Opponentrapport på examensarbete ”Utveckling av ett affärssystem med Unified Process” av Therese Sundström.

Författare Per Johansson, Henrik Wallinder

Generellt

Helhetsintrycket från genomläsning av uppsatsen är bra. Uppsatsen är lätt att läsa och har en bra struktur som är lätt att följa. Det är en bra uppgift för 20 veckor och det är bra med ett nyutvecklingsprojekt. Projektet innefattar allt från förstudie, krav, design, implementation, test, leverans och även dokumentation av hela arbetet, vilket gör det till ett komplett och bra projekt.

Upplägget på uppsatsen är bra, med introduktion av UP samtidigt som genomförande och resultat av projektet presenteras. Uppsatsen borde fungera bra som utbildningsmaterial för nya personer som vill sätta sig in i UP, den kan ses som en slags grundlig tutorial.

Titeln på uppsatsen avspeglar innehållet, eventuellt borde ”affärssystem” bytas ut mot ”ordersystem”. Affärssystem innefattar ofta mer funktionalitet för ekonomistyrning som löner, rapportering etc.

Det finns en bra röd tråd i uppsatsen. Projektet har två mål, huvudmålet är att utveckla ett system, och i andra hand att utvärdera UP, som en fallstudie på ett konkret projekt. Detta beskrivs i inledningen och beskrivs konsekvent genom hela uppsatsen samt sammanfattas på slutet.

Projektet verkar vara väl genomfört i bra samarbete med slutanvändarna. Ett fungerande system har levererats. Uppsatsen är intressant att läsa och innehåller mycket intressant information.

Layout

- Styckeindelningen är inte konsekvent. Antingen indrag, eller ny rad
- ”Ifrån” är talspråk, skriv ”från”
- Referenser: ”enligt [1]” bör skrivas ”enligt John Smith [1]”, ”[1] säger” bör skrivas ”John Smith [1] säger”
- Numrerade listor eller punktlistor bör inte innehålla ”.” (punkt)
- Förslag: Det vore snyggt och praktiskt med sidhuvud som beskriver vilket huvudkapitel, t.ex. ”Bakgrund”, man befinner sig i
- ”Sektion” bör bytas ut mot avsnitt alternativt underkapitel
- Fler referenser bör in, referenserna bör även anges direkt när ett nytt begrepp introduceras

Sammanfattning

Saknas resultat från utvärdering av UP.

Abstract

Saknas resultat från utvärdering av UP.

1 Inledning

Saknas resultat från utvärdering av UP.
Uppsats: ”Denna uppsats försöker att ge en inblick i området Software Engineering kan användas i praktiken”
Förslag: ”Denna uppsats ger en inblick i hur Unified Process kan användas i praktiken”
(SE känns för oprecist i detta sammanhang, UP stämmer mer överens med projektet.)

2 Bakgrund

2.1 Software Engineering

Ett flertal referenser bör in i texten, främst i 2.1.1 Historia.
En annan kommentar till historik är att man kanske kan poängtera att arbeta i projekt var en arbetsform som började användas generellt även utanför SE. SE var inte anledningen till att arbete i projektform, utan snarare ett område där det applicerades. En av anledningarna till att man började arbeta i projekt var t.ex. kapptävlingen mellan USA och forna Ryssland.

Definitionerna under 2.1.2 Översikt av SE är mycket bra att ha med men bör utgöra ett eget kapitel. Ta gärna med fler definitioner, t.ex. SE. Referenser måste med till definitioner. Det bör framgå att en Modell även kan var en abstrakt beskrivning av ett system. Det framgår inte heller vad skillnaden är mellan metod och process.

2.2 Översikt av utvecklingsprocessen

Under 2.2.2 Design och implementation så står det att implementationsfasen ligger ganska sent i utvecklingsprocessen, detta stämmer för vattenfallsmodellen men inte för t.ex. XP.

Under 2.2.3 Testning är ett förslag att även ta med begreppet Integrationstestning, antingen som en synonym till Delsystemstesting eller som ett nytt steg mellan Delsystemtestning och Systemtestning. Enhetstestning och Modultestning är ofta synonyma begrepp.

2.3 Modeller och processer inom Software Engineering

Figur 2.4 Vattenfallsmodellen ett förslag är att göra om figuren till ett ”rent vattenfall” utan uppåtgående pilar (”vatten rinner inte uppåt”).

Under 2.3.3 Spiralmodellen: Vad skiljer Spiralmodellen och iterativt arbetssätt som t.ex. RUP?

2.4 Jackson System Development

Är detta kapitel verkligen relevant för uppsatsen? Förslag är att antingen ta bort detta kapitel helt eller att utveckla resonemanget och jämföra JSD med UP och XP.

2.5 Extreme Programming

2.6 Unified Process

Det känns mycket viktigt att beskriva UML i samband med UP eftersom UML är en del av UP och används för all modellering.

Vad skiljer mellan RUP och UP?

Under 2.6.1. Strukturen i UP: Översiktlig beskrivning över ”Core Workflows” vore bra. Det saknas dessutom några i figur 2.8. Oftast ses ”Etableringsfasen” som en fördjupad förstudie, detta kanske kan vara bra att tillägga.

2.7 Sammanfattning

3 Kravinsamling & Användningsfallsmodellen

3.1 Kravinsamling

3.2 Kravinsamling av funktionella krav

Borde inte Produkt hanteras som övriga entiteter eftersom kravet var att produkterna skulle kunna hanteras på liknande sätt som Kund, Offert, Order och Faktura. Möjligheten att separat kunna skapa och redigera produkter är värdefullt i sig, t.ex. för att ändra produkter och generera produktlistor. Alternativt bör det finnas en motivering till varför produkthanteringen skiljer sig ifrån hanteringen av övriga entiteter.

3.3 Kravinsamling av ickefunktionella krav

3.4 Användningsfallsmodellen

3.4.2 Användningsfall: det står att varje interaktion som en aktör utför i samarbete med systemet är ett användningsfall, men är verkligen detta riktigt? Ett användningsfall är vanligen ganska stort och kan ofta innehålla flera interaktioner.

3.4.3 Användningsfallsdiagram: Aktören ”Användare” kanske kan göras tydligare, finns det någon tydlig roll? Kanske flera?

3.4.4 Specifikation av användningsfall: ”Lägg till ny produkt” bör ske i enlighet med de övriga entiteterna med liknande krav?

Visa Kundinformation/Visa Order/Visa Offert/Visa Faktura har beskrivningar inom parantes, dessa bör tas bort eftersom de inte tillför något till användningsfallet. (Användarnyttan med användningsfallet uppnås, vilket är det som är relevant.)

4 Kravanalys och Analysmodellen

4.1 Kravanalys

4.2 Analysklasser

Vår uppfattning är att analysklasser endast i möjligaste mån bör innefatta klasser i problemdomänen och inte designrelaterade klasser. Entitetsklasserna Kund, Offert, Produkt, Order och Faktura måste vara med här. Klassen Controller kanske inte bör vara med, det känns som om analysen i så fall tar med designaspekter (MVC?).

Den kanske viktigaste diagramtypen både i analys är klassdiagram. Det vore mycket nyttigt med ett konceptuellt klassdiagram med entitetsklasserna: Kund, Offert, Produkt, Order och Faktura likt ER-diagrammet i figur 5.4. Associationer bör namnges i detta diagram och även roller för klasserna som ingår. Med lite beskrivande text så sammanfattar detta diagram översiktligt hela systemet.

4.3 Realisering av användingsfall

Samarbetsdiagrammen innehåller designklasser vilka inte bör vara med i analysen, som: MainController, DBController, CustomerWindow, PopUpFrame, OfferWindow, PopUpProduct. Analysen vet inget om dessa klasser.

Numreringen i många nivåer gör att det inte är så lätt att följa sekvenserna t.ex. ”2.1.2.1.1”

4.3.3. Händelsehanteraren heter tidigare Controllerklassen, förslag: använd samma benämning.

5 Designmodellen och Databasen

Klassdiagram saknas i detta kapitel. De klassdiagram som finns i kapitel 6 Implentationsmodellen & Implementation bör flyttas hit.

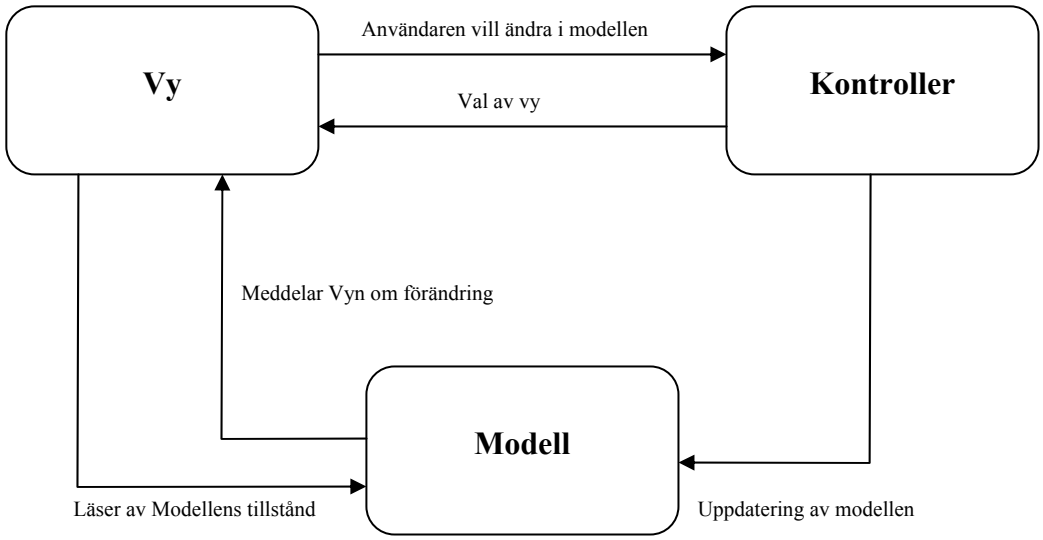
5.1 Design

5.2 Designsystemet

5.2.1 Subsystem: Vad betyder ”Rekursiva Subsystem”

5.3 Övergripande design

5.3.1 Model-View-Controller: Figur 5.3 bör kompletteras enligt figur nedan:



Vi saknar beroenden från kontroller till vy, samt från vy till modell. Det kanske även är bra att beskriva beroendena. Vissa av klasserna är listade i figuren, samtliga klasser bör vara med, alternativt presenteras efteråt, så det framgår tydligt vilka klasser som ingår i Modell, respektive Kontroller, respektive Vy.

5.4 Sekvensdiagram

Ett förslag är flytta samarbetsdiagrammen i bilaga i till detta kapitel, man får då kompletterande bilder att titta på när man läser texten i enlighet med kapitel 4.

5.5 MS Excel-mallar

5.6 Databasen

5.6.1 Entiteter och Attribut: Det vore bra om status för Offert, Order samt Faktura beskrivs, eventuellt med tillståndsdiagram. Bör beskrivas i samband med klassdiagram för entitetsklasserna.

5.6.3 Mappning från E-R-modellen till Relationsdatamodellen: Varför max 900 kunder? 2000 offerter? 2000 order? 5000 fakturor?

6 Implementationsmodellen & Implementation

Det vore bra att ta med deploymentdiagram i enlighet med UP/UML.

6.1 Implementationsmodellen

6.2 Implementationssystemet

6.3 Implementation av grafiskt användargränssnitt

6.3.1 Vyn: PopUpLogin – Login verkar vara ett krav som inte finns beskrivet tidigare? Är detta enbart för inloggning till databasen, i så fall kanske detta kan lösas i Connect satsen i koden.

6.4 Implementation av modell och kontroll

6.4.1 Kontroll: Behövs verkligen temporärlagring för Customer, Offer, Order, Invoice och Product. Varför behövs OfferProduct och OrderProduct? Räcker det ej med Product?

6.4.2 Modell: Entitetsklasserna – Customer, Offer, Order, Invoice, Product bör ingå i modellpaketet (samtliga klassdiagram i kapitel 6 bör flyttas till kapitel 5, enligt tidigare kommentar).

6.5 MS Excel-mall och frågor

7 Testmodellen & Överlämning

7.1 Inledning

7.2 Testsystemet

7.3 Specifikation av testfall

Ny_kund_existerar ”Databasen innehåller inte kunden med kundinformation bestående av” bör vara ”Databasen innehåller kunden med kundinformation bestående av”.

7.4 Överlämning

Det vore bra om ”Användarmanual: System” och ”Användarmanual: Excel/MS Query” fanns med som bilagor.

8 Resultat & Rekommendationer

I första stycket beskrivs uppgift samt resultat för affärssystemet som skapades. Det bör även stå om resultat om utvärderingen från UP. Detta stycke kan även användas som Slutsats, se även 8.3 Slutsats.

8.1 Resultat av projektet med UP

Det beskrivs att projektet har innefattat åtta iterationer. Detta verkar mycket om UP har följts. En iteration enligt UP är ett mini-projekt som skall inkludera samtliga ”Work flows” från kravhanteringen till leverans och resulterar i en ny utgåva av det körbara systemet. Normal tid för en iteration är tre till nio månader. Varje iteration beskrivs dessutom i en separat projektplan/iterationsplan, i detta fall hade det blivit åtta stycken. Ditt projekt borde alltså ha varit en iteration och inte åtta?

Problemet med ”att få tag på produkter” i databasen för offerter och order hade kanske kunna lösas med separat hantering av produkter enligt tidigare kommentarer. I detta fall hade användarna istället fått valt produkter från en lista istället för att en ny produkt skapas vid varje tillfälle. (Eller har vi missuppfattat användningssättet?)

8.2 Rekommendationer

8.3 Slutsats

Känns som en fortsättning på tidigare diskussion i kapitel 8.1/8.2. En mer koncis slutsats vore bra, t.ex. kan första stycket under kapitel 8 användas.

Referenser
Alla referenser finns med. Förlag och år saknas på vissa av referenserna.

Bilaga A: Diagram designmodellen
Bör flytta till huvudtexten enligt tidigare kommentar.

Dessa fyra sekvensdiagram är inte tillräckligt detaljerade. Det saknas flera interaktioner/meddelanden mellan objekten. Det är viktigt att man enkelt kan växla mellan sekvensdiagram och kod på denna nivå (fysiska diagram måste stämma med koden).

Notation för objekt/instanser stämmer ej, dessa skall vara understrukna samt ha ett ”:” (kolon) före klassnamnet. Model_Interface är ej en instans, bör bytas mot Ponderator.

Bilaga B: Systemets Källkod
Det vore bra med innehållsförteckning och en tydlig avdelning av källkodsfiler. Ett tips är att namnge variabler med utskrivna namn och inte två bokstäver, det blir då lättare att följa koden.

Det verkar tyvärr som om en del källkod har försvunnit vid sidbytet vid utskriften.