Software Engineering 1

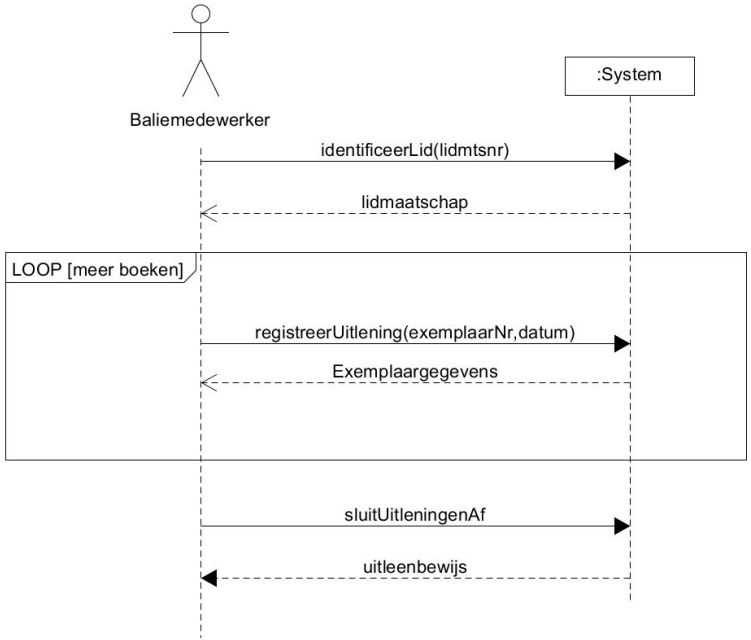
Samenvatting P3 & P4

KdG Karel de Grote Hogeschool

SSD (System Sequence Diagram)

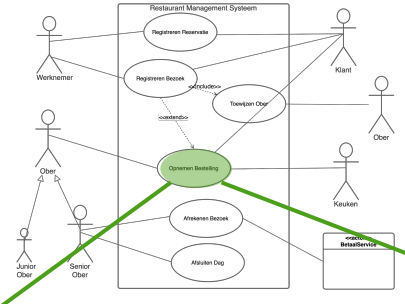
System Sequence diagrammen

- Zijn **interactiediagrammen**
  - Interactie tussen actor en systeem als geheel
- Zijn **sequentiediagrammen**
  - Geven de volgorde van interacties weer, van boven naar onder



Drie niveaus van gedrag van een systeem

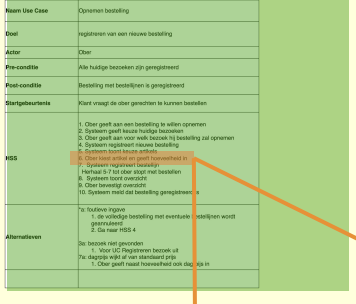
Use case diagram



Systeemniveau

Volledige beschrijving

Use case beschrijving



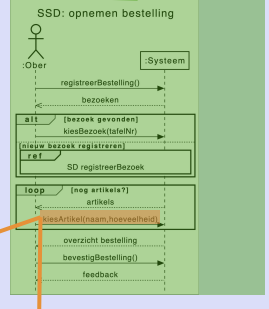
Naam Use Case	Opnemen bestelling
Doel	registreren van een nieuwe bestelling
Actor	Ober
Pre-conditie	Alle huidige bezoeken zijn geregistreerd
Post-conditie	Bestelling met bestellijnen is geregistreerd
Startgebeurtenis	Klant vraagt de ober gerechten te kunnen bestellen
HSS	1. Ober geeft aan een bestelling te willen opnemen 2. Systeem geeft keuze huidige bezoeken 3. Ober geeft aan voor welk bezoek hij bestelling zal opnemen 4. Systeem registreert nieuwe bestelling 5. Ober kiest artikel en geeft hoeveelheid in 6. Systeem registreert bestelling Herhaal 5-7 tot ober stopt met bestellen 8. Systeem toont overzicht 9. Ober bevestigt overzicht 10. Systeem meld dat bestelling geregistreerd is
Alternatieven	7a: foutieve ingave 1. de volledige bestelling met eventuele bestellijnen wordt geannuleerd 2. Ga naar HSS 4 3a: bezoek niet gevonden 1. Voer UC Registreren bezoek uit 7a: dagprijs wijkt af van standaard prijs 1. Ober geeft naast hoeveelheid ook dagprijs in

Contract CO5 : kiesArtikel

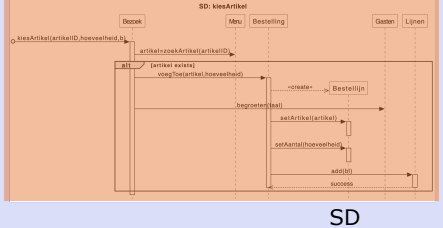
Operatie:	kiesArtikel(artikelID, hoeveelheid)
Kruisverwijzing:	Use Case(s): Opnemen Bestelling
Preconditie:	Instantie b van Bestelling was aanwezig
Postcondities:	Instantie bl van Bestellijn werd gecreëerd Instantie bl van Bestellijn werd gelinkt met instantie b van Bestelling Instantie bl van Bestellijn werd gelinkt met instantie mi van MenuItem bl.hoeveelheid kreeg de inputwaarde

Operation contract

SSD



Enkel interacties



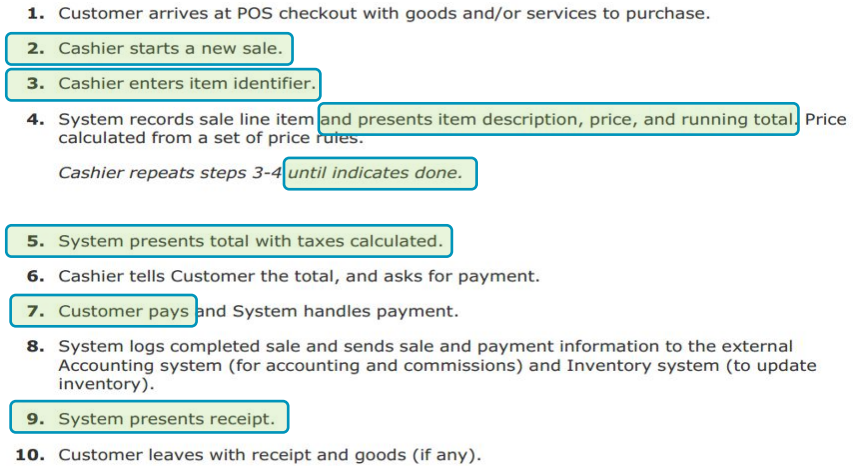
SD

Use case niveau

Operatie/methode niveau

Werkwijze System Sequence Diagram opstellen

1. **Actoren bepalen of overnemen uit de use case**
   - Eén primary actor, :Actornaam, aan de linkerkant
   - :Systeem als blackbox aan de rechterkant
   - mogelijks meerdere supporting actoren meer naar rechts
   - Titel van Use Case bovenaan
2. **Bepalen wat interacties zijn tussen actoren en systeem**



1. Customer arrives at POS checkout with goods and/or services to purchase.
2. Cashier starts a new sale.
3. Cashier enters item identifier.
4. System records sale line item and presents item description, price, and running total. Price calculated from a set of price rules.

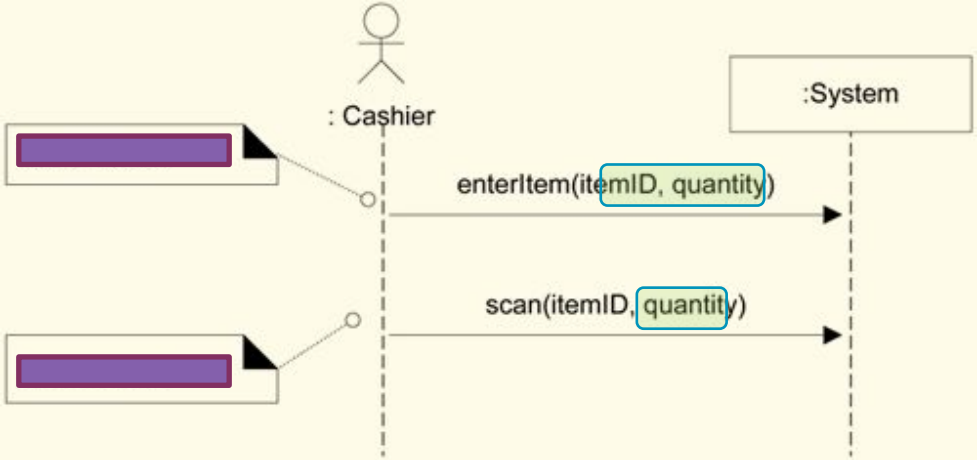
   Cashier repeats steps 3-4 until indicates done.

5. System presents total with taxes calculated.
6. Cashier tells Customer the total, and asks for payment.
7. Customer pays and System handles payment.
8. System logs completed sale and sends sale and payment information to the external Accounting system (for accounting and commissions) and Inventory system (to update inventory).
9. System presents receipt.
10. Customer leaves with receipt and goods (if any).

Werkwijze System Sequence Diagram opstellen

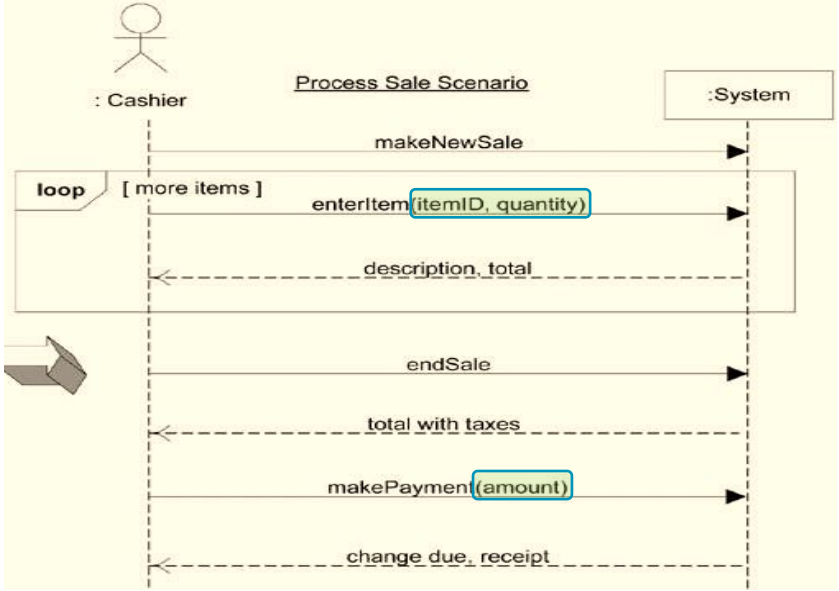
3. **Input messages benoemen**
   - Als een abstracte beschrijving van het event doel. Verwoord het in de **gebiedende wijs, als opdracht.**

- **Niet** in termen van **oplossing (dit is Ontwerp ipv Analyse)**
  - Welke van de 2, *scan* of *enterItem,* is beter?



Werkwijze System Sequence Diagram opstellen

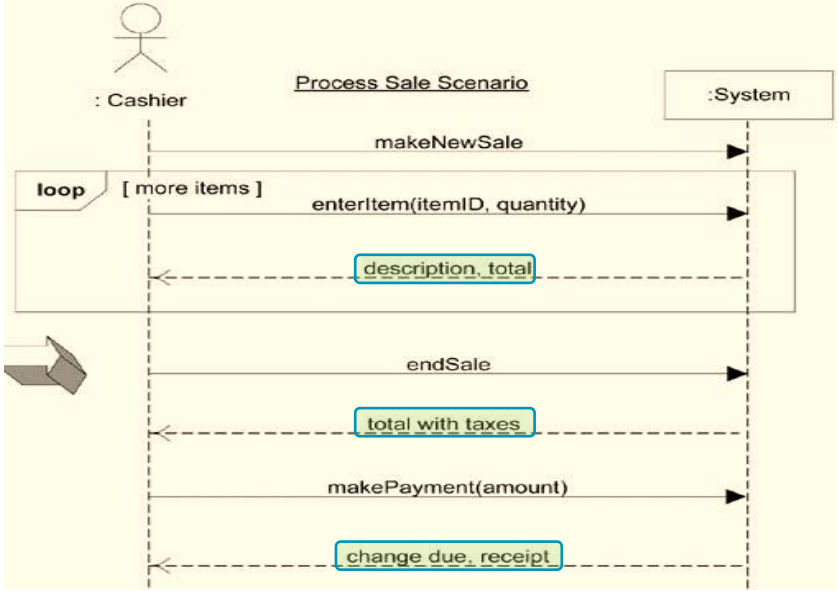
4. **Parameters op input messages bepalen**
   - Welke gegevens moeten we meegeven opdat het systeem deze operatie kan uitvoeren?
   - De gegevens kan je terugvinden als attributen in het domeinmodel. Alle gegevens moeten daar ook terug te vinden zijn. **Afstemming tussen de modellen!**



Werkwijze System Sequence Diagram opstellen

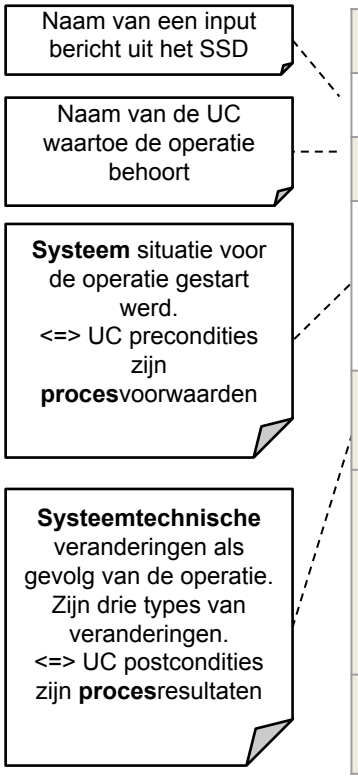
5. **Antwoorden/return messages bepalen**
   - Antwoord het systeem op een input message?
     - Is niet verplicht
   - Zoja: welke gegevens worden er teruggestuurd?

Let op vorm: onderbroken lijn met pijl in vorm van een vinkje



OC’s (Operation Contracts)

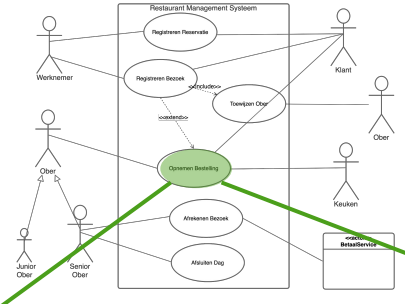
Operation contracts



Contract CO1 : enterItem	
Operatie	enterItem(itemID,quantity)
Kruisverwijzing	Use Case(s) Process Sale
Preconditie	**Instantie binnen scope van UC ophalen** ● Instantie sale van Sale was aanwezig ● Instantie pd van ProductDescription was aanwezig (via argument)
Postconditie	**Instantie creaties en verwijderingen** ● Instantie sli van SalesLineItem werd gecreëerd
	Link creaties en verwijderingen ● Instantie sli van SalesLineItem werd gelinkt met instantie s van Sale ● Instantie sli van SalesLineItem werd gelinkt met instantie pd van ProductDescription
	Veranderingen in attribuutwaarden ● sli.quantity kreeg de waarde quantity (via argument)

Drie niveaus van gedrag van een systeem

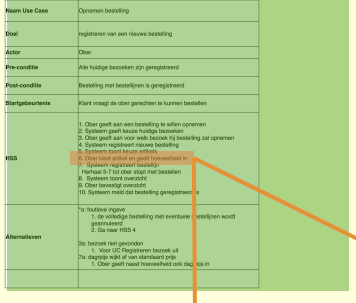
Use case diagram



Systeemniveau

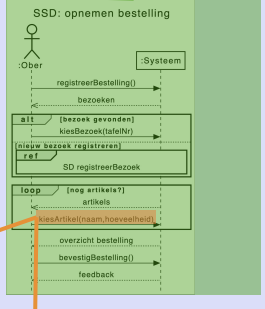
Volledige beschrijving

Use case beschrijving

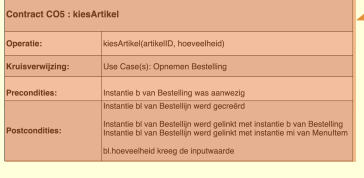


Enkel interacties

SSD

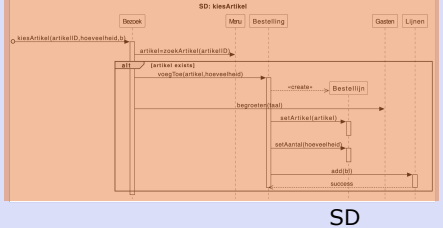


Use case niveau



Contract CO5 : kiesArtikel	
Operatie:	kiesArtikel(artikelID, hoeveelheid)
Kruisverwijzing:	Use Case(s): Opnemen Bestelling
Preconditities:	Instantie b van Bestelling was aanwezig
Postcondities:	Instantie bl van Bestelllijn werd gecreëerd Instantie bl van Bestelllijn werd gelinkt met instantie b van Bestelling Instantie bl van Bestelllijn werd gelinkt met instantie mi van MenuItem bl.hoeveelheid kreeg de inputwaarde

Operation contract



SD

Operatie/methode niveau

Werkwijze: hoe een Operation Contract (OC) opstellen?

1. **Bepalen welke operaties er zijn in één UC/SSD**
   - Alle volle pijlen in SSD = input messages
   - Operaties hebben dezelfde namen als input messages, en bevatten de reacties op deze vraag
2. **Bepalen postcondities:**
   - We vermelden enkel **systeemveranderingen** in OC postcondities (⇔ UC postcond.)
     1. Een OC focust op systeem, niet het UC-proces
     2. Bijv.: "Lidmaatschap werd gecontroleerd" is een proces/UC resultaat, maar wordt niet vermeld in OC
   - Drie soorten van postcondities:
   1. Instantie creaties en verwijderingen
      - Vorm: "Een instantie x van Xyz werd gecreëerd/verwijderd"
   2. Link creaties en verwijderingen
      - Vorm: "Instantie x werd gelinkt/ontkoppeld met instantie y van Xyz"
   3. Veranderingen in attribuutwaarden
      - Vorm: "x.attribuutnaam kreeg de waarde "..." "
3. **Bepalen precondities:**
   - Bepaal de precondities op basis van de postcondities: wat moet er bestaan voor de operatie om deze correct te kunnen uitvoeren?
   - We vermelden enkel systeemstatus in OC precondities (⇔ UC precondities)
     - Bijv.: welke objecten zijn aanwezig, welke links zijn aanwezig

CRUD als controle

- **CRUD = Create, Read, Update, Delete**
- **Dit zijn de 4 soorten basisfuncties van gegevensopslag**

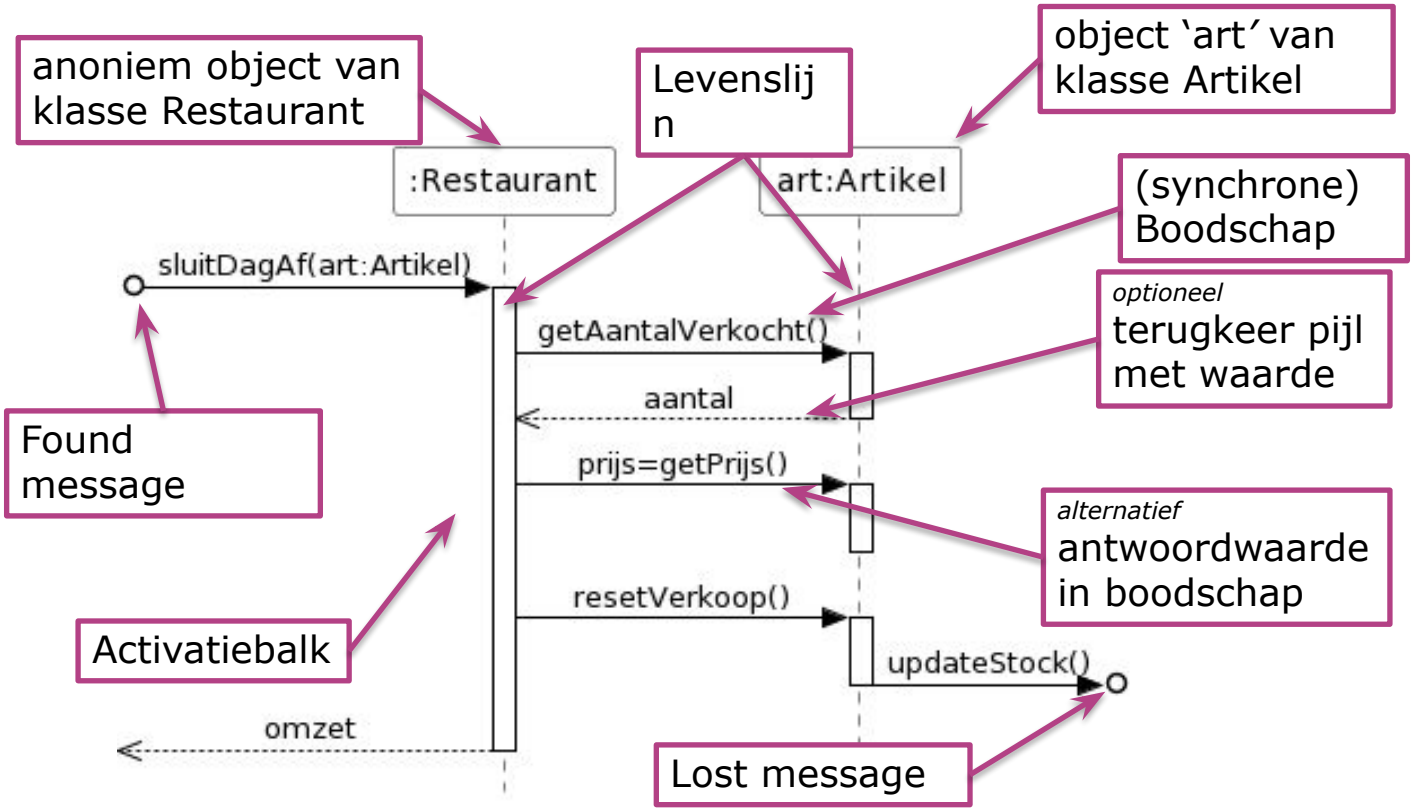
- **Dus, welke soorten postcondities moet je vermelden bij een OC:**

	Instantie	Link	Attribuutwaarde
Create	JA	JA	
Read			
Update			JA
Delete	JA	JA	

- **Merk op:** geen read/lezen van gegevens

SD (Sequence Diagram)

Sequence diagrams



Restaurant en Artikel zijn participanten of deelnemende objecten

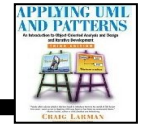
Werkwijze Sequence Diagram opstellen

1. Bepaal welke **operatie** het onderwerp is van het diagram
   - Vermeld deze operatie op de pijl van de found message links bovenaan het model
2. Bepaal wat de **returnwaarde van de operatie** in het diagram
   - Teken deze returnwaarde onderaan links het diagram
   - In het sequentiediagram moeten we alle interacties tekenen om deze returnwaarde te bekomen
3. Teken bovenaan het SD alle **objecten** die gebruikt worden om de returnwaarde te bekomen
4. **Bepaal** welke **operaties** in welke volgorde moeten opgeroepen worden
   - Je moet één ononderbroken lijn kunnen tekenen van found message naar de returnwaarde links onderaan het model
   - Operatie = activatiebalk op een levenslijn
   - Oproep van een operatie/event = pijl tussen levenslijnen
   - **Bepaal** per oproep van een operatie de **parameters** die moeten meegegeven worden

GRASP

4

GRASP



GRASP patronen in Larman hoofdstuk 17
Low Coupling (evaluerend basisprincipe)
High Cohesion (evaluerend basisprincipe)
Controller
Creator
Information Expert

Andere GRASP patronen

Polymorphism

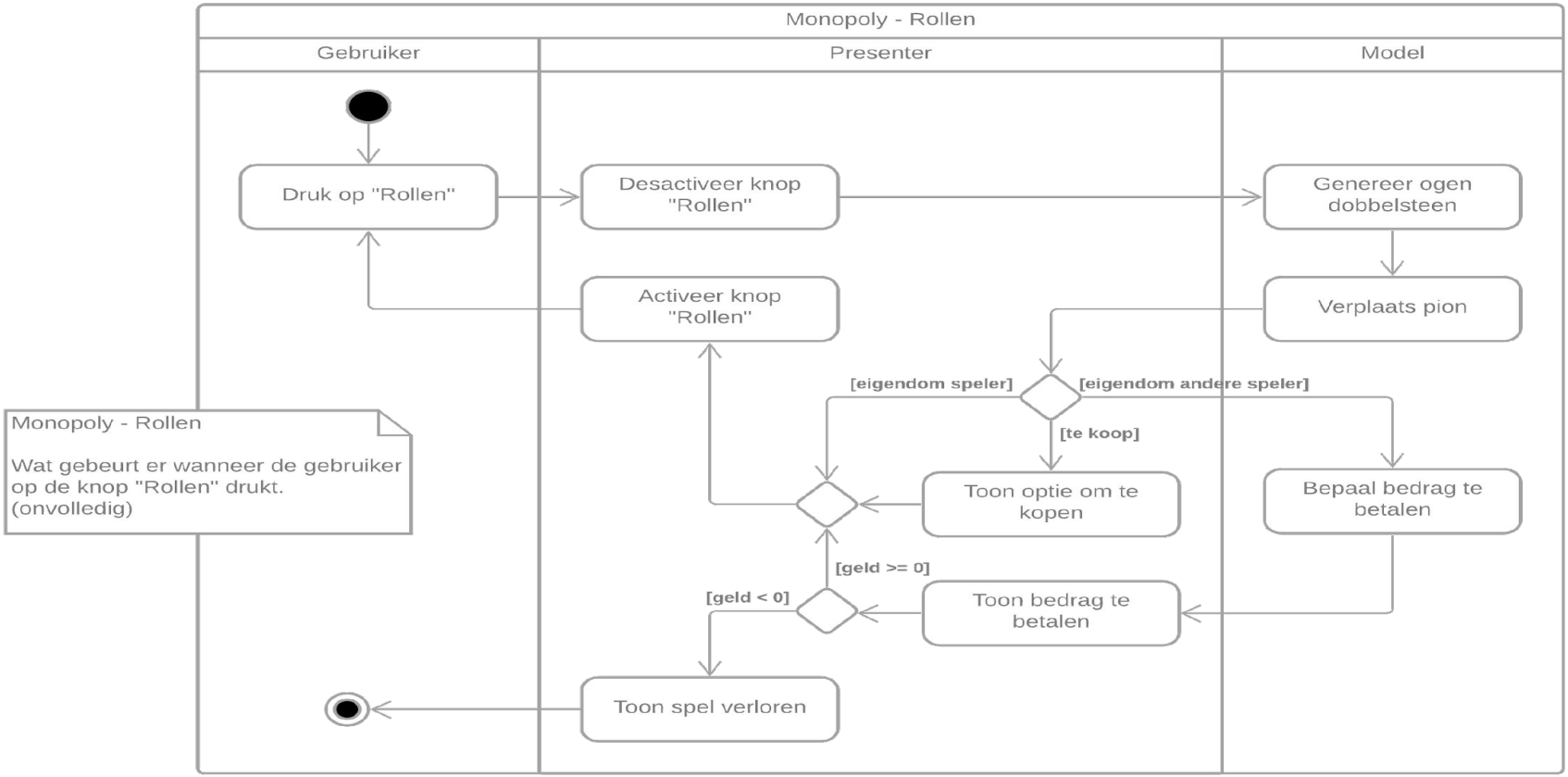
Pure fabrication

Indirection

Protected variations

Activity diagram

Voorbeeld

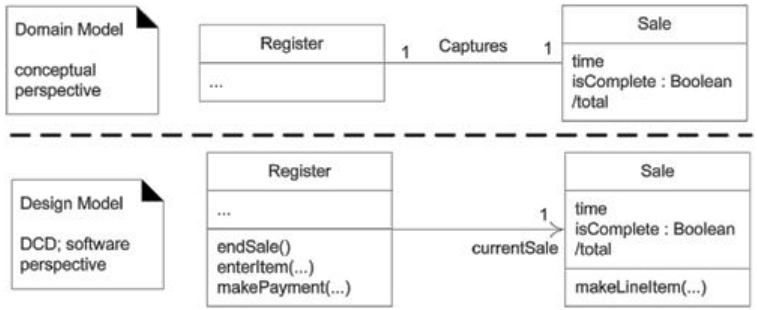

Monopoly - Rollen
Gebruiker
Presenter
Model
Druk op "Rollen"
Desactiveer knop "Rollen"
Genereer ogen dobbelsteen
Activeer knop "Rollen"
Verplaats pion
[eigendom speler]
[eigendom andere speler]
[te koop]
Toon optie om te kopen
Bepaal bedrag te betalen
[geld >= 0]
[geld < 0]
Toon bedrag te betalen
Toon spel verloren
Monopoly - Rollen
Wat gebeurt er wanneer de gebruiker op de knop "Rollen" drukt.
(onvolledig)

DCD (Ontwerpklassediagram)

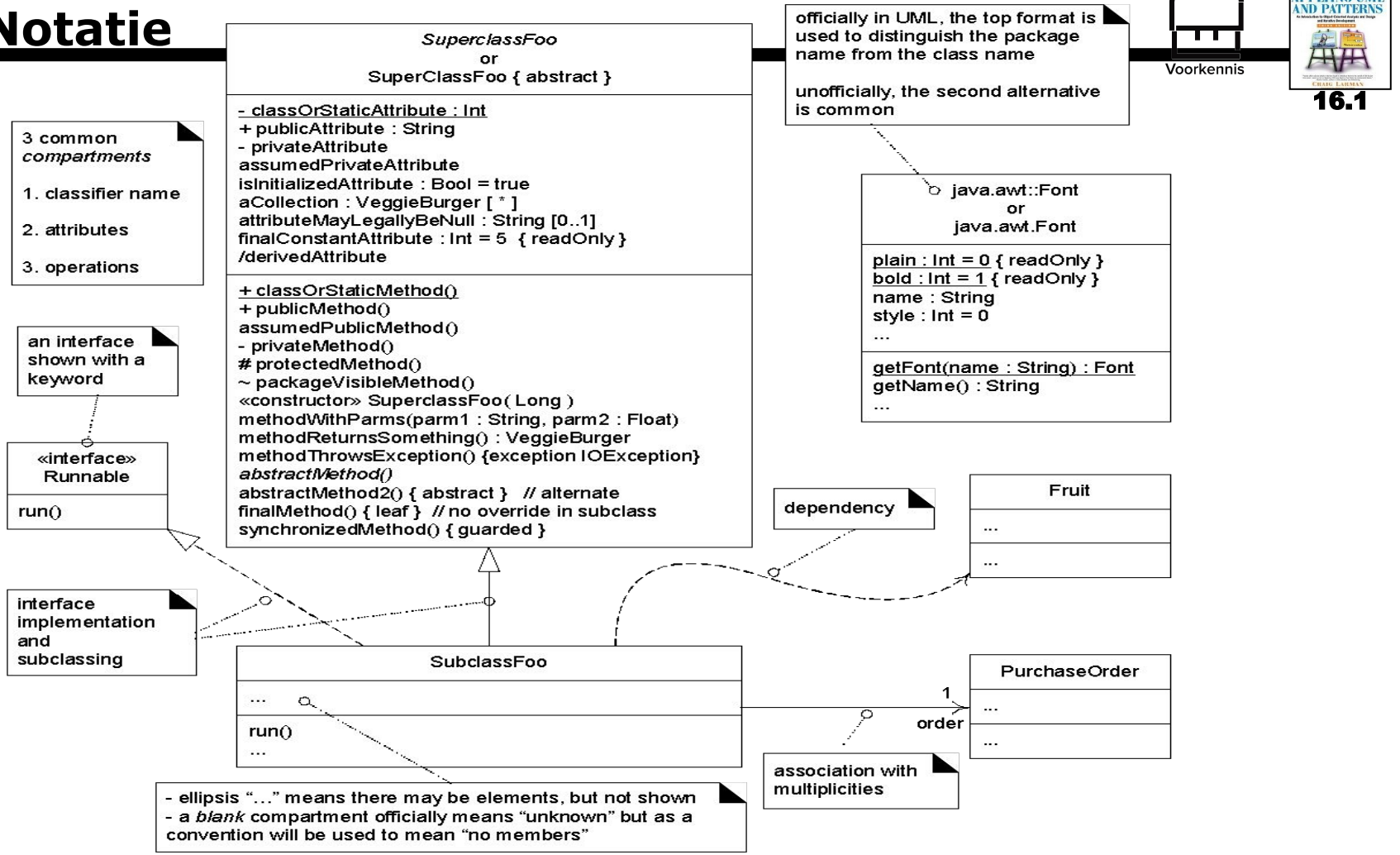
Domein model ⇔ Ontwerp Klassediagram (DCD)

Van domeinmodel naar ontwerp klassediagram door toevoeging/aanpassing van:

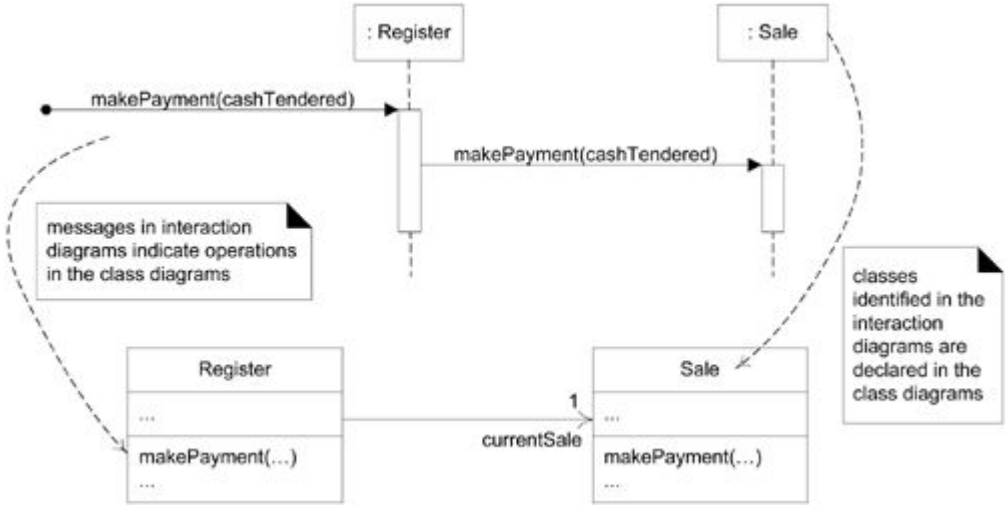
1. Attribuuttypes en attribuuttoegankelijkheid
2. Ontwerpassociaties
3. Operaties (met parameters en hun types, return types, toegankelijkheid)
4. Ontwerpklassen (zie GRASP)
5. Overige toevoegde informatie aan een DCD



Notatie


SuperclassFoo
or
SuperClassFoo { abstract }
officially in UML, the top format is used to distinguish the package name from the class name
unofficially, the second alternative is common
3 common compartments
1. classifier name
2. attributes
3. operations
- classOrStaticAttribute : Int
+ publicAttribute : String
- privateAttribute
assumedPrivateAttribute
isInitializedAttribute : Bool = true
aCollection : VeggieBurger [*]
attributeMayLegallyBeNull : String [0..1]
finalConstantAttribute : Int = 5 { readOnly }
/derivedAttribute
+ classOrStaticMethod()
+ publicMethod()
assumedPublicMethod()
- privateMethod()
protectedMethod()
~ packageVisibleMethod()
«constructor» SuperclassFoo(Long)
methodWithParms(parm1 : String, parm2 : Float)
methodReturnsSomething() : VeggieBurger
methodThrowsException() {exception IOException}
abstractMethod()
abstractMethod2() { abstract } // alternate
finalMethod() { leaf } // no override in subclass
synchronizedMethod() { guarded }
java.awt::Font
or
java.awt.Font
plain : Int = 0 { readOnly }
bold : Int = 1 { readOnly }
name : String
style : Int = 0
...
getFont(name : String) : Font
getName() : String
...
an interface shown with a keyword
«interface»
Runnable
run()
dependency
Fruit
...
...
interface implementation and subclassing
SubclassFoo
...
run()
...
1
order
PurchaseOrder
...
...
association with multiplicities
- ellipsis "..." means there may be elements, but not shown
- a blank compartment officially means "unknown" but as a convention will be used to mean "no members"

Van Sequence Diagram (SD) naar Ontwerp Klassediagram (DCD)



- Tijdens het maken van een dynamisch SD worden klassen en operaties bepaald. Attributen worden vermeld als in- en output.
- Typisch wordt er **eerst** een **SD** opgesteld, om **daaruit** een **DCD** op te stellen
  - Met meer wendbare ontwikkelingsmethoden (**agile**), gebeurt dit **tegelijk**

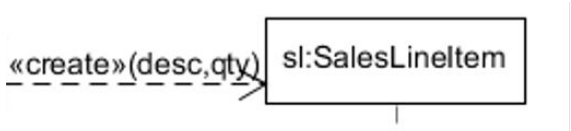
Werkwijze: objectzichtbaarheid in de praktijk

Hoe zichtbaarheid herkennen in een SD: stel je de vraag "Hoe kent een object 'a' de verwijzing naar object 'b'?"

Mogelijke manieren (in volgorde van eenvoudig te herkennen):

1. <<parameter>> Krijgt een object een verwijzing naar een ander object binnen als parameter bij een input message?

   Bijv.: "desc" is een verwijzing naar een object van de klasse ProductDescription



2. <<local>> Kent een object 'a' een verwijzing naar een ander object 'b' omdat 'b' opgezocht of aangemaakt wordt vanuit object 'a'?
3. <<global>> Het verwezen object is **permanent** globaal beschikbaar voor het hele systeem. Vermijd deze hoge zichtbaarheid en koppeling want nadelig!
4. <<attribute>> De verwijzing naar een object is **permanent** gekend als verwijzing die opgeslagen is in een attribuut van de verzender.

We kunnen attribuutzichtbaarheid dus niet visueel herkennen op een SD. Door de andere drie soorten uit te sluiten, kunnen we zeggen dat er een attribuut nodig is als verwijzing.

Opdracht 3 – Ontwerpklassediagram (DCD)

Werk een **ontwerp klassediagram uit** op basis van, en beperkt tot de inhoud van het uitgewerkte sequence diagram van de vorige slide.