The actor-based model worked very well when applied to the design of the TSA project. The actor-based design made modeling the interactions between different objects very straightforward and helped to keep things operating concurrently and smoothly. The use of actors and immutable messages removed potential deadlocks, and also made the design easy to comprehend. The interactions between actors were able to be limited to effectively transmitting passengers and bags and performing actions involving those passengers. By limiting the complicated interactions between different steps of the TSA check-in process, synchronization in the actor model was easy to implement.

We made a few changes between our original design and the final product. The first change we made was to get rid of the Queue actor/class. The queue class was unnecessary due to the nature of Akka messages, which would store in a FIFO data structure as they accumulated. Rather than maintain a queue of passengers waiting to be processed in the system, we could just send extra passengers to each scanner and know that they would be processed in order as time continued. By taking out the Queue, the only change was that the Doc Scanner was now in charge of sending the passenger and bag to the appropriate scanner in the next line. Other changes we made involved reworking how the actors closed at the end of the day. The Controller now sends a close message to the Doc Scanner which then sends a close message to the Bag Scanners and Body Scanners. When both Scanners in a specific line send messages that they are closed to a Security Station that Security Station closes. When a Security Station is closed a message is sent to the Jail; when the Jail receives a message from all security stations the Jail closes. Once the Jail closes a message is sent to the controller so that the controller can close down the entire system. We needed to ensure that a certain order was followed in closing the actors, to keep the system from closing before every line was fully processed.

Overall, the actor-based design translated very nicely to our implementation of the project. Very few changes needed to be made in the design, and the ones that were made were done to keep things simple and optimized, rather than out of necessity from a complication in the program. The use of Scala for the project also helped keep things operating smoothly in the actor system.