Some architectural changes to ICE

This post is a look at some architectural changes we're looking at for the ICE system, as we hit the limits of what we could squeeze out of the old architecture.

Ron Ward has just finished a major rewrite of lots of the application, designed to make it work on a central web server with multiple users, in addition to the 'classic' mode where everyone has their own ICE server running on their own computer. He's spent the last few months trying to get Subversion to do things it was clearly never meant to do.

ICE uses Subversion as a back-end version controlled data store. In the ICE classic mode multiple users work with checked-out working copies of a repository and hit 'Sync' to send their changes back to the server and get updates. Behind the Sync button is a fiendishly complicated bit of code that gets updates from the server, detects conflicts, tries to resolve them as gracefully as possible and provide a usable web GUI for the authors.

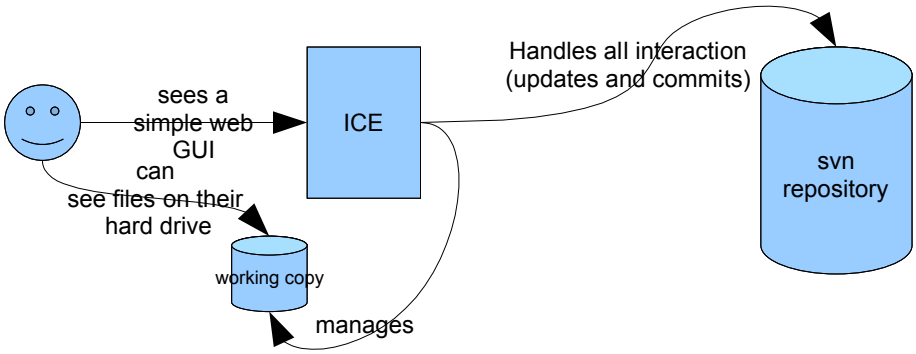

Figure 1: ICE Classic mode: each user has their own ICE application which looks after their working copy, ICE uses the Subversion protocol to synchronize everyone's work

Ron's big rewrite has lots of unit tests based on all the trouble we've come across (mis)using Subversion for the last couple of years so we're happy that it will be robust when running in classic mode.

But the new server version is a problem. If you have multiple users trying to access the same working copy all at once, then Subversion gets in the way – it starts locking files all over the place for example. One simple solution is just to put out a server version that doesn't allow distributed editing like ICE classic does, but our courseware authors really need the ability to manage large volumes of stuff on their own PCs as some courses are pretty big, with a lot of digital assets, while we want to have web access for reviewers and casual contributors to the same courses via a central web service.

So we're looking at a new server mode where ICE still has a working copy but it knows that it is the only user-agent who has it checked out so it doesn't need to do updates, it can just do commits. If all you want is a web based content management system then this will be all you need to install and it should run pretty well.

If you are following this technobabble then you'll be asking “but how does that help the ICE classic users work when there's an ICE server? That would mean that changes made on an

ICE client would never make it to the server!"

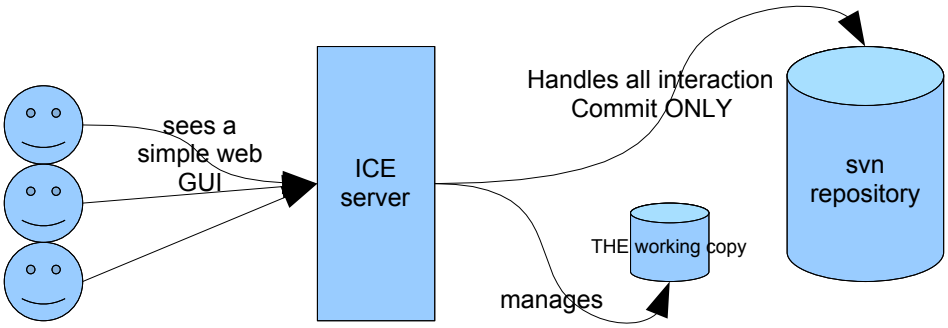

Figure 2: ICE Server mode: No subversion updates required as it is the only user-agent committing changes to the working copy

That's the tricky part – we need to create a new mode of operation for ICE where people want the benefits of the server version AND the classic distributed mode of working. In this mode the ICE application will work in a new 'client' mode. It will only ever get updates from the central repository. Any additions or changes won't be fed back to subversion directly – the ICE client will post them just like any other user into the ICE server.

This will require some more coding, but probably not as much as it would have taken to get the ICE server working any other way – and it opens up the possibility that we can replace Subversion and use a simpler version control system, possibly of our own devising in future. So a future model might have the ICE server acting not only as interface for humans but for other ICE systems.

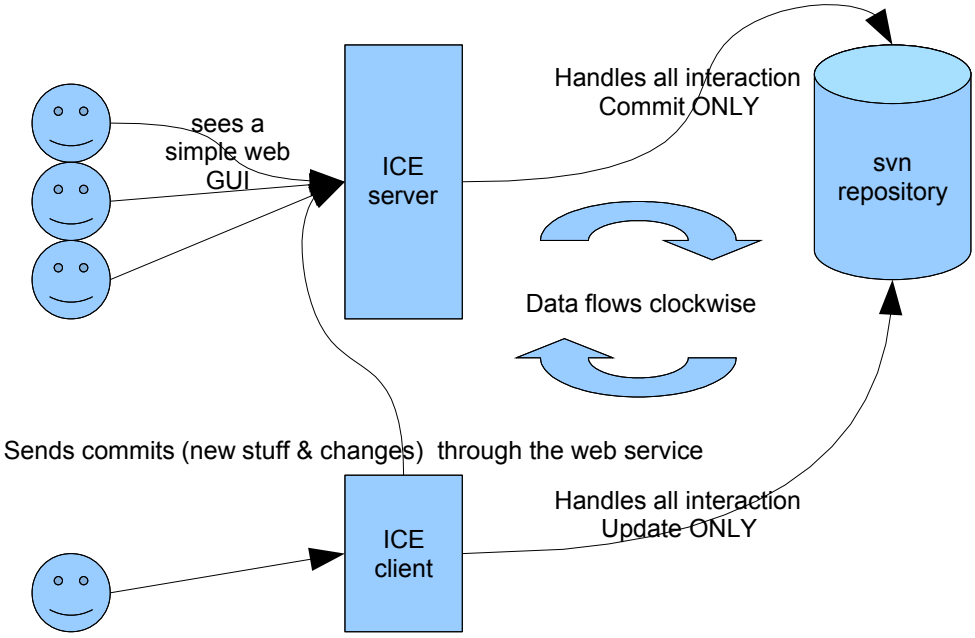

Figure 3: ICE Client mode: Users can update their local repository but all changes go via the ICE server. We will automatethis so it is seamless for users.

Having made this architectural decision we can press on with testing the ICE server straight away, even without making any changes to the client version. Here's the plan which we will roll through over then few weeks:

1. For the repositories which currently allow both server and classic access we turn off the ability for users to commit using ICE classic. If people want to check out their own copy of the content they can, as long as they post their changes back in through the server version manually.
2. We modify the ICE server so it now assumes that it has THE working copy and only commits changes – never updates – this will mean we can support multiple users with no dramas (that's the plan anyway).
3. We will make a new client mode for ICE which automate the process of detecting changes and posting them from the client version of ICE through the 'front door' of the server version pretty much like any other user. Updates will happen as they do now, from the subversion repository.