HW4

Traccia

Una macchina distributrice di cibo e bevande accetta pagamento in monete, eventualmente fornendo il resto, oppure tramite una chiavetta prepagata. Si tracci un diagramma di stato UML che descriva il comportamento atteso della macchina. Si tracci poi un ulteriore diagramma di stato UML che realizzi il comportamento atteso richiamando le opportune operazioni definite nella classe *VendingMachine*. All'interno della seconda macchina i comportamenti possono essere definiti come azioni associate alle transizioni o come azioni enter/do/exit associate agli stati.

Risoluzione

Per poter rappresentare al meglio uno state diagram è bene porsi nei panni della realtà che dovrà essere modellata, data che non potranno essere illustrate attività ma solo condizioni a cui sottostare, si ricordi che la descrizione degli stati avviene tramite l'utilizzo di partecipi al loro interno. In relazione al primo *chart* potrebbero essere adeguati differenti flussi di esecuzione, i quali a loro volta tendono ad aumentare la complessità modellativa; si potrebbe porre come primo stato la *selezione del prodotto*, l'*immissione di monete* oppure la combinazione delle due. Si osserva come l'ultima scelta introdotta rappresenti un livello di difficoltà maggiore delle prime due, in cui occorre stabilire una transizione che renda accomunabili i due processi esecutivi di selezione e acquisto del prodotto.

Nonostante di seguito è introdotto un diagramma che pone un certo flusso di esecuzione, ovviando alla possibilità di dinamicità alla realtà di una macchina distributrice.

Primo UML state machine

Considerazioni

Il diagramma pone un livello di complessità più elevato del necessario, tuttavia alcuni elementi per la realtà modellata sono introdotti correttamente. Innanzitutto è giusto stabilire una transizione ciclica che richieda l'inserimento di altre monete fino a quando non si raggiunga il credito necessario, ciò si potrebbe adeguare anche per la fase di *selezione*, in cui il token è mantenuto nello stesso stato affinchè non sia posto uno *string matching* adeguato rispetto ai codici univoci dei prodotti contenuti.

Nonostante, gli elementi negativi sono di numero maggiore e più importanti. L'aumento della complessità è data dalla presenza dello *pseudostato fork*; un fork allude alla presenza di due nodi iniziali di esecuzione che corrispondano a processi esecutivi paralleli, per cui in questo caso, è sbagliato considerarli come due flussi differenti. Per cui è bene eliminare lo pseudostato introducendo due percorsi contraddistinti, i quali successivamente si uniranno ad un nodo decisionale che comprenda l'insieme di archi uscenti per concludere nello *stato finale*. Inoltre, è errato porre al di sopra del fork eventi e azioni, essi devono essere totalmente liberi da vincoli.

Concludendo, un ultima nota considera la necessità di un evento posto tra lo stato che pre-

cede il fork e la transizione che termina nello pseudostato; un evento è valutato più di una volta, differentemente dalla guardia messa a disposizione, per cui se non rispettate, i *marcatori* sarebbero costretti a restare all'interno del participio *selecting product*. Da quanto detto, si evince una regola fondamentale all'interno dei *state machine diagram*, una *guardia* non è mai rivalutata in assenza di transizioni cicliche.

Secondo UML state machine

Considerazioni