

2009

PACIFIC NW SOFTWARE
QUALITY
CONFERENCE



MOVING
QUALITY
FORWARD

OCTOBER 27-28, 2009

Conference Paper Excerpt

From the

CONFERENCE
PROCEEDINGS

Permission to copy, without fee, all or part of this material, except copyrighted material as noted, is granted provided that the copies are not made or distributed for commercial use.

Reconfiguring the Box:

Thirteen Key Practices for Successful Change Management

Leesa Hicks
leesa.hicks@tek.com

Abstract

Existing processes help organizations work in an organized and predictable way, but they also provide resistance points for improving how they work. Tektronix “standardized” on IBM Rational ClearCase for configuration management years ago, but because it had no built-in process automation, each product team developed their own customized processes and automation for using ClearCase.

Ironically, the lack of process automation in the early versions of ClearCase was actually part of its initial appeal. However, both ClearCase and development organizations have matured greatly since then, and ClearCase now provides automated processes with Unified Change Management (UCM). More recently, Tektronix adopted IBM Rational ClearQuest for change tracking, which adds yet more value when integrated with ClearCase UCM.

Even with all the new capabilities and automation that UCM provides, motivating development teams to change how they use ClearCase has been challenging, in spite of the issues they have been experiencing with their existing ClearCase usage. They still need to be convinced that the value added by adopting UCM outweighs the cost of changing how they work, even though they are using the same tools. This paper describes how we have helped development teams break out of the constraints of their existing configuration management processes and improved their development processes with successful adoptions of UCM, including thirteen key practices used to bring about these beneficial changes.

Biography

Leesa Hicks is a principal software engineer in the Software Engineering Services group at Tektronix Instruments in Beaverton, Oregon. She has worked in a number of roles in software engineering throughout her career, including: software developer, project lead, technical lead, technical specialist, and manager. Leesa has also worked in a number of industries as well: banking, manufacturing, IC design automation, automated test equipment, software sales support, and IT. A common thread through all these roles and industries has been her work in process improvement. Leesa is currently responsible for ClearCase and ClearQuest strategy and development and ClearQuest operations at Tektronix.

Leesa has an M.S. in Computer Science from the University of California at San Diego. While at Tektronix, she has been nominated for a Technical Excellence award, and she has received two Customer Excellence awards.

Copyright Leesa Hicks August, 2009

Background

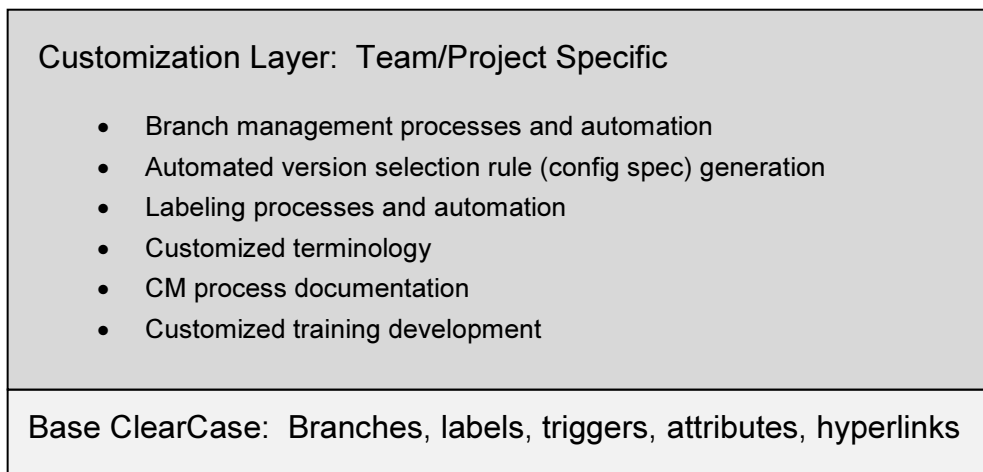
IBM Rational ClearCase and ClearQuest are the corporate-supported configuration management and change tracking tools at Tektronix. As a member of Software Engineering Services group, I work with teams to resolve issues they are having with using these tools. These interactions provide opportunities for introducing teams to Unified Change Management (UCM) when they have significant problems that can be solved by using UCM. The following sections provide a brief comparison of the effort required to adopt Base ClearCase versus UCM.

Configuration Management

There are many definitions of configuration management available from numerous sources. This paper concentrates on configuration management in terms of version control, configuration identification (baselines or labels), change process management, development process management, build and release management, defect and change tracking, and change status reporting.

Base ClearCase

When Tektronix first deployed ClearCase, it provided a set of mechanisms for managing concurrent, distributed development, but no automation. This is referred to as *Base ClearCase*, and using it requires developing a significant customization layer on top of ClearCase. It typically takes three to six months for a team to develop the initial customizations needed to adopt Base ClearCase, requiring maintenance and enhancements as needs change. The following diagram illustrates the types of automation and infrastructure necessary to use Base ClearCase. Note that this effort is not generally transferrable between teams, and possibly even between projects.



As a quick example, in order to work on a branch in ClearCase, you must create a branch type, and then you must create a configuration specification (version selection rules) to make changes on that branch. If you are supporting this for a team, common customizations include developing some type of branch management policies and processes, automating branch creation to support the team's development processes, automating the generation of configuration specifications, automating merging between branches to integrate work together, developing labeling policies and processes, and automating the creation of label types and label creation to support the build and release process.

Because Base ClearCase customizations tend to be unique between teams, there is a great deal of overhead that is repeated throughout the enterprise. Each customization incurs support overhead, and product updates can break these customizations. In addition, when developers move from one development team to another, they need to learn the new team's processes and terminology for using ClearCase. This can be significantly different from their previous team's processes, and may even rival the overhead of learning a completely different configuration management tool.

Unified Change Management (UCM)

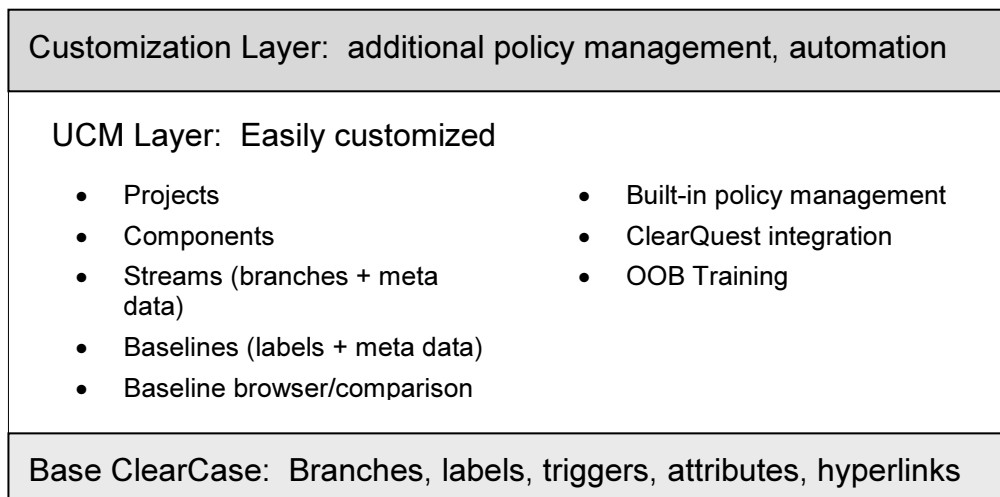
UCM is a layer on top of ClearCase that automates most of what is needed to use ClearCase out-of-the-box, and it provides more flexibility with organizing and managing development work. UCM models configuration management in familiar ways for development teams, using projects to organize significant development efforts, components to organize sets of related files versioned as a whole with baselines, and activities to track change sets for each task a developer works on.

Components generally consist of the significant architectural pieces of a system, and they are a key mechanism for providing flexibility with organizing development. Loosely-coupled components make it possible to define projects using producer-consumer relationships, for example, rather than the classic release model that treats all the files comprising a system as a single entity or configuration.

For example, one UCM project may be used to modify the user interface component, with read-only access to the components required to build it, and another project may be used to modify a component or set of components that the user interface is dependent upon, but which does not need access to the user interface component itself. This makes it possible for the user interface team to manage its own development, integration, testing, and releases of the UI, independently of the other project, and vice-versa. This flexibility is useful when release cycles differ between a system's architectural pieces or components, and any case where specific team members work exclusively on one portion of a system.

UCM can also be integrated with IBM Rational ClearQuest, which links each UCM activity in ClearCase to an actual defect, enhancement request, or any type of change request in ClearQuest. ClearCase information and operations are available from ClearQuest and vice-versa. For example, developers can view the change set for a defect (the list of file versions modified to fix that defect), and they can view the changes they made to any particular file to fix that defect. This integration also provides additional capabilities for change status reporting, which is particularly useful for build management and project status evaluation and reporting.

The following diagram provides an illustration of the types of automation that UCM provides, and there are many ways to customize it with built-in mechanisms. There may still be a need for some additional automation on top of UCM, but this is minimal compared to Base ClearCase.



With UCM, there is much more commonality between different teams and projects, so the overall overhead of using UCM throughout the enterprise is greatly reduced. Developers can move from team to team and project to project using UCM much more easily as well, because they only need to understand how this team or this project uses UCM rather than how this team or this project uses a completely customized implementation of ClearCase.

Configuration Management Advancements

This section describes the journey of three teams who have adopted UCM at Tektronix, including their challenges and requirements, as well as the results they have achieved with UCM.

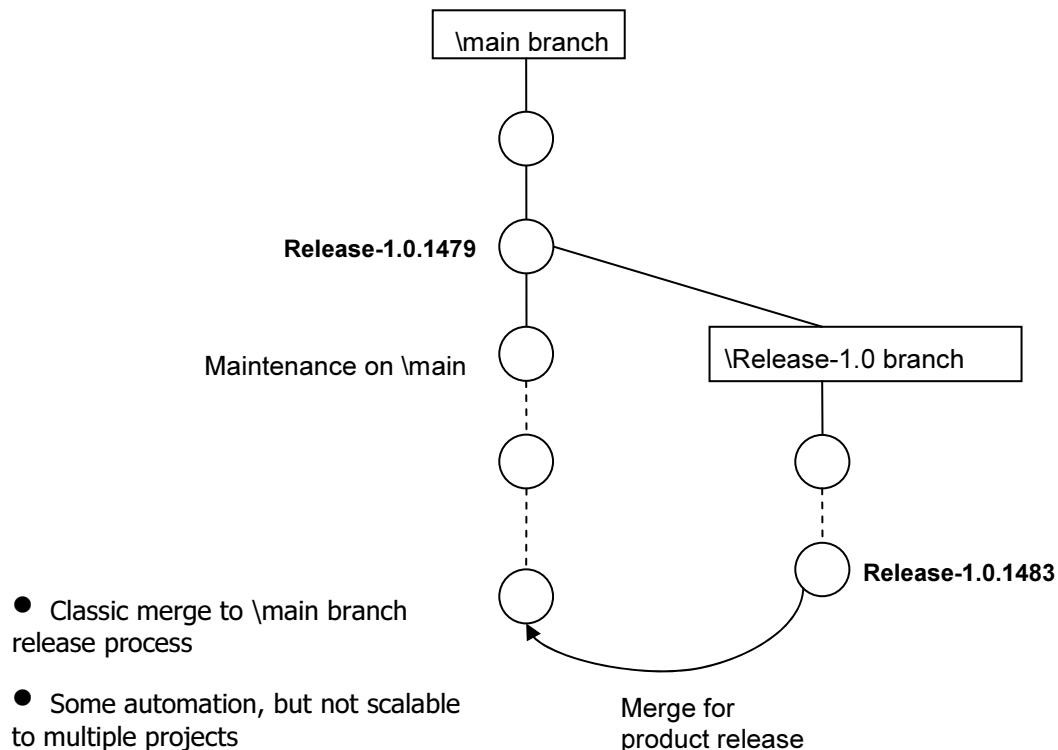
Red Team

The Red Team was the first team to adopt UCM at Tektronix. This team is often an early adopter of newer methods and tools, and they recognized the value of using UCM without the benefit of any other team's experience. They needed to change how they were using ClearCase primarily because they needed to change from working on one release at a time to working on multiple releases concurrently, although they had other issues as well. Here is a summary of their challenges and requirements:

Table 1: Red Team's Challenges and Requirements

Geographically distributed development team: ~20 Oregon team members, ~10 remote team members.
Developers need to work in isolation
Performance issues with merging, applying labels, opening the version tree browser
Difficult to identify changes made for a task
Needed to change from working on one release at a time to working on multiple releases

The following diagram shows the Red team's original branching process, which supported working on one release at a time while maintaining the existing release. This process would not scale to support working on multiple releases concurrently:



I started working with the Red Team several months before they needed to start working on multiple concurrent releases. Our options were for me to help them switch to using UCM, or they would need to develop some new processes and automation for using Base ClearCase. I worked with the key

stakeholders to understand their needs (see Table 1), and then provided some training and demonstrations of UCM. They did not have time to meet with me more than once a week, so we met weekly for an hour over seven months discussing UCM, identifying their use cases, determining whether UCM would meet their needs, and then planning the actual UCM deployment. Because they had so little time to work on this, I developed the entire configuration management plan for them, but I would have preferred that this be more of a joint effort.

Use of the ClearQuest integration was the only portion of the adoption that made the Red Team nervous, because UCM requires that you associate every change made with a ClearQuest activity. They were used to using ClearQuest to track defects and enhancement requests, but they would also need to track everyday tasks as well. There was no option here other than to provide assurances that the integration would work well for them.

In order to minimize the impact to the team and to manage the effort required by Software Engineering Services, we completed the UCM deployment in phases. Their deployment required restructuring their source code repositories, so we did this prior to their actual UCM adoption. We provided formal training the week before the UCM deployment, implemented the UCM configuration over a weekend, and I sat in their work area for the first two weeks of their UCM adoption to provide direct support.

The switch to UCM went rather smoothly for this team, and here is a summary of their results:

Table 2: Red Team's Results

Met their goal with developing multiple concurrent releases. Have supported as many as six concurrent programs
Some initial pushback from developers on the activity based tracking with ClearQuest, but they quickly came to see the benefits. No more manually recording what they had changed
No more version selection rule (config spec) confusion, since these are auto-generated by UCM
Remote team able to integrate work into the project with UCM remote deliveries. → Eliminated nightly merge scripts and manual conflict resolution for remote development
Eliminated Base ClearCase customizations – reduced script maintenance
Improved performance → UCM rebase and deliver (merge operations) much faster than custom scripts → Reduced nightly build time by ~20 minutes with incremental baselines
Have more options for sharing work

As you can see from Table 2 above, the Red Team's UCM adoption was a clear success for them, and they quickly concluded that it was the right thing for them to do. As predicted, they even appreciate the ClearQuest integration in spite of their initial apprehension. They have also become strong advocates for its usage, and they have been very helpful in assisting with other teams considering adopting UCM.

Purple Team

The opportunity to work with the Purple Team came from a change agent, Doug, who attended a presentation of the work we did for the Red Team. Doug obtained management's agreement to perform an assessment of whether UCM would be a good solution for their configuration management needs. Here is a summary of the Purple Team's issues and requirements:

Table 3: Purple Team's Issues and Requirements

Geographically distributed development: ~26 in Oregon, ~26 in India
Team members at each site use different configuration management processes. The Oregon team uses a home-grown DevRel automation system with ClearCase, but the India team is unable to use DevRel because it was designed for a co-located team. Neither group understands the other's configuration management processes.
DevRel is supported primarily by one individual, with one backup, requiring ~12 hours per week to

maintain (this primary individual has since left the company)
DevRel tracks change sets for Oregon team only; India work contributed by many developers is merged in as a single change set
Need to reduce the overhead of managing concurrent, distributed development
Developers need to work in isolation and project work needs to be managed in isolation
Oregon responsible for integrating India work into the overall project, even though most of the team members were in India, ~3-5 hours per week overhead in Oregon
Need to be able to determine what is being released. Sometimes defects fixed in one release are lost in subsequent releases

Doug solicited input from the Red Team to learn why they switched to UCM and what issues they have with using it. The Red Team communicated that they couldn't manage concurrent releases without having adopted UCM, and the only negative thing they had to say was that the developers initially didn't care for having to track all their tasks in ClearQuest, but that they had quickly adjusted. One of the comments from the Red Team was that it had been years since they had looked at a config spec, which for anyone who has used Base ClearCase, understands the difficulties with creating and maintaining config specs.

After considering the benefits and the cost of changing, the Purple Team decided to deploy UCM in phases. At the time of this writing, we have completed some preparatory work for adopting UCM for the entire Purple Team, and we have completed the first UCM adoption for one of the development teams within the overall group (phase 1). This team has ~16 members in India and ~4 in Oregon. Although most of the team works in India, all of the project integration work was originally being done in Oregon.

Doug adapted the Red Team's configuration management plan as a starting point for the Purple Team, and one of the Oregon software leads, Byron, pushed for this phase 1 team to adopt UCM. Byron saw value in having the ability for this team to manage their own project integration, and recognized UCM as having value for implementing a well-defined configuration management process.

I worked with Byron and key stakeholders in India to guide them through the adoption process similarly to what I did for the Red Team: education, demos, deployment decisions, etc. This was more challenging than for the Red Team because of the time differences between India and Oregon, as well as language and cultural differences. We provided training for the team just prior to the initial adoption, and we deployed UCM for this phase 1 team over a weekend, including re-structuring their source code repository.

With the phase 1 team's UCM adoption, we have accomplished the following:

Table 4: Purple Team's Phase 1 Results

The entire phase 1 team now uses the same configuration management processes
The India project leads now manage project integration and releases independently of Oregon
Oregon team members are able to contribute their work using built-in support for geographically distributed development
Change sets for each activity are tracked for all team members
Have the ability to determine what is in a release/baseline
Provides additional control over the integration, test, and release cycle

This adoption has also been successful, as you can see from Table 4, although it is fairly recent and the developers are going through the normal pushback on having to track all their tasks in ClearQuest. I expect that they will quickly get used to this, just as the Red Team did. Software engineering management is also happy with the results and is committed to adopting UCM for the remainder of the Purple Team later this year.

Green Team

The Green Team was experiencing some significant performance issues with using ClearCase, and they had started investigating options for switching to a different configuration management tool. I had an opportunity to explore their issues with two of the team leads and discovered that it was their own customization of ClearCase that was causing most of their performance issues. They were open to considering UCM, and here is a list of the Green Team's issues and requirements:

Table 5: Green Team's Issues and Requirements

Geographically distributed team
Partially committed changes accessible on the \main branch
Excessive overhead with merging files between branches
Difficulties integrating the remote team's work into the overall project
Need to reduce the configuration management support burden

Once I was able to educate the team leads on how UCM could solve these issues, they were willing to consider UCM but they were still skeptical. I completed some performance tests to compare to what they had done with Base ClearCase and Subversion, and this helped convince them that UCM was worth trying. I developed a configuration management plan for them, and in this case we deployed UCM for a new development project rather than re-configure work already underway.

Unlike the Red and Purple Teams, the Green Team prefers to use the command line for using ClearCase, and they were not interested in integrating ClearCase UCM with ClearQuest. They also don't care about UCM change set tracking per se, only in that it helps them check in files that they have modified for a task and automates merging. They had fixed ideas for how they wanted to work, so it was challenging at times to try to present other points of view and other things to consider. They prefer to write a script, for example, to filter out noise in textual output, rather than use a graphical user interface that already does this for them, as well as other benefits.

One thing that I was unable to convince them of was to take advantage of the remote development support that is built into ClearCase UCM. When they were using Base ClearCase, they had a lot of difficulties with merging in work done by the India team members, and this is admittedly a challenge with Base ClearCase. However, they insisted that the India team use the ClearCase remote client so that they work directly in the source code repository hosted in Oregon. Recently, this has become an issue due to WAN connectivity and bandwidth problems causing performance issues in India, so India will be changing to use native clients and the remote development support in UCM as I had originally tried to recommend.

Here is what we have accomplished with the Green Team:

Table 6: Green Team's Results

Geographically distributed development support
No more partially committed changes on the integration branch
The overhead of identifying what to merge was reduced from 20+ minutes to seconds
Individual developers now responsible for integrating work into the overall project; eliminated difficulties with integrating remote team's work
Software engineering services now providing most of the ClearCase support

The Green Team was trying to avoid defining aliases and writing scripts as wrappers around ClearCase operations, but they have implemented some of this for UCM, anyway. With their Base ClearCase usage, they had to reproduce any problem they were having outside of their ClearCase customizations before being able to seek support, which was rather time consuming for them and understandably frustrating. With UCM, they are able to obtain support from Software Engineering Services as needed, which is a very welcome improvement for them.

The Green Team has stated that they can't imagine how they managed development before adopting UCM, and they will be using it for all new projects going forward.

Adoption Cost Summary

The following table summarizes the effort required by Software Engineering Services to deploy UCM for the three adoptions. As you can see, the effort required to adopt UCM, even for the first team at Tektronix, is significantly less than the typical 3-6 month effort required to adopt Base ClearCase.

Table 7: Adoption Cost Summary

Category	Red Team	Purple Team	Green Team
Relative complexity	High	Medium	Low
Education, planning, and documentation	8 days	5 days	2.5 days
Source repository re-structuring	14 days	1 day	.5 days
Configure UCM	1 day	.5 days	.5 days
Deliver Training	2.5 days	2.5 days	2 days
Post-deployment support	7 days	5 days	5 days
Totals	32.5 days	14 days	10.5 days

Note: The Red Team's UCM adoption was the first one within Tektronix, so there was more effort involved overall to develop some basic infrastructure. We have been re-using plans, documentation, tools, etc. to reduce the effort for subsequent adoptions.

Thirteen Key Practices for Success

This section summarizes the primary activities and approaches that have made these UCM adoptions both possible and successful. Many of these topics will be familiar to anyone who has experience with trying to change how people work. Even though this paper focuses on improving configuration management processes, these practices apply no matter who is involved or what improvements are needed.

Listen and Understand

Since I don't work in the development groups directly, it is extra challenging to initiate changes in the way the development groups use ClearCase. What do I know about their problems? It doesn't matter to them that I have worked in software engineering my entire career. It doesn't matter to them that I have worked with ClearCase for over 15 years. It doesn't matter to them that I have years of experience with helping teams adopt ClearCase/ClearQuest/UCM. What gets their attention is whether I really understand their needs and their problems.

When I have the opportunity to work with a team, the first thing I do is have them tell me what their issues are, what they would like to see happen, and then repeat back to them what I think I heard. I don't do anything else until they see that I understand how they currently use ClearCase, as well as understanding the problems they need to solve. This is essential to establishing a good working relationship with the team.

Seek out Early Adopters

When I started at Tektronix, all the development teams were using Base ClearCase, and they didn't seem interested in changing how they use it. It is always difficult to change how people work, and this doesn't vary no matter how beneficial the changes are. Getting that first team to change is always the most difficult to accomplish, because everyone wants a successful example of a UCM adoption at the company to help in their decision making process.

So how do you get that first team? For anyone experienced with promoting process improvement, and consistent with my experience, this has always been an early adopter: individuals or teams that

understand the benefits of changing how they work compared to the cost of changing. In this case, two things needed to happen, a team whose configuration management requirements were changing, and a team that was able to understand that UCM could satisfy their requirements.

One Successful Adoption Leads to Others

Since the Red Team has adopted UCM, it has been much easier to discuss a possible UCM adoption with other teams. When the Purple Team was considering UCM, they met with two of the Red Team members to learn about their experience, and they recommended UCM wholeheartedly. The Red Team's software lead has been a continuing source of help with other teams as well. He provided a copy of their build scripts for the Purple Team to use, and was willing to help adapt them for the Purple Team's needs.

At the time of this writing, we have two additional teams considering a UCM adoption. One team has been asked to switch to UCM by the Purple Team's software engineering manager after their successful pilot adoption with phase 1. Another team is planning to adopt UCM because their team members have used it either at Tektronix or other companies.

Prototyping

Prototyping has a number of uses, including: 1) testing the new environment and processes prior to deploying them to verify that they will work, 2) developing any tools or automation in preparation for the adoption, 3) testing the conversion process. Prototyping is particularly important when a team is nervous about whether the planned changes will actually work for them.

We used prototyping for the Purple Team's phase 1 adoption to develop a build script for UCM, starting with the Red Team's UCM build script. The Purple Team's original build script only worked with their customized DevRel system. Developing and testing the new UCM-based build script prior to the adoption helped ensure a smooth transition to UCM.

Pilot Program

Using a pilot program to reduce the risk of changing is also a useful tool. We did this with the Purple Team's phase 1 adoption because they were still nervous about switching to UCM, and this was a good way to reduce the problem size and solve some significant issues for them. We reduced the team's size from ~52 to ~22, the number of components from ~10 to 4, with a corresponding decrease in the number of projects. Since this phase has been successful for the phase 1 team, the Purple Team is planning to adopt UCM for the remainder of the team later this year.

Work at the Pace of the Adopting Team

Development teams at Tektronix have a policy of changing tools, methods, processes, etc. only in between projects or releases to minimize schedule disruption. When I work with a team on a UCM adoption, we work out what their target date, and then we determine whether we can meet that date. If we can't meet that date, it will be weeks or months before the next opportunity comes up for that team, depending on the size of their development iteration or how close they are to a release. We do everything possible to meet their first target date; if this is not possible, we work out a later date and make sure to meet that date.

Education

After verifying my understanding of the team's requirements and issues, I spend time explaining and demonstrating how UCM works to their key stakeholders. I spend as much time as the team needs to develop sufficient understanding. These individuals will be deciding whether UCM will solve their problems, and if so, how to customize it for their needs. I only educate the team on what they need to know, however. There are many technical details that are important for the deployment infrastructure but are not relevant to how the team will use UCM, and so I don't share these details with the teams.

This educational process is a critical piece of planning a successful deployment, both for understanding what to expect, but also for helping the rest of the team understand why they are changing how they use ClearCase. When developers complain about having to track activities with UCM, the key stakeholders can articulate why this is valuable to the project team.

Allow Time for Absorption

This is related to the initial education process, and there are two aspects to this. First, it is important to give the key stakeholders time to absorb how UCM works prior to making decisions about how to customize it for their teams. There are a number of terms and concepts that are new in UCM, and it is important to understand these, to understand what is possible in UCM, as well as understanding its constraints. When the stakeholders start asking about usage scenarios with UCM, this tells me they understand the concepts, and this is where the planning process actually begins.

Secondly, the entire adopting team needs time to adjust to the coming changes. Allowing sufficient time for everyone involved to absorb the plans to move to UCM, including the reasons for changing and how and when it will be accomplished, significantly reduces resistance to the changes.

Help Solve Real Problems

UCM is not a panacea, nothing is. It is not the answer for every configuration management use case. Honesty is required for establishing and maintaining good working relationships with the development teams, and being honest about what UCM can and can't help with, or shouldn't be used for, is an important for solving the appropriate problems.

Practice Detachment

I enjoy helping a team solve problems, because I obtain a lot of satisfaction with helping people work in more efficient ways. It doesn't matter to me whether that is within my own group or with other development teams, it is all rewarding. However, it is important not to get attached to the outcome, but instead focus on a supportive role and help lead the teams to their own solutions. Some teams need more guidance than others, and so I try to match their needs in this regard.

Sometimes it is necessary to agree to a solution that you know is sub-optimal, because the team needs to experience for themselves how it will work out. This occurred with the Green Team when they insisted on a single-site solution for their distributed team; they based this on their experience with Base ClearCase and were not open to a distributed solution with UCM. If I were attached to what I considered was the optimal solution, I could have pushed for the distributed solution and possibly damaged my relationship with the team. Instead, after working with their single-site solution for many months, they came back to me when they needed to resolve the issues that were predictable from my perspective, but not from theirs, and we are working together to deploy a distributed solution.

Planning, Planning, Planning

As with most activities, sufficient planning is a key success criterion, and a UCM deployment is no exception. A UCM deployment plan includes the requirements and issues to be resolved, the composition of the team, responsibilities, the project and stream policies to be used, and all the major tasks involved in completing the adoption. When the effort is large enough, the deployment is organized into manageable phases. Since creating the initial deployment plan for the Red Team, we have been able to re-use large sections of it for subsequent adoptions, thereby reducing the overall planning effort required for each deployment.

The UCM deployment plan has also served as a reference for teams considering a UCM adoption, so they can gauge the effort involved and as assurance that Software Engineering Services will provide adequate support for the deployment.

Training

It is critical to adequately train the entire team on how to use UCM prior to the adoption. We were able to use general UCM training for developers from an external vendor, and then we added 1-2 hours for developers for their team's UCM processes, plus 2-4 hours for project integration training for project leads.

Training is the one aspect of UCM adoptions that Software Engineering Services insists upon because the Tektronix culture eschews training - this is a case where practicing detachment is inappropriate. It is only possible to have a successful adoption with UCM if the developers and project leads are properly trained. Sufficient training also minimizes the support burden for Software Engineering Services, a cost

that is easy for development groups to discount because it is not visible to them, but it is essential for ensuring adequate support resources for the development teams.

Post-Adoption Support

For each team's UCM adoption, we have dedicated a Software Engineering Services support person to physically sit in the adopting team's work area for the first week or two of the initial adoption. The actual time we dedicate is up to the adopting team. This eliminates the usual overhead of requesting support and waiting for a response, and has proven to be highly valuable in avoiding frustration during the initial adoption.

Summary

To date, we have completed three successful UCM adoptions at Tektronix, which has eliminated three sets of customized Base ClearCase infrastructure and helped teams use ClearCase more efficiently and effectively. As of this writing, we have two additional teams considering UCM adoptions, and the Red Team is advocating using UCM for managing code that is shared amongst several teams.

The key practices described in this paper were instrumental to the success of these UCM adoptions. I have learned these practices throughout my career using others' experiences and wisdom with process improvement efforts, as well as learning from my own failures. Of course I also have a wealth of technical skills that I utilized to ensure that UCM is used and deployed using best practices. However, the most difficult aspect of working on these adoptions has been convincing teams of their value in helping solve their problems. Based on feedback from the adopting teams and the current demands for more adoptions, we have been highly successful with advancing configuration management practices using these thirteen key practices.