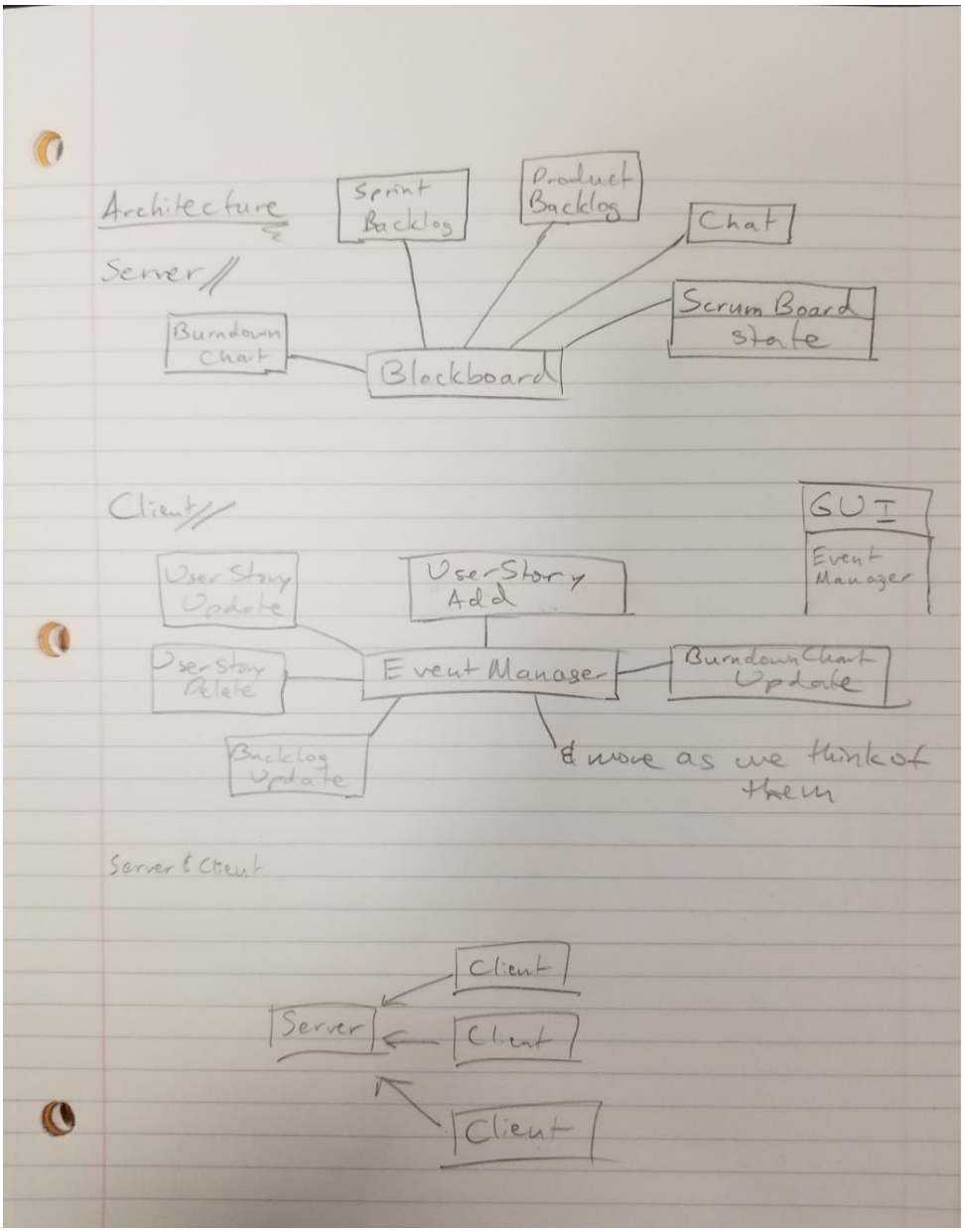
Joey DiMaria
Paul Kwak
Manuel Marroquin
Taylor Murray

1.UI design in JavaFX (you don't need to have everything working yet, but the screen layout should be renderable). This should include at least the SCRUM board and menus showing other features.

2. An architectural design model depicting the selected architecture. This is likely to be either one of the original architectural designs with improvements, or a blend of two or more proposed designs.



3. An executable architectural prototype that minimally shows that data entered or modified on one screen is broadcast (or otherwise distributed) and displayed on the other screens.

Joey DiMaria
Paul Kwak
Manuel Marroquin
Taylor Murray

4. Basic code to support ‘record locking’.

5. A short document explaining how (3) and (4) are implemented.

With the combination of both the Blackboard and the Read-write lock pattern, we will ensure that as multiple clients access the data structures in the Blackboard component, that all operations on the Blackboard are concurrent and displayed consistently across clients. To be more specific, we have modified our original design to store user stories from the HashMap to the thread safe HashTable. The interactive chat structure in the stories class will be maintained with a thread safe list using the Collections.synchronized initialization. When running our current prototype we are able to concurrently access stories across clients. Record Locking will thus reflect users’ changes to the server in terms of the stories uploaded to the scrum backlog, product backlog, and scrum board along with the chart. This will be achieved sing a client-server architecture along with synchronized data structures to ensure that no issues happen when multiple clients update the same data structure containing stories.

6. A simple list of any design patterns you plan to use with a one sentence rationale for using them

Design Patterns:

- Blackboard
    - Our finalized combined design will be utilizing a loosely implemented version of the Blackboard pattern. Since many different clients and the server itself will need to access information on the state of the sprint repeatedly, the Blackboard pattern will maintain the state.
- Read-write lock pattern
    - One again, many different components will be accessing monitors via threads, and we will be utilizing synchronized keywords for methods, along with thread safe data structures to ensure this pattern is correctly implemented.
- Builder pattern
    - We want to utilized the software engineering principle of letting components do one thing well, so instead of letting the server code put together a stories construction itself, we will use the builder pattern to separate the construction of stories from text areas in the UI from the actual implementation.