Seminarium 1

Objekt-Orienterad Design, IV1350

Daniel Westerlund
daweste@kth.se

2019-05-31

Innehållsförteckning

1 Introduktion

Seminarium ett gick ut på att bekanta sig med ett UML modellerings hjälpmedel, öva på att skapa en domänmodell samt att öva på att skapa ett systemsekvens diagram. Detta skall lösas genom att skapa en domänmodell utav en detaljhandelsbutik i ett UML program, och därefter skapa ett systemsekvens diagram utav denna domänmodell.

Jag har arbetat själv.

2 Metod

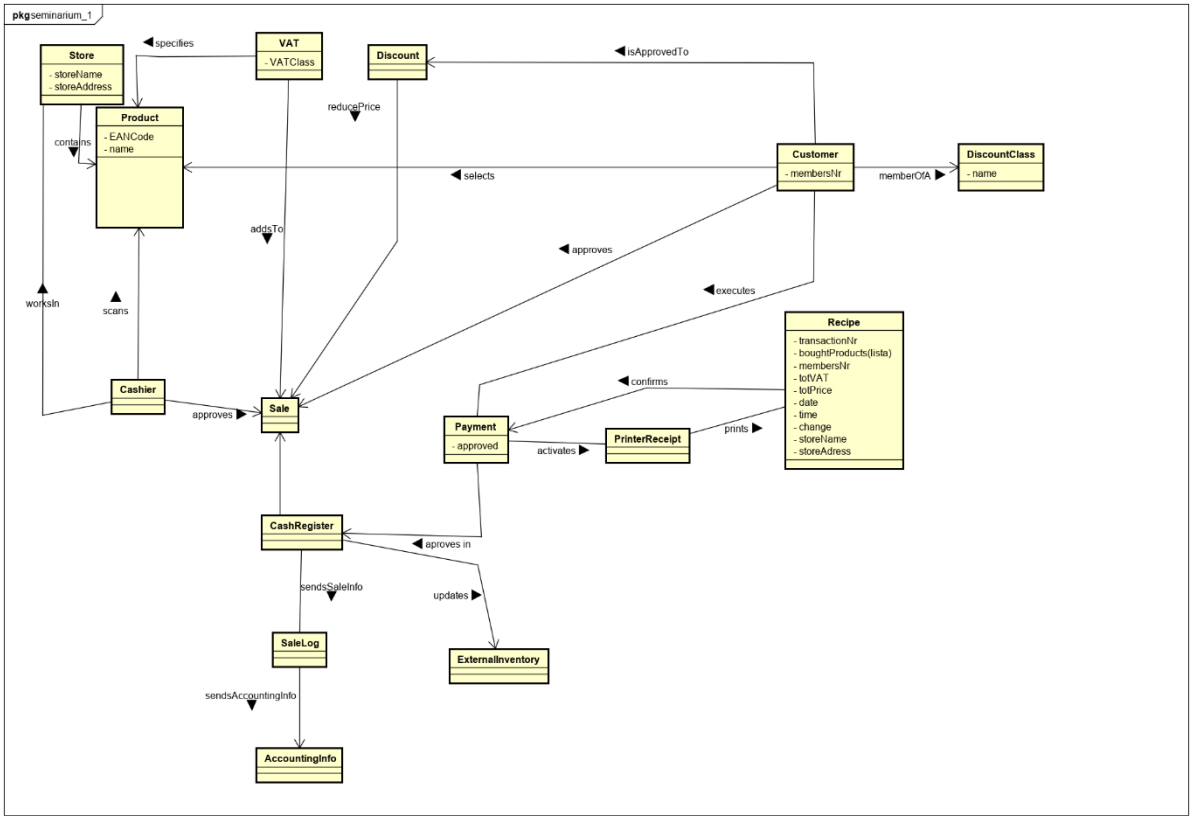
Först gicks kravspecifikationen för ”Process sale” igenom och substantiven noterades för att utgöra ett underlag till domänmodellens klasser. För att få ett ännu större underlag till domänmodellens klasser gick igenom hur ett köp går till ”i verkligheten”. Som UML modellerings program valdes Astah UML 8.1.0/3ac74f Model Version: 38. Alla substantiv skapades som klasser i Astah. Nu påbörjas processen att hitta lämpliga associationer mellan klasser samt att avgöra om någon klass bör gå in som ett attribut i en annan klass, eller rent av tas bort från domänmodellen. För att hitta lämpliga associationer mellan klassarna, hittar man klasser som hör ihop med varandra och försöker kombinera ihop dom i en mening med ett verb i mellan. Klasserna som gick ifrån att vara en egen klass till att bli ett attribut, valdes genom att det är klasser som beskriver egenskaper utav en annan klass. Under arbetat av att addera associationer uppstod det i några fall behov utav att addera en/flera attributer till vissa klasser. Detta resulterade i domänmodellen i kap 3.
Systemsekvens diagrammet skapades även det i samma Astah program. Genom att ange det olika system som personerna kommer att kommunicera med, de externa system som finns i domänmodellen. Samt att skapa ett flöde av interaktionerna mellan personerna och systemen och i vilken ordning det kommer att ske.

3 Resultat

Underlaget för klasser som identifierades ifrån kravspecifikationen var:

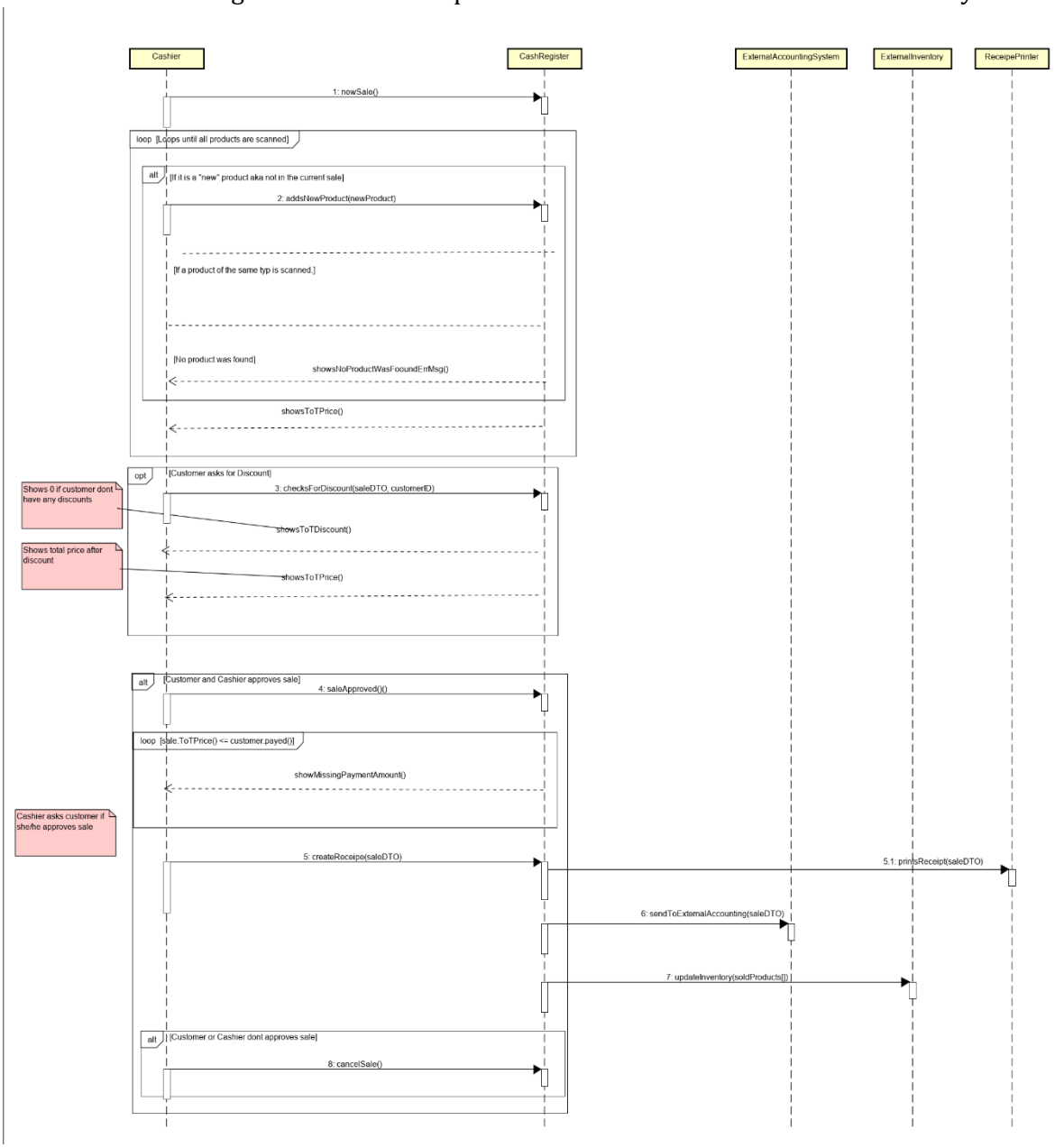
- Affär (Store)
- Kassa (CashRegister)
- Säljare (Cashier)
- Kund (Customer)
- Vara (Product)
- Kundkord/Kungdvagn(Specifies)
- Varuband (PartBands)
- Transaktion (Transaction)
- Lager (Inventory)
- Kvitto (Receipt)
- Kvittoskrivare (PrinterReceipt)

Figur 1 visar resultatet efter att vissa klasser har blivit till attributer samt att det under arbetets gång har upptäckts behov av flera attributeter. Samt att associationer mellan klasserna har lags till, en del klasser togs bort eftersom dom inte ansågs vara relevanta för programmet.



Figur 1 - Domänmollen

Systemsekvensdiagrammet som kan ses i figur 2, visar säljarens interaktion med systemet samt vilka anrop systemet gör med externa system som t.ex. ExternalAccounting. Steg 1, är att säljaren initierar en ny försäljning på systemet. Steg 2, försöker säljaren att lägga till en produkt med en EAN-kod till försäljningen och där finns det tre olika scenarion. Antingen finns det inte i den pågående försäljningen och därför läggs produkt med den EAN-koden till i försäljningen, finns det en produkt med den EAN-koden så ökas antalet av den produkten med ett. Sista scenariot är fall säljaren försöker lägga till en vara med en felaktigt EAN-kod. Därefter visas det total priset, och detta sker i en loop tills säljaren har lagt in alla produkter som kunden vill köpa. I steg tre visas eventuell rabatt och det totala priset uppdateras. I steg 4 kommer säljaren och kunden överens om köpet skall godkännas, detta sker muntligen, godkänns det ej, så avbryts köpet. Vid godkänt så påbörjas en loop av betalningen som visar hur mycket mera som skall betalas tills betalningen är klar. I steg 5 skapas ett kvitto som även skickas till kvittoskrivaren, skickar även iväg saleDTO till ExternalAccouting samt en lista med produkter som har sålts till ExternalInventory.



Figur 2 - Systemsekvensdiagram

4 Diskussion

Hade en det problem med att det vart en ”spindel i nätet” utav Sale-klassen, eftersom det är en sådan central del utav ett köp i en dagligvaruhandel-butik, men genom att skapa en CashRegister-klass fick jag bort en del interaktion med Sale-klassen, som det Payment och de externa systemen. Men Sale-klassen tillsammans med Product-klassen är fortfarande på gränsen till att vara en ”spindel i nätet”, men har ingen bra lösningen för att minska på det, då båda två är centrala delar i sammanhanget. Valen mellan att lägga en klass som ett attribut istället för en egen klass upplever jag ibland är svåra att avgöra.
En del uppdateringar ifrån semianrium 1, är bland annat att jag bytte ifrån svenska-klassnamn till engelska-klassnamn, känns egentligen mera naturligt och det är mera norm i sammanhanget. Har även skapat en Discount-klass som jag tänker har hand om allt som har med rabatter att göra och kommer troligtvis vara bättre än att endast ha ett attribut ”rabatt” på klassen produkt, som det tidigare var. Detta gör programmet mera skalbart och lättare att ändra/lägga till rabatter. Har även tagit bort en del funktioner som fanns med men ej var på kravspecifikationen.
Systemsekvens diagrammet var den del jag hade svårast för och är inte helt säker på fall jag har uppfattat exakt vad det ska symbolisera. Min uppfattning är att det ska visa hur interaktionen mellan personer och system, och interaktionen mellan systemen, men jag upplever att mitt systemsekvens diagram vart väldigt litet, men ser inte riktigt hur jag ska kunna utöka det. Inser nu att punkt fem eventuellt borde göras om och heta någonting i stilen med ”paymentDone” och sedan skapa kvitto, och aktivera alla externa system. Känns egentligen som en rimligare ”knapp” för säljare att trycka på.