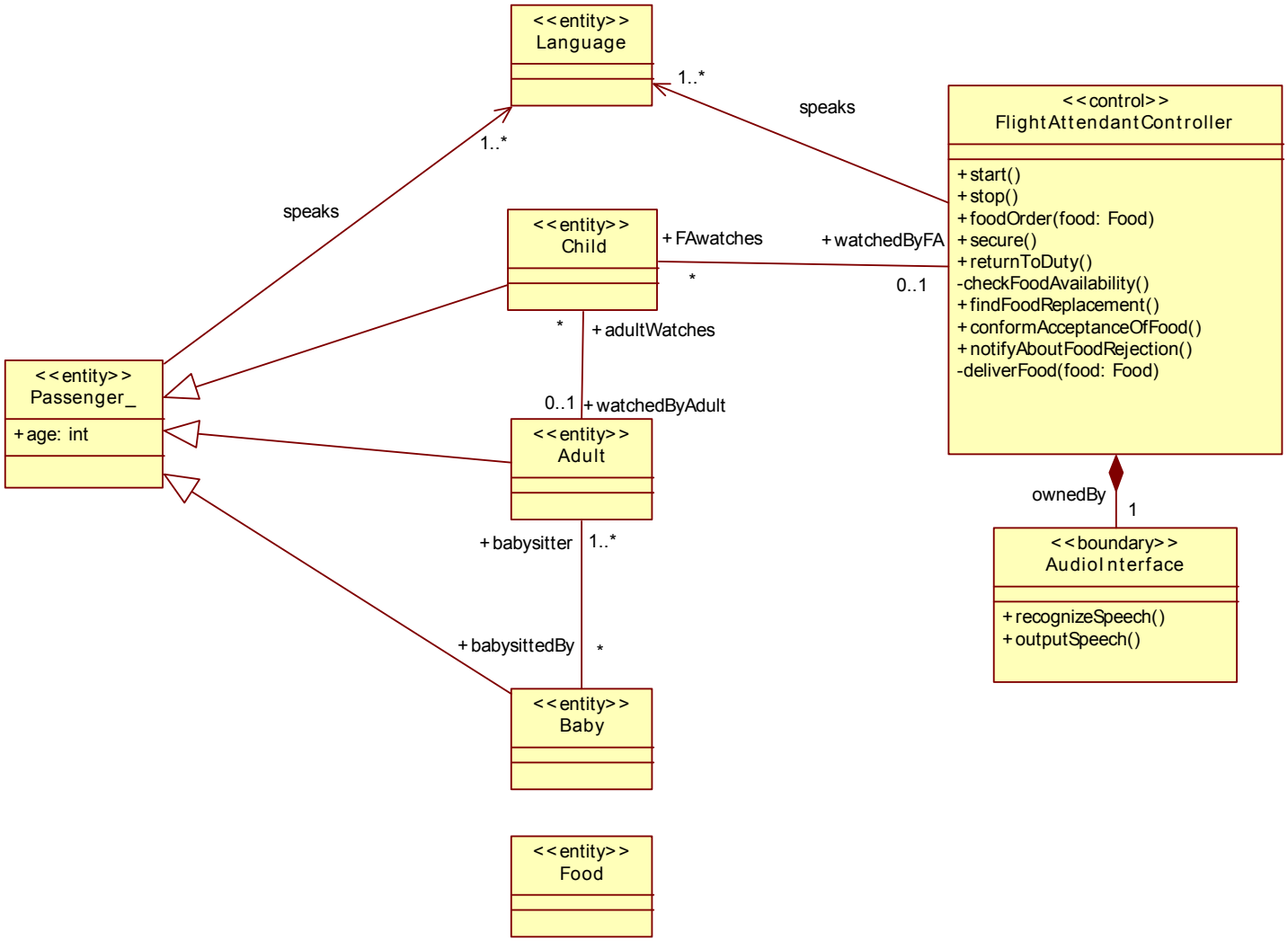
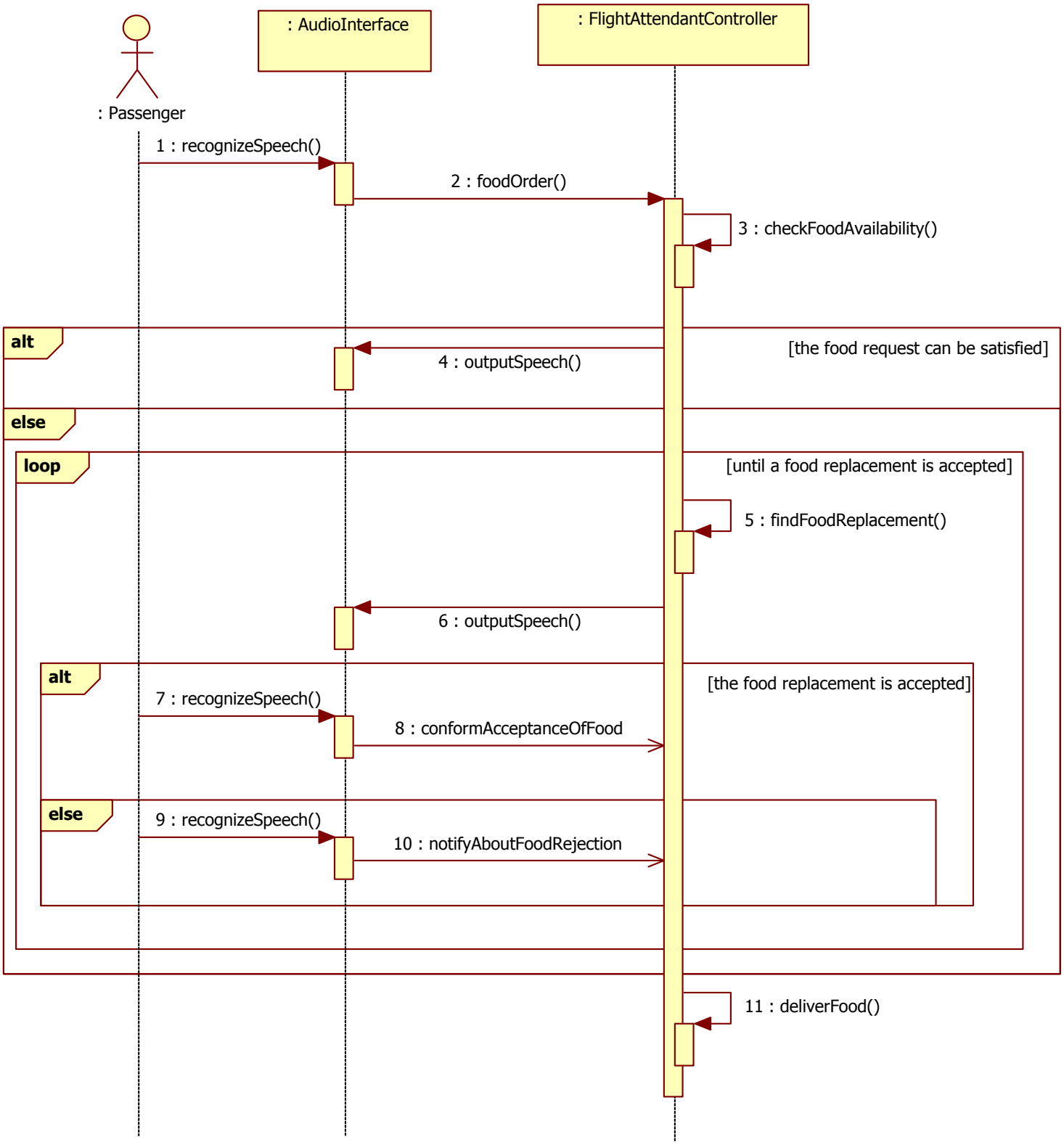
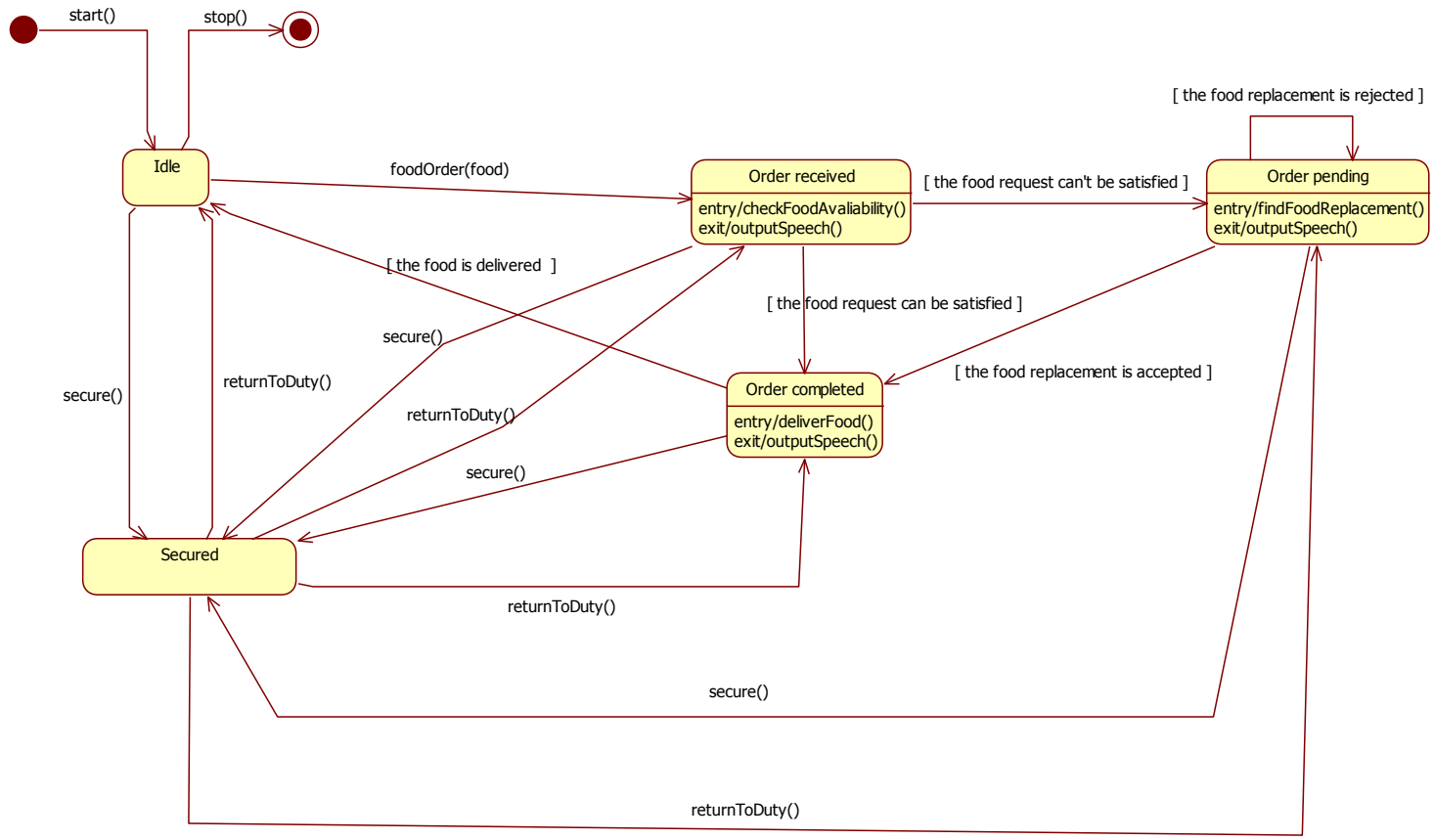

<<entity>>
Language
1..*
speaks
<<control>>
FlightAttendantController
+start()
+stop()
+foodOrder(food: Food)
+secure()
+returnToDuty()
-checkFoodAvailability()
+findFoodReplacement()
+conformAcceptanceOfFood()
+notifyAboutFoodRejection()
-deliverFood(food: Food)
1..*
speaks
<<entity>>
Child
+FAwatches
+watchedByFA
*
0..1
*
+adultWatches
0..1
+watchedByAdult
<<entity>>
Passenger_
+age: int
<<entity>>
Adult
ownedBy
1
<<boundary>>
AudioInterface
+recognizeSpeech()
+outputSpeech()
+babysitter
1..*
+babysittedBy
*
<<entity>>
Baby
<<entity>>
Food


: Passenger
: AudioInterface
: FlightAttendantController
1 : recognizeSpeech()
2 : foodOrder()
3 : checkFoodAvailability()
alt
[the food request can be satisfied]
4 : outputSpeech()
else
loop
[until a food replacement is accepted]
5 : findFoodReplacement()
6 : outputSpeech()
alt
[the food replacement is accepted]
7 : recognizeSpeech()
8 : conformAcceptanceOfFood
else
9 : recognizeSpeech()
10 : notifyAboutFoodRejection
11 : deliverFood()


start()
stop()
Idle
foodOrder(food)
Order received
entry/checkFoodAvaliability()
exit/outputSpeech()
[the food request can't be satisfied]
Order pending
entry/findFoodReplacement()
exit/outputSpeech()
[the food replacement is rejected]
[the food is delivered]
[the food request can be satisfied]
secure()
returnToDuty()
secure()
returnToDuty()
secure()
[the food replacement is accepted]
Order completed
entry/deliverFood()
exit/outputSpeech()
Secured
returnToDuty()
secure()
returnToDuty()