HOVEDPROSJEKT:

TITTEL

ByggPOS

Salgssystem for byggevarehandelen

Point Of Sale

(English title)

FORFATTERE:

Kjetil Ophus
Lars Otto Nymoen
May-Britt Hansen
Oddbjørn Brede Sjulstad
Vivian Atterås

Dato:
22.05.02

SAMMENDRAG AV HOVEDPROSJEKT

Tittel:	Bygg POS			Nr. :
	Engelsk tittel: Point Of Sale			Dato : 22.05.02
Deltaker(e):	May Britt Hansen	99HINDP		
	Vivian Atterås	99HINDS		
	Lars–Otto Nymoen	99HINDS		
	Kjetil Ophus	99HINDS	Oddbjørn Brede Sjulstad	99HINDD
Veileder:	Harald Liodden			
Oppdragsgiver:	Vidar Hovdet			
Kontaktperson:				
Stikkord (4 stk)	Butikksystem, Database, Delphi, Flerbrukersystem			
Antall sider: 69	Antall bilag: 13	Tilgjengelighet (åpen/konfidensiell): Åpen		

Kort beskrivelse av hovedprosjektet:

Målet var å utvikle et enkelt og brukervennlig, men allikevel fullverdig, stabilt og pålitelig flerbruker salgsstøttesystem for byggevarehandelen. Systemet har kun den funksjonaliteten som faktisk er nødvendig for brukerne, og med et standard Windows brukergrensesnitt som gjør det enkelt for brukerne å benytte seg av systemet.
Det er laget rabatt- og avtaleregister i systemet, hvor de respektive kundenes rabattsatser og eventuelle særavtaler finnes
Det kan også registreres tilbud til kunder, disse kan senere effektiviseres som salg.
Databasen er bygget opp med NOBB(Norsk Byggevarebase) som retningslinje for varestrukturen. NOBB er en database med produktinformasjon for bygg- og anleggsbransjen.

FORORD

ByggPOS er utviklet av fem studenter ved Høgskolen i Gjøvik gjennom et hovedprosjekt våren 2002. ByggPOS er et salgsstøttesystem for byggevarehandelen.
Oppdragsgiver for prosjektet er Vidar Hovdet.

Formålet med rapporten er å beskrive gjennomføringen av prosjektet, samt å gi informasjon til de som skal fullføre utviklingen av systemet.
Rapporten med vedlegg inneholder en grundig beskrivelse av prosessen vi har gjort, med forklaringer, eksempler og annen dokumentasjon.

Vi vil benytte anledningen til å takke vår oppdragsgiver for hjelp til å definere kravene til systemet, og for det utstyret han har stilt til disposisjon.

Mocon As, avdeling Moelv, skal ha mange takk for at de har hjulpet oss med informasjon omkring byggevarebasen NOBB.

Vår veileder Harald Liodden, takkes herved for assistanse når problemene under programmerningen tilsynelatende ikke lot seg bekjempe.

Tom Røise ved HiG fortjener også en stor takk for sin velvilje til å drøfte synspunkter. Du har vært til stor hjelp, Tom!

Gjøvik 22.05.02

Lars Otto Nymoen _____

Oddbjørn Brede Sjulstad _____

May-Britt Hansen _____

Vivian Atterås _____

Kjetil Ophus (prosjektleder)

_____

Figurliste

1. Innledning

1.1 organisering av rapporten

Kapittel 1 Innledning
Innledende kapittel. Litt generelt om prosjektoppgaven, og om strukturen på rapporten.

Kapittel 2 Kravspesifisering
Dette kapitlet er fullt og helt viet kravspesifikasjonene til systemet ByggPOS. Det starter med litt grunnleggende informasjon om bakgrunnen for prosjektideen, og litt overordnet om kravene til systemet. Den første delen er skrevet slik at den i høyeste grad er lesbar for personer uten store IT-kunnskaper.

Videre utover i kapitlet finner du oversikt over alle definerte usecase for systemet. For oversiktens skyld har vi valgt å ta de fullstendige usecasebeskrivelsene ut som vedlegg, men du finner ett eksempel på en fullstendig beskrivelse og ett systemsekvensdiagram i rapporten.

For komplett oversikt over beskrivelser og modeller, se *VEDLEGG G*.
Dette kapitlet, samt vedleggene, vil være spesielt nyttig for de som skal stå for videreutviklingen av dette systemet.

Kapittel 3 Design
Omhandler designdelen av prosjektet. Her vil du finne eksempler på designdokumenter, litt om systemarkitekturen, samt en oversikt over databasen. Kapitlet sier også litt om arbeidsmetodikken vi har benyttet oss av for å gjennomføre analyse- og designfasene i prosjektet.

Vi har gjort bevisste valg i forhold til hva databasen og hva applikasjonen skal ha ansvar for, dette er beskrevet her. Til slutt finner du litt om hvilke alternative løsninger som er vurdert for systemet.
Fullstendig designdokumentasjon finnes i *VEDLEGG H* og *VEDLEGG I*.

Kapittel 4 Implementering
Dette kapitlet forteller om vårt valg av utviklingsverktøy for prosjektet, og om valg av databaseplattform for systemet. Prinsipper og standard for kode finnes også her.
Du vil også finne eksempler på kode.

Kapittel 5 Test og kvalitetssikring
Beskriver hva vi har gjort for å ivareta kvaliteten på systemet. Hvilke teststrategier vi har benyttet oss av igjennom utviklingsprosessen, med eksempler på disse.

Kapittel 6 Oppsummering
Beskriver resultatet av prosjektet opp i mot den definerte kravspesifikasjonen, og drøfter det opp i mot de mål og rammer vi hadde satt for prosjektet.
Omtaler hva som står igjen for å oppfylle den fullstendige kravspesifikasjonen, dette var ikke vårt mål i løpet av dette prosjektet.

Presiserer hva vi har implementert i forhold til den komplette kravspesifikasjonen og dokumentasjonen vi har utarbeidet.

Kapittel 7 Konklusjon
Dette kapitlet er en kort oppsummering av prosjektet. Hvilke nyttige erfaringer vi har gjort, hva kan vi dra nytte av senere i livet.

Kapittel 8 Litteratur
Litteraturliste, bøker og internettressurser.

Kapittel 9 Vedlegg
Oversikt over alle vedlegg til rapporten.

1.2 Definisjon av oppgaven

1.2.1 Beskrivelse av oppgaven

Oppgaven går ut på å lage et enkelt salgsstøttesystem for byggevarehandelen som er godt tilpasset brukerne.

Målet med vårt system er å gi et alternativ til brukere av eksisterende systemer, samt å favne nye brukere som pr i dag faktisk bruker gammeldags kassaapparatet på grunn av at de ikke har økonomi til å investere i de dyre og komplekse løsningene som finnes på markedet.

Vare- og varegrupperegisteret i systemet skal baseres på byggevarebasen NOBB [NOBBweb].
Systemet skal bestå av to 'moduler', en for bruk i kasse i butikk og en for kontorbruk.

I kassen skal alle faser i forbindelse med et salg kunne håndteres, fra skanning av strekkode til betaling med bankkort, kontant eller kreditt (ordre).
Det skal også være mulig å finne og registrere nye kunder i kassen, samt se på informasjon om varer, priser, tilbudskampanjer og rabattsatser. Kundenes historikk på henholdsvis tilbud, ordre og rabattavtaler er også tilgjengelig.

Kontormodulen har utvidet funksjonalitet i forhold til kassen. Den funksjonaliteten som finnes i kassen er også tilgjengelig her, men selve 'salgsregistreringsbildet, vil være noe forskjellig.
Her kan også alle data registreres og oppdateres. Det skal være mulig å opprette generelle rabattmatriser på grunnlag av definerte kundegrupper. Det skal også være mulig å definere rabattavtaler på grunnlag av enkeltkunder.
Det finnes også rutiner for prisbehandling, felles kalkulasjon av varer, samt rutiner for behandling av hyllekantetiketter for pris- vareinformasjon. Det skal være rutiner for eksport av prisinformasjon til Microsoft Excel.

1.2.2 Vår avgrensning

Vi besluttet i forkant av prosjektet å legge vekt på grunnleggende og vesentlig funksjonalitet i systemet. Da vi i løpet av den tiden vi har til rådighet i dette prosjektet umulig kan rekke å implementere all ønsket funksjonalitet fullt ferdig, har vi gjort visse bevisste avgrensninger.

Vi bestemte tidlig å ikke implementere noe funksjonalitet for eksport, da det ikke på daværende tidspunkt var klarlagt hvilket format data skulle eksporteres til. Dette er enda ikke klarlagt.

Utvalget av rapporter / utskrifter vil også bli begrenset.
Interface mot kortterminal (bankkort) har vi også bevisst valgt å ikke ta med, da vi mener det er fornuftig å kjøpe dette ferdig.

Importrutiner for data eksportert fra NOBB er en vesentlig del i dette systemet for at en skal kunne nyttiggjøre seg den omfattende vareinformasjonen som finnes der. Å registrere dette manuelt vil være en enorm jobb.

I oppstartsfasen hadde vi planer om å implementere dette, men det ble tidlig klart at oppdragsgiver ikke kunne skaffe oss tilgang på NOBB. Da bestemte vi oss for å utelate denne delen fra prosjektet, da det er meningsløst å bruke ressurser på å implementere noe en ikke med sikkerhet vet hvordan skal fungere. Vi forsøkte også å kontakte produsenten, uten å få noen respons.

Noen installasjonsprosedyrer for systemet har vi ikke tatt i betraktning, heller ikke brukeradministrasjon, eller noen form for automatisert backup.
Dette er forhold som må vurderes i ettertid av dette prosjektet.

Men selv med disse avgrensninger har oppgaven vært mer enn omfattende nok, de manglende deler bør være greie å bygge inn i systemet i ettertid.

Generelt så vil kravspesifikasjonen vår være noe mer omfattende enn det vi vil implementere i løpet av prosjektet.

1.3 Målgruppe for systemet

Målgruppen for dette systemet er primært handelsbedrifter innen byggevarehandelen med fra en til opptil 20 brukere. Større bedrifter vil sannsynligvis ha større nytte av andre eksisterende systemer.

1.4 Målgruppe for rapporten

Primært er rapporten skrevet for oppdragsgiver, veileder og sensor, samt datafaggruppen ved Høgskolen i Gjøvik. Også nåværende og kommende datastudenter kan ha nytte av rapporten.

Men rapporten med vedlegg har også nytteverdi for de som skal videreutvikle systemet, med utgangspunkt i resultatet fra dette prosjektarbeidet. For disse vil spesielt kapittel to, tre og fire med vedlegg være interessante.

1.5 Gruppemedlemmenes bakgrunn

Kjetil Ophus,	3-årig dataingeniør, studieretning for systemutvikling.
Lars Otto Nymoen,	3-årig dataingeniør, studieretning for systemutvikling.
May-Britt Hansen,	3-årig dataingeniør, studieretning for programvareutvikling.
Oddbjørn Brede Sjulstad,	3-årig dataingeniør, studieretning for drift.
Vivian Atterås,	3-årig dataingeniør, studieretning for systemutvikling.

Vivian, Lars-Otto og Kjetil har i forkant av dette prosjektet dannet seg et rimelig godt forhold til problemstillingen, da de gjennomførte et prosjekt mot samme oppgaven i faget Objektorientert Systemutvikling ved HiG høsten 2001.
I det prosjektet ble rammeverket UP [RUP] benyttet, og det gav stort sett bare positive erfaringer.

Gjennom ingeniørstudiet har vi grunnleggende kunnskaper fra en rekke forskjellige fag som vi regner med å få utdypet i løpet av dette prosjektet. Spesielt vil nok våre kunnskaper innen databaser, systemutvikling og programmering bli utdypet i løpet av hovedprosjektoppgaven.

1.6 Arbeidsmetodikk i prosjektet

1.6.1 Innledning

På bakgrunn av de erfaringer vi har med UP som rammeverk, besluttet vi å benytte oss av denne metoden i dette prosjektet.

UP er 'bygget' med det utangspunkt at den skal utnytte styrken i alle de andre kjente utviklingsmodellene, for så å sy dette sammen til en helhetlig utviklingsmodell.
Dette mener vi gjør UP til en god modell. Dermed faller på sett og vis grunnen til å velge de andre modellene vekk, da vi med UP får med den enkelte modellens fortrinn uansett.
At den er iterativ og dermed god til å takle endringer anser vi som en stor fordel i vårt prosjekt.

I og med at vi er uerfarne, vil det sannsynligvis inntreffe flere behov for endringer enn om erfarne utviklere skulle utført prosjektet. Dette ville blitt tyngre for oss å takle med f. eks en tradisjonell 'fossefallsmodell'.

En alternativ modell er 'extreme programming', men den anså vi som uaktuell. Et viktig prinsipp med den er at den forutsetter høy grad av medvirkning fra kunde, såkalt 'onsite customer'. Dette hadde ikke vi mulighet til i dette prosjektet. Vi anser heller ikke dette systemet for å være så 'eksepsjonelt' at bruk av denne modellen har noe for seg.

Den dokumentasjonen som forelå fra prosjektet før jul, var et naturlig hjelpemiddel å ta med inn i dette prosjektet. Men vi har ikke lagt altfor stor vekt på det, da det prosjektet må anses for å være veldig 'teoretisk'.
Der var det kun snakk om dokumentasjon av analyse og design, samt litt systemarkitektur.
Det viste seg fort at dette måtte revideres til dels omfattende.

1.6.2 Hvordan vi har lagt opp arbeidet

Vi har hele tiden lagt opp arbeidet etter UP sine prinsipper, med gjentatte iterasjoner. Denne utviklingsmetoden er forholdsvis fleksibel, den har et rikholdig utvalg av artifakter som kan benyttes etter behov. Dette har vi bevisst benyttet oss av.

Vi har arbeidet etter en overordnet faseplan, som setter de grove rammene for prosjektet. Hver fase inneholder flere mer detaljerte iterasjonsplaner. Disse beskriver hvilke aktiviteter som skal foregå i gjeldende iterasjon. Disse er utarbeidet fortløpende av prosjektleder.

Vi har hele tiden fokusert på å jobbe parallelt med forskjellige oppgaver. Dette er etter vår oppfatning et viktig kriterium for å lykkes med et prosjekt av denne størrelsen.

Gruppen har hatt ukentlige interne statusmøter, av forholdsvis uformell art. På disse møtene ble status gjennomgått, og arbeidsoppgaver fordelt.

Det var også planlagt møter med veileder til fastsatt tid hver andre uke, men disse har i stor grad uteblitt. Statusrapporter er levert til veileder og oppdragsgiver til fastsatte tider.

Kontakt med oppdragsgiver har foregått gjennom prosjektleder, disse har stort sett kommunisert via telefon og email. Det har også vært enkelte møter med oppdragsgiver.

I de tilfeller det har dukket opp behov for assistanse fra andre personer, har gruppa på eget initiativ opprettet den nødvendige kontakten.

1.6.3 Eksempel på iterasjonsplan

Iterasjonsplan: Elaborasjon Iterasjon 1

Dato: 22.01.02
Forfatter: Kjetil Ophus (prosjektleder)

Varighet: 25.01.02 – 11.02.02

Ansvarlig :
gjennomføring: hele gruppen
overordnet ansvar: prosjektleder

Innledning:

Denne iterasjonen vil i hovedtrekk bestå av å designe og implementere en database (testversjon), nødvendige analyse- og designdokumenter for å starte implementasjon av kunde- varedelene av systemet, samt strekkodeskanner. For vare og kunde inngår også søkefunksjonalitet på disse.

Arbeidsoppgaver fordeles fortløpende av gruppeleder.

Innhold:

- Domenemodell for systemet skal være klar
- Arkitektur analyseres , dokumentasjon av dette forberedes. Viktig at denne stabiliseres i denne fasen.
- Kontinuerlig vedlikeholde all supplementær spesifikasjon
- Aktuelle usecase med nødvendige interaksjonsdiagrammer og designdokumenter for kundemodul utarbeides, gui for denne del skal implementeres. Det meste av funksjonaliteten rundt denne skal også implementeres.
- Aktuelle usecase med nødvendige interaksjonsdiagrammer og designdokumenter for varemodul (inkl. leverandør og varegrupper) utarbeides, GUI for denne del skal implementeres. Det meste av funksjonaliteten rundt denne skal også implementeres.
- Implementasjon av strekkodescanner startes i denne fase. Fullføres ikke i denne iterasjon!
- Usecase og nødvendige analyse- og designdokumenter for ordredelen i systemet påbegynnes.
- Iterasjonsplan for iterasjon2 i elaborationfasen skrives

Mål:

Ved endt iterasjon bør store deler av den grunnleggende arkitekturanalysen være stabilisert, og dokumentasjon av denne påbegynt. Vi skal ha gjort en del begrunnede valg for hvordan systemet skal organiseres. GUI med noe tilknyttet funksjonalitet for kunde- og varedel skal være implementert. En første utgave av databasen skal være implementert.

For komplette planer for prosjektet, se *VEDLEGG J.*

2. KRAVSPESIFIKASJON

2.1 Innledning

2.1.1 Bakgrunn

Et fellestrekk ved de løsninger som finnes på markedet for byggevarebransjen pr. i dag er at de er unødig komplekse, og ofte lite 'brukervennlige'. Det er et faktum at få eller ingen har bruk for all funksjonaliteten disse systemene tilbyr.

Et annet moment er at selv om disse løsningene er store og komplekse, fyller de kanskje ikke sluttbrukernes behov til fulle allikevel.
Disse systemene påfører også brukerne store IT-kostnader. Dette fører til at mange faktisk unnlater å investere i IT-systemer, rett og slett fordi 'inngangsbilletten' er for høy.

Erfaringer viser at det er en del problemer forbundet med bruken av de eksisterende løsninger. Flere av disse systemene kjøres på stormaskin med tilhørende terminaler med karakterbasert brukergrensesnitt. Dette føles tungvint og vanskelig for brukerne, og det har lett for å føre med seg mange unødvendige brukerfeil.

Disse påstandene tilsier at det finnes et behov for et enklere og rimeligere system som kun har den nødvendige funksjonaliteten, og som fyller brukerens behov til fulle. Med et standard windowsbasert brukergrensesnitt vil det bli enklere for brukerne å benytte seg av systemet, og vi kan dermed unngå unødige feilsituasjoner.
Med bakgrunn i den utbredelsen 'hjemmepc' har fått i dag, vil de aller fleste ha et eller annet forhold til windowsprogrammer.
Dermed blir det lett for brukeren å sette seg inn i systemet, og dette medfører igjen lavere opplæringskostnader for bedriften.

2.1.2 Generelt om systemkrav til ByggPOS

Tanken er at vare- og varegrupperegisteret i systemet skal baseres på byggevarebasen NOBB [NOBBweb]. Dette er en database med produktinformasjon for bygg- og anleggsbransjen.

Det skal også være mulig å bruke sin egen vare- varegruppestruktur, men denne må baseres på samme formatet som NOBB på grunn av at vår database baserer på denne strukturen.
I så fall må dette spesialtilpasses for hvert enkelt tilfelle.
Strukturen i NOBB er meget omfattende, og det skal ikke være påkrevd å benytte denne fullt ut.

Systemet består to 'moduler', en for bruk i kasse og en for kontorbruk. Begge jobber mot den samme databasen.
Prinsipielt kan begge disse modulene, samt databasen finnes på en og samme maskin. Dette er viktig for at systemet skal være mest mulig fleksibelt innenfor sin primære målgruppe.

2.1.2.1 Om kassemodulen i ByggPOS

I kassemodulen kan alle faser i forbindelse med et salg utføres.
Ved gjennomføring av et salg skannes eannummer inn via strekkode, alternativt identifiseres varen i kassadisplayet via søk, eller varenummer tastes direkte.

Det er mulig å betale kontakt, med bankkort, eller kreditt (faktura).
Fakturering er tenkt utført i eksternt system.
I kassen er det mulig å finne, samt registrere nye kunder. Det er tilgang til å se på kundenes historikk på ordre, rabattavtaler osv.
Det er også mulig å se på vareinformasjon og prisinformasjon.
Kassamodulen skal ha strekkodeskanner og kvitteringsprinter tilkoblet.

2.1.2.2 Om kontormodulen i ByggPOS

Den funksjonaliteten som finnes i kassen er også tilgjengelig her, men selve 'salgsregistreringsbildet', vil være noe forskjellig.
Her kan også alle disse data registreres og oppdateres. Det skal være mulig å opprette generelle rabattmatriser på grunnlag av definerte kundegrupper. Det skal også være mulig å definere rabattavtaler på grunnlag av enkeltkunder. Det skal kunne registreres tilbudsoppsett til kunder, hvor hele eller deler av dette tilbudet senere enkelt kan overføres til en ordre.

Det skal også være rutiner for prisbehandling, felles kalkulasjon av varer, samt for behandling av hyllekantetiketter for pris- vareinformasjon. Det skal være mulighet for eksport av prisinformasjon til Microsoft Excel.

Det er meget viktig at prisene behandles på en slik måte at systemet legger til rette for at det alltid skal være samsvar mellom prisene i systemet og prisene i hyllene ute i butikk.

Dette fungerer slik at ved alle endinger som gjøres i systemet som påvirker en paknings pris, skal systemet automatisk generere datagrunnlag for en prisetikett, denne legges så til i en 'kø'.

Det bør også være mulig å kunne eksportere datagrunnlag for senere import i faktura- økonomisystemer. Importfunksjonalitet for å kunne opprette varestrukturen på grunnlag av eksporterte data fra NOBB skal også finnes her. Diverse rapporteringsrutiner finnes også, eksempelvis dagsoppgjør.

For å opprette datagrunnlaget i systemet på grunnlag av andre eksisterende registre, må dette spesialtilpasses hvert enkelt tilfelle.

Denne modulen vil fungere som administrasjonsenhet i systemet.

2.1.3 Begge moduler detaljbeskrevet sammen

I de usecase som er skrevet for å detaljere kravspesifikasjonen, er kasse og kontormodulen beskrevet i sammen.

Dette er gjort bevisst, da deling kun vil medføre unødvendig gjentagelse av beskrivelser, siden all funksjonalitet som skal være i kassemodulen også skal være tilgjengelig i kontormodulen, i alle fall på tilnærmet samme måte.

Hvilken funksjonalitet som er tilgjengelig hvor, synliggjøres gjennom den supplerende spesifikasjonen, samt gjennom designmodeller og beskrivelser fra kapittel 3 og vedlegg.

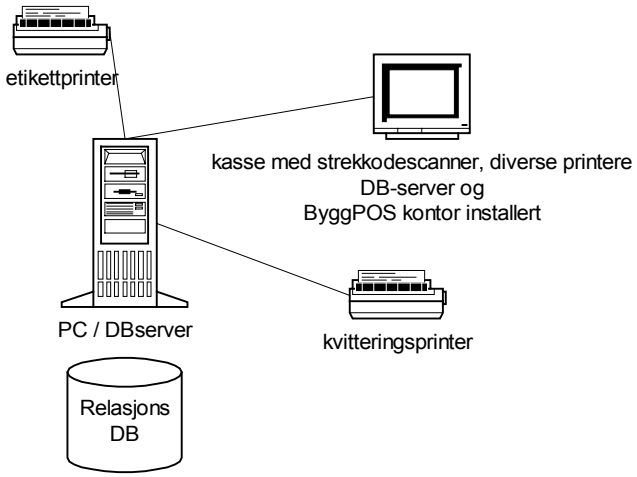
2.1.4 Systemets omgivelser

Systemet skal benyttes av handelsbedrifter innen byggevarehandelen, primært bedrifter med opptil ca 20 brukere. Systemet skal kjøres på Microsoft Windows operativsystem, med en Borland Interbase 6.0 databaseplattform.

Standard maskiner skal benyttes, med andre ord standard server, stasjonære og / eller bærbare maskiner. Standard printere (kvittering, etikett og vanlig) skal benyttes. Standard strekkodeskannere tilkoblet serieport (RS 232) benyttes.

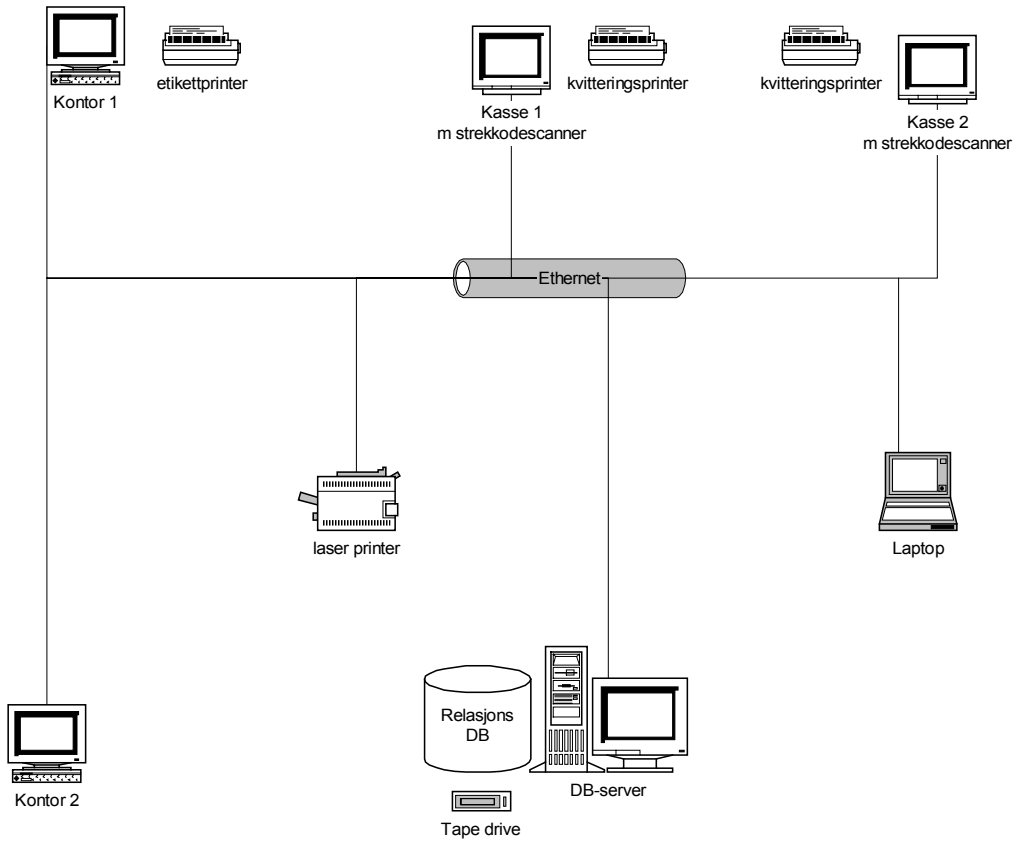
Løsningen skal være fleksibel, i enkleste form kan den komplette løsningen kjøres på en enkelt maskin (figur2-1).
Men systemet kan også kjøres som en nettverksløsning (figur 2-2), med egen databaseserver og mange klientmaskiner.

Eksempel på en enkel løsning, der alt kjøres på en enkelt maskin.



FIGUR 2-1: ENKELT MILJØ

Eksempel på en nettverksbasert ByggPOS løsning



FIGUR 2-2: ET STØRRE NETTVERKSMILJØ

2.1.5 Systemets brukere

Brukerne av systemet vil være ansatte i handelsbedrifter innen byggevarehandelen. Det er lite sannsynlig at disse besitter noen stor datakunnskap, men på grunnlag av den utbredelsen datamaskiner har fått i dag, vil antakelig de aller fleste potensielle brukere ha et eller annet forhold til windowsprogrammer. Dette prinsippet er det viktig å utnytte.
Derfor er det viktig at det grafiske brukergrensesnittet i systemet lages enkelt og intuitivt, i kjent 'windowsstil'.

Dermed blir det lett for brukeren å sette seg inn i systemet, og unødige misforståelser og feilsituasjoner kan unngås.
Med et enkelt brukergrensesnitt vil også opplæringskostnadene for bedriften bli lavere, det vil bli enklere for brukerne å lære seg systemet.

Administrasjonen av systemet skal også være enkel, med tanke på at den skal utføres av personer med begrenset datakunnskap.

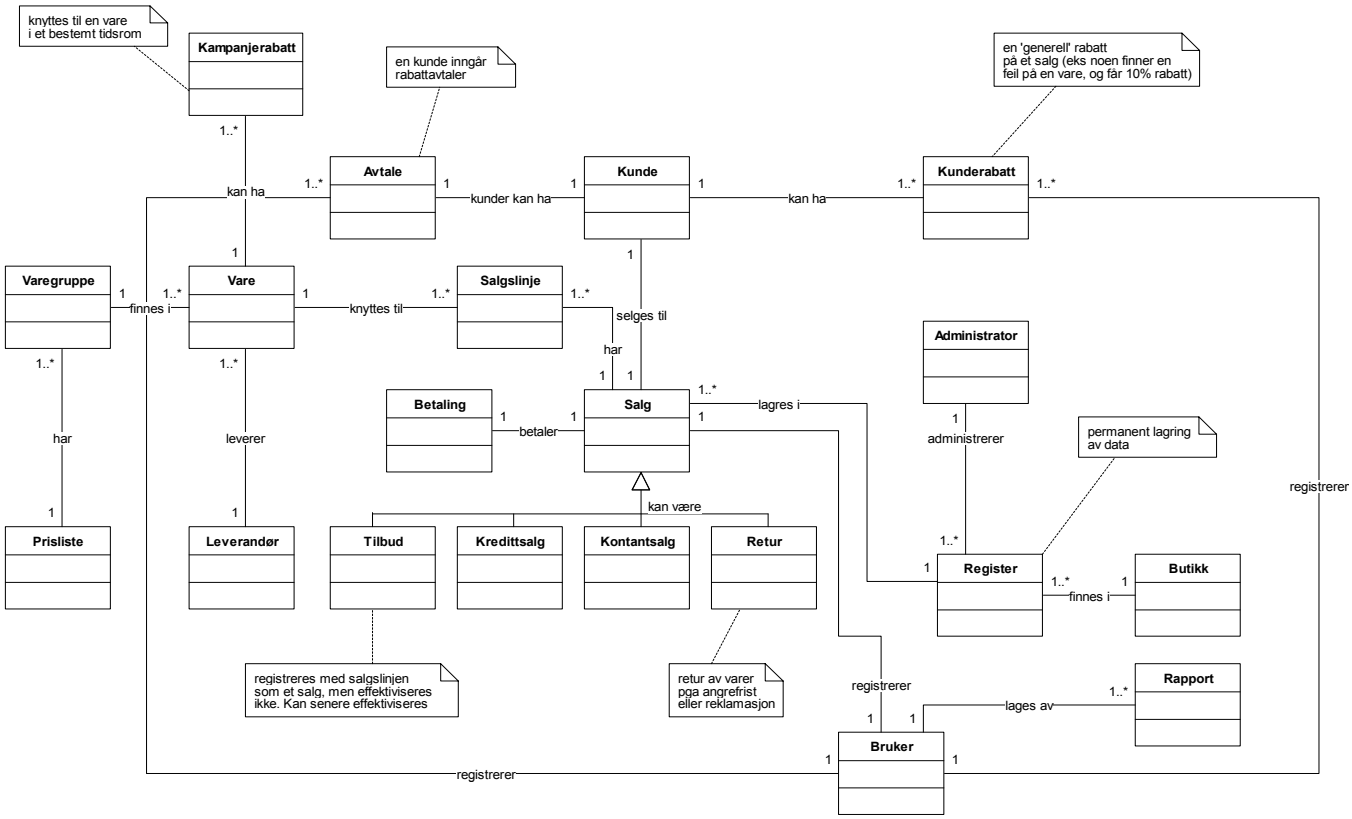
Systemets to applikasjoner er tilknyttet brukergrupper med forskjellige rettigheter, kun kontorapplikasjonen med administratorrettighet vil ha mulighet til å endre på systemets grunndata, eksempelvis rabattmatriser og prisbehandlingsrutiner.

2.2 Detaljert kravspesifikasjon

2.2.1 Domenemodell

Domenemodellen (figur 2-3) skal beskrive systemet ut i fra et virkelighetsperspektiv. Poenget med denne modellen er å få frem de fysiske objekter som finnes i det fysiske miljøet systemet skal operere.
Modellen er et godt hjelpemiddel videre i utviklingen, når softwareklasser skal designes.
Men bemerk deg at det er ikke alltid er noen klar og selvfølgelig sammenheng mellom klassene i den konseptuelle modellen og designmodellen.

Domenemodell, også kalt konseptuelt klassediagram



FIGUR 2-3: DOMENEMODELL FOR 'BYGGPOS'

2.2.2 Diverse supplementær spesifikasjon

2.2.2.1 Diverse forretningsmessig informasjon

- Riktig mva-grunnlag må ivaretas ved registrering av kalkulasjonsregler
- Pris til kunden er alltid den pr. i dag laveste (rabattavtale, generell rabatt, kampanjetilbud).
- Pakningskoder angis med tre bokstaver, eksempelvis STK.
- Varer må kunne rabatteres både på enkeltvare og gruppevis. Dette for å kunne gi større rabatt på bransjens 'brødvarer', for eksempel 24X98, justert, T18. Utover dette må rabatt også kunne gis på annet 'udefinert grunnlag'.
- Prisoppdateringer defineres alltid som prosent.
- Ved oppdateringer som påvirker en vares pris, bør prisetiketter skrives ut for at det alltid skal være samsvar mellom hyllepris og pris i datasystemet.

2.2.2.2 Begrensninger

Operativsystem og databaseplattform

Systemet skal og må kjøres på Microsoft Windows plattform
(9X, NT4.0, 2000, XP).
Databaseplattformen for ByggPOS er primært Borland Interbase 6.0.
Systemet *kan* om ønskelig tilpasses for andre databaseplattformer hvis en kunde skulle ha spesielle ønsker om det. Det vil i så fall måtte spesialtilpasses.

Strekkodescanner

Kassemodulen støtter bruk av strekkodeskanner for lesing av strekkodeetiketter. Men det er et krav at skanneren er tilboblet serieporten på maskinen.

Maskinkrav

Systemet har ikke noen fastsatte maskinkrav, det må vurderes ut fra det enkelte tilfelle. For klientene og kassene er en 'standard PC' av nyere årgang tilstrekkelig. Kravene til databaseserver må vurderes etter antall klienter.

Dataimport fra generelle registre

Systemet skal kunne importere data fra eksisterende systemer. Aktuelle data kan være kunderegister, vareregister, leverandørregister. Dette vil være engangshendelser som må sees som en installasjonsjobb, med de tilpasninger som er eventuelt er nødvendig for det aktuelle tilfelle.
Dataformatet i dette systemet vil være tilpasset datastrukturen fra NOBB, data eksportert fra denne vil med tilpasninger derfor kunne importeres for å opprette vare- og varegruppeutvalget i vårt system.

Det spesifiseres at det er kun for å opprette grunnlaget dette er aktuelt, oppdateringer vil måtte gjøres enten ved å gjøre en ny fullstendig oppretting, eller utføres manuelt ved hjelp av rutiner i vårt system.

Dataeksport til faktura- økonomisystemer

Systemet bør kunne eksportere data til fil som danner importgrunnlag i eksterne kommersielle faktura- økonomisystem. Det er pr. i dag ikke tatt stilling til dette.

Eksport av datagrunnlag for økonomisystemer kan eventuelt også i en viss grad tilpasses det aktuelle systemet kunden benytter. Dette må derfor programmeres for hvert enkelt system.

2.2.2.3 Grensesnitt

- Monitor
- Strekkodeleser
- Etikettprinter
- Kvitteringsprinter
- Tastatur
- Mus
- Betalingsterminal (kort-terminal)
- Faktura /økonomisystemer

2.2.2.4 Menneskelige faktorer

Kassadisplayet må være ryddig og oversiktlig. Det skal være en gjennomgående lik struktur i alle bilder, like knapper har samme funksjonalitet i alle bilder.
Knapper og tekst bør være stort og lett synlig.
Brukergrensesnittet må være ryddig og oversiktlig, det gjør det lett og lære og huske.

Lyd bør vurderes som varsling ved eventuelle feil, da kassaoperatør kanskje har øyekontakt med kunde, og ikke nødvendigvis ser på displayet. Systemet skal kunne opereres uten bruk av mus.

2.2.2.5 Systemets pålitelighet , håndtering av feilsituasjoner

Det overordnede mål er at systemet skal ha en oppetid på 100%. Dette er selvsagt uoppnåelig i praksis, men det bør være mulig å komme tett inntil.

Databasen bør i utgangspunktet speiles, da Interbase ikke har transaksjonslogger. Ved bruk av speilig til to fysisk forskjellige disker, og med periodisk manuelle backuprutiner, vil systemet være rimelig optimalt sikret for tap av data.

Hvis disse rutiner følges vil tap av data være minimalt. Ved en feilsituasjon som gjør at systemet går ned, vil da kun de transaksjoner som var aktive akkurat i det kritiske øyeblikket måtte kjøres om igjen.

Ved feilsituasjoner som oppstår på klienten(e), vil kun den operasjonen som ble utført i det kritiske øyeblikket måtte utføres på nytt. Databasen vil alltid forbli i en riktig tilstand, det sørger systemet selv for.

Systemet skal generelt ikke tillate at noen av klientapplikasjonene utfører noen ulovlige operasjoner på databasen.

2.2.2.6 Systemets livssyklus

Systemdokumentasjon

Det vil ved utgangen av dette prosjektet finnes en rimelig fyldig dokumentasjon av systemet, spesielt analyse- og designdokumenter. Databasen vil også være godt dokumentert.
Systemet vil også ha en integrert hjelpefil, denne må foreløpig fungere som brukermanual, da manualen ikke er prioritert enda.

Men siden dette er første betautgave av systemet, vil neppe dette bli den endelige systemdokumentasjonen. Men den danner et godt grunnlag for videre utvidelse under videreutviklingen / ferdigstillelsen.

Det er meget viktig at all dokumentasjonen oppdateres og holdes à jour ved fremtidige endringer i systemet.

Videreutvikling av systemet

Systemet vil ikke bli ferdig utviklet i løpet av dette prosjektet. Men med utgangspunkt i den systemdokumentasjonen som foreligger, er målet at det skal være enkelt å videreutvikle systemet.

Det har hele tiden vært meningen at noen, hvem er ikke avgjort, skal overta systemet når dette prosjektet er ferdig. De skal fortsette der vi slapp, og utvikle systemet frem til et fullt funksjonelt, testet og salgbart system.

Vedlikehold og support

Ideen er at det i fremtiden ferdige systemet kontinuerlig vedlikeholdes. Dette utføres av den eller de som etter dette prosjektet får ansvar for å videreutvikle systemet.

Tanken er at kunder som benytter seg av systemet betaler en årlig avgift for vedlikehold og support. Kunden vil da ha fri tilgang til support, samt alle oppdateringer.

Det er naturlig å sette den årlige kostnaden til en prosentsats av den opprinnelige lisenskostnaden.

2.2.2.7 Installasjon og tilpasninger

Systemet installeres og testkjøres av representanter fra utviklerne. Eventuelle spesialtilpasninger utføres også av disse.

Det forutsettes at kunden har den påkrevde maskinvare tilgjengelig.

2.2.2.8 Opplæring og hjelp

Systemet skal ha en hjelpefil som er tilgjengelig i alle skjermbilder. Det utarbeides også en brukermanual.

Kurs for den eller de av brukerrepresentantene som har rollen som systemadministrator kan også være aktuelt.

2.2.2.9 Oversikt over overordnede operasjonelle systemkrav

På bakgrunn av de hittil nevnte punkter kan vi nå definere et utvalg overordnede operasjonelle krav til systemet. Disse krav skal ivaretas i løsningen.

Operasjonelle Systemkrav
Systemet skal kunne kjøre på standard PC, uten noen spesielle krav til maskinvarekomponenter. Systemet skal kunne benytte seg av standard kvitteringsprintere og etikettprintere.
Brukere skal autoriseres hver gang han skal gjøre et salg for at vedkommende skal tilknyttes den aktuelle ordren.
I kassen skal salgsenheter kunne identifiserer ved hjelp av strekkodeleser.
Systemet skal automatisk gi forslag til riktig pris på en varelinje.
Klientapplikasjonen skal ikke kunne sette databasen i en ikke-konsistent tilstand
Alle ordretransaksjoner i systemet skal føres med sekvensielle nummer i en fullstendig tallrekke. Hull skal ikke forekomme.
Ved alle endringer som påvirker en paknings pris, skal ny pris beregnes automatisk.
Gjennomsnittlig tid for varesøk under normale omstendigheter skal være maksimalt 0.5 sek.
Systemet skal og må kjøres i et windowsmiljø
Speiling bør benyttes pga. sikkerhet og pålitelighet. Systemet kjøres i utgangspunktet mot Borland Interbase, og derfor er speilig viktigere enn ved bruk av andre databaseplattformer som har transaksjonslogger I tillegg må periodisk manuell backup utføres.
Systemet skal ha integrert hjelpefil
Det skal være mulig å gjennomføre et salg i systemet uten bruk av mus. Det vil si at det skal være hurtigtaster for all nødvendig funksjonalitet.
Det skal være like knapper med den samme funksjonaliteten i alle bilder (der det er naturlig). Dette er viktig for å sikre god brukervennlighet.

2.2.2.10 tabellarisk oversikt over funksjonelle systemkrav

På bakgrunn av de til nå nevnte momenter, kan vi nå liste opp et utvalg funksjonelle systemkrav.
Disse vil igjen danne grunnlaget for usecasene (*Vedlegg G*), gjennom beskrivelsene der skal disse kravene bli ivaretatt.

Funksjonalitet
Systemet skal være tilrettelagt for å kunne importere informasjon om varer, varegrupper og leverandører fra bransjens byggevarebase NOBB for å opprette strukturen av varegrupper og varer. Dette forutsetter at det er mulig å få tak i de aktuelle data. Dette krever abbonement på NOBB-basen. (Det skal også være mulig å importere data fra eksisterende registre uavhengig av NOBB.) I praksis vil dette si at databasen i dette systemet er basert på datagrunnlaget som finnes i NOBB. Rutiner for import må implementeres i hvert konkret tilfelle med bakgrunn i det eksisterende datagrunnlag. (Kundetilpasses)
Systemet skal kunne registrere/endre bedriftsinterne data(brukere, org.nr, kontonr, osv..), rabattmatriser, varegrupper, varer og kalkulasjonsregler. En kalkulasjonsregel sørger for felles prisbehandling av en samling varer.
Systemet skal kunne registrere data om kunder, disse data skal også kunne importeres fra allerede eksisterende kunderegistre forutsatt at dataformatet er tilpasset. Import er å betrakte som en installasjonsjobb, og er kun aktuelt for å opprette strukturen. Historikk (ordre, gitte tilbud, rabatter) skal være tilgjengelig for en valgt kunde.
Systemet skal ha et kassadisplay hvor både kontantsalg og kredittsalg kan gjennomføres. Varer registreres enten ved strekkode, eller søk på varenummer, eller fritekst. Alle salg er å betrakte som en ordre, ved 'vanlig' salg over disk, vil kunden få et standard kundenummer som ikke har noe spesifikk informasjon tiknyttet seg. Gjeldende bruker er identifisert i bildet helt til han logger seg av. Ordrenummer genereres, og eventuelle rabatter og avtaler effektiviseres. En ordre kan eventuelt merkes med referanser. Ordrebekreftelse og eventuelt plukkliste skal skrives ut. Ved kontantsalg skal kvittering med gjenpart skrives ut. Det skal være et eget(utvidet) brukergrensesnitt for kontoransatt/administrator. Her kan div. administrative oppgaver utføres, eksempelvis å legge inn nye varer være mulig. Kun kunder skal kunne registreres (i tillegg til ordre) i kassen.

Ved avsluttet salg skal data lagres permanent i systemet. Dette danner grunnlag for bruk i diverse rapporter. Data kan eventuelt eksporteres til fil for import til faktura/økonomisystem.
Systemet skal ha funksjonalitet for søk på varer, varegrupper og kunder med deres tilhørende avtaler og rabatter. Generelt skal det være mulig å søke på 'fornuftig valgte parametere' i alle deler av systemet.
Registrere en tilbudsforespørsel fra kunder og gi kunden tilbudsoppsettet. Hele eller deler av dette tilbudet skal senere kunne overføres direkte til et salg. (forutsetter selvfølgelig at kunden har akseptert tilbudet.)
Systemet skal ha mulighet for å håndtere varer som kommer i retur.
Utskrift / uttak av diverse rapporter på tilbud / salg.
Prislister skal kunne eksporteres til Microsoft Excel, og de skal kunne skrives ut direkte.
Hyllekantetiketter / strekkodeetiketter skal kunne skrives ut. Ved all oppdatering i systemet som omfatter pris på et eller annet vis (i praksis vil dette være prisoppdateringer og endring av kalkulasjonsregler) bør nye hyllekantetiketter automatisk skrives ut. Dette er nødvendig for at det alltid skal være samsvar mellom den prisen kunden finner for en vare og denne varens pris i datasystemet.

2.2.3 Usecase for systemet

Med utgangspunkt i analysering av den overordede kravspesifikasjonen har vi definert 25 usecase som til sammen skal dekke systemets krav til funksjonalitet. Disse beskriver på en 'ikke-teknisk' måte hva som skal skje for at brukeren skal få utført en ønsket handling av systemet.

Vi har for oversiktens skyld valgt å kun vise ett usecase her i rapporten, den komplette beskrivelsen finnes i VEDLEGG G.

Følgende usecase er definert:

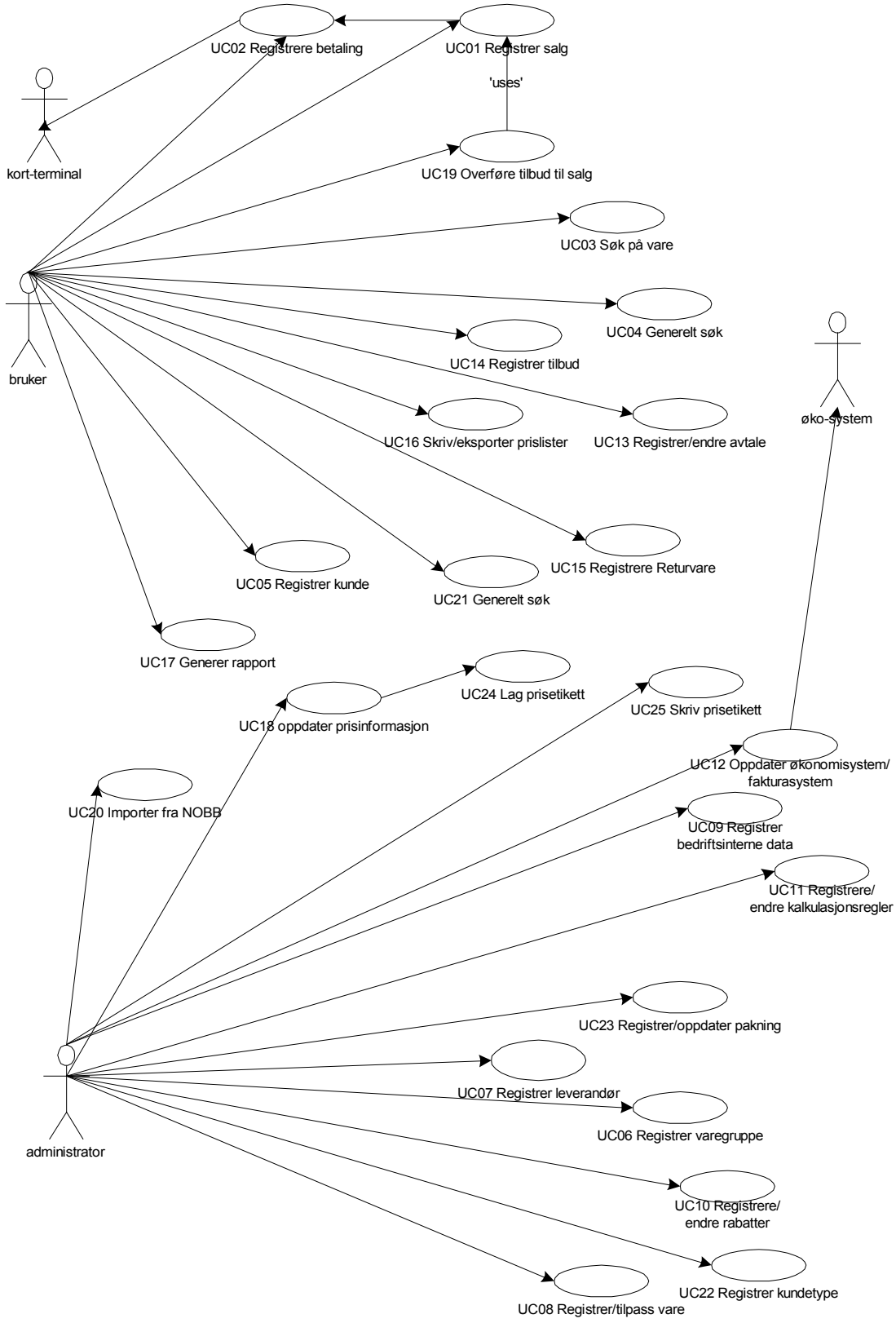
UC01 Registrer salg
UC02 Registrere betaling
UC03 Søk på vare
UC04 Søk på kunde
UC05 Registrer kunde
UC06 Registrer varegruppe
UC07 Registrer leverandør
UC08 Registrer/tilpass vare
UC09 Registrer bedriftsinterne data
UC10 Registrere/endre rabatter
UC11 Registrere/endre kalkulasjonsregler
UC12 Oppdater økonomisystem/fakturasystem
UC13 Registrer/endre avtale
UC14 Registrer tilbud
UC15 Registrere Returvare
UC16 Skriv/eksporter prislister
UC17 Generer rapport
UC18 oppdater prisinformasjon
UC19 Overføre tilbud til salg
UC20 Importer fra NOBB
UC21 Generelt søk
UC22 Registrer kundetype
UC23 Registrer/oppdater pakning
UC24 Lag prisetikett
UC25 Skriv prisetikett

Usecase-modell for ByggPOS

På neste side finnes en modell (figur 2-4) som viser interaksjonen mellom usecasene og systemaktørene, samt mellom usecase internt.
Det presiseres i forhold til denne oversikten at administrator også innehar rollen som bruker.

Kort fortalt beskriver et usecase av handlingsforløpet som må utføres for å oppfylle en bestemt del av kravene til et datasystem.
Det er viktig å merke seg at et usecase bestemmer hva som skal utføres, men på ingen måte hvordan det skal implementeres.

Usecasemodell



FIGUR 2-4: USECASEMODELL BYGGPOS

Usecase Registrer salg
Eksempel på et detaljert usecase, her beskrives gjennomføringen av et salg.

UC 01 : Registrer salg

Aktør: Bruker
Type : Primary

Mål:
- Bruker får rask tilgang
- Kunde får rask service

Pre-betingelse: Bruker er logget inn med lovlig brukerid

Hovedsenario:
1. Kunden kommer til kassen/ringer for å kjøpe/bestille varer.
2. Bruker starter nytt salg.
3. Bruker registrerer kunden i ordrebildet.
4. Bruker legger inn vare (ref. UC: 03 Søk på vare).

→ Punkt 4 repeteres til alle varer er registrert

5. Systemet registrerer varelinjen og viser varebeskrivelse, pris, antall og linjesum. Pris kalkuleres ut i fra rabattmatrise.
6. Bruker avslutter salg.

Extensions:
2a. Systemet genererer et Ordrenummer.
3a. Kunden er en tilfeldig kunde,
1. Det genereres et dummy (standard) kundenummer som kontantkunde.
3b. Kunden er en fast kunde.
1 Skriver inn kundens navn.
2 Systemet søker i kunderegisteret etter identifikasjon(ref. UC: 04 Søk kunde).
1. Systemet finner ingen matchende kunde og sender tilbakemelding om dette til bruker som må registrere kunden (ref. UC: 05 Registrer kunde)
2. Systemet finner kunden i registeret og viser kundedata i ordrebildet.
3. Systemet legger inn kundereferanser til ordrebildet.
3c. Systemet sjekker om kunden er registrert med tilbud.
1. Har kunden fått tilbud på ordren, blir tilbudet overført og effektuert som ordre (ref.UC: 19 Overføre tilbud til salg).
4a. Prisen som presenteres er ikke ønskelig
1.Bruker taster inn ny pris
2.Systemet presenterer ny pris
4b. Kunden ber bruker om å fjerne en vare fra salget
1. Bruker taster inn taster inn varenummeret på varen som skal fjernes
2. Bruker velger fra varelinjene (med mus eller tastatur) hvilken vare som skal fjernes
4c. Kunden ber bruker om å avbryte salget
1.Bruker avbryter salget
4d. Bruker setter salget på venting
1.Systemet lagrer salget slik at det kan hentes fram igjen fra hvilken som helst kasseterminal
4e. Ønsket vare er utsolgt eller definert som bestillingsvare
1. Bruker legger inn bestilling på varen
2. Kunden ønsker ikke lenger å kjøpe varen, og gjeldende varelinjeregistrering avbrytes.
5a. Bruker er registrert med avtale på innlagt vare (ref. UC: 13 Registrer avtale).
1. Avtalte rabatter effektueres ordren
5b. Det føres kampanje på innlagt vare
1. Systemet viser bruker kampanjepris i stedet for ordinær pris
NB! Alltid laveste pris som gjelder
5c. Kunden ønsker varene levert
1.Bruker legger inn leveringsadresse og dato for levering .
Kan evt. merkes med referanse (f.eks. byggeplass) og kjørerute.
2. Plukkliste skrives ut med Kundeinformasjon, Varenummer, antall og leveringsadresse.
6a. Betaling gjennomføres. (ref UC: 02 Registrere Betaling)

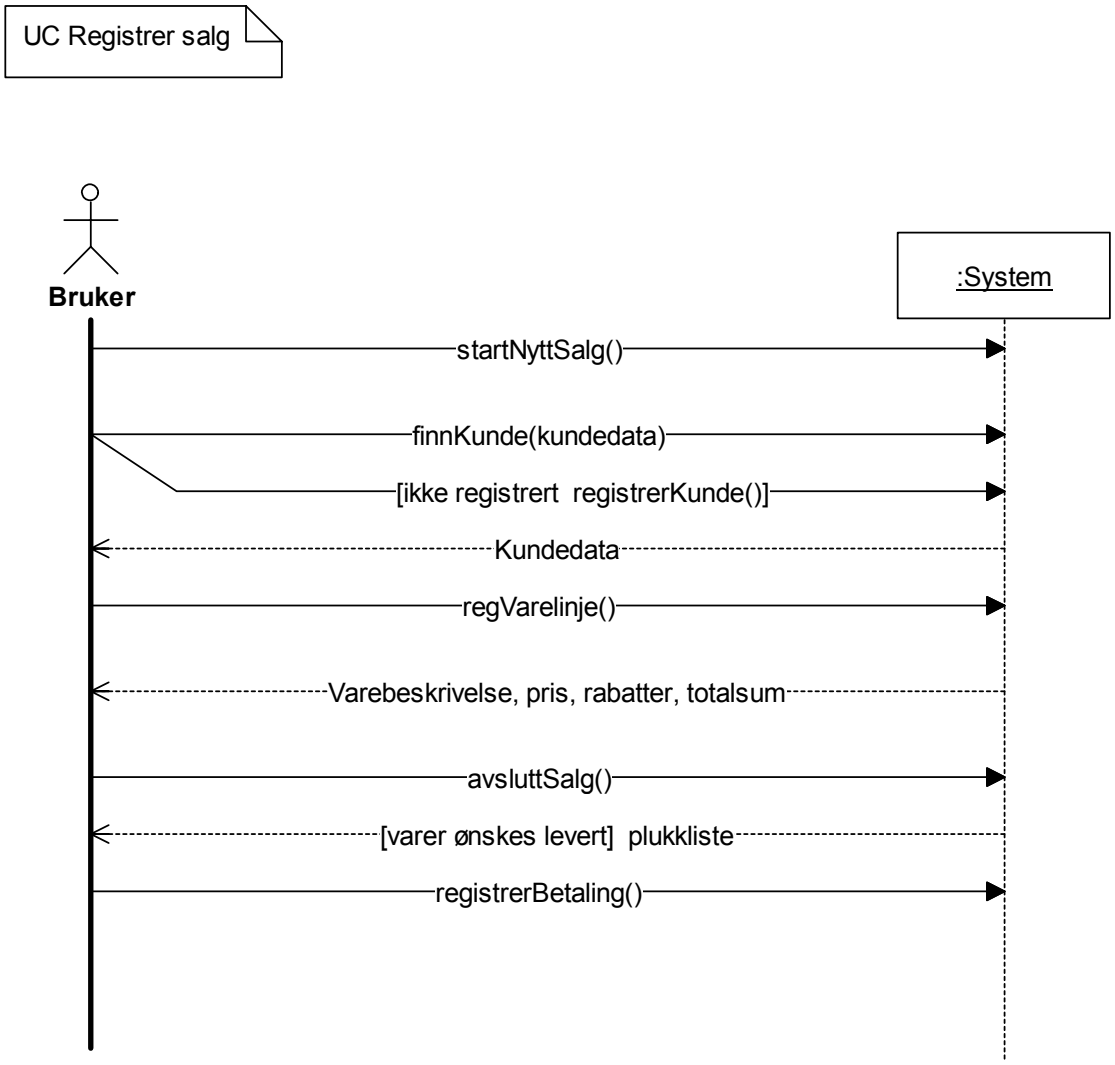
2.2.4 Systemsekvensdiagrammer

Formålet med disse diagrammene er å beskrive handlingssekvensen som ut utføres mellom bruker og system for å gjennomføre et usecase.
Meldingene som går mellom bruker og system gir oss en oversikt over hva som skal skje i gjennomføringen av usecaset.
Disse meldingene gir oss også en pekepinn på hvilke funksjoner som blir nødvendige i softwareklassene som skal utarbeides senere.

Diagrammer er utarbeidet kun for et utvalg av usecasene, vi anser det ikke som nødvendig å produsere et massivt utvalg av dokumenter for enkle og opplagte operasjoner.

Eksempel på systemsekvensdiagram
Eksempel på systemsekvensdiagram for usecaset 'registrer salg'.

For komplett oversikt, se VEDLEGG G, designdokumenter applikasjon



FIGUR 2-5: SYSTEMSEKVENSDIAGRAM FOR REGISTRERING AV SALG

2.3 Om kravspesifiseringsprosessen i prosjektet

2.3.1 Innledning

Kravspesifiseringen er den kanskje aller viktigste delen i ethvert systemutviklingsprosjekt. Hvis vi ikke får avdekket og klarlagt alle krav, kan det medføre at systemet som utvikles kanskje ikke blir det systemet kunden faktisk ønsker.
Og da hjelper det lite om systemet er aldri så godt laget.

Derfor er det viktig med et godt samspill mellom kunde og utviklere i denne delen av prosjektet. Det er mye enklere å gjøre endringer på dette nivået enn senere i utviklingsfasen. Kostnaden for endringer stiger med urovekkende høy faktor utover i utviklingsløpet.

2.3.2 Vurdering og rangering av Usecase

Vår vurdering
Tabellen viser at de usecase som innbefatter grensesnitt mot eksterne systemer bør få høy prioritet tidlig. Men da det ikke er tatt stilling til hvilke økonomisystemer som skal støttes, velger vi å ikke prioritere dette usecaset i prosjektet.

Da det innledningsvis i prosjektet ble klart at vi ikke fikk tilgang på NOBB, avgjorde det at også dette usecaset ble nedprioriert.

Salg er kjernefunksjonaliteten i dette systemet, tilsier det at alt omkring det må prioriteres høyt. For å kunne selge, må vare- og kundestrukturen være på plass. Derfor velger vi å prioritere alt omkring dette tidlig, selv om all 'grunndatabehandling' ifølge vår vurdering ikke er forbundet med noen høy risiko. Men ved å fokusere på 'grunndatabehandlingen' tidlig, gjør det implementeringen av alle aspekter rundt salg og rabattering enklere, da alle elementer som inngår i salget allerede er på plass.

I forhold til prototyping for grafisk brukergrensesnitt mener vi også at dette er riktig.

Vurderingen er gjort etter følgende kriterier :

A: Kritiske kjerneoppgaver for virksomheten (påvirker systemets nytteverdi).
B: Grensesnitt mot eksterne systemer.
C: Påvirkningen av arkitekturvalg og system- og databasedesign.
D: Kompleks funksjonalitet og/eller kompleks implementasjon. Vanskelig å estimere.

Vurdering fra en til tre, der tre viser den antatt største risikoen.

Usecase	**A**	**B**	**C**	**D**	**Sum**
UC01 Registrer salg	3	1	3	3	**10**
UC02 Registrere betaling	3	3	1	3	**10**
UC03 Søk på vare	3	1	2	1	**7**
UC04 Søk på kunde	2	1	2	1	**6**
UC05 Registrer kunde	3	1	1	1	**6**
UC06 Registrer varegruppe	2	1	1	1	**5**
UC07 Registrer leverandør	2	1	1	1	**5**
UC08 Registrer/tilpass vare	3	1	1	1	**6**
UC09 Registrer bedriftsinterne data	2	1	1	1	**5**
UC10 Registrere/endre rabatter	3	1	2	2	**8**
UC11 Registrere/endre kalkulasjonsregler	3	1	2	3	**9**
UC12 Oppdater økonomisystem/fakturasystem	2	3	3	3	**11**
UC13 Registrer/endre avtale	1	1	1	2	**5**
UC14 Registrer tilbud	1	1	1	2	**5**
UC15 Registrere Returvare	2	1	1	2	**6**
UC16 Skriv/eksporter prislister	2	1	1	2	**6**
UC17 Generer rapport	3	1	2	2	**8**
UC18 oppdater prisinformasjon	3	1	2	3	**9**
UC19 Overføre tilbud til salg	1	1	1	2	**5**
UC20 Importer fra NOBB	2	3	3	3	**11**
UC21 Generelt søk	2	1	2	2	**7**
UC22 Registrer kundetype	2	1	1	1	**5**
UC23 Registrer/oppdater pakning	3	1	1	2	**7**
UC24 Lag prisetikett	2	1	2	2	**7**
UC25 Skriv prisetikett	2	1	2	1	**6**

2.3.3 Hvordan vi har gått frem for å spesifisere kravene

Vårt grunnlag
Vår oppdragsgiver har utarbeidet et notat om forretningsideen som ligger til grunn for dette systemet. Der beskrives i grove trekk hvilke krav som stilles til systemet. Dette notatet benyttet vi som 'idebank' under analysen av kravene. I tillegg har samtaler og diskusjon med oppdragsgiver vært til hjelp for å klarlegge systemkravene i detalj.

Modell
Ved at vi har benyttet oss av prinsippene i UP for prosjektet, er fremgangsmåten for analysen og kravspesifiseringen i stor grad lagt av utviklingsmodellen.

UP bygger på objektorienterte prinsipper.
Usecase og diverse supplementær spesifikasjon benyttes for å avdekke krav, dette er ikke-teknisk dokumentasjon som i høyeste grad er forståelig for folk uten inngående datakunnskap.
Ideen er å tidlig få en grov oversikt over helheten, ikke detaljer. Populært kalt 'mile wide, inch deep'.

De krav som avdekkes i de tidlige iterasjoner, klarlegges i detalj senere.
Det er også enkelt å endre krav, da usecasene ikke blir designet og implementert før de er fullt og helt klarlagt. Dette er etter vår mening et vesentlig poeng, da krav sjelden er statiske under en utviklingsprosess.
Usecase som er skrevet i detalj blir deretter designet, for så å bli implementert og testet. Dette betyr i praksis at enkelte deler av et system er ferdig implementert før andre deler er ferdig analysert.

På grunnlag av dette kan en så analysere de enkelte deler for å avdekke risikomomenter. Poenget er så å ta tak i de risikable deler tidlig, for å unngå at disse kommer som et kjempeproblem sent i prosjektet.
Målet med dette er å gjøre hele utviklingsprosessen mer kontrollerbar, og enklere å estimere. Det gir muligheten for i verste fall å avbryte prosjektet mens 'leken er god'.

Ved hjelp av disse prinsipper har vi kommet frem til kravspesifikasjonen i prosjektet, en spesifikasjon som faktisk har endret seg gjentatte ganger gjennom utviklingsløpet.

3. DESIGN

3.1 Generelt om design i prosjektet

3.1.1 Innledning

Kravspesifiseringsprosessen klargjør ved hjelp av usecasene og annen supplementær spesifikasjon hva systemet skal gjøre, med ikke hvordan det skal utføres av systemet.

Det er dette som skal avklares i designprosessen.
De valgene vi gjør her på eksempelvis designklasser , skal senere 'mappes' til softwareklasser og implementeres i løsningen.

Elementer som inngår i designprosessen i vårt prosjekt er GUI-design, designklasser med tilhørende interaksjonsdiagrammer, systemarkitektur, samt en omfattende databasedesign.

3.1.2 Om dette kapitlet

Utover i kapitlet finner du litt om systemarkitekturarkitektur, eksempler på designklasser og interaksjonsdiagram. For fullstendig oversikt over arkitekturmodeller, designklasser og interaksjonsdiagrammer, se *VEDLEGG H*.

Videre følger noen momenter fra databasedesignen.
For fullstendig databasespesifikasjoner, se *VEDLEGG I*.

Vi viser eksempler fra GUI-design, og tilslutt nevner vi alternative løsninger vi har vurdert.

3.2 Systemarkitektur

3.2.1 Arkitekturen i ByggPOS

Databasen skal ligge på en sentral databasesserver. Denne og kasseklienter bør ha muligheter for nødstrøm. Dette for å sikre data, samt å hindre at butikken ikke kan selge varer pga at kassen ikke fungerer i tilfelle strømbrudd. Det er prinsipielt ikke noe i veien for at databasesserver og kasseklient kan være samme maskin, dette gjør at systemet er fleksibelt.

Vi velger å bruke 'tykke klienter' i denne applikasjonen. Presentasjonslaget og store deler av domene/applikasjonslaget vil bli lagt på klientene. Men det vil også bli litt applikasjonslogikk på server, blant annet for å håndtere grensesnittet mot eksterne systemer. Denne delen er også tenkt innbygget i kontorklienten.

Vi får altså på sett og vis et delt applikasjonslag. Systemet er delt i to 'moduler', tre hvis vi tar med den delen som håndterer eksterne systemer.
Hver og en av disse modulene har sin egen applikasjonslogikk, og fungerer helt uavhengig av hverandre.

Databasen vil få ansvar for å ivareta noe av applikasjonslogikken, eksempelvis å sørge for at alltid er samsvar mellom priser i systemet og priser ute i butikk.

Selv om denne løsningen med et sammenslått domene/applikasjonslag og såpass tykke klienter vil medføre en del ekstraarbeid ved eventuelle oppgraderinger.

I og med at det er lagt opp til at systemet skal kjøres standard pc'er som klienter, kan vi like godt utnytte den kapasiteten de har i stedet for eksempelvis å legge kun brukergrensesnittet på klient.

Vi er klar over at denne løsningen kanskje ikke pr. i dag er optimal hvis systemet eventuelt i fremtiden skal benyttes på trådløse enheter (PDA'er), men da dette ikke har vært tema så langt, velger vi å forsvare vårt valg.

Løsningen benytter seg av BDE (Borland Database Engine) i grensesnittet mellom database og applikasjon. Det gjør applikasjonen fleksibel i forhold til valg av databaseplattform.

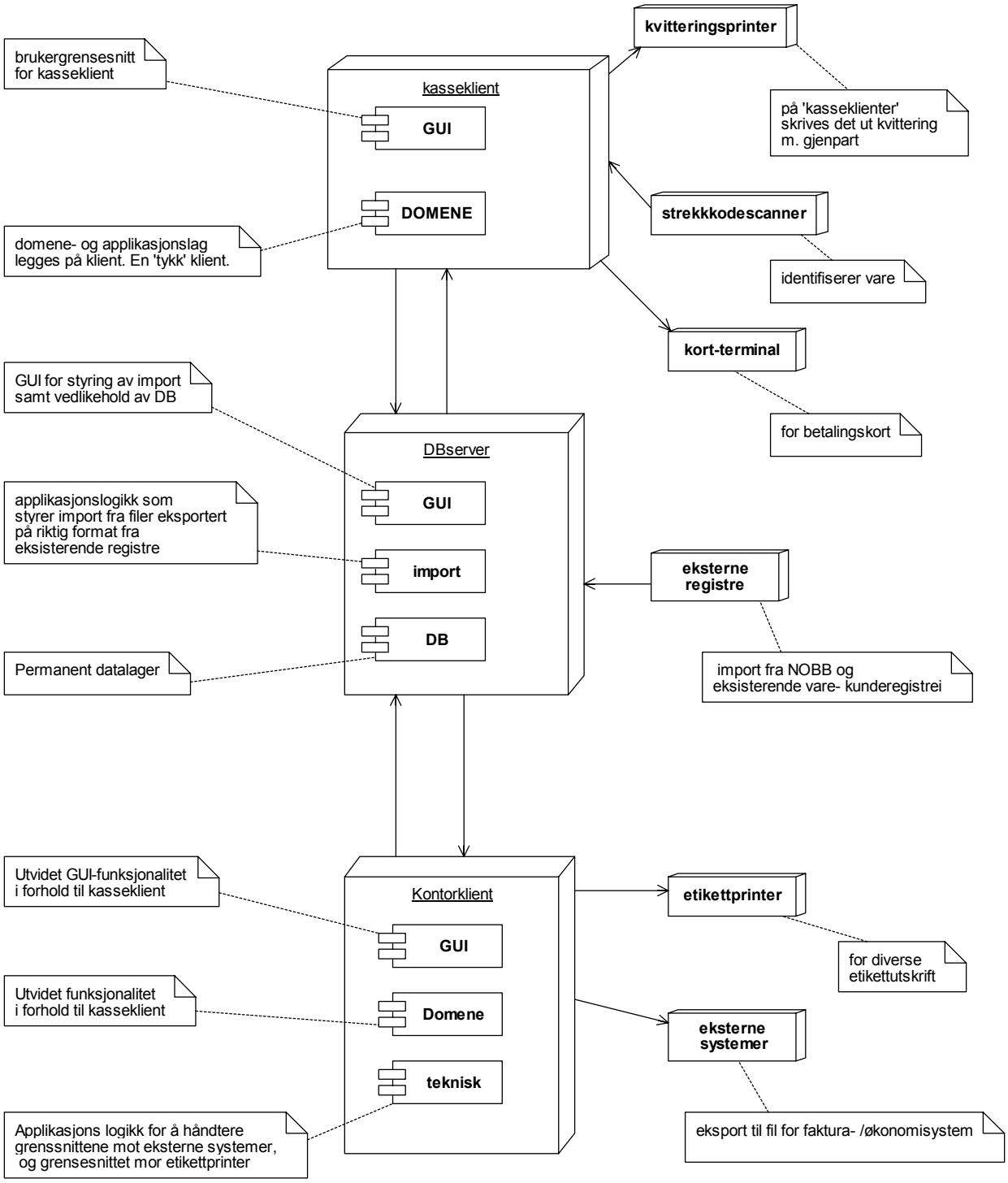
3.2.2 Arkitekturmodeller

Formålet med arkitekturmodellene er å illustrere kontrollstrukturen i utvalgte deler av systemet, den logiske oppbygningen av løsningen, samt funksjonalitet i- og samspillet mellom de enkelte deler av systemet.

Deploymentview Bygg-POS

Modellen (figur 3-1) viser hvordan lagdelingen i praksis vil bli i systemet. Presentasjonslaget ligger på klientene. Store deler av applikasjonslaget ligger også på klientene. Men noe forretingslogikk er plassert på databasen. Kontorklienten har støtte for kvitteringsprinter, strekkodeskanner og betalingsterminal.

På databaseserver har vi permanent datalager, med noe forretingslogikk bygd inn i databasen. Det er også tenkt en liten applikasjon her, som håndtererer import fra eksisterende registre.
Denne vil primært bli benyttet ved igangsetting av systemet for å opprette datagrunnlaget.



FIGUR 3-1: DEPLOYMENTVIEW BYGG-POS

3.3 Objektdesign

Denne delen av designprosessen består av å utvikle designklassediagrammer med tilhørende interaksjonsdiagrammer.

Vi velger å kun vise eksempler her, for fullstendig designdokumentasjon med modeller og beskrivelser, se *VEDLEGG H*.

Designklasser

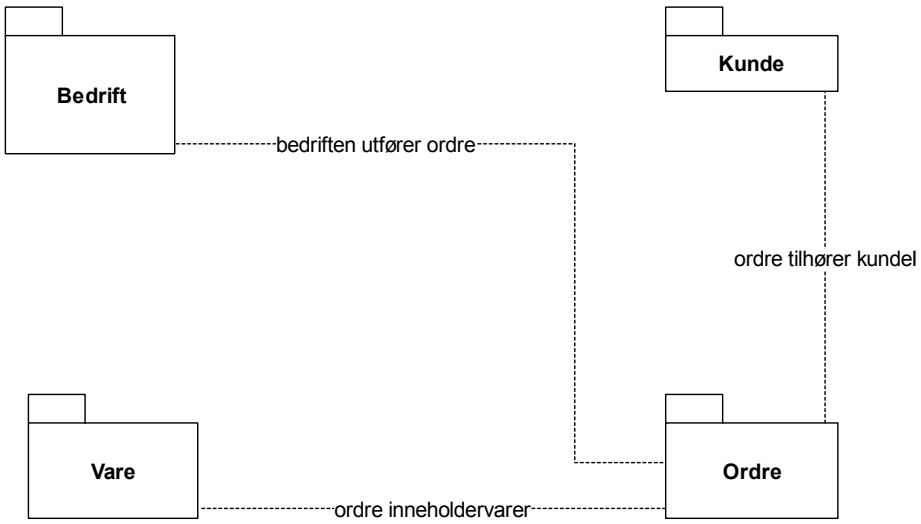
Designklassediagrammet skal vise de klasser som senere skal implementeres som softwareklasser i systemet, men det er ikke nødvendigvis noen eksakt likhet mellom en designklasse og den senere implementerte klassen.

Det er sjelden noen eksakt match, men det bør være mulig å kjenne seg igjen. Designklassene skal vise de viktigste attributter og funksjoner som et objekt av klassen har. Diagrammet skal også vise relasjonene mellom de forskjellige klassene.

Designklassediagrammet har sitt utspring i det konseptuelle klassediagrammet som ble til under kravspesifiseringsprosessen (se domenemodell, figur 2-3).

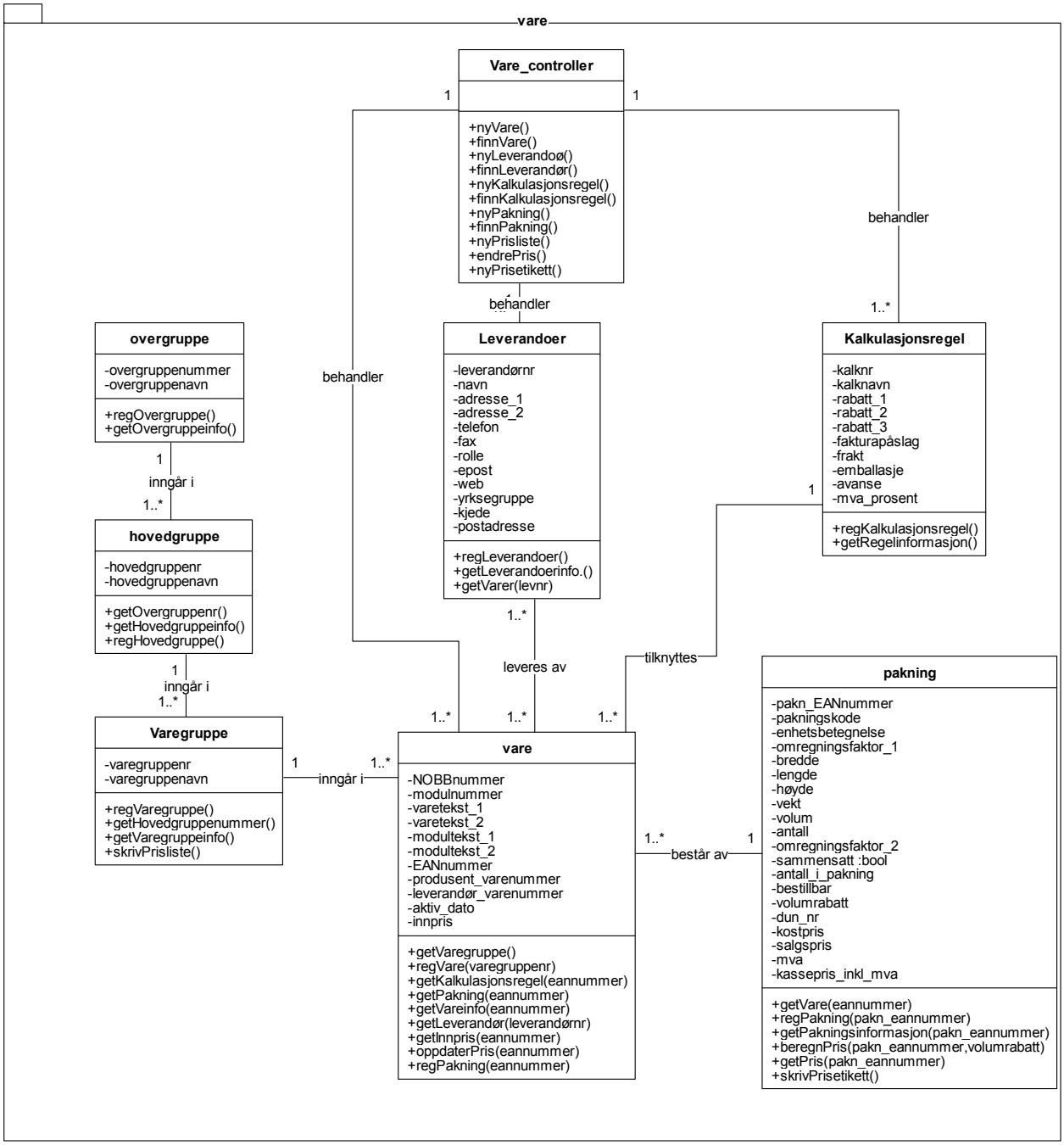
Bruk av 'pakker'

Vi har valgt å dele designklassene våre inn i fire 'pakker' som hver består av klasser som hører logisk sammen.



FIGUR 3-2: PAKKEDELING AV DESIGNKLASSER

Modellen (figur 3-3) viser pakken vare med de klasser som inngår i den, samt relasjoner mellom klassene. Det vil også være relasjoner mellom pakkene, dvs at et objekt av en klasse i eksempelvis pakken ordre vil kommunisere med et objekt av en av klassene i denne pakken.



FIGUR 3-3: PAKKEN VARE MED ALLE TILHØRENDE KLASSER

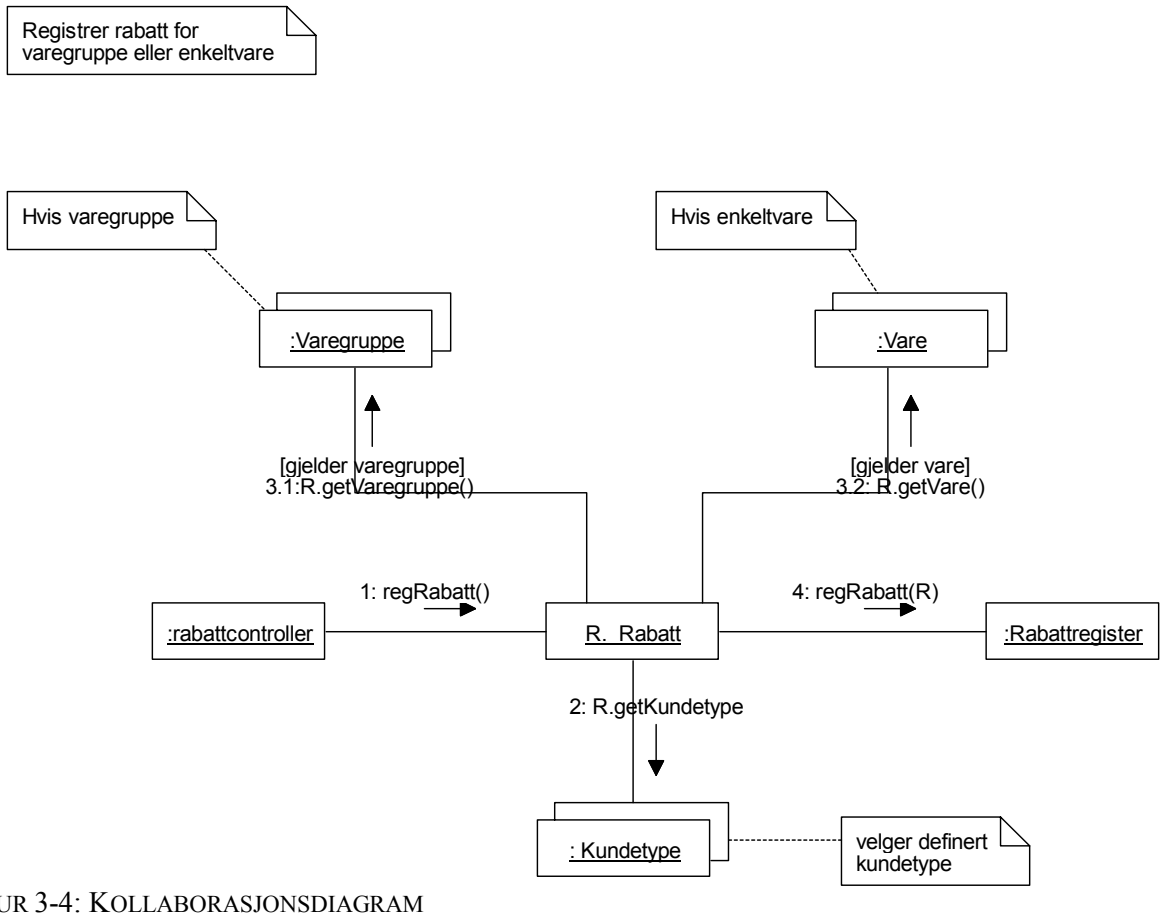
Interaksjonsdiagrammer

Interaksjonsdiagrammer kan lages både som sekvensdiagram og kollaborasjonsdiagram, de illustrerer begge det samme poenget.

Vi har benyttet oss av sistnevnte, rett og slett for å spare plass på grunn av at vi har mange objekter involvert i enkelte diagrammer.

Hensikten med interaksjonsdiagrammer er å vise hvordan objektinstanser av klassene kommuniserer med meldinger for å gjennomføre de krav (usecase) som er definert under kravspesifiseringen.

Kollaborasjonsdiagram for UC10 Registrere/endre rabatter
(ref. Vedlegg G)



FIGUR 3-4: KOLLABORASJONSDIAGRAM

3.4 Databasedesign

Innledning
Databasen er designet for å kunne benytte varegruppe- og vareinformasjon eksportert fra byggevarebasen NOBB.

I tillegg er har den tabeller som er nødvendige for permanent lagring av alle data som er nødvendige for å oppfylle kravspesifikasjonen for systemet.
Med disse tabellene er alle nødvendige grunndata på plass.

Utover dette, er databasen designet med et utvalg prosedyrer og triggere. Triggerne skal sørge for at eksempelvis alle ordrelinjer tilhørende en ordre slettes hvis ordren selv slettes. Dette er viktig for å beholde integriteten i systemet.

I et flerbrukermiljø der flere brukere samtidig jobber mot samme database, er det også viktig at databasen takler problematikken med samtidighetskontroll, eksempelvis ved tildeling av unike sekvensielle ordrenummer. Dette ivaretas av prosedyrer på databasen.

Applikasjonen er designet med rutiner for prisbehandling for en samling varer som velges på en av flere mulige måter. Prisbehandlingen skal i utgangspunktet gjøres kontrollert, det vil si at brukeren må velge beregningsmåte og så starte prisberegningen.

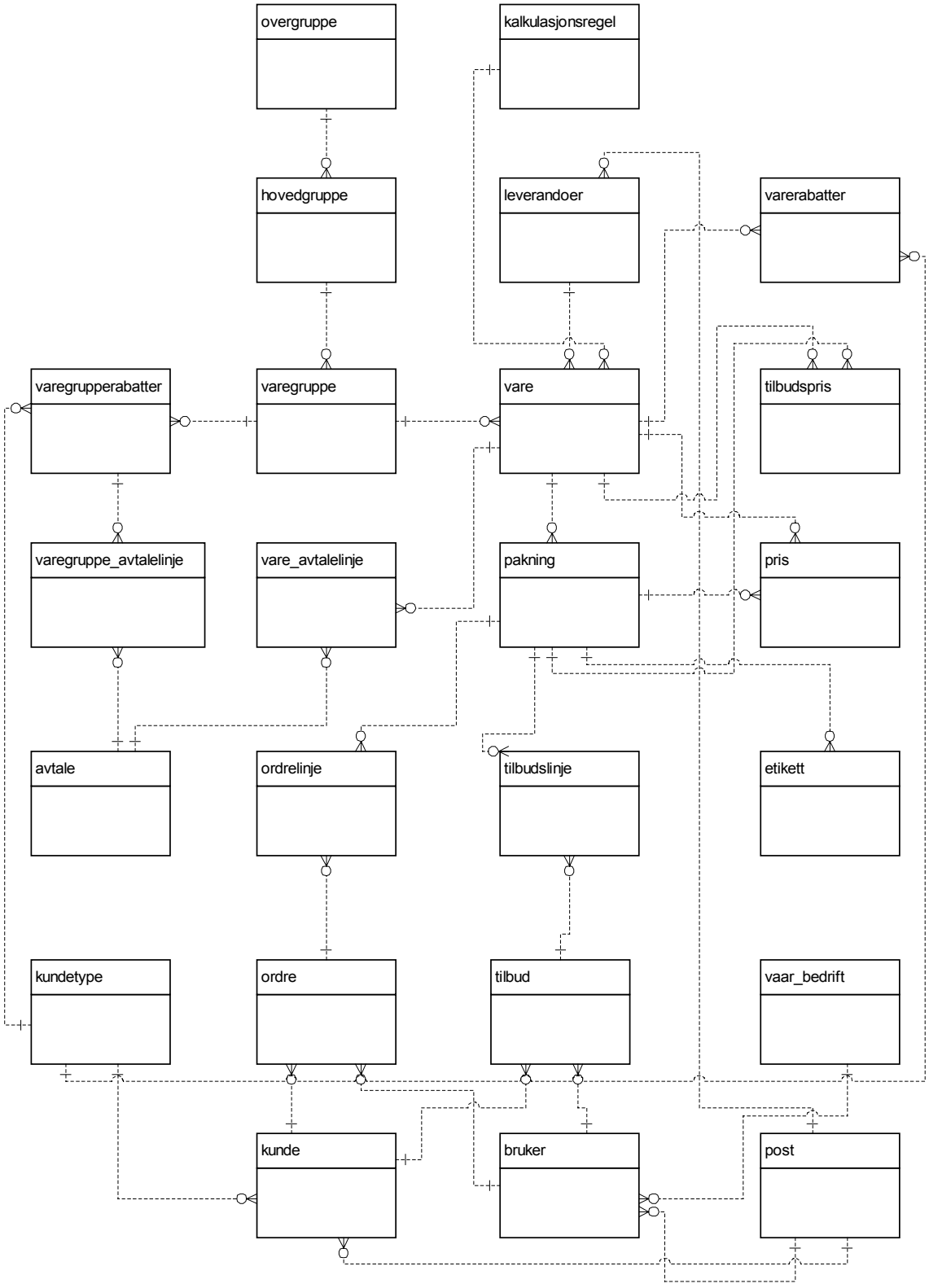
Den prisen som er registrert for en vareenhet, er kun en veiledende pris. Denne fungerer som input til kalkulasjonsregelen som gjelder for varen. Prisene som er interessante, er den reelle prisen som gjelder for en pakning av varen. Det er disse prisene som blir beregnet.

Prinsippet er databasen selv sørger for å gjøre dette. Det gjøres ved hjelp av lagrede prosedyrer på databasen. Applikasjonen gjør utvalget, og sender parametere til prosedyren på databasen. Deretter gjør databasen jobben.

Ved alle endringer gjort av applikasjonen som påvirker en vares pris, vil databasen automatisk takle dette og beregne den nye prisen for varens registrerte pakninger.

Se *VEDLEGG I* for fullstendig beskrivelse av databasen med prosedyrer og triggere.

ER- modell



FIGUR 3-5: ER- MODELL FOR DATABASEN

3.5 GUI-design

Innledning

Det er et viktig designmål for systemet at det skal være brukervennlig, og det har vi lagt mye vekt på.

Hva som kan defineres som brukervennlig, har neppe noe fasitsvar. Men det er et krav til systemet at det skal kjøres på Windows-plattformen, da er det et viktig poeng at brukergrensesnittet er basert på elementer som er kjent fra andre Windows-programmer, som menyer, ikoner og knapper.
Det er viktig å utnytte allerede eksisterende kunnskap hos brukerne.

Vi har fokusert på at systemet skal være lett å lære, og enkelt å huske. Derfor har vi basert oss på at ett skjermbilde forholder seg til en bestemt ting, ikke allverdens funksjonalitet i ett og samme bilde.
All 'grunndatabehandling' foregår i bilder bygd opp på samme måte, har du skjønt ett bilde vil du enkelt kunne sette deg inn i andre. Alle disse skjermbildene har de samme knapperekkene. Systemet kan betjenes enten ved bruk av mus, eller bruk av hurtigtaster. Alle knapper er tilknyttet hurtigtast. F1 er standard for hjelp i windows, slik er det også her.

Kassebildet i kassemodulen, samt ordreregistreringsbildet i kontormodulen fraviker fra disse generelle bildene. Disse bildene er også enkelt og oversiktlig oppbygd, slik at de er intuitive å lette og lære og huske. Vi har lagt vekt på at kassebildet skal være raskt å betjene uten bruk av mus, da bruk av mus er lite effektivt i en slik situasjon.

Vi har valgt å benytte noe tab-sheets, men kun i begrenset grad. Det vil ikke være slik at tabber 'hopper rundt', som er et kjent problem fra enkelte programmer. Alle elementer i et bilde er også plassert slik at det går klart frem hvilken tab de tilhører.

Vi besluttet tidlig at brukergrensesnittet skulle være SDI, altså at et vindu er åpent om gangen. Vi mente det ville gjøre systemet ryddig og oversiktlig.
Om dette var riktig eller ei har vi etter hvert lurt på, men vi tok nå engang den avgjørelsen. Dette ble også støttet av oppdragsgiver.

Prototyping

Tidlig i prosjektet benyttet vi oss i stor grad av prototyping. Vi startet med papir og blyant, for deretter å designe et utkast for skjermbildene i Delphi. Etter hvert fikk vi klarlagt det vi mener er en fornuftig struktur for skjermbildene for all grunndatabehandling.

Vi anser prototyping for å være en meget god metode for å avdekke feil i- ,eller mangler ved krav. I aller ytterste fall er det faktisk ikke sikkert kunden vet hva han egentlig er ute etter før han får se den ferdige applikasjonen, derfor er denne teknikken nyttig. På denne måten kan vi spare oss for mye unødige endringer. Vi sikrer på et vis at vi jobber ut i fra de rette kravene.

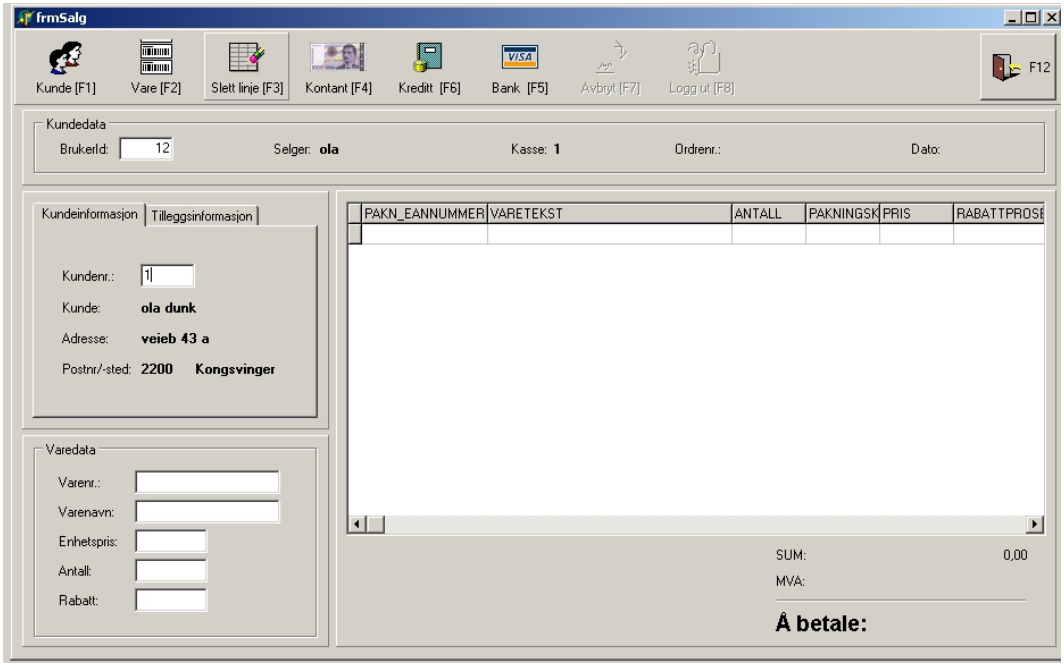
Oppdragsgiver har sett og vurdert de aller fleste prototypene, og kommet med synspunkter som er tatt til etterretning.

Vi har hele tiden benyttet oss av samme prinsippet ved design og implementering av nye deler av løsningen.

Eksempler

Da dette systemet inneholder ganske mange skjermbilder, anser vi det som meningsløst å vise alle her. Men noen få eksempler følger, bare for å illustrere prinsippene. For den fulle oversikt, må du prøve ut systemet.

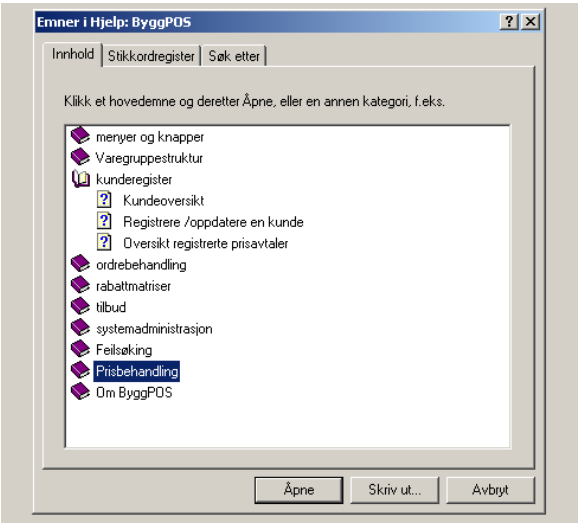
Kassebildet
Her vises skjermildet (figur 3-6) i kassemodulen som benyttes under gjennomføringen av salg. Her har bruker identifisert seg, og det er klart til å registrere linjer.



FIGUR 3-6: 'KASSEBILDET'

Hjelpefilen i ByggPOS

Her ser du innholdsoversikten for systemets hjelpefil (figur 3-7). Dette er en standard Windows hjelpefil som alle kan kjenne seg igjen i.



FIGUR 3-7: HJELPEFILEN I BYGGPOS

3.6 Ulike designalternativer vi har vurdert

Tykke kontra tynne klienter
I startfasen av prosjektet vurderte vi å designe systemet med tynne klienter, nærmere bestemt som en webløsning. Dette ble vurdert fordi vi mener det kunne vært det fordel å utvikle med tanke på plattformuavhengighet, selv om dette ikke er noe tema pr. i dag i følge kravspesifikasjonen. Systemet hadde da blitt designet med et fullstendig uavhengig brukergrensesnitt, som enkelt kunne byttes ut hvis ønskelig.

Dette ville i så fall endret vår systemarkitektur noe i forhold til den nåværende løsningen. Men dette alternativet ble fort avskrevet av oppdragsgiver, da han ville at systemet skulle implementeres som en standardapplikasjon ved hjelp av Borland Delphi (se kap. 4).
Dermed ble dette alternativet skrinlagt, og den nåværende løsningen ble påbegynt.

Lotus Notes
Vi fikk i utgangspunket noe feil opplysninger om NOBB fra oppdragsgiver, og dannet oss dermed et feilaktig inntrykk av hva dette faktisk er.

Men etter litt undersøkelser på egenhånd, fant vi ut at NOBB er basert på Lotus Notes. På bakgrunn av det, lekte vi en stund med tanken om å utnytte dette videre. Ideen var å utnytte NOBB-klienten, som allerede har all nødvendig vareinformasjon, for å utvide denne med nødvendig funksjonalitet for å oppfylle kravene.

Vi mener denne løsningen ville hatt en enorm fordel med tanke på oppdateringer i NOBB-systemet, på grunn av den replikeringsfunksjonaliteten som ligger i Notes. Den nåværende løsningen har en enorm utfordring på det området, hvis det skal kunne utføres på en enkel måte. Dette er det ikke tatt høyde for ennå, den nåværende løsningen er designet for å benytte NOBB-data kun for førstegangs opprettelse av datagrunnlaget.

Dette alternativet ble også avskrevet, da det ble klart at vi ikke fikk tilgang på NOBB. Også oppdragsgivers sterke ønske om at Delphi skulle benyttes, talte i mot dette.
Det finnes komponenter for Delphi som kan kommunisere med Notes, uten at vi i detalj har satt oss inn i hvordan de fungerer.

Databasen

Med unntak av den ideen vi hadde om bruk av Lotus Notes, har det hele tiden vært klart at vi må benytte oss av en seriøs relasjonsdatabase.

Spørsmålet har egentlig vært hva som skal utføres i applikasjonen, og hva som skal utføres i databasen. På grunn av flerbrukermiljøet er det naturlig å legge en del funksjonalitet på databasen, dette letter arbeidet med å sikre god samtidighetskontroll.

Men vi har også plassert funksjonalitet som ikke direkte påvirkes av samtidighetsproblematikk på databasen.
Et eksempel på dette er prosedyren som benyttes under salgsregistrering for registrering av ordrelinjer.

Et alternativ hadde vært å la applikasjonen gjøre en masse spørringer mot databasen for så å finne riktig pris selv, men siden dette er en operasjon som vil bli mye brukt mener vi det er riktig å la databasen utføre den.
Generelt er det å anbefale at operasjoner av denne type, som benyttes ofte, plasseres på databasen.

Grensesnittet applikasjon – database

Selv om det ikke er noe direkte krav, anser vi det som nyttig at grensesnittet mellom applikasjon og database er 'generelt', slik at applikasjonen ikke krever noen spesiell databaseplattform, men kan kjøres mot alternative relasjonsdatabaser.

3.7 Verktøystøtte for analyse og design

Vi har under arbeidet med kravspesifisering og design delvis benyttet oss av dataverktøy. For all tekstlig beskrivelse duger en generell tekstbehandler, vi har stort sett benyttet Microsoft Word. Til modeller for klasser og sekvensdiagrammer, samt kollaborasjonsdiagrammer, er Tau UML-suite brukt. Visio er benyttet for nettverksskisser og ER-modell for database, alle script for databasen er i stor grad skrevet 'for hånd', med unntak av de første utkast ble generert fra Modelator.
I tillegg er som nevnt Delphi benyttet for prototyping av GUI.

Vi har vært såpass dristige at vi har hatt et manuelt opplegg for versjonskontroll, det har faktisk fungert stort sett bra. Grunnen til dette er at vi valgte å ikke bruke ressurser på å sette oss inn i et program for dette på den for så vidt korte tiden vi har hatt til rådighet.

4. IMPLEMENTERING

4.1 Innledning

Tidlig i prosjektet fokuserte vi sterkt på databasedesign, med påfølgende implementering av første versjon. Vi så det som meget viktig å få implementert en database, om enn bare et utkast til det som skulle bli den endelige versjonen. Parallelt med dette pågikk prototyping for de første skjermbildene, samt analyse- og designdokumentasjon.

Så fort første versjon av databasen var implementert, startet implementeringen av de første deler av applikasjonen, der prototyper og designdokumenter var klare. Flere deler ble implementert parallelt og uavhengig av hverandre. Vi kan kanskje si det slik at vi har implementert systemet ved å dele det i mindre 'programenheter', å kalle det moduler er vel litt drøyt. Disse har kontinuerlig har blitt enhetstestet under implementeringen.

Etter hvert som prosjektet har skredet frem, har disse enhetene blitt integrert sammen i en stadig større applikasjon.
Dette prinsippet har vært fulgt gjennom hele prosjektet.

4.2 Valg av utviklingsverktøy

4.2.1 Om valg av utviklingsverktøy

I startfasen hadde vi Java, da med Borland Jbuilder, og Borland Delphi som reelle alternativer. Grunnen til at vi vurderte Java, er i hovedsak plattformuavhengigheten.

Men da vi la fram disse alternativene for oppdragsgiver, med argumenter for og imot begge, var han helt klar på at vi skulle benytte Delphi. Dermed var valget av språk og utviklingsverktøy avgjort.
Det er på ingen måte noen ulempe, da dette verktøyet har meget god støtte for databaseprogrammering.

På grunn av at vi benytter Delphi, vil det ikke være noen 'direkte mapping' mellom funksjonene i designdokumentene og implementerte funksjoner i systemet.

4.2.2 Komponenter, gjenbruk

Vi har benyttet oss av enkelte frie komponenter i tillegg til de som standard finnes i Delphi sitt bibliotek.
Dette gjelder en komponent for eksport til Microsoft Excel, da Delphi ikke har dette.
Applikasjonen har også en annen grid enn den Delphi har som standard. Grunnen til dette er at vi ønsket noe mer fleksibilitet, spesielt med tanke på utseende.
En fri komponent for import og eksport til CSV-filer er også testet ut, men denne er ikke benyttet, da disse delene ikke er implementert pr i dag.

Vi programmerte tidlig et interface mot serieport for bruk til strekkodeskanner. Denne delen gikk i en egen tråd med lav prioritet, altså flertrådsprogrammering. Dette ble testet ut og funnet ok, og lagt bort for å tas i bruk senere når salgsdelen for kassemodulen skulle implementeres.

Da dette ble gjort, og skanneren testet på forskjellige maskiner, viste det seg at den allikevel ikke fungerte tilfredsstillende. Uvisst av hvilken grunn. Flere har sett på koden, men finner ikke ut av problemet.

Derfor prøvde vi en demoversjon av en kommersiell komponent for bruk mot strekkodeskanner, og den fungerte perfekt. Vi så da ingen grunn til å slite mer med problemet, og gikk til innkjøp av denne [Delphi 4].
Denne har vi foreløpig ikke fått, derfor er demonversjonen implementert i applikasjonen pr. i dag.

4.3 Valg av databaseplattform

Et mål med dette systemet er at prisen skal være moderat, samtidig som vi må benytte oss av en seriøs relasjonsdatabase som har støtte for triggere og lagrede prosedyrer. Dette er i utgangspunktet to 'litt motstridende krav'.

Derfor falt valget på Borland Interbase 6.0. Den leveres i en fri og en lisensiert versjon. Vi har valgt å benytte den frie versjonen, da den tilfredsstiller de behov vi har til dette systemet.
Databaseløsningen vår baserer seg på standard SQL- dialect 3. Databasen vil dermed enkelt kunne implementeres på enhver databaseplattform som støtter denne standarden.

Men dette gjelder kun selve databasen, triggere og prosedyrer vil måtte omskrives for den aktuelle databaseplattformen, da det på dette området råder endel syntaksforskjeller. Dette er ingen stor jobb, og vil ganske enkelt kunne utføres hvis noen skulle ha behov for det.

4.4 grensesnitt applikasjon - database

Designmålet for dette er som nevnt å ha et generelt grensesnitt. Det har vi oppnådd ved hjelp av BDE(Borland Database Engine). Applikasjonen benytter seg av en generell databasekomponent, som kobles mot en database med navn POS.
Databasen POS defineres som et såkalt alias i BDE under kontrollpanelet på klientmaskinen. Dette aliaset definerer plasseringen av den fysiske databasen, samt typen databasedriver. Når dette aliaset er definert og ikke minst fungerer, kan applikasjonen startes, og systemet er i gang.

Vi har dermed oppnådd designmålet om et generelt grensesnitt.

4.5 Systemets hjelpefil

Når systemets hjelpefil skulle skrives, så vi behovet for verktøystøtte til dette. Da vi ikke hadde noe kommersielt verktøy tilgjengelig, valgte vi å benytte oss av en demoversjon av Helpscribble. (www.helpscribble.com)

4.6 Kodestandard

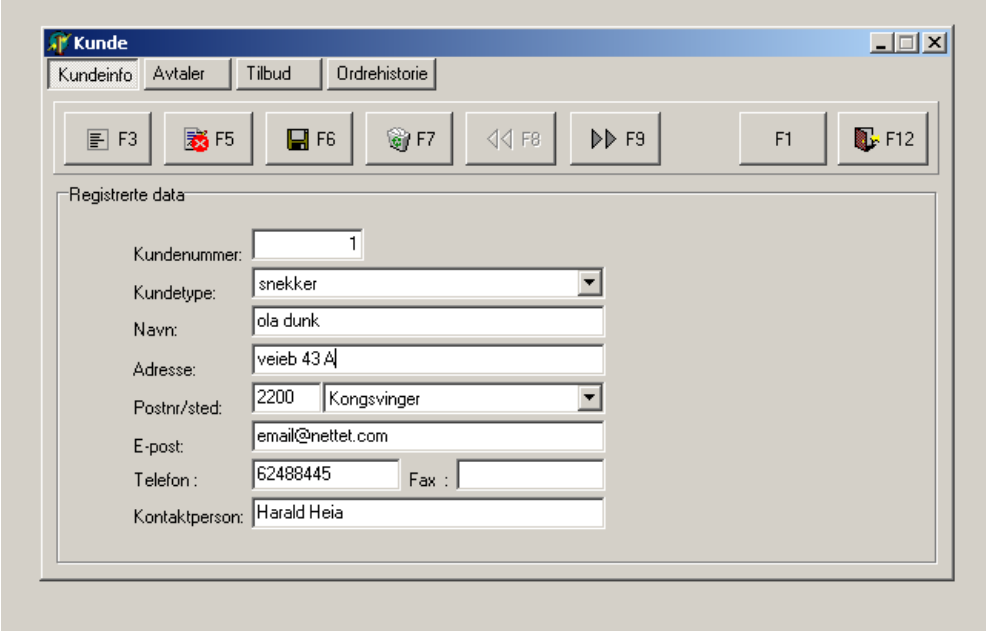
All implementert kode følger en mal for prefiksing av alle variable og komponenter, se *VEDLEGG L*.
Dette gjør koden blir lettere å forstå for programmereren selv og andre. Generelt er koden kommentert i den grad det føles nødvendig.

Alle kodefiler har heading etter følgende mal:

```
//-------------------------------------------------------------------------------------------
// navn:              filnavn.pas
// dato:              30.01.02
// versjon:           0.1
// oppdatert av:      Ola Nordmann
// beskrivelse:       hva filen gjør….
//                    ……..
//                    ……
//-------------------------------------------------------------------------------------------
```

Alle SQL- script for databasen er også versjonsnummerert liknende måte, og de inneholder beskrivelse over hva scriptet gjør, eventuelt og hva som er endret i forhold til tidligere versjon.

4.7 Eksempler på kode



FIGUR 4-1: SKJERMBILDE FOR KUNDEINFORMASJON

Utdrag fra koden som ligger bak figur 4-1

```
////////////////////////////////////////////////////////////////////////////
//                                                                        //
//    Fil:         unitSeKunde.pas                                        //
//    Dato:          15.05.02                                             //
//    Beskrivelse: Bilde for registrering av nye kunder og for å vise info.  //
//                 om en allerede registrert kunde og hans Avtaler, Tilbud   //
//                 og Ordrer.                                             //
//    Versjon: ByggPOS beta 1
//
////////////////////////////////////////////////////////////////////////////

unit unitSeKunde;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ComCtrls, ExtCtrls, ToolWin, DBxNav, Buttons, dbBoxGrd,
  StdCtrls, Mask, DBCtrls, rs, DBActns, ActnList, DB, DBTables;

type
  TfrmSeKunde = class(TForm)
    pctSeKunde: TPageControl;
    tshKundeinfo: TTabSheet;
    tshAvtaler: TTabSheet;
    tshTilbud: TTabSheet;
```

tshOrdrehistorie: TTabSheet;
gbxKundeinfo: TGroupBox;
lblNummer: TLabel;
lblNavn: TLabel;
lblAdresse: TLabel;
lblPost: TLabel;
lblKundetype: TLabel;
lblEmail: TLabel;
lblKontaktperson: TLabel;
lblTelefon: TLabel;
dbLcbKundetype: TDBLookupComboBox;
dbLcbPoststed: TDBLookupComboBox;
dbEdtNummer: TDBEdit;
dbEdtNavn: TDBEdit;
dbEdtAdresse: TDBEdit;
dbEdtPost: TDBEdit;
dbEdtEmail: TDBEdit;
dbEdtkonataktperson: TDBEdit;
dbEdtTelefon: TDBEdit;
dbEdtFax: TDBEdit;
gbxAvtaler: TGroupBox;
gbxTilbud: TGroupBox;
gbxOrdrehistorie: TGroupBox;
pnlKnapper: TPanel;
Panel1: TPanel;
Panel2: TPanel;
Panel3: TPanel;
sbtAvbryt: TSpeedButton;
sbtLagre: TSpeedButton;
sbtSlett: TSpeedButton;
sbtForrige: TSpeedButton;
sbtNeste: TSpeedButton;
sbtLukk: TSpeedButton;
ActionList: TActionList;
DataSetCancel: TDataSetCancel;
DataSetPost: TDataSetPost;
DataSetDelete: TDataSetDelete;
DataSetPrior: TDataSetPrior;
DataSetNext: TDataSetNext;
F12: TAction;
stpKundenr: TStoredProc;
sbtNy: TSpeedButton;
F3: TAction;
sbtVelgAvtale: TSpeedButton;
sbtVelgTilbud: TSpeedButton;
sbtVelgOrdre: TSpeedButton;
F2: TAction;
sptLukk: TSpeedButton;
dbGrdKundeAvtale: TDBGridPro;
dbgrdpKundeOrdre: TDBGridPro;
dbgrdpKundeTilbud: TDBGridPro;
SpeedButton1: TSpeedButton;
SpeedButton2: TSpeedButton;

```
    Label1: TLabel;
    SpeedButton3: TSpeedButton;
    F1: TAction;
    SpeedButton4: TSpeedButton;
    SpeedButton5: TSpeedButton;
    SpeedButton6: TSpeedButton;

    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure F12Execute(Sender: TObject);
    procedure DataSetPostExecute(Sender: TObject);
    procedure F3Execute(Sender: TObject);
    procedure F2Execute(Sender: TObject);
    procedure pctSeKundeChange(Sender: TObject);
    procedure dbGrdKundeAvtaleDblClick(Sender: TObject);
    procedure dbGrdKundeAvtaleKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure dbgrdpKundeTilbudDblClick(Sender: TObject);
    procedure dbgrdpKundeTilbudKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure dbgrdpKundeOrdreDblClick(Sender: TObject);
    procedure dbgrdpKundeOrdreKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure DataSetDeleteExecute(Sender: TObject);
    procedure DataSetPriorExecute(Sender: TObject);
    procedure DataSetNextExecute(Sender: TObject);
    procedure F1Execute(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmSeKunde: TfrmSeKunde;
  booKundeBilde: Boolean = False;  //Holder orden på hvilket bilde som skal vises
                                   //når SeKunde bildet lukkes.
  booKundeAvtale: Boolean = False; //Variebel som blir satt til true når Avtale
                                    //bildet skal vises fra kundebildet.
  booKundeTilbud: Boolean = False; //Variebel som blir satt til true når Tilbud
                                    //skal vises fra kundebildet.
  booKundeOrdre: Boolean = False; //Variebel som blir satt til true når Salgsbildet
                                    //skal vises fra kundebildet.

implementation

uses dmAdministrasjon, unitKunder,
UnitStartside,dmAvtaler,dmTilbud,unitSePrisavtale,
    unitSeTilbud,unitAvtalelinjer, frmSalgsbildet, dmOrdrer;

{$R *.dfm}

procedure TfrmSeKunde.FormClose(Sender: TObject; var Action: TCloseAction);
```

```
begin                                  //For at blanke poster ikke skal vises i gridden ved
  dmAdmin.qryKunde.Cancel;      //lukking av bildet.
  if booKundeBilde then         //Dersom SeKunde ble åpnet via Kundeoversikt
     frmKunder.Show;            //skal Kundeoversiketen vises når bildet lukkes
  Action := caFree;
end;

procedure TfrmSeKunde.F12Execute(Sender: TObject);
begin
  dmAdmin.qryKunde.Close;     //Refresh.
  dmAdmin.qryKunde.Open;
  frmSeKunde.Close;
end;

procedure TfrmSeKunde.F3Execute(Sender: TObject);
begin
   frmKunder.NyKunde;
end;
```

Eksempel på databasescript

Dette scriptet genererer den prosedyren som brukes for å finne riktig pris for en pakning(salgsenhet) på grunnlag av kundenummer

```
//****************************************************************

// prosedyre som lager ordrelinjer
// og FINNER PRIS for en kunde for en gitt pakning
// returnerer rabattprosent hvis ikke kampanjetilbud blir
billigere
// prisen som returneres er pakningens standardpris, rabatten må
regnes inn // i kode
// mva må også regnes ut og legges til
// skrevet av: Kjetil Ophus
// versjon 0.2 (byggpos beta 1.0)
//****************************************************************


set term^;
create procedure reg_ordre(pakn_nummer Varchar(13), kundenr
Integer)

RETURNS (pakn_eannummer Varchar(13), rabatt Float, pris Float,
          varetekst Varchar(35),pristekst Varchar(30), paknkode
char(3), mva Float)

as

declare variable kundetype Integer;
declare variable eannummer Varchar(13);
declare variable varegruppenr Integer;
declare variable varegr_rab Float;
declare variable vare_rab Float;
declare variable varegr_avt Float;
declare variable vare_avt Float;
declare variable temp_pris Float;
declare variable kamp_pris Float;
declare variable rabattprosent Float;



begin
 rabattprosent=0;

 select distinct mva_prosent from kalkulasjonsregel into :mva ;


 select pakn_eannummer,eannummer, salgspris ,varetekst1,
 pakningskode,     varegruppenummer, tekst
   from pakning, vare, pris where
pakning.pakn_eannummer=:pakn_nummer
      and vare.eannummer=pakning.eannummer
      and pris.pakn_eannummer=:pakn_nummer
      and pris.fom_dato<'today' and pris.tom_dato>'today'
      into :pakn_eannummer,:eannummer, :pris, :varetekst,
      :paknkode,   :varegruppenr, :pristekst ;
```

```
 select kundetype from kunde where kunde.kundenummer=:kundenr
      into :kundetype ;

 select rabattsats from varegrupperabatter where
kundetype=:kundetype
   and varegruppenummer=:varegruppenr into :varegr_rab ;
  if(:varegr_rab is not null) then begin
   rabattprosent=:varegr_rab;
  end


  select rabattsats from varerabatter where kundetype=:kundetype
     and eannummer=:eannummer into :vare_rab ;
  if(:vare_rab is not null and :vare_rab>:rabattprosent ) then
begin
   rabattprosent=:vare_rab;
  end


  select rabattsats from varegruppe_avtalelinje v, avtale a where
  a.kundenummer=:kundenr
      and a.fradato<'today' and a.tildato>'today'
      and v.avtalenummer=a.avtalenummer
      and v.varegruppenummer=:varegruppenr  into :varegr_avt ;
  if(:varegr_avt is not null and :varegr_avt>:rabattprosent ) then
begin
   rabattprosent=:varegr_avt;
  end


  select rabattsats from vare_avtalelinje v, avtale a where
  a.kundenummer=:kundenr
   and a.fradato<'today' and a.tildato>'today'
   and v.avtalenummer=a.avtalenummer and v.eannummer=:eannummer
   into :vare_avt ;
  if(:vare_avt is not null and :vare_avt>:rabattprosent ) then
begin
   rabattprosent=:vare_avt;
  end


  rabatt=:rabattprosent;

  temp_pris=:pris-(:pris*(:rabattprosent/100));


  select salgspris from tilbudspris where
pakn_eannummer=:pakn_nummer
       and fom_dato<'today' and tom_dato>'today' into :kamp_pris ;
  if(:kamp_pris is not null and :kamp_pris<:temp_pris ) then begin
   temp_pris=:kamp_pris ;
   pristekst='tilbudspris';
   pris=:kamp_pris;
   rabatt=0;

  end

suspend;
end^
set term;^
```

4.8 Versjonskontroll

Vi har som nevnt valgt å gjennomføre prosjektet med manuelle rutiner for versjonskontroll. Prinsipielt er nok det en dårlig løsning.
Men da ingen av oss hadde noen erfaring med noe verktøy for støtte til dette, antok vi at det ville 'koste mer enn det smakte' at alle skulle sette seg inn i et slikt verktøy på den forholdsvis korte tiden dette prosjektet varer.

Vi har med våre manuelle rutiner greid oss rimelig bra, men ser helt klart behovet for verktøystøtte. Vi må innrømme at vi har opplevd problemer som antakelig kunne vært unngått med verktøystøtte.
Men uten gode rutiner hjelper heller ikke verktøystøtte stort.
Så vi har i alle fall fått erfaringer både på godt og vondt om versjonskontroll som er gode å ha med seg videre.

5. TESTING og kvalitetssikring

5.1 Om testing og kvalitetssikring

Hensikten med testing er å finne og fjerne så mange feil som mulig i før et programmet tas i bruk.
Målet er ikke å vise at programmet er feilfritt, det er i praksis umulig å gjøre gjennom testing. Gjennomføring av tester kan bare avdekke feil, vi kan ikke garantere at programmet ikke inneholder feil.

Testing er et nødvendig middel for å sikre kvaliteten på programsystemet. Men for å oppnå kvalitet, må dette målrettet og bevisst bygges inn i programmet gjennom grundig arbeid med kravspesifisering, analyse, design og implementasjon.

For å kunne utføre tester, er det et krav at systemet som skal testes kan kjøres. Det betyr at testingen av programmet først kan begynne etter at man har begynt å implementere. Da kan allerede mange feil ha kommet inn i løsningen gjennom analyse- og designfasen. For å avdekke feil tidlig i prosessen, kan revisjoner og inspeksjoner være et godt alternativ.

Aktuelle typer tester
Det finnes mange forskjellige typer tester:
Enhetstest, integrasjonstest, systemtest, akseptansetest, regresjonstest, ytelsestester pluss flere andre.

Enhetstester kan være test for en prosedyre eller klasse, men også for en modul eller en gruppe moduler.
Disse testene utføres etter black- eller whitebox-prinsippet. Blackbox betyr at du ikke har innsyn, du sender testdata inn og får resultat ut.
Whiteboxprinsippet bygger på fullt innsyn i enheten.

Testene planlegges og gjennomføres etter bestemte strategier, eksempelvis 'top down'. Prinsippet her er at vi begynner med å teste de 'øverste' delene av systemet, for så å fortsette nedover. Det motsatte kalles 'bottom up'. Det er også mulig å benytte seg av en blanding av disse prinsippene, såkalt kombinasjon.

Programdeler som er enhetstestet, integreres gjerne med hverandre i det mer og mer komplette systemet. Testingen i denne forbindelsen går ut på å konstantere at systemet fungerer i henhold til spesifikasjonene.

Disse testene planlegges og gjennomføres også etter top-down-, bottom-up- eller kombinasjons-prinsippet.

Regresjonstester er aktuelt når det er gjort endringer i systemet. Prinsippet er at tester som tidligere er kjørt og akseptert, kjøres på nytt for å avdekke om endringene har ført med seg nye feil inn i løsningen

Systemtester innebærer en rekke forskjellige tester, men den eneste vi har berørt i prosjektet, er funksjonstesten.
Disse testene bør utføres i et miljø som ligner mest mulig på den situasjonen som systemet vil møte når det er i operativ drift.

Generelt gjennomfører vi tester i følgende tre trinn:
- planlegging av testen
- Vurdere seg frem til riktige testdata
- gjennomføre testen og evaluere resultatet

5.2 Våre teststrategier

Designfase
Vi har som tidligere nevnt benyttet oss mye av prototyping. Vi tillater oss å si at dette har en effekt i forhold til testing. På denne måten kan vi bedre få avdekket og klargjort krav. Særlig i tidlige faser er dette viktig, der har vi har i utstrakt grad benyttet prototyping.
I tillegg har vi gjennomført enkelte revisjoner av mer uformell art. Vårt mål med disse revisjonene er i hovedsak å avdekke om de funksjonelle kravene ivaretas. Vi har fokusert sterkt på å hele tiden jobbe videre med de riktige krav som utgangspunkt.

Databasemodellene har vi evaluert ved hjelp av uformelle revisjoner, for å avklare om modellen legger opp til at de funksjonelle krav i systemet kan ivaretas.

Implementering
Etter hvert som database og de første deler av applikasjonen ble implementert, kom andre teststrategier inn i bildet.
Enhetstesting har vært planlagt og utført kontinuerlig, etter hvert som systemet har blitt implementert.

Med enheter mener vi alt fra funksjoner og prosedyrer til et ferdig skjermbilde med all sin funksjonalitet.

Enhetstestene for applikasjonen har vi stort sett utført etter bottom-up prinsippet. Både black- og whiteboxprisippet er benyttet.
Vi anser blackbox som greit, så der dette fungerer har vi benyttet dette prinsippet. Ved slik testing er det viktig å finne de rette testdata, slik at vi har reell mulighet for å påvise eventuelle feil.

Når en enhet er testet og funnet ok, så har vi integrert enheten med de eksisterende enhetene i applikasjonen. Her kommer integrasjonstestene våre inn i bildet. Etter integrasjon har vi testet at all datautveksling mellom enhetene fungerer som forutsatt, deretter har vi for sikkerhets skyld kjørt regresjonstest på de enheter som påvirkes av akkurat denne integreringen.

Vi har lagt opp til og kontinuerlig gjennomført funksjonalitetstester på alle deler av systemet ettersom de har blitt implementert. Dette for å sikre at de utfører den funksjonaliteten de skal på riktig måte. Spesielt er det testet at alt fungerer i forhold til databasen med tanke på lagring, oppdatering og sletting.

Database

Databasen er kontinuerlig testet generelt med tanke på integritet og datatyper.

For prosedyrer og triggere på databasen har vi valgt å teste utelukkende etter blackbox-prinsippet. Alle prosedyrer er først testet direkte i databasen, deretter er de testkjørt fra applikasjon.

På grunn av at det er triggere inne i bildet, er integrasjonstester med påfølgende regresjonstesting utrolig viktig.
Da triggere kjøres automatisk, kan det fort medføre uønskede resultater.

Endelige systemtester

Systemtester har vi ikke lagt noen stor vekt på, med unntak av funksjonalitetstestene vi har utført enkeltvis på de enkelte enhetene.

De forskjellige systemtestene, eksempelvis endelig funksjonalitetstest, stresstest og volumtest, bør utføres mot et ferdig system som opererer i et tilnærmet 'normalt' miljø. Disse testene er også ganske tidkrevende å gjennomføre.

Da systemet ikke er ferdig implementert i henhold til kravspesifikasjonen, anser vi det som lite hensiktsmessig å bruke ressurser på disse testene enda. Det er naturlig at systemtestene utføres på et senere tidspunkt, da hele løsningen er ferdig implementert.

5.3 gjennomføring av testene

Vi viser her noen eksempler for å illustrere prisnippene for gjennomføringen av testene.

Noen fullstendig testrapport har vi ikke utarbeidet ennå, men vi har 'maler' som er benyttet under planleggingen og gjennomføringen av testene.

5.3.1 Enhetstesting

Dette eksemplet illustrerer planlegging og gjennomføring av test av en prosedyre lagret på databasen som skal benyttes til å finne riktig rabattprosent på grunnlag av en pakning og et kundenummer.

Det er viktig å definere riktige testdata, slik at testen går innom alle mulige veier i prosedyren minst en gang.
Her er det en rabattprosent som blir returnert fra prosedyren vi ønsker å sjekke. For å finne denne rabatten er det tre mulige veier å gå inne i prosedyren.

På grunnlag av det setter vi opp testklasse på grunnlag av data som vi vet er registrert i databasen, vi må systematisk vurdere oss frem til dette slik at vi får testet alle mulige veier.

Planlegging og vurdering av testdata
Hovedprinsippet for denne prosedyren er at den først finner kundens kundetype på grunnlag av kundenummeret. Så finner den eventuell rabatt på varen som er i pakningen ut i fra vare eller varegruppe på grunnlag av kundetypen.

Deretter sjekker den eventuell rabatt på grunnlag av kundenummeret satt på enten vare eller varegruppe. Hvis denne er høyere enn den forrige rabatten, er det denne som gjelder. Hvis ikke, er det den forrige som fortsatt gjelder. Deretter sjekkes det om det pr i dag er kampanjetilbud på varen. Hvis denne prisen er lavere enn veiledende pris med funnet rabatt innregnet, er det tilbudsprisen som gjelder. Da skal ikke rabatt regnes inn.

På grunnlag av dette kan vi sette opp en testklasse som skal teste alle muligheter:

nr	Pakn_eannummer	kundenummer	Resultat	Forventet resultat
1	8699444466743	2	0	0
2	7855444333341	2	18	18
3	7855444333990	2	12	12
4	7855444333341	1	0	0
5	8699444466743	1	15	15
6	7855444333990	1	25	25

Gjennomføring og evaluering
På grunnlag av testklassen kan vi nå gjennomføre testen. Hvis resultat svarer til forventet resultat kan vi anta at prosedyren er god nok.
Hvis vi derimot ikke får forventet resultat, må vi inn å finne feil og rette.

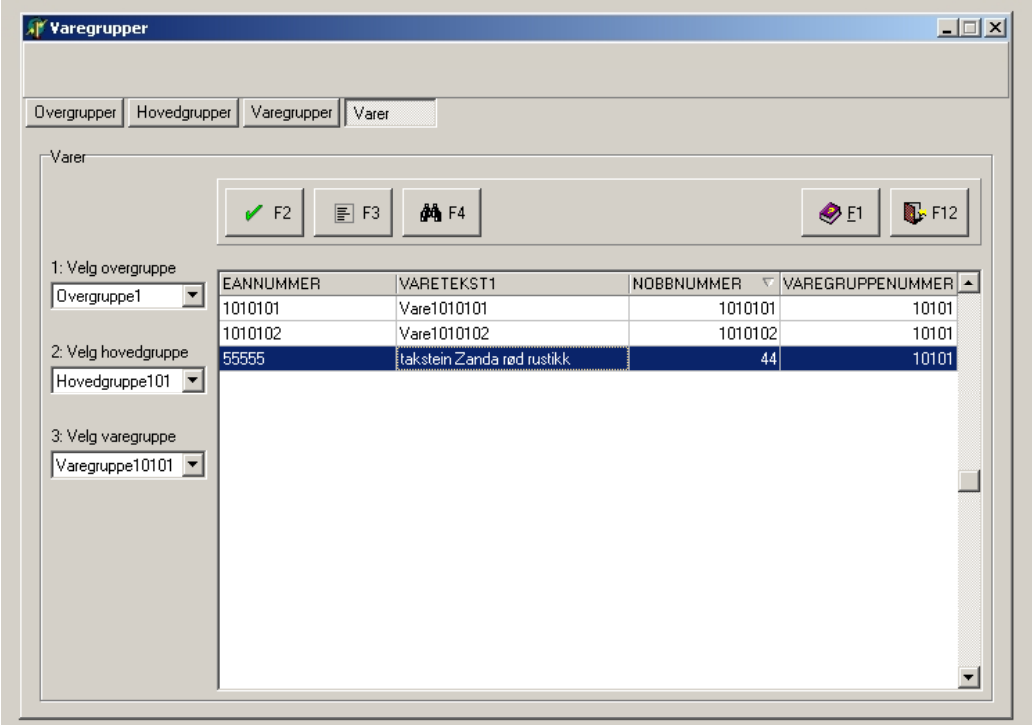
En liten kommentar: denne testen ga ikke forventet resultat ved førstegangs kjøring.

Samme prinsippet har vi fulgt ved det meste av enhetstestingen, først planlegging med valg av testdata, så gjennomføring og eventuell feilsøking.

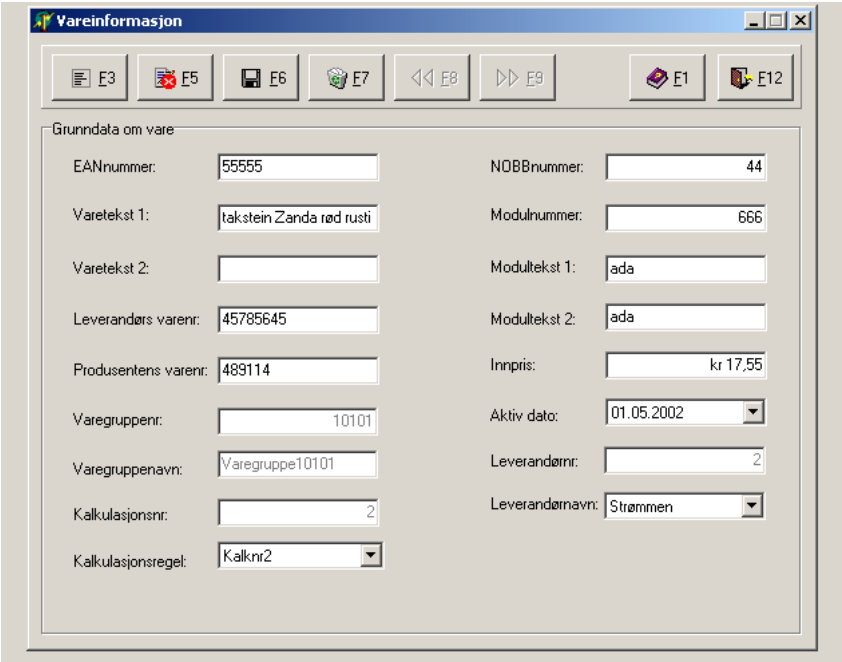
5.3.2 Funksjonalitetstest

Vi illustrerer her hvordan vi har testet funksjonaliteten i implementerte skjermbilder.

Testingen her går ut på å teste at alle knapper og listebokser fungerer som de er tenkt å gjøre, samt at enheten inneholder tilstrekkelig funksjonalitet slikt at kravspesifikasjonen oppfylles.



FIGUR 5-1: VAREOVERSIKT



FIGUR 5-2: VAREINFORMASJON

5.3.3 Integrasjonstest med påfølgende regresjonstester

Etter at enhetene for vareoversikt og vareinformasjon (figur 5-1 og 5-2) er testet hver for seg og funnet ok, er tiden inne for å teste om de fungerer som tenkt i sammen. Eksempelvis skal klikk på knapp F2 i figur 5-1 åpne vareinformasjonsbildet klart til å registrere data om en ny vare. Når registreringen er ferdig og bildet lukkes, skal data om den nye varen vises i oversiktsbildet (figur 5-1).

Etter at alle operasjoner som omfatter begge bildene er testet og godkjent, er det naturlig å kjøre i det minste et utvalg av enhetstestene igjen. Dette for å sikre at integrasjonen ikke har medført nye feilkilder.

Et prinsipp vi konsekvent har fulgt, er det å aldri legge til mer enn en ny enhet om gangen for så å teste ut denne integrasjonen.
Ved å integrere flere enheter enn nødvendig samtidig, vil eventuell feilsøking bli mer omfattende enn nødvendig.

5.4 Vesentlige feil oppdaget under testing

I mange tilfeller har vi funnet småfeil i deler av applikasjonen under enhetstestingen. Disse har alle blitt funnet etter mer eller mindre omfattende feilsøking. Til dette har vi hatt god hjelp av debuggeren i Delphi.
Vi ser ingen hensikt i å gå detaljert inn på disse feilene.

Feil av mer graverende art har blitt avdekket i testene av databasen, og grensesnittet applikasjon database.

En tid slet vi med et tilsynelatende uforklarlig fenomen, databasen oppdaterte seg tilsynelatende selv. Vi fant fort ut at det var klientapplikasjonen som forårsaket problemet. Men å lokalisere feilen var ikke så enkelt.
For å løse dette problemet, benyttet vi oss av kodeinspeksjon.
Vi gikk systematisk gjennom kildekoden bit for bit, og fant ettervert en uønsket kobling til databasen. Da den var fjernet, var problemet løst.

Vi opererer med EANnummer på 13 siffer i systemet. Disse var lagret som flyttall i databasen. Langt ut i prosjektet oppdaget vi at disse på et merkelig vis ikke ble lagret riktig. I visse tilfeller ble de også lagret og vist som eksponentialtall. Dette måtte vi da endre, og vi har endt opp med å benytte et charfelt for å lagre disse numrene i databasen.

Det var den eneste løsningen vi fant, uten at det medførte for store endringer i applikasjonen. Vi ser ingen problemer ved at tallet lagres som et alfanumerisk felt.

Ved testing av triggere og prosedyrer på databasen dukket det også opp enkelte feil. Problemet var generelt at 'flere kjempet om plassen', de påvirket hverandre. Det kan være vanskelig å holde oversikten når det blir mange triggere og prosedyrer på databasen.

Løsningen på de feilene vi fant var enkel nok, det var bare å lokalisere feilkilden for så å skrive om koden.

6. DISKUSJON AV PROSJEKTET

6.1 Prosjektet generelt

Ved de tider vi startet opp prosjektet, var det en enkelte småproblemer i forbindelse med grupperom og datamaskiner til bruk under prosjektet. Dette løste seg imidlertid rimelig greit, og vi installerte oss etter hvert på grupperommet.

Vi bestemte oss for å gjennomføre prosjektet med UP som rammeverk. Denne er inkrementell, og det anså vi som en stor fordel. Objektorientert utvikling var mest nærliggende for oss, dette forsvarer også valget.

Under forprosjektet forsøkte vi på beste vis å estimere et utvalg definerte oppgaver, og på grunnlag av det sette opp en fremdriftsplan. Overraskende nok, så har vi i grove trekk hold denne ganske godt gjennom hele prosjektet. Enkelte ting har gått raskere enn planlagt, andre ting har tatt mer tid. Men tross alt trekker denne planen bare opp de grove retningslinjene. Noen detaljplanlegging på det tidspunkt er urealistisk, dette støttes også opp av metodeverket vårt.

Underveis i prosjektet har vi hele tiden hatt ukentlige statusmøter hvor løsninger og problemer, samt generell status ble drøftet. Arbeidsoppgaver ble fordelt på disse sammenkomstene, på grunnlag av detaljerte iterasjonsplaner som gruppeleder hadde utarbeidet på forhånd. Vi har konsekvent benyttet oss av planlegging og evaluering.
Dette er viktig for å sikre en effektiv fremdrift, samt å holde en god kontroll.

I startfasen hadde vi en tung jobb med analyse og kravspesifisering, men det gikk ikke lenge før vi begynte å implementere de første delene.
Vi har sett at ferdigheten klart har økt ettersom tiden har gått, spesielt med kodingen.

Vi vil kommentere litt omkring vår utviklingsprosess i forhold til prinsippene i UP:

- hvis vi reelt skulle utviklet dette systemet ferdig i sin helhet, ville vi ikke vært ferdig med constructionfasen ennå, så sånn sett blir det litt kunstig at vi har valgt å gå inn i transitionfasen. Men vi valgte nå engang å gjøre det slik, for liksom å komme gjennom alle fasene i løpet av prosjektet.

- UP sier generelt at en skal starte med de aktiviteter som er forbundet med stor risiko. Det har vi til en viss grad valgt å gå bort i fra. Det begrunner vi med at vi mener det er vesentlig å få på plass 'grunnstrukturen' tidlig, og den er ikke forbundet med noen vesentlig risiko i dette systemet.

  Et annet poeng som også støtter vårt valg, er at med den begrensede erfaringen vi har så er det lettere å komme godt i gang hvis en ikke begynner med de mest kompliserte oppgavene.

6.2 Forhold mellom kravspesifikasjon og produkt

Vi har en kravspesifikasjon som definerer mer enn det som er implementert i løsningen pr i dag. Dette har vi vært bevisste på helt fra starten av, vi har aldri hatt som mål å ha noen komplett ferdig løsning, til det er dette systemet for komplekst i løpet av såpass kort tid.

Men nå har det ettervert vist seg at vi har utelatt å implementere enkelte deler som vi opprinnelig hadde planer om å ta med. Det er flere grunner til dette, enkelte rår vi ikke over.

Opprinnelig var det meningen å implementere importrutiner i løsningen.
Men siden vi ikke fikk tilgang på noe å importere fra, følte vi dette som meningsløst. For kommentarer omkring dette, se *VEDLEGG F*.
I den opprinnelige planen hadde vi da også satt visse forbehold ved enkelte punkter.

Kanskje var vi også vel optimistiske med målsettingen vår i utgangspunktet, men det er ikke lett å estimere et såpass stort system med det erfaringsgrunnlaget vi tross alt har.

6.3 Hva vi har oppnådd

Vi føler vi har oppnådd vært hovedmål for prosjektet, som i grove trekk var å kunne gjennomføre et salg med systemet. Det kan vi sånn sett gjøre nå. Vi har gjennomgått en grundig analyse og designprosess, og vi har fått implementert all kjernefunksjonaliteten i systemet.

De elementer som mangler pr i dag, er i stor grad slik det var planlagt ved prosjektstart. Det vi mener burde vært med, er de nevnte importrutiner for å opprette varestrukturen. Dette er en 'håpløs' jobb å gjøre for hånd.
Men dette er forhold som vi ikke rår over, det nytter ikke å kode uten å vite hva en skal kode for!

Det er etter hvert utarbeidet en grundig dokumentasjon som skal være et godt utgangspunkt å bygge videre på.
Databasen for systemet er vi også rimelig godt fornøyd med, men den er nok allikevel ikke slik den endelig vil bli.

Den er blant annet ikke fullt optimalisert pr i dag, da vi ikke har prioritert dette i løpet av prosjektet.
Systemet som det er i dag, kan sees som en første betautgave. Ved grundig utprøving av dette, vil en sikkert se rom for forbedringer av det som er laget til nå.

Som en oppsummering: Vi føler at vi i grove trekk har nådd være mål.

6.4 Status for ByggPOS i dag

All grunndatabehandling (kunder, varer, rabatter, osv) er på plass. Alle de vesentlige momentene i kravspesifikasjonen er også ivaretatt. Salg kan gjennomføres kjapt i kassemodulen ved at varene skannes, ordre kan registreres i kontormodulen. Tilbud kan registreres, og raskt effektueres som ordre. Felles prisbehandling etter forskjellige kriterier kan utføres.

Det er sørget for at det alltid skal være samsvar mellom priser i systemet og priser i butikk, ved at grunnlag for prisetiketter alltid genereres ved alle endringer som utføres i systemet som påvirker priser.

De vesentligste elementer som mangler er en del hardwarerelaterte ting, som integrasjon mot betalingsterminal og kasse. Importrutiner må også implementeres.
Kvitteringsprinter og etikettprinter er heller ikke testet ut, da vi ikke har hatt dette tilgjengelig.
Rapporter er heller ikke utviklet i noen særlig grad, da det ikke er tatt stilling til hvilke rapporter som skal være tilgjengelig.

Vi har også valgt å ikke lage noen brukermanual ennå, det er litt i tidligste laget for den på grunn av systemets status. Men en første, noe enkel, hjelpefunksjonalitet er tilgjengelig i applikasjonene.

Brukeradministrasjon er heller ikke implementert i noen vesentlig grad, utover at det er definert to brukergrupper med differensierte rettigheter i databasen.

6.5 Videreutvikling

Vi anser det nåværende produktet for å være et godt grunnlag å bygge videre på. De første ting som bør utføres, er de punkter som er nevnt under 6.4.
Det må også tas stilling til om fakturamodul skal bygges inn i løsningen, eller om data skal eksporteres for bruk i ekstern fakturaløsning.

Uansett vil vi få et forhold til det med økonomisystem, det å ta stilling til om data skal utveksles på papir, eller om data fra vårt system skal importeres.

Antakelig vil det være hensiktsmessig å benytte seg av ferdig og godkjent interface mot eksempelvis betalingsterminal.

6.6 Gruppearbeidet

Vi har gjennom hele prosjektet hatt grupperom på skolen, der vi har satt opp maskiner i et nettverksmiljø. Dette rommet har vært mye benyttet, men til tider har enkelte valgt å arbeide en del hjemme. Dette har fungert helt fint, da vi har hatt ukentlige møter til fastsatt tid der vi har diskutert løsninger, status, og fordelt oppgaver. Vi har også kommunisert via epost og telefon når det har vært behov.

Vi har hele tiden jobbet parallelt med våre respektive oppgaver, det har vært lite samarbeid i så måte med unntak av litt i startfasen. Dette har vi vært bevisste på, for å få et godt resultat anså vi dette som helt påkrevd. Men når behovet har vært der har vi selvfølgelig samarbeidet for å finne gode løsninger.

Det er litt forskjell på kompetansen innen forskjellige fagområder blant gruppemedlemmene, dette har vi har utnyttet bevisst. Vi har vel i stor grad arbeidet med 'det vi kan best', ulempen med dette er at andre ikke får utviklet sine ferdigheter på de samme områdene i noen vesentlig grad. Men sånn har vi nå engang valgt å gjøre det.

Som de aller fleste gruppesammensetninger, har også vi hatt våre diskusjoner, men vi har alltid kommet greit til enighet. Til tider har vel også noen og enhver følt seg oppgitt og motløs, men det har vi taklet bra ved å støtte hverandre opp på best mulig måte.

Men vi har hatt et problem. Ett av gruppemedlemmene syntes å ha liten interesse for å ta del i prosjektet i noe særlig grad. Dette ble ettervert slik at de andre på gruppa mente noe måtte gjøres, så prosjektleder tok problemet opp med vedkommende. Etter dette, bedret situasjonen seg noe.

Denne saken ble også tatt opp med veileder, og mulige reaksjoner ble vurdert. Men etter nærmere vurdering, valgte de resterende gruppemedlemmene å ikke ta noen videre affære.

Denne saken kjenner veileder og faggruppen ved HiG til, så vi sier ikke mer om dette.

Prosjektleders personlige kommentar

Med unntak av de ovenfor nevnte problemer, så synes jeg samarbeidet har fungert meget bra. Selvfølgelig har det enkelte ganger vært diskusjoner og enkelte misforståelser, men det har alltid løst seg på en smertefri måte.

Jeg vil takke alle på gruppa for samarbeidet, dere har gjort en flott jobb! Det har vært en glede å samarbeide med dere! Jeg føler vi har kommunisert veldig bra, og dere har stått på hele veien. Jeg vil påstå at alle sammen har utviklet seg gjennom dette prosjektet, denne erfaringen er god å ta med seg!

6.7 Veileder

Harald Liodden har vært vår veileder i gjennom prosjektet. I startfasen hadde prosjektgruppa møte med han. Der la veileder frem at han mente vi burde ha møter hver andre uke, og vi fastsatte dag og klokkeslett til annenhver tirsdag kl. 10.00.
Men disse møtene har uteblitt, med unntak av ett tilfelle, som var på dagen to uker etter det første møtet vi hadde.

Men vi har ikke følt det som noe problem at vi ikke har hatt disse møtene, vi har kjørt vårt løp på egen hånd. Prosjektleder har også hatt jevnlig kontakt med veileder i en annen sammenheng, og fortløpende diskutert ting hvis behovet har vært tilstede.
Men vi tør å påstå at ikke alle gruppesammensetninger hadde taklet dette like godt.

Nå fra det litt kritiske over til 'rosen'. Harald har gode kunnskaper i Delphi, utviklingsverktøyet vi har benyttet. Så fremt han har hatt tid, så har han har velvillig stilt opp og hjulpet oss når problemene har tårnet seg opp. Han har aldri vært vanskelig å spørre.

Og takk skal du ha for det, Harald!

6.8 Oppdragsgiver

Vidar Hovdet er oppdragsgiver for dette prosjektet. Han jobber i byggevarebransjen, og har derfor sett behovet for denne type system. På bakgrunn av det, ble dette prosjektet brakt på banen.

Byggevarebransjen har et bredt spekter av 'regler og rutiner' for kundebehandling, varebehandling og rabattering. Dette er noe som er fullstendig håpløst å sette seg detaljert inn i uten assistanse fra en 'innsider'. Her har Vidar vært til god støtte for oss under prosjektet.

Vidar startet like etter nyttår opp sitt eget byggevaresenter, derfor har han hatt lite tid til å følge opp prosjektet. Litt for lite, etter vår mening.

Dette, pluss at han har forholdsvis lite datakunnskap når det kommer til det å utvikle systemer, har vi følt noe problematisk.

Selv om ideen bak prosjektet er at dette systemet i fremtiden skal ut i markedet, er det Vidar som må anses for å være kunden i vårt prosjekt. Og ved utviklingen av et såpass stort system som dette, er det utrolig viktig å ha et tett samarbeid med kunden tidlig i prosessen. Det føler vi har sviktet litt.

Vi håper systemet en dag kommer ut på markedet, Vidar!

7. KONKLUSJON

Nå er hovedprosjektet over, etter nærmere et halvt års til tider ganske så intenst arbeid.
Det var en interessant utfordring vi tok fatt på ved juletider.
Ingen av gruppemedlemmene hadde noen erfaring fra denne type prosjekt tidligere, så det var virkelig en utfordring å sette opp en plan som det var realistisk å holde for et såpass stort system som dette er.
Dette var nok kanskje den største utfordringen i startfasen. Det var mer enn nok å ta av, så vi måtte finne den riktige avgrensningen. Og det har vi faktisk greid rimelig bra, synes vi.

Det var også spennende å se hvordan samarbeidet mellom gruppen, oppdragsgiver og veileder ville bli, og ikke minst internt i gruppen.
Enkelte av oss følte nok på forhånd at vi hadde potensielle utfordringer her, men i det store og hele har det gått greit selv om vi stort sett har jobbet på egenhånd uten så mye 'innblanding' fra veileder og oppdragsgiver gjennom prosjektet.

Gjennomføringen av dette prosjektet har gitt oss meget nyttig erfaring i det å gjennomføre en hel utviklingsprosess. Fra analysering for å klarlegge krav, frem til et ferdig implementert system. I alle fall et delvis ferdig system, det gjenstår jo som nevnt en del elementer.

Vi har også fått utvidet vår kunnskap innen databaser og Delphi-programmering. Spesielt databasekunnskapen er nyttig å ha med seg videre i karrieren. Dette er kunnskap som er etterspurt i næringslivet.

Prosjektarbeid er også flittig benyttet rundt i bedrifter, så det er en uvurderlig nyttig erfaring å ha vært gjennom et prosjekt som dette, nå tenker vi på erfaringer både på godt og vondt. Det er viktig å lære å takle eventuelle problematiske situasjoner som måtte oppstå, dette vil alle komme borti før eller siden.

Vi er godt fornøyd med resultatet vårt, det har vært en både interessant og utfordrende jobb å komme dit vi er i dag. Vi er strålende fornøyd med den iterative metoden vi har jobbet etter. Hadde vi kunnet gjøre prosjektet på nytt, er i alle fall ikke det noe vi ville ha endret!

Systemet har all kjernefunksjonalitet ferdig, så det bør være et godt utgangspunkt å bygge videre på. Vi skulle gjerne hatt mer tid, slik at vi kunne fått systemet ferdig!

Om vi er litt lei og slitne nå i de aller siste dagene før innlevering, så vil vi nok etter en tid sitte igjen med positive minner fra våren 2002.

Vi håper oppdragsgiver også vil bli fornøyd, og at han vil få ByggPOS ut på markedet etter hvert.

8. LITTERATURLISTE

8.1 Litteratur

[Cantu] Marco Cantu: “Mastering Delphi 3”, Second Edition, Sybex, ISBN 0-7821-2052-0

[Sommerville] Ian Sommerville “Software Engineering”, Sixth Edition, Addision-Westly, ISBN 0-201-39815-X

[Larman] Craig Larman “Applying UML and Patterns”, Second Edition, Prentince Hall, ISBN 0-13-092569-1

Diverse dokumentasjon om NOBB lånt fra Mocon AS

Diverse dokumentasjon om Borland Interbase

8.2 Internettressurser

[NOBB web] Beskrivelse av strukturen
http://www.varedata.no/web/odaweb.nsf?OpenDatabase

[Delphi 1] Komponenter og diverse om Delphi
http://www.torry.net/index.htm

[Delphi 2] Komponenter og diverse om Delphi
http://vclcomponents.com/

[Delphi 3] Komponenter og diverse om Delphi
http://freebyte.com/programming/delphi/

[Delphi 4] Komponenten vi benytter for strekkodeskanner
http://www.deepsoftware.ru/nrcomm/

[HiG] Retningslinjer fpor hovedprosjekter
http://www.hig.no/avdelinger/hovedprosjekter_at/

[RUP] beskrivelse av utviklingsmodellen RUP
http://www.rational.com/products/whitepapers/100420.jsp

[Arkitektur] Philippe Kruchten: Beskrivelse av 4+1 View software arch.
http://www.rational.com/products/whitepapers/350.jsp

[Cockburn] Structuring Use Cases with Goals
http://members.aol.com/acockburn/papers/usecases.htm

9. VEDLEGG

Vedlegg A:	Ordforklaringer (terminologiliste)
Vedlegg B:	Forprosjektrapport
Vedlegg C:	Prosjektavtale
Vedlegg D:	Statusrapporter
Vedlegg E:	Møtereferater
Vedlegg F:	Dokumentasjon av endringer
Vedlegg G:	Analysedokumenter applikasjon (kravspesifisering)
Vedlegg H:	Designdokumenter applikasjon
Vedlegg I:	Databasedesign
Vedlegg J:	Opprinnelig fremdriftsplan og iterasjonsplaner
Vedlegg K:	Ressursforbruk
Vedlegg L:	Kodeprefiks
Vedlegg M:	CD-rom med kildekode for applikasjoner, script for DB, kompilerte applikasjoner, GDB-fil, komponenter, beskrivelse for oppsett, hjelpefil