Høgskulen på Vestlandet

BACHELOROPPGAVE

Automatisert modellering av integrasjoner

Automatic modeling of integrations

Sindre Holtan

Mathias Pham

Dat191

Fakultet for ingeniør- og naturvitenskap

Institutt for datateknologi, elektroteknologi og realfag

Informasjonsteknologi

Veileder Yngve Lamo

Innleveringsdato: 22/05/2023

TITTELSIDE FOR HOVEDOPPGAVE

Rapportens tittel: Automatisert modellering av integrasjoner Automatic modeling of integrations	*Dato:* *22.05.2023*
Forfatter(e): Mathias Pham, Sindre Holtan	*Antall sider u/vedlegg: 42*
	Antall sider vedlegg: 93
Studieretning: Informasjonsteknologi	*Antall disketter/CD-er:* *0*
Kontaktperson ved studieretning: Yngve Lamo	*Gradering: Ingen*
Merknader: *Ingen*	

Oppdragsgiver: Helse Vest IKT	*Oppdragsgivers referanse:* EB-10
Oppdragsgiver kontaktperson: Ronny Heitmann Andersen Nikita Tsaritson	*Telefon:* *918 07 199* *936 23 904*

Sammendrag: Denne rapporten handler om utviklingen av en automatisert modelleringsløsning for Helse Vest IKT, kalt MOI-Automatic. Løsningen er ment å erstatte manuelt arbeid som utføres i dag, og vil i utgangspunktet være en pilot som vil bli videreutviklet av bedriften.

Stikkord:

PowerShell, skript	Integrasjoner, relasjoner	Automatisert modellering

Høgskulen på Vestlandet, Fakultet for ingeniør- og naturvitskap
Postadresse: Postboks 7030, 5020 BERGEN Besøksadresse: Inndalsveien 28, Bergen
Tlf. 55 58 75 00 Fax 55 58 77 90 E-post: post@hvl.no Hjemmeside: http://www.hvl.no

Forord

Dette prosjektet dokumenterer arbeidet utført i forbindelse med bachelorprosjektet *"Automatisert modellering av integrasjoner"* ved Høgskulen på Vestlandet (HVL) våren 2023. Rapporten ble skrevet av Sindre Holtan og Mathias Pham.

Gruppen ønsker å takke:

- Ronny Heitmann Andersen for det gode og verdifulle samarbeidet og veiledningen gjennom rapporten, prosjektet og systemutviklingen. Vi ønsker også å uttrykke vår takknemlighet for hans engasjement i prosjektet og hans bidrag til å hjelpe gruppen med å utarbeide figurtegninger.

- Nikita Tsaritson for samarbeid, bidrag til prosjektet og viktige kunnskap om integrasjoner.

- Yngve Lamo for god veiledning og kloke råd i forbindelse med rapporten.

- Helse Vest IKT som ga oss tilgang til deres lokale og et godt arbeidsmiljø som har vært avgjørende for å kunne gjennomføre det spennende prosjektet.

Ordliste

MOI-AUTOMATIC	Navnet til programmet som ble utviklet av gruppen.
Assyst	Dette er bedriftens CMDB hvor de har oversikt over ressursene.
BizTalk	Integrasjonsplattform som brukes av bedriften.
BizTalk Binding files	Xml fil som brukes til å eksportere og importere konfigurasjonsinformasjon i BizTalk-databaser.
Relasjon	Dette er en tilknytning mellom to eller flere ting eller elementer. Den viser til forholdene til integrasjonene med alvorlighetsgrad, for eksempel.
Integrasjon	Når to eller flere systemer ikke har muligheten til å kommunisere med hverandre, kan en integrasjon sørge for å få disse systemene til å fungere som en helhet.
Skript	Dette er en samling av instruksjoner eller kommandoer skrevet i et programmerings- eller skriptspråk som kan utføres av en datamaskin.
XML	XML er basert på en hierarkisk struktur av elementer og attributter og kan brukes til å beskrive og utveksle informasjon mellom ulike datamaskiner og applikasjoner på en plattformuavhengig måte.
JSON	JSON står for JavaScript Object Notation og er et tekstbasert format for å representere strukturert data basert på JavaScript Object Syntax.
PowerShell	PowerShell er en kommandolinje og et skriptspråk som er vanlig å bruke i et Windows miljø.
Sesjonsvariabel	Sesjonsvariabler brukes i skript eller andre programmeringskontekster for å lagre brukerspesifikk informasjon under en økt, og blir slettet når økten avsluttes.
Archi	*"Archi"* er en forkortelse av "AchiMate". Programmet er en programvareløsning for modellering og visualisering av bedriftsarkitektur.

INNHOLDSFORTEGNELSE

1 INNLEDNING

Dette kapittelet gir en innføring i bakgrunnen for problemet og forklarer hvorfor det er nødvendig å finne en løsning. Videre vil prosjektets eier og mål bli introdusert.

1.1 Kontekst

Helse Vest IKT er en av de største integrasjonsmiljøene i Norge og er ansvarlig for å sikre at informasjonsflyten mellom helsesystemer og ulike interessenter i sektoren, som kommuner, fastleger og helsenorge.no, fungerer optimalt. Derfor er de avhengige av å ha god oversikt over systemene sine. I løpet av det siste tiåret har selskapet doblet antall ansatte og er fortsatt under vekst.

For tiden utføres det meste av systemmodellering manuelt, noe som er tidkrevende. Derfor ønsker selskapet en mer automatisert løsning for modellering/presentasjon av integrasjonsløsninger.

1.2 Motivasjon

Helse Vest IKT mangler for øyeblikket en automatisert løsning for modellering av systemene og eksterne tjenester. Bedriften har omtrent 300 integrasjonsprodukter med over 2000 grensesnitt som er i stadig endring. Som følge av dette blir stadig nye applikasjoner etablert, som igjen skaper nye relasjoner mellom dem og krever jevnlige oppdateringer. Som et resultat brukes det unødvendig tid på manuell modellering og feilsøking blant personell som ikke jobber med dette daglig. Derfor har selskapet et behov for en automatisert modelleringsløsning for systemene sine.

1.3 Prosjekteier

Helse Vest IKT er et aksjeselskap som ble etablert i 2004, og eies av Helse Vest RHF (Regionalt helseforetak). Bedriften har ansvar for å levere IKT-tjenester til spesialisthelsetjenesten i Helse Vest, og har i dag rundt 700 medarbeidere. Hovedkontoret ligger i Bergen, men det finnes også kontorer i Stavanger, Haugesund og Førde (Helse Vest IKT, 2022).

1.4 Problembeskrivelse og mål

Helse Vest IKT har stadig endringer i systemer, integrasjoner og relasjoner på grunn av behovene som oppstår rundt tjenester, drift og vedlikehold. Derfor er det viktig for bedriften å få en automatisert integrasjonsløsning for modellering av relasjoner, slik at de kan spare tid i fremtiden når endringer oppstår.

Målet med prosjektet er å utvikle en pilot for sammenstilling av bedriftens kildedata ved hjelp av Assyst (hovedkilden for dokumentasjon av systemer og integrasjoner) og ved hjelp av BizTalk (integrasjonsplattform). Ved hjelp av disse informasjonskildene, skal piloten kunne finne relasjoner der imellom og gruppere informasjonen på en strukturert måte. Denne piloten bør kunne brukes til å modellere informasjon slik at den kan presenteres i et arkitektur- eller modelleringsprogram. Målet er å utvikle en pilot som kan utvides videre av bedriften.

Oppgaven gir stor frihet til hvordan den skal løses, men med de målene som er satt, er problemstillingen formulert som følger:

- **“Hvordan kan man utvikle en pilot som automatisert viser sammenhengen mellom relasjonene i applikasjonene?"**

- **“Hvordan kan man finne relasjon mellom applikasjonene i integrasjonssystemet som definerer og grupperer den?"**

1.5 Oppbygging av rapporten

Rapporten inneholder 7 hovedkapitler som beskriver prosessen ved å utvikle produktet, *MOI-Automatic*.

Kapittel 1 INNLEDNING gir et større bilde for prosjektet, hva problemstillingen er, hvem bedriften er, og hva målet for prosjektet inneholder.

Kapittel 2 PROSJEKTBESKRIVELSE beskriver om praktisk bakgrunn, avgrensninger, og ressursene som ble brukt.

Kapittel 3 DESIGN AV PROSJEKTET omhandler forslag til løsning, diskusjon av alternativene, den valgte løsningen, samt prosjektmetodikk.

Kapittel 4 DETALJERT LØSNING inneholder arkitekturskisse for systemet samt programmet.

Kapittel 5 RESULTATER viser til evalueringresultat samt prosjektgjennomføring.

Kapittel 6 DISKUSJON drøfter rundt valgte løsninger med konsekvenser og hvordan det påvirket resultatene.

Kapittel 7 KONKLUSJON OG VIDERE ARBEID konkluderer om målet til prosjektet samt forslag til videre arbeid.

Kapittel 8 Referanseliste inneholder en oversikt over kilder og referanser som har blitt brukt i rapporten.

Kapittel 9 Vedlegg viser til støttedokumenter for prosjektet.

2 PROSJEKTBESKRIVELSE

Kapittelet inneholder en forklaring av prosjektets bakgrunn og krav. Videre forteller den om avgrensninger satt til prosjektet og om hvilke tilgjengelige ressurser som ble tilrettelagt.

2.1 Praktisk bakgrunn

2.1.1 Initielle krav

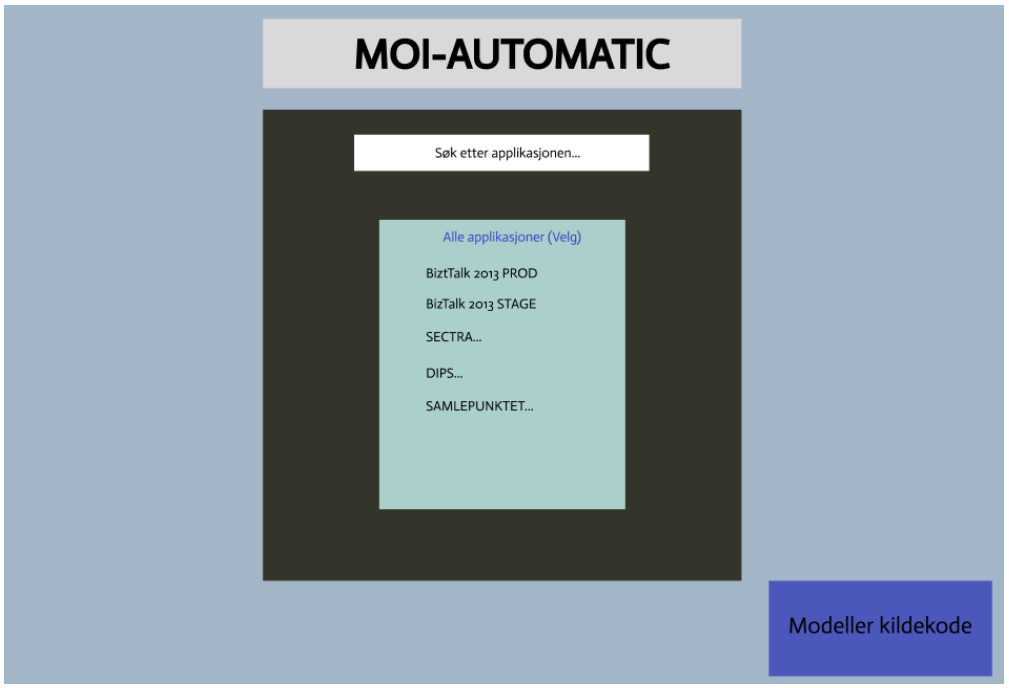
Hovedkravet til prosjektet er å lage en pilot hvor man henter inn data fra ulike informasjonskilder, og automatisk modellerer dataen på en strukturert måte. Det er ønsket at modellene gir oversikt over relasjonene og avhengighetene mellom applikasjonene. I tillegg ønskes det at løsningen gir brukeren mulighet til å finne eller søke etter ønsket applikasjon og dens relasjoner til andre systemer.

Oppdragsgiver har tidligere nevnt at gruppen skal bruke Assyst sammen med Biztalk Binding files for å hente inn nødvendig informasjon omkring relasjonene i systemet for å lage en pilot. Ved å analysere innhentede data skal piloten inneholde den nødvendige informasjonen som trengs for å modellere relasjonene i systemene. Assyst inneholder store mengder informasjon og dokumentasjon om applikasjoner og integrasjoner lagt inn av systemansvarlige. Ved bruk av denne informasjonen vil gruppen kunne bruke dette som "hovedkilde" for informasjon til modellen. Det er derfor et krav for gruppen å hente inn ekstra informasjon fra andre systemer som f.eks BizTalk, for å kunne tilføre ekstra informasjon i modellen. Ved å bruke Biztalk Binding-files, vil gruppen kunne koble ekstra informasjon som ellers ikke normalt ville vært tilgjengelig i Assyst til programmet.

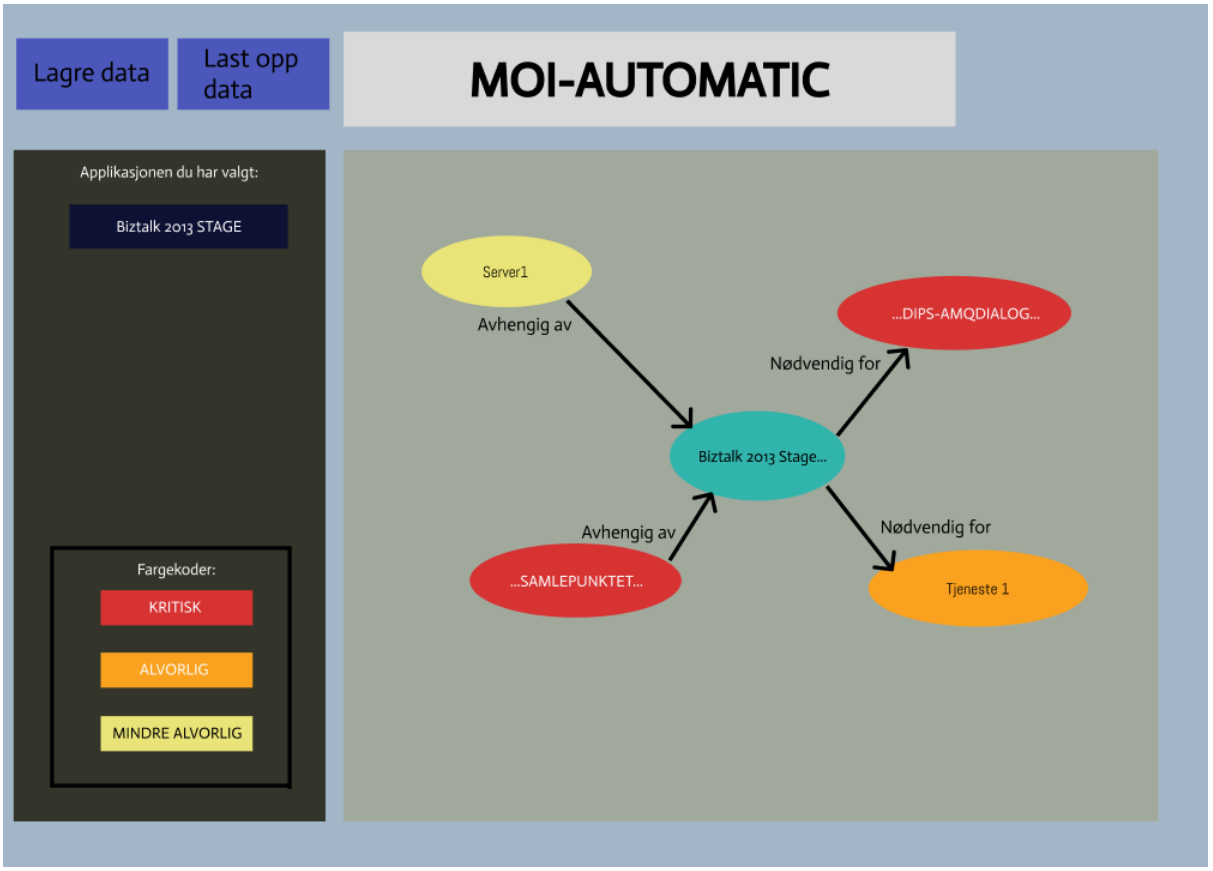
Ettersom dette er en pilot som er ønsket for videreutvikling, har oppdragsgiver sagt at det er først og fremst viktig for studentene å finne relasjoner mellom de innhentede dataene for applikasjonene, og lage en metode hvor denne informasjonen kan hentes inn. Det er ønsket at relasjonene skal kunne eksporteres til et program hvor informasjon omkring applikasjonene og relasjoner vises visuelt som for eksempel Archi (Beauvoir, Sarrodie, 2023) eller tilsvarende applikasjoner.

2.1.2 Initiell løsnings-idé

I figur 2.1 vil tjenesten inneholde en side hvor brukeren kan velge eller søke etter en applikasjon som de ønsker å få mer informasjon om. Etter å ha valgt en applikasjon skal brukeren få en oversikt over relasjonene og avhengighetene til den ønskede applikasjonen ved å trykke på “Modeller kildekode”.



Figur 2.1: Wireframe - velg applikasjon (se vedlegg 3: kravdokumentasjon)

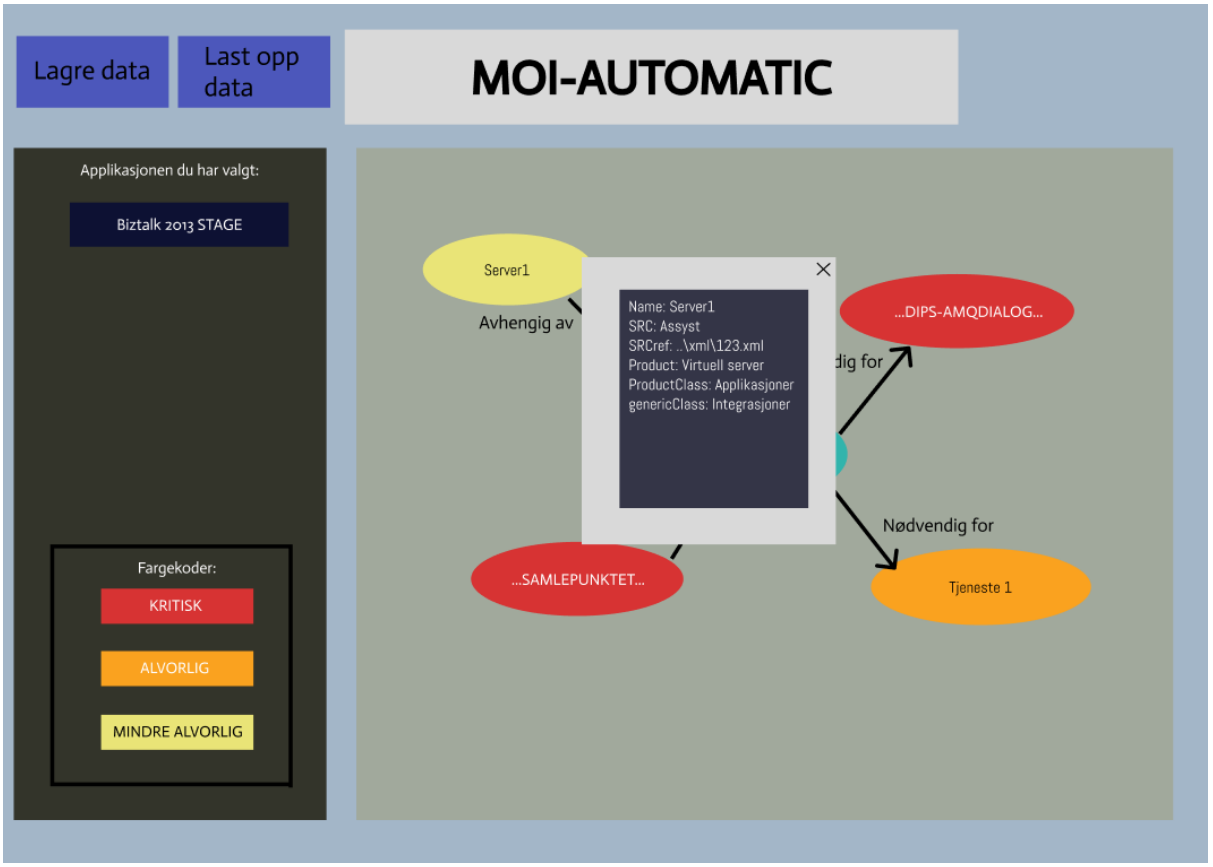


Figur 2.2: Wireframe - kildemodelleringen (se vedlegg 3: kravdokumentasjon)

Det er seks relasjoner som definerer alvorlighetsgraden i applikasjonene. De er delt inn i 2 hovedkategorier: “avhengig av” og “nødvendig for”, som hver er delt inn i tre alvorlighetsgrader: “kritisk”, “alvorlig”, og “mindre alvorlig”. Disse relasjonene beskriver hvor viktig og kritiske de forskjellige applikasjonene er til hverandre.

I figur 2.2 presenteres relevante applikasjoner som den valgte applikasjonen har relasjon til, sammen med alvorlighetsgraden knyttet til disse applikasjonene. De røde applikasjonene anses som kritiske, da de kan ha en betydelig innvirkning på den valgte applikasjonen. Applikasjonene som er markert oransje, betraktes som alvorlig og har en viss innvirkning på den valgte applikasjonen, mens applikasjonen som er markert i gult, har en lavere alvorlighetsgrad og minimal innvirkning.

Dersom brukeren klikker på en applikasjon, vil det åpne seg et nytt vindu som gir mer detaljert informasjon om applikasjonen, som vist under i figur 2.3.



Figur 2.3: Wireframe - tilleggsinformasjon om applikasjonen (se vedlegg 3: kravdokumentasjon)

2.2 Avgrensninger

Som nevnt i delkapittel 2.1.2 Initielle krav, tar prosjektet utgangspunkt i at gruppen skal lage en pilot som bedriften kan videreutvikle. Derfor er minimumskravet å kunne hente inn informasjon fra Assyst og koble dette sammen med relevant informasjon fra BizTalk Bindings files. I denne forbindelsen kommer ikke gruppen til å fokusere på visuell presentasjon av integrasjonene, men heller om løsning for å hente inn data og finne relasjon mellom dem.

2.3 Ressurser

I forbindelse med prosjektet har gruppen fått tilrettelagt enkelte ressurser fra arbeidsgiver for å kunne utføre prosjektet.

2.3.1 Interne Ressurser

For å jobbe med dette prosjektet har studentene fått tildelt arbeidsplass ved oppdragsgiverens hovedkontor, med tilgang til nødvendig utstyr og programvare. Studentene har fått tildelt en server og hver sin PC med et smartkort. Ved bruk av smartkort og VPN, har gruppen muligheten til å jobbe hjemmefra og få tilgang til bedriftens interne ressurser som Assyst og tildelt server. I tillegg fikk gruppen også mulighet til å kommunisere med oppdragsgiverne via Teams og Outlook.

Andre viktige ressurser har vært kontaktpersoner hos bedriften, som gruppen har hatt mye kontakt med - Ronny Heitmann Andersen og Nikita Tsaritson. Gruppens interne veileder, Yngve Lamo, har også vært en viktig ressurs for rapportskrivingen.

2.3.2 Assyst

Assyst er en Configuration Management Database (CMDB) som brukes til å dokumentere informasjon omkring ulike integrasjoner, systemer og andre enheter (Montgomery, 2023). Bedriften bruker dette for å få en oversikt over viktige sammenhenger og avhengigheter mellom sine applikasjoner. Assyst fungerer som den primære kilden for informasjon, og tilbyr også et REST-grensesnitt som muliggjør henting av detaljert informasjon om enheter og relasjoner i systemet.

2.3.3 BizTalk

BizTalk er en plattform som tilbyr integrasjonstjenester, blant annet programmer for orkestrering, adaptere for forskjellige kommunikasjonsformer, og data-transformasjon (Machiraju, 2018). Disse tjenestene kan initieres på flere måter, for eksempel tidsstyring og triggere. Definisjoner for en BizTalk-applikasjon kan eksporteres som en XML-basert fil som kalles en binding-fil. Binding-filer er på et lavere detaljnivå, men har sammenheng med informasjonen i Assyst. Denne filen inneholder detaljert informasjon om selve integrasjonen, inkludert informasjon om send- og receiveports (Machiraju, 2018). Ved å gruppere send- og receiveports basert på alvorlighetsgraden, kan meldingene som overføres mellom forskjellige applikasjoner enkelt håndteres og prioriteres.

3 DESIGN AV PROSJEKTET

I dette delkapittelet vil det presenteres ulike alternative løsninger knyttet til prosjektet, samt en diskusjon av alternativene.

3.1 Forslag til løsning

En løsning for automatisert modellering av integrasjonsløsninger krever omfattende datainnsamling og analyse. Det er avgjørende at løsningen kan vise relasjonene mellom dataene på en strukturert måte for å gi brukerne en oversikt over systemet. For å innhente oppdaterte data er det hensiktsmessig å bruke Binding files, Assyst og andre systemer som kan gi den nødvendige informasjonen. Dette vil bidra til å skape en mer pålitelig og effektiv løsning for modellering av integrasjonsløsninger.

3.1.1 Filformater for datahåndtering

Når informasjon hentes fra Assyst og binding files, er de i et format som er representert for det avgivende systemet. Disse formatene gjør det mulig å omstrukturere dataene på en måte som gjør det mer tilgjengelig for videre analyse og visualisering. Det er mulig å hente informasjon fra andre datakilder i andre formater, men da må det skrives et program som tolker og strukturerer disse data inn mot modellene i programmet.

3.1.1.1 XML

Løsning basert på analysering av filformat i XML-format. XML står for Extensible Markup Language (XML) og er et universelt og utvidbart markeringsspråk (Microsoft, 2021).

3.1.1.2 JSON

JSON er en populær løsning for analysering av filformat i webapplikasjoner. JSON står for JavaScript Object Notation og er et tekstbasert format for å representere strukturert data basert på JavaScript Object Syntax (MDN, u.å).

3.1.1.3 Diskusjon av alternativene

Både XML og JSON er plattformuavhengig, som betyr at de kan bli brukt på tvers av ulike operativsystemer og maskinvare, og alle programmer som støtter XML og JSON kan lese og behandle dataene. Dette har medført at både XML og JSON er blitt de mest populære teknologiene for utveksling av data mellom databaser og applikasjoner (Microsoft, u.å).

Studien gruppen har sett på sammenligner ytelsen til XML og JSON i form av datamengde, parsinghastighet og minnebruk. Når det gjelder datamengde, er JSON generelt mer kompakt enn XML, spesielt for komplekse datatyper. Dette skyldes at XML krever flere tegn for å representere data på grunn av dens markeringssyntaks, mens JSON bruker en mer nøkkelverdi syntaks (Zmaranda, Cornelia, Robert, Anca, 2018).

3.1.2 Lagring av data

Tjenesten trenger en løsning for å lagre informasjonen knyttet til datakildene, med mulighet for å hente ut nødvendige data. Det finnes flere ulike måter å løse dette på, og gruppen har diskutert med oppdragsgiver om hva som vil være den beste løsningen for prosjektet. Etter diskusjonen kom gruppen frem til flere mulige alternativer. Gruppen har avgrenset disse alternativene til 3 alternativer, hvor første valget var å lagre informasjonen som en sesjonsvariabel, hvor data kan beholdes og suppleres. Et annet alternativ var lagring på fil, mens det tredje alternativet var å lagre informasjonen i en database.

3.1.2.1 Sesjonsvariabel

Informasjon lagres som en sesjon variabel hvor dataen kan lett inspiseres “on the fly”. Da kan kommandoer bli brukt for enkelt fremvisning av informasjon (Holmes, 2021).

3.1.2.2 Filformat

Innhentet informasjon kan bli lagret lokalt som en fil. Deretter kan informasjonen bli analysert og bearbeidet for å skape en mer strukturert og oversiktlig modell av integrasjonsløsningene.

3.1.2.3 Database

En mulig løsning er å lagre informasjon om integrasjonsløsningene i en database. Dette kan gjøres ved å bruke en modelleringsteknikk som Entity-Relationship (ER) modellering for å beskrive dataene og relasjonene mellom dem. ER-modellen kan deretter bli implementert som en database ved hjelp av et RDBMS (Relational Database Management System) som MySQL eller PostgreSQL (Malik, Burney, Ahmed, 2020).

3.1.2.3 Diskusjon av alternativene

Fordelen med å lagre informasjon i en fil er at det gir større fleksibilitet og mulighet for å arbeide offline. Imidlertid kan det være utfordrende å håndtere store datamengder og sørge for at informasjonen er oppdatert.

Ved bruk av sesjonsvariabel er fordelen at man kan jobbe med variablene i minne (Holmes, 2021). Dette fører til god fleksibilitet hvis du skal jobbe med flere systemer samtidig. Flere sesjonsvariable kan benyttes og det kan legges til rette for at også en sesjonsvariabel (modell) kan være til kilde til informasjon for en annen modell.

Fordelen med å lagre informasjon i en database er at det gir mulighet for å håndtere store datamengder mer effektivt, samt muligheten for å dele informasjonen med andre systemer som kan ha nytte av den. Det gir også mulighet for å håndtere oppdateringer og endringer på en mer strukturert og kontrollert måte. Ulempen med denne løsningen er at det krever mer tid og ressurser for å opprette og vedlikeholde databasen. Dette er fordi det krever planlegging og design av en strukturert og optimal database som skal håndtere dataene på en effektiv måte. Det kreves også en god forståelse av dataene som skal lagres og hvordan de skal organiseres. For å sikre at databasen fungerer som den skal, må den administreres og vedlikeholdes jevnlig, inkludert backup og gjenoppretting av data.

3.2 Valgt løsning

Etter å ha diskutert med oppdragsgiver, ble det konkludert at sesjonsvariabel-løsningen passet best for prosjektet. Selv om en database er bedre egnet for store mengder data, valgte gruppen å jobbe ved bruk av sesjonsvariabel på grunn av fordelene omkring behandling og lagring. En annen årsak til at gruppen valgte bruk av sesjonsvariabel var den umiddelbare tilgjengeligheten, noe som var positivt med tanke på den begrensede tidsrammen.

Assyst og Biztalk benytter XML som format for informasjonsuthenting, til tross for at JSON kanskje kunne vært foretrukket. Oppdragsgiver benytter XML siden dette er formatet de fleste av standardene er representert i.

Siden oppgaven primært handler om å lage en pilot som bedriften kan videreutvikle, vil bedriften ha muligheten til å fortsette å bruke den gjeldende løsningen eller bytte til en database-løsning uten å måtte gjøre store endringer.

3.3 Valg av verktøy

3.3.1 Verktøy

Visual Studio Code
Programmet vil bli utviklet i Visual Studio Code, et utviklingsmiljø som allerede er i bruk hos bedriften og som støtter et bredt spekter av programmerings- og skriptspråk som Python, Java, PowerShell og JavaScript (Visual Studio Code, 2023). Gruppen valgte å bruke Visual Studio Code til å lage programmet ved skripting, og fant at dette utviklingsmiljøet var godt egnet til oppgaven.

Microsoft Excel
Microsoft Excel er et regnearkprogram utviklet av Microsoft. Den brukes til å organisere, analysere og presentere data på en strukturert og visuell måte (Gillish, 2021). Gruppen brukte Microsoft Excel for å lage timelister og grafer i forbindelse med prosjektet.

UMLetino
UMLetino er et gratis UML verktøy for å lage enkle UML diagrammer som gruppen brukte for å lage use-case diagrammer (UMLetino, u.å).

Figma
Figma er et prototypingsplattform som brukes til å designe brukergrensesnitt (Figma, u.å). Gruppen benyttet seg av Figma for å utforme og skape brukergrensesnittene til produktet.

Notepad++
Notepad++ er en populær tekstredigerer og kildekodeeditor som er tilgjengelig for Windows-operativsystemet (Ho, u.å). I prosjektet brukte gruppen notepad++ for å inspisere innholdet og relasjonene mellom applikasjonene fra Assyst og BizTalk binding files.

PowerShell:
PowerShell er en kommandolinje og et skriptspråk som er vanlig å bruke i et Windows miljø. (Microsoft, 2022). Dette er et verktøy som blir brukt for å automatisere oppgaver, og gruppen har brukt dette i sammenheng med å innhente kildedata og skripting av programmet.

3.3.2 Samarbeidsverktøy

Google Docs
Google Docs tillater live og åpent samarbeid, og dette verktøyet ble benyttet av gruppen til å dokumentere prosjektet.

Trello
Trello har gruppen benyttet for å lage kanban board som kan visuelt deles med andre angående prosjektmål i utviklingsprosessen.

Discord
Gruppen benyttet Discord som en kommunikasjonsplattform for å snakke sammen, planlegge og diskutere oppgaven. Selv om Discord er markedsført hovedsakelig for dataspill-relatert bruk, tilbyr plattformen gode verktøy for kommunikasjon og deling av informasjon, som gjorde den til et viktig verktøy for gruppen.

Microsoft Teams
Microsoft Teams er en kommunikasjonsplattform for tekst, lyd og video som ble benyttet av gruppen for kommunikasjon innenfor bedriften og interne veileder.

Zoom Meetings
Gruppen benyttet Zoom Meetings som standard kommunikasjonsplattform da det ble digitale møter sammen med intern veileder.

Outlook
For å planlegge møtene, anbefalte gruppens interne veileder å legge inn møteinnkallinger i kalenderen på Outlook. Dette gjorde det enklere å planlegge og organisere møtene, samt å holde oversikt over møteplanen.

3.4 Prosjektmetodikk

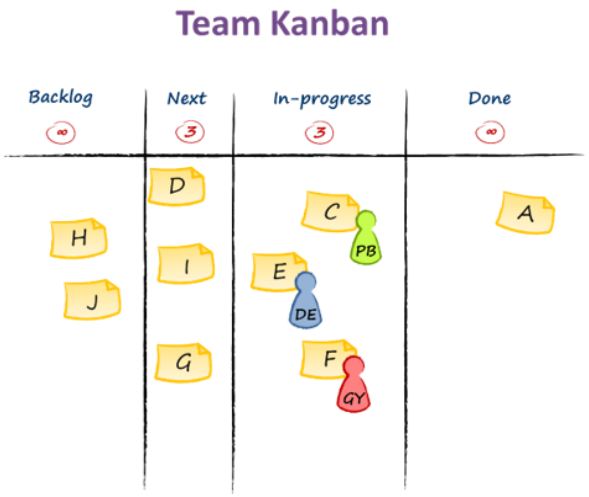
3.4.1 Utviklingsmetodikk

I utgangspunktet vurderte gruppen å bruke Scrum som arbeidsmetode, men etter veilederens anbefaling om at dette ikke var aktuelt, valgte gruppen å organisere arbeidet i faser med underfaser og mål. Som en enkel fremgangsmåte som passet godt til gruppens behov, ble Kanban Board valgt som arbeidsmetode. Siden prosjektet allerede var påbegynt passet Kanban Board utmerket for allerede påbegynt arbeidsflyt, da Kanban Board kunne forbedre arbeidsflyten for arbeidet uten å forstyrre strukturen i organisasjonen (Anderson, 2016).

Et av de største fordelene med denne utviklingsmetoden er at det gir et visuelt bilde av arbeidsflyten, som gjør det lettere for teamet å se hvor oppgavene står og hva som må gjøres. Dette gjør det enkelt å identifisere forhindringer i arbeidsflyten og å tilpasse seg endringer i prosjektet. For eksempel, hvis det oppstår en forsinkelse på ett trinn i

arbeidsflyten, kan teamet enkelt se hvordan dette vil påvirke hele prosjektet og justere arbeidsflyten deretter.

Kanban board kan også bidra til å redusere stress og øke produktiviteten i teamet. Ved å ha en visuell oversikt over arbeidsflyten og oppgavene som må gjøres, kan teamet enkelt identifisere hva som må prioriteres og hvor mye tid som må tildeles hver oppgave. Dette kan bidra til å redusere stressnivået ved å sørge for at alle er på samme side når det gjelder hva som må gjøres og når det må gjøres. Det kan også øke produktiviteten ved å hjelpe teamet med å fokusere på oppgavene som må gjøres og ved å unngå at tid og ressurser blir brukt på unødvendige oppgaver.



Figur 3.1: Eksempel på Kanban board (Anderson, 2016)

3.4.2 Prosjektplan

Et Gantt-diagram er en type tidsplan som brukes til å visualisere aktiviteter, oppgaver og tidsrammer i et prosjekt (Gantt, u.å). Dette ble benyttet for å planlegge fremdriftsplanen for prosjektet både for innleveringer, rapportskriving og for utvikling av piloten. Den ble revidert inn i flere iterasjoner hvor siste versjon er tilgjengelig i prosjekthåndboken for prosjektet (vedlegg 1: Figur 1.5). Diagrammet ble benyttet av gruppen for å gi en oversikt over hvilke arbeid som skal gjøres og til hvilke tider de forskjellige aktiviteter skal gjennomføres. Diagrammet viser forventet tidsbruk knyttet til hver ulik aktivitet, men ettersom den lages tidlig i prosjektet, blir gantt-diagrammet endret og justert flere ganger.

Fremdriftsplanen er inndelt i forskjellige hovedaktiviteter:

Oppstartsfase: Fasen gjelder for starten av prosjektet hvor gruppen tar kontakt med oppdragsgiver og veileder for å få en mer forståelse for oppgaven.

Forprosjektfase: I forprosjektfasen henter gruppen inn nødvendig informasjon knyttet til prosjektet, planlagte design for løsningen og formulerte krav og avgrensninger til prosjektet.

Hovedprosjektfase: i hovedprosjektfasen skal gruppen ferdigstille prosjektet både når det kommer til produktet, rapporten og støttedokumenter.

Systemutvikling: Systemutviklingsfasen omfatter utviklingen av systemet og varer fra slutten av forprosjektfasen og gjennom hele hovedprosjektfasen. I løpet av denne fasen skal gruppen benytte Kanban Board som utviklingsmetode for å planlegge og organisere arbeidet.

Kompetanseinnhenting: Kompetanseinnhenting er en aktivitet som strekker seg gjennom hele prosjektet, hvor gruppen innhenter nødvendig kompetanse som er nødvendig for å utføre prosjektet.

3.4.3 Risikovurdering

Tabell 3.1: Risikomatrise fra (Infoklikk, 2016)

Konsekvens: Sannsynlighet:	1. Ubetydelig	2. Mindre alvorlig/ En viss fare	3. Betydelig/ Kritisk	4. Alvorlig/farlig	5. Svært alvorlig/ katastrofalt
5. Svært sannsynlig	5	10	15	20	25
4. Meget Sannsynlig	4	8	12	16	20
3. Sannsynlig	3	6	9	12	15
2. Mindre sannsynlig	2	4	6	5	10
1. Lite sannsynlig	1	2	3	4	5

I risikoanalysen ble alvorlighetsgraden av risiko vurdert basert på sannsynligheten for at noe skulle skje og hvordan konsekvensene vil påvirke prosjektet. Risikoanalysen er gradert fra 1-5 i sannsynlighet og konsekvens (Infoklikk, 2016). Risikofaktoren for en hendelse blir deretter validert ved å multiplisere sannsynligheten med konsekvensen. Deretter blir de identifisert i en risikomatrise der den totale risikofaktoren for prosjektet er synlig. I gruppens risikoanalyse er det foreslått tiltak for å minimere både sannsynligheten og konsekvensene av risikoene i prosjektet. Gjennom hele prosjektet vil det bli gjennomført kontinuerlig evaluering og vurdering av eventuelle risikoer.

Tabell 3.2: Risikotabell (Vedlegg 1)

nr	Hendelse /Risiko	Årsak	S	K	RP	Tiltak
1	Ufullstending produktleveranse	Ineffektiv arbeidsmetodikk og ambisiøse mål.	3	4	**12**	Sette realistiske og håndterbare delmål, samt forbedre planleggingen av arbeidet.
2	Sykdom	Manglende hygienepraksis	3	3	**9**	Opprettholde god hygiene, inkludert hyppig håndvask og fokus på personlig helse.
3	Manglende kompetanse innenfor området	Begrenset kunnskap.	4	4	**16**	Utføre grundig forskning, søke veiledning og utvikle nødvendig kompetanse for å løse utfordringene.
4	Misforståelse av oppgaven	Manglende kommunikasjon med oppdragsgiver	2	4	**8**	Ha jevnlig møter med oppdragsgiver for å klargjøre oppgavens mål og krav.
5	Ikke oppfylt krav til produktet	Begrenset kompetanse og kunnskap.	3	3	**9**	Kontinuerlig utvide kompetansen innen relevante programmeringsverktøy for å møte kravene.
6	Noen av skriptene får feilmelding eller stopp ved kjøring.	Det kan være noe feil i piloten eller mangel på tilganger/rettigheter.	3	3	**9**	Jevnlig teste programmet for å identifisere og håndtere potensielle feil og avbrudd.
7	Begrenset tilgang til nødvendige ressurser	Konfidensialitet og begrensninger i ressurstilgang	5	2	**10**	Klarlegge og kommunisere behovene til oppdragsgiver for å sikre tilgang til nødvendige ressurser og systemer.
8	Manglende implement-ering av kravene	Manglende ferdigheter og kompetanse	3	4	**12**	Opprettholde god kommunikasjon med oppdragsgiveren og sette grenser for prosjektet når det er nødvendig, slik at det kan oppnås et vellykket resultat.

3.5 Evalueringsplan

3.5.2 Ytelsestest

I dette prosjektet vil gruppen utføre en ytelsestest for å evaluere hvordan PC-en fra bedriften håndterer den store mengden data. Hovedmålet med testen er å undersøke om det oppstår noen betydelig forsinkelse når man henter inn dataen.

3.5.3 Sluttevaluering

Avslutningsvis vil det bli gjennomført en brukertest som er gjennomført av oppdragsgiver for å sjekke om det oppfyller kravene. Testen vil også inneholde et evalueringsskjema for å se om prosjektets mål har blitt oppfylt.

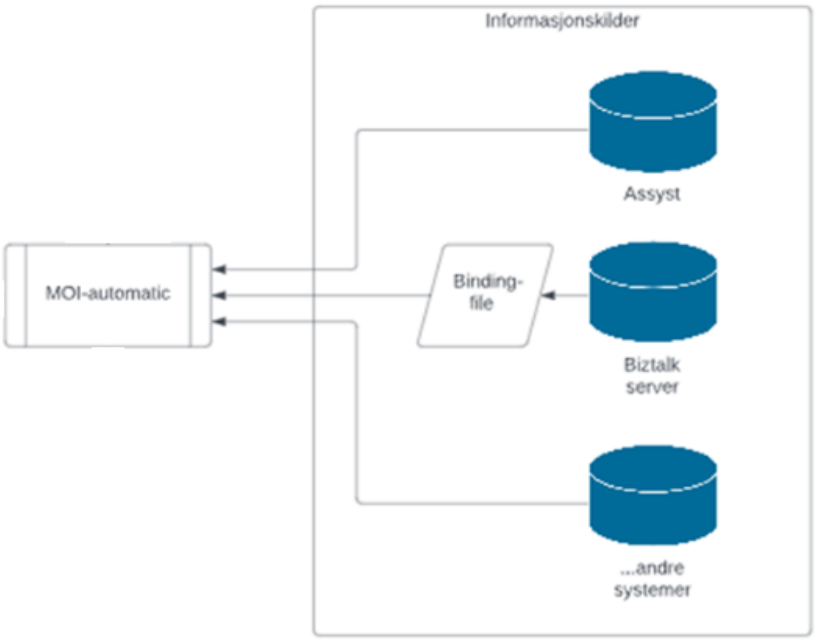
4 DETALJERT LØSNING

Dette kapittelet tar for seg metodene som ble benyttet for å gjennomføre prosjektet, og gir en omfattende oversikt over løsningen, prosessen og den konseptuelle arkitekturen av systemet. I kravdokumentet for prosjektet (vedlegg 3) finnes det et brukstilfellediagram med tilhørende brukstilfellebeskrivelser samt en domenemodell. I systemdokumentasjonen (vedlegg 5) er det også inkludert en grundig beskrivelse av løsningens design og implementasjon. I dette kapittelet vil enkelte navn til integrasjoner og applikasjoner bli sensurert på grunn av konfidensialitet.

4.1 Bakgrunn for design

Programmet er designet med formålet om å automatisk modellere relasjoner mellom systemene hos Helse Vest IKT, da det ikke finnes en automatisert løsning til dags dato. For å oppnå dette ble det nødvendig å samle og knytte informasjon fra ulike systemer for å vise de relevante relasjonene. Arbeidet inkluderte å analysere den omfattende informasjonsmengden i systemene og finne en strukturert måte å presentere informasjonen i programmet. Dataflyten fra ulike informasjonskilder kan sees i figur 4.1.

For at programmet skal bli pålitelig og nøyaktig er det viktig at den innhentede informasjonen er riktig. Dette ble gjort ved å ha mange møter og diskusjoner med oppdragsgiver om informasjonen som lå på systemer som Assyst og BizTalk binding-files, hvor informasjonen ble deretter avgrenset til bare de mest aktuelle for relasjoner.



Figur 4.1: Overordnet skisse for informasjonsflyt

4.2 Informasjonsinnhenting og behandling

4.2.1 Assyst

Assyst er den primære kilden for informasjon og tilbyr også et REST-grensesnitt hvor det kan hentes ut detaljert informasjon omkring enheter og relasjoner i systemet. Miljøet det arbeides i er et Microsoft-basert driftsmiljø, så valget falt på bruk av det som allerede er i bruk – powershell og funksjonaliteten som ligger i skriptspråket.

HentAssyst.ps1

For å innhente informasjon fra Assyst måtte gruppen lage et powershell-skript som hentet informasjon omkring en integrasjon basert på dens unike ID i Assyst. Når gruppen kjører skriptet i PowerShell, ble det hentet ut informasjon fra Assyst og deretter skrevet til en XML-fil lokalt på maskinen.

Attributter

For å finne ut hvilken informasjon som er relevant og betydningsfull for produktet *MOI-Automatic*, har gruppen analysert attributtene fra integrasjoner fra Assyst. Programmet henter informasjon om integrasjonen for å kunne koble relasjonene på best mulig måte. Disse delene er delt inn i: shortCode og ID, produkt og relasjoner.

shortCode og ID

Entitene eller objektene fra Assyst er definert ved *ID og shortCode*. Ved bruk av disse attributtene, vil det bli mulig å hente ut informasjon fra Assyst.

- **ID** - blir brukt som en unik identifikator mellom objekter i Assyst.
- **ShortCode** - blir brukt for definering av unikt navn i systemet. Det benyttes også som oppslagsverdi for Assyst.

Produkt

Product er en del av definisjonen av objekter i Assyst. Produkter tilhører en produktklasse, som igjen tilhører en generisk klasse. Dermed får man et produkthierarki som består av tre deler:

- **Product** - gruppering for å samle alle applikasjonsdelene av et system og samle systemer som benytter samme applikasjon eller tjeneste.
  Eksempler: *SQL database, virtuell server og brannmur.*
- **productClass** - definerer hvilken gruppe av applikasjonstjenester de skal tilhøre.
  Eksempler: *Databaser, applikasjoner og nettverk.*
- **genericClass** - Generell inndeling av produkter i CMDB. (Inndeling av product)
  Eksempler: *Infrastruktur, integrasjoner og applikasjoner.*

Relasjoner

Manglende tilgang til disse attributtene vil hindre innsikten i hvilke applikasjoner den gitte applikasjonen har relasjoner til. Det er derfor avgjørende for prosjektet at disse attributtene inkluderes i analysen.

- **RelatedType** - forteller noe om avhengigheten mellom integrasjonene.
  Tilgjengelige valg er: *A AVHENGIG AV, B NØDVENDIG FOR, UKJENT*
- **RelatedName** - beskriver alvorlighetsgraden til integrasjonen.
  Tilgjengelige valg er: *A KRITISK, B ALVORLIG, C MINDRE VIKTIG.*
- **RelatedItemId** - viser til navnet til integrasjonen som produktet er relatert til.
  Eksempel på valg: *BIZTALK 2013 STAGE HVN og SMS.AGFA-SMSPAAMINING.HBE STAGE*

4.2.2 BizTalk binding-files

På grunn av begrensninger i omgivelsene var det ikke mulig å få direkte tilgang til BizTalk for uthenting av informasjon. Derfor ble informasjon om BizTalk hentet ut på fil (binding-files) og oversendt til gruppen for analyse. Etter en analyse av disse filene fant gruppen tre interessante attributter:

- **ModuleRef Name = "[Application: "*navnet til applikasjonen*"]**"- applikasjonsnavnet viser til navnet til applikasjonen binding filen omhandler og er definert som en modul.
- **sendPort** - inneholder lokasjon/er (fileshare/webservice osv) der integrasjonen sender data til.
- **receivePort** - inneholder lokasjon/er (fileshare/webservice osv) der integrasjonen henter data fra.

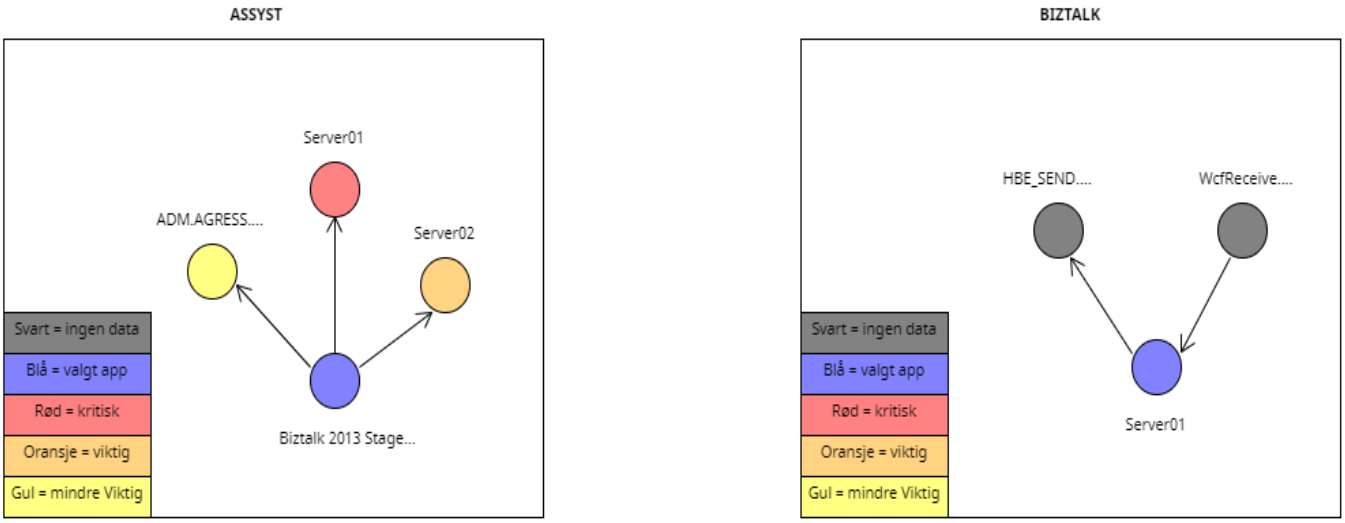
4.2.3 MOI-Automatic

Når informasjonen er analysert, kan man sammenligne informasjonen som ligger mellom systemene. I figur 4.2 vises det en tydelig likhet mellom informasjonen fra de to kildene. Det er at "[Application:Server01]" fra BizTalk er tilsvarende "shortCode" i Assyst. Den røde linjen viser til dette eksempelet at begge filene omhandler samme applikasjon da de har samme applikasjonsnavn, og dette skaper dermed en kobling mellom dem.

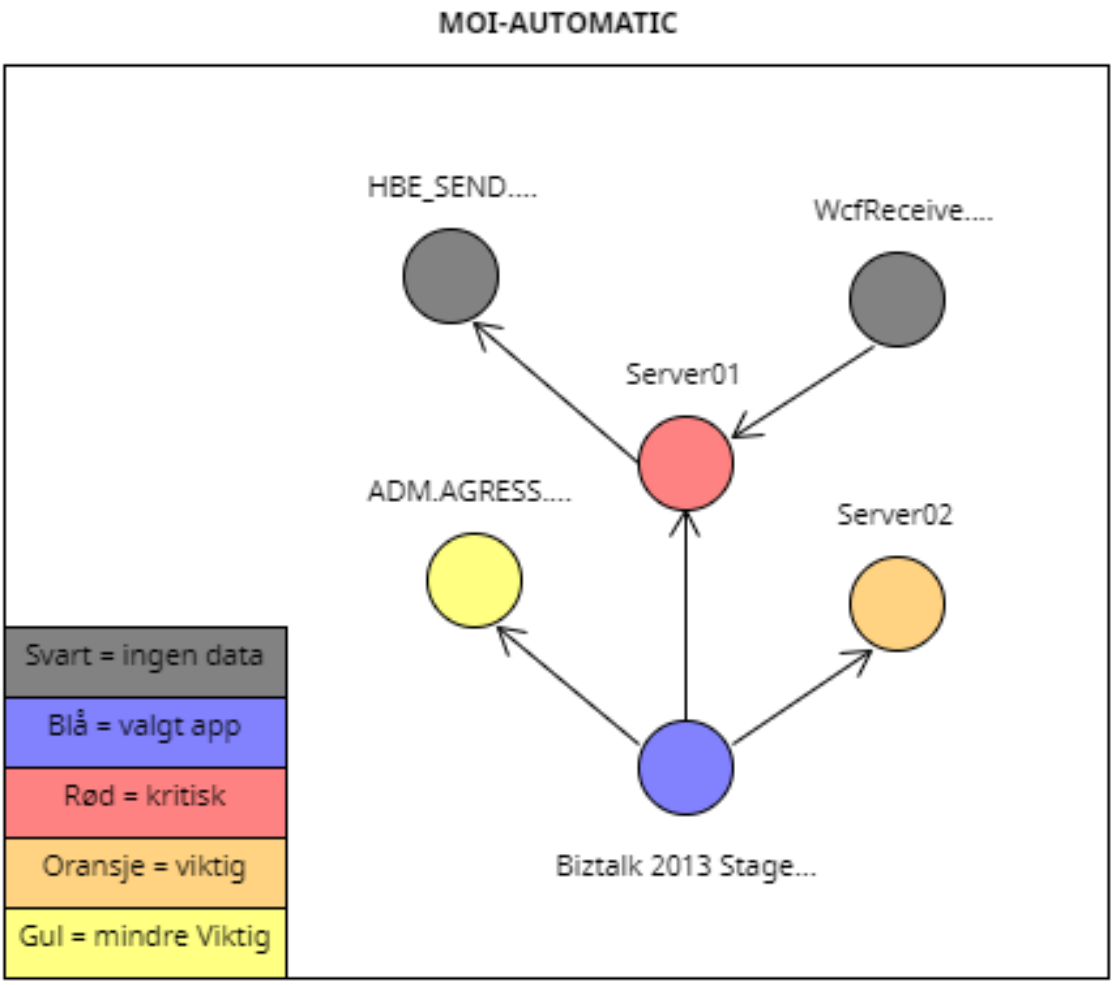
Figur 4.2: Utdrag av en applikasjon i Assyst (venstre) og en Biztalk Binding file (høyre) i XML-fil.

Gjennom bruk av både *sendPort* og *receivePort* kan det bli vist detaljert informasjon om lokasjonene som integrasjonen sender data til, hva slags data som blir overført (slik som filer eller webtjenester), og hvor integrasjonen mottar data fra (Machiraju, 2018). Ved å samle inn denne informasjonen kan det etableres relasjoner mellom disse systemene, som potensielt kan knyttes til relasjoner fra Assyst eller andre systemer.

Ved å kombinere datakildene fra de to informasjonskildene, vil *MOI-Automatic* ha tilgang til sammensatt informasjon som et sluttresultat etter innhenting og analyse av dataene. Et eksempel på dette er vist nedenfor i figur 4.3 og 4.4.

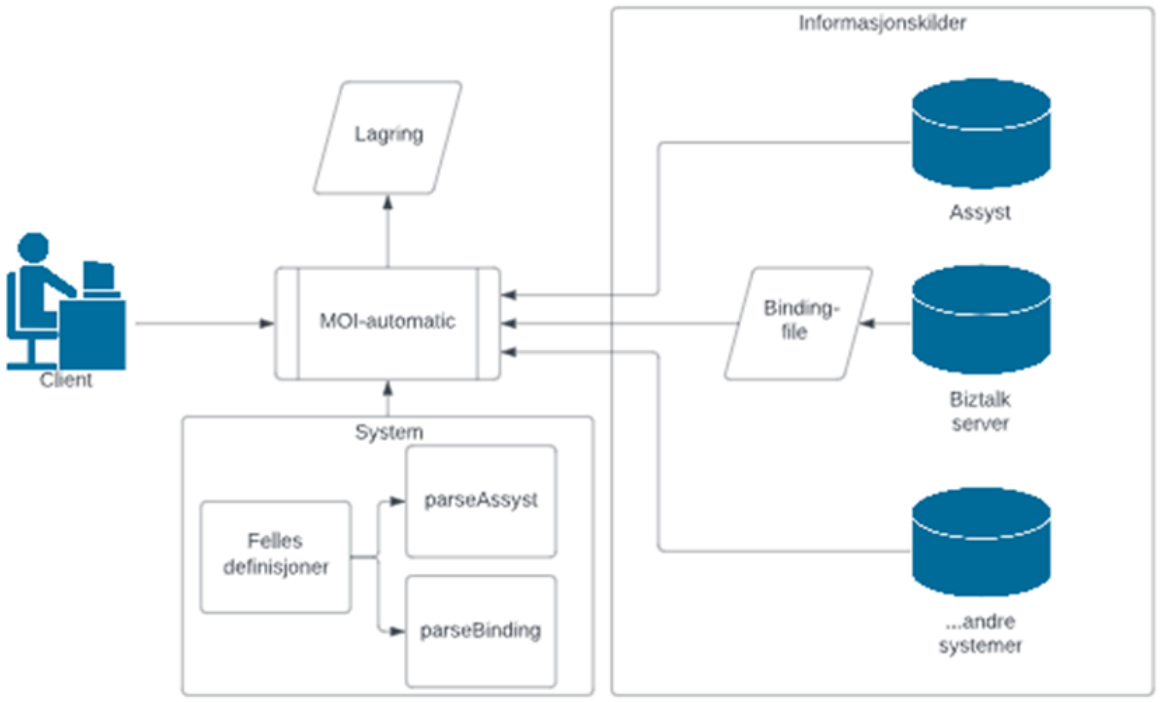


Figur 4.3: Viser til ulik informasjon fra to forskjellige systemer (Assyst og BizTalk).



Figur 4.4: Resultat: kobler sammen informasjon fra forskjellige systemer i MOI-Automatic.

4.3 Arkitektur



Figur 4.5 : Overordnet skisse for arkitektur av systemet (Se vedlegg 4, Systemdokumentasjon)

4.3.1 System

I senter av systemet befinner det seg en felles modul som er koblet til både *parseAssyst* og *parseBinding*. Disse parse-skriptene kjøres ved innhenting av data, hvor resultatet lagres i en array ved hjelp av sesjonsvariabler. *Felles definisjoner* er en fellesmodul som skal vise den sammenstilte informasjonen i en modell.

4.3.2 Modellen - *felles*

Dette er en modul som inneholder fellesfunksjoner som brukes av parse-skriptene i *MOI-Automatic*. Modulen brukes til å opprette og finne objekter og knytte relasjoner mellom de forskjellige skriptene som henter informasjon fra ulike systemer. Ved å bruke fellesmodulen kan alle skript hente ønskede objekter og koble dem sammen på en strukturert måte mot variabler fra Assyst. På denne måten kan informasjon fra Assyst brukes til å fylle inn mangler når det blir laget skript som henter informasjon fra andre systemer. Formålet med å bruke en modul er å organisere koden og gjøre den mer håndterlig ved å bryte den ned i mindre deler. Modulen importeres i begynnelsen av skriptene ved å bruke følgende kommando: ". .\felles.ps1".

Figuren 4.6 oppretter et objekt ved hjelp av funksjonen "createObject()". Skriptet definerer forskjellige egenskaper for objektet, inkludert generisk ID, ID, navn, kilde, avhengighetssjekk (depedencyDiscovery) og produkthierarki. Det legger også til en hashmap av egenskaper, en array av relaterte objekter og navn på send- og receiveport.

```
20  function createObject() {
21      $object = New-Object -TypeName PSObject
22
23      Add-Member -InputObject $object -MemberType NoteProperty -Name GenID -Value "" # Generisk ID for alle typer refera
24      $object.GenID = "id-"+[guid]::NewGuid()
25      Add-Member -InputObject $object -MemberType NoteProperty -Name id -Value ""  # Identifikator for artifakt. (Assyst
26      Add-Member -InputObject $object -MemberType NoteProperty -Name name -Value ""  # Navn på artifaktet - eks Assyst i
27
28      Add-Member -InputObject $object -MemberType NoteProperty -Name SRC     -Value ""  # Kilde/system for dette artifak
29      Add-Member -InputObject $object -MemberType NoteProperty -Name SRCref  -Value ""  # Kildereferanse for dette artif
30
31      # Har denne blir sjekket for avhengigheter?  Hvis ikke, kan dette være 'en løs tråd'.
32      # Eks. kan man lage et objekt basert på info fra Assyst, men det er ikke sikkert vi har sjekket alle avhengigheter
33      Add-Member -InputObject $object -MemberType NoteProperty -Name dependencyDiscovery -Value $false
34
35      # Produkthierarki (har ikke tatt med id'er) - her har vi det meste fra Assyst.
36      Add-Member -InputObject $object -MemberType NoteProperty -Name product         -Value $null  # product.shortCode
37      Add-Member -InputObject $object -MemberType NoteProperty -Name productClass    -Value $null  # product.productClas
38      Add-Member -InputObject $object -MemberType NoteProperty -Name genericClass    -Value $null  # product.productClas
39
40      Add-Member -InputObject $object -MemberType NoteProperty -Name Properties      -Value @{}  # Hasmap of properties
41      Add-Member -InputObject $object -MemberType NoteProperty -Name Related -Value @() # Array of related objects
42      Add-Member -InputObject $object -MemberType NoteProperty -Name sendPortCollectionName    -Value $null  # SendPortL
43      Add-Member -InputObject $object -MemberType NoteProperty -Name receivePortCollectionName -Value $null  # ReceivePo
44      $object.Related = @()  # Array av relasjoner fra dette objektet.
45      return $object
46  }
```

Figur 4.6: Skjermdump av felles.ps1 hvor fellesmodulen oppretter objekter.

‘

```
64  function createRelationObject() {
65      $object = New-Object -TypeName PSObject
66      # Oppretter en beskrivelse om et objekt man har en relasjon til
67
68      Add-Member -InputObject $object -MemberType NoteProperty -Name GenID -Value "" # Generell ID
69      $object.GenID = "id-"+[guid]::NewGuid()
70
71      Add-Member -InputObject $object -MemberType NoteProperty -Name RelatedType -Value $null # relations.relatedDetail.id
72      Add-Member -InputObject $object -MemberType NoteProperty -Name RelatedName -Value $null # relations.relatedDetail.name
73
74      Add-Member -InputObject $object -MemberType NoteProperty -Name RelatedItemId -Value $null # KEY til relatert objekt.
75
76      Add-Member -InputObject $object -MemberType NoteProperty -Name RelationSource -Value $null # Hvor fant vi denne relasjonen
77
78      return $object
79  }
```

Figur 4.7: Skjermdump av felles.ps1 hvor det fellesmodulen oppretter relasjonsobjekter.

4.3.3 Skript - *parseAssyst*

Dette skriptet tar to parametere: $fileName og $currentModel. $fileName-parameteren brukes til å spesifisere stien til en XML-fil som inneholder data om et objekt, mens $currentModel-parameteren brukes til å lagre en array av objekter som representerer modellen som arbeides på.

```
24  Param(
25      [Parameter(HelpMessage="Input fil")][string] $fileName
26      ,[Parameter(Mandatory=$true,HelpMessage="Gjeldende model")][AllowEmptyCollection()][collections.arraylist]$currentModel
27  )
```

Figur 4.8: Skjermdump i parseAssyst-skriptet hvor de to parameterne blir brukt

Skriptet leser XML-filen som er spesifisert av $fileName og trekker informasjon om objektet den representerer. Hvis objektet ikke allerede finnes i $currentModel-arrayet, legger skriptet det til i arrayet.

Skriptet sjekker også om det er noen relasjoner som objektet har med andre objekter. Hvis en relasjon eksisterer, oppretter skriptet et nytt objekt for den relaterte relasjonen dersom den ikke allerede finnes i $currentModel-arrayet. Skriptet oppretter deretter et relasjonsobjekt som representerer forholdet mellom de to objektene og legger det til begge objektene.

Oppsummert er hovedformålet med skriptet å lese XML-filer som inneholder data om objekter og opprette tilsvarende objekter i en array som representerer den nåværende modellen. Skriptet håndterer også opprettelsen av relasjoner mellom objekter. Formålet med modulen *felles,* er å gi fellesfunksjoner som skriptet bruker for å oppnå disse oppgavene.

4.3.4 Skript - *parseBinding*

Dette er et skript som fungerer på samme måte som *parseAssyst*, men forskjellen er at skriptet henter inn data og objekter fra Binding files i stedet. Dette fører til at skriptet supplerer den manglende informasjonen med informasjon som ligger i Assyst.

4.3.5 Lagring

Programmet tar i bruk sesjonsvariabler, men siden disse blir kjørt i minnet, risikerer informasjonen å bli tapt ved restart av sesjonen. Derfor har programmet en lagringsmetode for å sikre at dataene blir lagret og ikke går tapt. Resultatet av dette er to skriptfiler, hvor det ene skriptet lagrer modellen, og den andre laster opp modellen for videre arbeid.

saveModel - lagrer modellen (sesjonsvariabel) til fil. På denne måten kan arbeidet som blir gjort i sesjonsminnet bli lagret, slik at dette arbeidet ikke forsvinner.

LoadModel - henter inn en fil med lagret informasjon av sesjonsvariabel.

4.3.6 Fremvisning av informasjon

Når programmet brukes, må brukeren først opprette en tom tabell i en sesjonsvariabel. Deretter må brukeren bruke et av parseskriptene for å innhente data om det angitte systemet for å så plotte informasjonen inn i en tabellen. Som resultatet av dette, kan det bli brukt kommandoer for å fremvise informasjon om relasjoner og applikasjoner som da ligger tilgjengelig i tabellen:

ParseAssyst

- **$a[1]** viser objekt nummer to i sesjonsvariablelen (Arrays starter på index 0). Denne informasjonen viser til forholdet mellom objekt nummer 1 (BIZTALK STAGE 2013 HVN) og objektnummer 2 (KLIN.CARESTREAM_DIPS.HST.STAGE).

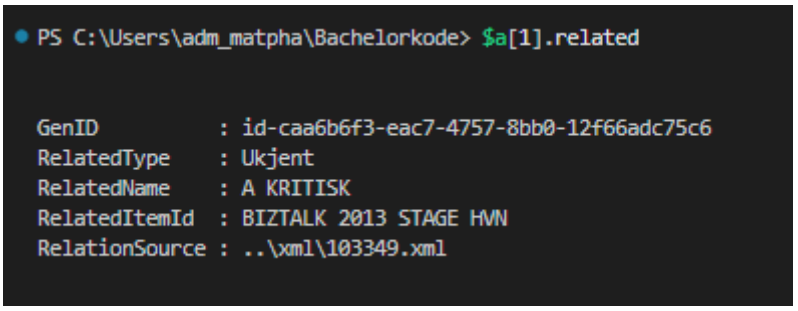
```
PS C:\Users\adm_matpha\Bachelorkode> $a[1]


GenID              : id-191cc903-2f9c-4ad9-8046-dac355b2a432
id                 : KLIN.CARESTREAM_DIPS.HST
name               : KLIN.Carestream_DIPS.HST.Stage
SRC                : ASSYST
SRCref             : ..\xml\103349.xml
dependencyDiscovery : False
product            : KLIN.CARESTREAM_DIPS.HST
productClass       : BIZTALK APPLIKASJONER
genericClass       : INTEGRASJONER
Properties         : {}
Related            : {@{GenID=id-caa6b6f3-eac7-4757-8bb0-12f66adc75c6; RelatedType=Ukjent; RelatedName=A KRITISK; RelatedItemId=BIZTALK 2013 STAGE HVN
                     ; RelationSource=..\xml\103349.xml}}
```

Figur 4.9: Skjermbilde av terminalen når kommandoen blir kjørt.

Her blir det vist informasjon som for eksempel om *ID, product, ProductClass, genericClass og Related.* Basert på denne informasjonen fremgår det at applikasjonen BizTalk 2013 STAGE HVN har en kritisk rolle og har stor påvirkning på den applikasjonen den har en relasjon til, som er KLIN.Carestream_DIPS.HST.

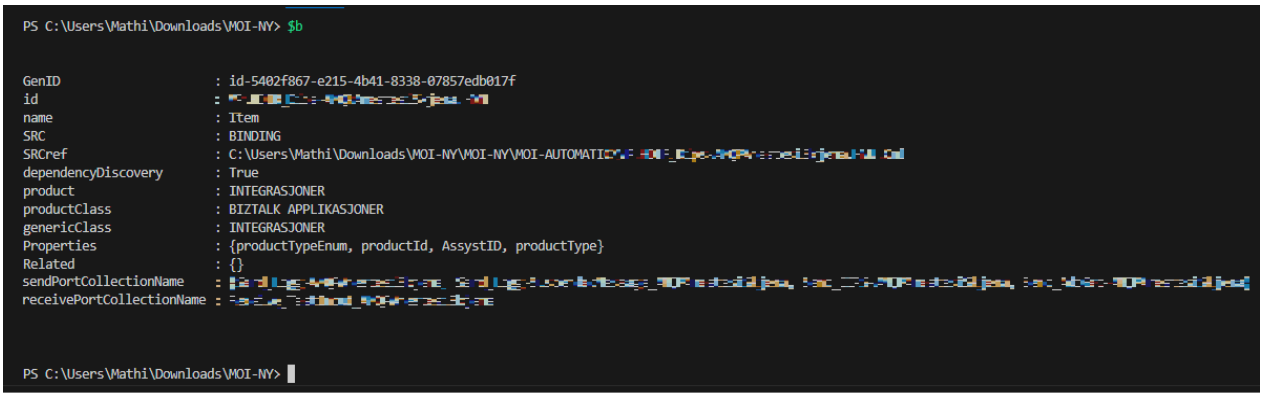
- **$a[1].related** - viser til informasjonen på en mer strukturert måte av informasjonen som bare ligger på *Related.*



Figur 4.10: Skjermbilde av terminalen når kommandoen blir kjørt.

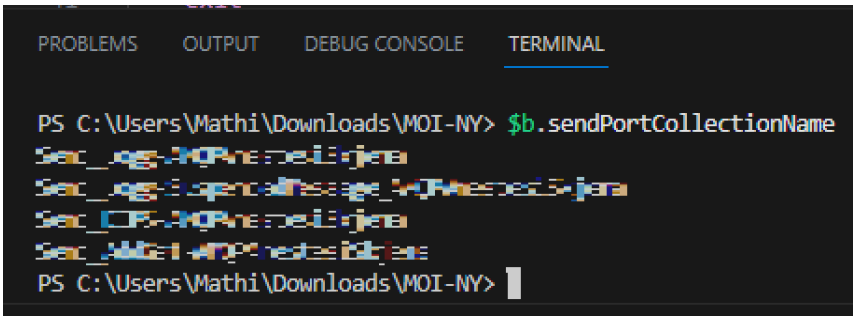
ParseBinding

- **$b** viser til all informasjon om den valgte binding filen.



Figur 4.11: Kommandoen viser til binding-filen med informasjon i terminalen

- **$b.sendPortCollectionName -** viser navnene på sendport-lokasjoner som er angitt i en tilfeldig valgt binding-fil for å gi en mer oversiktlig presentasjon.



Figur 4.12: Kommandoen som viser til sendport-lokasjon mer oversiktlig i terminalen

5 RESULTATER

Dette kapittelet fokuserer på evaluering og testing av programsystemet i prosjektet. Gjennom ulike evalueringsmetoder og tester sikrer gruppen at programmet oppfyller oppdragsgiverens krav og fungerer som forventet.

5.1 Evalueringsmetode

Som tidligere nevnt, vil prosjektet benytte seg av to evalueringsformer for å vurdere fremdriften og måloppnåelsen. Disse inkluderer ytelsestest, brukertest samt en sluttevaluering fra oppdragsgiver. Ved å kombinere disse evalueringene, vil gruppen få et helhetlig overblikk over prosjektets gjennomføring i forhold til planlagt funksjonalitet og målsettinger.

5.1.1 Ytelsestest

For å evaluere programmets hastighet ble PowerShell-kommandoen *Measure-Command { skriptet }* benyttet. Dette måler utførelsestiden til et gitt skript eller en kommando og gir informasjon om tidsbruken, inkludert total tid og CPU-tid. Dette hjelper med å evaluere og sammenligne ytelsen til forskjellige skript og optimaliseringer (Microsoft, 2023).

5.1.2 Brukertest

En av oppdragsgiverne gjennomførte brukertesten for å sikre at programsystemet oppfyller kravene. Brukertesten ga gruppen verdifull tilbakemelding om brukervennlighet, funksjonalitet og eventuelle forbedringsområder i programmet. Testresultatene brukes til å identifisere og adressere eventuelle utfordringer eller behov for justeringer, og bidrar dermed til å sikre at det endelige produktet møter oppdragsgiverens forventninger og behov.

5.1.3 Sluttevaluering

I sluttevalueringen har en av oppdragsgiverne gitt en vurdering av gruppens prestasjoner og prosjektets utførelse. Skjemaet vil dekke ulike aspekter av prosjektet, inkludert prosjektplanlegging, implementering av funksjonalitet, kvaliteten på kode og dokumentasjon, samarbeid og kommunikasjon, og tilfredsstillelse av prosjektmålene.

5.2 Evalueringsresultat

Evaluering av ytelsestest

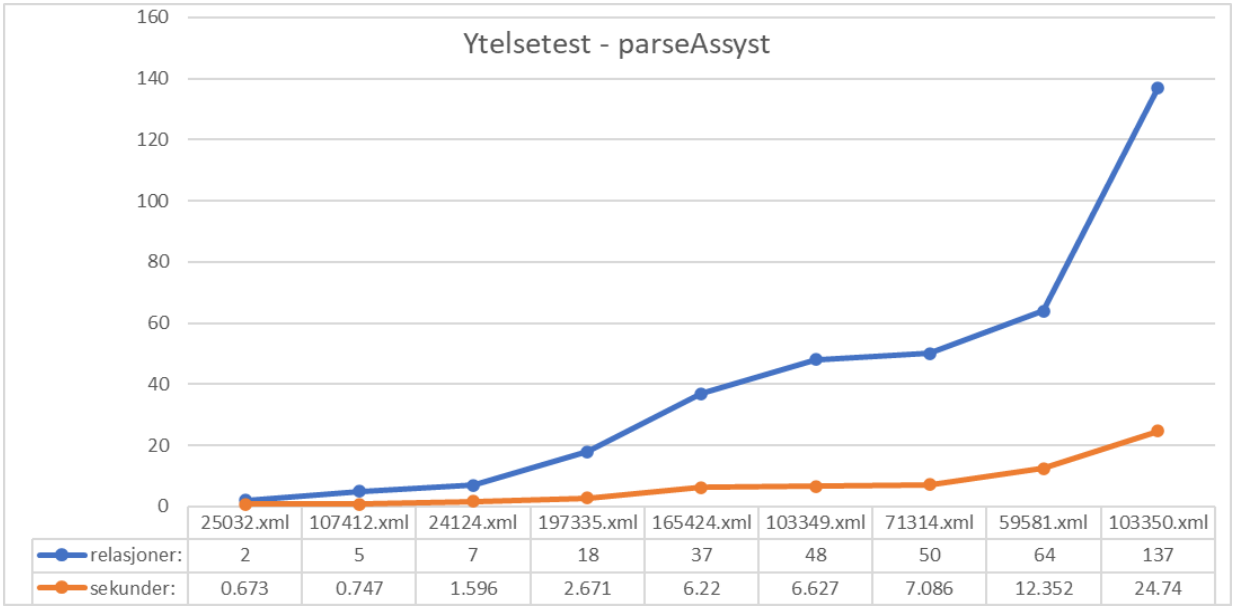
Testen ble utført på en server som hadde følgende spesifikasjoner:

RAM: 12GB
Hastighet: 2.60 GHZ
Prosessor: Intel(R) Xeon(R) Gold 6126 CPU

I figur 5.1 vises resultatet når *Measure-Command { parseAssyst.ps1 }* kjøres i terminalen for en fil med 137 relasjoner. Outputen gir innsikt i den tidsmessige ytelsen til skriptet under analysen av en relativt stor og kompleks fil.

```
Days              : 0
Hours             : 0
Minutes           : 0
Seconds           : 24
Milliseconds      : 739
Ticks             : 247397574
TotalDays         : 0,000286339784722222
TotalHours        : 0,00687215483333333
TotalMinutes      : 0,41232929
TotalSeconds      : 24,7397574
TotalMilliseconds : 24739,7574
```

Figur 5.1: Resultatet av en fil med 137 relasjoner.



	25032.xml	107412.xml	24124.xml	197335.xml	165424.xml	103349.xml	71314.xml	59581.xml	103350.xml
relasjoner:	2	5	7	18	37	48	50	64	137
sekunder:	0.673	0.747	1.596	2.671	6.22	6.627	7.086	12.352	24.74

Figur 5.2: Graf som viser ytelsestest fra 9 filer - hvor den måler relasjoner opp mot innhentingstid.

Resultatene fra ytelsestesten, presentert i figur 5.2, gir verdifull innsikt i skriptets ytelseskapasitet når det behandler 9 XML-filer med varierende relasjoner. Gruppen observerte at kjøretiden økte proporsjonalt med antallet relasjoner, noe som indikerer en sammenheng mellom kompleksiteten i dataene og behandlingstiden. Gjennom ytelsestesten fikk gruppen muligheten til å evaluere skriptets effektivitet og vurdere dets skalerbarhet i håndteringen av ulike filstørrelser og kompleksiteten i relasjonene.

Det er viktig å merke seg at foreløpig viser ytelsestesten kun tiden det tar for skriptet å behandle applikasjoner og deres direkte relasjoner (første ledd). Skriptet henter ikke informasjon fra de sekundære relasjonene, det vil si relasjonene til relasjonene til hovedapplikasjonen. Hvis gruppen hadde gått videre til det tredje leddet, ville skriptet trolig brukt mer tid, og det er mulig at det ikke ville vært like skalerbart.

Denne begrensningen i ytelsestesten gir rom for videre utforskning og evaluering av skriptets kapasitet til å håndtere flere ledd med relasjoner. Det kan være interessant å undersøke hvordan skriptet oppfører seg når det jobber med flere nivåer av relasjoner og vurdere om det forblir skalerbart under slike forhold.

Evaluering av brukertest

En av oppdragsgiverne, Nikita Tsaritson, gjennomgikk programmet mens gruppen observerte. Ved å observere kunne gruppen hjelpe og veilede ham med eventuelle problemer og "feil" i programmet. Under kjøring av skriptene oppstod det en feilmelding på grunn av restriksjoner, men ved bruk av en spesifikk kommando ble dette raskt løst.

Brukeren fikk heller ikke kjørt *hentAssyst* og *loopAssyst* siden brukeren ikke hadde de riktige tilgangene i Assyst for å hente ut informasjon via rest-grensesnittet.

Tilbakemeldingene fra oppdragsgiveren og brukertesten ga gruppen innsikt i hvilken type "feil" som oppsto, hva som fungerte bra, og hva som bør prioriteres for videre arbeid (se vedlegg 5). Tilbakemeldingen gav gruppen en solid forståelse av hvordan programmet kan forbedres før prosjektets avslutning.

Evaluering av oppdragsgiver

I sluttevalueringen ble det sendt ut et spørreskjema til oppdragsgiverne for å få deres tilbakemeldinger (se vedlegg 6). Gruppen har også utarbeidet en skala fra 1-5, hvor 1 representerer lavest score og 5 representerer høyest score. Her følger en oppsummering av evalueringen basert på følgende punkter:

Hva synes du/dere er bra med vår løsning?
Oppdragsgiveren mente at det var en lovende start på noe som kan bli viktig i fremtiden, spesielt med tanke på bedriftens økende mengde programvare og den utfordringen med manglende dokumentasjon.

Hvordan vil du evaluere gruppens kommunikasjon og arbeidsflyt?
Oppdragsgiver var svært fornøyd med dette (4).

I hvor stor grad føler du at en løsning er implementert på en måte som gjør den enkel å utvide?
Oppdragsgiver syntes at implementeringen var akseptabel (3).

I hvilken grad føler du at dine tilbakemeldinger har bidratt til å oppnå et ønsket resultat i prosjektet?
Dette var oppdragsgiver veldig fornøyd med (5).

Hva bør gruppen prioritere for videre arbeid?
Det å samle all nødvendig informasjon på ett sted (skript).

Har du/dere eventuelle andre tilbakemeldinger for prosjektet?
Oppdragsgiveren er klar over at det var en omfattende oppgave, og de mener at gruppen har gjort en god jobb. Når man sammenligner hvor gruppen startet og hvor de er nå, er det tydelig at det har vært betydelig fremgang. Hadde gruppen hatt mer tid til rådighet, ville programmet blitt ytterligere forbedret, spesielt med tanke på brukervennlighet, vedlikehold og distribusjon av programmet (for eksempel via en webserver eller lignende).

5.3 Prosjektresultat

Prosjektets krav er delt inn i funksjonelle og ikke-funksjonelle krav (se vedlegg 2 visjonsdokument) som er beskrevet i detalj. Noen av kravene har blitt forandret etter at prosjektets avgrensninger ble fastsatt og det vil være ulikheter i rapporten i forhold til visjonsdokumentet. Begrunnelsen for dette er at i visjonsdokumentet hadde gruppen som mål å utvikle en automatisert løsning for å modellere integrasjonene i et arkitektprogram for visuell representasjon. Men underveis i utviklingen, ble det satt noen avgrensninger til dette hvor fokuset heller ble byttet til det å hente inn informasjon om relasjoner og applikasjoner for å så analysere denne informasjonen slik at den kan bli modellert på en strukturert måte.

Tabellen nedenfor viser implementerte funksjonelle og ikke-funksjonelle krav i prosjektet, samt de tilhørende delkravene som er nødvendige for å oppfylle dem. I tabellen er planlagte

og fullførte delkrav merket som *"Ja"* med grønn bakgrunnsfarge, mens ikke-planlagte og ikke-fullførte delkrav er markert som *"Nei"* med rød bakgrunnsfarge.

Tabell 5.1: Implementert funksjonelle krav.

Nr.	Funksjonelt krav	Delkrav			
		Nr.	Beskrivelse	Planlagt	Fullført
1	Produktet må kunne hente inn informasjon fra Assyst samt sammenstille informasjon fra flere informasjonskilder.	1.1	Assyst	Ja	Ja
		1.2	Biztalk, ved bruk av binding files	Ja	Ja
		1.3	Andre Systemer	Nei	Nei
2	Produktet må kunne modellere på en strukturell og forståelig måte.	2.1	Tekstlig informasjon	Ja	Ja
		2.2	Visuelt med figurer og farger	Nei	Nei
3	Produktet må kunne enkelt innhente ny informasjon når nye endringer i Helse Vest IKT sine systemer blir oppdatert - ved oppdatering av dagens systemer eller nye systemer.	3.1	Ny innhenting av oppdatert fil.	Ja	Ja
		3.2	En funksjon i programmet som ser etter nye oppdateringer fra Assyst.	Nei	Nei
4	Bruken må kunne velge en eller flere applikasjoner og modellere og vise relasjoner for disse.	4.1	I et scriptprogram	Ja	Ja
		4.2	I et brukergrensesnittprogram	Nei	Nei

Tabell 5.2: Implementert ikke-funksjonelle krav.

Nr.	Beskrivelse	Planlagt	Fullført
1	**Pålitelighet:** Produktet må modellere på en pålitelig måte som samsvarer med hvordan systemet faktisk er.	Ja	Ja
2	**Pålitelighet:** Produktet må også kunne meddele mulig misvisende innhold (f.eks duplikater eller utdaterte relasjoner), slik at det er mulig å kunne gjøre korreksjoner i datagrunnlaget eller i konsept/modell.	Ja	Ja
3	**Vedlikehold:** Produktet må være lett å bruke dersom nye endringer oppstår.	Ja	Ja
4	**Sensitivitet:** Da resultatet av piloten vil kunne inneholde sensitiv informasjon, vil rapporterte resultater bli anonymisert hvor det er hensiktsmessig.	Ja	Ja

5.4 Prosjektgjennomføring

Prosjektet ble gjennomført fra januar 2023 til mai 2023, og gruppen valgte å jobbe som et samlet team i stedet for å fordele oppgaver individuelt. Imidlertid opplevde gruppen utfordringer knyttet til tidsbruken på grunn av parallelle forpliktelser i andre fagområder. Dette resulterte i begrenset tid tilgjengelig i både forprosjektfasen og tidlig i hovedprosjektfasen. For å håndtere dette økte gruppen innsatsen mot slutten av hovedprosjektet for å kompensere for tap av tid i starten av hovedprosjektfasen.

Opprinnelig hadde gruppen en planlagt timebudsjettering på 344 timer per medlem og et totalbudsjett på 688 timer, inkludert refleksjonsnotater, presentasjon og EXPO-arrangementet. Imidlertid endte den faktiske tidsbruken for prosjektet på 702 timer, uten å ta hensyn til refleksjonsnotater, presentasjon og EXPO. Med en satt tidsramme på 50 timer for disse aktivitetene, ender gruppen opp med å overskride den estimerte tidsplanen med 64 timer og en total tidsbruk på 752 timer, under forutsetning av at alt går etter planen.

Til tross for utfordringene knyttet til tidsstyring, klarte gruppen i stor grad å følge gantt-diagrammet som ble presentert i prosjekthåndboken (se vedlegg 1). Prosjekthåndboken vil også inkludere detaljerte timelister og statusrapporter for å dokumentere prosjektets fremgang gjennom hele perioden.

6 DISKUSJON

I dette kapittelet vil det bli diskutert problemene som oppstod i prosjektet, hva som fungerte bra, hva som kan forbedres samt sluttresultatet.

6.1 Utfordringer i prosjektet

En av utfordringene gruppen møtte var restriksjonene i systemet. Dette resulterte i utfordringer knyttet til arbeidet på både PC og server. På grunn av ulike begrensninger, måtte gruppen benytte PC-en for å hente inn data fra Assyst, og serveren for å kjøre spesifikke skript i programmet. Dette førte til en ekstra arbeidsprosess der gruppen måtte overføre data fra PC-en til serveren, noe som krevde ekstra tid og arbeidsinnsats. I tillegg måtte gruppen få tilsendt binding-files fra oppdragsgiver ettersom de ikke hadde direkte tilgang til BizTalk Serveren. Disse restriksjonene la til ekstra kompleksitet i prosjektet og medførte behov for ekstra arbeid og kommunikasjon med oppdragsgiver for å få nødvendige filer og informasjon.

En annen utfordring var det reduserte antallet medlemmer i gruppen. Opprinnelig besto gruppen av tre medlemmer, men ett av medlemmene forlot gruppen tidlig i prosjektet. Dette resulterte i en betydelig økning i arbeidsbelastningen for de gjenværende to medlemmene. Siden oppgaven allerede var stor, krevende og større enn forventet, ble det ekstra utfordrende å oppnå det ønskede resultatet innenfor den begrensede tidsrammen.

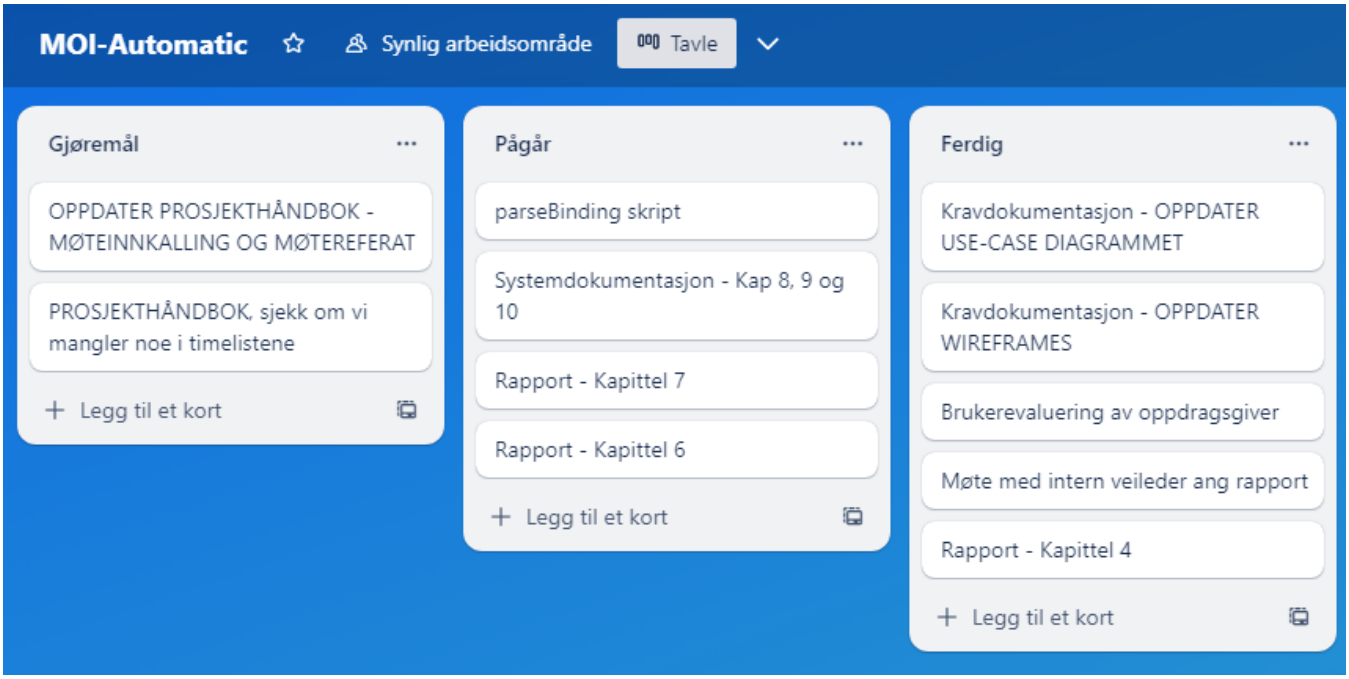
Manglende erfaring med skripting var en annen utfordring for gruppen. Skripting krever en viss grad av teknisk kompetanse og forståelse, noe som gruppen hadde begrenset erfaring med. Dette førte til utfordringer med utvikling og implementering av et fungerende skript.

For å takle prosjektets omfang og krav, dedikerte gruppen ekstra tid til å oppnå en dypere forståelse av oppgaven. Dette inkluderte å tilegne seg grundig kunnskap om systemene som ble brukt, for eksempel Assyst og BizTalk-serveren. Denne kunnskapen var nødvendig for å kunne håndtere og implementere prosjektet på en effektiv måte.

Etter at prosjektet ble avgrenset, oppsto det endringer i fokuset som medførte ytterligere utfordringer for gruppen. Dette skyldtes den allerede investerte tiden i planleggingen av brukergrensesnittet og forberedelsene til visualisering og implementering av modellen.

6.2 Suksessfaktorer

Gruppens valgte utviklingsmetodikk har spilt en avgjørende rolle for det endelige resultatet av prosjektet. Kanban Board har vist seg å være en svært effektiv metode for oppgavehåndtering og har gitt gruppen en enkel måte å organisere oppgavene mellom gruppemedlemmene (Anderson, 2016). Ved å ha en visuell oversikt over alle oppgavene som måtte fullføres og til hvilken tid, kunne gruppen jobbe mer strukturert og prioritere oppgavene på en mer effektiv måte. En annen fordel med Kanban Board var at det hjalp gruppen med å unngå overbelastning av arbeid ved å enkelt identifisere oppgaver som var på vent og hvilke som hadde høyere prioritet.
Kanban Boardet bidro også til å øke kommunikasjonen og samarbeidet mellom gruppemedlemmene. Ved å ha en visuell oversikt over hva som ble gjort, hva som var på vent og hvilke problemer som måtte løses, kunne gruppen raskt finne ut hvor de kunne hjelpe hverandre og samarbeide mer effektivt. Dette bidro til å øke effektiviteten i prosjektet og forbedre teamets samarbeid.



Figur 6.1: Et utdrag av Kanban Boardet brukt i Trello

Gjennom muligheten til å arbeide på kontorplassen til Helse Vest IKT, oppnådde gruppen en betydelig fordel med tanke på å oppnå en dypere forståelse av oppgaven. Denne nærheten til arbeidsmiljøet ga gruppen en bedre innsikt i konseptene og kompleksiteten knyttet til relasjoner, integrasjoner og systemfunksjonalitet.

En annen viktig suksessfaktor i prosjektet var de regelmessige møtene med oppdragsgiver. Ved å ha fysiske møter, muliggjorde direkte og effektiv kommunikasjon med

oppdragsgiveren. Gjennom disse møtene kunne gruppen få presiseringer, avklaringer og ytterligere veiledning for å sikre en felles forståelse av prosjektets krav og mål. Denne tette samarbeidsformen bidro til å redusere misforståelser, forbedre prosjektflyten og sikre at prosjektet beveget seg i riktig retning.

6.3 Sluttprodukt

Sluttproduktet er begrenset til det interne miljøet på Helse Vest IKT, og tilgangen til programmets verktøy og funksjonaliteter er forbeholdt de ansatte i organisasjonen. Spesielt vil integrasjonsavdelingen dra nytte av programmet, da det gir dem muligheten til raskt og effektivt å samle inn og organisere informasjon om ulike applikasjoner og relasjoner. Dette vil forbedre forståelsen av integrasjonene i systemene og bidra til å effektivisere arbeidsprosessene deres. I tillegg kan også andre avdelinger ved Helse Vest IKT dra nytte av programmet, da verktøyene som er utviklet for feilsøking og informasjonsinnhenting kan hjelpe de ansatte med å løse problemer i deres daglige arbeid. Dette vil bidra til å redusere nedetid og øke produktiviteten.

Gjennom tilbakemeldinger fra oppdragsgiveren har gruppen fått bekreftet at de har oppnådd de viktigste funksjonalitetene, og dette markerer en lovende start på noe som kan få betydelig betydning i fremtiden. Oppdragsgiveren uttrykker sin tilfredshet med programmet og hvordan det presenteres relevant informasjon om applikasjonene. De bemerker at programmet gir en klar og oversiktlig visning av relasjonene og avhengighetene mellom systemene, noe som bidrar til bedre forståelse. Basert på tilbakemeldingene ser gruppen et potensial for videreutvikling og forbedring av programmet for å møte fremtidige behov og utfordringer.

Oppdragsgiver påpekte viktigheten av å integrere skriptene i en helhetlig løsning for å hente informasjon fra flere systemer hos Helse Vest IKT. Gruppen har så langt ikke lykkes med å oppnå dette, men en slik løsning ville ha resultert i at all relevant informasjon ville bli presentert på et sted, i motsetning til å være spredt mellom ulike skript som *parseAssyst* og *parseBinding*. Selv om dette ikke ble implementert, ble oppdragsgiveren likevel fornøyd med det som ble oppnådd.

Han ga også uttrykk for ønsket om et dedikert brukergrensesnitt for produktet, da dette ikke var tilgjengelig i den nåværende versjonen. Mangelen på et dette, fører til at brukerne må håndtere skriptfiler direkte, noe som kan være utfordrende for de som ikke er kjent med skripting, og det kan kreve ekstra innsats for å bruke programmet effektivt.

6.4 Forbedringer

Hvis prosjektet skulle startes på nytt, ville gruppen ha gjort noen forbedringer basert på erfaringene fra det nåværende prosjektet. Først og fremst ville gruppen ha etablert en bedre kommunikasjon med oppdragsgiver tidlig i prosessen for å tydelig definere og avklare forventningene til prosjektets resultat. På denne måten kunne gruppen allerede tidlig i prosjektet ha fokusert mer på analyseringen av informasjonen og hvordan de kunne koble informasjon fra forskjellige systemer sammen. Gruppen ville da også ha hatt mer tid til utviklingen av systemet, og potensielt fått til et bedre sluttresultat når det kom til selve produktet.

Gruppen ville også ha prioritert faget mer tidlig i prosjektet og investert mer tid i arbeidet. Dette ville ha gitt mer rom for grundig utforskning og utførelse av prosjektet. En annen forbedring ville vært å visualisere oppgaven mer ved å lage skisser og figurer i samarbeid med oppdragsgiveren. Dette ville ha bidratt til en bedre felles forståelse og tydeligere definisjon av prosjektets omfang og mål.

I tillegg innser gruppen at hvis prosjektet skulle ha startet på nytt, så ville en databaseløsning vært en foretrukket tilnærming. Selv om Powershell-skriptet som ble brukt i dette prosjektet var funksjonelt og ga ønsket resultat, ville det ha vært bedre og mer effektivt å bruke en database løsning. Ved å benytte en database kunne man håndtere store mengder data på en mer strukturert og organisert måte. Dette ville ha ført til enklere håndtering av data, raskere dataanalyse og en mer effektiv behandling av dataene. Videre ville en database løsning tillate en mer fleksibel og skalerbar tilnærming. Ved å bruke en database kan man enkelt legge til eller fjerne data, samt gjøre endringer i databasestrukturen, uten å måtte endre kode eller skript. Som resultat av dette, vil det bidra til å spare tid og ressurser ved eventuelle fremtidige videreutviklinger eller oppgraderinger.

7 KONKLUSJON OG VIDERE ARBEID

Dette kapittelet inneholder en oppsummering av prosjektet, og gir en oversikt over muligheter for videre arbeid.

7.1 Konklusjon

I dette bachelorprosjektet var målet å utvikle et program for automatisert modellering av applikasjoner og relasjoner i systemene til Helse Vest IKT, med potensial for videreutvikling av bedriften. Forskningsspørsmålene i prosjektet var rettet mot å identifisere relasjoner i integrasjonssystemet og utvikle en pilot som kunne visualisere sammenhengen mellom applikasjoner og relasjoner.

Resultatet av prosjektet er et skriptprogram som gir en helhetlig oversikt over applikasjoner i Helse Vest IKT-systemene ved å samle og presentere informasjon om avhengigheter og relasjoner. Tilbakemelding fra oppdragsgiver indikerer at programmet henter relevant informasjon om applikasjonene og identifiserer deres relasjoner. Ved å ha denne informasjonen tilgjengelig på ett sted, blir det enklere for brukerne å identifisere sammenhenger og forstå integrasjonen mellom systemene. Dette legger grunnlaget for mer effektiv problemløsning. Videre gir programmet også rom for videreutvikling og tilpasning etter behovene til Helse Vest IKT, slik at det kan fortsette å støtte og forbedre deres integrasjonsarbeid.

7.2 Videre arbeid

I videre arbeid vil det være en sentral oppgave å utvikle en mer brukervennlig frontend- og backend-løsning som gir brukerne et enklere og mer effektivt grensesnitt. Dette vil inkludere søkefunksjonaliteter som letter prosessen med å finne ønskede applikasjoner. Ved å utvikle et mer brukervennlig grensesnitt vil man redusere behovet for brukerne å lære seg skripting eller å håndtere komplekse tekniske detaljer. Programmet bør tilrettelegge for en enkel og intuitiv interaksjon, slik at brukerne kan fokusere på å utføre oppgavene sine effektivt uten å bli hindret av tekniske barrierer.

Videre arbeid inkluderer også implementering av en funksjonalitet for "eksporter til andre formater", som har som mål å muliggjøre eksport av modellen med informasjonen til et annet program. Dette åpner opp muligheten for visuell modellering av informasjonen i en annen kontekst. Ved å legge til denne funksjonaliteten vil brukerne kunne utvide bruksområdet for modellen og samhandle med dem på en mer visuell og intuitiv måte i et egnet program. Dette vil gi en mer fleksibel og tilpasningsdyktig tilnærming til datahåndtering og visualisering, og kan bidra til å støtte ulike behov og preferanser blant

brukerne.

For å kunne videreutvikle prosjektet og øke nøyaktigheten i relasjonsmodelleringen, vil det være nødvendig å samle inn mer detaljert informasjon om applikasjonene. Dette kan gjøres ved å utføre en grundigere analyse av systemene og deres funksjonaliteter. Det vil også være viktig å hente informasjon fra andre systemer som kan gi ytterligere innsikt i relasjonene mellom applikasjonene.

Per nå er informasjonen lagret i XML-format og behandles av et Powershell-skript. Ved å overføre dataene til en database kan man effektivt analysere informasjonen ved hjelp av SQL-spørringer. På denne måten vil man kunne hente ut og visualisere relasjonene mellom applikasjonene på en mer strukturert og oversiktlig måte.

I videre arbeid vil det være en prioritet å utvikle en mer effektiv løsning som fjerner behovet for å sende zipfiler og forenkler tilgangen til programmet for ansatte i Helse Vest IKT. Målet er å utvikle et frittstående program som gir brukerne en enkel måte å få tilgang til *MOI-Automatic* på. Dette programmet vil eliminere behovet for manuell distribusjon av skriptfiler og sikre at brukerne enkelt kan få tilgang til programmet.

8 Referanseliste

Microsoft (2021) *Binding Files and Application Deployment.* Tilgjengelig fra https://learn.microsoft.com/en-us/biztalk/core/binding-files-and-application-deployment (Hentet 22.02.2023)

Machiraju, S. (2018) *BizTalk: Azure applications.* Tilgjengelig fra: https://ebookcentral.proquest.com/lib/hogskbergen-ebooks/detail.action?pq-origsite=primo&docID=5156290 (Hentet 22.02.2023)

Helse Vest IKT (2022) *Om oss.* Tilgjengelig fra https://helse-vest-ikt.no/om-oss#avdelingar (Hentet 09.02.2023)

MDN (u.å) *Working with JSON.* Tilgjengelig fra: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON (Hentet 28.02.2023)

UMLet (u.å) *Umlentino* https://www.umletino.com/ (Hentet: 20.02.2023)

Infoklikk (2016) *Driv din forretning, ellers vil den drive deg. Benjamin Franklin.* Tilgjengelig fra https://www.infoklikk.no/tabeller-artikler/bedrift/maler-analyse/115-ros-analyse-inkl-risikomatrise (Hentet 27.01.2023)

Gantt.com (u.å) *Hva er et Gantt-diagram?* Tilgjengelig fra: https://www.gantt.com/no/ (Hentet: 10.02.2023)

Figma (u.å) *What is Figma?* Tilgjengelig fra: https://help.figma.com/hc/en-us/articles/14563969806359-What-is-Figma- (Hentet 15.02.2023)

Gillish, A. (2021) *What is Excel?* Tilgjengelig fra: https://www.techtarget.com/searchenterprisedesktop/definition/Excel (Hentet 03.02.2023)

Ho, D. *(u.å) What is notepad++* https://notepad-plus-plus.org/ (Hentet 11.02.2023)

Montgomery, J. (2020) *Configuration Management database.* Tilgjengelig fra https://www.techtarget.com/searchdatacenter/definition/configuration-management-database (Hentet 08.02.2023)

Visual Studio Code (2023) *Learn to code with Visual Studio Code*. Tilgjengelig fra: https://code.visualstudio.com/learn (hentet 08.03.2023)

Microsoft (2023) *XML for nybegynnere*. Tilgjengelig fra: https://support.microsoft.com/nb-no/office/xml-for-nybegynnere-a87d234d-4c2e-4409-9cbc-45e4eb857d44 (hentet 17.03.2023)

W3schools (Uten år) *Introduction to XML.* Tilgjengelig fra: https://www.w3schools.com/xml/xml_whatis.asp (Hentet 08.03.2023)

Zmaranda D., Cornelia G., Robert G., & Anca-Raluca B. (2018) *Comparative study of data sending methods for XML and JSON models.* Tilgjengelig fra: https://doc.presentica.com/11395747/5ebad4ca96297.pdf (Hentet 28.02.2023)

Anderson, D (2016) *Patterns of Kanban Maturity (part 2)* Tilgjengelig fra: https://resources.kanban.university/patterns-of-kanban-maturity/ (Hentet 16.03.2023)

Microsoft (2022) *What is PowerShell* Tilgjengelig fra: https://learn.microsoft.com/en-us/powershell/scripting/overview?view=powershell-7.3 (Hentet 01.03.2023)

Beauvoir, P. og Sarrodie, J. (2023) *Archi.* Tilgjengelig fra: https://www.archimatetool.com/ (Hentet 05.02.2023)

Microsoft (2023) *Measure-Command.* Tilgjengelig fra: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/measure-command?view=powershell-7.3 (Hentet 01.05.2023)

Lee Holmes (2021) *PowerShell Cookbook.*

Malik, A., Burney, A. og Ahmed, F. (2020) *A Comparative Study of Unstructured Data with SQL and NO-SQL Database Management Systems*

https://www.scirp.org/journal/paperinformation.aspx?paperid=99539 (Hentet 01.03.2023)

9 Vedlegg

Vedlegg 1 Prosjekthåndbok

Vedlegg 2 Visjonsdokument

Vedlegg 3 Kravdokumentasjon

Vedlegg 4 Systemdokumentasjon

Vedlegg 5 Brukertest

Vedlegg 6 Sluttevaluering

Vedlegg 7 Zip-mappe av skript (MOI-AUTOMATIC)