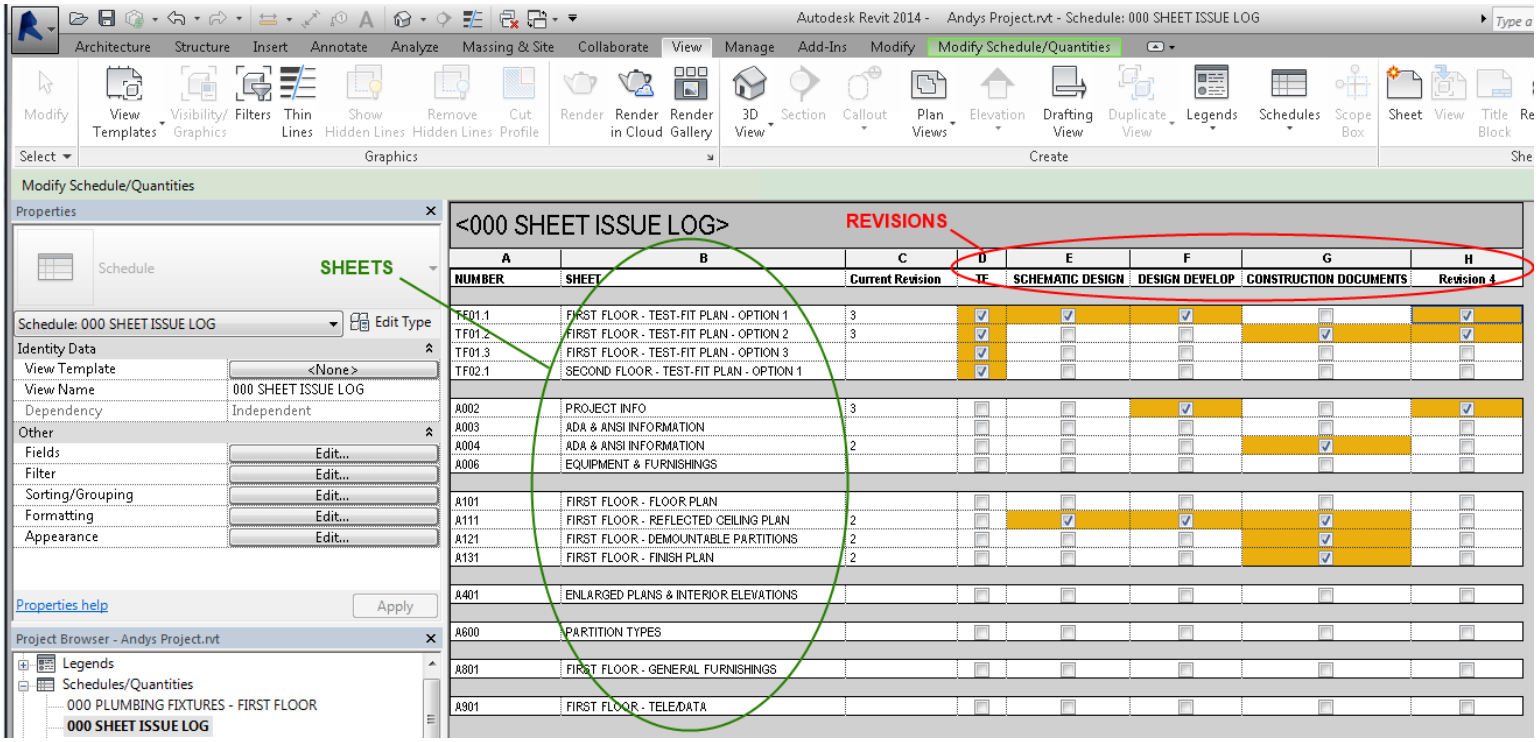
Revit API: Wrangling Revisions with Ruby

Revit API: Wrangling Revisions with Ruby

OPXer Mat Hart came up with an idea for managing Revit project sheets: he wanted to have a master schedule log that would keep track of all of the revisions for every sheet, like this:

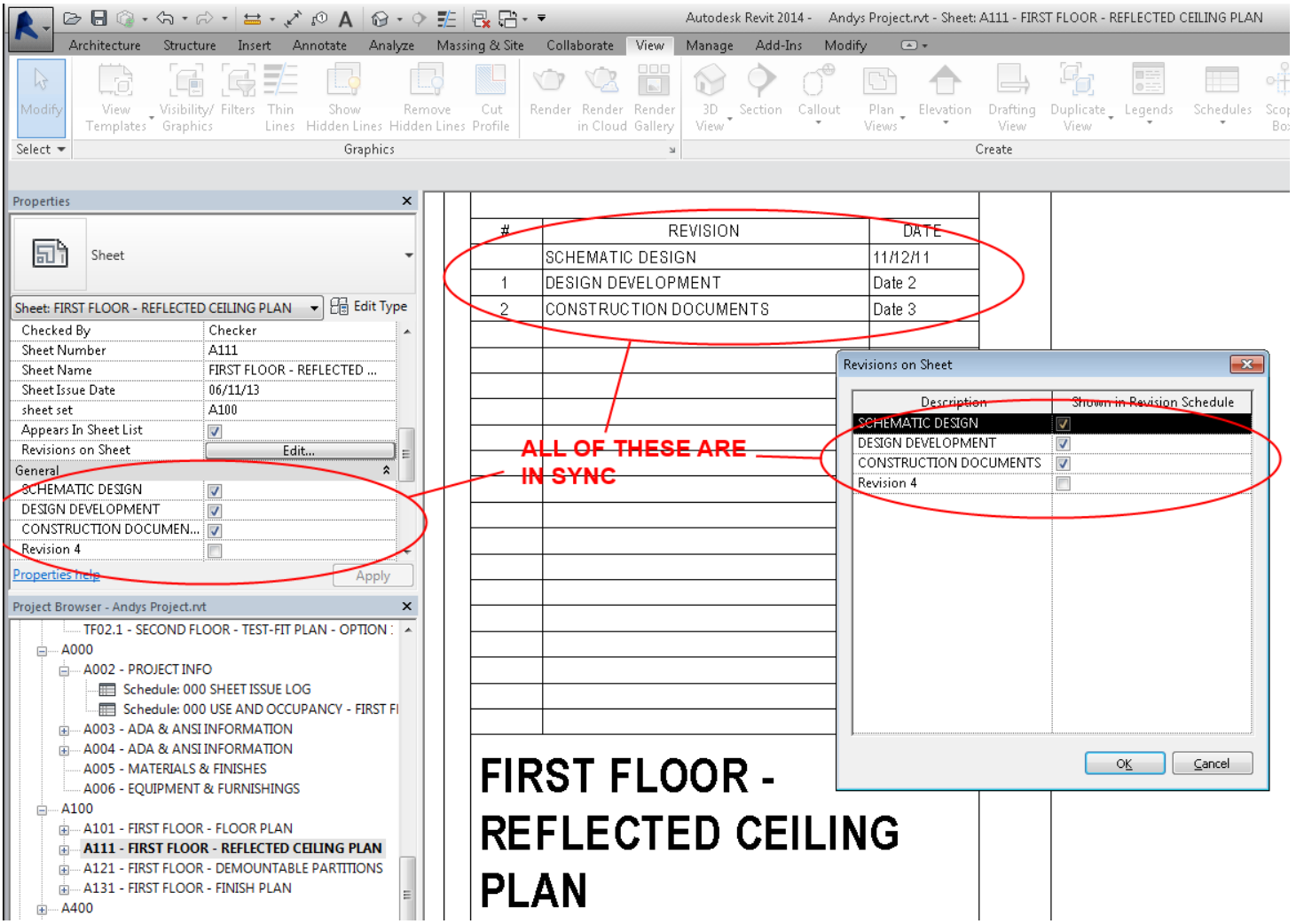


The problem is that, as of Revit 2014, you can't make Revisions be columns in a schedule – it's just not something that Revit can do. In the picture above, the Revisions columns aren't really Revisions, they are actually Fields added to a sheet list Schedule from Parameters that were given the exact same names as Revisions.

What Mat wanted was for the sheet list Schedule (henceforth being referred to as the "Sheet Log") to automatically stay

in sync with the columns that are named after Revisions:

1. If you added a new Revision to the project, the Sheet Log would automatically get a new column added with the name of the Revision
2. Checking or unchecking a Revision column for a sheet would automatically update that sheet's Revisions on Sheet parameter AND Title Block Revision schedule, as shown here:



3. Adding/deleting Revision Cloud to a sheet or checking/unchecking values in a sheet's Revisions on Sheet parameter would automatically update the Sheet Log

So, in other words, the goal for this effort was to allow the Sheet Log would be a single master list of all sheets showing which sheets were part of each Revision. Automating all of this would avoid the human errors that come from having to manually keep track of the relationships between sheet revisions and a master sheet log.

Revit 2014 API: A Must-Have for Revisions and Ruby

It turns out that this project would not have been possible prior to Revit 2014, since only in 2014 did Autodesk add to the Revit API the ability to access and manipulate Revisions on Sheet. Fortunately, they did add Revisions on Sheet to the 2014 API, so we were in luck.

Also, the 2014 API (via the SharpDevelop API development tool provided with Revit) added the ability to develop Revit macros in Ruby (rather than C# or Python), so Ruby was used to develop the functionality. (IronRuby, the Microsoft .NET-specific Ruby version needed for Revit API development, is actually no longer under development by Microsoft, but it works for Revit development).

opxRevisionAndSheetLogUpdater

The resulting project, **opxRevisionAndSheetLogUpdater**, is a set of Ruby files that can be used by copying them into the Revit 2014 macros directory. That very long directory is something like:

```
C:\ProgramData\Autodesk\Revit\Macros\2014\Revit\AppHookup\opxRevisionAndSheetLogUpdater\Source\opxRevisionAndSheetLogUpdater
```

which is where new macros are created when using the Macro Manager.

The Ruby code for this project is available from GitHub at https://github.com/RealHandy/opxRevisionAndSheetLogUpdater. You can download the code and explore it. You can also email questions about the project to aholmes@opxglobal.com.

Some key hurdles that were cleared in creating this project:

- Creating Ruby code that could easily run in Revit

  I used RevitRubyShell and tested all my code in it before I ever turned it into a macro. I develop in Sublime Text and load the files into RevitRubyShell for testing. It's fast and easy.
- Determining the IronRuby syntax for accessing Revit .NET classes and interfaces

  It took some time to find the proper ways to interact with .NET from Ruby. There were a number of these that would be easier if you're familiar with .NET CLRs and DLRs. I did go into the IronRuby source code on GitHub at one or two points. For example, using the IUpdater interface and adding a Ruby method as a Revit DocumentChanged event handler uses this syntax:

```
# IUpdater-based class definition.
class SheetRevisionChangeUpdater
    include Autodesk::Revit::DB::IUpdater
    …
end
# Create and register an updater.
updater = SheetRevisionChangeUpdater.new( app.ActiveAddInId )
UpdaterRegistry.RegisterUpdater( updater )
…
app.DocumentChanged.Add( updater.method(:on_doc_changed_new_revision_handler) )
```

- Working around Revit and Revit API limitations and edge cases

  You often don't know exactly how the Revit API will behave until you test it, particularly when working with DMU Updaters and events like *DocumentCreated* and *DocumentChanged*. You have to try something to see whether it works, and the opxRevisionAndSheetLogUpdater code has some code and comment sections that elaborate on

these. One example: the revision titleblock doesn't update correctly when the API removes the last Revision on Sheet. That just seems like a bug, really. The workaround was to re-add and re-remove the revision in separate DB transactions.

Also, the Revit API simply doesn't allow you to do certain things. You tend to find these limitations only when you reach the point of needing to code certain functionality. You can't set conditional formatting of fields, or create non-shared parameters, or manipulate revision clouds, as examples.

- Turning Ruby code into a Revit macro that loads on startup

You can't create a Revit add-in in Ruby, because there is no IronRuby compiler, and add-ins have to be .NET assemblies (i.e., DLLs). So, rather than translate the entire project into C# to make an add-in, the code remains as a macro. The trick is that there is no macro – the code is a Ruby Revit macro module that contains zero macros, but loads on startup. That way, the user can't forget to run a macro, or disable macros, or other things that would prevent the code from running.

Invaluable assistance

These sources were hugely useful in developing this code:

The Building Coder (http://thebuildingcoder.typepad.com) – Jeremy Tammik's enormously valuable blog that examines the Revit API.

Boost Your BIM (http://boostyourbim.wordpress.com) – Harry Mattison's outstanding Revit API blog. Harry also created the Revit 2014 Macros in Ruby YouTube video.

Revit 2014 Macros in Ruby (http://www.youtube.com/watch?v=3rCu1acxwR0) — The how-to for using the Revit Macro Manager and SharpDevelop to create a Ruby Revit macro. This shows how the ThisApplication.rb is created and how to switch back and forth between SharpDevelop and Revit to test a Ruby macro.

Spiderinnet (http://spiderinnet.typepad.com) – Another source of Revit API knowledge.