Problem statement:

DMPR Batch Processing

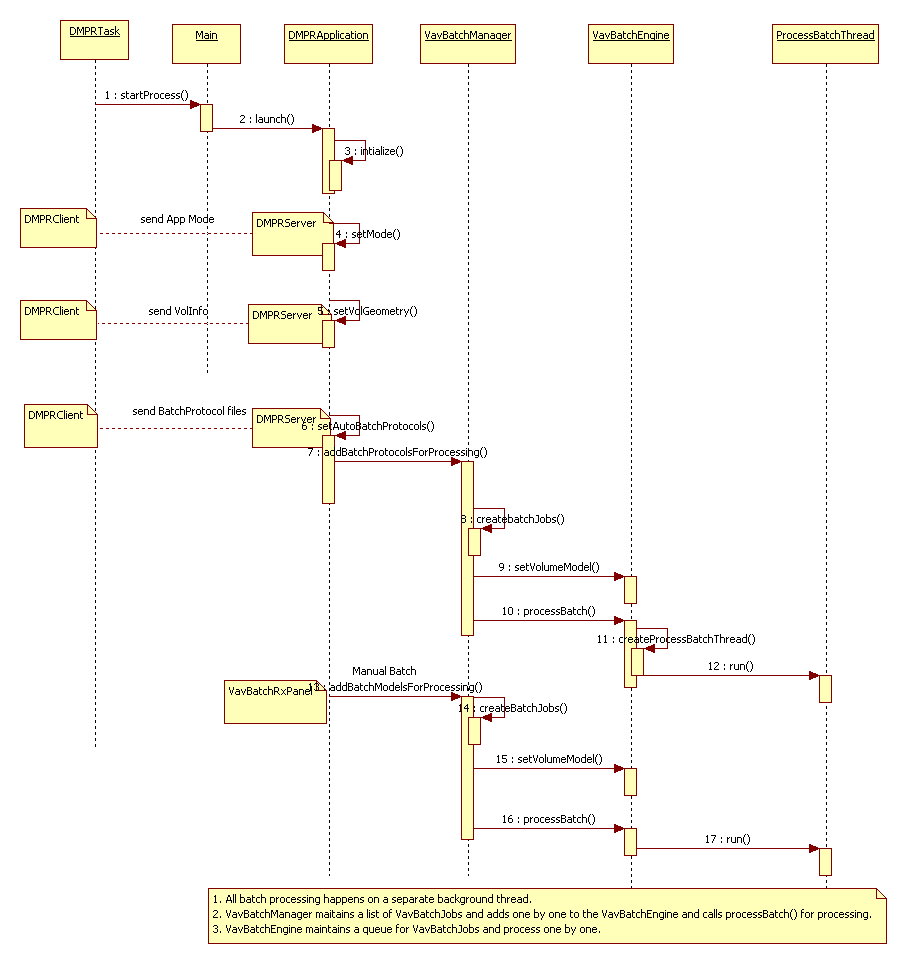
* Two types of batch processing needs to be supported
  + Auto Batch – This is done by selecting a pre-defined multi-planar batch reformat protocol before the scan is acquired. Once the recon data is available, the batch images will be created according to the chosen protocol.
  + Manual Batch – This is done after the recon data is available. A Graphic Object is used on default reformat volume views (Oblique, Axial, Coronal, and Sagittal). A pre-defined multi-planar batch protocol or a generic multi-planar batch protocol can be used to create the batch reformat images.
* Auto batches will process in FIFO order, not stopping at a manual reformat that has been flagged to be done.
* If a manual reformat is launched and started, it will be granted system resources and become the most important or highest priority active reformat. Active auto batches continue to process at a second or lower priority in the order set by the task list.
* As soon as the manual reformat is done, active auto batch / s resume at the same priority in the order represented in the task list

Current Design and its drawbacks:

Single Threaded

* All batch processing happens on a separate dedicated thread
* The batch manager keeps a queue of the prescribed auto/manual batch processing jobs and processes them one by one
* An off-screen T3D render engine is used to generate the batch reformat frames

***VavBatchManager*** is the main class that manages and drives the whole batch processing workflow. All auto/manual batch models prescribed are submitted to the manager for processing. The ***VavBatchManager*** creates batch jobs for each batch model submitted to it and keeps them in a FIFO queue. When the batch manager is triggered to start processing, it processes the jobs one by one using the ***VavBatchEngine***, till all the jobs are completed.



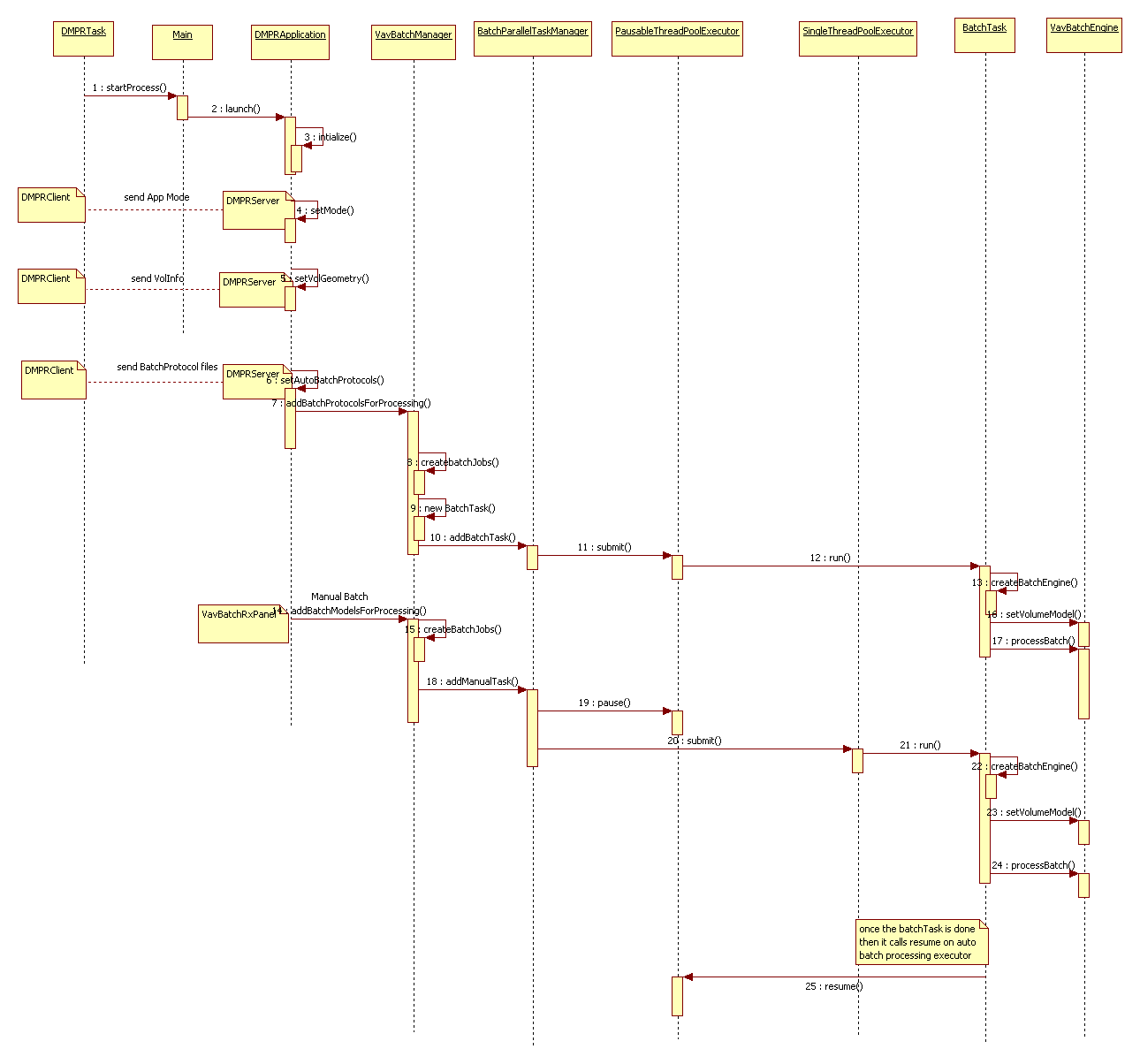
In the current design VavBatchEngine process the batch jobs one by one. So the limitations of the current design are

1. There is no parallel processing of the batchjobs.
2. If user launches the DMPR Apllication and request for the manual batch then processing of this manual batchjob will be in queue until all the auto batch jobs are finished(if they are specified during the scan). In the current design we cannot prioritize the manual batch job to run first by pausing all the auto batchjobs.

To address these requirements we have modified the batch scheduling/processing design as Parallel BatchJob Processing.

Proposed Design solution:

Parallel BatchJob Processing



ParallelBatchTaskManager executes each added task parallely in a separate thread by using TreadPoolExecutor. Number of active threads in TreadPoolExecutor are configurable.

In this new design if user requests the manual batch when auto batch jobs are running then the priority is given to the manual batch job by pausing the auto batch job processing executor. Once the list of manual batch jobs are done then it resumes the auto batch job processing executor.

Class Diagram:

