Seminar 1

Object-Oriented Design, IV1350

Maher Jabbar maherj@kth.se

2016-04-07

Contents

1 Introduktion

Uppgiften för denna labb var att lära känna själva UML språket och träna och lära sig att skapa Class diagram som ger en statisk bild av något och Domain model och System Sequence Diagram som visar hur olika objekt skickar meddelanden till varandra. En viktig uppgift var också att visa hur vi kom fram till vår domain model , de metoder som vi använt och vår resonemang.

2 Method

För detta lab använde jag mig av programmet Astah. Syftet med Domain model är att ge en logisk bild av det område i verkligheten som program ska hantera, I vårt fall är en bild av Bilprovning. Det finns olika sätt att hitta domän modeller klasser, substantiv identifiering och kategori lista. I detta lab har jag använt mig av substantiv identifiering för att hitta möjliga klasser för min domain model.

Programmet Specifikation med Substantiv Identifiering:

1. **Inspector** specifies that a new **inspection** may be started.
2. **Program displays** next customer's queue number on display outside **garage** and **opens** garage door.
3. **Customer** enters **garage**.
4. **Inspector** instructs **program** to close **garage door.**
5. **Program** closes **garage door.**
6. **Inspector** enters vehicle's **license number.**
7. **Program** retrieves appropriate **inspections** for **vehicle**.
8. **Program** tells **cost** for **inspection**.
9. **Customer** enters **credit card information**.
10. **Program** sends **payment authorisation** request to an external payment authorisation system, and requests **payment approval**.
11. **Program** prints **receipt**.
12. **Customer** receives **receipt.**
13. **Inspector** specifies that **payment** is completed.
14. **Program** tells **inspector** what to **inspect** on **customer**'s **vehicle.**
15. **Inspector** performs the specified **inspection**.
16. **Inspector** enters **result** of the specified **inspection**. The **result** is either **pass** or **fail**.
17. **Inspector** repeats steps 14-16 until **program** indicates that all **inspections** are completed.

18. **Program stores** inspection **results**. These **results** include information about failed **inspections** that must be repeated when the **vehicle** is fixed.

19. **Program prints inspection results.**

20. **Inspector** informs **customer** about **results** and hands over the **printout**.

21. **Customer** leaves **garage**.

Total Substantividentifikation :

Inspector, Inspection, Customer, Garage, Display, Vehicle, License number, Cost, Credit Card, credit card information, payment, payment authorisation, external payment authorisation, print, receipt, Pass,fail, result, open garage, close garage, Store Result,door. inspection Results,information, approval, enter, perform

Kandidater för klass:

Inspector, VehicleInspection, Customer, Garage, Display, VehicleLicense number, Cost, Credit Card, payment, external payment authorisation, Receipt,Inspection Result,Printer,Register

Klasser som passar bättre som attribut:

Pass, Open, Information, credit card information,check.

Klasser som passar bättre som associations:

print out, receive, store result, approval, open, enter, perform,Next number.

System sequence diagram

Kort förklaring för min SSD

Först bestämde jag klasser för SSD mellan system och inspector, Kunden får inte vara med i SSD för att han har inte något och göra med systemet. alltså det finns inget att konstatera mellan system och kunden. I punkt 1 i Basic Flow så starter inspektorer besiktning för bilen då kan vi kalla det startNewInspection. I punkt två så sker det i systemet då programmet visar nästa kund kö nummer på skärmen och så öppnar dörren. I punkt 4 så instruerar inspektor program för stänga dörren. I punkt 6-8 så kan vi göra en loop mellan inspector och system. Punkt 9-11 samtidigt med alternative pay with cash alltså det uppstår if-else. Om betalning sker så starter inspektorer besiktning för bilen. Punkt 17-18 Bilen besiktas flera gångar tills inspection är klar och sendan informerar inspektorer om vila fel som finns på bilden, Jag gjorde loop som det står på min SSD(inspectionRemain).

3 Result

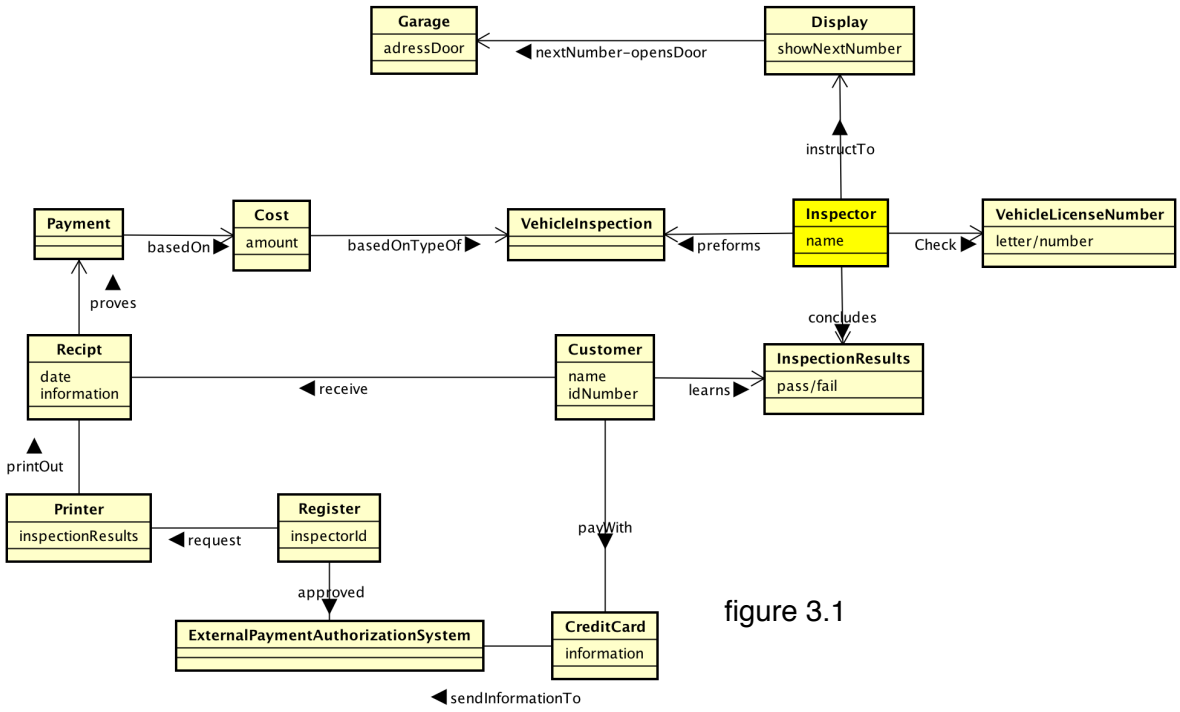

figure 3.1

Kort Kort förklaring for min Domain Model

Inspector Instruct Display for next number and open the garage

Inspector check Vehicle License Number

inspector preforms the vehicle inspection

Inspector concludes the result (pass or fail) and inform the customer

customer learn the result and pay with card .

Register check if the payment is approved

Register send request to print out the receipt

Customer receive the receipt

System sequence diagram

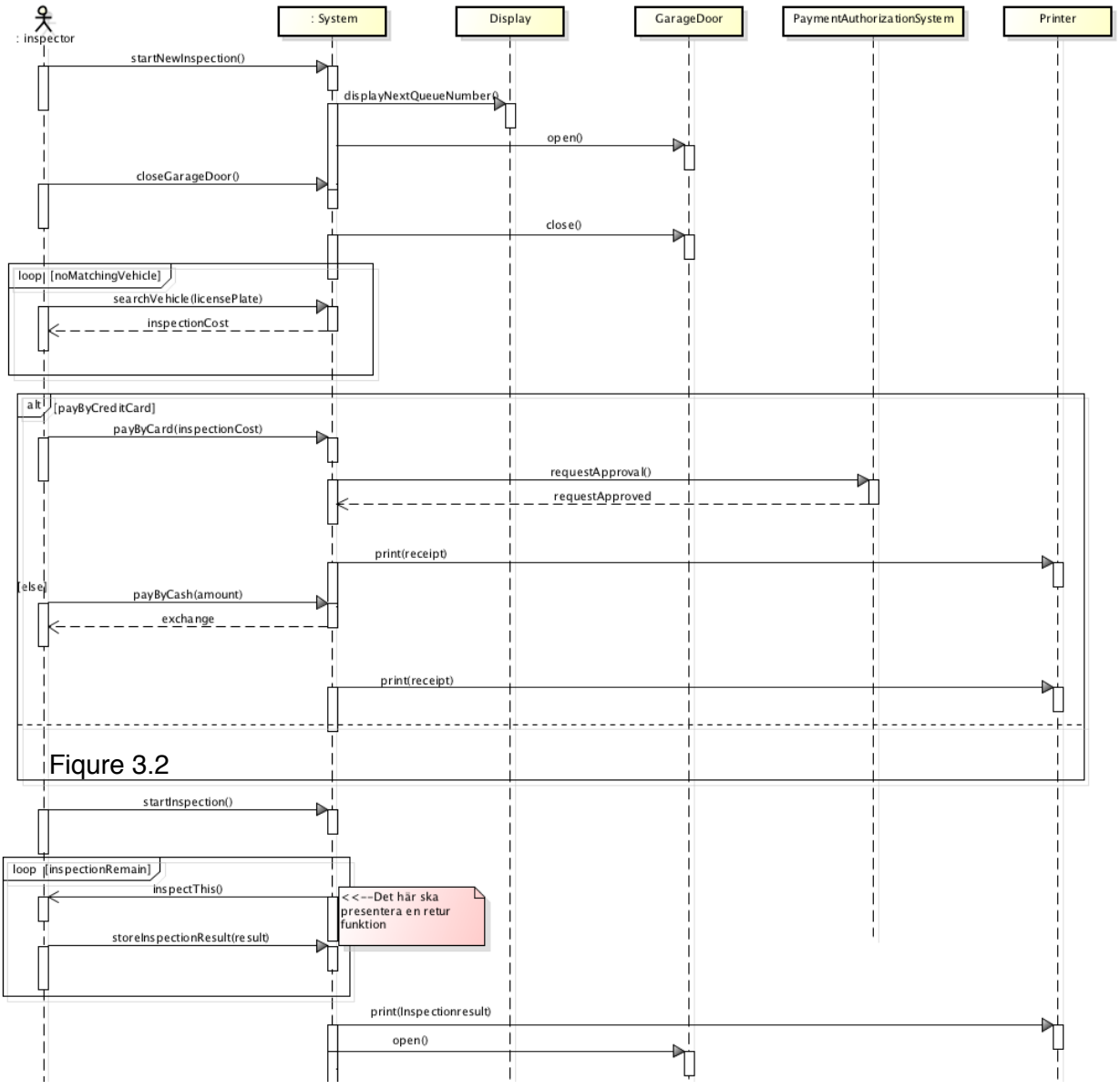

Figure 3.2

4 Discussion

I modellen kan finna en enstaka klasser som lagt i ”tidsordning” och det är något man vill undvika för uppnå det riktiga syftet vilket är att spegla verkligheten. Ibland är det svårt att uppfatta hur klasserna ska placeras ut för att undvika det. Min domain liknar ”naiv DM” men skiljer mycket för att vara det. Jag anser att det inte finns “spider-in-the-web” i min domain modell för att undvika att klasserna blir för centrerande. En Dm med “spider-in-the-web” kan fortfarande vara värdefull men skulle förmodligen vara högre värde av associationer som är mer jämnt fördelade. Jag anser att beskrivning är tillräckligt förståelig för att läsaren ska kunna bilda en uppfattning om hur domain modell fungerar samt att läsaren förstår hur besiktningen sker ur ett verkligt perspektiv. Den första, och viktigaste steget när jag skapar en domänmodell , är att hitta så många klassar som möjligt. Jag använde mig av metoden substantiv identifiering för att hitta klasser. Det är mycket vanligare att ha för lite klasser än att ha för många. Det är också långt mer problematiskt att ha för några klasser, eftersom det är mycket lättare att avbryta existerande klasserna än till hitta nya. Vissa klasser bör inte förbli klasser, men i stället förvandlas till attribut. det är mycket användbart. I figure 3.1 under klassen “Customer” så har jag “name och id number alltså person nummer” och det skulle räcka med det i verkligheten för att få den nödvändiga informationen om kunden. Syftet med Associationer i DM är bara att klargöra, utan namn till associationer blir det svårt att klargöra alls. Vi kan se i min SSD att Klassen Customer är inte med. eftersom Kunden inte har en interaktion med system alltså det finns inget att konstatera mellan system och kunden. Det är Inspector Som har direkt kontakt med System, Exempelvis. Inspector säger till systemet att ge inpsectiontresult och skriva ut kvitto. Argument och retur värden medföljer i SSD.