Midterm 2014

Jörg Kienzle

Self-Check In Booth (1)

Many airlines have switched to self-service check-in kiosks, especially for domestic flights. If you've never used a kiosk before, we'll walk you through the process.

Walk up to an open kiosk, bringing your luggage with you. The kiosk will prompt you to identify yourself, either by inserting a credit card (don't worry, it won't be charged), typing in your flight confirmation code (you won't see a keyboard; enter your identifying information using the "keyboard" shown on the touch screen), swiping your frequent flyer card or scanning your passport.

You should then see a screen which shows your name and air travel itinerary. You will be asked to confirm your flight information by touching an "OK" or "enter" button on the screen. Then you will be able to review and change your seat assignment, if desired.

Self-Check In Booth (2)

Next, you'll be asked about checked baggage. You will have to enter the number of bags you want to check, but some touch screens use an up- or down- arrow system or a "+" and "-" key instead. In that case, you will touch the up arrow or plus sign to increase the total number of bags on the screen. You will need to press "OK" or "enter" to confirm the number of bags you're checking.

At this point, the kiosk will offer you the choice to either print your boarding pass (or passes, if you have a connecting flight), or send it to you using an SMS message (in which case you must provide a phone number). Finally, your baggage tags, if you are checking any luggage, are printed. This completes your check-in.

In case the check-in kiosk remains idle for too long, the check-in is cancelled and you must start again. You can also voluntarily abort the check-in procedure as long as you did not print your boarding pass.

Task to Complete: Please provide a use case for the "Self Check-In" procedure. Please consider all relevant secondary actors and how they interact with the system.

Self-Check In UC (1)

1 **Use Case**: CheckIn
1 **Scope**: SelfCheckInBooth
1 **Level**: User-Goal
1 **Intention in Context**: The User wants to check-in for his flight to get his boarding pass and baggage tags.
2 **Multiplicity**: Only one User can check-in at a booth at a given time.
2 **Primary Actor**: User 2 2 2 2 2
Secondary Actors: CardReader, PassportScanner, TouchScreen, Printer, SMSInterface
Main Success Scenario:
User walks up to the booth with his luggage.
4 *1.* User identifies with *System* by swiping his credit card or frequent flyer card through the *CardReader*.
1 *2. System* presents flight information and itinerary to User by means of the *TouchScreen.*
1 *3.* User confirms flight information to *System* using *TouchScreen*.
1 *4. System* asks User if he wishes to check any baggage using *TouchScreen.*
2 *5.* User communicates number of bags to check to *System* using *TouchScreen*.
1 *6. System* asks User if he wishes to print his boarding passes or receive them as an SMS message using *TouchScreen*.
1 7. User informs *System* that he wishes to print the boarding passes using *TouchScreen*.
1 8. *System* instructs *Printer* to print boarding passes.
2 9. *System* instructs *Printer* to print baggage tags, if any.

Parking Garage Use Case (2)

Extensions:

2 1a. User identifies with *System* by scanning his passport using the *PassportScanner*. Use case continues at step 2.

2 1b. User identifies with *System* by providing his flight confirmation code using the *TouchScreen*. Use case continues at step 2.

1 4a. User notifies *System* that he wishes to change seat assignment using *TouchScreen*.

2 4a.1 *System* presents available seats to User using *TouchScreen*.

2 4a.2 User informs *System* of chosen seat using *TouchScreen*. Use case continues at step 4.

2 7a. User provides *System* with phone number using *TouchScreen* in order to receive his boarding passes via SMS.

2 7a.1 *System* instructs *SMSInterface* to send boarding passes to User's phone. Use case continues at step 9.

1 (3-7)b: Timeout: User was idle for too long.

1 (3-7)b.1 *System* informs User of timeout using *TouchScreen*. Use case ends in failure (or continues at step 1). 1

1 (3-7)c: User informs *System* that he wishes to cancel self-check in using *TouchScreen*. Use case ends in failure (or continues at step 1). 1

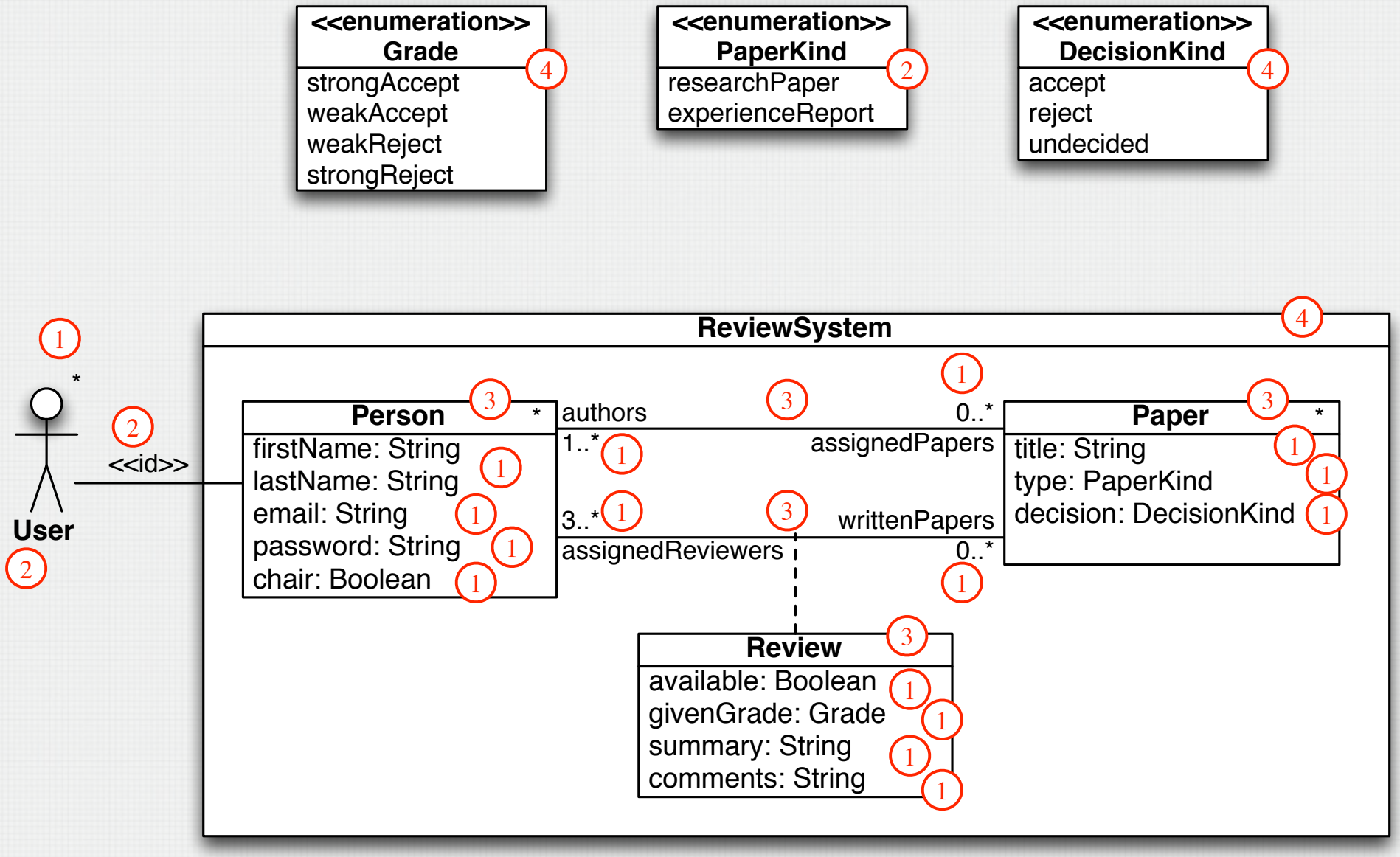
Review System Concept Model

The program chair of a conference has a very difficult task. He has to coordinate the review process of the research papers submitted by authors to decide on which papers should be part of the final program for the conference, and which ones should be rejected. Each paper is reviewed by at least 3 members of the program committee, and each reviewer submits a review comprising of a summary of the paper, a grade for the paper ("strong reject", "weak reject", "weak accept", "strong accept"), and detailed comments that justify the grade. Unfortunately, though, some reviewers do not complete their reviews in time.

To help the program chair, we are planning to develop a computer-assisted review system. In the system, authors first have to create an account, which stores their first name, their last name, their email address and a password. Papers are known by the title, the set of authors, and whether they are research papers or experience reports. For reviewers, the system also stores the first name, last name, email address and password. When a reviewer logs in, she can consult the list of papers that have been assigned to her for review, and submit reviews for each of those papers. The submitted reviews are initially not visible to the authors, but only to the program committee. Only once the program chair has reached a decision on each paper ("accept" or "reject"), the authors of the paper are allowed to consult the reviews.

Devise a concept model that contains the actors, conceptual classes, attributes and relationships relevant in the context of a (single) review system.

Review System Concept Model



Recycling Machine (1)

Based on the Recycle Items use case, establish an environment model for the recycling machine. For each message, provide:

• The parameters of the message, if any. The allowed parameter types are all the standard OCL types. If you use other types, please provide their definition using OCL or UML notation.

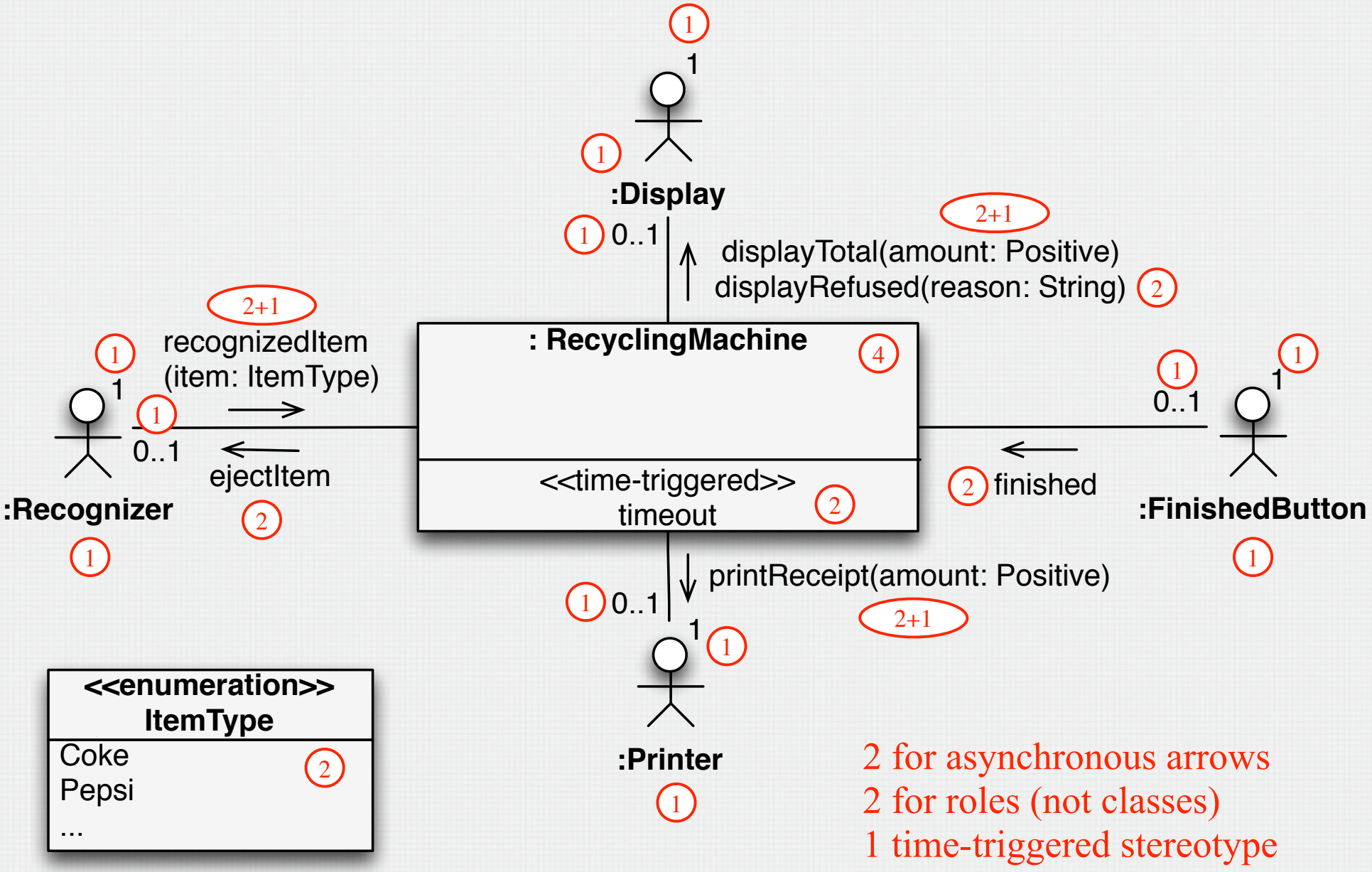
• If the name of the message does not describe its functionality in an unambiguous way, provide a small textual description of what the message does.

Don't forget to add the multiplicities to the actors and the communications in the diagram.

Recycling Machine (2)

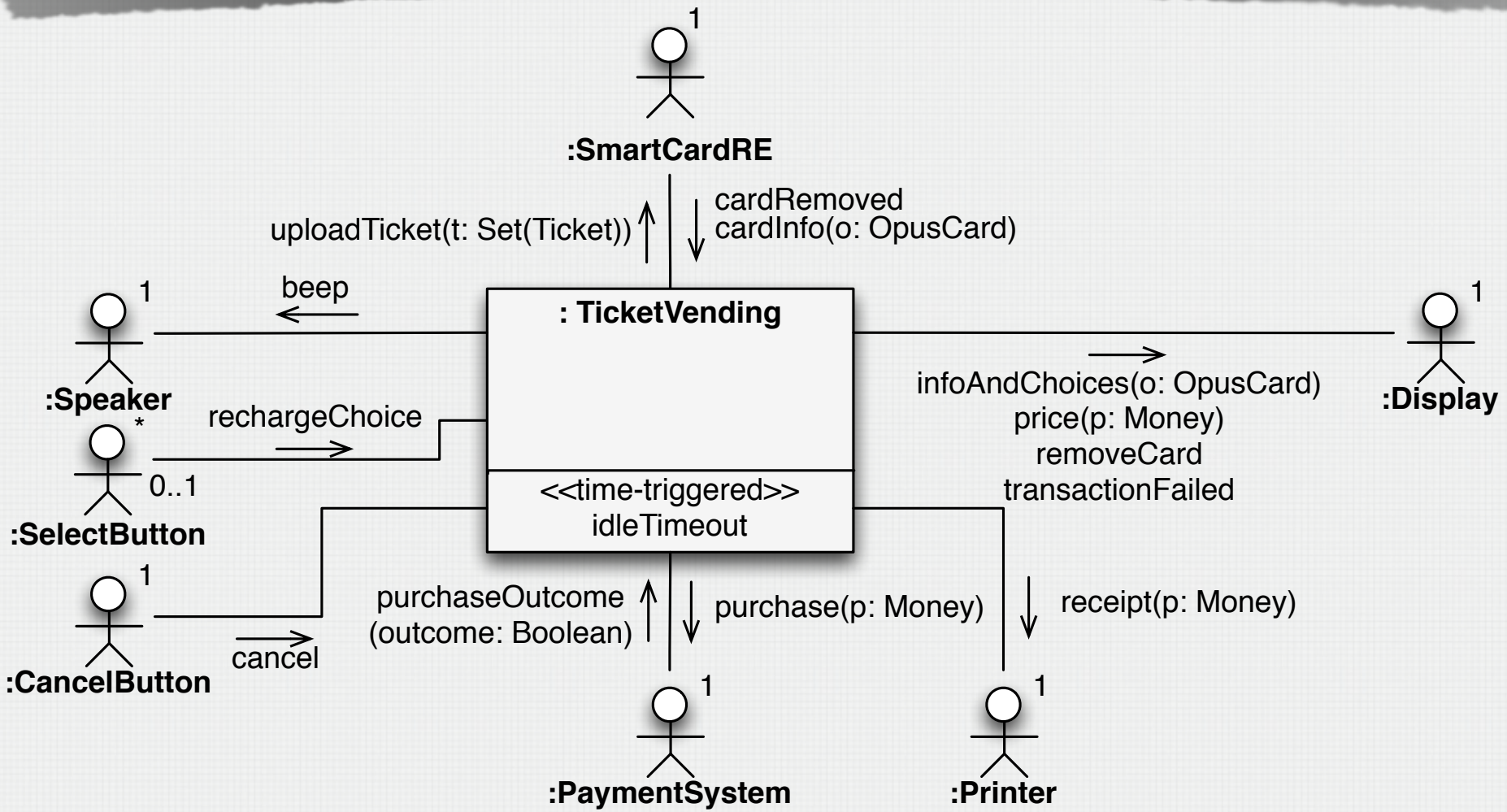
Use Case: Recycle Items
Scope: RecyclingMachine
Level: User-Goal
Intention in Context: The User wants to recycle bottles and cans in exchange for money.
Multiplicity: Only one User can recycle items at a given time.
Primary Actor: User
Secondary Actors: Recognizer, Display, FinishedButton, Printer
Main Success Scenario:
Step 1, 2 and 3 are repeated for each item the User wishes to recycle.
1. User inserts a can or a bottle into the Recognizer (interaction happening in the environment).
2. *Recognizer* informs *System* about the item that was recognized.
3. *System* acknowledges recognition to User by updating the refund total on the *Display*.
4. User informs *System* that s/he has no more items to process using the *FinishedButton*.
5. *System* prints receipt using *Printer*.
User takes receipt to a cashier in order to exchange it for cash.
Extensions:
2a. Timeout occurs because user has not inserted any item into the recognizer for more than 2 minutes. Use case continues at step 5.
3a. System determines that the inserted item is not accepted at this store.
3a.1. *System* informs user that item is not accepted at this store using the *Display*.
3a.2. *System* instructs the *Recognizer* to reject the inserted item. Use case continues with next item at step 1.

RecyclingMachine Environment Model

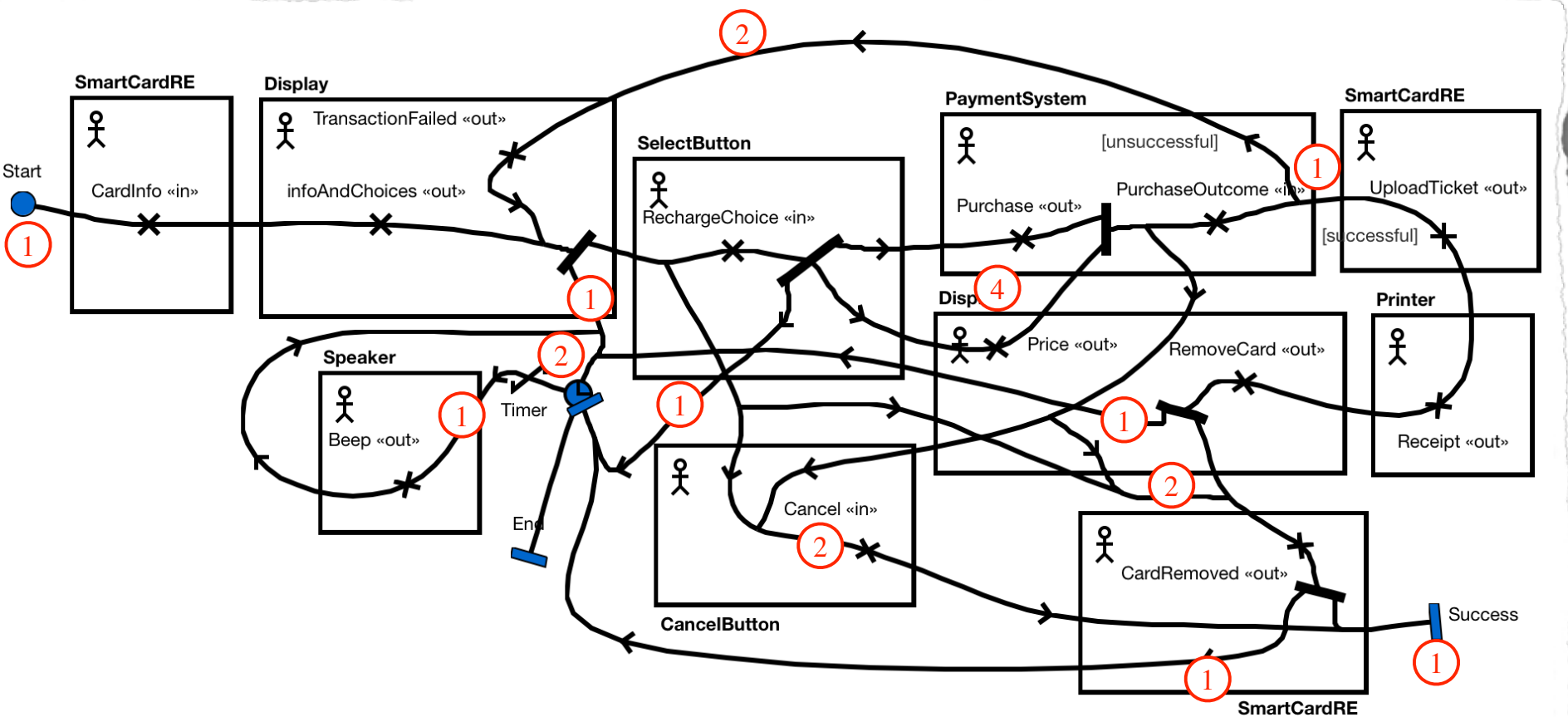


Ticket Vending Machine

Establish a Protocol Model for the Ticket Vending Machine that specifies the possible sequencing of input and output messages precisely.



Ticket Vending Protocol Model



max 10 for operations (1 for each, with correct <<in>> or <<out>>)