ALF Roles Viewpoint

This document represents Aldon's viewpoint of Roles within ALF. Aldon is a change management vendor that will enable their product to ALF.

Implementation Assumptions

Companies seeking to integrate their development tools into a cohesive product that supports all parts of the development lifecycle will likely adopt ALF. These tools will be selected from those vendors that have enabled to ALF, and will be licensed seperately, and at different times. The tools will not all be from the same vendor, but rather those selected as best for the company's needs.

The web services server will run on the company's intranet, and be accessable from the outside through the VPN. Thus one level of security is who is authorized to use the intranet. Tool integration, users and the roles they have, will be administered by the organization. ALF should run on any platform in use at the company.

Users and Roles

User credentials are for the purpose of identifying and authenticating a user of the system. A role is for the purpose of authorizing the user to perform various actions within a tool. A user may have many roles assigned, each specifying different capabilities. At any one time, a user is acting in only one role. This is done to prevent mistakes that could occur when a person has the authority to delete a Release in an Administration role, but not in the normal developer role.

Each tool in ALF could have its own notion of what the valid roles are. In some cases they may not even be using any role-based authorization. A couple of examples: Microsoft Project, and Eclipse itself. These tools don't necessarily have any idea of who the user is, let alone what role that user is acting in. And what you can do with them is anything the tool lets you do.

One thing that is necessary is that the integration between the tools be as seamless as possible. One thing that might be a big negative is for an event to be fired from one tool kicking off a service flow, and then have the tool handling the service flow reject it because the user didn't have the correct role when the event was created. The event is then lost, and the recovery flow has to now deal with the error condition, and somebody may have to manually handle it.

A Sample use case

A discrepancy report in the bug tracking system has been reviewed and a decision made to include it as a requirement for the next release. User Sam is operating in a bug

tracking “Approver” role when the event for ALF is activated to create an entry in the Requirements Management system. Does Sam have to have an “ALF user” role to make sure the service flow can run? What if he doesn’t? Should the tool check the roles before it creates the event? Worse yet, Sam does not have a “Requirements Manager” role in the Requirements Management system. This causes the Requirements Management system to reject the service flow, and the new requirement has the potential of getting lost.

It seems like there has to be something common among the roles the tools use in ALF that prevents this scenario from happening. A possiblity mentioned was having each tool in ALF accept the ALF role and override its internal role system, or to use the ALF role and assign the correct permissions to it that allow the action to take place.

A Proposal

ALF should have a roles and the security module that is general enough so that a role can serve any product that is registered in the framework. This module should allow an administrator to register tools and objects to which permissions are to be attached; organize those objects in a tree; define users and roles; define actions; associate actions, objects and roles; and query for a roles permission to do an action on an object. Roles are managed in trees and support permission inheritance.

Of course this implies extra work on a tool vendors part that wants to integrate their tool to ALF. They would have to use the ALF permissions model instead of their own. That might make it a lot tougher to get agreement.

ALF also needs a common authentication module used by all the tools. This could be any number of tools that implement a common service provider interface. But at a minimum, it should provide and example implementation in release 1.

Within the context of an enterprise that is integrating their lifecycle development tools, this makes sense. When you look at the similar toolsets from IBM/Rational and Microsoft, their tools integrate in a seamless manner. And they use a common approach to authorization and authentication. The one thing lacking in their approach is the ability for a customer to select tools from a different vendor that may be a better solution for them.