PipelineMethodSorter

PipelineMethodSorter intercepts the lists of tests to be run, and reorganizes them, according to several specified properties:

- **xnat.pipeline.useDynamicOrdering:** If false, this class will simply leave the test methods in the order in which they were defined within the class. Otherwise, it will reorder the tests to try and optimize total runtime (minimize makespan), based on the **estimatedRuntime** property for the **@PipelineLaunchParams** annotation on the pipeline launching tests. Using this feature then requires additional properties to be set...
    - **xnat.pipeline.slots:** This property specifies how many jobs may be run concurrently, in order to figure out the best ordering. This is done using a Multiprocessor Scheduling algorithm from https://bitbucket.org/xnatdev/jobshop. If the **xnat.pipeline.glpk** property is set to true, the library will use glpsol from GLPK (which must be installed), to get an ordering with minimal makespan. An example of how to write this problem as a MILP problem can be found here: http://docs.mosek.com/8.1/cxxfusion/case-studies-multi-proc-scheduling.html. If **xnat.pipeline.glpk** is set to false (or not included), the Hochbaum-Shmoys algorithm (http://www.columbia.edu/~cs2035/courses/ieor6400.F07/hs.pdf) is used. The result should be that the pipeline launches and checks are ordered to roughly minimize the makespan of the jobs. In either case, a slight simplification is performed before using the Multiprocessor Scheduling algorithm as described here: reduction.pdf. Essentially, if a job exceeds the average runtime for a processor, we can assign it to it's own processor, and then solve the remaining problem of the rest of the pipelines with 1 less processing slot. It's useful to simplify the problem a bit in this case, because the FreeSurfer pipeline has such an inordinate run-time.