KobrA. előadás

A KobrA
programfejlesztési modell
II. rész
Komponens specifikáció

Kozma László tanár úr előadásai alapján

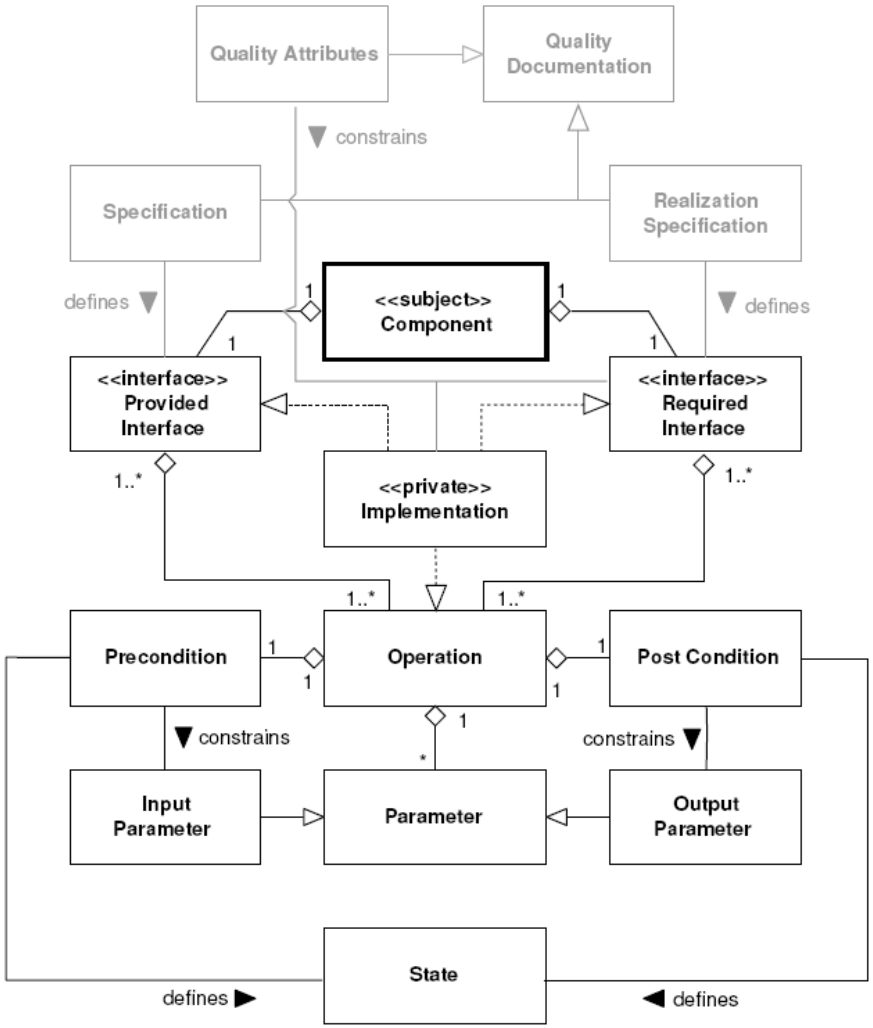
- eddig volt:
  - mi a komponens
  - mi a komponens modell
  - mik a komponens modell részei
  - hogyan írhatunk le komponensalapú rendszereket UML segítségével
  - módszer komponensalapú rendszerek tervezéséhez:
    - háromdimenziós modell, környezeti térkép
- most jön:
  - módszer komponensalapú rendszerek tervezéséhez
    - komponensek, integráció

Komponens specifikáció

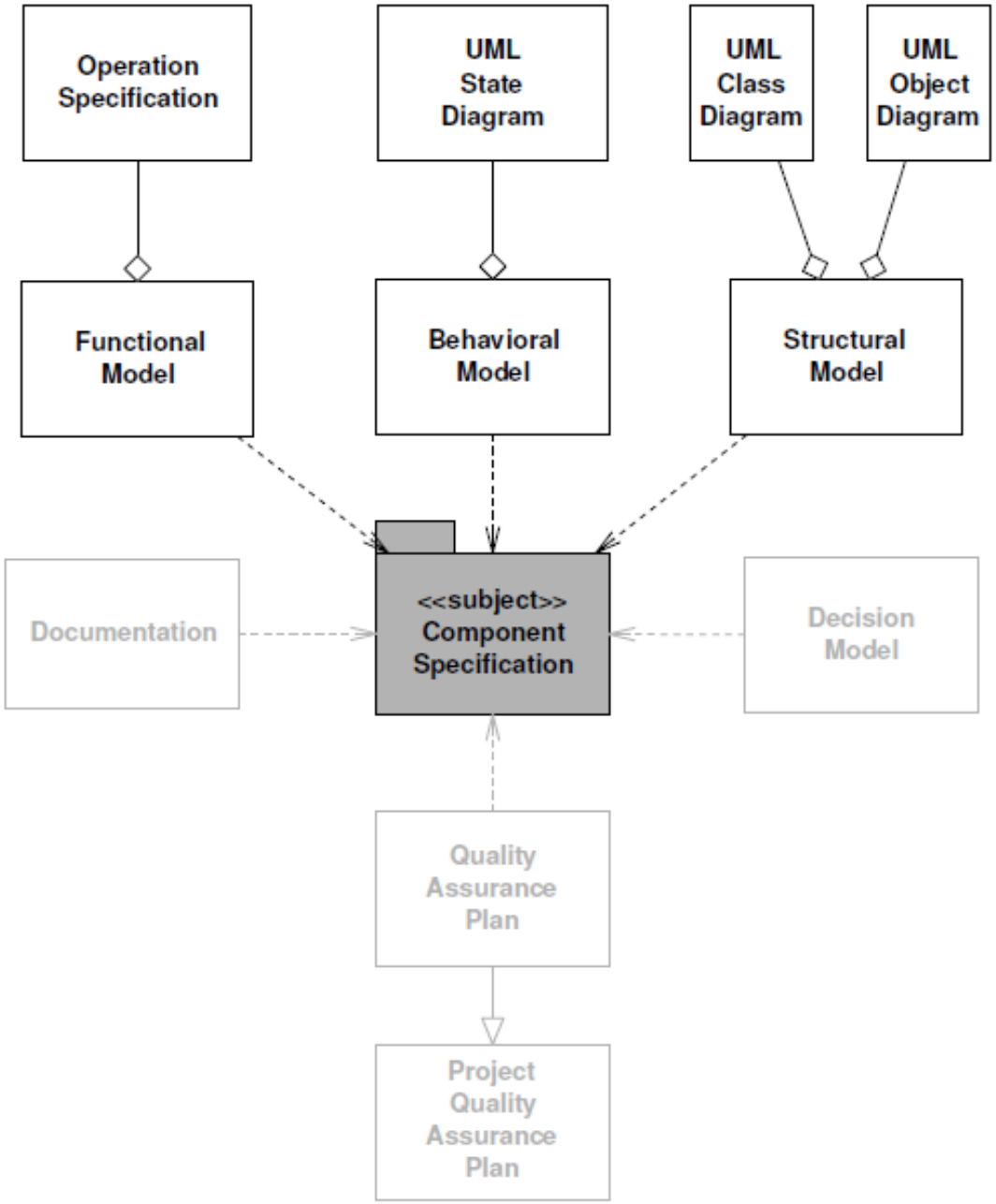
- A komponens specifikáció azon leíró dokumentumok összessége, amely definiálja a komponens működését
- Minden ilyen dokumentum a komponens működésének valamilyen aspektusát vizsgálja és csak arra koncentrál, hogy az adott aspektusból a komponens mit csinál.

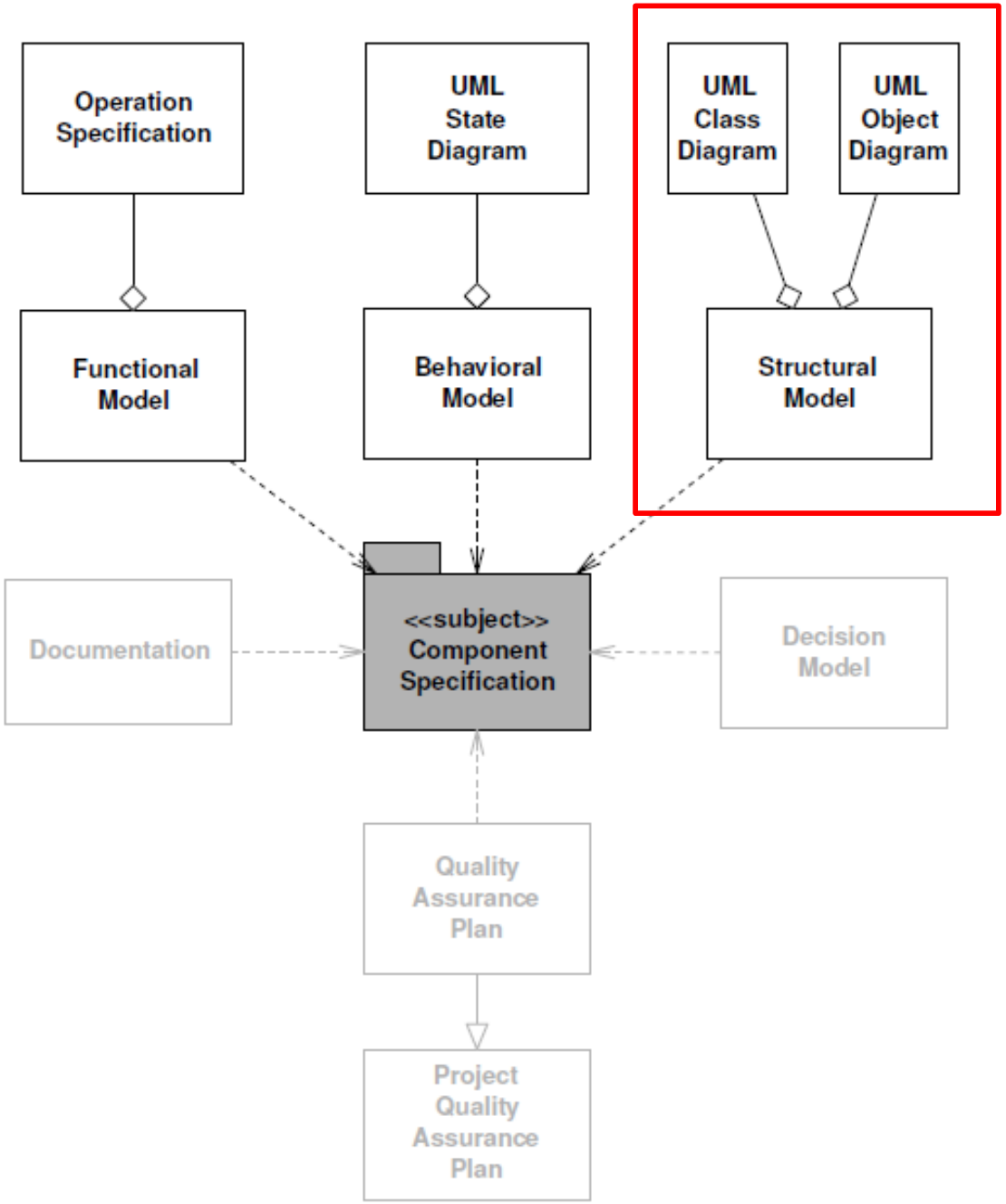
- A komponens specifikáció eszközei:
  - Természetes nyelven informális leírás
  - Grafikusan (UML)
  - Formális eszközökkel
    - például Object Constraint Language (OCL), amelyet az UML szabványban definiáltak, vagy
    - **QoS Modelling Language** a **Q**uality **o**f **S**ervice kritériumok specifikálására
- Mindegy, hogyan történik, a lényeg, hogy a lehető legteljesebben adjuk meg a specifikációt, hogy megértsük a komponens viselkedését és használni tudjuk.
- A cél: az elvárt és szolgáltatott interfészek leírása.

Komponens meta-modell diagram



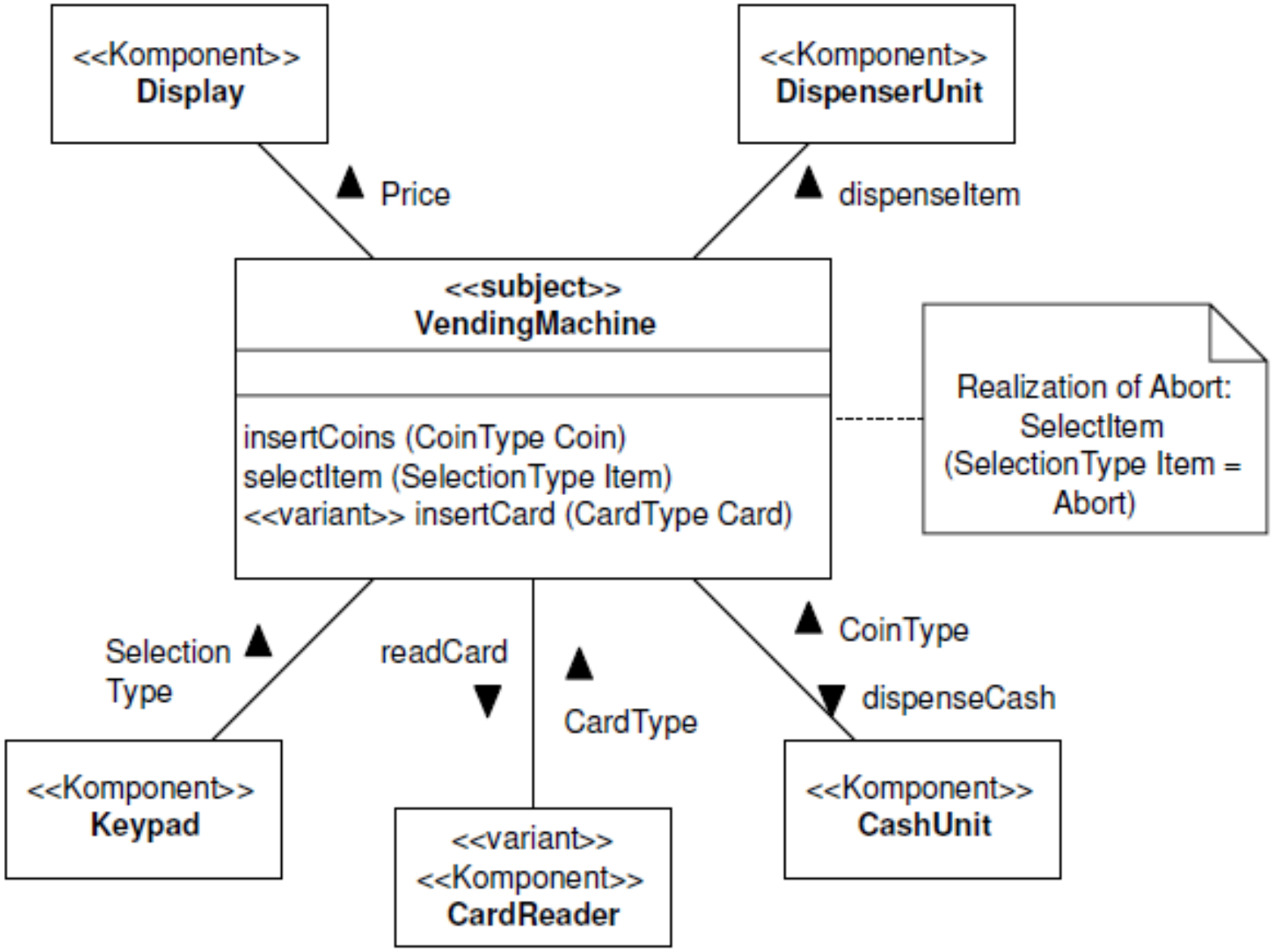
- A komponens specifikáció mindent tartalmaz, amit tudni lehet és kell a komponensről
  - A **strukturális modell** komponens felépítéséről és a vele kapcsolatban levő komponensekről ad felvilágosítást
  - A **funkcionális modell** a komponens funkcionalitását írja le. Itt kell megnevezni a szolgáltatott és az elvárt műveleteket.
  - A **viselkedési modell** a komponens viselkedését definiálja. Itt adjuk meg az elő- és utófeltételeket.
- A 2.10. ábra a komponens specifikáció egy keretrendszerét mutatja be. A keretrendszer mint látható további elemeket is tartalmaz
  - Minőségbiztosítási tervet
  - A komponens teljes dokumentációját
  - Döntési modellt, amely a komponensbe beépített változatosságot írja le. Ezt a fajta változatosságot a konfigurációs interfész támogatja, de ezzel a továbbiakban részletesen nem foglalkozunk.


Operation Specification
UML State Diagram
UML Class Diagram
UML Object Diagram
Functional Model
Behavioral Model
Structural Model
<<subject>> Component Specification
Documentation
Decision Model
Quality Assurance Plan
Project Quality Assurance Plan


Operation Specification
UML State Diagram
UML Class Diagram
UML Object Diagram
Functional Model
Behavioral Model
Structural Model
<<subject>> Component Specification
Documentation
Decision Model
Quality Assurance Plan
Project Quality Assurance Plan

A komponens strukturális specifikációja

- A strukturális specifikáció definiálja a komponens műveleteit, attribútumait és azt, hogy milyen komponensekkel van kapcsolatban (kliensek és szerverek), valamint e kapcsolatokra vonatkozó megszorításokat is.
- A szoftver projektek a strukturális specifikációt általában nem használták, a modellvezérelt fejlesztés megjelenésével vált fontossá.
- Hasznos eszköz, azt mutatja meg, hogy a komponens milyen kölcsönhatásban van a környezetével.
- A környezeti térkép definiálásánál megismert strukturális modellhez képest lényeges különbség az, hogy a komponens strukturális modelljének fókuszában a rendszer áll és nem a környezet.
- A 2.11. ábra az árusító automata komponens strukturális modelljét mutatja be.
- Egy jó strukturális specifikáció rávilágít a jó dekompozíciós lehetőségekre.


<<Komponent>>
Display
<<Komponent>>
DispenserUnit
Price
dispenseItem
<<subject>>
VendingMachine
insertCoins (CoinType Coin)
selectItem (SelectionType Item)
<<variant>> insertCard (CardType Card)
Realization of Abort:
SelectItem
(SelectionType Item =
Abort)
Selection
Type
readCard
CardType
CoinType
dispenseCash
<<Komponent>>
Keypad
<<variant>>
<<Komponent>>
CardReader
<<Komponent>>
CashUnit

- Nézzük meg, hogy az árusító automatánk környezeti térképének strukturális modellje hogyan nézett ki.
- Vessük össze a kétféle strukturális modellt.
- Figyeljük meg, hogy az abort műveletet a környezeti strukturális modellben önállóan definiáltuk, míg a komponens strukturális modelljében a *SelectItem* művelet részeként.

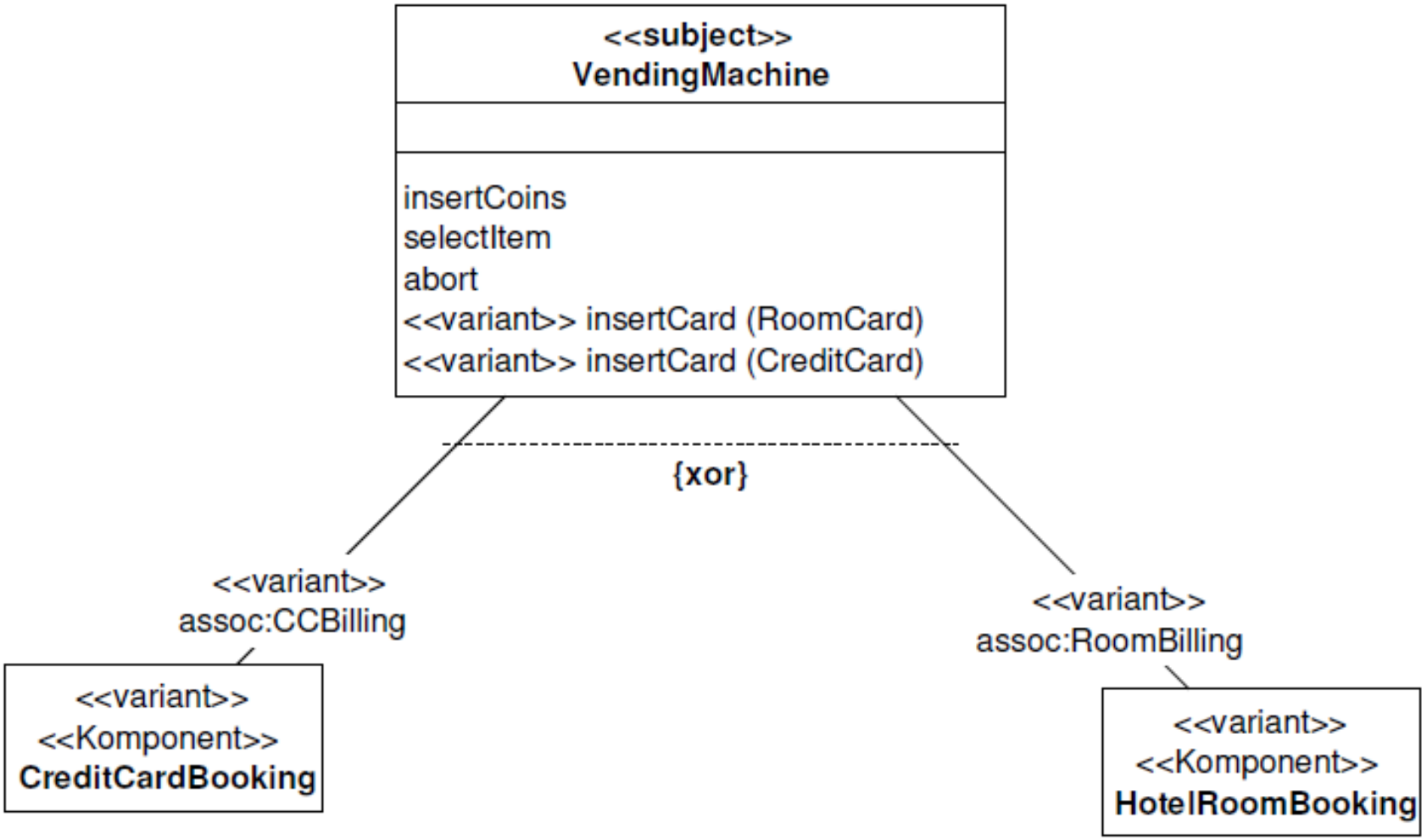
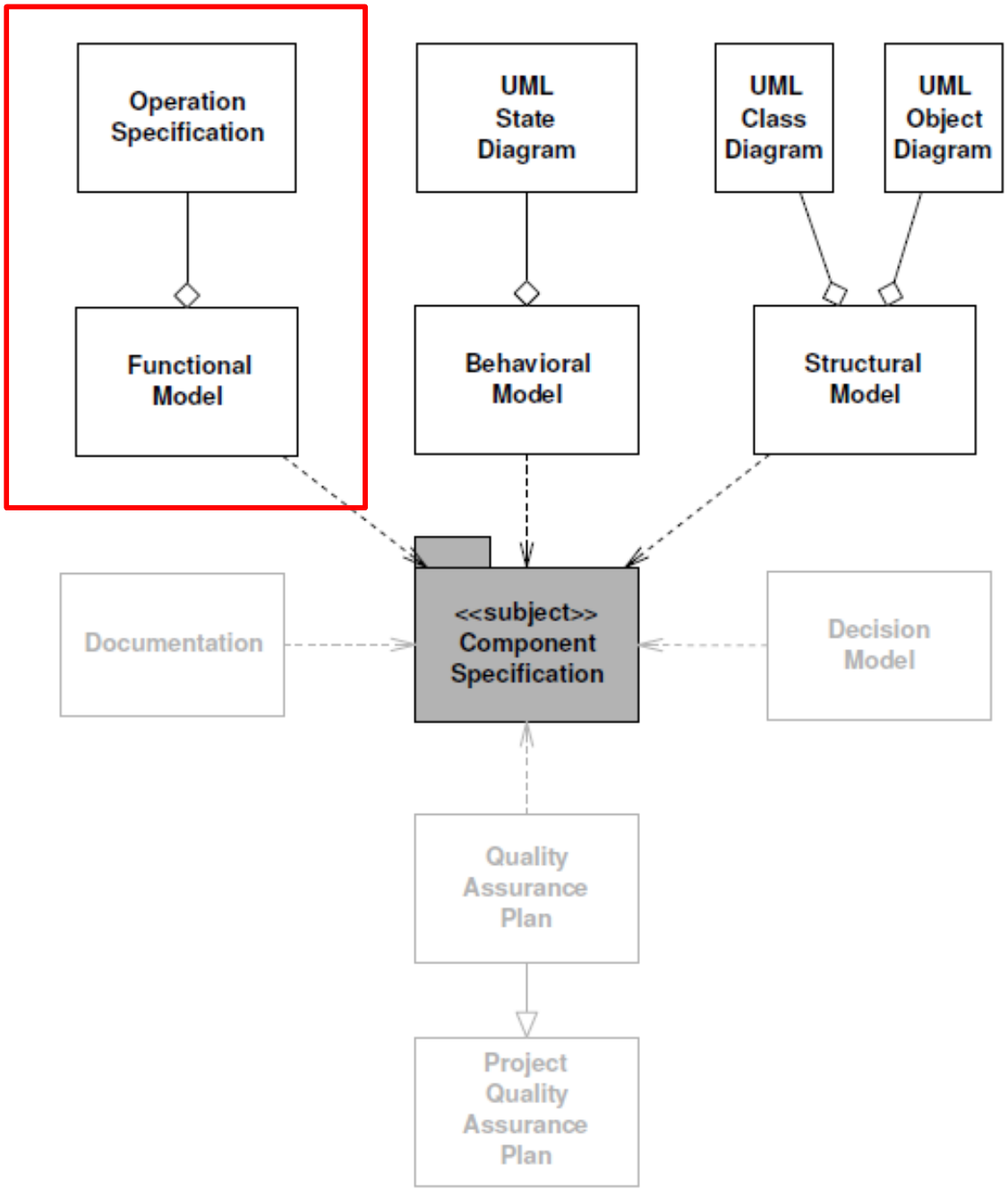


Fig. 2.7. Context realization structural model for the vending machine context

A kétféle strukturális modell összevetése a példánkon keresztül

- Az árusító automata műveleteinek definiálásában van az alapvető különbség a kétféle modell között. A komponens strukturális modelljében a műveletek megvalósításának részletesebb leírását adjuk meg.
  - A környezeti térkép strukturális modelljében szereplő *abort* műveletet a komponens strukturális modelljében specifikált módon a *SelectItem* művelettel realizáljuk mint a művelet input paramétere. Ezt a tényt a komponens strukturális modelljében kommentárként jelezzük.
  - A két strukturális modell konzisztenciájának megőrzése érdekében visszatérhetünk a környezeti térkép strukturális modelljéhez is és bejegyezhetjük a változást, vagy egy kommentárt helyezhetünk el a megfelelő helyre.
- Az újabb tervezési döntések átvezetésére bármikor visszatérhetünk egy korábbi modellhez.!!!


Operation Specification
UML State Diagram
UML Class Diagram
UML Object Diagram
Functional Model
Behavioral Model
Structural Model
Documentation
<<subject>> Component Specification
Decision Model
Quality Assurance Plan
Project Quality Assurance Plan

A komponens funkcionális specifikációja

- A Funkcionális specifikáció szerepe az, hogy definiálja a komponens által szolgáltatott műveletek kívülről látható hatásait.
- A 2.3. táblázat bemutatja azt a sablont amelyet a KobrA módszer erre a célra javasol ezt a Fusion módszertől vették át.
- A táblázat legfontosabb elemei az **Assumes** és **Result** klózok, amelyek a műveletek elő- és utófeltételeit reprezentálják. Ezek alapvetően fontosak a műveletek helyes működésének vizsgálatakor.
  - A műveletek elő- és utófeltételei deklaratív módon írják le azok hatásait, ez azt jelent, hogy azt definiáljuk, hogy a művelet mit csinál és a „hogyan csinálja ?” kérdésre itt még nem adunk választ.

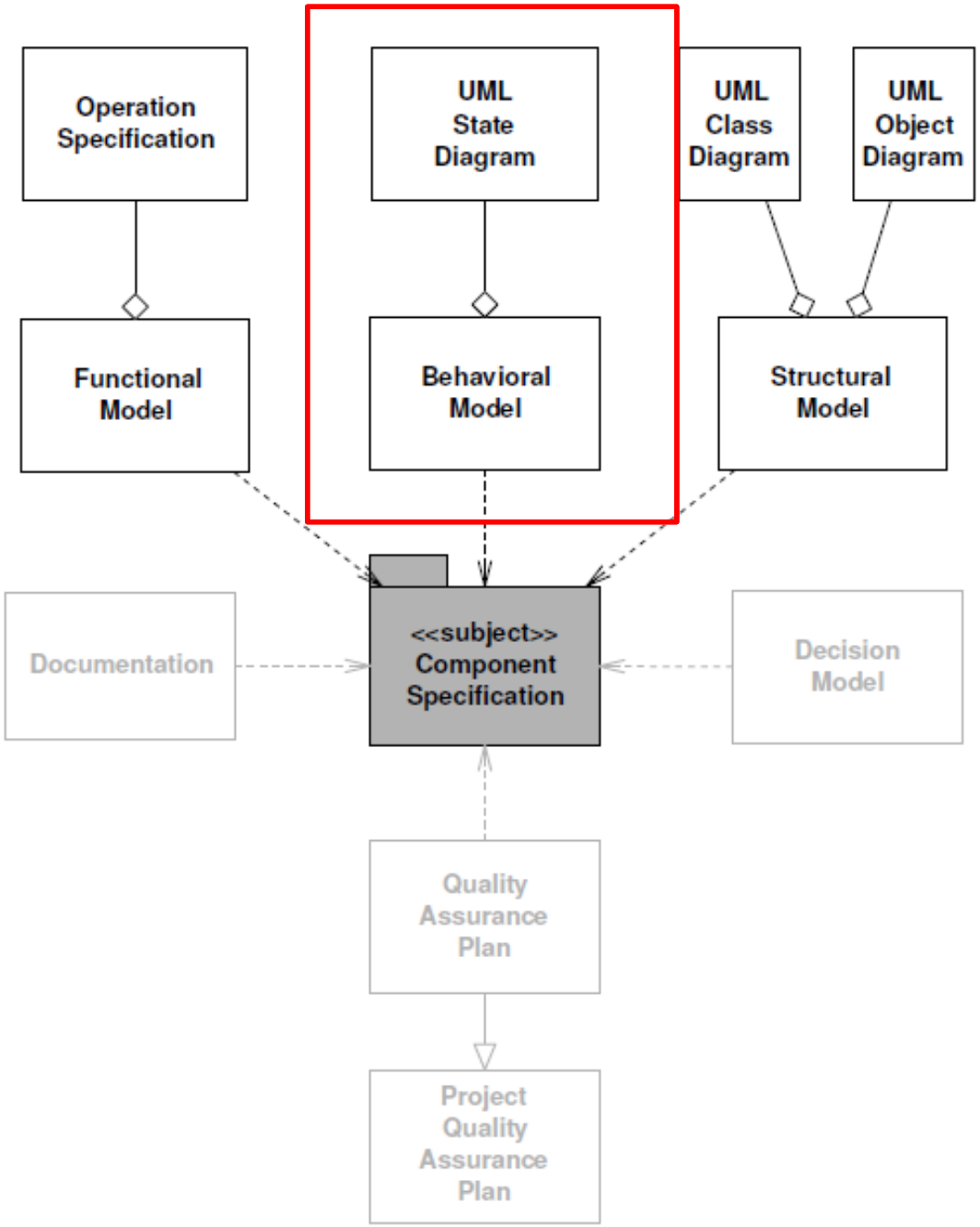
Table 2.3. Operation specification template according to the KobrA method and Fusion

Name	Name of the operation
Description	Identification of the purpose of the operation, followed by an informal description of the normal and exceptional effects.
Constraints	Properties that constrain the realization of and implementation of the operation.
Receives	Information input to the operation by the invoker.
Returns	Information returned to the invoker of the operation.
Sends	Signals that the operation sends to imported components (can be events or operation invocations).
Reads	Externally visible information accessed by the operation.
Changes	Externally visible information changed by the operation.
Rules	Rules governing the computation of the result.
Assumes	Weakest precondition on the externally visible states of the component and on the inputs (in receives clause) that must be true for the component to guarantee the postcondition (in the result clause).
Result	Strongest postcondition on the externally visible properties of the component and the returned entities (returns clause) that becomes true after execution of the operation with a valid assumes clause.

Név	Leírás
Leírás	A művelet informális leírása.
Megszorítások	Megszorítások a megvalósításra.
Kap	Input adatok, paraméterezés leírása.
Visszaad	Return adatok, eredmény leírása.
Olvas	Adatok, amikhez a művelet hozzáfér
Változtat	Adatok, melyeket a művelet megváltoztat
Szabály	Művelet formális leírása
Feltételez	Leggyengébb előfeltétel
Eredményez	Legerősebb utófeltétel

Table 2.4. Example operation specification for *VendingMachine::SelectItem*

Name	SelectItem
Description	User selects an item to buy.
Constraints	<none>
Receives	`SelectedItem`.
Returns	`SelectedItem.Price` OR `SelectedItem`.
Sends	`SelectedItem.Price` to the Display OR/AND `DispenseItem` to the Dispenser.
Reads	`Amount` from the CashUnit.
Changes	Number of items in the DispenserUnit.
Rules	If [*CoinsInserted*] AND *SelectedItem* = *Abort* dispenseChange; If [*Item.Empty*] display empty; If ([*Idle*]*OR*[*Amount* < *Item.Price*]) AND [!*ItemEmpty*] display `SelectedItem.Price`; If [*CoinsInserted*] and [*Amount* >= *Item.Price*] dispenseItem();
Assumes	Vending machine is `Idle` OR `CoinsInserted`.
Result	Item AND/OR Change (if applicable) dispensed; OR Item Empty/Item.Price displayed.


Operation Specification
UML State Diagram
UML Class Diagram
UML Object Diagram
Functional Model
Behavioral Model
Structural Model
Documentation
<<subject>> Component Specification
Decision Model
Quality Assurance Plan
Project Quality Assurance Plan

A komponens viselkedési specifikációja

- Objektum-orientált (OO) paradigma: adat és funkcionalitás egybezárását (encapsulation) jelenti első közelítésben.
- Ez azt jelenti, hogy egy objektumnak vannak állapotai és állapotátmenetei, amelyek az objektum működése során bekövetkezhetnek.
- A *komponens paradigma* magába foglalja az objektum-orientált technológia alapelveit, ezért a komponens paradigma pontosan ezen elveken alapul.
- A komponenseknek lehetnek állapotai, ha ilyenek nem léteznek, akkor funkcionális komponensről beszélünk, amelyeknek tehát nincsenek kívülről látható állapotai és állapotátmenetei.

- A viselkedési specifikáció megmutatja, hogyan viselkedik külső hatásokra a komponensünk.
- UML állapotdiagramok formájában készül el
- A funkcionális specifikációban megadott **Assumes** and **Result** klózokra koncentrál (2.3. táblázat).
- Ha egy komponensnek nincsenek állapotai (tisztán funkcionális komponens), akkor nincs szükség viselkedési modellre.
- A viselkedési specifikáció a komponensek viselkedését a kívülről látható állapotaival és egy külső esemény által kiváltott állapotváltozásokkal írja le.
  - Egy komponens egy külső eseményre az állapotától függően különböző módon reagálhat.
  - Egy átmenetet (transition), vagy egy állapotváltozást egy esemény vált ki, amely rendszerint egy, a komponenshez érkező üzenet. Egy őrfeltétel (guard) kapcsolható egy átmenethez, amelynek igaznak kell lennie, ha az átmenet végrehajtható.

Az árusító automata viselkedési specifikációja

- A 2.12 ábra bemutatja az árusító automata viselkedési specifikációját egy UML állapotdiagramon
- A 2.5. táblázat az pedig a megfelelő állapotokat sorolja fel.
- A diagram és a táblázat többé kevésbé ugyanazon információt hordozza. A diagramot könnyebb megérteni, a táblázatot pedig könnyebb automatikusan feldolgozni.

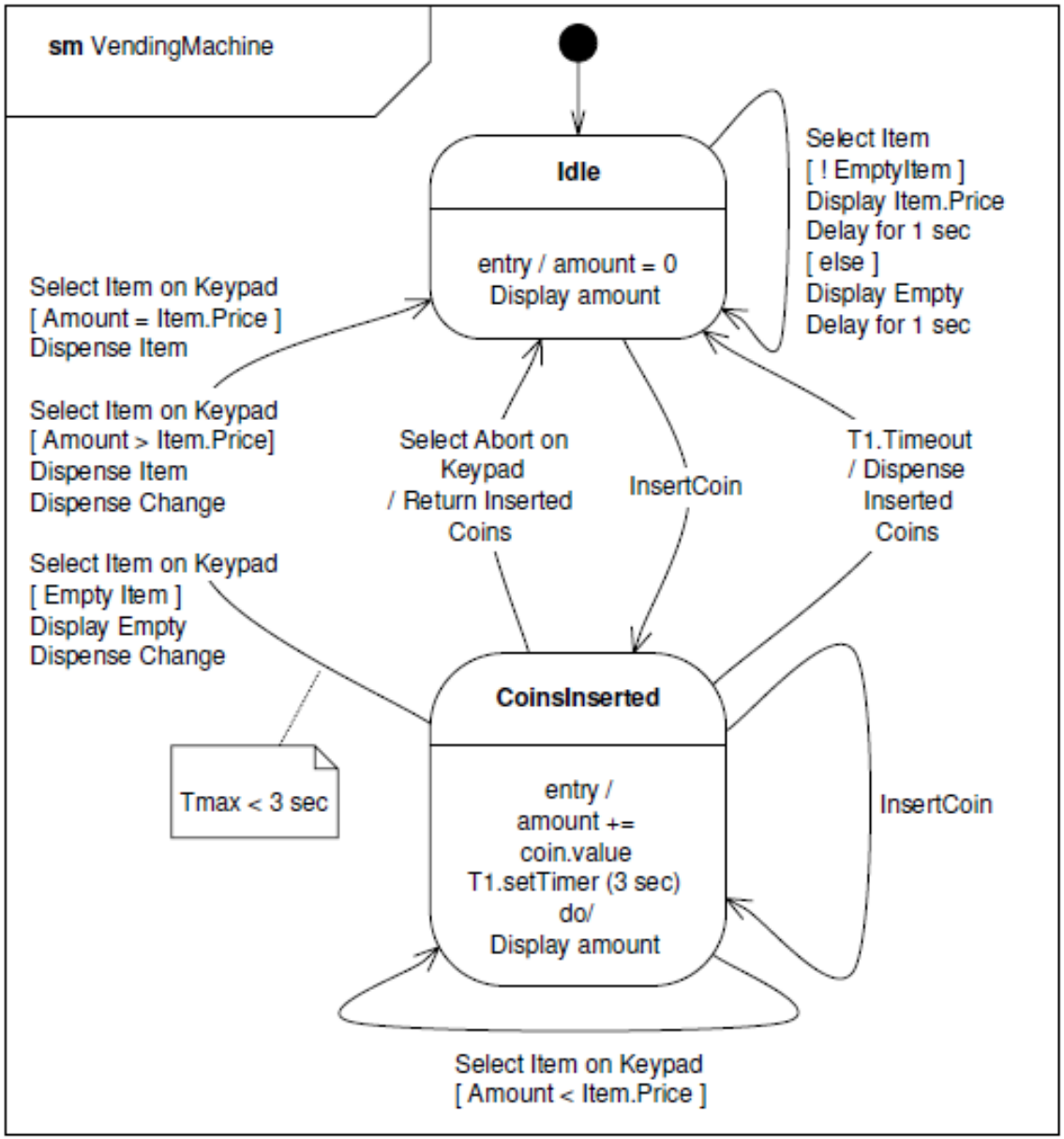


Fig. 2.12. Behavioral specification of the vending machine in the form of a state-chart diagram

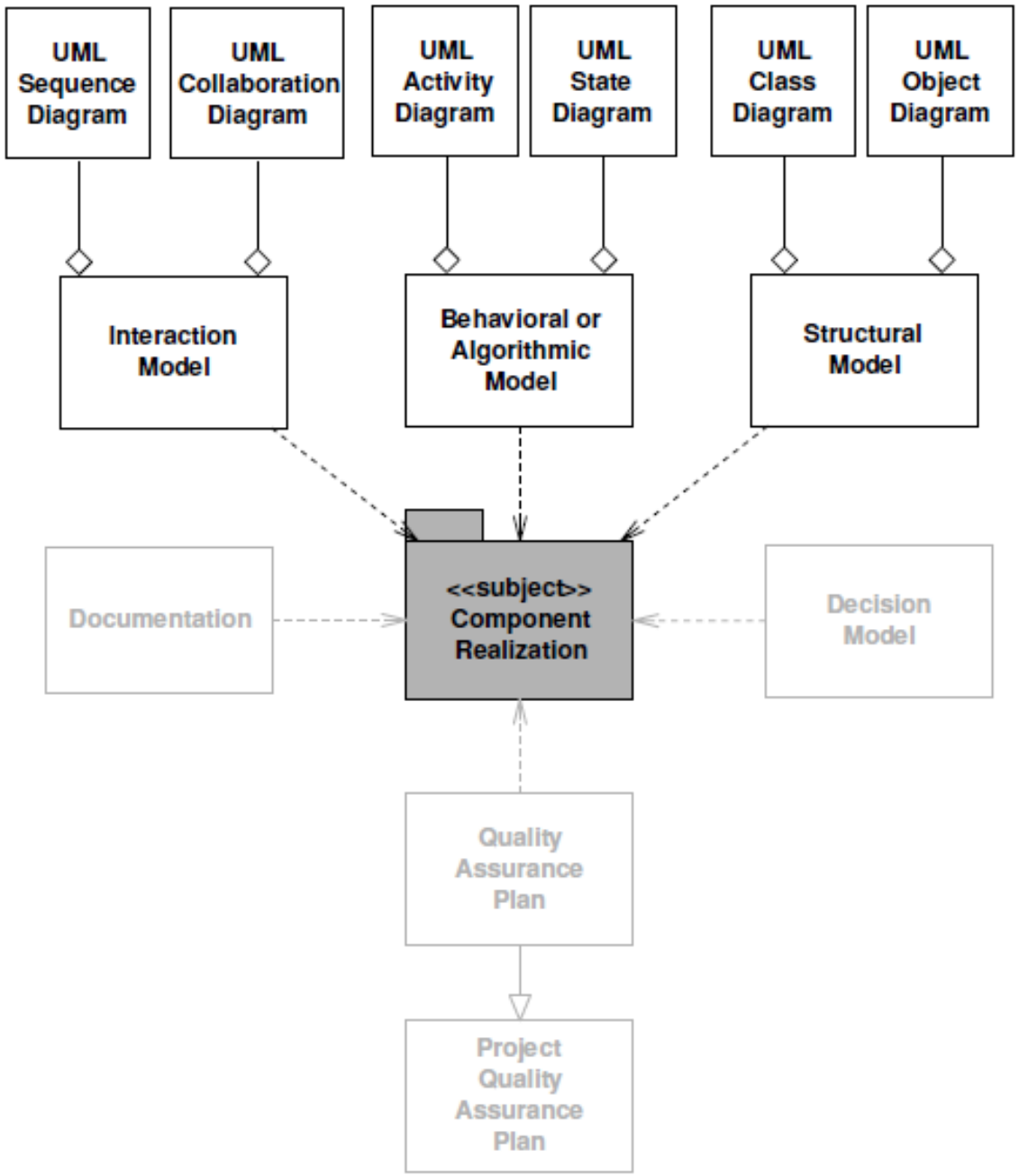
Table 2.5. Behavioral specification of the vending machine in the form of a state transition table

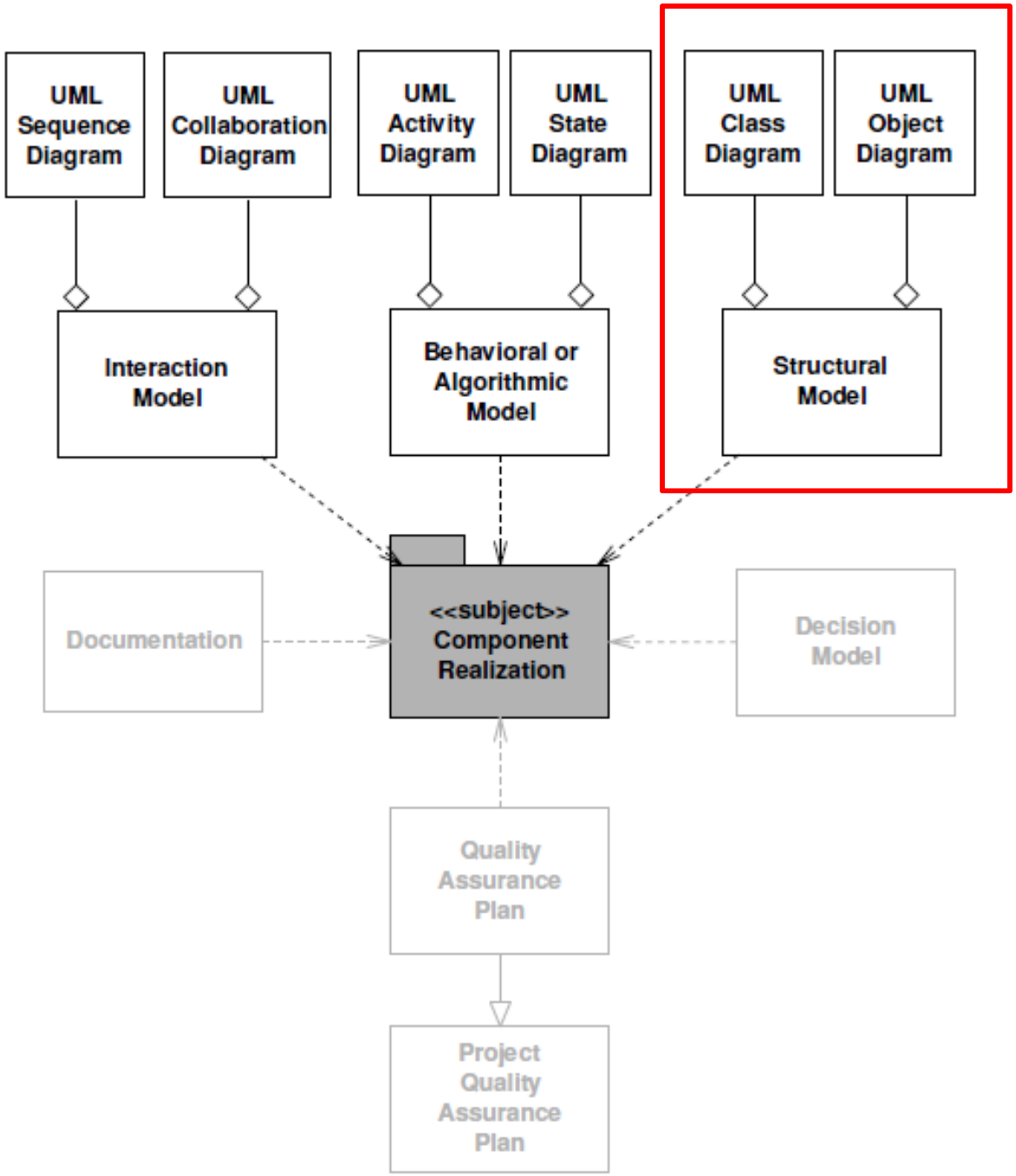
Initial State & Precond (Guard)	Operation, Transition	Result	Final State & Postcond
Idle AND [!*EmptyItem*]	SelectItem (Item)	Display Price	Idle
Idle AND [*EmptyItem*]	SelectItem (Item)	Display Empty	Idle
Idle	InsertCoin (Coin)	Display Amount	CoinsInserted
CoinsInserted	InsertCoin (Coin)	Display Amount	CoinsInserted
CoinsInserted AND [*Amount* < *Item.Price*]	SelectItem (Item)		CoinsInserted
CoinsInserted AND [*Amount* == *Item.Price*]	SelectItem (Item)	dispense Item	Idle
CoinsInserted AND [*Amount* > *Item.Price*]	SelectItem (Item)	dispense Item dispense Change	Idle
CoinsInserted [*EmptyItem*]	SelectItem (Item)	Display Empty Dispense Change	Idle
CoinsInserted [*Item* == *Abort*]	SelectItem (Item)	Dispense Cash	Idle

Komponens megvalósítása

- A komponens megvalósítása azon dokumentumoknak a gyűjteménye, amelyek együttesen leírják a komponens megvalósításának módját.
- Mindent tartalmaz, ami a specifikáció alapján történő implementációhoz szükséges.
- Magasabb szintű komponensek az alacsonyabb szintű komponensek kompozíciójával valósíthatók meg. Ezek a magasabb szintű komponens számára a szerver szerepét töltik be.
- A megvalósítás tehát tartalmazza
  - az alkomponensek specifikációját, a tőlük elvárt szolgáltatások interfészét és a
  - saját megvalósítását, amely alatt a komponens nem látható,privát részének megvalósítását értjük.

- A megvalósítás dokumentumai tartalmazzák:
  - A komponens belső felépítésének strukturális modelljét UML osztály és objektum diagramok formájában
  - A komponensben megtestesülő algoritmusok specifikációját UML aktivitás diagramok formájában
  - Más komponensekkel való kölcsönhatás leírását UML szekvencia és együttműködési diagramokkal.
  - A 2.13. ábra bemutatja a komponens megvalósításának UML leírását, amelyet a szokásos módon részleteiben elmagyarázunk a következőkben.


UML Sequence Diagram
UML Collaboration Diagram
UML Activity Diagram
UML State Diagram
UML Class Diagram
UML Object Diagram
Interaction Model
Behavioral or Algorithmic Model
Structural Model
<<subject>> Component Realization
Documentation
Decision Model
Quality Assurance Plan
Project Quality Assurance Plan


UML Sequence Diagram
UML Collaboration Diagram
UML Activity Diagram
UML State Diagram
UML Class Diagram
UML Object Diagram
Interaction Model
Behavioral or Algorithmic Model
Structural Model
<<subject>> Component Realization
Documentation
Decision Model
Quality Assurance Plan
Project Quality Assurance Plan

Megvalósítás strukturális modellje

- A megvalósítás strukturális modellje leírja
  - azon osztályok fajtáit és kapcsolatait (UML diagramok), amelyekből a komponens felépül valamint a
  - komponens belső architektúráját.
- Az osztálydiagramban a fókuszban lévő komponenst <<subject>> címkével jelezzük.
- A 2.14. ábra az árusító automata **VendingMachine** komponense megvalósításának UML osztálydiagramját mutatja be.
  - Az UML diagram a komponens belső struktúráját reprezentálja.
- A megvalósítás strukturális modellje tipikusan a specifikáció strukturális modelljének finomítása.

emlékeztetőül
a specifikáció

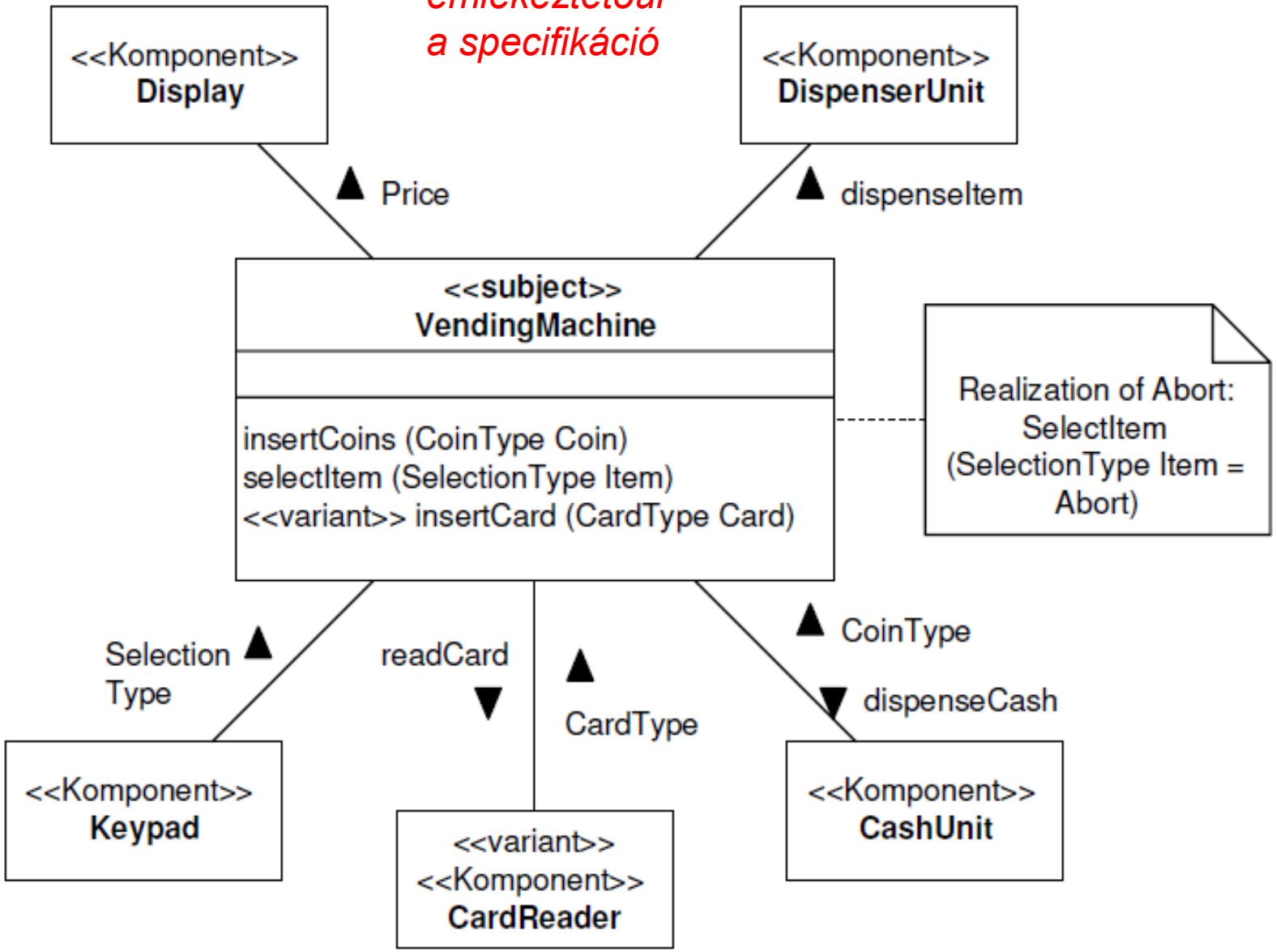


Fig. 2.11. Specification structural model for the vending machine component

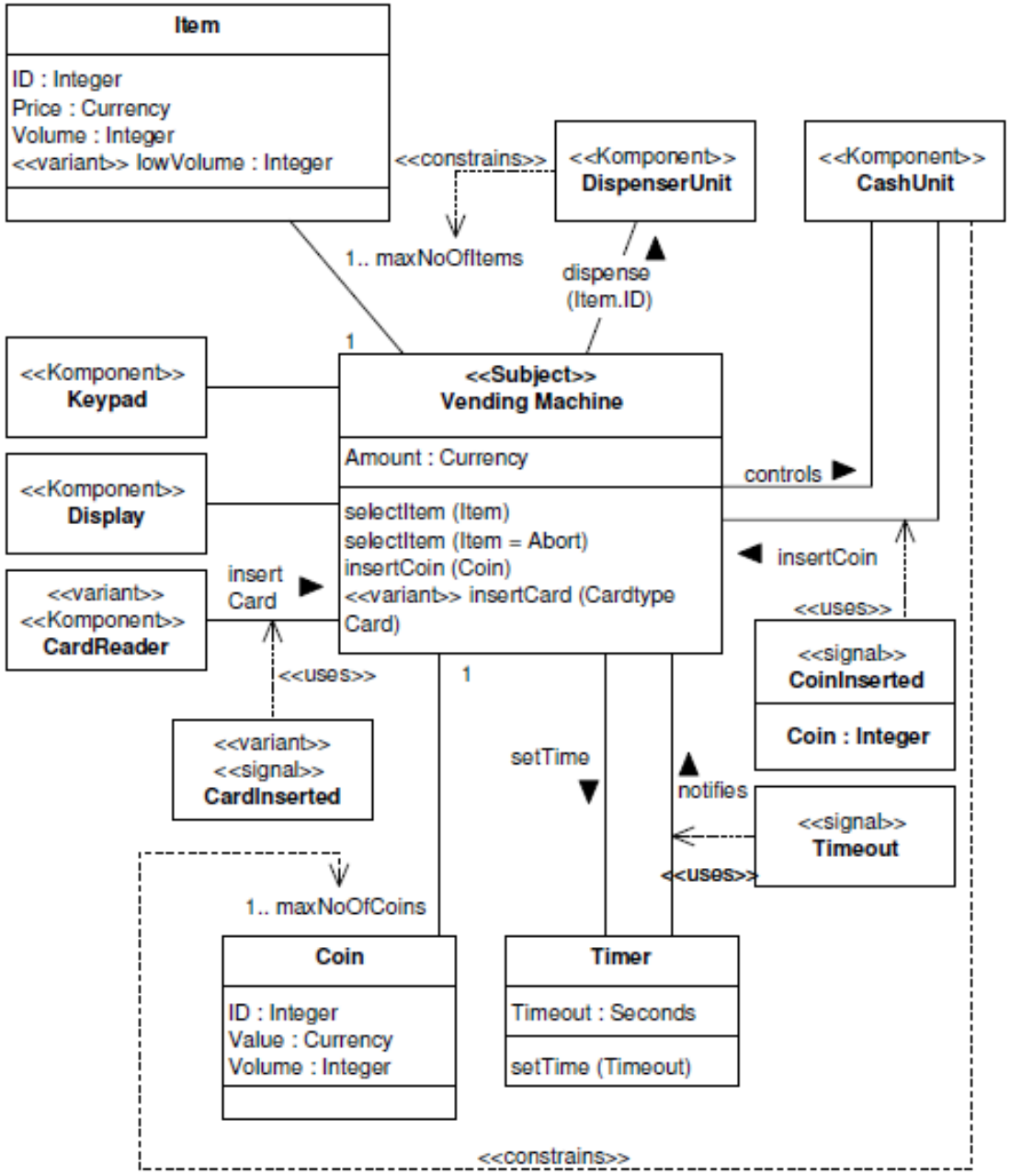
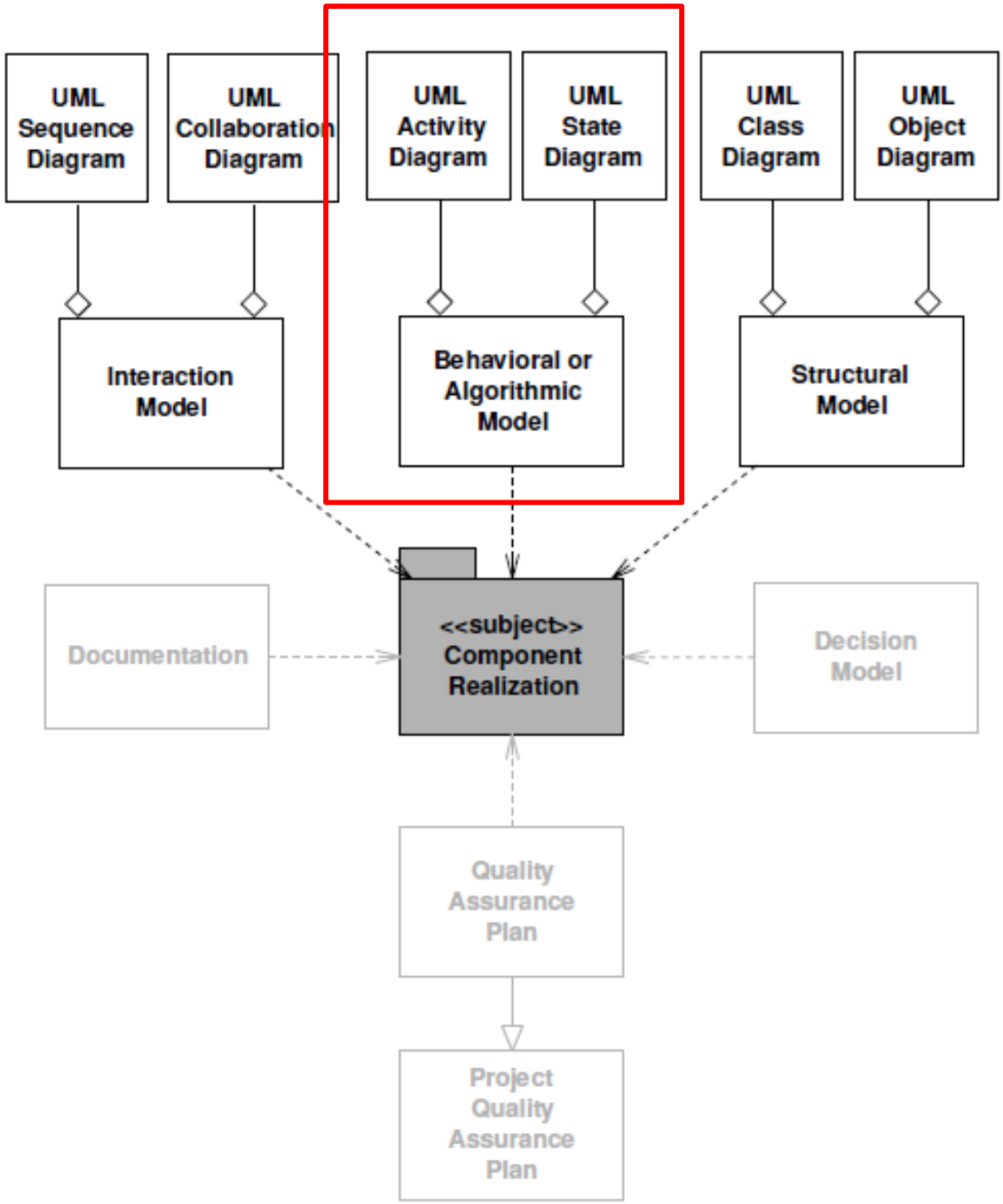


Fig. 2.14. Realization structural model of the *VendingMachine* component

- A megvalósítás szintjén olyan elemek is megjelennek, amelyek a specifikáció szintjén nem voltak láthatók, mert ott nem bírtak jelentőséggel.
  - A 2.14. ábrán látható, hogy vannak osztályok, amik
    - nem komponensek, ezek a való világ objektumainak belső reprezentációs osztályai (pl. Timer) és
    - alárendelt objektumok, amelyeket <<Komponent>> jelöl.
- Szükségünk lehet az eladott termékek listájára és azokéra is, amelyek még eladhatók, mivel a valóságot akarjuk leképezni, ez lényeges része a beágyazott rendszernek. A lista mérete függ az adagoló (dispenser) méretétől ezért a diagramunkon megjelenik egy függőségi asszociáció az adagoló irányában.

- Hasonlóképpen szükségünk a lehetséges pénzérmék listájára is, amelyeket a *Cashunit* elfogadhat.
  - Ez viszont a *Cashunit* típusától függhet.
  - A Cashunit egységet viszont egy külső szolgáltatótól vesszük.
- Néhány alárendelt komponens esetében hozzákezdhetünk az elvárt interfész definiálásához. Ez azonban erősen iteratív folyamat és attól is nagyban függ, hogy milyen harmadik parti (third-party) komponens implementációt találunk.
  - A kiválasztott implementált komponens ismeretében akár megváltoztathatjuk az elvárt interfészt is.
- A komponensek közötti néhány interakció esetében konkrétabbak vagyunk, mások esetében kevésbé specifikusak.

- A 2.14. ábra definícióit még finomíthatjuk később iteratív módon, attól függően, hogy egyre tisztább lesz a képünk a megvalósításukkal kapcsolatban.


UML Sequence Diagram
UML Collaboration Diagram
UML Activity Diagram
UML State Diagram
UML Class Diagram
UML Object Diagram
Interaction Model
Behavioral or Algorithmic Model
Structural Model
Documentation
<<subject>> Component Realization
Decision Model
Quality Assurance Plan
Project Quality Assurance Plan

Megvalósítás algoritmikus modellje

- A komponens műveleteit részletesebben specifikáljuk.
- Aktivitás diagramokat készítünk, ezekkel leírjuk azon algoritmusokat, amelyek segítségével a specifikált műveleteket meg kívánjuk valósítani.
- A 2.15. és 2.16. ábrák az **insertcoin** és **selectItem** műveletek UML aktivitásdiagramjait tartalmazzák.

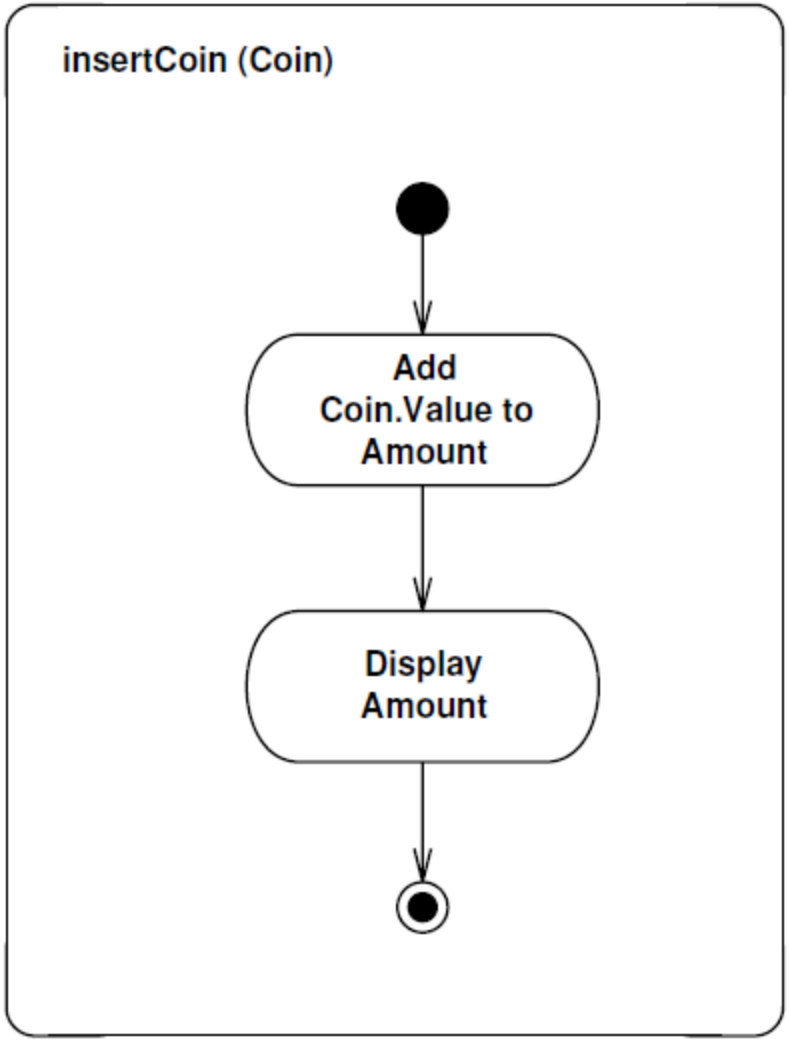


Fig. 2.15. Realization algorithmic model for the operation *insertCoin* of the *VendingMachine* component

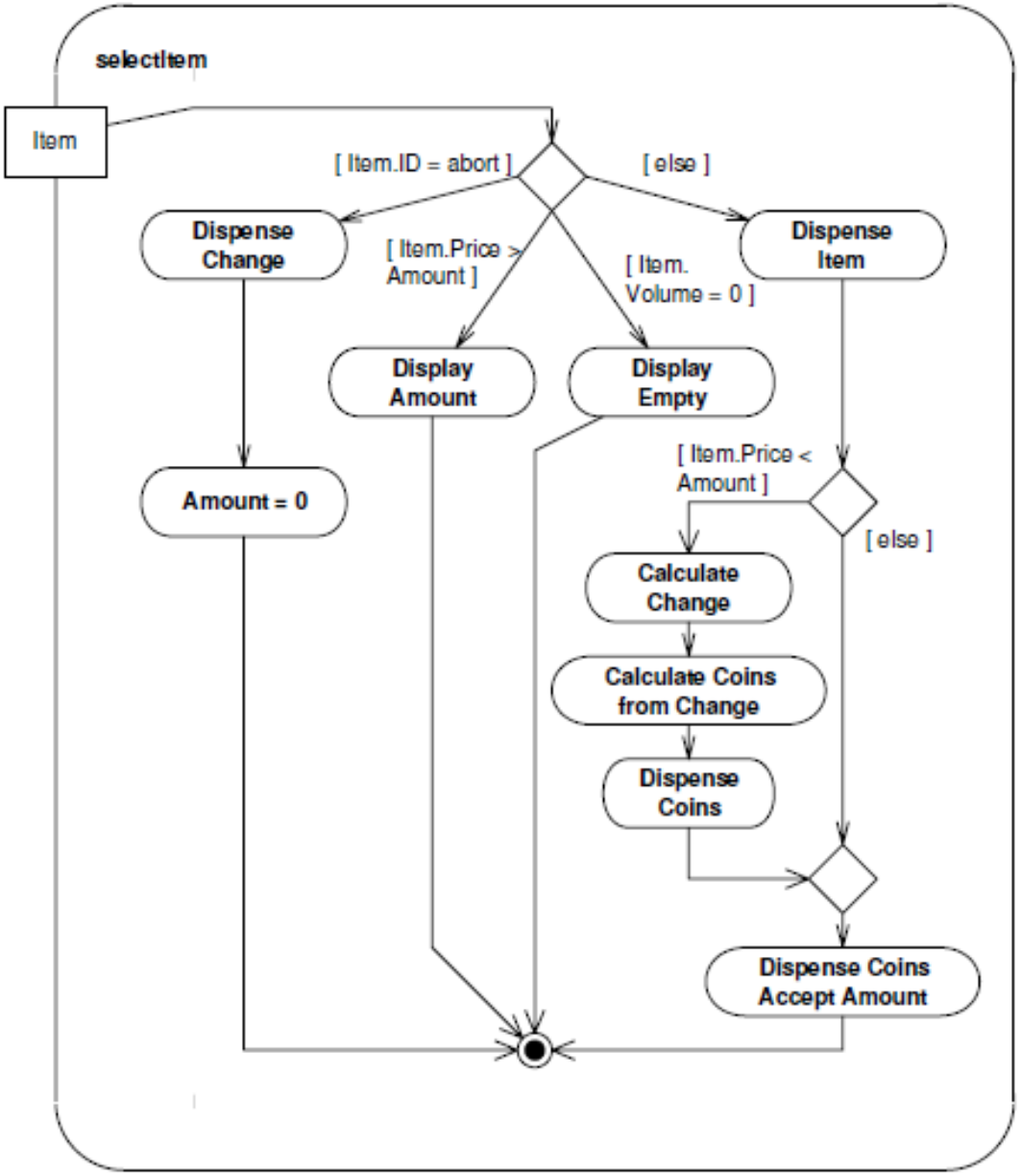
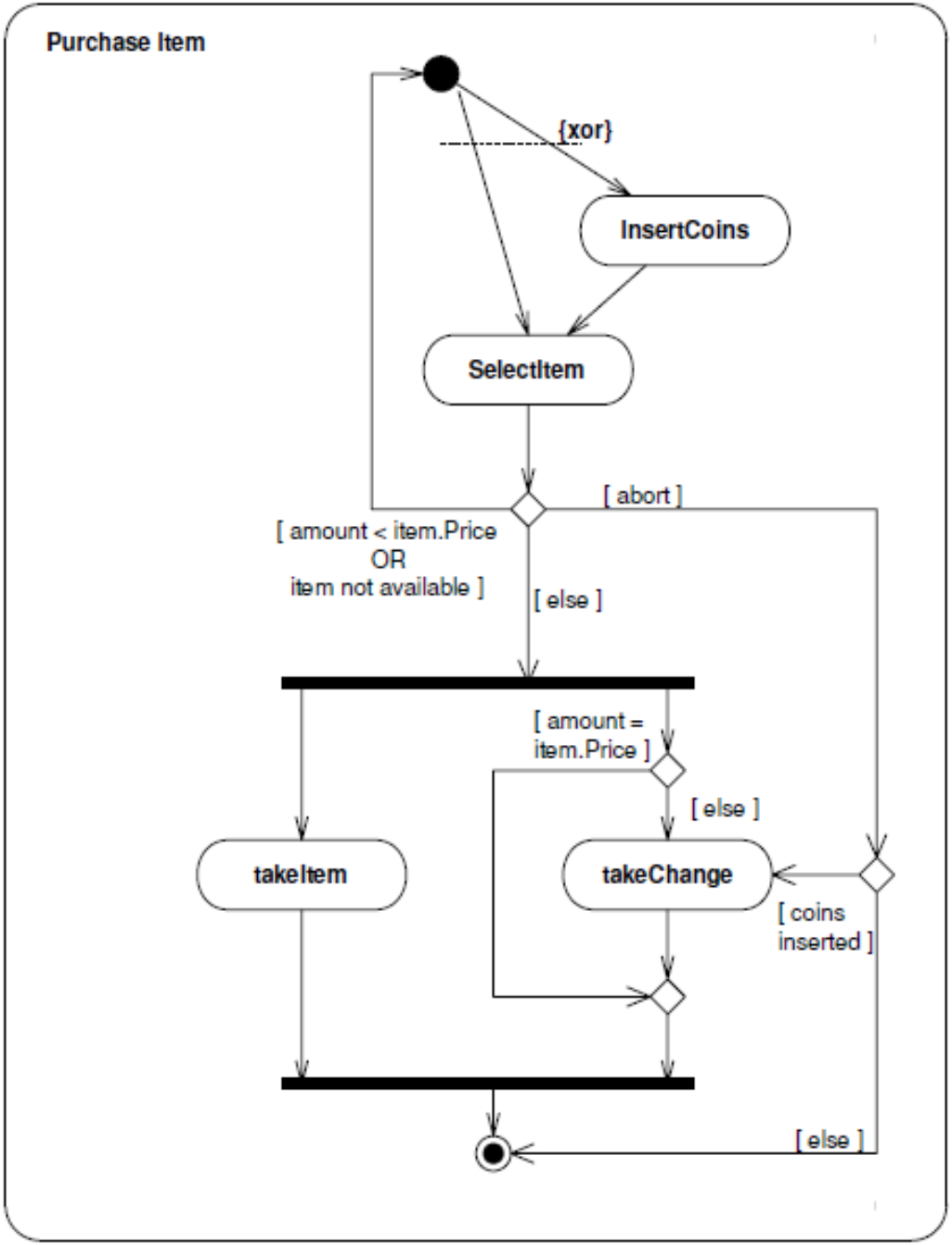


Fig. 2.16. Realization algorithmic model for the operation *selectItem* of the *VendingMachine* component

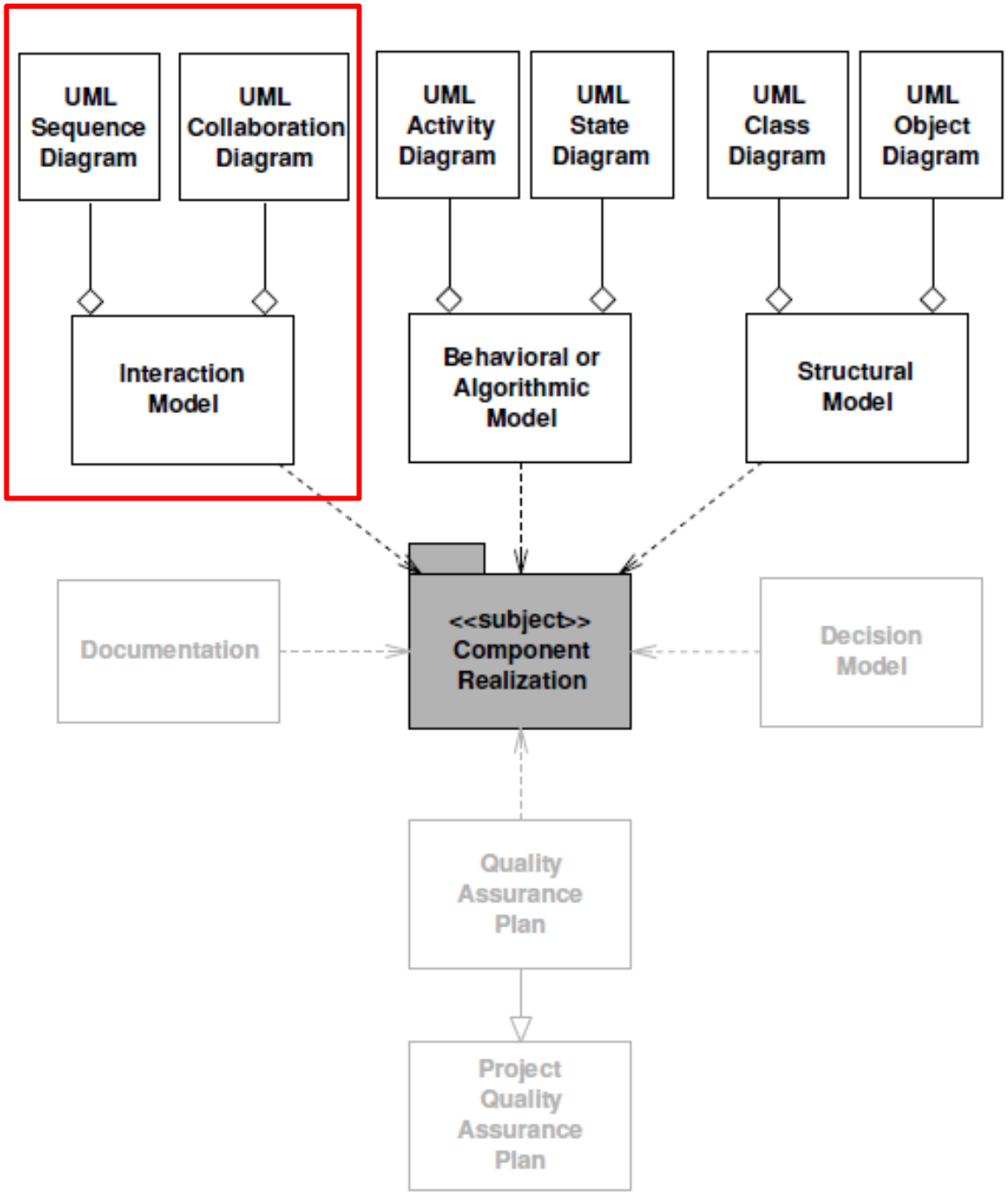
- Vessük össze a(z előző) 2.16. ábrán definiált **SelectItem** művelet aktivitásdiagramját és a **Purchase Item** aktivitásdiagramját, amelyet a 2.8. ábrán definiáltunk (következő slide-on is).
  - A **SelectItem** művelet más nézőpontból és részletesebb leírását adja, hogy a felhasználó milyen módon választja ki a kiszemelt árucikket.
  - Másképpen fogalmazva a **Purchase Item** több információt hordoz a felhasználó számára a „mi történik” szempontjából, míg a **SelectItem** algoritmusának modellje a művelet belső részleteit is definiálja és a „hogyan történik „ kérdésére is kitér.

2.8. ábra



emlékeztetőül a környezeti térképből

- Néhány aktivitást a későbbiekben még finomítani fogunk annak függvényében, hogy a kapcsolatban álló, egymással együttműködő komponenseket hogyan konkretizáljuk majd.
  - Például tudjuk, hogy egy jelre várunk, de most nem mondjuk meg, hogy ezt pontosan milyen formában várjuk.
  - Ezeket további belső (privát) eljárásként kezeljük és beillesztjük őket a további modellekbe, például a megvalósítás strukturális modelljébe és hasonló módon ott definiáljuk azokat.
- Felvethető, hogy adjuk meg a komponensek megvalósítását például Java nyelven. Ez a modellezés ezen szintjén még korai, mert a továbbiakban esetleg más nyelven megírt alkalmas komponenst találunk és egy ilyen korai implementációs döntés korlátozna a komponens felhasználhatóságában.


UML Sequence Diagram
UML Collaboration Diagram
UML Activity Diagram
UML State Diagram
UML Class Diagram
UML Object Diagram
Interaction Model
Behavioral or Algorithmic Model
Structural Model
<<subject>> Component Realization
Documentation
Decision Model
Quality Assurance Plan
Project Quality Assurance Plan

Megvalósítás kölcsönhatás modellje

- A megvalósítás **kölcsönhatás modelljei** a vezérlés folyamatát mutatják be a komponens példány perspektívájából.
- A *kölcsönhatás diagramok* azt írják le, hogy a komponens példányok csoportjai egymással hogyan kollaborálnak, hogy egy műveletet, vagy egy művelet résztevékenységét megvalósítsák.
- A 2.17. ábra a **VendingMachine.insertCoin** aktivitásának UML együttműködési diagramját mutatja be.
- A 2.18. ábra pedig a **VendingMachine.selectItem** aktivitását.
- Kicsi aktivitás esetében nem szükséges az aktivitás és az együttműködési diagramok mindegyikét elkészíteni.

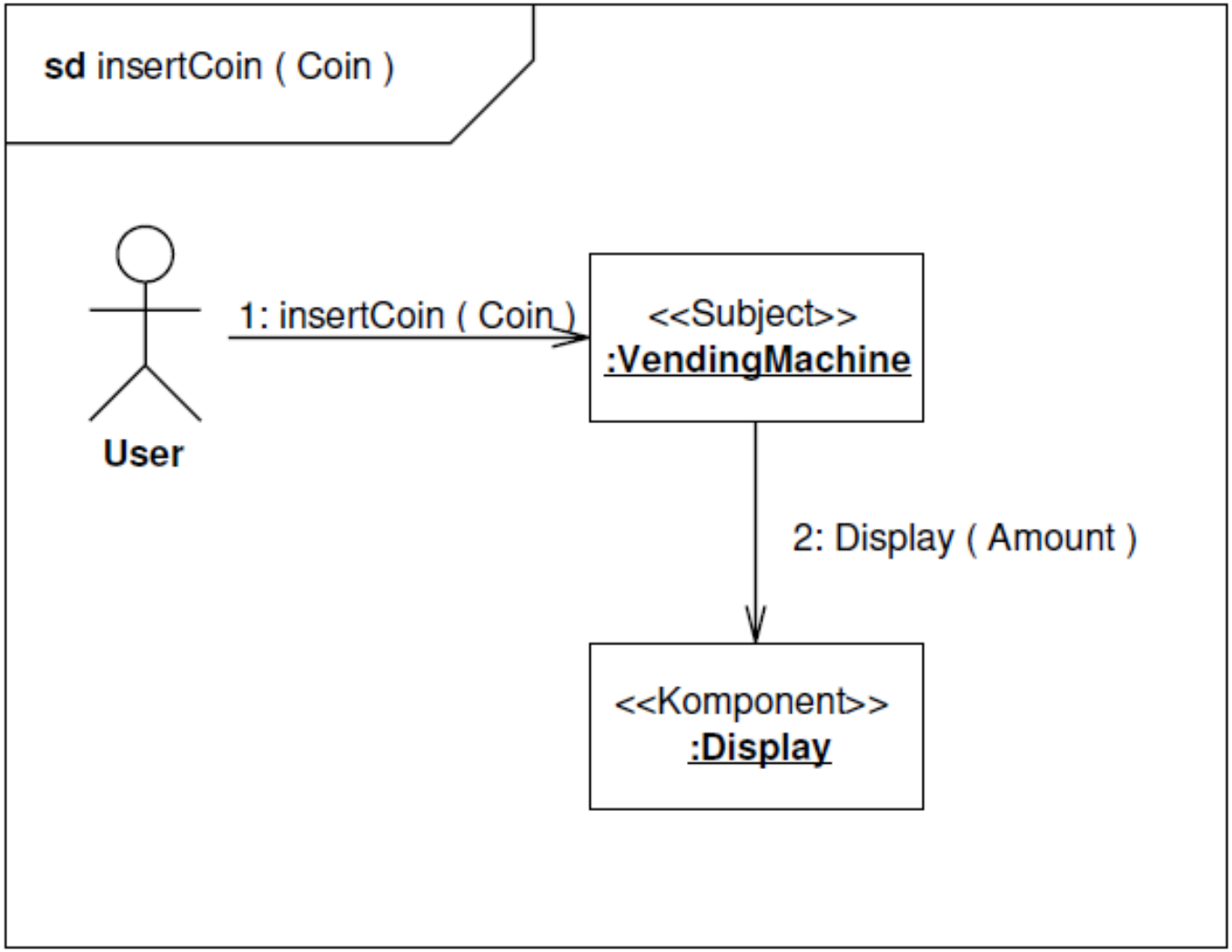


Fig. 2.17. Realization interaction model for the operation *insertCoin* of the *VendingMachine* component

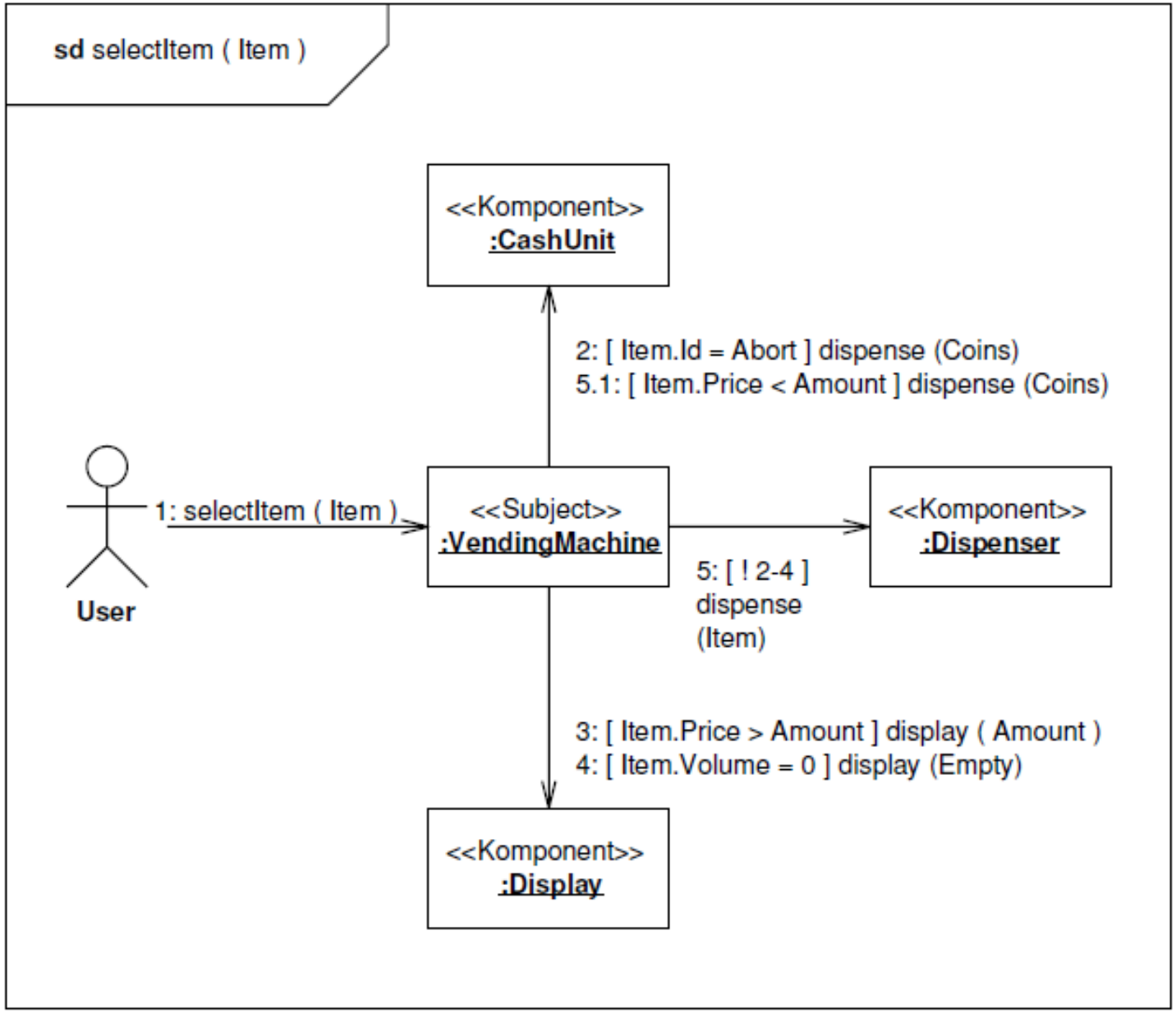


Fig. 2.18. Realization interaction model for the operation *selectItem* of the *VendingMachine* component

Foglaljuk össze az eddigieket!

- Ezzel elkészítettük a legfelső szint első modelljeit.
- Az alrendszerek egy részéről konkrét, másik részéről kevésbé konkrét elképzelésekkel rendelkezünk.
- A következő lépésekben
  - már létező komponenseket keressünk, amely megfelelnek az elvárásainknak majd
  - integráljuk ezeket a rendszerünkbe
  - ez megfelel a megtestesítés (embodiment) lépésének.