that view might not completely correspond to your private branch because you might not have merged recent changes into your private branch. All in all, it is a very confusing many-step process which requires you to maintain two complete development environments.

AccuRev was architected with private branches in mind from the beginning. The workspace is a high-level object that has a unique name in the system and can be operated on directly. Making changes in a workspace is a three-step process: edit, keep, promote. AccuRev uses all writeable files by default, there's no need to check out files in order to edit them.

First, you edit files to incorporate your changes. As you reach a milestone, or you just want to save your work in progress, you keep your edits. In base ClearCase, this would be a check-in that would be immediately visible to other users and potentially break their builds. If you are working on files for a long period of time, it is not practical to delay checking-in changes. As a result, users often copy files or whole trees aside and add ".2," ".3," ".4," or something similar to simulate private versioning. In AccuRev, the keep command creates a new version in the user's stream, which is stored on the server. This enables easy diffing, history, and rollback of multiple private versions.

As users make changes in their workspace, AccuRev maintains a list of their work in progress. These are known as the "active members" of that workspace. In a new workspace, there are no active members. As the user makes changes, the list of active members grows. This is similar in ClearCase to finding out which elements have been branched on the user's private branch.

When the user is ready to make their changes public, they promote some or all of the active members. First, AccuRev checks to see if any of the elements have also been changed in the destination stream. If so, it alerts the user that they will need to merge those changes in first. The merge operation is a simple operation which brings all of the changes into the user's workspace. Once the merge is completed, the promote can now proceed. This then reduces the list of active members to just the list of whatever is left, if anything. Now that the changes are in the parent stream, they no longer need to reside in the workspace and they are removed from the workspace and inherited from the parent stream. Thus, when other people make changes to the same files that the user just promoted, they will be inherited instead of requiring merges with the changes on the user's private branch.

The list of active members provides a handy means to find out workspace-wide everything that the user is currently working on. There are many searches that are available to find out the workspace-wide status of files and the search results can be directly acted on. For instance with just a few clicks, you can find all pending changes, find the differences between them and the versions in the parent stream, and then promote the changes.

As changes are moved from stream to stream, they maintain their association with the workspace they originated in. This makes it easy to get a history of all changes that a user made in the workspace. The user can also get a history of his or her changes on a per-file basis, depot-wide, during a period of time, associated with a particular comment, or in any combination of these criteria.

Productivity gain: .5 minutes per developer.

Frequency: 10 times per day.

Waiting for ClearCase Admins/Release Engineers to Create New Configurations (A2)

Whether you are using base ClearCase, UCM, or base ClearCase with a layer of scripts and other tools built on top, it takes a long time to get a new configuration set up and ready for developers to use. The time it takes includes the fact that you have to send an e-mail request to Release Engineering and then wait for that request to be fulfilled. The amount of time before you have a new configuration varies from an hour to a day. During this time, no developers that depend on this new configuration can do any work on that new configuration. If they have other things to do, then they will not be blocked, but it reduces the chance that they will be fully productive.

The reason that creating a new configuration in ClearCase is so involved is that there are many operations involved in setting up a new configuration including setting up new config specs or config spec templates, laying down labels, and updating scripts. Information also needs to be created and disseminated to users about how to start using the configuration.

In AccuRev, all configurations are represented by "streams." A stream in AccuRev is akin to a config spec in ClearCase, but can be referred to as a first class object and used as part of the definition of other streams. All information related to a configuration is contained within a stream, and streams can be set up via a GUI instead of editing a text file. All stream information is also contained in the AccuRev repository and versioned instead of being stored outside of the repository.

As a result, it is easy to set up a new stream. In fact, it is so easy that you can allow team leads to do it themselves without worrying that they will make a mistake. The new stream will show up in the stream browser which gives an overview of the relationships between streams. It is akin to the project explorer in UCM, but with much richer functionality. For instance, you can click on any two streams and instantly find the difference between them in terms of files or issues (activities in UCM terminology). Since the stream shows up in the stream browser, it is easy for people to find and use your new stream.

Lastly, creating a new stream in AccuRev is instantaneous. Simply select the stream that you want to use as the basis, set a time rule if desired, and your new stream is ready immediately. There is no need to create branches or labels and there is no need to label files on a file-by-file basis.

The simplicity of setting up and the speed of creating a stream mean that the developers no longer have to wait to set up a new configuration. Set up only takes as long as it takes to define the new stream.

For more information about how AccuRev's streams work, please read the paper "Stream Based Architecture of SCM" available on our web site.

Productivity gain: 1-8 hours per developer that needs a new configuration.
Frequency: once per week.

Difficulty of Branch Management Leads to Shallow Development Hierarchies and Broken Builds (A3)

As described in the previous section, setting up branches is difficult in ClearCase. On top of that, effective use of branches is also difficult in ClearCase. As a result, ClearCase users rarely use private, team, or feature branches.

AccuRev was designed from the ground up to support easy stream visualization and manipulation. As previously described, stream setup is easy enough that team leads and individual developers can do it themselves. It is also possible to re-parent streams through a fast and easy drag and drop operation.

Operations that developers normally only do in their own views are simple to do as stream to stream operations. For instance, when a developer is done making a change in ClearCase, they will check it in. A check-in is made against a shared branch, thus the change becomes immediately available to other users. If the developer is not yet ready to share their changes, then they have to either keep working on their files without checking in, or create a private branch. Once they have created a private branch, their normal day to day operations are now more complicated. To make a change available to others they need to check in to their private branch, check out from the shared branch, merge to the shared branch, and then check in on the shared branch. And that's not even a good best practice because now the change was merged to the shared branch without being tested first.

The simplicity of using streams in day to day operations means that creating a development hierarchy that exactly matches an organization's needs is now possible. By having a deeper hierarchy, there will be less interference between developers and teams. Less interference means that there will be fewer broken builds. Fewer broken builds mean there will be fewer incidents of developers unable to proceed with their tasks because they are waiting for a change to be reverted or fixed.

Productivity gain: 10 minutes per developer.
Frequency: once per day.

Manual Patch Support (A4)

If you need to migrate a change from new development into maintenance, there is a good chance that there are multiple changes that have been made and you only want to deliver a specific change and not all of the differences in the affected files between the two projects. In the SCM world, this is called a "patch." ClearCase does not directly support this concept. Setting up a patch in ClearCase must be done manually. Furthermore, once you have done the patch ClearCase does not track the fact that that patch has been done. Thus, ClearCase cannot tell you what has been patched from one release to another and it cannot use that information to simplify future merges.

Productivity gain: 20 minutes per developer.
Frequency: once per week.

ClearCase Does Not Fully Track Rename Operations (A5)

Refactoring is a commonplace part of software development today. As a result, file names and file locations are changed very frequently. ClearCase handles move operations as two separate operations, a remove and an add. Although it tracks the namespace operation on a single element (unlike other tools which break the history), it does not link the remove and add operations. If you are trying to debug a problem and you need to track the location history of an element, the lack of a link makes it much more complicated than it is with AccuRev. In AccuRev, a move operation is recorded as a move operation and both the source and destination locations are readily available.

Productivity gain: 10 minutes per developer.

Frequency: 2 times per week.

Retargeting Work in Progress is Extremely Difficult (A6)

In software development, plans change frequently. Work that was originally being done towards one release is often retargeted towards a different release. In UCM, taking the work being done in one stream and moving it to be based on an entirely different stream is not well supported. But even in base ClearCase it is not a straightforward operation.

Let's say that you have a branch which represents work towards a particular project and that project now needs to be retargeted to a different release. To do the retargeting, the branch contents will need to be changed to look as though all work had originally been done against the new basis. That means coming up with a plan for making the changes, disseminating the plan, having everybody follow it, properly updating all affected config specs, and doing all of the required merges and rollbacks. There is a lot of room for error in that process due to the high dependence on manual steps such as determining what needs to be done, making sure everybody knows what needs to be done, and people doing all of the right steps.

The description above of how to retarget work in ClearCase actually only applies in the case that everybody involved is using private branches. Furthermore, it also assumes that teams are using team branches. To the extent that they are not using either or both of these strategies, any work which is in elements that are checked out cannot participate in the retargeting described above because only checked in changes can be merged to another branch. In that case, there really isn't a good solution.

Thanks to the O-O model of AccuRev's streams, you can just drag and drop the project stream from one stream to another stream. Since the only thing that changes is the basis, it is a single database record insert. The content of that stream is now whatever changes people have made to the stream and everything else is inherited from its parent stream. If the changes in the stream conflict with inherited changes, they are flagged. No merging is required unless there is an actual conflict.

Productivity gain: 60 minutes per developer.

Frequency: once per week.

Waiting for Broken Deliver Operations to be Fixed (A7)

ClearCase deliver operations often run into problems which require an admin to fix. This is because the deliver operations are not atomic. They do not check to make sure that they will succeed prior to starting, and once they fail, they do not rollback.

In AccuRev, all operations are atomic transactions. An operation is guaranteed to either fully complete or never even start. If you are checking in a collection of files and one of

them requires a merge, no files are checked in, and the need for a merge is indicated right away. If the power goes out in the middle of an operation, it either cleanly completes or it is completely and cleanly backed out when the server is restarted. There is never any need to resume an operation or clean up after a failure.

Productivity gain: 60 minutes per developer.
Frequency: once per week.

When a deliver operation fails, builds will also fail until the deliver operation is either completed or backed out.

Productivity gain: 5 minutes per developer.
Frequency: once per day.

Slow Operation Due to Slow Trigger Operation (A8)

Most if not all ClearCase installations make extensive use of triggers. ClearCase triggers fire once per file per operation. The result is that operations that use triggers are slowed down.

Since AccuRev uses transactions, triggers fire only once per operation.

Productivity gain: 10 seconds per developer.
Frequency: 3 times per day.

Triggers Fire Per Element Which Can Break Check-In (A9)

If you are checking in multiple files and for some reason one of the triggers fails, then you have some files in and some not which will break the build. In AccuRev, triggers fire once per transaction so either the whole operation succeeds or fails and there are no broken builds.

Productivity gain: 60 minutes per developer.
Frequency: once per week.

Virtual File System Slows Down Builds (A10)

While it is absolutely true that there are advantages to the virtual file system, and they are discussed later in this paper, with today's hardware infrastructure, the result is slower builds. If you are using snapshot views, then this does not apply, but then the advantages of the virtual file system do not apply.

AccuRev does not use a virtual file system, much like snapshot views. AccuRev uses all writeable files by default.

Productivity gain: 30 seconds per developer.
Frequency: 10 times per day.

ClearCase Does Not Guarantee 100% Reproducibility (A11)

For more information about AccuRev's ability to provide 100% reproducibility, please see the paper "The Timesafe Property", available at <http://www.accurev.com/accurev/info/timesafe.html> .

Productivity gain: 30 minutes per developer.
Frequency: once per week.

Reliability Problems (A12)

ClearCase is a very sophisticated and complicated system. It relies heavily on NFS, which is one of the least reliable methods for hosting databases and includes a virtual file system which modifies the kernel of the operating system. Configuring ClearCase to work well requires a great deal of training and experience. As a result, ClearCase is often down.

Productivity gain: 60 minutes per developer.
Frequency: twice per month.

MultiSite Replication Model is Not Ideal for All Usage Patterns (A13)

ClearCase MultiSite was designed in the early nineties when bandwidth was much more expensive and unreliable. As a result, MultiSite was built with the assumption that the normal usage pattern is infrequent synchronization between sites. In turn, that means that mastership is important and branches can only be updated by one site at a time. Thus, even if groups are working on the same project, each site must have its own branch for that project. Multiple branches means more merging, the merging is generally done by somebody other than the developers involved, and there is a large time delay between sites getting each other's changes.

AccuRev's current replication solution provides an always-up-to-date read-only replica which provides the same level of performance for read operations as MultiSite. All write operations are performed against the master repository. Thus, write operations are only as fast as the connection to the main repository. However, write operations are done less frequently than read operations and there is no need to wait for updates from other sites.

From a usability perspective, there is no difference in usage models between using AccuReplica and not using it. There is no need for branch mastership or extra merging and all merges can be done by the developers involved in the conflict.

Productivity gain: 1 minute per developer.
Frequency: 2 times per day

Check-ins are Slow (A14)

ClearCase operations are just plain slow. Check-ins are particularly slow.

Productivity gain: 1 minute per developer.
Frequency: 2 times per day

Check-outs are Slow (A15)

ClearCase operations are just plain slow. Check-outs are particularly slow. AccuRev uses an all writeable file model by default and thus no check-outs are required at all.

Productivity gain: 1 minute per developer.
Frequency: 2 times per day

Finding Changes is Slow (A16)

Figuring out which files you have changed and which files need merging view-wide takes a long time in ClearCase. AccuRev is much faster.

Productivity gain: 1 minute per developer.

Frequency: 2 times per day

UCM Specific Issues

Have to Rebase to See Changes (A17)

When using UCM, the only way for developers to get changes from other team members is for a new baseline to be created and to do a re-base operation to that new baseline. Baselines are usually either created by an admin or created periodically (say once per hour) as needed. In either case, there is a significant delay.

Productivity gain: 1 hour per developer.

Frequency: 2 times per day.

Activities Lose Individuality During Deliver Operation (A18)

When delivering activities from one stream to another, the result is a new activity in the destination stream with an auto-generated name which is a roll-up of all of the delivered changes. While it is possible to see the details about the activities that were delivered, it is not possible to then selectively deliver those activities to another stream. In AccuRev, change packages retain their individuality when they are delivered from stream to stream.

This problem is somewhat hidden by the fact that setting up a stream hierarchy in ClearCase is difficult and thus most organizations have a very shallow stream hierarchy compared to AccuRev customers.

Productivity gain: 15 minutes per developer.

Frequency: once per week.

ClearCase Advantages

Dynamic Views (C1)

In ClearCase, when you are using dynamic views, if you change the rules that determine the content of your view, the affect is instantaneous. You can immediately start using your view. In AccuRev, you need to do an update and wait until files are physically brought in, removed, renamed, or moved. While this is still fast in AccuRev, it is not as fast as ClearCase.

Productivity gain: 10 minute per developer.

Frequency: 2 times per week

Build Auditing (C2)

If you are using dynamic views and you are using clearmake, then your builds are audited. That means that you can easily find out the exact bill of materials that went into building an object, library, or executable. You can also run difference reports on the bill of materials associated with two executables and quickly and easily find out exactly what changed in the sources between the two executables. This is useful if you have one executable which is doing the right thing and another one which is not and you are trying to track down the change that caused the problem.

clearmake is not supported for .NET and thus this advantage is not available to .NET developers.

Productivity gain: 30 minutes per developer.

Frequency: once per week

Faster Writes due to Disconnected Bi-directional Replication (C3)

On the plus side, all write operations are done against the local replica and are thus as fast as the local network allows

Productivity gain: 5 seconds per developer.

Frequency: 3 times per day

Disconnected Bi-directional Replication (C4)

Also, the only thing that is interrupted if the connection goes down is synchronization, which is also true for AccuRev.

Productivity gain: 10 minutes per developer.

Frequency: once per week.

Views Can Span Depots (C5)

In general, there is less need to create lots of repositories (VOBs) in AccuRev. However, if you have a need to use multiple repositories, you will need to create a workspace per repository that you want to access. In ClearCase, a single view can span multiple repositories.

Productivity gain: 5 minutes per developer.

Frequency: once per day.

Scripted Private Branching (C6)

Some shops invest scripting to provide a more sophisticated private workspace model in ClearCase than what ClearCase provides out of the box.

Productivity gain: 5 minutes per developer.

Frequency: once per day.