**TestMessage class (Alia):** just a data holder, with information on source/destination, message type, body, etc.

**TestBlockingQueue class (Jiawen):** I don’t think C++ has a std blocking queue class defined, so we will have to create our own.Implements a queue that is thread safe. Blocking queues should override enqueuing and dequeuing methods to be thread-safe, so only one process/thread is allowed to access a queue at a time. If a process/thread attempts to enqueue or dequeue while another thread is already there, the process/thread should be blocked and wait for the queue to be free. We should also have a maximum queue capacity that can be set, and have a process/thread block and wait if it attempts to enqueue onto a full queue or dequeue from an empty queue. I’d probably use a semaphore to control threads on queues, but there might be a better way. This is probably our most complex class, so we can help with this whenever needed.

**TestMessageHandler class (Santhosh):** Should implement three blocking queues for three different purposes:

1. readyMessageQueue – child processes add to this queue when they are available.
2. testRequestMessageQueue – holds each individual test for child processes to run.
3. testRequestListMessageQueue – each item is a message with a list of DLLs (sent from the client). We don’t have to implement any methods for this queue yet (we can add that when we add a client).

This class should provide methods to add a message to the appropriate queue based on the message’s ‘type’ field, and to remove a message from the appropriate queue based on its ‘type’ field.

**Existing Code Changes (Alia):** TestHarness will start up child processes, which will be used to run TestRunner. TestHarness will enqueue request messages for child processes in TestRunner to dequeue. Processes/threads running TestHarness and TestRunner will communicate through the TestMessageHandler interface.