<REQKEEPER> DESIGN PROPOSAL

Document Author(s): Troy Boone

Date: 6/12/20

Design Rationale

The main objects that will be important to the system implementation will be the GUI object, the Requirement object itself, the interface for handling each individual state, and the object that deals with the I/O of the software. The data that will be required to implement the system will mainly be the requirement specific information (id, state, summary, test, priority, estimate, developer, and rejection). Out of all of these specific variables, the summary and id are needed to identify what the requirement is to begin with. The priority, state, estimate, and test are needed to assign developers efficiently to debug and test software. The rejection and state variables will help as the requirement moves between different states and why it was moved to begin with.

The requirement object is what the state machine requirement state interface interacts with. The RequirementList class handles how the requirement is handled with user inputs, which relies on the RequirementIO to handle import and export, and RequirementGUI which is the format the user sees the data in.

The limitations and constraints of the system cannot be fully stated at the moment, but mainly rely on user limitation. A user will be able to bog down the system with requirements that have been rejected, but still shown in the UI. This is to keep a log of all rejections, but there will be a point where most of the UI shown is just rejected requirements instead of completed ones.

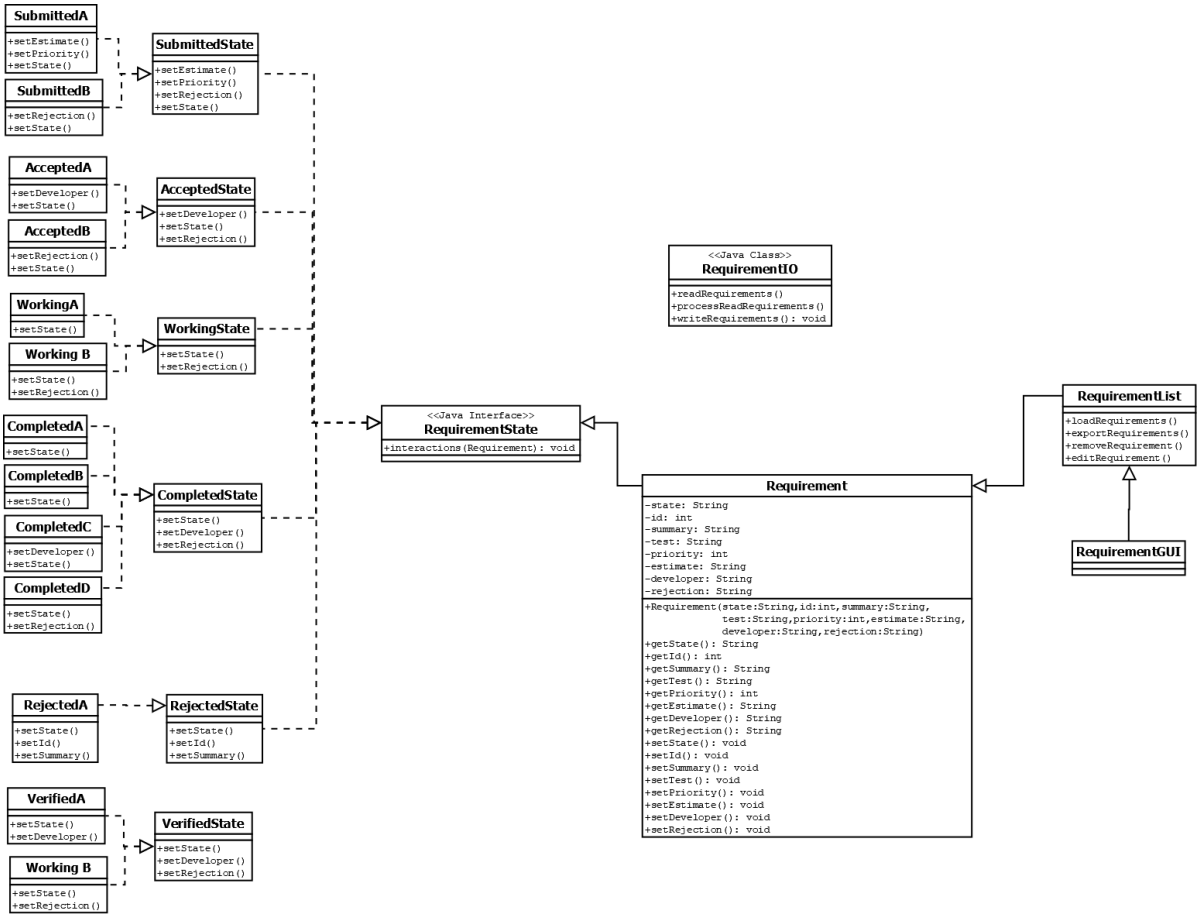

Figure 1: Class Diagram for <ReqKeeper>

Document Revision History

Date	Author	Change Description
6/12/20	Troy Boone	• UML, Design Proposal, and Black Board Testing documentation completed
		•