4.1 Finish and document your expected requirements for the workflow system from the usability perspective, that you have done in the in-class workshop.

For building an extensive, useful workflow tool, there are several factors that we take into consideration: usability and adaptation, onboarding ease, workflow development capabilities, and overall tool aesthetics. Instead of starting from scratch, we've done an analysis of popular workflow development tools, including Taverna, Kepler, and Vistrails with respect to the qualities listed and will use these findings to build requirements for our tool. A few key findings are as follows:

1. Taverna is fairly easy to use, provides the necessary development capabilities with the ability to define beanshell scripts, and has a clean UI. However, getting started with Taverna is not as straight-forward as users have to install several dependencies for its configuration. Additionally, Taverna lacks customization in workflow visibility as the components are automatically placed in a linear fashion.

2. Although it is fairly easy to setup Kepler, it lacks core user experience qualities and requires the additional understanding of "Director" components making it not as straight-forward to build workflows. It is also not as simply to define customized scripts for components.

3. Vistrails has a customizable, drag and drop UI which is convenient for the user. It also allows for definition of python scripts to extend workflow capabilities. Vistrails is fairly easy to setup and use. In addition, it uses color signals to show the progression of workflow execution.

4. All of the tools require native downloads rather than being SaaS.

Given these observations we are deciding to design a workflow tool which uses Vistrails as its base model and adds upon it by pulling out some of the other positive capabilities from the remaining workflows and other additions.

The usability requirements for our workflow tool are as follows:

1. Customizable workflow views through drag and drop capabilities.
2. The ability for users to visualize the execution of each component in the workflow, in real-time, when running a workflow.
3. The ability to drag and drop workflow components onto the canvas to build the workflow.

4. Simple design for defining custom scripts with the option to choose from a diverse set of programming languages, including python source and Java Beanshell.
5. The ability to onboard with ease through a SaaS model where users are able to access the capabilities of the workflow tool without downloading packagings locally.