

**Webshop modellezése Uppaal programban**

**Biczó Tamás**

**Mérnök Informatikus MSc levelező szak**

**Győr, 2018**

Tartalom

[1 Bevezetés 1](#_Toc503123019)

[2 Templatek 1](#_Toc503123020)

[2.1 Globális változók 1](#_Toc503123021)

[2.2 WebshopInitializer 2](#_Toc503123022)

[2.3 ShoppingSession 2](#_Toc503123023)

[2.4 StockManager 3](#_Toc503123024)

[2.5 PackagingService 4](#_Toc503123025)

[2.6 DeliveryService 4](#_Toc503123026)

[3 Query 5](#_Toc503123027)

# Bevezetés

A formális módszerek tárgyhoz készített beadandó feladatomban egy webshop működését modelleztem Uppaal programban.

Az elkészült modell egy egyszerűsített vásárlási folyamatot mutat be, ahol két termék létezik az internetes áruházban és a felhasználók ezeket tudják megvásárolni. Egy vásárlási session a szükséges mennyiség megadásával kezdődik az adott termékekből, majd addig tart ameddig kiszállításra nem kerül az összes igényelt termék. Közben előfordulhat készlethiányt, ezt a problémát is kezeli az elkészített automata. A következőkben bemutatásra kerülnek az elkészített templatek majd pár query is amelyekkel ellenőrizni tudjuk a helyes működést.

# Templatek

## Globális változók

A rendszerben globális változókat tudunk létrehozni, melyek a rendszer összes templatejéből elérhetőek. Ezen felül itt kerülnek felsorolásra a rendszerben található broadcast és normál események.

Channels:

* **checked**: Ez az esemény, az egyes termék készlet kezelő rendszerek bocsátják ki, amint frissítették a készletet egy vásárlás után.
* **creteOrder**: Ezt a shoppingSession a checked esemény elkapása után lövődik el
* **readyForDelivery**: Ez a szállítás megkezdését jelzi
* **delivered**: Ez a termék(ek) kézhezvételét jelzi

Broadcast chanels:

* **initWebshop**: Ez az első esemény, ami megtörténik, ez állítja be a sessionöket és a stock managereket a kezdeti állapotukba
* **checkout**: Ez jelzi azt, hogy a vásárló akar vásárolni x mennyiségű terméket

**itemsInShoppingCart**[3][2]: tárolja a bevásárló kosár tartalmát. Ez egy kétdimenziós tömb, melynél az első dimenzió sessionönként értelmezett a második pedig termékenként.

**itemsInOrder**[3][2]: tárolja az aktuálisan megrendelt termékeket. Ez egy kétdimenziós tömb, melynél az első dimenzió sessionönként értelmezett a második pedig termékenként.

**isStockManagerFree:** ez a globális változó, a stock menedzserek aktuális állapotát jelzi, hogy ha készlet frissítés van épp folyamatban, akkor ez a flag false különben true

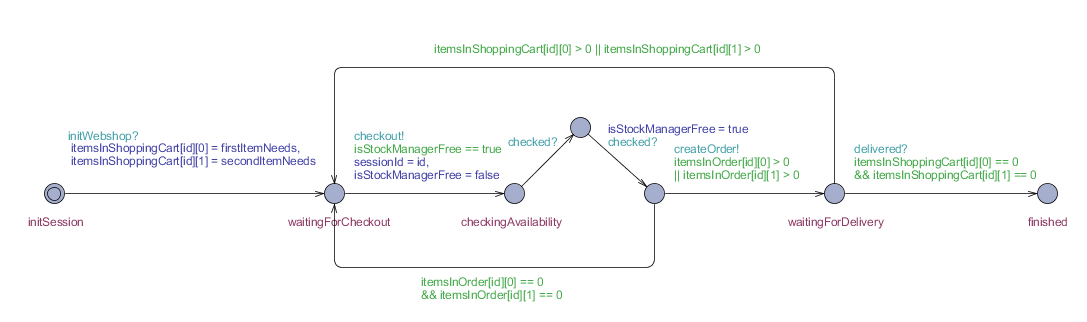
**sessionId:** ez a segédváltozó az éppen aktuális rendeléshez tartozó sessionId-t tárolja, melyet a stock kezelők felhasználnak a készlet megfelelő frissítéséhez.

## WebshopInitializer



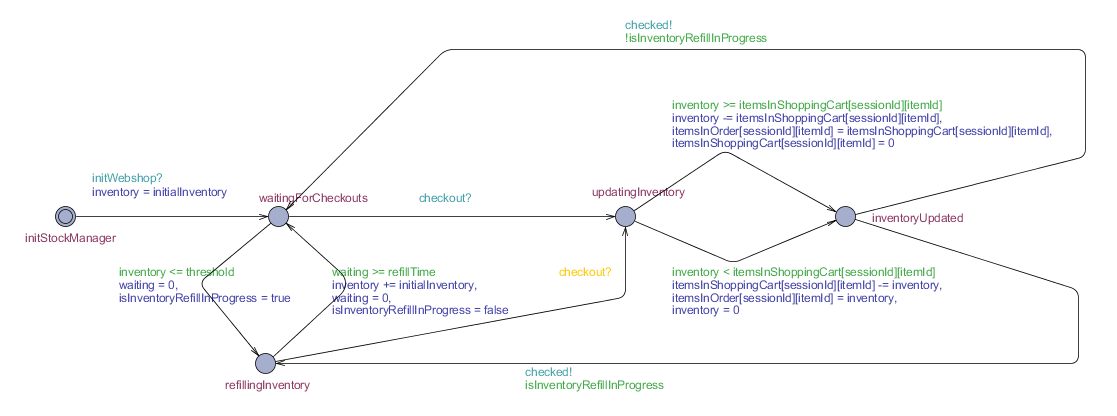
Ez az egyszerű automata a rendszer kezdeti állapotba való állásáért felelős. Az initWebshop esemény hatására az összes feliratkozott automata a kezdeti állapotába lép.

## ShoppingSession



Ennek a template-nek három bemenő paramétere van egy id mely azonosítja, egy firstItemNeeds illetve egy secondItemNeeds melyek a szükségletet jelzik az első, illetve második termékből. Az első állapot átmenetnél kezdeti állapotba lép, feltölti a saját bevásárló kosarát a megvásárolandó termékekkel majd várakozik addig ameddig a készlet kezelő rendszerek elérhetőek lesznek és generál egy checkout eventet. Ezzel párhuzamosan lefoglalja a stock managereket és beállítja a saját id-ját a sessionId változóba. Ezután várakozik, míg mindkét termék kezelőtől megkapja a checked jelzést. Aztán megnézi, hogy ténylegesen mennyi termék került bele a rendelés visszaigazolásba a termék kezelők ellenőrzése alapján, ha minden termékre nulla ez az érték akkor visszatér kezdeti állapotába, ha viszont van valami az orderben, akkor megrendeli azokat. Itt egy újabb elágazás következik. Megnézi a saját bevásárló kosarát és hogy ha maradt még benn elem, akkor visszatér a kezdeti állapotába, különben vár a delivered eseményre és befejezi a vásárlási folyamatot.

## StockManager

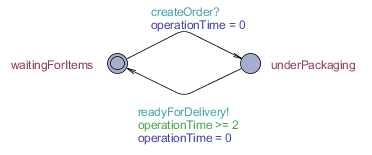


A következő a stock manager, mely az egyes termékek raktárkészletének az ellenőrzésével és a frissítésével, illetve utántöltésével foglalkozik. Négy bemenő paramétere van ennek a template-nek. Az első egy itemId mely egy azonosító, a következő az initialInventory, mely a kezdeti raktárkészletet jelöli. A következő két paraméter az utánrendeléssel kapcsolatos, a threshold egy határértéket jelöl, ami azt jelenti, hogy ha a raktárkészlet ez alá esik, akkor utánrendelés szükséges az adott termékből. Az utolsó paraméter pedig egy időhatározó, mely azt mutatja, hogy mennyi ideig tart a raktárkészlet feltöltése. Továbbá 3 lokális változója is van egy waiting, amely clock típusú és a raktárkészlet felöltés idejének a mérésére használjuk. Egy inventory, mely az aktuális raktárkészletet tárolja, illetve egy isInventoryRefillInProgress flag, mely akkor true, hogy ha éppen folyamatban van a feltöltés.

Az első állapotátmenet során kezdeti állapotba lép az automata, az inventory változóba betöltődik a kezdeti inventory érték. Ezután az automata várakozik egy checkout eventre, mely hatására az inventory aktuális állapota szerint frissíti az aktuális sessionhöz tartozó shopping cart-t, ordert illetve a készletet. Hogy ha a készlet nagyobb, mint a megvásárolandó mennyiség, akkor teljesen kiüríti a bevásárló kosarat és annak tartalmát átteszi az order-be, ezzel párhuzamosan pedig levonja a készletből ezt a mennyiséget. Ha viszont több elemet akar megvásárolni, akkor kiüríti teljesen az inventoryt és csak a készleten lévő mennyiséggel csökkenti a bevásárló kosarat és állít be az orderbe. Ezután, ha nincs éppen készletfeltöltés, akkor egy checked eseménnyel visszatér a kezdeti állapotába. Feltöltés esetén pedig a refillingInventory állapotba.

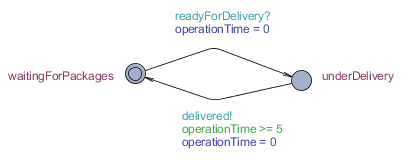
További funkció, hogy ha az automata a kezdeti állapotában van, illetve a készlet a határérték alá esik, akkor utánrendelés szükséges, ekkor elindít egy órát és a paraméterül kapott értékig várakozik a feltöltéssel. Előfordulhat viszont, hogy a feltöltés ideje alatt érkezik checkout esemény, ezt is lekell kezelni. Ekkor ugyanúgy a készlet frissítő állapotba lép mintha a kezdeti állapotban lett volna, annyi különbséggel, hogy az isInventoryRefillInProgress flaget true-ra állítja, ennek hatására a frissítés után visszatér a készlet felöltőt állapotba. Ha letelik a feltöltés ideje, akkor megnöveli a készlet értékét a paraméterül kapott kezdeti raktárkészlet értékkel.

## PackagingService



Ez az automata a rendelt termékek csomagolásáért felelős. Két állapota van egy várakozó és egy csomagoló állapot. E két állapot között egy createOrder esemény hatására megy végbe állapotváltás. Ez az automata is egy órával van ellátva, mely méri a csomagolás idejét, aminek egy minimum érték van meghatározva.

## DeliveryService



Ez az előbbihez nagyon hasonló automata, a működési elve azonos a csomagolóéval, annyi különbséggel, hogy a readyForDelivery hatására kezdi el a kiszállítást. Majd, ha végez akkor egy delivered eseménnyel jelzi a csomag kiszállítását.

# Query

Létrehoztam pár query-t melyek segítségével a rendszerünk helyes működését tudjuk ellenőrizni. Ezek kerülnek bemutatásra ebben a szakaszban.

E<> (itemsInShoppingCart[0][0] == 0 && itemsInShoppingCart[0][1] == 0 && itemsInShoppingCart[1][0] == 0 && itemsInShoppingCart[1][1] == 0 && itemsInShoppingCart[2][0] == 0 && itemsInShoppingCart[2][1] == 0)

Ez azt ellenőrzi, hogy van-e olyan állapota a rendszernek, amikor minden bevásárló kosár kiürült, azaz mindenki megtudta venni a számára szükséges termékeket.

A[] (!(StockManagerOne.inventory < 0) && !(StockManagerTwo.inventory < 0))

Ez azt mutatja, hogy a rendszer egyik állapotában sem megy a készlet nulla érték alá.

E<> (processClock <= 50 && SessionOne.finished && SessionTwo.finished && SessionThree.finished)

Ez azt ellenőrzi, hogy a minden vásárlás végbement, azaz eljutott a finished állapotva és mindezt 50 időegység alatt kitudta szolgálni a webshop.

E<>!(StockManagerOne.waitingForCheckouts && StockManagerOne.isInventoryRefillInProgress)

Ez azt ellenőrzi, hogy a rendszerben nincs olyan állapot, amikor készlet feltölés van folyamatban, mégis a waitingForCheckouts állapotban van az automata.

E<> !(itemsInOrder[0][0] > SessionOne.firstItemNeeds && itemsInOrder[0][1] > SessionOne.secondItemNeeds)

Ez pedig, azt ellenőrzi, hogy az orderbe nem került-e be több termék, mint amennyi az adott vásárlási session igénye.

