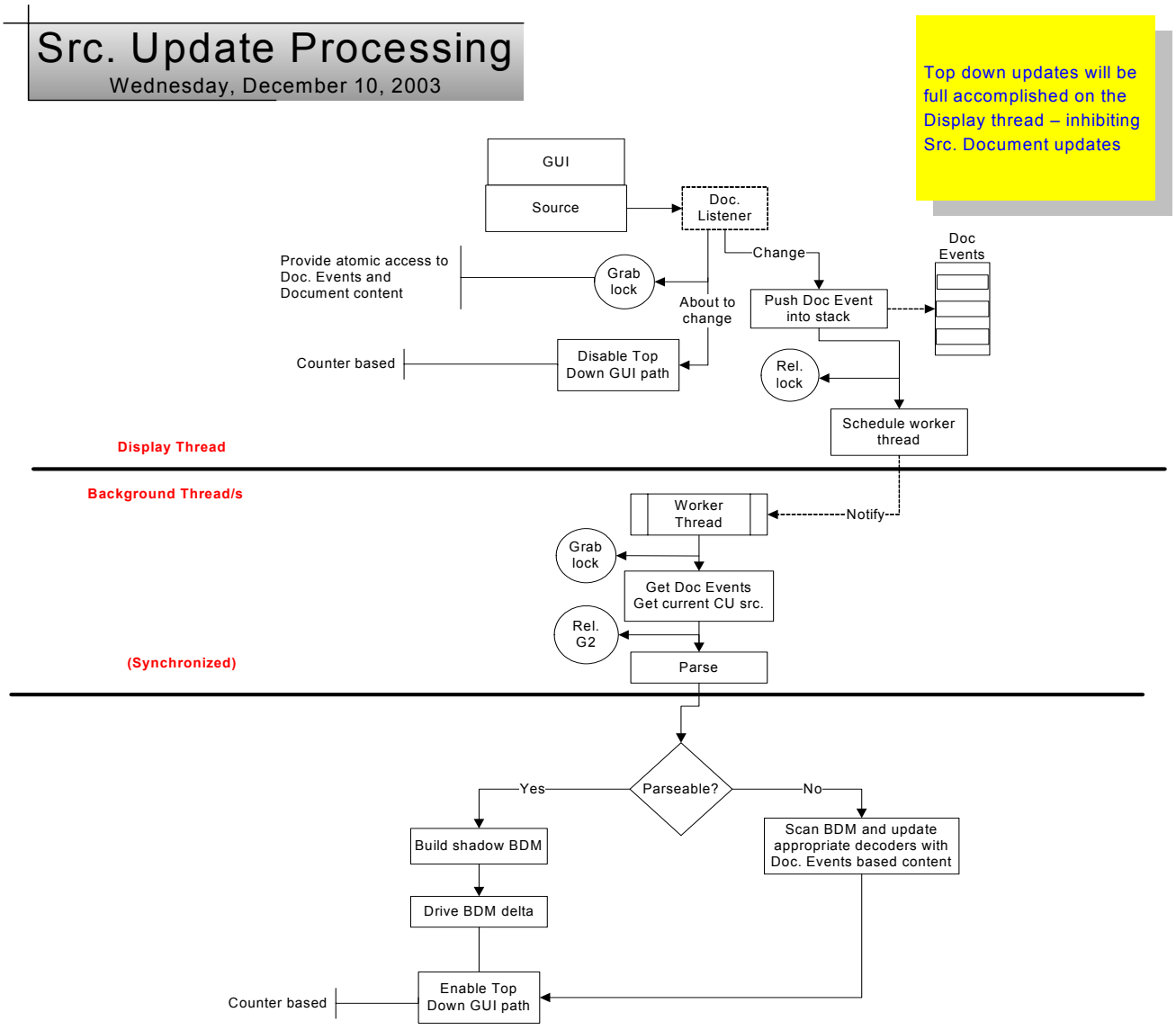
This document provides a high level description of the way VE in v1.0.0 M1 is going to process changes to the source code. This change is part of the Performance improvements effort.

Currently, when the source is changed VE (on the doc. change event) will determine the IJavaElement that was changed, take the content of that IJavaElement and relevant source, and schedule a background thread to process a snippet delta. Snippet delta was only implemented to deal with changes within a method.

This approach above is not scalable, as some analysis is done during the doc. change events (inhibit quick typing), and the content that is stored can be quite large for a large document.

In M1 we intend to change this process as following:



Top-Down updates (GUI to Source) will be completely driven on the Display thread – disabling the possibility that a user is able to update the source code while a Top-Down delta is being processed. (CodeGen has to limit its CU reconcile needs for multiple updates, like a layout mgr. change).

Bottom up updates (Source to GUI) will be processed mostly on a low priority background thread. A a/sync Display execution will be used only to update the GUI. One or more thread will be available to run the update strategy routine. Processing could be potentially canceled if a new change is submitted while the previous change has not completed.

On a (source) Document about to change event, VE will disable the ability to process a Top-Down path, and queue the document event change. No document content or analysis will be collected at this point. Later on, a low priority background thread (UI processing will always take precedence) will pick up the list of deltas to be processed. A lock will guarantee that the current (source) Document reflects the latest delta on the queue.

If the document is parseable, VE will build a shadow Bean Declaration Model (as it does today) and drive the delta from the shadow BDM to the active BDM (a shadow BDM is a AST wrapper that holds expressions VE is interested with; active BDM holds a decoder for every one of its expressions).

If the document is not parseable, the deltas contents will be applied to the appropriate decoders of the active BDM. A decoder will at least update its content, and document offset. It will also potentially try to make sense of the change (without an AST node), and apply the change to the model.

On the non parseable path, VE will NOT make an attempt to add new element to the model. Delta processing may remove elements if the delta removes the content of a given active BDM expression/decoder.