Universitetet i Bergen

Masteroppgave

Visualisering og mønstergjenkjenning på transaksjonsbaserte data fra Enterprise Resource Planning og Point of Sale baserte bedrifter

Skrevet av:

Eskil A. Hesselroth

Institutt for informasjons- og medievitenskap

1. mai 2015

Abstrakt

De fleste bedrifter benytter i dag forretningssystemer for å samle inn og behandle forretningsrelaterte data. *Enterprise Resource Planning(ERP)* og *Point of Sale(POS)* programvare er eksempler på systemer som begge generer og lagrer store mengder av denne typen data relatert til handel. I en spørreundersøkelse Stodder (2013) gjennomførte, uttrykte 37 % av de 453 deltagende respondentene at tilgang til data fra disse og andre typer forretningssystemer er meget viktig i datavisualiseringsapplikasjoner. Basert på dette, identifiseres det et behov for å kunne uthente informasjon fra transaksjonsdata, som kan brukes til for eksempel mersalg, markedsføring og vareplassering. For å kunne tilfredsstille dette behovet, presenteres det i dette prosjektet en applikasjon ved navn EinBlick, et verktøy utviklet iterativt i Java. Applikasjonen benytter metoder fra forskningsområdene datavisualisering og data mining, for å tilby brukerne funksjonalitet for å navigere, filtrere, søke, visualisere og analysere transaksjonsdata. Ved å støtte oppkobling og uthenting av data fra relasjonsdatabaser, forhindrer den håndtering av data lokalt i for eksempel tekstfiler. I tillegg er det mulighet for brukeren å kunne kombinere data fra flere tabeller, og dette gir støtte for å få innsyn i transaksjonsdata med samme struktur fra ulike kilder. EinBlick gir brukeren mulighet til å generere flere ulike interaktive visualiseringer og finne asssosieringsregler basert på brukerens data. Denne funksjonaliteten gjør det mulig for brukere å kunne dykke inn i dataene, for både å bekrefte og utforske relasjoner og mønstre som ikke ville vært synlig i en skjematisk form. Det har blitt gjennomført to evalueringer i prosjektet ved hjelp av testdeltagere fra konsulentselskapet Advania. Den første evalueringen ble gjort for å danne et inntrykk vedrørende deltagernes meninger om applikasjonen, og hvorvidt de fant den brukervennlig. I den andre evalueringen ble applikasjonen med implementert data mining funksjonalitet evaluert, for å undersøke hvorvidt denne funksjonaliteten forbedret deltagernes opprinnelige inntrykk. Funnene fra evalueringen viser at de har svært positive inntrykk av applikasjonen, og at de finner den brukervennlig. De viser også at implementering av data mining funksjonalitet forbedrer dette inntrykket ytterligere.

Takk til

En stor takk rettes til min veileder Weiqin Chen for hennes veiledning, tilbakemelding og tålmodighet. Jeg vil også takke min samboer og familie som har både støttet og motivert meg. Til slutt vil jeg takke ansatte i Advania for deres deltagelse i evalueringer og møter.

Innhold

Figurer

Tabeller

Kapittel 1

Introduksjon

Mange bedrifter benytter i dag både *Enterprise Resource Planning(ERP)* og *Point of Sale(POS)* systemer for å loggføre store mengder transaksjonsbaserte data. Stodder (2013) har ved hjelp av en spørreundersøkelse, hvor det var 453 deltagende respondenter, identifisert at 37 % av respondentene synes tilgang til data fra ERP og andre typer forretningssystemer er meget viktig i datavisualiseringsapplikasjoner. Disse systemene lagrer store mengder data relatert til handel, og i følge Leonard og Wolfe (2005) har bedrifter ofte et behov for å uthente kunnskap fra denne dataen. En måte å finne denne informasjonen på er å bruke metoder fra forskningsområdene datavisualisering og data mining. Den uthentede informasjonen kan bistå bedrifter med å danne forståelse for forretningen, øke fortjeneste og forstå kunder bedre.

Forskningsområdet visualisering har igjennom tidene fått mye oppmerksomhet, men den største andelen av forskningen tilhører fagområder med ekstreme mengder data, eksempelvis biologi eller geografi. I dette prosjektet er det visualisering av transaksjonsdata som primært er interessant, og innenfor dette området er det begrenset med forskning. Det finnes dog noe, og i artikkelen til Keim, Hao, Dayal og Lyons (2007) presenteres det en ny måte å visualisere transaksjonsdata på ved hjelp av en forbedret utgave av det klassiske stolpediagrammet. Durand, Crémilleux og Suzuki (2006) presenterer visualisering av transaksjonsbaserte data ved hjelp av *clustering*, hvor transaksjonene grupperes ved hjelp av ulike parametere. I artikkelen til Theus (2002) presenteres datavisualiseringsapplikasjonen Mondrian, et verktøy som støtter mange ulike visualiseringstyper og er utviklet i Java. Morales-Chaparro, Preciado og Sánchez-Figueroa (2011) legger i deres

artikkel vekt på at de fleste visualiseringsverktøy avviker fra hva brukeren ønsker å få se, og hva de får se ved endrede brukerbehov. I artikkelen deres presenteres det også en vevbasert applikasjon med flere ulike visualiseringer som kan visualisere flere typer datasett.

Under det store begrepet data mining er det gjort mye forskning, og det finnes flere ulike metoder som kan benyttes for å uthente innsiktsfull informasjon. En mulighet er å benytte seg av metodene for å finne hyppige/sjeldne produktsett, og assosieringsregler basert på disse. Det velkjente, og kanskje til og med fabrikkerte eksemplet på at øl og bleier ofte kjøpes sammen, er et eksempel på et hyppige produktsett som kan være tilgjengelig i transaksjonsbaserte data (Power, 2002). For å finne de hyppige eller sjeldne produktsettene kreves det stor beregningskraft av maskinvaren, og i forskningsområdet er det gjort mye for å gjøre denne prosessen mindre krevende. Koh og Rountree (2005), Bodon (2003), Gu, Wang, Zhang, Wang og Gao (2011) og Han, Pei og Yin (2000) presenterer alle metoder som skal være mer effektive enn den originale *Apriori* algoritmen presentert i artikkelen til Agrawal, Imieliński og Swami (1993). Biblioteket *Sequential Pattern Mining Framework* presentert av Gomariz, Gueniche og Fournier-viger (2014) har støtte for alle disse algoritmene, og åpner opp for å kunne hente ut både produktsett og assosieringsregler fra store databaser.

1.1 Forskningsspørsmål og metode

Basert på behovet for å finne innsiktsfull informasjon i transaksjonsdata har denne oppgaven tatt for seg følgende forskningsspørsmål:

> Hvordan utvikle et data mining- og datavisualiseringsverktøy for transaksjonsdata fra Enterprise Resource Planning og Point of Sale systemer

For å gjennomføre en best mulig forskning er rammeverket *Design Science* presentert av Hevner, March, Park og Ram (2004), tatt i bruk. Rammeverket presenterer syv retningslinjer som hjelper forskeren med å gjennomføre god forskning. For å kunne besvare forskningsspørsmålet er det utviklet en applikasjon ved navn EinBlick. Sluttbrukeren av denne applikasjonen defineres som mennesker med IT-kompetanse ansatt i en organisasjon som benytter seg av, eller har kunder som benytter seg av *Enterprise Resource*

Planning(ERP) og *Point of Sale(POS)*systemer. Det defineres to scenarioer for sluttbrukere, hvor det første er når ansatte i konsulentselskap som videredistribuerer ERP og POS systemer benytter seg av applikasjonen for å gi sine kunder informasjon. Det andre scenarioet er hvor butikkjeder selv benytter applikasjonen for å skaffe denne informasjonen.

1.2 Organisering av oppgaven

I dette kapittelet har forskningsspørsmålet blitt introdusert og begrunnet. Kapittel to vil introdusere relevant teori og definere terminologier. Etter dette beskriver kapittel tre hvordan rammeverket *design science* og andre metoder er benyttet for å sikre en god forskning. Kapittel fire beskriver design og utviklingsprosessen av applikasjonen, og kapittel fem presenterer evalueringen. Til slutt konkluderes oppgaven i kapittel seks, med en oppsummering og diskusjon av fremtidig arbeid.

Kapittel 2

Litteratur

I dette kapittelet presenteres relevant teori til prosjektet. Forskningsområdene datavisualisering og data mining vil defineres, og relevant teori innen forskningsområdene presenteres.

2.1 Datavisualisering

Den amerikanske avisen *The New York Times* bruker avanserte dataanalyser og innovativ datagrafikk for å gi brukerne innsikt i nyhetshistorier som ellers ville vært begravd i tekst (Stodder, 2013). Det er nettopp dette datavisualisering er, en metode for å fremstille data grafisk. En grafisk visualisering av store mengder data gjør dem mer forståelige og viser sammenhenger og trender. Kaidi (2000) definerer visualisering som en grafisk representasjon av informasjon, hvor målet er å gi seeren kvalitativ forståelse av innholdet i informasjonen. Det er også prosessen med å transformere objekter, konsepter og tall til en form som er synlig for det menneskelige øye. Han mener et datavisualiseringsverktøy gir brukere mulighet til å se mønstre, teste hypoteser, teste unntak og forklare funn med illustrasjoner for andre brukere. Friendly (2009) definerer datavisualisering som forskningen av visuell representering av data, hvor data blir definert som informasjon abstrahert i en skjematisk form, med attributter eller variabler for enhetene til informasjonen.

2.1.1 Hvorfor datavisualisering

I følge Kaidi (2000) er det å se og forstå bilder et naturlig instinkt for mennesker, i motsetning til å forstå tallbaserte data som krever mange års trening. Fra et veltegnet bilde, er det mye lettere å finne trender og relasjoner. Fayyad, Grinstein og Wierse (2001) mener mennesker leter etter struktur, mønstre, egenskaper, avvik og relasjoner i data. Visualiseringer støtter dette ved å representere dataen i forskjellige former, og med interaksjoner. En visualisering gir mulighet for å vise et kvalitativt bilde av store og komplekse datasett. Den kan oppsummere og hjelpe til med å identifisere områder av interesse for å gjøre mer spesifikke kvantitative analyser. Visualiseringene kan også brukes til å utforske data, bekrefte en hypotese, eller til å manipulere betrakteren, for eksempel i en markedsbrosjyre.

2.1.2 Datatyper for visualisering

I følge Keim (2002) finnes det mange velkjente teknikker for å visualisere data, for eksempel X og Y plotter, linjeplotter og histogrammer. Disse teknikkene er veldig gode og brukbare, men begrenset til få-dimensjonale data. I den siste tiden har det i større grad blitt fokusert på å utvikle fler-dimensjonale metoder for å visualisere data. I artikkelen presenteres det en måte å klassifisere de ulike typene data på:

- Endimensjonal

  Data med kun én dimensjon. Et typisk eksempel er temporal data, for eksempel tidsserier fra aksjepriser.

- Todimensjonal

  Data med to dimensjoner der et typisk eksempel er longitude og latitude data relatert til geografi.

- Multidimensjonal

  Data med flere enn to eller tre dimensjoner. Eksempelvis data eksportert fra relasjonsdatabaser.

- Tekst og hypertekst

  Ikke alle typer data kan bli gruppert i dimensjonstyper. Verdensveven inneholder

store mengder tekst, hypertekst og multimediabaserte data. Disse dataene er forskjellige fra det vi er vant med, da de ikke enkelt kan forklares med tall, og av den grunn kan de færreste standard visualiseringsmetoder brukes. Ofte må dataene bli representert i vektorer før de kan visualiseres. Antall ord er et eksempel på dette.

- Hierarkier og grafer

  Grafvisualisering av data er en moderne visualiseringsteknologi, og kan for eksempel brukes på data som kommunikasjon i et e-postnettverk. Hierarkisk data kan for eksempel være strukturen på en harddisk.

- Algoritmer og software

  Datavisualisering kan også benyttes for å hjelpe med å forstå en algoritme, for eksempel ved å vise flyten av informasjon i et program.

2.1.3 Relatert arbeid i feltet datavisualisering

Som nevnt i kapittel 1 er det begrenset forskning i forskningsområdet visualisering som omhandler visualisering av transaksjonsbaserte data. Det finnes dog noe, og i dette avsnittet presenteres relevant forskning.

I artikkelen til Morales-Chaparro et al. (2011) legges det vekt på hvordan flere eksisterende datavisualiseringsverktøy avviker fra hva brukeren ønsker å se, og faktisk får se ved endrende brukerbehov. Det påpekes at dersom en bruker vet hva slags informasjon hun vil ha, bør det være mulighet for å tilpasse og lage visualiseringer for å uthente korrekt informasjon på en korrekt måte, basert på brukerens behov. Artikkelen presenterer en applikasjon som benytter seg av en data- og brukerbasert tilnærming for å visualisere flerdimensjonal brukervalgt data i en nettleser. Ved hjelp av visualiseringsontologier håndteres visualiseringstyper og deres krav. Ontologien gir mulighet for å tilby brukeren passende visualiseringstyper for de ønskede dataene. For eksempel kan en geografisk visualisering tilbys hvis dataene inneholder x og y koordinater. Artikkelen relaterer seg til prosjektet, da den presenterer en måte å visualisere data på, uavhengig av datakilden. Den illustrer hvordan en visualiseringsapplikasjon ikke skal utvikles basert på ett datasett, men heller være åpen for flere typer data. Den påpeker også viktigheten av at visualiseringen skal kunne tilfredsstille brukernes behov, som endrer seg med tiden.

Applikasjonen *Mondrian* som blir presentert av Theus (2002) er et datavisualiseringsverktøy som støtter flere ulike visualiseringstyper. Mondrian tilbyr interaktiv spørring på datasettet ved hjelp av et utvelgelsesrektangel. Rektangelet gir mulighet for å velge deler av datasettet, og kjøre spørringer direkte. I stolpediagrammer bruker Mondrian horisontale i stedet for vertikale stolper, da dette gir mulighet for å skrive ut hele navnet på stolpene. I artikkelen påpekes det at utviklingen av grafiske applikasjoner i Java er enklere enn noen gang. Java har mulighet for alle grafiske visninger, og godt utviklede Java applikasjoner kjører smertefritt på dagens maskinvare. Mondrian står frem som et godt eksempel for hvordan datavisualiseringsapplikasjoner kan utvikles i Java, og gir inspirasjon til flere ulike typer visualiseringer. I tillegg presenterer Mondrian den unike utvelgelsesprosessen ved hjelp av rektangelet, samt at den aksepterer datasett fra flere kilder.

I artikkelen til Ko, Maciejewski, Jang og Ebert (2012) presenteres den interaktive applikasjonen *MarketAnalyzer*. Applikasjonen er ment for leverandører som mottar *Point of Sale(POS)* data fra forhandlere. I artikkelen presenteres det hvordan denne dataen kan brukes for å gi leverandøren såkalt *competitive intelligence(CI)*. De definerer *CI* som handlingen med å utforske, analysere og forutsi markedsandelen. Bedrifter kan benytte seg av *CI* for å sammenligne seg med andre bedrifter, identifisere markedsrisiko og muligheter, samt for å evaluere innvirkningen av nye salgsstrategier. Ved hjelp av avanserte visualiseringer og filtreringsfunksjonalitet, gir applikasjonen mulighet for å finne mengder med informasjon klassifisert innenfor *CI*. Applikasjonen er relevant til forskningen ettersom den også baserer seg på transaksjonsdata for å gi bedrifter innsiktsfull informasjon. Scenarioet der dataen brukes for å gi leverandører *CI*, skiller seg fra prosjektets forskningsspørsmål, men det er fortsatt interessant å se hvordan de løser visualiseringen av transaksjonsbaserte data.

Keim et al. (2007) presenterer en ny type visualisering for store mengder transaksjonsdata. Den nye visualiseringen kalles *value-cell bar charts*, og ligner på det klassiske stolpediagrammet. Diagrammet er i følge utviklerne enkelt og morsomt å bruke. Ved hjelp av diagrammet kan brukere analysere store mengder transaksjonsbaserte data ved hjelp av et diagram som ligner på det velkjente stolpediagrammet. I *value-cell bar charts* representerer en stolpe flere transaksjoner, og hver transaksjon får sin egen celle så lenge verdien er tilsvarende en celle. Er verdien av transaksjonen lavere enn hva en celle representerer, vil flere transaksjoner være representert av cellen. Denne visualiseringstypen gir mulighet

for å visualisere store mengder data, samtidig som den beholder den klassiske ideen bak stolpediagrammer, hvor høyden på stolpen representerer verdien. Diagrammet har også såkalt *drill down* funksjonalitet, som gjør det mulig å se nærmere på dataen i en stolpe. Visualiseringstypen presentert i denne artikkelen relaterer seg til prosjektet, og illustrerer hvordan store mengder transaksjonsdata kan visualiseres.

I artikkelen til Durand et al. (2006) introduseres konseptet med å visualisere transaksjonsbaserte data ved hjelp av *clustering*. Dette gir mulighet for å gruppere lignende transaksjoner basert på ulike parametere. Ved bruk av ulike *clustering* metoder kan det også sammenlignes hvilke metoder som egner seg best, samt gir best resultat. Artikkelen relaterer seg til prosjektet på grunn av deres visualisering av transaksjoner, men det påpekes dog at det i dette prosjektet ikke er tatt i bruk *clustering* algoritmer.

Stodder (2013) har gjennomført en spørreundersøkelse med 453 deltagende respondenter. Undersøkelsen tar for seg organisasjoners erfaringer med datavisualisering, og praksiser for data discovery og teknologier. Samtidig anbefaler den de beste praksisene for å forbedre beslutningstaking og analyse. I artikkelen poengteres det viktigheten av datavisualisering, og viktigheten av at organisasjoner undersøker hvordan de kan matche visualiseringsteknologier, og praksiser til brukerkrav. Det poengteres også at datainteraksjon er et sentralt element for å oppnå suksessrike visualiseringer. Videre legges det vekt på hvor viktig datavisualisering er for profesjonelle dataanalytikere, samt for å få innsyn i bedriften. Visualiseringer er utgangspunktet for hvordan dataen blir presentert for brukeren. Gode visualiseringer er kritisk for å utføre smarte beslutninger, og forbedre produktivitet. Dårlige visualiseringer kan derimot villede brukeren, og gjøre det vanskelig å overkomme de store datamengdene. Artikkelen relaterer seg til prosjektet på grunn av formålet med, og funnene gjort i spørreundersøkelsen.

2.2 Data mining

Han, Kamber og Pei (2012) definerer data mining som prosessen for å finne interessante mønstre og kunnskap fra store mengder data, hvor datakildene kan være databaser, datavarehus, Internett eller andre informasjonslager. Data mining er som omtalt i boken til Edelstein (1999) ikke en tryllestav, og selv med data mining kreves det kunnskap om forretningen, forståelse for dataen og kunnskap om de analytiske metodene. Det er også

viktig å forstå at mønstre funnet ved hjelp av data mining bør verifiseres i den virkelige verden.

Data mining kan brukes til alle fasene i kundenes livssyklus, for eksempel for å anskaffe nye kunder, øke inntekter fra eksisterende kunder og beholde eksisterende gode kunder. Et eksempel er å gi kunder tilpassede tilbud etter hva lignende kunder har kjøpt tidligere. Det er som nevnt i boken til Edelstein (1999) flere ulike industrier som benytter seg av data mining. For eksempel bruker kredittkortutstedere, telekommunikasjonsselskaper, forsikringsselskaper og aksjemeglerselskaper teknologien for å redusere svindel. I medisin bruker noen data mining for å anta effektiviteten av kirurgiske operasjoner, medisinske tester og medisiner. Handelsbedrifter benytter seg av data mining for bestemme hvilke produkter som skal kjøpes inn i hvilke butikker, og hvordan de skal plasseres i butikken. De bruker også teknologien for å vurdere effektiviteten til kampanjer og kuponger. Dette er selvfølgelig bare noen av områdene og eksemplene data mining brukes til.

I noen sammenhenger brukes data mining for å referere til hele prosessen i *knowledge discovery(KDD)*, men egentlig er data mining kun ett av stegene i *KDD*. Data mining defineres i prosessen som ett essensielt steg for å uthente datamønstre ved hjelp av intelligente metoder. I boken defineres stegene i *KDD* med:

1. Datarensing(fjerne støy og inkonsistente data)
2. Dataintegrering(flere datakilder kan kombineres)
3. Datautvelgelse(hent ut data som er relevant for analysen)
4. Datatransformasjon(dataen blir transformert eller konsolidert til former som passer til uthenting)
5. Data mining(en avgjørende prosess hvor intelligente metoder brukes for å hente ut datamønstre)
6. Mønsterevaluering(finn interessante mønstre som representerer kunnskap basert på interessante målinger)
7. Kunnskapspresentasjon(bruker datavisualisering og kunnskapsrepresentering for å presentere kunnskapen

2.2.1 Data mining algoritmer

I artikkelen til Agrawal et al. (1993) ble problemet med å finne assosieringsregler i store transaksjonsdatabaser introdusert. De deler dette problemet opp i to subproblemer for å effektivisere prosessen. Det første problemet er å finne hyppige produktsett, og det andre problemet er å finne assosieringsregler basert på disse produktsettene. Å finne disse reglene er ett felt innenfor data mining, og er som Han et al. (2000) nevner velstudert. Assosieringsregler er i følge Han et al. (2012) mønstre som forekommer hyppig i datasett, og et hyppig produktsett referer til et sett av produkter som forekommer ofte i en transaksjonsdatabase. I senere tid er det også blitt interessant å finne produktsett som forekommer sjelden, som forskningen til Koh og Rountree (2005) presenterer.

2.2.1.1 Assosieringsregler

Assosieringsregler er interessante relasjoner ofte gjemt i store datasett funnet ved hjelp av assosieringsanalyse. Eksempelet vedrørende sammenhengen mellom øl og bleier nevnt i kapittelet Introduksjon, er et klassisk eksempel på en assosieringsregel. Agrawal et al. (1993) mener denne informasjonen kan brukes av handelsbransjen til for eksempel vareplassering, mersalg og markedsføring.

I sin bok definerer Tan, Steinbach og Kumar (2005) en assosieringsregel med $X \longrightarrow Y$, hvor X og Y representerer ulike produktsett. Styrken av reglen kan defineres ved hjelp av støtte(*support*) og tiltro(*confidence*). Støtte representerer hvor mange ganger reglen forekommer i datasettet, og tiltro hvor mange ganger Y forekommer i transaksjoner som inneholder X. I boken begrunnes disse to målingene og viktigheten av dem:

1. Støtte trengs for å bevise om reglen forekommer av ren tilfeldighet, og en regel med lav støtte kan bevise dette. Det påpekes også at bedrifter mest antagelig ikke er interesserte i regler som forekommer sjelden.

2. Tiltro måler hvor pålitelig regelen er, og jo høyere tiltro for $X \longrightarrow Y$, desto høyere sjanse er det for at Y forekommer i transaksjoner som inneholder X.

Som det også nevnes i boken, er en mulighet for å finne assosieringsregler å regne ut støtte og tiltro for hver eneste mulige regel, men dette vil være veldig kostbart. I stedet for er en

vanlig tilnærming å først finne hyppige eller sporadiske produktsett som forekommer over eller under en grense satt for støtte, og deretter finne assosiseringsregler i produktsettene som har høyere tiltro enn den satte grensen. Denne tilnærmingen gjør prosessen med å finne reglene ofte mindre kostbar enn prosessen med å finne selve produktsettene. Det finnes flere ulike tilnærminger for å finne produktsettene, og de som er relevante til forskningen presenteres nedenfor.

2.2.1.2 Apriori

Apriori ble presentert av Agrawal et al. (1993) som en rask og effektiv tilnærming for å finne assosieringsregler i en transaksjonsdatabase basert på hyppige produktsett. Dette er den kjente tilnærmingen diskutert ovenfor(2.2.1.1), der problemet med å finne assoiseringsregler deles opp i to subproblemer:

- Finn produktsett som har en støtte over minimumsstøtte. Kall disse produktsettene hyppige produktsett, og alle andre ikke hyppige.
- Basert på de hyppige produktsettene, finn regler som har en tiltro over grensen for minimumstiltro.

Navnet på algoritmen er i følge Han et al. (2012) basert på at den bruker kunnskap fra tidligere hyppige produktsett for å finne nye hyppige produktsett. I praksis betyr det at hvis et produktsett forekommer frekvent, så vil alle subsettene også forekomme frekvent. I tillegg kan det konkluderes med at hvis et produktsett ikke forekommer frekvent, så vil heller ikke supersettene forekomme frekvent. Formelt kan det defineres som en iterativ tilnærming der k-produktsett blir brukt for å finne $k+1$ produktsett. I første omgang skannes databasen for å finne produktsett som inneholder ett produkt($k = 1$), og hvor ofte de forekommer. Produktsettene som er over minimumsgrensen blir lagt til i resultatet L_1. Deretter brukes L_1 for å finne produktsett som inneholder to produkter(L_2), som igjen brukes for å finne produktsett som inneholder tre produkter(L_3) og så videre. Denne prosessen fortsetter helt til det ikke kan finnes flere produktsett.

I senere tid viser det seg som Han et al. (2012), Bodon (2003) og Gu et al. (2011) poengterer at den originale *Apriori* tilnærmingen ikke er så rask som først antatt. Det er forsøkt å lage flere nye versjoner av *Apriori* for å effektivisere den originale algoritmen.

Eksempler på to av disse er diskutert i artiklene til Bodon (2003) og Gu et al. (2011). Det finnes også flere alternativer til *Apriori*, og noen av dem er betydelig raskere. I artikkelen til Han et al. (2000) presenteres *FP-Growth*, en tilnærming som er omtrent ti ganger raskere. Denne algoritmen vil bli diskutert senere i avsnitt 2.2.1.5. Det er viktig å påpeke at det er det første subproblemet, å finne hyppige produktsett som ofte blir optimalisert i nye tilnærminger. Subproblemet med å finne regler basert på disse hyppige produktsettene er såpass effektivt, at det heller fokuseres på å effektivisere prosessen med å finne hyppige produktsett.

2.2.1.3 Apriori-Inverse

For å finne sporadiske produktsett med en vanlig *Apriori* tilnærming må minimumsgrensen som diskutert i artikkelen til Koh og Rountree (2005) settes veldig lavt, og dette øker kjøretiden til algoritmen drastisk da mengden potensielle produktsett økes betydelig. *Apriori-Inverse* metoden derimot benytter en tilpasset tilnærming for å finne sporadiske produktsett. Algoritmen har i motsetning til en vanlig *Apriori* algoritme to terskler, en for minimums- og en for maksimumsstøtte. Minimum setter grensen for hvor sporadiske produksett algoritmen skal lete etter, og maksimum setter en grense for hvor ofte de kan forekomme. For at algoritmen i det hele tatt skal klare å kjøre, behøves minimumsgrensen for å forhindre den store mengden potensielle produktsett. En transaksjonsdatabase kan inneholde millionvis av sjeldne produktsett, og terskelen sørger for at algoritmen kan være kjørbar ved å kun inkludere en viss andel av disse.

I artikkelen defineres det to ulike typer sporadiske produktsett, perfekte og ikke-perfekte. Et perfekt produksett kan defineres som et produktsett hvor settet ikke har noen subsett som overstiger maksimumsstøtten. Et ikke-perfekt subsett derimot, kan inneholde subsett som har en høyere støtte enn maksimumsstøtten. En *Apriori-Inverse* algoritme finner sporadiske perfekte produktsett, og prosessen for å finne disse er gjengitt nedenfor.

- Generer en invertert indeks I som inneholder produkter og transaksjonsID-er.(Illustrert på tabell 2.1)
- Generer sjeldne produktsett hvor $k = 1$(produktsett basert på ett produkt)
  - Lag et tomt sett S_1

TransaksjonsID	Produkter
0001	c & d & a
0002	c & d & b
0003	d & a

Produkt	TransaksjonsID-er
a	001 & 003
b	002
c	001 & 002
d	001 & 002 & 003

Tabell 2.1: Normal og invertert transaksjonstabellindeks

  - For hvert produkt i tilhørende den inverterte indeksen I, iterer:
    * Hvis antallet av produktet i overstiger minimumsstøtten og er under maksimumsstøtten, legg i til i settet S_1.

- Finn S_k sett med sjeldne k-produktsett hvor k er større enn eller er lik to
  - For hvert sett, sjekk om alle subsettene er høyere enn minimumsstøtten ved hjelp av I, hvis ja, er settet et sjeldent sett, hvis ikke fjern det.

2.2.1.4 Frequent Pattern Tree

For å kunne forklare den effektive algoritmen *FP-Growth*, er man avhengig av å forstå strukturen Frequent Pattern Tree(*FP-tree*). Denne strukturen er en kompakt struktur som representerer transaksjonsdata i en trestruktur. Treet har en rotnode med navn *null*, og hvert produktsett i transaksjonsdatabasen har en sti i trestrukturen. For å lage et *FP-tree* av en transaksjonsdatabase, sorteres produktene etter hvor ofte de forekommer. På denne måten kan transaksjoner som inneholder de samme produktene gjenbruke en sti, eller deler av en sti i treet. I artikkelen til Han et al. (2000) oppsummeres egenskapene av et *FP-tree*:

- Treet inneholder en rotnode med navn null. Denne rotnoden er koblet til et sett av subtrær.
- Treet inneholder en tabell for å koble produkter og deres første forekomst i treet. Denne tabellen har to felter, et felt for produktnavnet og et felt som peker til den første noden i treet med dette produktnavnet.
- Hver node i treet inneholder tre felter: produktnavn, antall og nodekobling. Produktnavnet er hvilket produkt denne noden representerer. Antallet representerer antall transaksjoner som inneholder stien frem til denne noden. Nodekoblingen kobler noden til neste node i treet som har samme produktnavn.

Ser vi på node a i figur 2.1, forstår vi at denne representerer produktsettet $c\&d\&a$. Ettersom noden har et antall på to, kan vi forstå at dette produktsettet forekommer to ganger(transaksjon 0001 og 002). Dette gjør som nevnt ovenfor at treet blir betydelig mer kompakt enn transaksjonsdatabasen, med mindre hver eneste transaksjon inneholder unike produkter.

I artikkelen forklares det hvordan algoritmen går frem for å lage et *FP-tree* basert på en transaksjonsdatabase:

1. Skann transaksjonsdatabasen D èn gang, og samle inn de hyppige produktene F , og deres støtte(antall ganger de forekommer). F sorteres synkende basert på støtte(beste først). Det sorterte datasettet kalles L, og representer listen over hyppige produkter.

2. Lag rotnoden T for FP-treet, og gi den navnet "null". For hver transaksjon $Trans$ i D gjøres følgende.

   - Velg og sorter de hyppige produktene i $Trans$ i henhold til den sorterende rekkefølgen L. La det sorterte hyppige produktsettet i $Trans$ være $[p|P]$, hvor p er det første produktet i produktsettet, og P er de resterende. Kall funksjonen legg_til_i_tre($[p|P]$,T), som gjør følgende:

     (a) Hvis T har en node N som har samme navn som p(T.navn = p.navn), øk antallet for N med 1.

     (b) Hvis ikke, lag en ny node N og la antallet være 1. Opprett en kant mellom rotnoden T og N, og opprett koblinger mellom noder som representerer det samme produktet.

     (c) Hvis P ikke er tom, kall metoden legg_til_i_tre(P,N) rekursivt.

På figur 2.1 illustreres et forenklet eksempel på hvordan treet generes ut ifra en transaksjonsbasert database. De stiplede linjene representerer koblinger mellom noder som representerer det samme produktet. I praksis kan algoritmen litt forenklet forklares slik:

1. Skann igjennom en transaksjonsdatabase, og samle inn de hyppige produktene og deres støtte(hvor mange ganger de forekommer). Sorter synkende, etter hvor hyppig produktene forekommer(beste tilfelle først).

2. Lag en rotnode i FP-treet, og gi den navnet "null". For hver transaksjon i databasen gjør følgende:

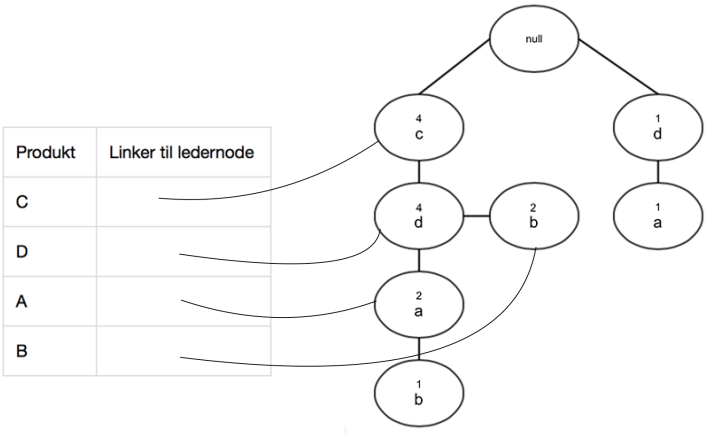
   - Sorter produktene i transaksjonen i henhold til hvor hyppig de forekommer. Legg til denne transaksjonens produktsett i treet ved å:

     (a) Hvis rotnoden allerede har en node direkte koblet til seg, med samme navn som det første produktet i produktsettet, øk antallet med én for denne noden.

     (b) Hvis ikke, lag en ny node og sett antallet til én. Opprett en kant mellom rotnoden og denne nye noden, og opprett koblinger mellom andre noder som representerer det samme produktet.

     (c) Hvis det er flere produkter i produktsettet, legg til disse rekursivt.

TransaksjonsID	Produkter
0001	c & d & a
0002	c & d & b
0003	d & a
0004	c & d & b
0005	c & d & a & b



FIGUR 2.1: Frequent Pattern Tree eksempel

2.2.1.5 FP-Growth

Den mest velbrukte tilnærmingen for å finne produktsett som ofte eller sjelden kjøpes sammen er gjerne *Apriori*. Denne tilnærmingen er som diskutert i avsnitt 2.2.1.2 veldig krevende av maskinvaren når det brukes en lav minimumsstøtte, og som en reaksjon på dette har Han et al. (2000) kommet med en ny tilnærming ved navn *FP-Growth*. Denne tilnærmingen benytter seg av datastrukturen *Frequent Pattern Tree* beskrevet i avsnitt 2.2.1.4.

Ved testing har utviklerne av *FP-Growth* funnet ut at deres algoritme er minst ti ganger raskere enn en *Apriori* basert tilnærming. I artikkelen oppsummeres det følgende fordeler med *FP-Growth* i forhold til andre metoder:

1. Algoritmen lager et veldig kompakt *FP-tree*. I de fleste tilfeller vil dette treet bli betydelig mindre enn transaksjonsdatabasen, og av den grunn kreves det mindre databeregningskraft for å skanne treet, i motsetning til å skanne hele transaksjonsdatabasen.

2. Den bruker en mønsterøkende metode som forhindrer dyre kandidatgenerasjoner, og tester ved å linke sammen hyppige produksett bestående av ett produkt funnet i de betingede FP trærne.

3. FP-Growth benytter seg av en *divide and conquer* metode - en tilnærming for å dele problemene opp i subproblemer. Dette reduserer drastisk størrelsen på mønstre som brukes som baser for andre mønstre, og *condtional pattern trees*.

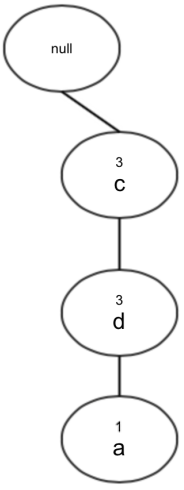
For å finne hyppige produktsett benytter algoritmen to steg. Først analyseres transaksjonsdatabasen og det generes et Frequent Pattern Tree. Deretter hentes det ut hyppige produktsett fra treet. For å finne de hyppige produktsettene bruker *FP-Growth* en nedenfra og opp metode. Metoden starter på nodene i enden av grenene og beveger seg opp mot rotnoden. I algoritme 1 vises det hvordan algoritmen finner ofte kjøpte produktsett basert på en transaksjonsdatabase.

Algoritme 1 FP_Growth

α er produktsett fra databasen
β er et produktsett i α sin *condtional pattern base*
Hvis $Tree$ har kun en sti P
    For hver kombinasjon β av noder i P **gjør:**
        Slå sammen α og β til et nytt produktsett κ
        Sett støtten til κ til den laveste støtten av nodene i β
        Legg til κ i listen over hyppige produktsett
Hvis $Tree$ har mer enn en sti
    For hver produkt a_i i listen over ledernodene **gjør:**
        Slå sammen a_i og α til et nytt produktsett β
        Sett støtten til β til a_i sin støtte
        Lag β sin *conditional pattern base*
        Basert på *condtional pattern base*, lag β sin *condtional pattern tree* $Tree_\beta$
        Hvis $Tree_\beta$ ikke er tomt
            kall metoden FP_Growth($Tree\beta$,β)

Hvis figur 2.1 og en minimumsstøtte på tre brukes som eksempel, vil algoritmen gå rett til linje 8 i koden vist på algoritme 1, da treet har mer enn en sti. Den vil da starte på produkt "B"(a_0), og ettersom α er et tomt sett vil β kun inneholde produktet "B". Støtten til β settes til a_0 sin støtte, og ettersom den er 3, anses den å være over minimumsgrensen. β er med andre ord et frekvent produktsett, og av den grunn lages β sin *condtional pattern base* [b,a,d,c]. Basert på *condtional pattern base* genereres β sin *condtional pattern tree* $Tree_\beta$(vises på figur 2.2). Algoritmen vil til slutt kalle på seg selv med $Tree_\beta$ og produktsettet β som parameter.

Ettersom $Tree_\beta$ kun har en sti, vil linje 3 i koden nå være gyldig. Algoritmen vil nå legge til hyppige produktsett over minimumsstøtten, for hver kombinasjon av nodene i treet. Da "A" er den første noden, starter algoritmen med denne, og β settes til produkt "A". α vil i dette tilfellet hele tiden være produktet "B", ettersom treet representerer "B" sitt condtional pattern tree. κ vil av den grunn representere produktsettet [B,A]. κ har en for lav støtte(1) i henhold til minimumsstøtten, så produktsettet blir ikke lagt til. I neste iterasjon gjøres akkurat det samme, og her representerer κ produktsettet [B,D]. κ sin støtte settes til β(produkt D) sin støtte(3). Da støtten er over grensen for minimumsstøtte blir produktsettet lagt til i listen over hyppige produktsett. Algoritmen vil fortsette til det ikke kan finnes flere hyppige produktsett.



Produkt	Produktsett
b	$b\&c\&d$, $b\&d$, $b\&c$ og b
a	$a\&c\&d$, $a\&d$, $a\&c$ og a
c	$c\&d$ og c
d	d

FIGUR 2.2: Tabell over alle produktsett funnet over minimumsstøtte, og Condtional Pattern Tree for produkt b

2.2.2 Relatert arbeid i feltet data mining

I denne oppgaven er det relevant å bruke et bibliotek som inneholder relevante data mining algoritmer, og i forskningsområdet er det allerede flere bibliotek tilgjengelig. Dette avsnittet vil av den grunn deles opp i to deler, hvor den første diskuterer relevant forskning, og den andre relevante bibliotek.

2.2.2.1 Relevant forskning

I artikkelen til Yong (2012) presenteres spørrespråket *Transaction Data Mining Language(TDML)*. Språket tilbyr en ad-hoc tilnærming for å finne flere ulike typer kunnskap i transaksjonsbaserte databaser. Det støtter flere ulike data mining teknikker, som for eksempel funn av assosieringsregler. For å finne disse bruker språket tilnærmingene *FP-Growth*(2.2.1.5) og *Apriori*(2.2.1.2). I artikkelen presenteres det en benchmarking av algoritmene. Som vist i denne bruker *Apriori* betydelig lengre tid på å finne de hyppige produktsettene. For 500 000 transaksjoner bruker *Apriori* 1113 sekunder, og *FP-Growth* bare 258. Forskningen er først og fremst relatert fordi den viser en sammenligning av *Apriori* og *FP-Growth*. For å lese og skrive data benytter språket seg av flate tekstfiler, noe som gjør datahåndtering veldig tungvint og lite brukervennlig. Under fremtidig arbeid foreslås det å implementere funksjonalitet for å lese relasjonsdatabaser, men det stilles fortsatt spørsmål til hvordan funnene skal presenteres.

I artikkelen til Legler, Lehner og Ross (2006) presenteres det hvordan forfatterne har brukt data mining på verktøyet *SAP NetWeaver Business Intelligence Accelator*. I henhold til artikkelen tilbyr verktøyet betydelig raskere hastighet ved aksessering av data i relasjonsdatabaser. De benytter algoritmen *BUC iceberg cubing* for å finne frekvente produktsett. For å gjøre dataaksessering raskere bruker verktøyet blant annet hashmaps for å mappe strengverdier til integerverdier, da dette krever mindre datakraft. Artikkelen relaterer seg til forskningen da også forfatterne benytter data mining på transaksjonsbaserte data. Ved hjelp av *BUC iceberg cubing* finner de hyppige produktsett basert på det velkjente datasettet *mushrooms*, men også på virkelige forretningsdata lagret i et SAP ERP system. Det er viktig å påpeke at deres løsning vil knytte seg opp mot ett spesifikt ERP system ved å benytte SAP sitt verktøy. Løsninger som for eksempel Microsoft Dynamics AX vil ikke ha tilgang på dette. Av den grunn baserer prosjektets applikasjon

EinBlick seg i stedet for på relasjonsdatabaser, så det potensielt sett skal være mulig å aksessere alle mulige forretningssystemer, og kombinere data fra dem.

Kaur og Singh (2013) har benyttet data mining for å analysere handelsdata basert på sporter spilt i India. Sportene analysert var cricket, hockey, badminton, bordtennis og tennis. De benytter seg av *FP-Growth* presentert i avsnitt 2.2.1.5 for å finne assosieringsregler. Ved analysering fant de flere assosieringsregler med en tillit på 100%. For eksempel fant de den logiske regelen som sier at hvis en kunde kjøper en badmintonball, er det 100% sannsynlighet at kunden også kjøper en badmintonrekkert. De fant også andre regler, som for eksempel at mange kunder som kjøper håndkle og badmintonball, kjøper badmintonrekkert. Begge disse reglene forekom over 0.3(normalisert) ganger i datasettet, altså var de hyppige. Basert på reglene blir det i artikkelen presentert flere tips for hvordan sportsbedrifter kan øke fortjenesten. De skiller tipsene i tre kategorier:

1. Produktanbefalinger og promoteringer

   Anbefale kunder produkter basert på andre produkter de har i handlevognen, for eksempel en rekkert hvis kunden kjøper badmintonball og håndkle. Promoteringer kan for eksempel være å gi kunden en gratis badmintonball hvis de kjøper badmintonrekkert.

2. Produktplassering

   Assosieringsreglene gir tydelig inntrykk av hvordan varene bør plasseres i butikken. For eksempel bør håndkler være lett tilgjengelig ved siden av badmintonutstyret.

3. Next best offer

   Ved hjelp av assosieringsregler kan man ha kunnskap om hvilke produkter som kan være interessante ved kundens neste kjøp.

Denne artikkelen gir et godt innblikk av hvordan assosieringsregler kan gi større forståelse av bedriften, og i tillegg øke fortjeneste ved hjelp av for eksempel smartere vareplassering. Den er høyst relevant ettersom formålet med forskningens data mining funksjonalitet er nettopp dette. Det eksperimentelle resultatet er kanskje ikke like overraskende som bleier $\longrightarrow$ øl, men dette er på grunn av de tydelige varekategoriene i sportsbutikker. Sportsbutikker selger hobbybaserte produkter, og kunder går mindre sjelden på tvers av varekategorier enn i for eksempel en matvarebutikk.

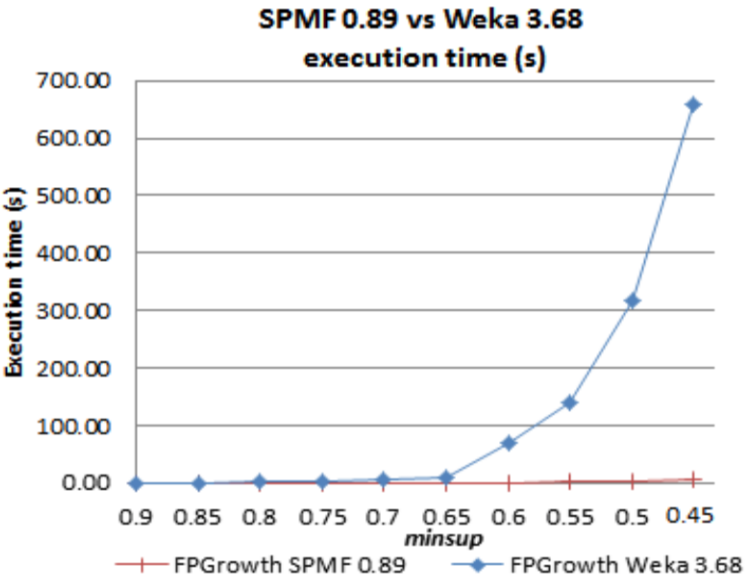
Forskningen til Shim, Choi og Suh (2012) benytter seg av data mining for å gi *customer relatetionship management(CRM)* anbefalinger. Først klassifiserer de kundene i to grupper, VIP kunder og ikke-VIP kunder ved hjelp av data mining verktøyet WEKA. Deretter finner de assosieringsregler og sekvensielle mønstre basert på VIP kundenes transaksjonsdata. For å teste algoritmen benytter de seg av transaksjonsdata fra et lite online kjøpesenter i Korea. Nettbutikken selger husholdningsprodukter som for eksempel håndklær, såpe og tannbørster. Basert på disse dataene finner de flere assosiseringsregler sortert på kategorier, som for eksempel at kunder som kjøper boller ofte kjøper kopper. Basert på funnene presenterer de CRM anbefalinger, som for eksempel redesign av nettsiden etter hvilke kategorier som ofte kjøpes sammen, og personlige markedsføringsstrategier. Denne forskningen er også høyst relevant, da den beviser hvor stor gevinst en bedrift kan ha av å få innsyn i transaksjonsdataene. Den viser videre mange gode forslag til hvordan data mining kan brukes for CRM.

2.2.2.2 Relevante bibliotek

Biblioteket *Sequential Pattern Mining Framework(SPMF)* presentert i artikkelen til Gomariz et al. (2014) er et data mining bibliotek som følger standarden for åpen kildekode(*open source*) ledet av Philippe Fournier-Viger. Biblioteket er spesialisert på å finne mønstre i transaksjons- og sekvensdatabaser. Eksempler på mønstre er ofte kjøpte produktsett, assosieringsregler og sekvensielle mønstre. I følge artikkelen er SPMF siden 2010 brukt i flere enn 70 forskningsprosjekter i forskjellige domener, som for eksempel salgsprognoser og data mining for nettbruk. Ettersom all kildekode til SPMF er tilgjengelig for å bli implementert i andre Java prosjekter, åpner det for gjenbruk av kildekode og enklere implementering av data mining algoritmer i prosjektets applikasjon. SPMF tilbyr alle de nevnte data mining algoritmene i avsnitt 2.2.1, og kan av den grunn sees på som et velutviklet bibliotek med implementasjoner av algoritmer fra solid forskning.

I deres artikkelen presenterer Hall et al. (2009) applikasjonen WEKA, et prosjekt finansiert av New Zealand som har som mål å tilby en omfattende kolleksjon av *machine learning* algoritmer og *data preprocessing* verktøy. Prosjektet ble startet i 1992 og har siden den gang kommet en lang vei takket være fellesskapet og bidragsyterne. WEKA tilbyr som SPMF en åpen kildekode der funksjonalitet tilgjengelig i prosjektet kan implementeres i egne Java applikasjoner. WEKA er akseptert i både forskningsområdet og

i næringslivet. Det påpekes at SPMF likevel er et bedre alternativ for å finne assosieringsregler og sporadiske/hyppige produktsett, da biblioteket har et mye større utvalg av algoritmer. I tillegg viser en benchmark av kjøretiden for implementasjonen av *FP-Growth* algoritmen til både WEKA og SPMF, at sistnevntes implementasjon er betydelig raskere. Sammenligningen er gjort på flere ulike datasett, og på figur 2.3 vises en benchmark for ett av datasettene(Fournier-Viger, 2015)



FIGUR 2.3: Sammenlignet kjøretid av SPMF og WEKA sin *FP-growth* implementasjon

Det finnes også flere andre gode alternativer, og RapidMiner(RapidMiner, 2015), Mahoot(Apache, 2015), Knime(Knime, 2015) og Coron(Coron, 2015) er noen av dem. Coron er ikke *open source*, og er av den grunn ikke mulig å implementere i egen kode(Gomariz et al., 2014). Mahoot, Knime og RapidMiner følger derimot denne standarden, og skal alle være mulig å implementere. SPMF viser uansett et mye større utvalg av algoritmer for å finne hyppige produktsett. I tillegg virker det som kun SPMF tilbyr algoritmer for å finne sjeldne produktsett. Ingen av de ovennevnte tilbyr for eksempel en implementasjon av *Apriori-Inverse* algoritmen diskutert i avsnitt 2.2.1.3. Av den grunn stiller SPMF som et mye sterkere og bedre dokumentert bibliotek for dette prosjektet.

Kapittel 3

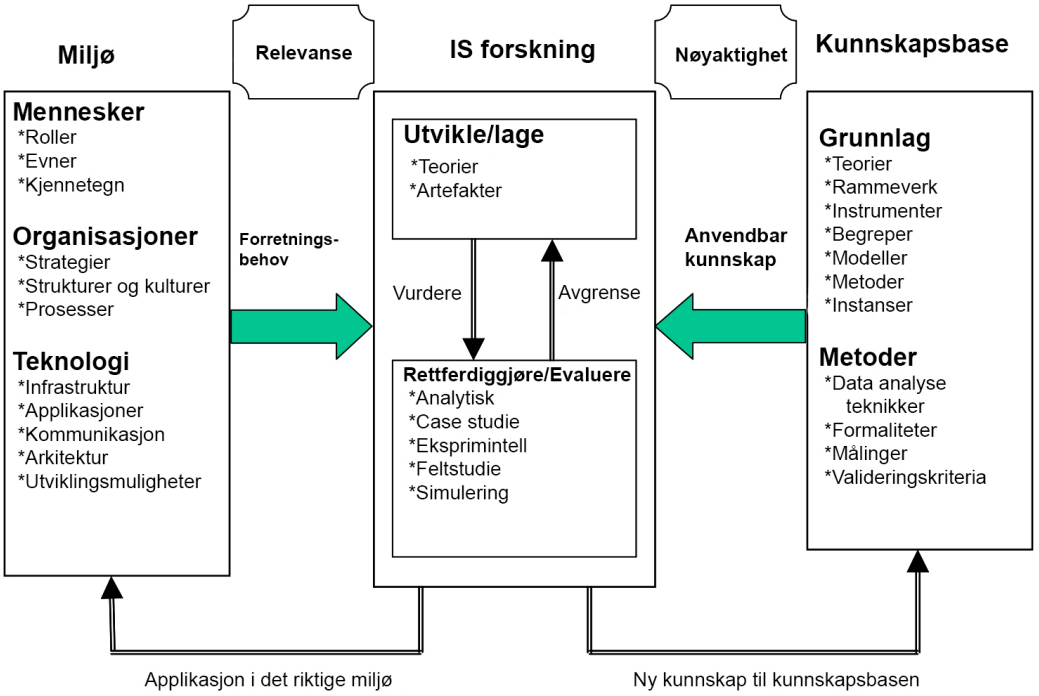
Metode

Dette kapittelet tar for seg metoder benyttet i forskningsprosjektet. Først vil rammeverket *Design Science* presenteres, og videre de syv retningslinjene som Hevner et al. (2004) introduserer i deres artikkel. Deretter vil datainnhentingsteknikker forklares, og til slutt vil brukervennlighet defineres.

3.1 Design Science

I dette avsnittet presenteres rammeverket *Design Science*, et rammeverk for å lage innovative artefakter. I følge Hevner et al. (2004) er *Design Science* et rammeverk for å utvide evnene til mennesker og organisasjoner ved å skape nye og innovative artefakter. De definerer IT artefakter som instanser, modeller, metoder og begreper brukt i utvikling og bruk av informasjonssystemer. Begrepene defineres som vokabularet og symbolene brukt for å definere problemer og løsninger. De har en betydelig påvirkning på måten oppgaver og problemer blir tilnærmet på. Representasjonen av problemer og løsninger har en betydelig påvirkning på arbeid med forskning og design. En instans er kulturelle egenskaper i organisasjonen, pakket inn i noe sosialt gjenkjennelig som maskinvare eller programvare. Et eksempel på et instans artefakt er intellektuelle verktøy eller programvareverktøy som forbedrer prosessen for utvikling av informasjonssystemer.

På figur 3.1 illustreres rammeverket *Design Science*. Miljøet som består av mennesker, organisasjoner og teknologi skaper forretningsbehov som er igangsettere for å starte forskning. Under forskningen utvikles det teorier og artefakter som hele tiden vurderes



FIGUR 3.1: Rammeverk for forskning på informasjonssystemer

og avgrenses ved å rettferdiggjøre dem. Forskningen både benytter seg av og legger til kunnskap i kunnskapsbasen, som inneholder grunnlag og metoder. Resultatet av hele forskningsprosessen er en applikasjon som legges til i det riktige miljøet, og nyvinnende kunnskap til kunnskapsbasen.

Hevner et al. (2004) oppgir syv retningslinjer, som vist i tabell 3.1, for hvordan *Design Science* kan gjennomføres. Retningslinjene behøver ikke følges i rekkefølge eller med rigide krav, men hver enkelt retningslinje bør tas opp og evalueres ved forskning. Nedenfor presenteres og defineres de syv retningslinjene, og under hver retningslinje diskuteres det hvordan denne oppgaven følger dem.

1. Design som et artefakt

Resultatet av en *Design Science* forskning skal returnere et meningsfullt IT artefakt, laget for å adressere et viktig organisasjonsproblem. Det er viktig at artefaktet beskrives på en effektiv måte for å muliggjøre implementering og anvendelse i egnet domene. Artefakter konstruert i *Design Science* forskning,er sjelden ferdige informasjonssystemer som

Retningslinje	Beskrivelse
1. Design som et artefakt	Designforskning skal returnere et meningsfullt artefakt i form av modell, metode, begrep eller instans.
2. Problemrelevans	Formålet med å gjøre designforskning er å utvikle teknologibaserte løsninger for viktige og relevante forretningsproblemer.
3. Design evaluering	Nytten, kvaliteten og effekten av et design artefakt må være nøyaktig demonstrert via velutviklede evalueringsmetoder.
4. Bidra til forskningsområdet	Effektiv designforskning må gi klare og verifiserbare bidrag til forskningsområdet artefaktet hører til.
5. Nøyaktighet	Designforskning avhenger av bruken av nøyaktige metoder i både utviklingen og evalueringen av artefaktet.
6. Design som en søkeprosess	Letingen etter et godt artefakt krever at tilgjengelige midler brukes for å nå ønskede mål, samtidig som lover tilfredsstilles i miljøet.
7. Kommunikasjon av forskingen	Designforskning må bli presentert godt til både teknologi- og administrasjonsorienterte publikum.

Tabell 3.1: Retningslinjer i rammeverket *Design Science*

blir brukt i praksis. Artefaktene skal definere ideer, praksiser, tekniske muligheter og produkter som igjennom analyse, design, implementering og bruk av informasjonssystemer effektivt kan oppnås (Hevner et al., 2004).

I dette prosjektet er resultatet artefaktet EinBlick, en applikasjon som viser bedrifter som bruker *Enterprise Resource Planning* og *Point of Sale* systemer hvordan de kan dra nytte av metoder fra forskningsområdene datavisualisering og data mining.

2. Problem relevans

Formålet med å gjøre forskning innenfor informasjonssystemer er i følge Hevner et al. (2004) å anskaffe kunnskap og forståelse. Kunnskapen gir mulighet for utvikling og implementering av teknologibaserte artefakter som løser ikke løste og viktige forretningsproblemer. Et eksempel på et problem kan være forskjellen mellom et mål og dagens tilstand på et system. Hevner et al. (2004) mener at forretninger er målorienterte og lever i en økonomisk setting. Av den grunn er ofte forretningsproblemene til organisasjonen å øke fortjeneste eller å redusere kostnader.

Applikasjonen EinBlick illustrerer hvordan bedrifter kan oppnå både økt fortjeneste og reduserte kostnader ved hjelp av metoder fra datavisualisering og data mining. Ved hjelp av innsyn i dataene til en handelsbedrift, kan bedrifter som diskutert i avsnitt 2.2.1.1

øke salget ved hjelp av for eksempel forståelse av vareplassering. Informasjonen kan også brukes på flere andre områder, for eksempel kundespesifikk markedsføring og promotering. Forskningen til Kaur og Singh (2013) diskutert i avsnitt 2.2.2 er et godt eksempel på dette.

3. Evaluering av design

Evaluering av design er en kritisk komponent av forskningsprosessen. Forretningen danner krav og ønsker basert på evaluering av artefaktet underveis. Dette gir rom for å bruke en inkrementell utvikling av IT artefakter. Artefaktene evalueres på funksjonalitet, ferdighet, konsistens, presishet, ytelse, pålitelighet, brukervennlighet, hvor mye det passer organisasjonen og andre relevante attributter. I følge Hevner et al. (2004) kan matematikk brukes for å evaluere artefakter som bruker analytiske målinger. For eksempel kan databasedesign algoritmer evalueres ved å bruke forventet maskinvarekost og forventet responstid. Algoritmer relatert til søk kan bruke *precision* og *recall* ved evaluering. Det designede artefaktet er i følge Hevner et al. (2004) ferdig og effektivt når det tilfredsstiller krav og begrensninger til problemet det var ment for å løse. Evalueringen av det designede artefaktet bruker typisk metoder tilgjengelig i kunnskapsbasen. De utvalgte evalueringsmetodene må passe til det designede artefaktet og evalueringsberegningene.

Det er primært holdt to evalueringer av applikasjonen i dette prosjektet, som diskutert i avsnitt 5.2. I tillegg er applikasjonen kontinuerlig blitt fremvist til domeneeksperten. Som Hevner et al. (2004) nevner er det flere ulike attributter en applikasjon kan evalueres på, og en av dem er brukervennlighet. Under evalueringen har brukervennlighet hatt et stort fokus, og for å få en indikasjon på hvorvidt applikasjonen er brukervennlig er det tatt i bruk metoden *System Usability Scale* diskutert i avsnitt 3.2.1.

4. Bidra til forskningsområdet

Artefaktet må i følge Hevner et al. (2004) gi tydelige og verifiserbare bidrag til forskningsområdet. Det beste scenario for vurdering av forskning er "hva er nytt med forskningen og hva er det interessante bidraget?". *Design Science* forskning har tre ulike muligheter for å bidra til forskningsområdet basert på nyheten, generaliteten og betydeligheten av artefaktet.

1. **Det designede artefaktet**

   I de fleste tilfeller vil bidraget til forskningsområdet være artefaktet selv, som vist med den venstrepekende pilen på figur 3.1. Artefaktet må løse et problem som i utgangspunktet var et uløst problem. Det kan utvide kunnskapsbasen, eller tilby eksisterende kunnskap på en ny og innovativ måte.

2. **Stiftelser**

   På figur 3.1 illustreres det ved hjelp av den høyrepekende pilen hvordan forskningen kan bidra. Den kan bidra via den kreative utviklingen av nye, passede evaluerte begreper, modeller, metoder eller instanser som kan utvide eller forbedre eksisterende stiftelser i kunnskapsbasen.

3. **Metoder**

   Den høyrepekende pilen i figur 3.1 viser også hvordan den kreative utviklingen og bruken av evalueringsmetoder gir forskningsbidrag og legger til i kunnskapsbasen. Måling og evalueringsberegninger er spesielt viktige komponenter i *Design Science*.

Det er altså tre ulike muligheter for en forskningsprosess å bidra til forskningsområdet. I dette prosjektet bidras det ved hjelp av den første muligheten, det designede artefaktet. Det skal bidra til forskningsområdet ved å vise bedrifter som benytter seg av *Enterprise Resource Planning(ERP)* og *Point of sale(POS)* systemer, hvordan metoder fra datavisualisering og data mining kan brukes for å skaffe innsyn i deres transaksjonsbaserte data.

5. Forskningsnøyaktighet

Design Science avhenger av nøyaktige metoder i både utvikling og evaluering av det designede artefaktet. Forskningsnøyaktighet handler om måten forskning blir gjennomført på, og god forskning forekommer ved effektiv bruk av kunnskapsbasen. Suksessen til artefaktet avhenger av riktig utvelgelse av relevante teknikker fra kunnskapsbasen for å utvikle, og evaluere artefaktet (Hevner et al., 2004).

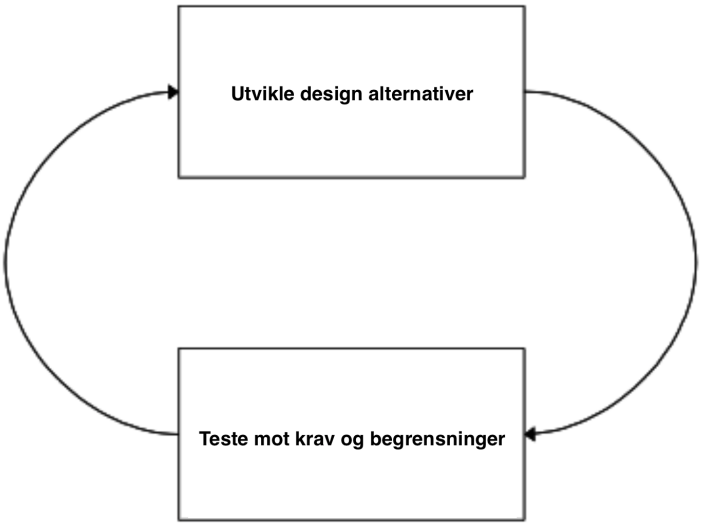
Forskningsprosessen har i dette tilfellet benyttet seg av flere ulike teknikker tilgjengelig i kunnskapsbasen. For håndtering av utviklingsprosessen, er verktøyet Kanban diskutert i avsnitt 4.6.2 benyttet. Ved hjelp av evalueringsteknikker tilgjengelige i litteraturen

til Rogers, Sharp og Preece (2011) har EinBlick blitt evaluert. Som tidligere nevnt i avsnitt 3.2.1 er også rammeverket *System Usability Scale* benyttet for å gi en indikasjon hvorvidt applikasjonen er brukervennlig. Sammen sørger disse teknikkene for at dette prosjektet har tatt i bruk gode metoder fra kunnskapsbasen i både utvikling og evaluering av den designede applikasjonen.

6. Design som en søkeprosess

I artikkelen til Hevner et al. (2004) diskuteres det hvordan det ofte er vanskelig å finne det beste eller optimale designet for realistiske informasjonssystemsproblemer. De definerer design som søkeprosessen for å oppdage en effektiv løsning på et problem, som i seg selv er inkrementell og iterativ. Denne iterative prosessen er gjengitt på figur 3.2. For å finne den effektive løsningen på et problem defineres det i artikkelen tre faktorer. Den første er midler, hvor midler defineres som handlinger og ressurser tilgjengelig for å utvikle en løsning. Den andre faktoren er mål, hvor mål representerer mål og begrensninger løsningen skal inneholde. Den siste faktoren er lovene, og lovene representerer ukontrollerbare styrker i miljøet. I følge Hevner et al. (2004) kan abstraksjon og representasjon av disse faktorene være viktige komponenter av design. Disse faktorene er problem- og miljøavhengige, og involverer kreativitet og innovasjon. Ofte i *Design Science* forskning, kan forskningsproblemet representeres forenklet ved å kun representere en mindre del av de relevante midlene, mål og lover. En slik forenkling er ofte ikke realistisk nok, men kan representere et startpunkt. Problemet vil bli iterativt behandlet etter hvert som forskningsproblemet utvides, og midler, mål og lover vil bli redefinert. I artikkelen nevenes det videre at designløsningen kan bli urealistisk, og stille for høye krav til artefaktet. Kravene må da reduseres, og designløsningen må resultere i en tilfredsstillende løsning.

I dette prosjektet har det som tidligere nevnt blitt benyttet en iterativ og inkrementell utviklingsprosess, og denne prosessen har forsøkt å oppnå en effektiv løsning på et problem. Svaret på forskningsspørsmålet og eventuelle begrensninger som dukket opp underveis representerer målet med selve prosjektet. Verktøy, tilgjengelig domeneekspert, evalueringsdeltagere representerer de tilgjengelige midlene. Til slutt representerer de ukontrollerbare styrkene i miljøet lovene, for eksempel datastrukturen til systemene dataen hentes fra.



FIGUR 3.2: Utviklings- og testsyklus

7. Kommunikasjonen av forskningen

Forskningen som *Design Science* prosessen resulterer i, må presenteres til både tekniske og administrasjonsorienterte publikum. De teknisk orienterte trenger tilstrekkelig detaljer for å kunne implementere artefaktet, og bruke den i korrekt organisatorisk kontekst. Dette gir mulighet for de tiltenkte brukerne til å bruke artefaktet, og det gir mulighet for forskerne å oppdatere forskningsbasen for ytterligere arbeid. De administrasjonsrelaterte trenger tilstrekkelig detaljer for å avgjøre om artefaktet skal videreutvikles eller implementeres. De administrative trenger også noen tekniske detaljer for å forstå nytten, og sette pris på artefaktet (Hevner et al., 2004).

Denne oppgaven skal legge til grunn for de ovennevnte faktorene. Den skal kunne leses både av det tekniske og det administrasjonsorienterte publikum.

3.2 Brukervennlighet

Tross utvikling innenfor fagfeltet databehandling, henger deler av gammel tenkning med oss. I de tidligere dager, da CPU-en var hjertet til systemet, diskuterte designere terminaler og enheter. I denne perioden begynte mennesker å bruke termen "sluttbruker", som ofte ble assosiert med det siste som ble tenkt på under design av systemet. Designere ser ofte på brukerne av informasjonssystemer som sluttbrukere. Dette synet gjør ofte at

systemene ender opp med dårlige brukerdesign, og feiler på brukervennlighet. Shackel (2009) mener designere må se brukeren som senteret av systemet, ikke en uvesentlig del.

Brukervennlighet avhenger av designet på verktøyet med hensyn til brukere, oppgavene, miljøene og suksessen av gitt brukerstøtte. Brukervennlighet for individuelle brukere blir evaluert på subjektiv vurdering av hvor lett designet er å anvende sammen med brukerstøtten, og på objektive mål av effektiviteten til verktøyet. Shackel (2009) definerer brukervennlighet av et system som evnen for å bli anvendt av mennesker enkelt og effektivt, hvor enkelt defineres som et bestemt nivå av subjektiv vurdering, og effektivt defineres som et bestemt nivå av menneskelige prestasjoner.

Når brukere skal gjennomføre valg mellom flere systemer, baserer deres valg seg i følge Shackel (2009) på flere ulike faktorer.

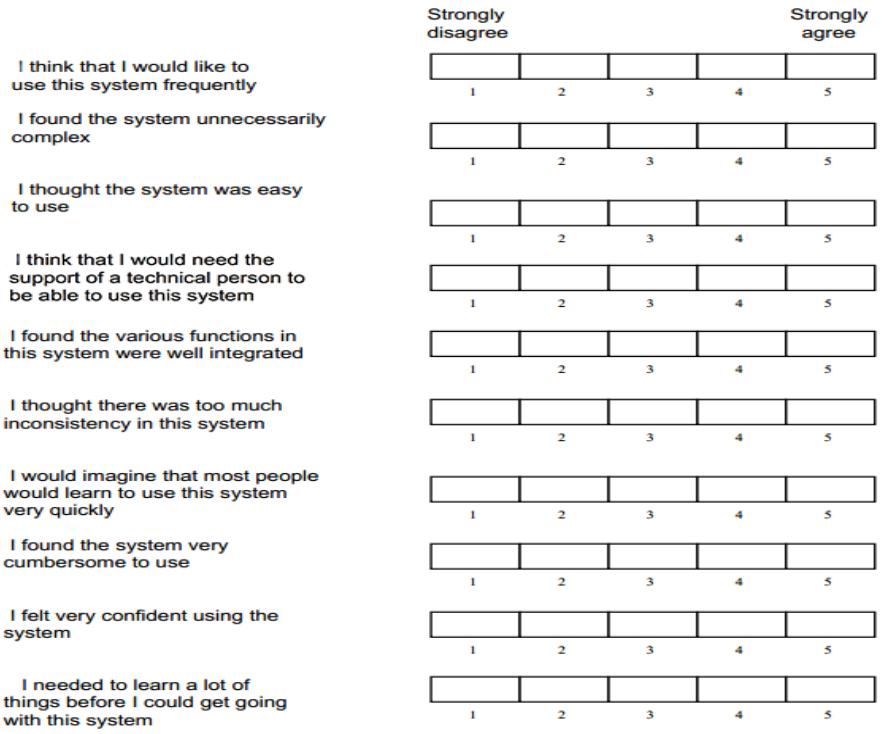
- Brukervennlighet
- Hvor nyttig systemer er
- Om det passer brukerens behov
- Om brukeren vil anvende det
- Kostnad

Med andre ord kan vi si at brukeraksept avhenger av om brukeren ser på systemet som nyttig, anvendbart og likendes i forhold til hva det vil koste. Brukervennlighet blir i økende grad relevant i utvelgelsesprosessen av systemer, da datamaskiner blir mindre kostbare og mer kraftfulle. Shackel (2009) mener kombinasjonen av iterativ evaluering og design for brukervennlighet vil resultere i god brukervennlighet og aksept.

Tidligere ble brukervennlighet bare definert som generaliserte former, som ikke spesifiserte hva brukervennlighet er i kvantitative eller målbare former. Som en reaksjon på dette poengterer Shackel (2009) viktigheten av å lage kvantitative verdier som mål. Ved å bruke kvantitative verdier åpnes muligheten for å lage kriterier for å teste design underveis, og forbedre det ved et iterativt design. Verdiene åpner også for å sette minimumsverdier som definerer om systemet er akseptabelt eller ikke. Tilfredsstiller ikke verdiene minimumsgrensen, er ikke systemet aksepterbart og designet må forbedres. Ved hjelp av disse målbare formene blir det lettere å kontrollere om dataen innhentet fra datainnhentningsteknikker oppfyller målene i brukervennlighetsspesifikasjonen.

3.2.1 Brukervennlighetsskala

SUS(*System Usability Scale*) er en *quick and dirty* metode for å innhente kvantitative rangeringer gitt av en bruker om et system. Som nevnt i 3.2 er det viktig å ha kvantitative verdier for å kunne sette kriterier og teste design underveis, og SUS gir muligheten for dette. SUS brukes i følge Brooke (1996) etter at respondenten har hatt mulighet for å bruke systemet som skal evalueres, men før en diskusjon eller debriefing gjøres. Han mener at SUS har vist seg å være et verdifullt evalueringsverktøy som er både robust og pålitelig. SUS er åpent for bruk i brukervennlighetsevalueringer og har allerede blitt brukt i mange forskjellige forskningsprosjekter og i industrielle evalueringer. Figur 3.3 viser brukervennlighetsskjemaet. Deltageren rangerer hver påstand fra en til fem, hvor én representerer sterkt uenig og fem sterkt enig.



FIGUR 3.3: Brukervennlighetsskala

3.3 Datainnhentingsteknikker

Det finnes flere ulike datainnhentingsteknikker som kan brukes for å innhente data til å evaluere brukervennlighet:

3.3.1 Intervju

Et intervju er i følge Rogers et al. (2011) en samtale med mening. Hvor likt intervjuet er en normal samtale, avhenger av typen intervjumetode brukt. Det finnes fire forskjellige typer intervju: ustrukturert, strukturert, semi-strukturert og gruppeintervju. De tre første er navngitt etter hvor mye kontroll intervjueren har over samtalen, mens gruppeintervju involverer en liten gruppe mennesker som blir veiledet av en møteleder.

Ustrukturert intervju

Et ustrukturert intervju brukes for å holde utforskende samtaler rundt et spesielt emne. Intervjuene går ofte i dybde, og spørsmålene stilt av intervjueren er åpne i den forstand at det ikke forventes noe spesielt format eller innhold av svarene. Selv om intervjuet er ustrukturert, er det viktig å ha en plan over hovedemnene det skal inneholde. Det er en viktig ferdighet å kunne balansere mellom å passe på at spørsmålene blir svart med relevante svar og følge nye retninger det kan gå (Rogers et al., 2011).

Strukturert intervju

I et strukturert intervju stilles det forhåndsdefinerte spørsmål. Disse spørsmålene skal være korte og tydelige. Det er viktig at de samme spørsmålene stilles for alle deltagerne. For at et strukturert intervju skal lykkes, må målene og spørsmålene være tydelig definert og mulig å forstå (Rogers et al., 2011).

Semi-strukturert intervju

Et semi-strukturert intervju kombinerer egenskaper fra både strukturert og ustrukturert intervju. Intervjueren har et skript for veiledning og på denne måten forsikres det at samme emner blir dekket i hvert intervju. Det stilles først forhåndsdefinerte spørsmål, og deretter åpnes det for at intervjuobjektet kan si mer (Rogers et al., 2011).

Fokusgrupper

Det må ikke være et enkelt intervjuobjekt, det er også mulig å foreta intervju i grupper. En form for gruppeintervju er fokusgrupper, der tre til ti mennesker er involvert, og diskusjonen blir ledet av en trent møteleder. Deltagerne er utvalgt for å representere et representativt utvalg (Rogers et al., 2011).

3.3.2 Spørreskjema

Spørreskjema brukes for å innhente demografiske data og brukeres valg. Et spørreskjema, kan som et intervju, ha både åpne og lukkede spørsmål. Det er viktig at spørsmålene er tydelig forklart og at dataen kan analyseres effektivt. Spørreskjema er effektivt når det ønskes å få svar fra en stor mengde mennesker, spesielt hvis menneskene finnes geografisk spredt (Rogers et al., 2011).

3.3.3 Observasjon

Observasjon i en tidlig fase er en god teknikk for å observere brukeres kontekst, oppgaver og mål. En observasjon i en senere fase kan brukes for å evaluere hvor godt en prototype tilfredsstiller disse oppgavene og målene. Brukerne kan bli observert direkte av observatøren, eller ved å ta opp skjermen til brukeren (Rogers et al., 2011).

Direkte observasjon i felten

Det kan være vanskelig for mennesker å forklare konkret hva de gjør for å utføre en oppgave. Av den grunn er det vanskelig for en designer å uthente en fullverdig forklaring ved å kun bruke intervju eller spørreskjema. Ved å observere brukeren i felten kan manglende detaljer fra spørreskjema eller intervju utfylles. En observasjon i felten åpner opp for å få detaljer om hvordan brukere samhandler med hverandre, teknologien og miljøet (Rogers et al., 2011).

Direkte observasjon i kontrollerte miljø

Direkte observasjon i et kontrollert miljø kan enten skje i et konstruert laboratorium, eller i et portabelt laboratorium som kan settes opp der det passer for brukeren. Observasjon i kontrollerte miljø er en mer formell form for observasjon enn for eksempel observasjon i felten, og dette kan fort gjøre brukeren engstelig. For å samle inn dataen brukes for eksempel fotografering, videoopptak og notater. I motsetning til en observasjon i felten, er fokuset på detaljene av hva brukeren gjør (Rogers et al., 2011).

3.3.4 Velge riktige teknikker

Rogers et al. (2011) poengterer viktigheten av å bruke flere teknikker for å få ulike perspektiv. Teknikkene oppsummeres også i en tabell i boken, med fordeler, ulemper, type data teknikken resulterer i, og hvor teknikkene kan brukes. Tabell 3.2 viser en gjengivelse av tabellen. Ved å bruke flere teknikker oppnår investeringen en triangulering, en terminologi brukt for å referere til investigering som skjer fra minst to perspektiver. Trianguleringen sørger for at funnene blir mer kritiske og forsvarlige. Rogers et al. (2011) nevner fire ulike typer av triangulering:

1. **Triangulering av data**
   Dataen er innhentet fra ulike kilder ved ulike tider, i forskjellige steder eller fra forskjellige mennesker.

2. **Investeringstriangulering**
   Forskjellige forskere har blitt brukt for å innhente og tolke dataen.

3. **Triangulering av teorier**
   Bruken av forskjellige teoretiske rammeverk for å vise dataene i funnene.

4. **Metodisk triangulering**
   Å bruke forskjellige datainnhentingsteknikker.

Teknikk	**Hva teknikken er god til**	**Type data**	**Fordeler**	**Ulemper**
Intervju	Finne feil	For det meste kvalitativ data, noe kvantitativ	Intervjueren kan styre intervjuobjektet hvis det trengs. Motiverer til kontakt mellom utviklere og brukere	Tidskrevende. Kunstig miljø kan skremme intervjuobjektet
Gruppeintervju	Skaffe fler synspunkter	For det meste kvalitativ data, noe kvantitativt	Frembringer enigheter og konflikter. Motiverer kontakt mellom utviklere og brukere	Noen karakterer kan være dominante og ta for mye fokus.
Spørreskjema	Svare på spesifikke spørsmål	Kvantitativ og kvalitativ	Kan nå en stor folkemengde uten at det er kostbart	Designet av spørreskjema er avgjørende. Svarprosenten kan være lav. Resulterende svar kan være svar som ikke er det du ønsker
Direkte observasjon i felten	Forstå konteksten av brukeraktiviteten	Kvalitativ	Å observere faktisk arbeid kan gi innsikter andre teknikker ikke kan gi	Veldig tidskrevende. Store mengder data
Direkte observasjon i et kontrollert miljø	For å fange detaljer av hva individer gjør	Kvantitativ og kvalitativ	Kan fokusere på detaljer av oppgaven uten forstyrrelser	Resultatene kan ha begrenser bruk i et normalt miljø ettersom forholdene er kunstige.
Indirekte observasjon	Observere brukere uten å forstyrre oppgavene deres, data blir innhentet automatisk.	Kvantitativ logging og kvalitativ dagbok	Brukere blir ikke forstyrret av datainnhenting. Den automatiske innsamlingen kan foregå over lengre tider	En veldig stor mengde av kvantitative data trenger verktøy for å analyseres. Deltagerne sin hukommelse kan overdrive i dagboken

Tabell 3.2: Oversikt over ulike datainnhentingsteknikker

Kapittel 4

Design og utvikling

Kapittelet vil ta for seg hvordan applikasjonen med navn EinBlick har blitt utviklet. I detaljer vil det beskrive utforming av krav, design, tekniske utfordringer og de syv iterasjonene som har resultert i artefaktet.

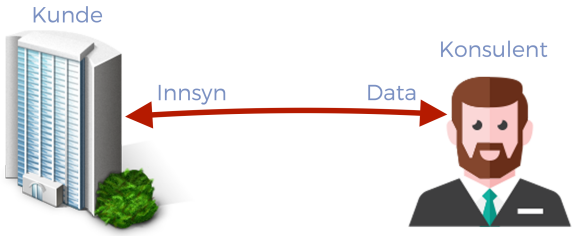
4.1 Brukerscenarioer

I kapittel 1 ble sluttbrukeren av applikasjonen definert, og det ble også kort presentert to ulike brukerscenarioer. I denne seksjonen vil disse to scenarioene forklares ytterligere.

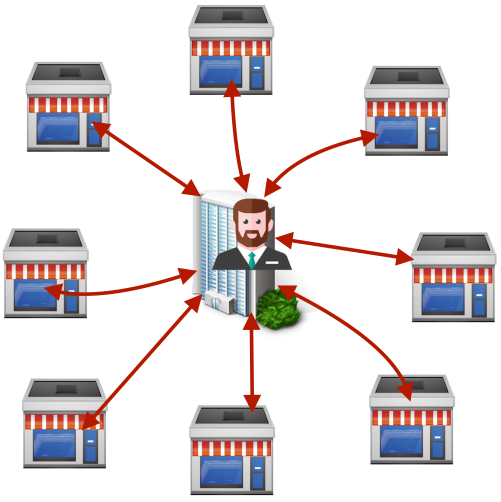
Det første scenarioet er illustrert på figur 4.1. Scenarioet illustrerer et eksempel hvor konsulenthus som videredistribuerer *Enterprise Resurce Planning(ERP)* og *Point of Sale(POS)* systemer, bruker applikasjonen for å gi kunder informasjon basert på kundenes data. Det vil med andre ord si at kunden loggfører forretningsdata ved hjelp av ERP og POS systemer, og har av den grunn store mengder data tilgjengelige. Konsulenthuset kan da tilby kunder å skaffe informasjon basert på denne store mengden transaksjonsdata ved hjelp av EinBlick.

Det andre scenarioet illustrert på figur 4.2 illustrerer et eksempel der en kjede av handelsbutikker som bruker ERP og POS systemer selv tar i bruk applikasjonen. For eksempel kan en markedsansvarlig i kjeden bruke applikasjonen til felles markedsføring, eller til individuell markedsføring for hver enkelt butikk. Et annet eksempel kan være at innkjøpsansvarlig for kjeden bruker applikasjonen for å danne en forståelse for hvilke varer

som må kjøpes inn. Denne typen informasjon kan uthentes ved hjelp av data mining funksjonaliteten i applikasjonen.



FIGUR 4.1: Konsulent benytter seg av EinBlick for å tilby kunde tjenester



FIGUR 4.2: Butikkjede benytter seg av EinBlick

4.2 Brukerscenarioeksempel og domeneekspert

I utviklingsprosessen har konsulenthuset Advania blitt brukt som et eksempel, et konsern med over 10 000 kunder fordelt over verden (Advania, 2015). I dette prosjektet representerer Advania det første brukerscenarioet definert i avsnitt 4.1, et konsulenthus som benytter applikasjonen for å gi kunder informasjon basert på kundenes data. En av de ansatte i Advania ble også tatt med som en domeneekspert, på grunn av hans mange års erfaring med både handelsbransjen, *Point of Sale(POS)* og *Enterprise Resource Planning(ERP)* systemer. I utviklingsprosessen ble det kontinuerlig avholdt møter med domeneekspert for å sørge for at applikasjonen ble optimalisert for sitt domene.

4.3 Krav

Et krav er som nevnt av Vliet (2008) en tilstand eller en mulighet som en bruker behøver for å løse et problem eller oppnå et mål. Han spesifiserer terminologien *requirements engineering* for å vektlegge at kravspesifisering er en iterativ prosess. Det nevnes også at i agile utviklingsprosjekter dukker kravene opp sammen med et kjørende system. Kravene opplyst i denne seksjonen er av den grunn utviklet iterativt i takt med utviklingsprosessen, via møter med kontaktperson og domeneekspert hos Advania.

I følge Vliet (2008) er hovedkildene bak informasjonen for utvikling av kravene brukerne og domenet. Han vektlegger også at det bør brukes flere innhentingsteknikker for å skaffe seg denne informasjonen. Prosjektet har primært benyttet seg av møter hvor prototyper er blitt fremvist. Reaksjonene på prototypen har sørget for en kontinuerlig oppdatert kravspesifikasjon.

4.3.1 Endelige funksjonelle krav

Sommerville (2010) definerer funksjonelle krav som krav for hva et system skal utføre. De endelig funksjonelle kravene for dette prosjektet listes nedenfor.

- I tabellvisningen av data må det være mulighet for å kunne filtrere dataene.
- Det ønskes en funksjonalitet som kan kombinere kolonner fra to eller flere databasetabeller. Et gitt eksempel er at det finnes en kolonne x i tabell a, og en kolonne z i tabell b. Ved kombinering av disse vil en ny kolonne y lages basert på alle feltene i kolonne x og z.
- Det skal være mulighet for brukeren til å kunne skjule hovedmenyen til venstre for å oppnå en full visning
- Applikasjonen må inneholde en veiviser for tilkobling av database. Veiviseren må ha en funksjon for å kontrollere tilkobling. Denne funksjonen skal sørge for at, brukeren ikke får kommet videre i veiviseren hvis tilkoblingen ikke godkjennes.
- Den må også inneholde en veiviser der brukeren kan bestemme hvilke kolonner det skal visualiseres på.

- Det skal være en hjelpefunksjon som viser innhold basert på hvilken visning som er aktiv. For eksempel skal tabellvisningen ha annerledes hjelpeinnhold enn visualiseringsvisningen.
- Applikasjonen skal inneholde datainnsynsfunksjonalitet som kan gi brukeren informasjon om hvilke sett med varer som selges ofte eller sjelden sammen(*itemset mining*). I tillegg skal det være en funksjonalitet som kan fortelle brukeren hvor stor sannsynlighet det er for at vare x kjøpes dersom vare y kjøpes(*association rules*).
- Knapper skal inneholde hjelpetekster som forteller deres funksjon når musen holdes over knappen.

4.3.2 Endelige ikke-funksjonelle krav

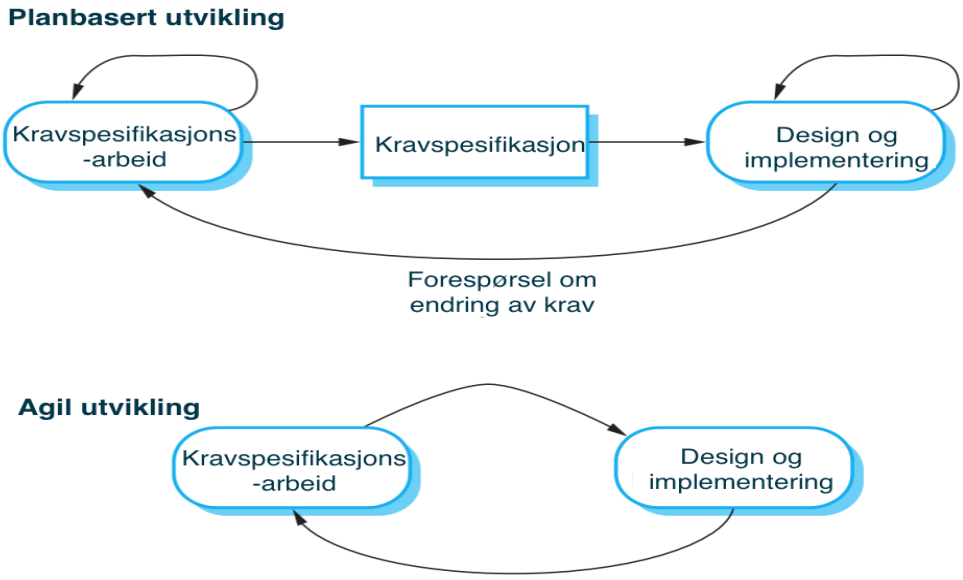
Sommerville (2010) definerer ikke-funksjonelle krav som krav som ikke er direkte knyttet til tjenester som leveres av systemet til brukerne. De endelig ikke funksjonelle kravene for dette prosjektet listes nedenfor.

- Folk med normal IT bakgrunn må kunne bruke programmet uten noe særlig opplæring.
- I følge Brooke (1996) øker aksepten for en applikasjon betraktelig så fort SUS poengsummen blir 70 eller høyere, og i tillegg er 68 gjennomsnittet. Applikasjonen må av den grunn ha en SUS score på over 70.
- Brukergrensesnittet til applikasjonen bør ha et godt design med god brukervennlighet.

4.4 Design

Sommerville (2010) definerer programvaredesign som en kreativ prosess der det defineres en beskrivelse av strukturen på programvaren som skal implementeres, datamodellene og strukturen brukt av systemet, grensesnittene mellom systemkomponentene og i noen tilfeller algoritmene som blir brukt. Han poengterer også at designere ikke kommer frem til et endelig design øyeblikkelig, men at programvaredesign er en iterativ prosess. I følge Sommerville (2010) går kravspesifikasjon- og designprosessen under en prosess ved bruk

av en agil tilnærming, i motsetning til i en planbasert tilnærming hvor programvaredesignet kommer som et eget steg etter kravspesifikasjonen som vist på figur 4.3.



FIGUR 4.3: Forskjellen på kravspesifikasjon- og designprosess i en agil og en planbasert utvikling(figur fra (Sommerville, 2010))

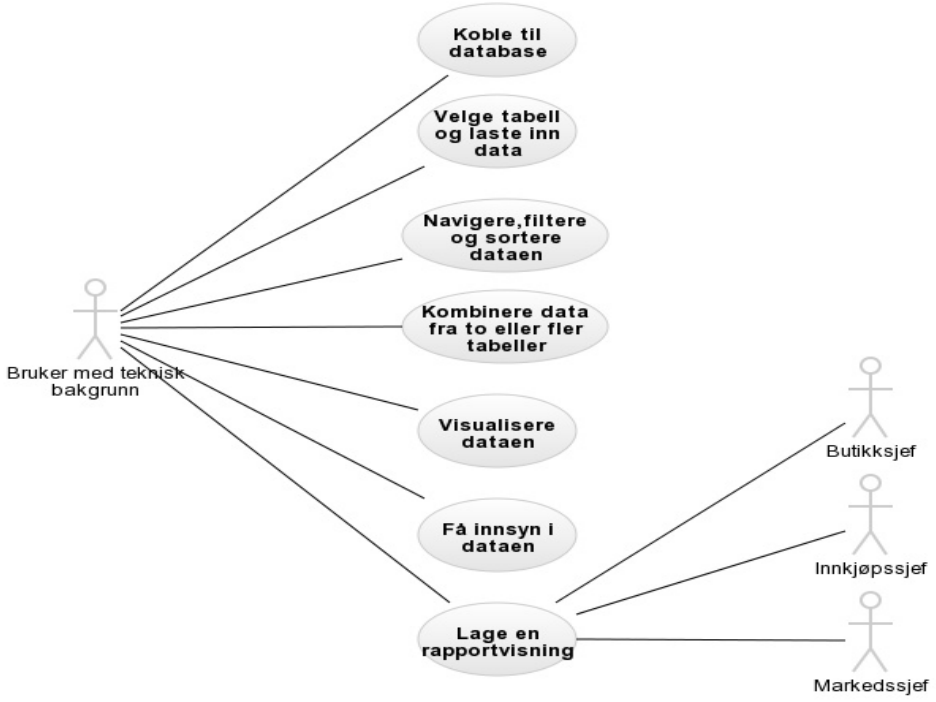
4.4.1 Brukertilfeller

Brukertilfeller brukes ofte til å utforme kravspesifikasjonen, og hvert brukertilfelle kan ofte i følge Sommerville (2010) representere scenarioer som en bruker forventer av systemet. Av den grunn skal brukertilfellene representere de mulige interaksjonene i kravspesifikasjonen. Figur 4.4 viser en oppdatert versjon av alle de mulige interaksjonene som gjenspeiler kravspesifikasjonen. Det er viktig å merke seg at butikk-, innkjøp- og markedssjefer også er inkludert i diagrammet. Dette for å illustrere at oppsummeringsvisningen diskutert under 4.5.3 genereres av den tekniske brukeren, men informasjonen er tiltenkt å brukes av markedssjefer, butikksjefer og innkjøpssjefer.

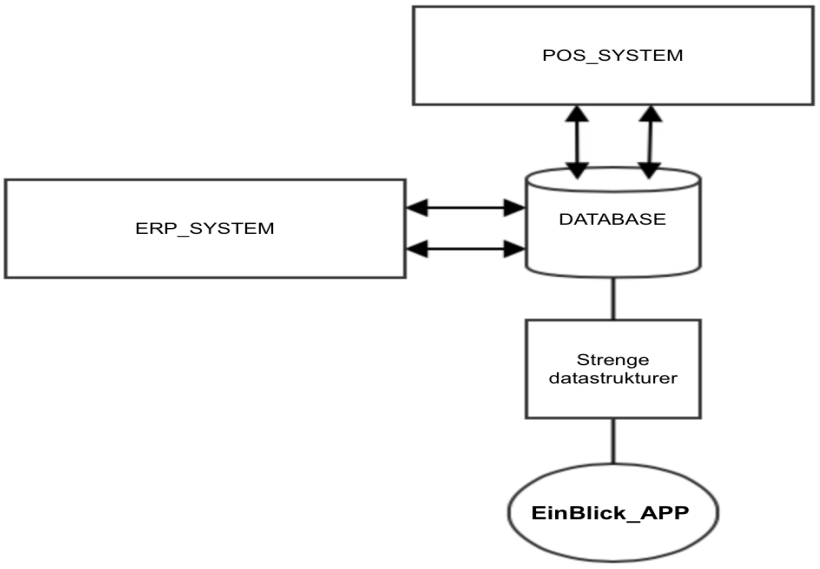
4.4.2 Kontekstdiagram

Kontekstdiagrammer skal i følge Vliet (2008) illustrere samspillet systemet har med miljøet og enhetene i miljøet. Figur 4.5 viser konteksten av EinBlick i miljøet til et firma som

benytter seg av *Enterprise Resource Planning(ERP)* og *Point of Sale(POS)* baserte systemer. ERP og POS systemene skriver og leser til en database med streng datastruktur, som brukere av EinBlick kobler til for å uthente data.



FIGUR 4.4: Brukertilfeller



FIGUR 4.5: Kontekstdiagram

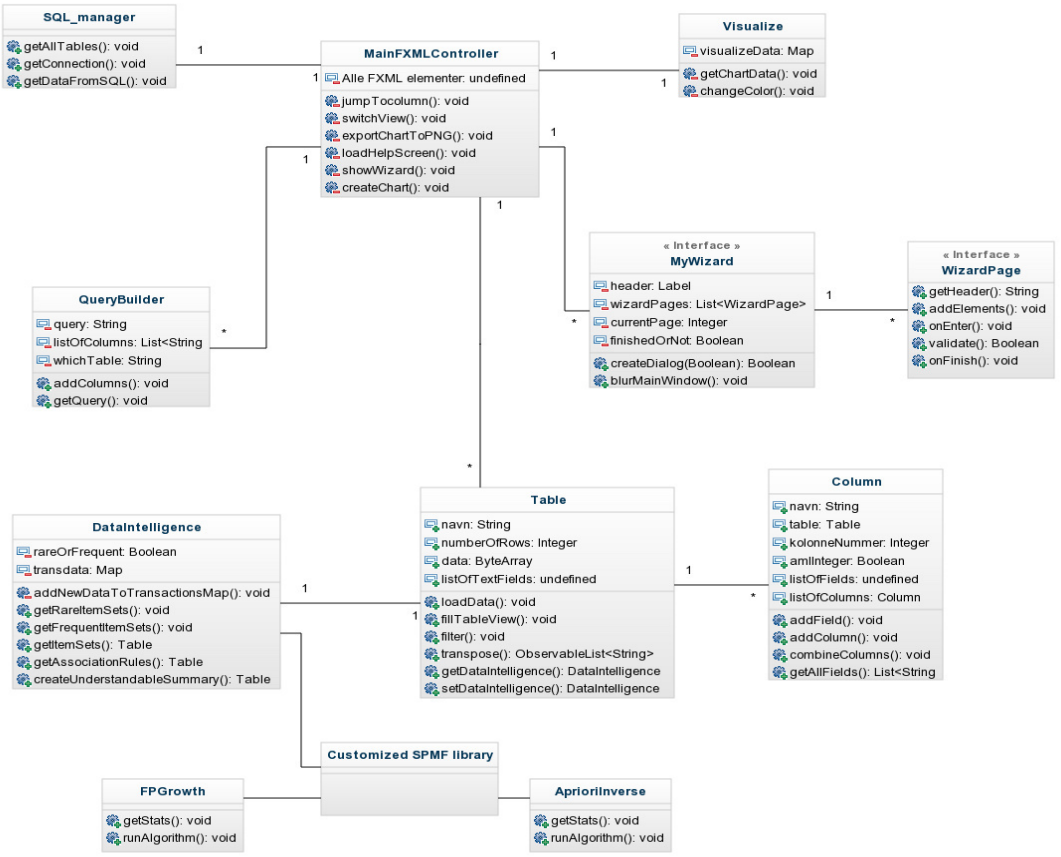
4.4.3 Klassediagram

Sommerville (2010) definerer bruken av klassediagram til å benyttes ved utvikling av objektorienterte systemer. Diagrammene skal vise klassene i systemet og deres relasjoner. På figur 4.6 vises et klassediagram som representerer nåværende klasser i EinBlick. Ut ifra diagrammet illustreres følgende forhold mellom klassene i systemet.

- Hovedkontrolleren for applikasjonen MainFXMLController har ett objekt SQL_manager og ett objekt Visualize.
- Ett objekt tabell har en eller flere av objektet kolonne.
- Ett objekt kolonne har kun ett objekt tabell.
- Ett objekt tabell har kun ett objekt DataIntelligence.
- Objektet DataIntelligence er tilknyttet en tilpasset versjon av SPMF biblioteket.
- Ett objekt som følger *interfacet* MyWizard har ett eller flere objekter som følger *interfacet* WizardPage.
- Ett objekt WizardPage har kun ett objekt MyWizard.

4.5 Tekniske utfordringer

Underveis i utviklingen ble flere tekniske utfordringer møtt. Først måtte det utvikles funksjonalitet for å kunne aksessere tabelldata fra de store databaseleverandørene. Videre viste det seg at visualisering av relasjonsbasert transaksjonsdata skulle skape flere utfordringer. Datahåndteringen i JavaFX skapte også problemer, og det måtte gjøres en total omstrukturering på hvordan applikasjonen håndterte data fra relasjonsdatabaser. Etter dette var på plass krevde biblioteket *Sequential Pattern Mining Framework* mange tilpasninger, og det ble brukt mye tid på å forbedre biblioteket. Alle disse tekniske utfordringene diskuteres ytterligere nedenfor.



FIGUR 4.6: Klassediagram

4.5.1 Datatilgang

Det er viktig at applikasjonen skal kunne koble til databaser fra de aller fleste leverandører. Dermed tilrettelegges det for at applikasjonen kan brukes uavhengig av hvilken databaseleverandør brukeren har lagret data i. I applikasjonen er det lagt til støtte for følgende databaseleverandører:

- Microsoft SQL
- Oracle Database
- MySQL

Ved hjelp av *Java Database Connectivity(JDBC)* API åpnes det opp for å aksessere dataen fra de ulike databasene. JDBC er en API for å aksessere omtrent nesten alle typer tabellbaserte data. API-en tilbyr klasser og grensesnitt(*interface*) skrevet i Java, som gir utviklere mulighet for å skrive gode databaseapplikasjoner kun ved hjelp av

Java(Fisher & Bruce, 2003). Utfordringen blir selvfølgelig å håndtere spørrespråket til de ulike databasestandardene. For å løse dette er det i dette prosjektet utviklet et eget objekt som generer spørringene for det relevante spørrespråket. Objektet har ulike "case" og bygger opp spørrestrengen etter hvilken database brukeren ønsker å koble til. Objektet returnerer til slutt den ferdige spørrestrengen. Objektet kan sees i klassediagrammet på figur 4.6 med navn QueryBuilder.

4.5.2 Utfordringer med visualisering av relasjonsbasert transaksjonsdata

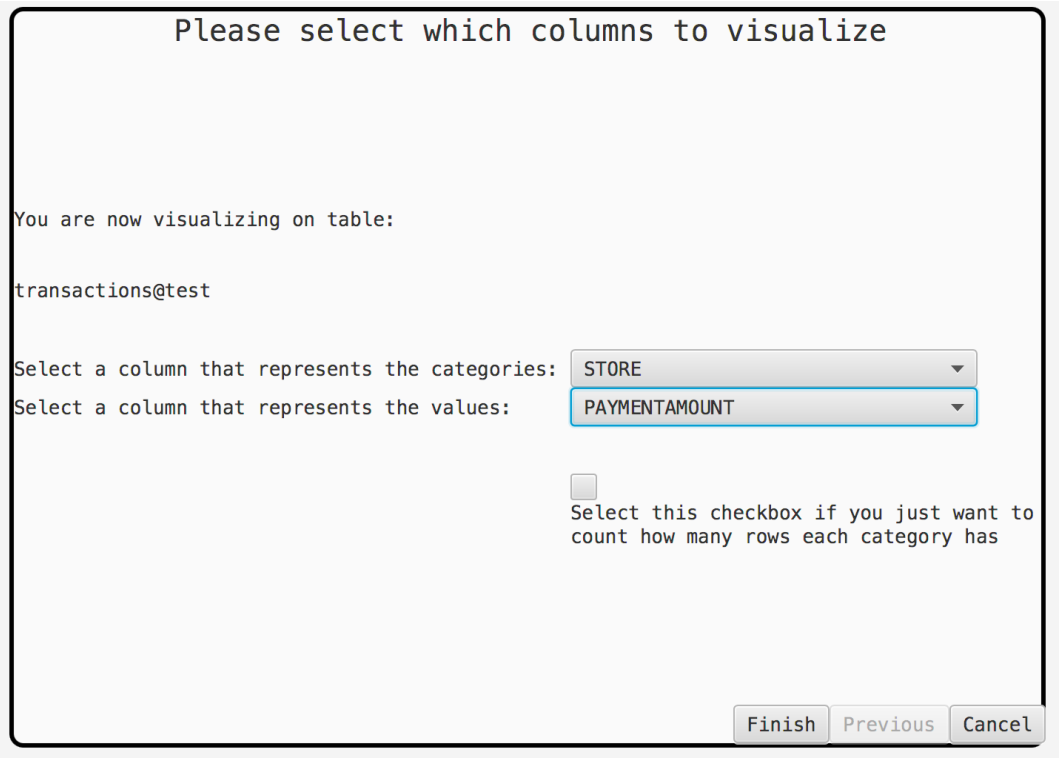
Ved utvikling av visualiseringsfunksjonaliteten ble det oppdaget flere problemer og utfordringer. I denne seksjonen vil disse beskrives.

4.5.2.1 Hvilke kolonner som skal visualiseres

Ettersom brukeren kobler til databasetabeller med mange kolonner, må det i visualiseringsfunksjonaliteten være en funksjon hvor brukeren velger hvilke kolonner som skal visualiseres. I visualiseringene implementert i denne applikasjonen er det støtte for to dimensjoner, hvor den ene representerer kategorien, og den andre verdiene på dataen. For eksempel vil butikkeidentifikatoren representere kategorien, og hvor mye butikken har solgt for representere verdien. For å løse dette i praksis velger brukeren selv, ved hjelp av veiviseren, hvilke kolonner visualiseringen skal basere seg på. På figur 4.7 illustreres et eksempel hvor brukeren har gjort nettopp dette.

4.5.2.2 Kombinere rader som tilhører samme kategori

Dessverre lagrer de fleste forretningssystemer én rad for hver transaksjon i databasetabeller, og av den grunn trengs det også en funksjonalitet som grupperer transaksjoner sammen. For eksempel ønskes det ikke en stolpe for hver eneste transaksjon i databasen i et stolpediagram, men heller en stolpe for hver butikk. Av den grunn er det lagt til en funksjonalitet som grupperer radene tilhørende samme kategori. På figur 4.8 kan kildekoden for denne funksjonaliteten sees. Dette problemet er løst ved hjelp av et *Lambda* uttrykk.



FIGUR 4.7: Bruker velger hvilke kolonner som ønskes visualisert

```
data.entrySet().stream().map(entry -> new XYChart.Data(entry.getKey(),
    entry.getValue())).forEach(barChartData::add);
```

FIGUR 4.8: Kildekode for kombinering av rader tilhørende samme kategori

4.5.2.3 Visualisere antall rader

I enkelte tilfeller ønskes det ikke å visualisere på total sum av for eksempel transaksjoner, men heller hvor mange transaksjoner hver butikk har. For å løse dette i visualiseringen er det lagt til en avhukningsboks, hvor brukerne kan velge om det skal telles antall rader i stedet for å summere verdiene. På figur 4.7 kan denne avhukingsboksen sees.

4.5.3 Sequential Pattern Mining Framework

Applikasjonen benytter seg av en egentilpasset versjon av biblioteket *Sequential Pattern Mining Framework(SPMF)*, diskutert i avsnitt 2.2.2. Biblioteket benyttes for å finne ofte eller sjeldne kjøpte produktsett og assosieringsregler som diskutert i avsnitt 2.2. Ved implementering av kildekoden tilgjengelig i dette biblioteket ble det primært funnet to problemer.

Det første problemet er hvordan biblioteket håndterer data. SPMF håndterer data ved hjelp av flate tekstbaserte filer, og denne tilnærmingen gjør håndteringen både tungvinn og treg. For å forbedre datahåndteringen er applikasjonens versjon av biblioteket videreutviklet til en minnebasert, tilnærming hvor transaksjonsdataene håndteres ved hjelp av et map. Et map er enkelt forklart en datatype designet for å effektivt oppbevare, og uthente verdier basert på unike identifikatorer. Spesifikt bruker maps i Java *keys* og *values*. Identifikatoren(*keys*) må være unik, og den kan inneholde flere verdier(*values*). En viktig egenskap med maps er å kunne bruke funksjonen *get*. Dette åpner opp for muligheten å spørre på mappet med identifikator som parameter, og det vil da returneres verdiene assosiert med denne unike identifikatoren(Goodrich, Tamassia & Morrisey, 2014).

Det andre problemet er hvordan SPMF baserer seg på unike produktidentifikasjoner i integer format, og i en transaksjonsdatabase er ofte dette ikke tilfellet. *Enterprise Resource Planning(ERP)* systemet *Microsoft Dynamics AX(AX)* for eksempel, benytter en kombinasjon av tekst og tall som identifikator for produkter. Av den grunn blir det ikke mulig å bruke algoritmene tilgjengelig i biblioteket, da datagrunnlaget ikke er i et integerbasert format. Ved personlig kontakt med Philippe Fournier-Viger ble det forklart at biblioteket er utviklet på denne måten for å være mer effektivt (personlig kommunikasjon, 03. november, 2014). En integer verdi krever betydelig mindre datakraft enn hva en streng gjør. Hvis algoritmene skal analysere mange tusen transaksjoner vil strengverdier ofte forårsake at algoritmene krasjer. For å håndtere strengverdiene fra transaksjonsdatabasene til ERP systemer, som for eksempel AX, er det av den grunn i EinBlick utviklet kode som mapper strengene opp mot tilfeldige integerverdier ved hjelp av et map. Med andre ord får hvert unike produkt en egengenerert integerverdi som representerer produktet, og på denne måten kan hyppige eller sjeldne produktsett og assosieringsregler detekteres effektivt ved å bruke integerverdier.

I praksis løses de to problemene diskutert ovenfor ved å ha følgende maps:

- itemMap
- invertedItemMap
- transdata

itemMap inneholder produktets originale identifikator, og den oppdiktede integer identifikatoren. **invertedItemMap** inneholder akkurat det samme, men i motsatt rekkefølge.

transdata inneholder transaksjonsID-er og produktene for hver transaksjon. På figur 4.9 vises kildekoden for hvordan en transaksjon legges til i transdata, hvordan det først sjekkes om produktet finnes i itemMap, og eventuelt hvordan et produkt får en ny generert integer produktidentifikator.

Når SPMF skal finne de frekvente produktsettene, sendes transdata som parameter til algoritmen, i stedet for en tekstfil. For å presentere funnene til SPMF, brukes invertedItemMap til å hente tilbake de originale produktID-ene. Denne tilnærmingen gjør at applikasjonen kan nytte av den betydelig økte hastigheten ved å bruke integerverdier, samtidig som den støtter transaksjonsdatabaser som benytter produktidentifikatorer i strengformat.

Ettersom SPMF både baserer seg på og presenterer funn med unike produktidentfikatorer, blir funnene ikke veldig brukervennlige. Assosieringsreglen 1394⟶1455 gir ikke så mye informasjon for brukeren, med mindre hun husker alle produktidentifikatorene for produktene som er i salg. Av den grunn ble det utviklet en ekstra oppsummeringsfunksjonalitet som konverterer funnene til mer forståelig informasjon. Funksjonalitetens parameter er en produktdatabasetabell, som inneholder felter med produktID-er og produkttekst, hvor produkttekst beskriver hva produktet er, med klartekst. Deretter itererer metoden igjennom samtlige produktID-er, og erstatter dem med deres beskrivende produkttekst. Overflødige felter fjernes, og det legges til en mulighet for å dobbeltklikke på en assosieringsregel, eller et produktsett, for å få opp en forklarende tekst. På figur 4.10 og figur 4.11 illustreres det hvor mye mer forståelig, og oversiktlig oppsummeringsvisningen er. I visningen grupperes produktsettene i faner etter hvor mange produkter produktsettet inneholder, og dette gjør tabellvisningen mer oversiktlig.

4.5.4 Matrisetransponering *(Matrix Transpose)*

JavaFX håndterer tabelldata i et radformat, og dette skaper problemer ved utvikling av funksjonaliteten for kombinering av data i kolonner, diskutert i avsnitt 4.3.1. Det er ikke mulig å hente ut alle feltene i en kolonne i standard JavaFX, og av den grunn bruker EinBlick egenutviklede objekter som håndterer tabelldataen i et kolonneformat. Dette skaper igjen problemer når dataen skal visualiseres i grafer eller tabellvisninger, ettersom JavaFX kun støtter et radformat. For å snu dataen tilbake til et radformat før visualisering, brukes matrisetransponering. Dataene blir med andre ord lagret i et

```
void addNewDataToTransactionsMap(String key, String name) {
    Item item;
    if (itemMap.containsKey(name)) {
        item = itemMap.get(name);

    }
    else {
        item = new Item(name, createdProductNumber);
        itemMap.put(name, item);
        createdProductNumber++;
        invertedItemMap.put(item.createdInt, item.ID);
    }

    if (transdata.containsKey(key)) {
        if (transdata.get(key).contains(item)) {

        } else {
            transdata.get(key).add(item);
        }

    } else {
        List<Item> transaction = new ArrayList();
        transaction.add(item);
        transdata.put(key, transaction);

    }
}
```

FIGUR 4.9: Kildekode for håndtering av transaksjonsdata

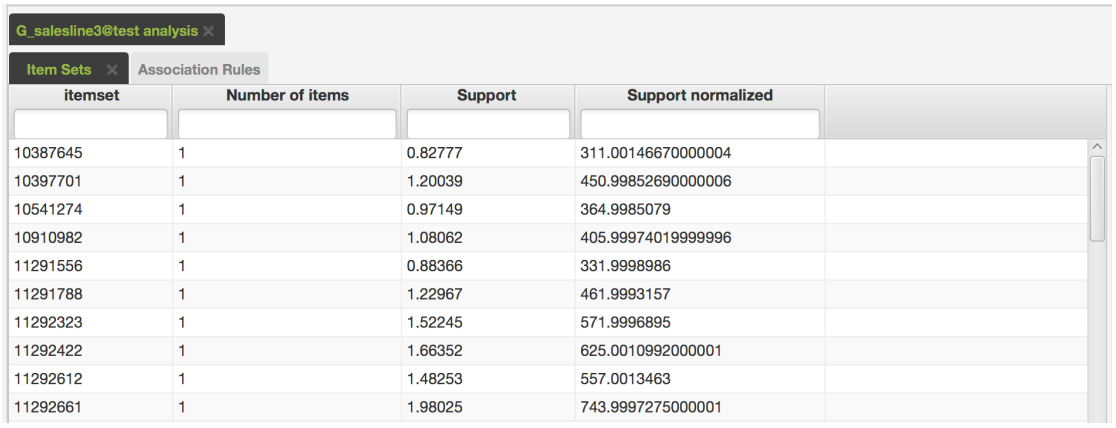
transponert format som vist i matrise m^t på figur 4.12. Før visualisering transponeres dataen tilbake til et radformat som vist i matrise m(MathWorks, 2015)

4.6 Verktøy

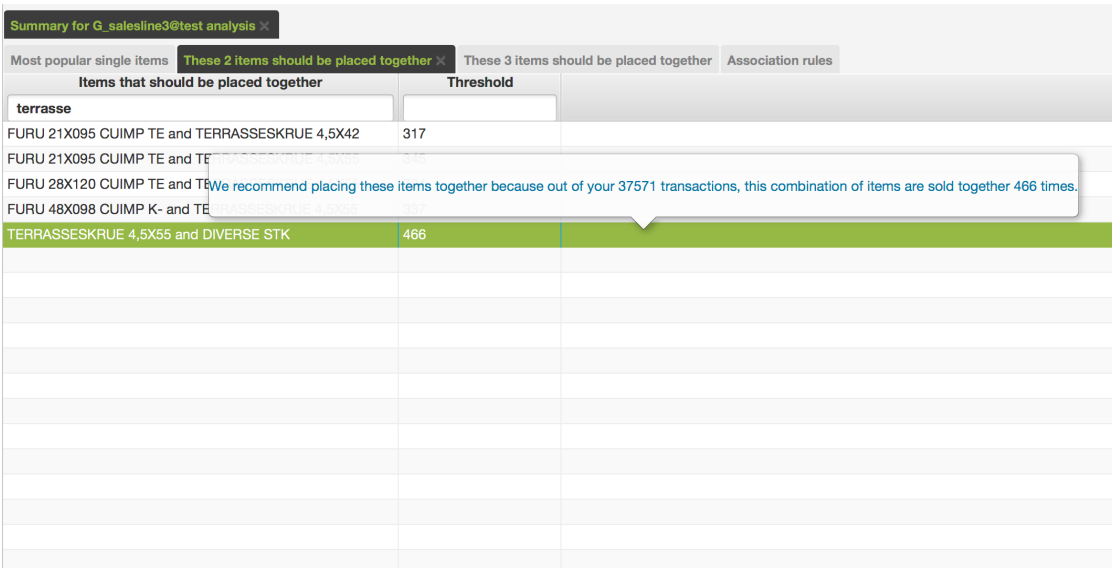
Under prosjektet ble det benyttet flere ulike verktøy, og i dette avsnittet vil de ulike presenteres.

4.6.1 Programmeringsspråk

Java og *Graphical User Interface(GUI)* plattformen JavaFX ble valgt som programmeringsspråk. Årsaken til dette er at JavaFX har en betydelig bredere støtte for visualiseringer enn hva alternative rammeverk som for eksempel Microsoft .NET har. I tillegg, er



FIGUR 4.10: Normalvisning av datainnsynsfunksjonaliteten



FIGUR 4.11: Oppsummeringsvisning

$$m^t = \begin{pmatrix} a & d & g \\ b & e & h \\ c & f & i \end{pmatrix}$$

$$m = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

FIGUR 4.12: Transponert og ikke-transponert matrise

Java i motsetning til .NET plattform, uavhengig. Dette åpner for muligheten til å kjøre Java applikasjoner på alle operativsystem med støtte for Java.

4.6.2 Personal Kanban

Kanban er en teknikk for å administrere utviklingsprosesser på en høyst effektiv måte. En utviklingsprosess kan sees på som et rør, hvor funksjonalitetskrav kommer inn i røret, og improvisert programvare kommer ut. I utviklingsprosessen(røret) er det ofte en flaskehals som forhindrer flyten i prosessen. Dermed blir utviklingsprosessen aldri mer effektiv enn flaskehalsen. Eksempelvis kan testere utgjøre flaskehalsen dersom de kun klarer å teste fem funksjoner i uken, når utviklerne leverer ti. Hvis utviklerne fortsetter å levere ti funksjoner til testerne, vil køen for testing hope seg opp. Denne opphopingen fører til at testerne blir stresset, og tar snarveier for å bli ferdige med den store køen. Verktøyet Kanban håndterer dette ved å redusere arbeidet, så utviklerne tilpasser seg testerne. I stedet for å fortsette å levere funksjonalitet til testerne, kan de for eksempel bistå i testingen. Kanban gjennomfører dette ved å ha en stor tavle, som vist på figur 4.13. De grønne feltene representerer arbeid, og plasseres i kolonner som representerer faser i utviklingsprosessen. Hver kolonne har et maks antall lapper som kan plasseres. Et maks antall i hver kolonne forhindrer overproduksjon, og flaskehalser tydeliggjøres. Ved å fremvise flaskehalsene kan disse adresseres, slik at de ikke går ut av kontroll(Peterson, 2009).

Benson (2009) nevner fire steg for å komme i gang med Kanban:

- **Lag et kart over arbeidsflyt**
  Kartet representerer flyten fra øyeblikket du starter, til øyeblikket du er ferdig. Kartet inneholder arbeid som skal gjøres, arbeid som holder på å bli gjort og arbeid som er ferdig.

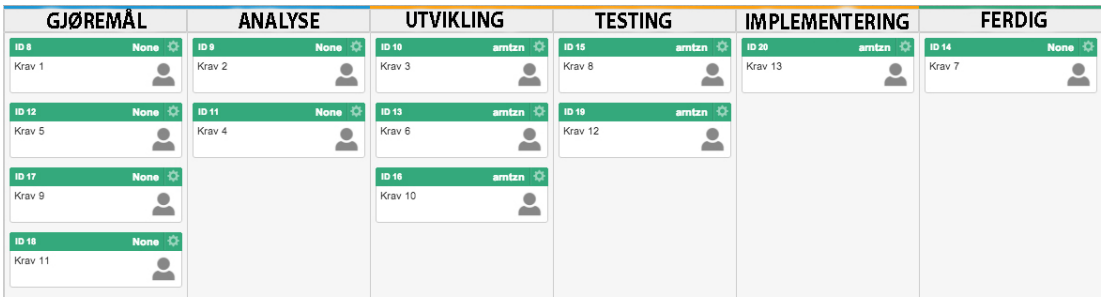
- **Utvikle en logg over gjøremål**
  Gjøremålene er alt av arbeid som skal gjøres. Skriv ned alle gjøremålene på post-it lapper, og fest alle på gjøremålskolonnen i tabellen.

- **Etabler en begrensning for gjøremål i arbeid**
  Begrensningen skal representere hvor mye arbeid du kan gjøre samtidig. Mennesker gjør, i følge Benson (2009), ofte ting halvferdig, og et viktig punkt i *Personal*

Kanban er å finne den riktige mengden arbeid du kan gjennomføre samtidig. I første omgang kan det settes et vilkårlig nummer, som senere tilpasses.

- **Begynn å utføre arbeid**

  Hent arbeid fra gjøremålskolonnen, og begynn å arbeide.



FIGUR 4.13: Skjermbilde av kartet over arbeidsflyten ved bruk av Kanbanize

I dette prosjektet er Kanban valgt, da verktøyet støtter oppunder en iterativ og inkrementell prosess. Den inkrementelle og iterative prosessen kreves for å kunne følge rammeverket *Design Science* sin retningslinje *Design som en søkeprosess* diskutert i avsnitt 3.1. Ved å følge den inkrementelle prosessen sørger Kanban for at utviklingsprosessen blir utført på en høyst mulig effektiv måte. I tillegg beviser Middleton og Joyce (2012) i deres artikkel forbedrede resultater i en utviklingsprosess, dersom det tas i bruk Kanban. I motsetning til den kjente agile metoden Scrum, er Kanban en *lean* utviklingsmetode. Metoden sørger for at det hele tiden kontrolleres hvor mye arbeid som utføres samtidig, samt at det ikke overstiger den definerte grensen.

4.6.3 Andre verktøy

I prosjektet ble det i tillegg tatt i bruk andre verktøy som ikke trenger ytterligere forklaring:

- Oracle, MySQL og MSSQL databaser.
- Git, et verktøy for versjonskontroll.
- Evernote, verktøy for notater.
- TeamViewer, verktøy for skjermdeling og kommunikasjon.
- Kanbanize, en portal for å kunne benytte seg av Kanban

4.7 Programvarevalidering

Programvarevalidering er som Sommerville (2010) diskuterer, ment for å vise at systemet er i overensstemmelse med kravspesifikasjonen, og at det oppfyller kundens krav. Han nevner også at dersom systemet ikke er et lite program, bør det ikke testes som en enkelt enhet. Komponentene av systemet bør heller testes enkeltvis underveis i utviklingen, og til slutt bør systemet med alle komponentene implementert, testes med kundens data.

Dersom det brukes en inkrementell utviklingsprosess, mener Sommerville (2010) at hvert inkrement bør testes etter det utvikles, og inkrementet bør testes mot de relevante kravene. Avslutningsvis i utviklingsprosjektet holdes det en aksepttest, som pågår helt til kunden, og utvikler er enige om at systemet er akseptabelt i henhold til kravene.

4.8 Iterasjoner

Som nevnt i avsnitt 4.6.2 benyttet utviklingsprosessen seg av teknikken *Personal Kanban*, en agil håndteringsmetode for å optimalisere arbeidsflyt og effektivisering (Peterson, 2009). Utviklingsprosessen ble oppdelt i inkrementelle iterasjoner, hvor en inkrementell prosess i følge, Sommerville (2010), er en prosess hvor programvaren blir utviklet i inkrementer, og hvor kunden spesifiserer hvilke krav som skal være med i neste inkrement. Etter hver iterasjon underveis i prosjektet ble det avholdt en tilbakemeldingssesjon hvor utvikler og domeneekspert var til stede. Domeneekspert kom med tilbakemeldinger, og kravspesifikasjonen ble oppdatert. Ettersom prosessen var inkrementell, ble det videre avtalt hvilke krav som skulle utvikles under den neste iterasjonen. Avslutningsvis ble det avholdt en systemvalidering, hvor domeneekspert godkjente hvorvidt applikasjonen gjorde det den var ment å gjøre.

4.8.1 Første iterasjon

Første iterasjon ble brukt for å utvikle en *Proof of Concept(POC)* som inneholdt følgende funksjonalitet:

- Forenklede visualiseringsmuligheter, basert på fastbestemte data ved hjelp av PieChart, BarChart og LineChart.

- Mulighet for å koble til en database, og hente ut data fra tabell, samt vise den i en tabellvisning.
- Mulighet for å velge hvilke kolonner som skal visualiseres.

Etter iterasjonen ble det avholdt en tilbakemeldingssesjon der PoC ble fremvist. Det ble mottatt positive tilbakemeldinger fra domeneekspert, og han mente at applikasjonen allerede var på god vei. Domeneeksperten ytret et ønske om en funksjonalitet som kan kombinere alle felter fra to eller flere kolonner fra to eller flere tabeller. Det ble forklart at bedrifter ofte har flere ulike versjoner av et *Enterprise Resource Planning(ERP)* systemer. Dette medfører ofte at bedriften har flere ERP databaser, og av den grunn kan det være interessant for bedriften å kunne kombinere dataen fra to eller flere tabeller.

4.8.2 Andre iterasjon

Under den andre iterasjonen, ble ønsket kombineringsfunksjonalitet diskutert i avsnitt 4.8.1 utviklet. Underveis i utviklingen ble det funnet et vesentlig problem med hvordan JavaFX håndterer tabellbasert data som diskutert i avsnitt 4.5.4. Dersom et program skal kunne kombinere alle feltene i to eller flere kolonner, må dataene håndteres i et kolonneformat i stedet for et radformat, og av denne grunn måtte datahåndteringen restruktureres. Det ble brukt betraktelig med tid på å bygge opp objekter som håndterte dataene som kolonner. I tillegg måtte dataene transponeres tilbake før de igjen kunne visualiseres. For å løse dette ble det tatt i bruk matrise transponering, som også diskutert i avsnitt 4.5.4.

Ved fremvisning av funksjonaliteten ga domeneekspert generelt god tilbakemelding. Frem til nå hadde utviklingsarbeidet primært bestått av koding, og det var gjort lite med tanke på design. I tillegg hadde mesteparten av funksjonaliteten fra kravspesifikasjonen blitt utviklet i separate moduler for å ikke blande kode. Av den grunn uttrykte domeneeksperten under tilbakemeldingssesjonen ønske om at all funksjonalitet kunne implementeres i en applikasjon, og at det skulle fokuseres på design. I tillegg ytret domeneeksperten ønske om funksjonalitet for filtrering av data i tabellvisning.

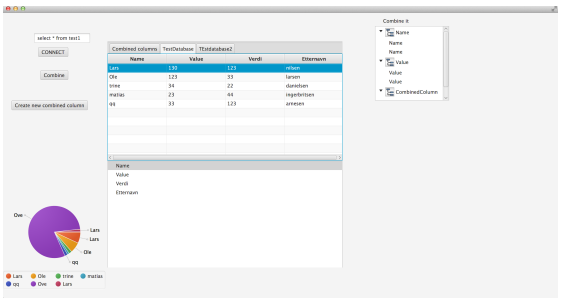
4.8.3 Tredje iterasjon

Den tredje iterasjonen ble benyttet for å utvikle én applikasjon, som inneholdt all tidligere fremvist funksjonalitet med et godt design. På figur 4.14 og figur 4.15 illustreres forskjellen på brukergrensesnittet fra andre til tredje iterasjon. Brukergrensesnittet har en hovedmeny til venstre som skifter mellom ulike visninger i applikasjonen. Eksempelvis er det tabellvisning og visualiseringsvisning. I tillegg er det en innholdsbasert meny på toppen som varierer etter hvilken visning som er aktiv.

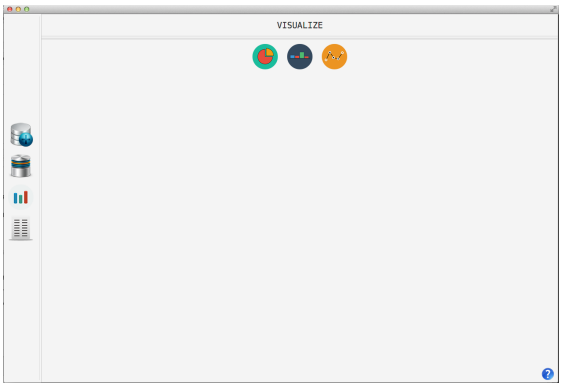
Når applikasjonen fremvises til domeneekspert under tilbakemeldingssesjonen, kommer domeneekspert lite kritikk. Han uttrykker at han er veldig fornøyd, og at designet er betydelig forbedret. All funksjonalitet fra modulene er ennå ikke implementert, men dette vil arbeides med i de neste iterasjonene. Det mottas på dette tidspunktet en rekke nye funksjonalitetskrav som legges til i kravspesifikasjonen. Kravene er gjengitt nedenfor, og rekkefølgen representerer deres prioritet:

- En veiviser for tilkobling av database ønskes.
- En veiviser der brukeren har mulighet for å velge hvilke kolonner som skal visualiseres ønskes.
- Hovedmenyen til venstre bør ha mulighet for å minimeres, slik at applikasjonen kan sees i en fullskjermvisning.
- Filtreringsfunksjonalitet som nevnt i avsnitt 4.8.2.
- All funksjonalitet vist i tidligere moduler må legges til i hovedapplikasjonen.
- Utvikle en hjelpefunksjonalitet som viser innhold basert på hvilken visning som er aktiv.

I tillegg til funksjonalitetskravene kom domeneekspert i samarbeid med Advania med en retningslinje til applikasjonen. De ytret ønske om at applikasjonen skal kunne brukes av folk med normal IT bakgrunn, uten for mye opplæring.



FIGUR 4.14: Skjermbilde av applikasjonen etter andre iterasjon



FIGUR 4.15: Skjermbilde av applikasjonen etter tredje iterasjon

4.8.4 Fjerde iterasjon

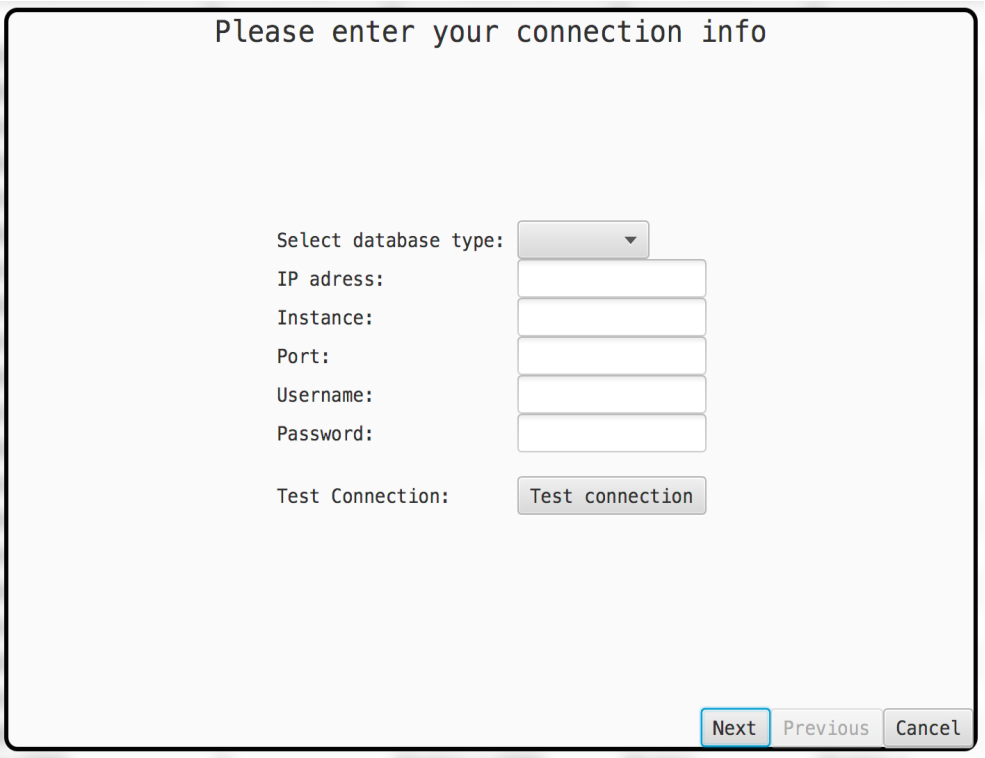
Under den fjerde iterasjonen ble all ønsket funksjonalitet lagt til, både veivisere, mulighet for å minimere hovedmeny, hjelpeskjermer og filtreringsfunksjonalitet. Det gjenstår fortsatt noe funksjonalitet fra de tidligere modulene. Veivisere er nesten ferdig, men det oppstår problemer med validering, og hjelpeskjermene er ikke helt ferdige. Det ble i tillegg oppdaget utfordringer med tanke på hvordan forretningssystemer lagrer transaksjonsdata i relasjonstabeller. Det ble brukt mye tid på å løse dette på en best mulig måte, som diskutert i avsnitt 4.5.4.

Under tilbakemeldingssesjonen kommer domeneekspert med følgende funksjonalitetskrav til neste gang:

- Utvikle ferdig veivisere
- Legge til ikke-implementert PoC funksjonalitet
- Hvis mulig, utvikle en drill down funksjonalitet i PieChart.
- Fullføre hjelpefunksjonen
- Kombinering av kolonner trenger mer arbeid når det gjelder design

4.8.5 Femte iterasjon

Under denne iterasjonen ble alle funksjonalitetskrav bortsett fra *drill down* funksjonalitet ferdigstilt. Denne funksjonaliteten anses som for tidskrevende, og domeneekspert er



FIGUR 4.16: Brukergrensesnitt for veiviser ved oppkobling til database

enig i at det ikke skal brukes tid på å prioritere denne. All funksjonalitet fra tidligere moduler blir importert. Ved utvikling av veivisere blir det funnet flere bugs på biblioteket veiviserne bruker, og av den grunn lages det et eget veiviserbibliotek helt fra starten. På denne måten kan biblioteket tilpasses og designes som ønskelig. Funksjonaliteten for å kombinere kolonner var tidligere en drag and drop funksjonalitet, men ettersom domeneeksperten ikke likte designet, blir det nå omgjort til en veiviserbasert funksjonalitet ved hjelp av veiviserbiblioteket. Dette forhindrer brukeren i en større grad fra å gjøre feil, og fremstår som mer brukervennlig. Figur 4.16 viser et skjermbilde fra veiviseren brukeren blir møtt av ved oppkobling til ny database.

Etter denne iterasjonen ble det til slutt avholdt en systemvalidering, hvor domeneekspert validerte hvorvidt systemet skulle aksepteres eller ikke. Domeneeksperten aksepterte systemet, og godkjente at alle funksjonelle krav, bortsett fra *drill down* funksjonaliteten, var implementert. Ettersom domeneeksperten ikke har noe tidligere kunnskap om data mining, ble utvikling og planlegging av dette overlatt til utvikler.

4.8.6 Sjette iterasjon

Hele denne iterasjonen ble brukt med formål om å utvikle funksjonalitet for datainnsyn og mønstergjennkjenning. Biblioteket *Sequential Pattern Mining Framework(SPMF)* introdusert i avsnitt 2.2.2.2 ble benyttet for å kunne tilby funksjonalitet fra relevante data mining metoder. Ettersom biblioteket som nevnt i avsnitt 4.5.3 benytter seg av tekstfiler for håndtering og lesing av data, ble kildekoden betydelig tilpasset og videreutviklet. Den implementerte funksjonaliteten gir mulighet for å finne hyppige produktsett ved hjelp av *FP-Growth* algoritmen, diskutert i avsnitt 2.2.1.5, samt for å finne sjeldne produktsett ved hjelp av *Apriori-Inverse* algoritmen, diskutert i avsnitt 2.2.1.3. Basert på de hyppige eller sjeldne produktsettene kan det som diskutert i avsnitt 2.2.1.1 detekteres assosieringsregler. For å teste denne funksjonaliteten ble den testet på 500 000 rader med transaksjonsdata fra en stor byggevarehandel i Norge, som resulterte i 37 571 unike transaksjoner. På tabell 4.1 vises noen av funnene gjort i denne dataen. Assosieringsreglen *terrassebord* $\longrightarrow$ *terrasseskruer* er en logisk regel, men som Edelstein (1999) spesifiserer, er det viktig å verifisere funnene gjort ved hjelp av metoder i data mining. Denne logiske assosiasjonen kan bekrefte at metodene presenterer realistiske funn. Det blir også funnet interessante funn, for eksempel at Coca Cola kjøpes betydelig oftere enn Bonaqua, og at det kun er 46 % sannsynlighet at kunder som kjøper propantank, kjøper lovpålagt CO^2 avgift.

Funn	Beskrivelse
Coca Cola kjøpes betydelig oftere enn Bonaqua	Coca Cola forekommer i 3.15 % av transaksjonene, i motsetning til Bonaqua som forekommer 0.01 %.
Coca Cola er det 14. mest kjøpte produktet	I hele datasettet er Coca Cola som nevnt ovenfor i 3.15 % av transaksjonene.
Terrassebord og terrasseskruer kjøpes ofte sammen	Terrassebord og terrasseskruer kjøpes sammen i 4.61 % av transaksjonene.
CO^2 avgift selges ikke alltid sammen med produkter hvor det er pålagt	Ved kjøp av propantank, er det i en av assosieringsreglene kun 46 % sannsynlighet at det selges pålagt CO^2 avgift.
Den mest solgte varen er registrert som "diverse"	Den mest solgte varen i transaksjonene er registrert som "diverse", og den forekommer i 13.84 % av transaksjonene.

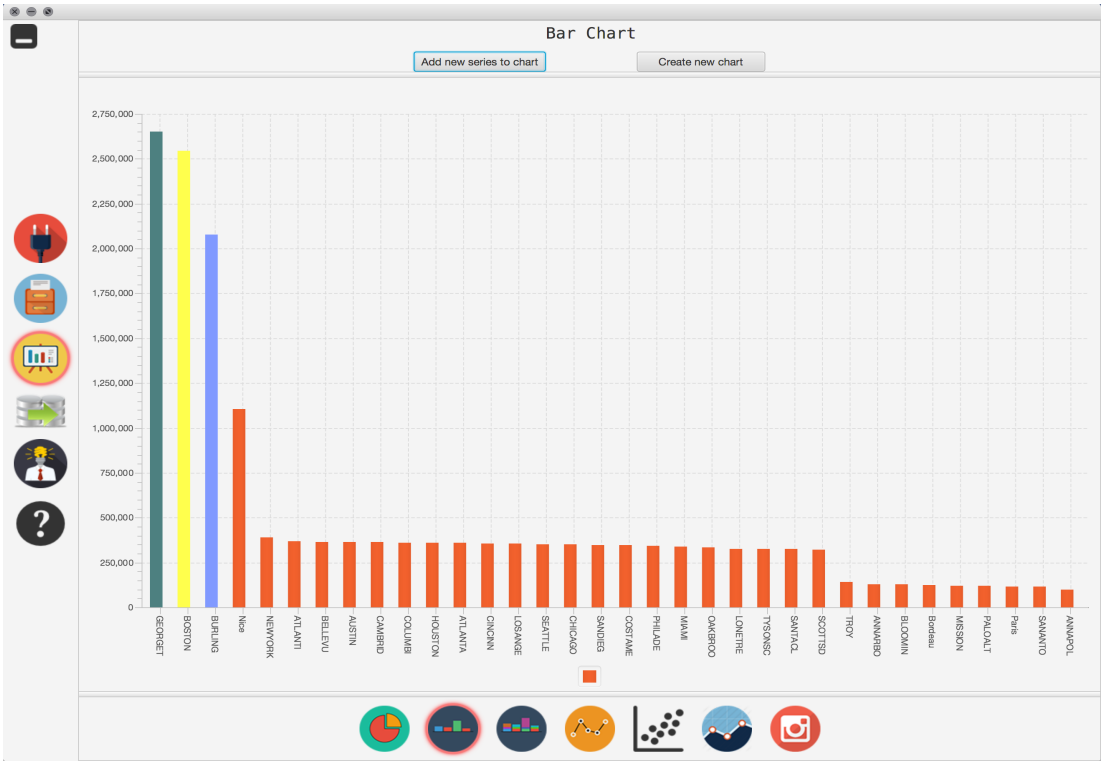
Tabell 4.1: Funn ved test av data mining funksjonalitet på en byggevarehandel i Norge

På siste og avsluttende møte fremvises den nye datainnsyns- og mønstergjennkjenningsfunksjonaliteten domeneekspert. Funksjonaliteten blir godt tatt

i mot, han liker nytten den kan tilby, og mener at dette er en nyttig funksjonalitet for en butikksjef som ønsker info om populære og ikke-populære produkter. Han mener informasjonen for eksempel kan brukes til vareplassering eller kundespesifikk markedsføring. Videre mener domeneeksperten også at dette er funksjonalitet de fleste andre konkurrenter ikke har implementert i dagens løsning, og at det av den grunn kan gi konkurransefortrinn for brukere av applikasjonen.

4.8.7 Syvende iterasjon

I etterkant av evalueringene av applikasjonen ble det i tillegg holdt en iterasjon, der ønsker og tilbakemeldinger gitt under evalueringen ble implementert i applikasjonen. Eksempelvis fremkom det et ønske om en prosentvis visning i kakediagrammet, og da denne viste seg å være tidskrevende ble den utviklet etter evalueringene var ferdige. Denne iterasjonen ble også brukt til å rydde opp i, samt rense kode. På figur 4.17 vises siste versjon av brukergrensesnittet etter syvende og siste iterasjon.



FIGUR 4.17: Brukergrensesnitt etter siste iterasjon

Kapittel 5

Evaluering

Dette kapittelet vil ta for seg evalueringen av applikasjonen, retningslinje nummer tre for rammeverket *Design Science* diskutert i avsnitt 3.1. Først vil det begrunnes hvorfor evaluering trengs, og deretter vil forskningens evaluering presenteres og oppsummeres.

5.1 Hvorfor evaluering

Evaluering er, som nevnt i rammeverket *Design Science* diskutert i avsnitt 3.1, en kritisk komponent av forskningsprosessen. I utvikling av et system, er det i følge Hevner et al. (2004), et behov for evaluering av systemet for å kontrollere at brukere kan benytte seg av, og være fornøyde med artefaktet forskningsprosessen resulterer i. En bruker ønsker at interaktive systemer skal være enkle å lære, trygge, tilfredsstillende og effektive i både tids- og ressursbruk. For å kunne oppnå dette kreves det i følge Rogers et al. (2011) at artefaktet evalueres. Effektive evalueringer involverer å forstå hvorfor evaluering er viktig, hva som bør evalueres, og hvor og når evaluering skal gjennomføres.

Som nevnt i avsnitt 3.1 bruker *Design Science*, i følge Hevner et al. (2004), typisk metoder tilgjengelig i kunnskapsbasen for å evaluere et artefakt. De utvalgte evalueringsmetodene må passe til det designede artefaktet, og evalueringsberegningene. Brukervennlighet er, som Hevner et al. (2004) nevner, én av attributtene det kan evalueres på i forskningsdesign, og ettersom design er en iterativ og inkrementell prosess, kreves det evaluering underveis i utviklingen av en applikasjon.

5.2 Evalueringsdesign

Under utvikling av artefaktet er det holdt to evalueringer. Én for å kontrollere og forbedre designet av applikasjonen underveis, samt én for å undersøke hvorvidt deltagernes holdning til applikasjon forandres når en datainnsynsfunksjonalitet implementeres. Det er benyttet tre testdeltagere fra Advania, hvorav alle har både god teknisk bakgrunn og veldig god kunnskap på fagfeltet handel og *Enterprise Resousrce Planning* og *Point of Sale* systemer. Testdeltagerne representerer med andre ord brukere fra det første scenarioet diskutert i avsnitt 4.1, hvor brukerne av applikasjonen er ansatte i et konsulenthus, tiltenkt å bruke applikasjonen for å gi sine kunder innsyn i deres data.

Som diskutert i avsnitt 3.3.4 er det viktig å bruke flere datainnhentingsteknikker for å få ulike perspektiv. Ved å bruke flere innhentingsteknikker oppnår man en triangulering, som i følge Rogers et al. (2011), sørger for at funnene blir mer kritiske og forsvarlige. I prosjektets to evalueringer er det benyttet to eller flere datainnhentingsteknikker. I første evaluering er det teknikkene intervju, observasjon og metoden *System Usability Scale(SUS)* benyttet. I andre evaluering ble teknikkene intervju og SUS benyttet. Ved å kombinere de ulike metodene sørger man for at prosjektet innhenter både kvalitative og kvantitative data, og da oppnås en triangulering.

Alle evalueringene ble holdt over nettbaserte kommunikasjonstjenester med muntlig kommunikasjon. Deltagerne ble tilsendt oppgavene de skulle løse skriftlig, og observasjonen ble foretatt ved hjelp av skjermdeling. Før første evaluering ble det utført to pilottester for både å forbedre brukertesten, og detektere eventuelle småfeil. Oppgavelisten representert i avsnitt 5.4.2 er resultatet av forbedringene gjort etter pilottestene.

5.3 Datasett brukt ved evaluering

Genov, Keavney og Zazelenchuk (2009) poengterer i deres artikkel at brukervennlighetstesting med fabrikkerte data kan produsere resultater som ikke er pålitelige og gyldige. Data som er fabrikkert kan fremstå ukjent, og til og med kanskje urealistisk for testdeltageren, og dette kan påvirke deltagerens evne til å gjenkjenne, tolke og samhandle med den fabrikkerte dataen. På grunn av dette kan testarrangører ende opp med falske testresultater, da brukerne kan bli forvirret av den oppdiktede dataen.

Som Genov et al. (2009) diskuterer i deres artikkel, snakkes det ofte om viktigheten av å la ekte brukere utføre genuine oppgaver for å kunne effektivt evaluere brukervennlighet. De poengterer av den grunn at dataen bør være genuin, og ikke fiktiv. I dette prosjektets evaluering er det i første evaluering brukt fiktiv data, men det påpekes at denne dataen er godt kjent for alle testdeltagerne. I tillegg ble det kun utvalgt fiktiv transaksjonsdata. Denne demodataen er inkludert i demoversjoner av *Microsoft Dynamics AX(AX)*, og alle deltagerne har ved flere anledninger brukt disse demoversjonene, og er av den grunn kjent med dataen. I den andre evalueringen ble det benyttet genuin transaksjonsbasert data fra en byggevarehandel i Norge for å bevise nytten av datainnsynsfunksjonaliteten. Det var viktig å bevise at resultatene fra funksjonaliteten faktisk var givende, og av den grunn måtte resultatene være genuine. Funnene som ble funnet i denne dataen er diskutert under avsnitt 4.8.6.

5.4 Første brukertest

Første brukertest ble holdt for å kartlegge hvorvidt systemet var brukervennlig, og for å danne et bilde av hva slags inntrykk deltagerne hadde av applikasjonen.

5.4.1 Datainnhenting

Den første brukertesten tok i bruk to datainnhentingsteknikker fra teknikkene diskutert i avsnitt 3.3 for innhenting av data, både direkte observasjon og ustrukturert intervju. I tillegg ble metoden *System Usability Scale(avsnitt 3.2.1)* brukt for å gi brukerne en mulighet til å rangere artefaktet med kvantitative målbare verdier. Hver bruker fikk et sett med oppgaver de skulle løse som gjengitt i avsnitt 5.4.2, og direkte observasjon ble benyttet for å observere hvordan deltagerne løste dem. Etter oppgavene var løst ble brukertesten avsluttet med et ustrukturert intervju, hvor deltageren fikk komme med tilbakemelding vedrørende applikasjonen. Ettersom intervjuet var ustrukturert, var dette deltagernes sjanse for å komme med sine tilbakemeldinger. Intervjuet ble innledet med å spørre om tilbakemeldinger, og deltagerne uttrykte sine tanker og meninger.

5.4.2 Oppgaveliste

Tekst representert i kursiv tekst med store bokstaver, illustrer kolonner i en standard *Microsoft Dynamics AX* databasetabell. Datastrukturen og tabellen var godt kjent for brukerne og trengte ikke ytterligere forklaring, men kan for utenforstående fremstå forvirrende.

1. Filtrer frem alle transaksjoner som har *TRANSDATE* '2012-12'
2. Fjern alle filter og filtrer frem transaksjoner som tilhører *STORE* 'Bordeau', har *TRANSDATE* gjennomført i '2012-12' og har en *CUSTACCOUNT* som begynner på '200'
3. Fjern alle filter og finn transaksjonen med *SALESORDERID* '000044'
4. Fjern alle filter og sorter tabellvisningen på *INVOICEID*
5. Fjern alle filter og lag et kakediagram hvor *STORE* representerer kategori og *PAYMENTAMOUNT* representerer verdi
6. Finn ut hvilken *STORE*(butikk) som hadde størst total *PAYMENTAMOUNT*(total salgsinntekt) i fjor
7. Finn ut hvilken *STORE*(butikk) som hadde totalt høyest antall *NUMBER OF ITEMS*(antall produkter solgt)
8. Lag en ny tabell som inneholder to kombinerte kolonner. Den første kombinerte kolonnen skal inneholde kolonnen *STORE* fra tabellen *transactions* og *STORE* fra tabellen *transbig*. Den andre kombinerte kolonnen skal inneholde *PAYMENTAMOUNT* fra *transactions* og *PAYMENTAMOUNT* fra *transbig*
9. Lag et scatterchart basert på den nye kombinerte tabellen du akkurat lagde, hvor kategori er *STORE* og verdi *PAYMENTAMOUNT*. Hvilken butikk hadde størst totalt *PAYMENTAMOUNT*, og hvor stor var denne?
10. Lagre et av diagrammene dine som en PNG fil.

5.4.3 Resultat

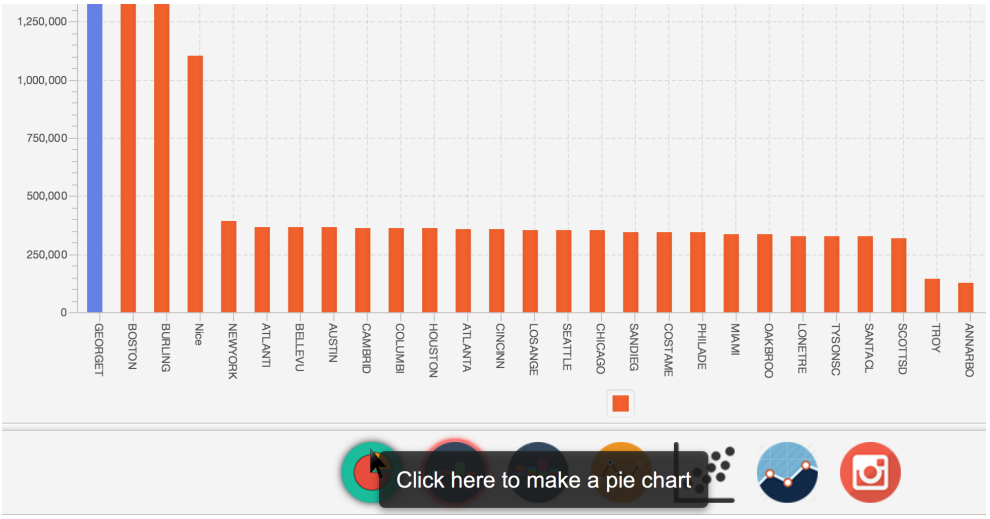
Brukerne rangerte i gjennomsnitt applikasjonen med en *System Usability Score* poengsum på 89.17, som vist på tabell 5.1. Det er betydelig høyere enn kravet satt under de ikke-funksjonelle kravene diskutert under avsnitt 4.3.2. Rangeringen stemmer godt overens med inntrykket utvikler fikk av deltagerne, når de løste oppgavene. Oppgavene gikk som regel feilfritt, og det skjedde aldri at en deltager ikke klarte å løse en av dem. Det ble lagt merke til at funksjonaliteten for hjelpetekst ved *mouse hoovering*, som vist på figur 5.1, hjalp deltagerne med å finne frem til ting de var usikre på. Deltagerne prioriterte å bruke denne funksjonaliteten fremfor knappen for hjelp. Det ble også lagt merke til at oppgaven for kombinering av kolonner var oppgaven deltagerne brukte lengst tid på å løse. Under det ustrukturerte intervjuet kom samtlige deltagere med positive tilbakemeldinger vedrørende design, og det virket som alle satt igjen med et positivt inntrykk. Det ble ved hjelp av det ustrukturerte intervjuet gitt tilbakemeldinger på potensielle forbedringer:

- Noen knapper er litt spredd, kanskje de kan samles mer? For eksempel er knappene "new series", "new chart", "remove filters" og "jump to column" litt spredd, disse bør samles.
- En markering av hvilken visning, og hvilken visualisering som er valgt, bør implementeres.
- En mulighet for prosentvisning på kakene i kakediagrammet.

Alle forbedringene ble umiddelbart implementert, bortsett fra prosentvisning på kakediagram som ble implementert etter andre brukertest(diskutert i avsnitt 4.8.7).

5.5 Andre brukertest

Andre brukertest ble holdt for å evaluere hvorvidt holdningen testdeltagerne hadde til applikasjonen forandres når en datainnsynsfunksjonalitet implementeres, samt for å konkludere hvorvidt applikasjonen er brukervennlig.



FIGUR 5.1: Hjelpetekst ved *mouse hoovering*

5.5.1 Datainnhenting

Andre brukertest tok i bruk én av datainnhentingsteknikkene fra teknikkene diskutert i avsnitt 3.3, strukturert intervju. Testdeltagerne var de samme som ble benyttet under første brukertest. I motsetning til den oppgavebaserte tilnærmingen benyttet i første brukertest, ble det her benyttet en tilnærming hvor deltagerne ble presentert funksjonaliteten. Ettersom testdeltagerne hadde liten kunnskap til data mining, ble denne tilnærmingen valgt for å kunne vise deltagerne hva som er mulig å gjøre ved hjelp av funksjonaliteten. I etterkant av demonstrasjonen ble det strukturerte intervjuet vist i avsnitt 5.5.2 holdt, og til slutt rangerte testdeltagerne applikasjonen igjen ved hjelp av *System Usability Scale* skjemaet.

5.5.2 Strukturert intervju

Deltagerne ble før intervjuet informert om at formålet med intervjuet, var å danne et bilde av om holdningen til applikasjonen hadde forandret seg etter implementering av datainnsynsfunksjonaliteten.

1. Har du forstått formålet med funksjonaliteten? Hvis ikke, forklarer jeg deg gjerne mer.

2. Hvordan liker du denne nye funksjonaliteten?

3. Hvis denne funksjonaliteten implementeres i applikasjonen, hvordan vil den påvirke din holdning til programmet?

4. Synes du informasjonen funksjonaliteten tilbyr deg er meningsfull og gir den deg noe nytte?

5. Hvordan tror du informasjonen vil kunne påvirke plassering av varer i en butikk?

6. Gjør "summary view" det mye lettere å forstå informasjonen i dataene?

7. Hvilke inntrykk sitter du igjen med om programmet etter at jeg har vist deg denne nye funksjonaliteten?

8. Har du sett en lignende funksjonalitet før?

5.5.3 Resultat

Etter fremvisning av datainnsynsfunksjonaliteten, rangerte brukerne applikasjonen med en betydelig bedre *System Usability Score(SUS)* poengsum på 93.33 i gjennomsnitt, som vist på tabell 5.1. Alle deltagerne ga en høyere SUS score og én rangerte applikasjonen med en toppscore på 100. Under det strukturerte intervjuet, kom det frem fra samtlige deltagere at funksjonaliteten vil styrke deres forhold til applikasjonen. Én deltager svarte "i utgangspunktet vil jeg si at jeg hadde en positiv holdning i utgangspunktet, og holdningen vil forbli positiv" på spørsmål nummer tre. En annen deltager svarte "positivt, applikasjonen tilbyr mer og blir sterkere". Alle deltagerne mente at informasjonen ga nytte, og at den også kunne ha nytte for markedssjefer og butikksjefer. Deltagerne mente videre at informasjonen kunne påvirke plasseringer av varer i en butikk. Informasjonen ville i følge deltagerne hjelpe til med å kartlegge hva folk kjøper sammen, og dette kan brukes ved plassering av varer. For eksempel svarte en av deltagerne at "dataen gir bedre grunnlag for å plassere tingene sammen eller ikke sammen i butikk. Det påvirker det positivt, og butikksjefer vil få mer innsyn i data" på spørsmål nummer fem. Det var også enighet om at "summmary view" ga en enklere visning som var lettere å forstå. En deltager begrunnet dette godt, da deltageren svarte følgende på spørsmål nummer seks: "ja, absolutt. Spesielt når du ikke kjenner datagrunnlaget veldig godt. For eksempel at du ikke husker varenummer".

Deltager	Score
1	87.5
2	97.5
3	82.5
Gj. snitt	89.17

Deltager	Score
1	90
2	100
3	90
Gj. snitt	93.33

Tabell 5.1: SUS rangering fra første og andre evaluering sammenlignet

5.6 Oppsummering

Én av deltagerne svarte én (*strongly disagree*) på den første påstanden *I think that I would like to use this system frequently* i *System Usability(SUS)* skjemaet på den første evalueringen. Deltageren ble forhørt hvorfor, og begrunnelsen var at personen ikke kunne se hvordan systemet kunne brukes i arbeidsmiljøet sitt. Dette kan være forståelig da denne deltageren er utvikler, og en utvikler ofte ikke har direkte kontakt med kundene. Det vil heller være konsulenten som tilbyr som en tjeneste å skaffe informasjon i kundens datasett ved hjelp applikasjonen. Det påpekes at ved andre SUS rangering økte interessen for programmet og deltageren økte rangeringen på påstanden fra én til to.

Én annen deltager uttrykte at funnene presentert i datainnsynsfunksjonaliteten kanskje var for avansert, og at for eksempel butikksjefer vil ha enda enklere informasjon og visninger. Deltageren mente at "butikksjefer er ganske enkle folk, og de vil kanskje ha presentert funnene enda enklere". I tillegg økte deltagerens enighet av påstanden *I thought there was too much inconsistency in this system* fra én(*strongly disagree*) til to etter presentasjon av datainnsynsfunksjonaliteten. Oppsummeringsvisningen nevnt i 4.8 var tiltenkt å gi en forenklet visning til for eksempel butikksjefer eller markedssjefer, men det kan hende denne skulle vært laget enda enklere.

De summerte SUS rangeringene illustreres på tabell 5.1. En betydelig høyere SUS rangering enn den kravsatte i kravspesifikasjonen, kan gi en indikasjon på at applikasjonen er brukervennlig. Deltagerne viste stor interesse for systemet, og forbedringen av SUS rangeringen fra første til andre evaluering, viser at datainnsynsfunksjonalitet betydelig øker interessen for programmet. Under observasjon og intervjuer ble det også gitt inntrykk for at deltagerne har en positiv holdning til applikasjonen. Det ble påpekt og observert noe forbedringspotensiale, og dette ble rettet og implementert.

Kapittel 6

Konklusjon og fremtidig arbeid

Dette kapittelet konkluderer denne oppgaven, som har tatt for seg følgende forskningsspørsmål:

> Hvordan utvikle et data mining- og datavisualiseringsverktøy for transaksjonsdata fra Enterprise Resource Planning og Point of Sale systemer

For å besvare forskningsspørsmålet har dette prosjektet tatt i bruk rammeverket *Design Science*, og på tabell 6.1 oppsummeres det hvilke kapitler som har tatt for seg de ulike retningslinjene fra rammeverket. Prosjektet har resultert i applikasjonen EinBlick, et verktøy for å visualisere og skaffe innsyn i transaksjonsbaserte data. Applikasjonen er resultatet av en inkrementell utviklingsprosess gjennomført ved hjelp av prosesshåndteringsverktøyet Kanban. Under utviklingsprosessen har tre testdeltagere fra det første brukerscenarioet, identifisert i avsnitt 4.1, evaluert applikasjonen. Selv om EinBlick primært har benyttet data fra fra ett firma, bør metodene implementert i applikasjonen også være anvendelig på transaksjonsdata fra to eller flere firma.

Utviklingsprosessen har totalt inneholdt syv iterasjoner, hvor hver av iterasjonene hadde avsluttende tilbakemeldingssesjoner med domeneekspert. Kravspesifikasjonen ble forbedret og oppdatert kontinuerlig etter hver tilbakemeldingssesjon, og dette har sørget for at applikasjonen dekker de krav og behov som har oppstått. Utviklingsprosessen har vært både effektiv og lærerik på tross av tidsbegrensninger. JavaFX 8 opplevdes som noe utfordrende, og på grunn av dette og den begrensede tiden, ble visualiseringdelen i applikasjonen kanskje ikke like avansert som planlagt. EinBlick har likevel støtte for

flere velkjente visualiseringer som for eksempel kakediagram, stolpediagram og stablet stolpediagram.

Retningslinje	Kapittel
1. Design som et artefakt	Oppgaven omhandler EinBlick, en applikasjon som benytter metoder fra datavisualisering og data mining. I kapittel 4 beskrives designprosessen av EinBlick, og i kapittel 5 beskrives evalueringen av applikasjonen.
2. Problemrelevans	I kapittel 1 presenteres forskningsspørsmålet, og hvordan metoder fra datavisualisering og data mining kan gi bedrifter innsyn i deres data.
3. Design evaluering	Evalueringsprosessen av applikasjonen presenteres i kapittel 5, hvor det evalueres på brukervennligheten av EinBlick. Ved hjelp av *System Usability Scale*, intervju og observasjon, evalueres applikasjonen i to omganger.
4. Bidra til forskningsområdet	Kapittel 4 og kapittel 5 presenterer EinBlick, det designede artefakt som bidrar til forskningsområdet.
5. Forskningsnøyaktighet	I kapittel 3, kapittel 4 og kapittel 5 presenteres relevante metoder benyttet i både evaluering og utvikling av artefaktet.
6. Design som en søkeprosess	Designet av applikasjonen har vært en inkrementell og iterativ prosess for å finne en effektiv løsning på et problem. I kapittel 4 og kapittel 5 presenteres prosessen benyttet for å komme frem til en effektiv løsning.
7. Kommunikasjon av forskningen	Forskningen blir kommunisert ved hjelp av oppgaven som en helhet.

Tabell 6.1: Hvilke kapitler som har tatt for seg de ulike retningslinjene i Design Science

Det ble holdt to evalueringer, hvor den første evalueringen ga et inntrykk om hva deltagerne syntes om applikasjonen, og en indikasjon på om den var brukervennlig. I den andre evalueringen ble det testet hvordan funksjonalitet fra data mining påvirket deltagernes holdning til applikasjonen. Ut i fra evalueringene av artefaktet fremkom det en indikasjon på at artefaktet er brukervennlig. Den kvantitative rangeringen ved hjelp av *System Usability Scale* økte betydelig når data mining funksjonaliteten ble implementert, og deltagerne uttrykte videre gjennom intervjuet at denne funksjonaliteten er nyttig.

6.1 Begrensninger og fremtidig arbeid

Evalueringsprosessen skulle gjerne pågått over lenger tid, men ettersom denne oppgaven hadde en relativ kort tidsramme, ble det utført en begrenset evaluering. Det er kun avholdt evalueringer med testdeltagere fra ett av de to brukerscenarioene identifisert, og ideelt sett skulle gjerne testdeltagere fra butikkjeder også vært inkludert. Det ble som nevnt ovenfor valgt tre testdeltagere, hvorav to av dem er kundekonsulenter, og den siste en utvikler. I etterpåklokskapens lys burde kanskje alle deltagerne vært konsulenter, ettersom utviklere sjelden har direkte kontakt med kunder. Av den grunn regnes ikke denne deltageren som en av sluttbrukerne. Dette forklarer også hvorfor utvikleren, som diskutert i avsnitt 5.6, ikke viste så stor interesse for å bruke applikasjonen i sitt arbeidsmiljø. Testdeltageren har allikevel bidratt til applikasjonen med tilbakemeldinger og ideer ved hjelp av sin erfaring. Det kan i fremtiden være hensiktsmessig å evaluere det andre brukerscenarioet, presentert i avsnitt 4.1, hvor butikkjeder selv bruker applikasjonen for å skaffe innsiktsfull informasjon. Det vil videre være ideelt å inkludere flere testdeltagere for å gjøre testresultatene mer solide. Dette vil også gjøre *System Usability Score* poengsummen mer generalisert.

Einblick benytter seg, som nevnt ovenfor, av visualiseringene tilgjengelige i biblioteket JavaFX, og foreløpig er dette visualiseringer som for eksempel kake-, linje- og stolpediagram. Som Stodder (2013) nevner, er dette velkjente visualiseringstyper, som brukere enkelt forstår, men det kan i fremtiden være av stor nytte å legge til flere og mer avanserte visualiseringer. I artikkelen til Keim et al. (2007) og Durand et al. (2006), presenteres nye måter å visualisere større mengder transaksjonsdata på, og disse visualiseringene er gode eksempler på mer avanserte visualiseringer av transaksjonsdata som kan implementeres i applikasjonen.

Foreløpig har applikasjonen heller ingen mulighet for å forstå forskjellen på tidspunkter og intervaller. I artikkelen til Aigner, Miksch, Müller, Schumann og Tominski (2007) og artikkelen til Stodder (2013), nevnes dette som et viktig poeng for å kunne visualisere i bestemte tidsperioder. I applikasjonen er det lagt til flere ulike typer filtreringsfunksjonaliteter, og det er mulighet for å både søke på, eller velge en tid før/etter en spesifikk tid. I fremtiden kan det også legges til funksjonalitet som forstår forskjellen på intervaller og tidspunkter, og dette vil gi mulighet for å søke i spesifikke tidsrom.

Stodder (2013) nevner også at bedrifter ofte har ønske om å kunne visualisere geografiske data. Som et eksempel nevnes det hvordan handelsbedrifter kan bruke handelsdata og kart for å avgjøre hvor de skal plassere butikker. Det vil av den grunn være nyttig å implementere geografiske visualiseringsmuligheter i applikasjonen ved en senere anledning.

Referanser

Advania. (2015). *Advania konsern.* Hentet 2015-04-23 fra http://www.advania.no/om-advania/advania-konsern/

Agrawal, R., Imieliński, T. & Swami, A. (1993). Mining association rules between sets of items in large databases. *SIGMOD Rec.*, *22*(2), 207-216. Hentet fra http://doi.acm.org/10.1145/170036.170072

Aigner, W., Miksch, S., Müller, W., Schumann, H. & Tominski, C. (2007). Visualizing time-oriented data-a systematic view. *Computers and Graphics (Pergamon)*, *31*(3), 401-409. Hentet 2014-01-22 fra http://linkinghub.elsevier.com/retrieve/pii/S0097849307000611

Apache. (2015). *Apache mahoot.* Hentet 2015-03-03 fra http://mahout.apache.org

Benson, J. (2009). *Personal kanban.* Hentet 2015-03-11 fra http://www.personalkanban.com/pk/personal-kanban-101/#sthash.Ot3DLZtK.QmEoQbj5.dpbs

Bodon, F. (2003). *A fast apriori implementation* (vol. 90). Hentet fra http://www.cs.bme.hu/~bodon/kozos/papers/bodon_trie.pdf

Brooke, J. (1996). Sus-a quick and dirty usability scale. *Usability evaluation in industry*, *189*(1). Hentet fra http://www.tbistafftraining.info/smartphones/documents/b5_during_the_trial_usability_scale_v1_09aug11.pdf

Coron. (2015). *Coron.* Hentet 2015-03-03 fra http://coron.loria.fr/site/index.php

Durand, N., Crémilleux, B. & Suzuki, E. (2006). Visualizing transactional data with multiple clusterings for knowledge discovery. *Foundations of Intelligent Systems*. Hentet fra http://link.springer.com/chapter/10.1007/11875604_7

Edelstein, H.A. (1999). *Introduction to data mining and knowledge disocvery* (3rd utg.). Two Crows.

Fayyad, U., Grinstein, G. & Wierse, A. (2001). *Information visualization in data mining and knowledge discovery* (1. utg.). Morgan Kaufmann.

Fisher, M. & Bruce, J. (2003). *Jdbc api tutorial and reference, third edition.* Addison-Wesley Professional.

Fournier-Viger, P. (2015). *Performance.* Hentet 2015-03-02 fra `http://www.philippe-fournier-viger.com/spmf/index.php?link=performance.php`

Friendly, M. (2009). *Milestones in the history of thematic cartography, statistical graphics, and data visualization.* Hentet fra `http://www.math.yorku.ca/SCS/Gallery/milestone/milestone.pdf`

Genov, A., Keavney, M. & Zazelenchuk, T. (2009). Usability testing with real data. , *4*(2), 85-92. Hentet fra `http://uxpajournal.org/wp-content/uploads/pdf/JUS_Genov_Feb2009.pdf`

Gomariz, A., Gueniche, T. & Fournier-viger, P. (2014). Spmf : A java open-source pattern mining library. *Journal of Machine Learning Research*, *15*, 3389-3393. Hentet fra `http://jmlr.org/papers/volume15/fournierviger14a/fournierviger14a.pdf`

Goodrich, M.T., Tamassia, R. & Morrisey, B. (2014). *Data structures and algorithms in java* (6. utg.). Wiley.

Gu, J., Wang, B., Zhang, F., Wang, W. & Gao, M. (2011). An improved apriori algorithm. I *Communications in computer and information science* (vol. 224 CCIS, s. 127-133). Springer Berlin Heidelberg. Hentet fra `http://dx.doi.org/10.1007/978-3-642-23214-5_18`

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. & Witten, I.H. (2009). The weka data mining software: An update. *SIGKDD Explor. Newsl.*, *11*(1), 10-18. Hentet fra `http://doi.acm.org/10.1145/1656274.1656278`

Han, J., Kamber, M. & Pei, J. (2012). *Data mining: Concepts and techniques* (3. utg.). Morgan Kaufmann.

Han, J., Pei, J. & Yin, Y. (2000). Mining frequent patterns without candidate generation. *SIGMOD Rec.*, *29*(2), 1-12. Hentet fra `http://doi.acm.org/10.1145/335191.335372`

Hevner, A.R., March, S.T., Park, J. & Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, *28*(1), 75-105. Hentet 2014-03-24 fra `http://www.springerlink.com/index/pdf/10.1007/s11576-006-0028-8,http://dblp.uni-trier.de/rec/bibtex/journals/misq/HevnerMPR04`

Kaidi, Z. (2000). *Data visualization.* Hentet fra http://www.cs.uic.edu/~kzhao/Papers/00_course_Data_visualization.pdf

Kaur, H. & Singh, K. (2013). Market basket analysis of sports store using association rules. *International Journal of Recent trends in Electrical & Electronics Engineering*, *3*(1), 81-85. Hentet fra http://www.ijrte.net/files/5N5-IJRTE0301126-v3-iss1-81-85.pdf

Keim, D.A. (2002). Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, *8*(1), 1-8. Hentet fra http://www.vis.uni-stuttgart.de/plain/vdl/vdl_upload/280_4_2002-Keim-Visualization.pdf

Keim, D.A., Hao, M.C., Dayal, U. & Lyons, M. (2007). Value-cell bar charts for visualizing large transaction data sets. *IEEE Transactions on Visualization and Computer Graphics*, *13*(4), 822-833. Hentet fra http://kops.uni-konstanz.de/bitstream/handle/123456789/5574/tvcg_0156_0906_corrected_3.pdf?sequence=1&isAllowed=y

Knime. (2015). *Knime.* Hentet 2015-03-03 fra http://www.knime.org

Ko, S., Maciejewski, R., Jang, Y. & Ebert, D.S. (2012). Marketanalyzer: An interactive visual analytics system for analyzing competitive advantage using point of sale data. *Computer Graphics Forum*, *31*(3), 1245-1254. Hentet fra http://rmaciejewski.faculty.asu.edu/papers/2012/Ko_Market.pdf

Koh, Y. & Rountree, N. (2005). Finding sporadic rules using apriori-inverse. I *Advances in knowledge discovery and data mining se - 13* (vol. 3518, s. 97-106). Springer Berlin Heidelberg. Hentet fra http://dx.doi.org/10.1007/11430919_13

Legler, T., Lehner, W. & Ross, A. (2006). Data mining with the sap netweaver bi accelerator. *VLDB 06 Proceedings of the 32nd international conference on Very large data bases*, 1059-1068. Hentet fra http://portal.acm.org/citation.cfm?id=1164218

Leonard, M. & Wolfe, B. (2005). Mining transactional and time series data. I *Sugi 30 proceedings.* Philadelphia, Pennsylvania. Hentet fra http://www2.sas.com/proceedings/sugi30/080-30.pdf

MathWorks. (2015). *Matlab - transpose.* Hentet 2015-01-20 fra http://se.mathworks.com/help/matlab/ref/transpose.html

Middleton, P. & Joyce, D. (2012). Lean software management: Bbc worldwide case

study. *IEEE Transactions on Engineering Management*, *59*(1), 20-32. Hentet 2014-04-14 fra http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5741001

Morales-Chaparro, R., Preciado, J.C. & Sánchez-Figueroa, F. (2011). Data-driven and user-driven multidimensional data visualization. I *Lecture notes in computer science including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics* (vol. 7059 LNCS, s. 159-166). Hentet fra http://www.scopus.com/inward/record.url?eid=2-s2.0-84857766246&partnerID=40&md5=0f30b598e4342cc87fe69d87004ea7a6

Peterson, D. (2009). *What is kanban.* Hentet 2014-03-09 fra http://www.kanbanblog.com/explained/

Power, D.J. (2002). *What is the 'true story' about data mining, beer and diapers?* (vol. 3). Hentet 2015-04-20 fra http://www.dssresources.com/newsletters/66.php

RapidMiner. (2015). *Rapidminer.* Hentet 2015-03-03 fra https://rapidminer.com

Rogers, Y., Sharp, H. & Preece, J. (2011). *Interaction design* (3. utg.). Wiley.

Shackel, B. (2009). Usability - context, framework, definition, design and evaluation. *Interacting with Computers*, *21*(5-6), 339-346. Hentet fra http://dx.doi.org/10.1016/j.intcom.2009.04.007

Shim, B., Choi, K. & Suh, Y. (2012). Crm strategies for a small-sized online shopping mall based on association rules and sequential patterns. *Expert Systems with Applications*, *39*(9), 7736-7742. Hentet fra http://www.sciencedirect.com/science/article/pii/S0957417412000930

Sommerville, I. (2010). *Software engineering* (9. utg.). Addison-Wesley.

Stodder, D. (2013). Data visualization and discovery for better business decisions. *The Data Warehouse Institute Best Practices Report*(Third Quarter). Hentet fra http://www.pentaho.com/sites/default/files/uploads/resources/data_visualization_and_discovery_for_better_business_decisions.pdf

Tan, P.N., Steinbach, M. & Kumar, V. (2005). *Introduction to data mining.* Addison-Wesley.

Theus, M. (2002). Interactive data visualization using mondrian. *Journal of Statistical Software*, *7*, 9. Hentet fra http://www.jstatsoft.org/v07/i11/paper

Vliet, H.V. (2008). *Software engineering: Principles and practice* (3. utg.). Wiley.

Yong, L. (2012). The building of data mining systems based on transaction data mining language using java. *JDCTA: International Journal of Digital Content Technology*

and its Applications, *6*(14), 298-305. Hentet fra `http://www.aicit.org/JDCTA/ppl/JDCTA1538PPL.pdf`