- Design strengths and weaknesses
  - Strengths
    - Modularity
    - High Cohesion
    - Event Driven Design
    - Observable *and* Model View Control
    - Multi-threaded implementation
  - Weaknesses
    - Using the Observer Model results in the passing of unwanted information to some subscribers
    - Subclasses of ControlDesk directly access BowlerFile
- Positive Aspects of Design
  - Easily maintained
  - Good usability
  - Intuitive
- Major Tasks of Each Component
  - PinGUI
    - Displays current state of Pinsetter and the current Roll-State
  - LaneGUI
    - Displays the names and current score of the bowlers
  - ControlDesk
    - Adds Bowlers to Bowler database
    - Adds Bowlers to Parties
    - Assigns a Party to a Lane
    - Can observe scores on lanes
  - Alley
    - Contains and Constructs the ControlDesk object
  - Lane
    - Keeps track of what’s happening on a lane
    - Handles scoring
    - “Powerhouse”
  - Party
    - The container for Bowlers to be assigned a lane
  - Bowler
    - Data holding class that contains vital info on a bowler
  - LaneEvent
    - Contains all pertinent data from a lane
  - Queue
    - Container object that acts as a queue for waiting Parties
  - BowlerFile
    - Class that interfaces with Bowler database
  - ControlDeskEvent
    - Contains all pertinent data from the ControlDesk
  - Pinsetter
    - Controls all info regarding pins
- How would we improve the design?
  - Decrease Coupling between objects by better implementing Observer pattern in some Views

- TBD