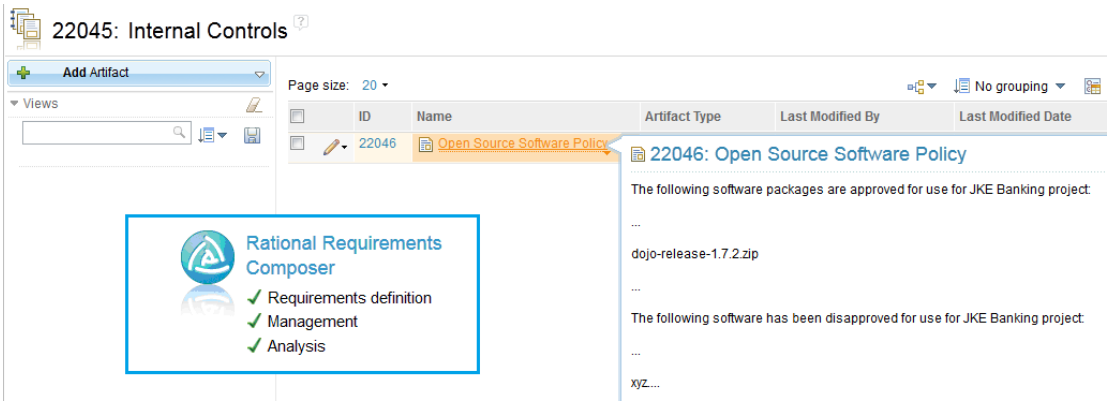
Productivity Through Open Source Policy Compliance

This article is part of a series on how Rational Collaborative Lifecycle Management (CLM) solutions support software development compliance.

Today the use of open source components is a necessary and desirable part of software development. In order to be protected from the various challenges and risks of using open source, organizations need policies and processes to ensure policy compliance. This article shows how a using Rational Team Concert and Black Duck's Protex product and organization can simply manage and guide developers' use of open source.

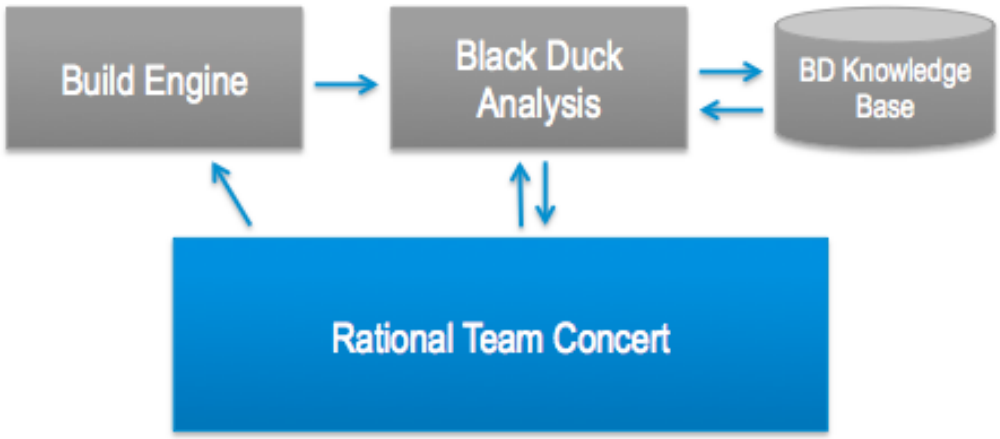
Consider the example of a company having great success in speeding time to market and accelerating innovation by encouraging its developers to use appropriate open source components to implement basic functionality, thus allowing them to focus resources on differentiating features. "Only develop what you can't download," half-jokes one development manager. Our company's senior management realizes, however, that they need to protect the delivery of software from various risks by only allowing the use of authorized open source components that conform to company policy.
Risks include IP (from uninformed use of certain licenses), security, quality and supportability.

Someone representing the management team could add this requirement to the Internal Controls requirements in Requirements Composer.

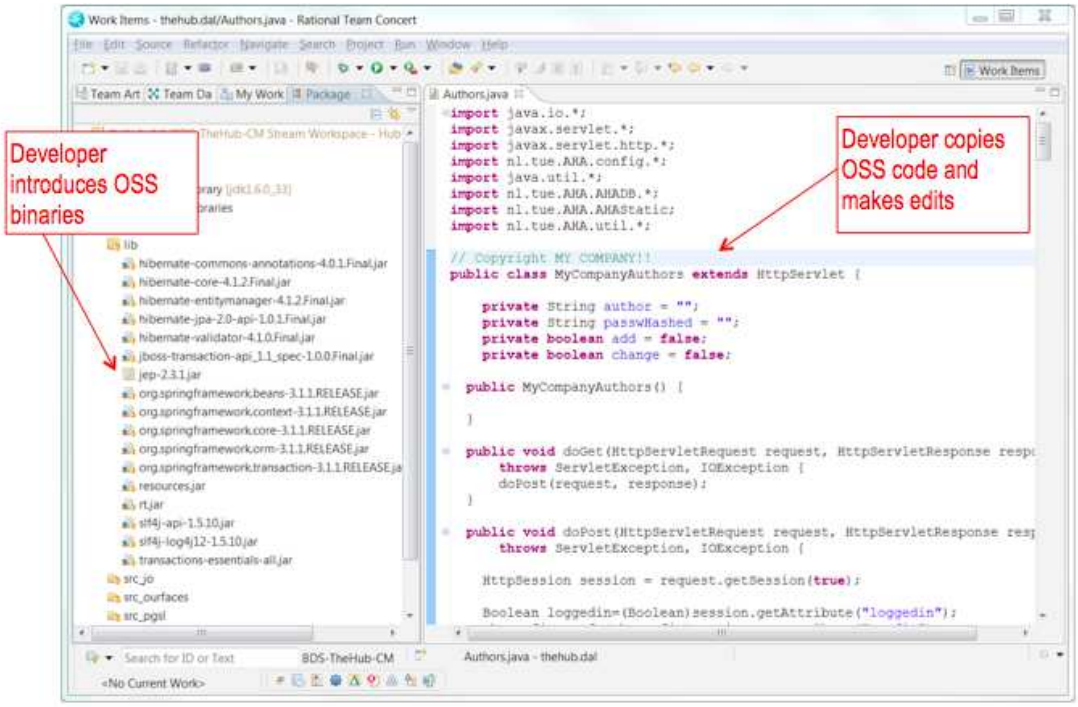


Let’s say a team of development managers and lawyers who have expertise in this area have worked out a policy. A simple open source policy might include a list of approved and disapproved licenses and the provision that developers may only use components with approved licenses. Many companies, for example, avoid the use of software governed under reciprocal licenses, such as the GNU General Public License or GPL.

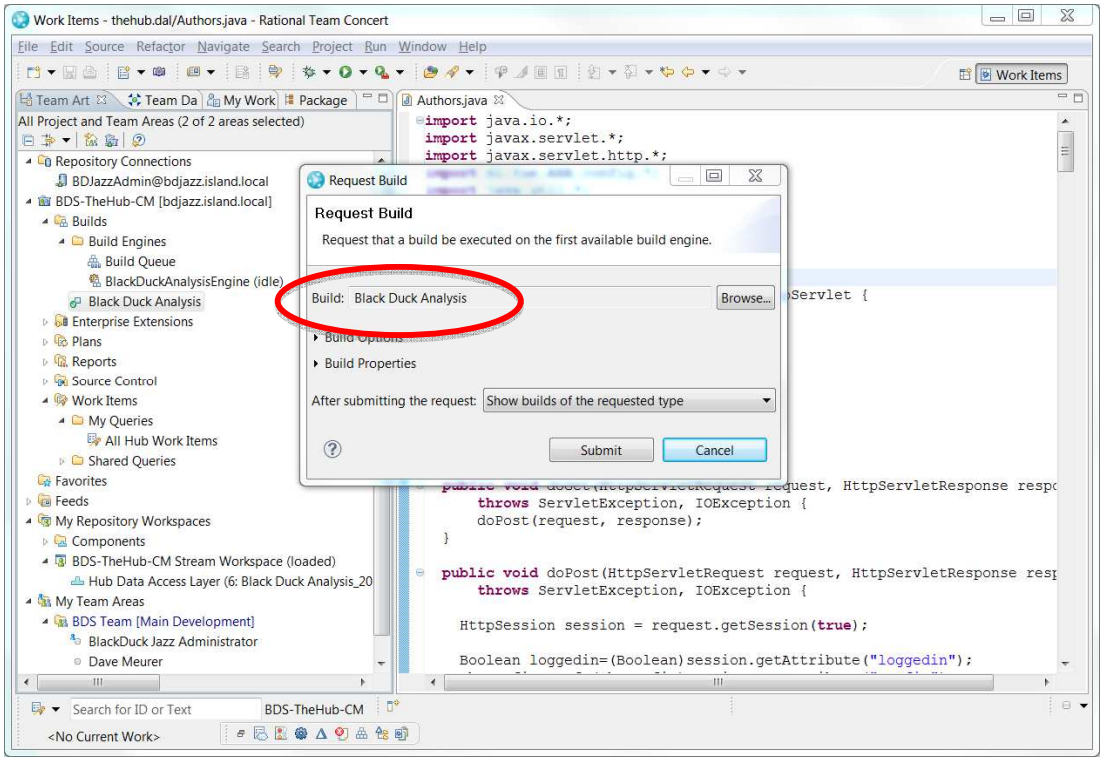
Now a process and tooling engineer could configure Black Duck to run automatically with the each Rational Team Concert build in order to assure compliance with the policy. In this example, let’s say that an issue that arises from a policy violation will not kill the build, but will be flagged to responsible developers for verification and remediation.



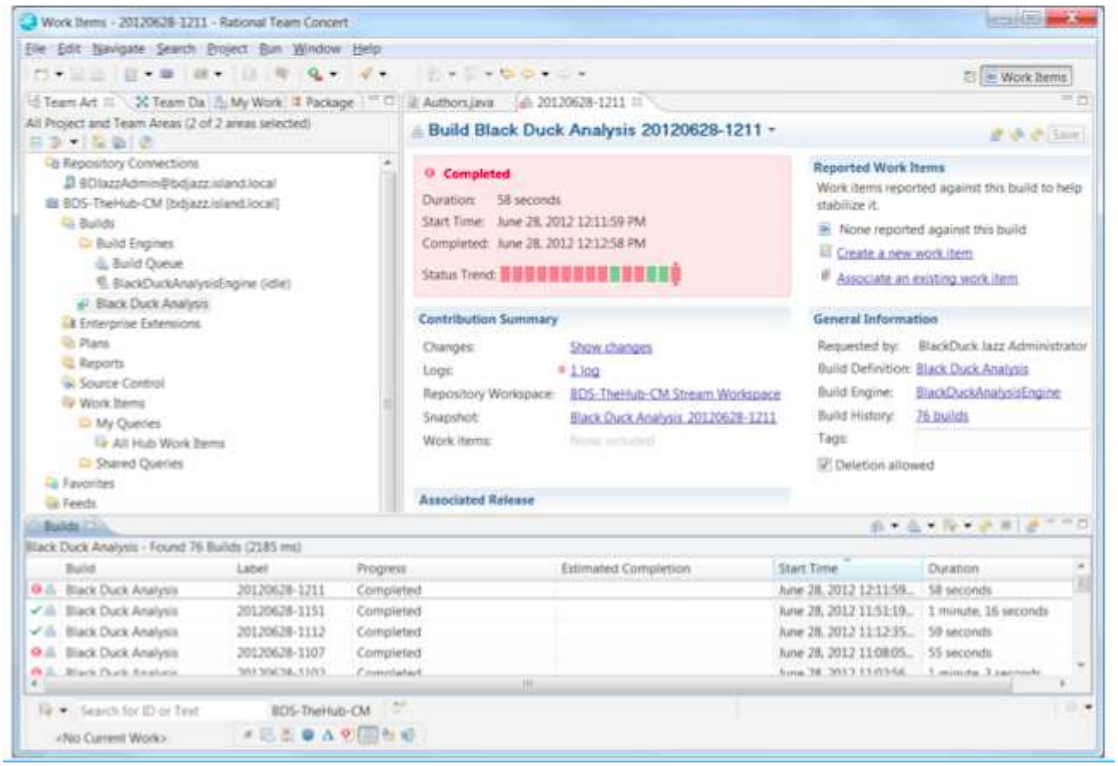
Now, a developer introduces some open source code into the code they are developing for a project. They utilize some binaries that are part of open source projects. They also copy and paste source code into their own, and perhaps modify the code. The nature of open source, and a big advantage, is that the source code is available and can be customized to the purpose at hand.



The Black Duck analysis runs automatically as part of the build process.

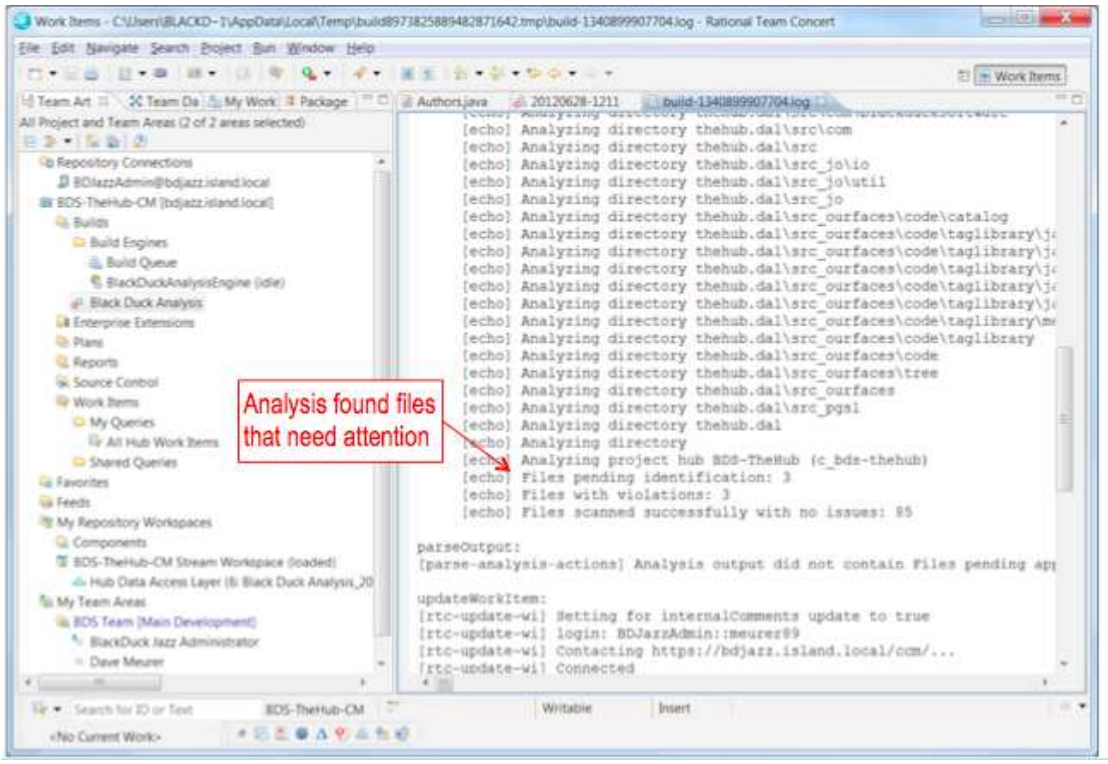


The analysis will create and audit report including a Bill of Materials, identify open source components in the code and flag potential license conflicts. Rational Team Concert tracks the progress of the analysis and indicates completion.

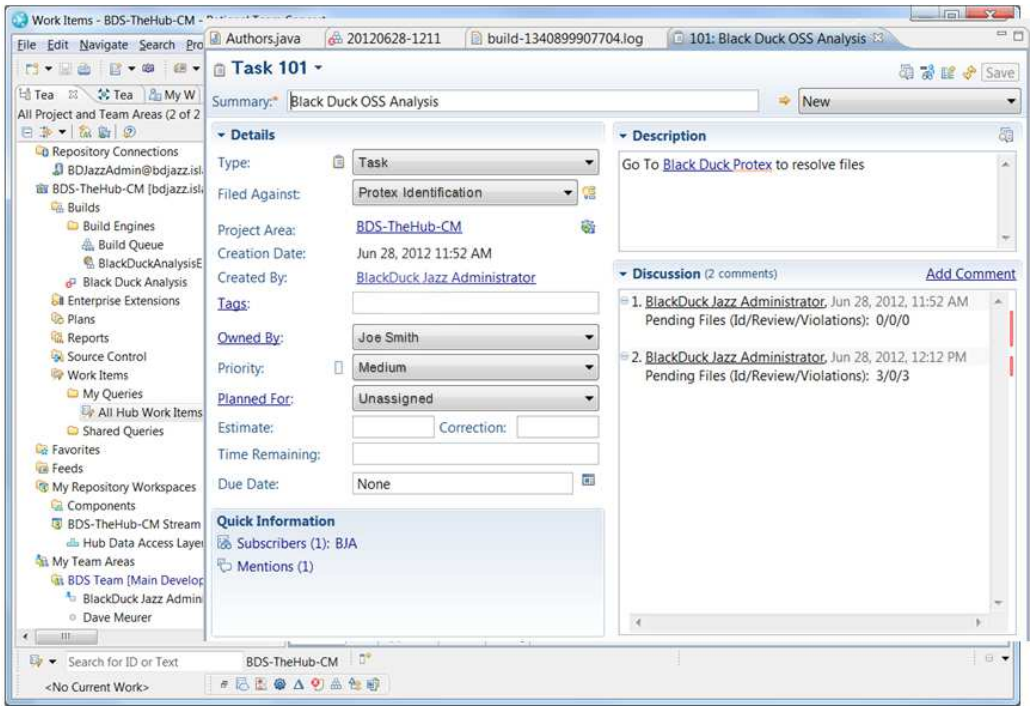


The Black Duck analysis is intergrated into the build environment. Here's a zoomed in view.

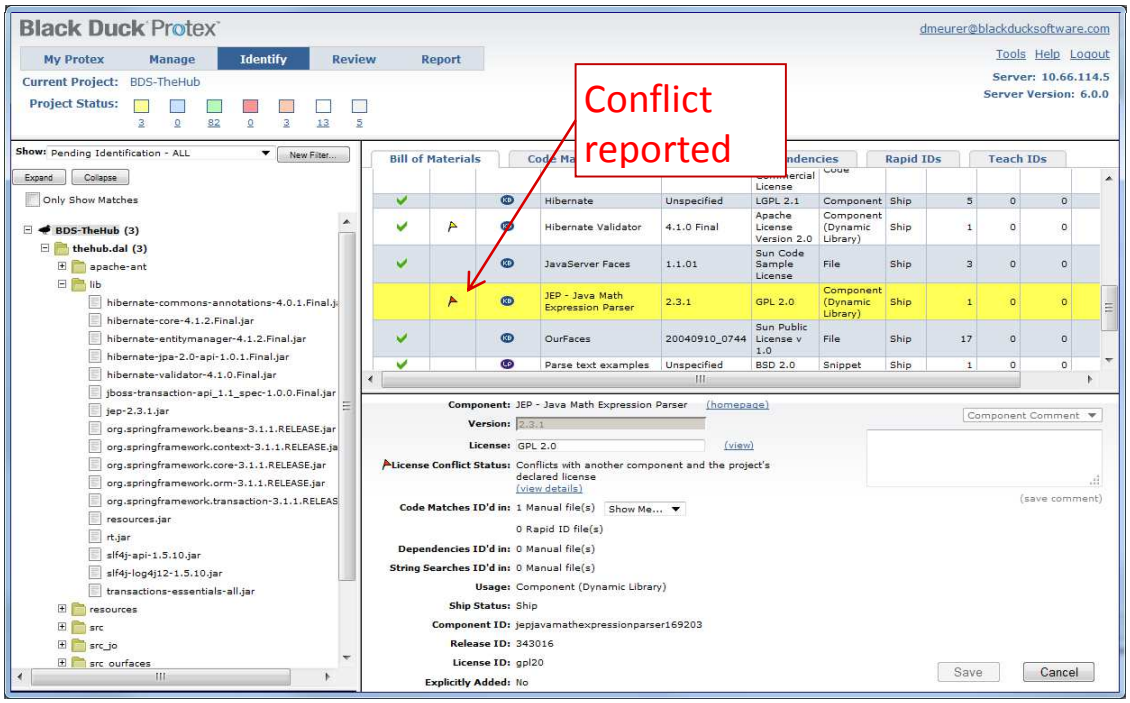
The build log shows that there are some files that need attention.



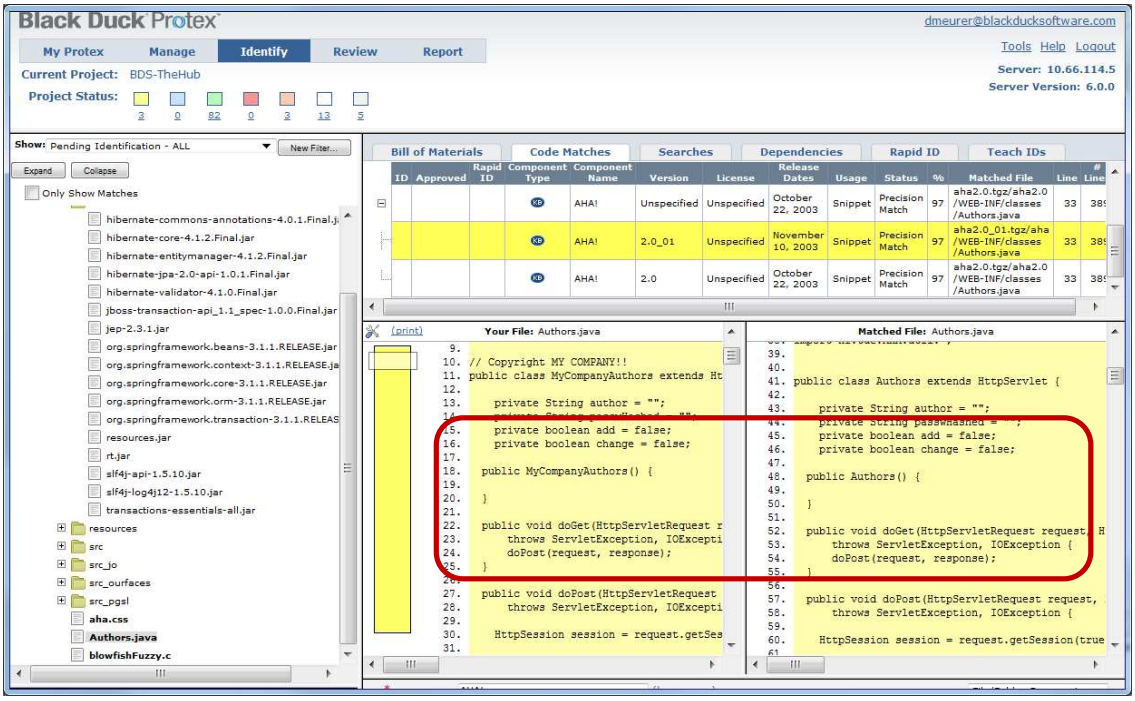
Rational Team Concert is configured to automatically create a work item for the engineer responsible for the code in question as well as a link to more detail on the issue at hand. Here's a zoomed in view of the task being created with Rational Team Concert.



The developer can see the potential license issue clearly flagged and described. In this case it appears that a component called JEP is licensed under the GPL license, clearly a conflict with the company's open source policy.

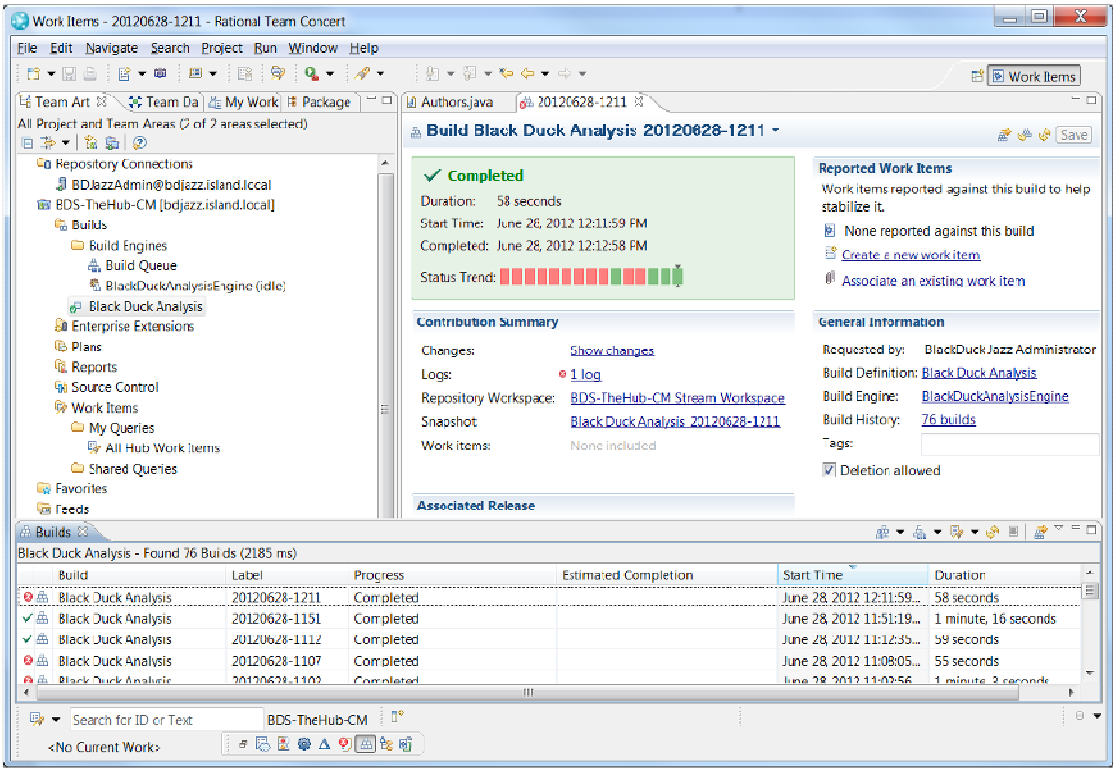


In many cases, such as this, the developer may not have intentionally included code from this component. There are many ways this can happen, for example, the developer may have borrowed the code from elsewhere inside the company. Perhaps, that implementation was only intended to be used internally, in which case the GPL-licensed component might not be an issue. To verify that the code is in fact code from JEP, the developer can look at a side by side comparison.



Clearly the code matches. The developer needs to go back and either create or find another component that will perform the same function. Once that is done they can close out the issue in Rational Team Concert and it should not surface in the net build.

Here's the result of the analysis that runs with the next build which verifies that the issue has been remediated.



In this article we looked at how a company can automate assurance of compliance with their open source policy using Rational Team Concert integrated Black Duck.