Bacheloroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

Kimia Abtahi, Camilla Haaheim Larsen
og Nora Othilie Ringdal

Utvikling og implementering av RPA-roboter for Skatteetaten

Bacheloroppgave i Dataingeniør
Veileder: Majid Rouhani
Mai 2019

NTNU
Kunnskap for en bedre verden

Skatteetaten

Kimia Abtahi, Camilla Haaheim Larsen
og Nora Othilie Ringdal

Utvikling og implementering av RPA-roboter for Skatteetaten

Bacheloroppgave i Dataingeniør
Veileder: Majid Rouhani
Mai 2019

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

NTNU
Kunnskap for en bedre verden

Forord

Denne bacheloroppgaven er skrevet som avsluttende oppgave i forbindelse med studieretningen dataingeniør ved Institutt for datateknologi og informatikk ved Norges teknisk - naturvitenskapelige universitet. Oppgaven utføres hos Skatteetaten i perioden januar til mai 2019. RPA-robotene er utviklet med Blue Prism, av tre studenter uten tidligere erfaring med RPA-teknologi.

Vi vil takke Skatteetaten og Hanne Martea Solum for muligheten til å gjennomføre prosjektet. Spesielt takk til Kai Arne Andreassen og Ingrid Annette Haave for å ha delt sin ekspertise og vært tilgjengelig under utviklingen. Videre vil vi takke Skatteetatens RPA-team som har tatt seg tid til å svare på tekniske spørsmål og hjulpet med prøveproduksjon. Vi vil også rette en stor takk til vår veileder, Majid Rouhani, for å ha stilt seg tilgjengelig med gode råd og veiledning underveis.

Heretter vil vi refereres til som teamet, utviklerne og studentene. AT Økonomi, representert ved Kai Arne og Ingrid, vil refereres til som kunde.

Trondheim, 20.mai 2019

Kimia Abtahi Camilla Haaheim Larsen Nora Othilie Ringdal

Oppgavetekst

Opprinnelig oppgavetekst var *Utvikling av Remedy-robot for Skatteetaten*. Denne ble erstattet med *Utvikling og implementering av RPA-roboter for Skatteetaten*, ettersom RPA er teknologien som brukes for å utvikle roboten og Remedy er et av programmene den samhandler med.

Roboten som jobber i programmet Remedy er gitt som primæroppgave. Det ble også definert en sekundæroppgave, dersom primæroppgaven ble løst på kort tid. Sekundæroppgaven tar for seg arkivering av bilag fra e-post. Det var ikke et krav å ferdigstille denne oppgaven.

Herav vil primæroppgaven refereres til som Remedy-robot og sekundæroppgaven som arkiv-robot.

Sammendrag

I en tid med hurtig teknologisk vekst og stor konkurranse, har effektivisering av administrative tjenester blitt svært viktig for å spare tid og kostnader for bedrifter og offentlige etater. Interessen for Robotisert prosessautomatisering (RPA) har økt kraftig over de siste årene som en mulig løsning for disse utfordringene. Tidligere prosjekter med RPA-teknologi viser til gode resultater innen nøyaktighet, løsningstid og fleksibilitet, men også utfordringer med tanke på tid og kostader knyttet til selve implementasjonen. I denne oppgaven undersøkes det om det er hensiktsmessig å implementere en RPA-robot for å utføre manuelle oppgaver, og hvordan en slik robot kan implementeres i et eksisterende system.

Det er utviklet to RPA-roboter som skal brukes av avdelingen for Administrative Tjenester, seksjon Økonomi i Skatteetaten. Utviklingen av robotene er gjort med Blue Prism. Den første roboten skal fordele saker til ansatte i avdelingen i programmet Remedy, og er ved prosjektets slutt godkjent for produksjon. Den andre roboten skal arkivere bilag som blir sendt som vedlegg til avdelingen via e-post, og er godkjent gjennom kvalitetstester i samråd med kunden, men er ikke produksjonssatt ved prosjektets slutt. Begge robotene bruker Excel-ark, som er utformet slik at verdiene enkelt kan skiftes ut, slik at roboten skal kunne skaleres og utføre lignende oppgaver for andre avdelinger.

Resultatetene viser at det er hensiktsmessig å implementere RPA-robot for bedre nøyaktighet, løsningstid og høyere kvalitet, men at det avhenger av hvor standardisert oppgaven er, samt volum av oppgaver. Antallet oppgaver var for få til at robotene i dette prosjektet ble lønnsomme. Kontinuerlig og tett samarbeid med kunden under utviklingsprosessen var avgjørende for utvikling av robotene.

Innhold

Figurer

Akronymer

DFØ Direktoratet for økonomistyring. 27, 101

HTML HyperText Markup Language. 6, 20

IKT Informasjons - og kommunikasjonsteknologi. 2, 4

NTNU Norges teknisk-naturvitenskapelige universitet. 33

QA Quality Assurance. 13, 21

RPA Robotisk prosessautomasjon (engelsk: Robotic Process Automation). 5

SSO Single Sign-on. 25, 45, 60, 72, 73, 98

VBO Virtual Business Object. 9, 21, 25, 77

Kapittel 1

Introduksjon og relevans

1.1 Bakgrunn

Med den raske veksten av ny teknologi de siste tiårene har digitale virkemidler ikke bare blitt en del av folks hverdag, men også en viktig del av samfunnets infrastruktur. Digitalisering er en naturlig og integrert del av samfunnsutviklingen [1]. Selv om digitalisering og automatisering er store utfordringer for det offentlige, legger det til rette for økt verdiskapning og innovasjon [2]. Den teknologiske utviklingen vil trolig føre til at maskiner i økende grad overtar mer avanserte arbeidsoppgaver og ikke bare "mekaniske" rutineoppgaver [2].
Effektivisering av blant annet administrative tjenester har blitt viktig for å spare tid og kostnader. Dette gjelder offentlig sektor i like stor grad som privat sektor. Kommunal- og moderniseringsdepartementet har for den norske regjering utarbeidet en strategi med navn *Helhet, kvalitet og effektivitet* for å utvikle og effektivisere de administrative fellestjenestene i departementsfellesskapet [3, s.3]. Strategien har som mål at kostnadene ved de administrative tjenestene i regjeringskvartalet skal reduseres gjennom å ta i bruk flere fellestjenester som organiseres og leveres kostnadseffektivt, til rett tid og med riktig kvalitet [3, s.7].

Skatteetaten er en offentlig etat som er underlagt Finansdepartementet. Etaten har ansvar for å holde det norske folkeregisteret oppdatert, samt sørge for at innbetalingen av skatter og avgifter foregår på riktig måte [4]. Å sikre finansieringen av velferdsamfunnet er en stor oppgave som krever ressurser [4]. Med målrettet bruk av midler og effektivisering av prosesser, kan det norske samfunnet kutte utgifter og forvalte dem i andre sektorer. Her vil digitalisering og automatisering spille en stor rolle, og etaten ser allerede resultatet av slik satsning. Fra

og med 2014 sendte Skatteetaten skattemeldingen digitalt til alle som ikke aktivt reserverte seg mot det [5]. Dette resulterte i store kutt i kostnadene som tidligere gikk til å sende alle skattemeldingene i posten. Jan Tore Sanner, som på den tiden var kommunal - og moderniseringsminister, sa at at overgangen til nettbasert kommunikasjon kan spare staten for én milliard kroner, bare i portoutgifter [5].

I deres videre satsning på utvikling av IKT, har Skatteetaten tatt i bruk Robotisk prosessautomasjon, herav forkortet til RPA, for automatisering av manuelle oppgaver. Årsaken til denne satsningen er et ønske om å redusere tid og kapasitet brukt på enkle, repeterende rutineoppgaver. Dersom en robot kan utføre slike oppgaver, vil kapasitet frigjøres slik at mer krevende oppgaver kan prioriteres.

Skatteetatens seksjon Økonomi under avdeling for Administrative Tjenester, herav forkortet til AT, ønsker å implementere RPA-roboter i sitt arbeid. AT Økonomis viktigste oppgave er å forvalte og utvikle landsdekkende administrative tjenester i henhold til etatens behov innenfor økonomi og regnskap [6]. Seksjonen skal sørge for at etatens regnskap og budsjett er oppdatert i henhold til statens krav [6]. AT Økonomi har flere oppgaver som egner seg for automatisering, deriblant fordeling av innkommende saker til ansatte i seksjonen via datasystemet Remedy, samt arkivering av bilag fra e-post. Skatteetaten ønsker å bruke studenter til å utvikle disse to robotene for å kartlegge tidsforbruk av opplæring og utvikling, samt kunnskap og kvaliteter som egner seg ved RPA-utvikling i Skatteetaten. Dette behovet reflekteres i effektmålene utarbeidet av utviklerne, som sier at det skal utvikles en robot som øker AT Økonomis effektivitet ved å automatisere manuelle oppgaver. Studentene skal lære å programmere i Blue Prism og øke forståelsen av prosesser i programmering.

1.2 Problemstilling

Teamet har fra starten av prosjektet ønsket å undersøke fordeler og ulemper med implementering av RPA-robot. Ettersom ingen av teammedlemmene har erfaring med RPA-utvikling var de også nysgjerrige på hvordan utvikling- og produksjonsprosessen så ut. Å formulere problemstillingen har vært utfordrende, og det ble formulert både for spesifikke og for vage problemstillinger underveis.

Problemstillingen teamet valgte å undersøke er: *Er det hensiktsmessig å implementere RPA-robot for å utføre manuelle oppgaver, og hvordan kan den implementeres i et allerede eksisterende system?* For å kunne vurdere om det er hensiktsmessig vil vi sammenligne hvordan oppgavene løses manuelt med hvordan roboten utfører oppgaven. Spesifikt vurderes hastighet, automasjonsgrad, samhandling med eksisterende systemer, utførelse av oppgaven og om det gir økonomisk gevinst.

1.3 Oppgavens struktur

Oppgaven starter med et teorikapittel som beskriver det som ligger til grunn for utviklingen av robotene. Neste kapittel forklarer valg av metoder og teknologiene beskrevet i teorikapittelet. De neste kapittelene beskriver resultatet av arbeidet og i diskusjonen drøftes resultatene som er presentert. Avslutningsvis vil konklusjonen svare på problemstillingen og hva som er veien videre. Det vil refereres til vedlagte dokumenter underveis i oppgaven.

Kapittel 2

Teori

2.1 Effektivisering av administrative tjenester

Som tidligere nevnt i kapittel 1.1, er administrative tjenester fra konkurranseutsatte sektorer under stadig press for å redusere tid og kostnader, uten at det skal gå på bekostning av kvalitet, service og sikkerhet [7, s.3]. I takt med den digitale utviklingen, har digitale verktøy og tjenester blitt tatt i bruk for å støtte opp eller erstatte manuelle oppgaver. Ofte blir de digitale verktøyene utviklet for et enkelt problem, dermed må nye IKT-systemer fungere sammen med eldre, eksisterende programmer som er basert på ulik teknologi. Dette har ført til et komplekst lappeteppe av digitale verktøy i bedrifter og etater som ikke kommuniserer med hverandre [8], der samme informasjon ofte blir lagret på flere steder [9, s.270-71]. I mange bedrifter må derfor ansatte bruke tid på å samhandle med med ulike digitale verktøy, ved å logge seg inn og hente informasjon fra et program for så å logge seg inn i et annet program som behandler informasjonen [9, s.270] [7, s.5]. Slik enkel og rutinepreget interaksjon kan forbedres og effektiviseres på flere måter, blant annet gjennom felles løsninger og standardisering, endring av rutiner og automatisering [3, s.7].

I store bedrifter finnes det tjenester som er felles mellom avdelinger og oppgaver som løses på lik måte. Standardisering av disse felles løsningene vil bidra til bedre bruk av ressursene og effektivisering av driften [3, s.8]. De administrative tjenestene skal fokusere på å benytte prosedyrer og rutiner basert på beste-praksis prinsippet. Ved endring av rutiner vil det første steget være å analysere prosessen og identifisere hvilke steg som hindrer effektivitet. Deretter er neste steg å endre prosessen med mål om å oppnå høyere effektivitet og kvalitet. For at

dette skal ha ønsket effekt, er det essensielt at disse rutinene blir fulgt. Med tanke på at det ofte er mange ansatte ved forskjellige avdelinger som skal utføre like oppgaver, vil kvaliteten på arbeidet avhenge av i hvilken grad de følger rutinene som er satt.
Løsningene beskrevet over går ut på å forbedre hvordan mennesker løser arbeidsoppgavene. I neste delkapittel skal vi se nærmere på en mulig tredje løsning for å effektivisere oppgaver; automatisering.

2.2 Automatisering og RPA

Automatisering er teknikken å få systemer til å fungere uten, eller med liten grad, av menneskelig medvirkning [10]. Dette har tidligere vært aktuelt i sammenheng med mekanisering i industriell produksjon der maskiner har erstattet fysiske arbeidsoppgaver [10]. Automatisering av administrative tjenester med digitale verktøy er derimot av relativ ny dato [10], og vokser i takt med utviklingen av ny teknologi, deriblant RPA.

2.2.1 Hva er RPA?

RPA-begrepet omfatter bruken av programvarerobot for å utføre oppgaver og opererer på brukergrensesnittet til datasystemer på samme måte som et menneske [7, s.6] [9, s.270] [11]. Teknologien ble utviklet med fokus på å erstatte mennesker i utførelse av repeterende rutineoppgaver i bedrifter.

RPA anses som lettvektsteknologi i den forstand at det er kommersiell teknologi der selskap, som utvikler RPA-programvare, selger lisenser til bedrifter, som selv konfigurerer programvaren til å utføre oppgaver [7, s.6]. Selve konfigureringen er designet for å være enkel og fri fra programmering med kode, slik at dette kan utføres av personer som ikke er programvareutviklere eller IT-spesialister [7, s.7]. RPA er også lettvekt med tanke på at det ikke forstyrrer underliggende datasystemer. RPA-roboter aksesserer andre programmer på samme måte som et menneske benytter brukergrensesnittet, via presentasjonslaget. Programmene åpnes lokalt, og roboten logger seg inn ved å skrive inn brukernavn og passord. Dermed er eksisterende sikkerhetsmekanismer ivaretatt. RPA-teknologi implementeres på toppen av det eksisterende applikasjonslaget i en bedrift, uten å endre infrastruktur og andre systemer [12]. Dermed kan en RPA-robot fungere som sytråden som holder lappeteppet av digitale verktøy sammen.

2.2.2 Utviklingen av RPA

Det er først de siste årene RPA-utvikling virkelig har skutt fart og blitt tatt i bruk som et digitalt verktøy i bedrifter. Selve betegnelsen RPA stammer fra tidlig 2000-tallet, men teknologien som utnyttes har vært i utvikling siden rundt 1990-tallet [13]. Det kan være nyttig å forstå hvordan forgjengerne til RPA fungerer for å få en forståelse av RPA-teknologiens utvikling. UiPath, som er et ledende RPA-selskap, trekker frem blant annet skjermskraping og automatisert arbeidsflyt som to av hovedforgjengerne til RPA [13]. Gjenbruk av kode for andre prosesser gir effektiv utvikling av automatiserte prosesser.

Skjermskraping

Skjermskraping defineres som en prosess for å hente ut og kombinere ønsket innhold fra Internett på en systematisk måte [14]. Slik programvare etterligner de typiske interaksjonene mellom webtjenere og mennesker som bruker nettlesere, og henter ut de dataene som ønskes og analyserer disse [14]. Dette utføres som regel via en bot som omformer ustrukturert data, typisk i HTML-format, til strukturert data som kan lagres i databaser eller regneark [15]. Skjermskraping blir i dag blant annet brukt på Internett på prissammenlignings nettsider, nettforsikring, monitoring av værdata og for å se om en nettside har blitt endret.

Skjermskraping forstår bare hvordan leser data fra et felt i et fast sted og flytter dataene til et felt i et annet fast sted på skjermen. Hvis feltet beveger seg, stopper verktøy. RPA er mer enn skjermskrapingsverktøy. Skjermskraping er som individuelle makroer, skrevet av enkeltpersoner. RPA er forskjellig som kan kobles til presentasjonslaget i HTML, på klientserveren, Java Access Bridge, mainframe-applikasjoner og ulike API ved hjelp av innebygd applikasjonsstøtte. Når prosessene er designet, kan de kjøres på en sentralisert server og samtidige av flere roboter, som gir revisjonsstatistikk i sanntid. RPA-verktøy har eget dashbord for å vise når roboten fungerer og akkurat hva den gjør.

Automatisert arbeidsflyt

Selv om opprinnelsen av begrepet stammer fra 1920-tallet ved fremveksten av industriell produksjon, har bruken av automatisert arbeidsflyt økt i omfang fra 1990-tallet i sammenheng med programvare [13]. Arbeidsflyt er en sekvens av trinn som er involvert fra start til slutt i en arbeidsprosess [16]. Formålet med slik programvare er å automatisere deler av en prosess,

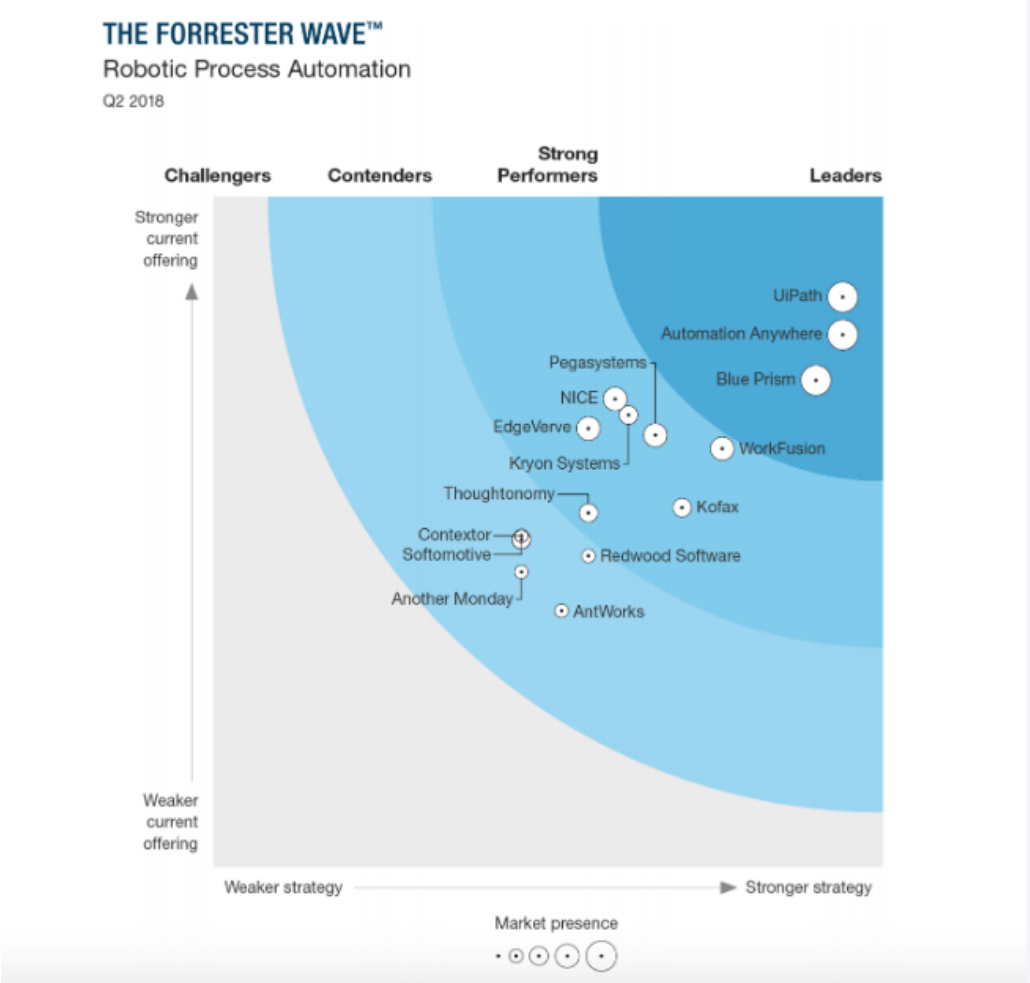
som oftest i en bedrift. Disse prosessene er gjerne rutineoppgaver som må utføres, men som ikke er særlig komplekse eller givende for den ansatte. Et eksempel på en slik oppgave er å besvare nytt medlemskap i en bedrift med en velkomstmail. Siden bedrifter har retningslinjer for når og hvordan oppgaver skal løses i en arbeidsflyt, kan slike delprosesser automatiseres. Slik automatisert programvare kan bygges ved hjelp av kode eller ved grafiske applikasjoner.

Fremveksten av RPA

RPA er avhengig av både skjermskraping og automatisert arbeidsflyt, men har også flere fordeler enn disse teknologiene separat. Skjermskraping forstår bare hvordan man kan lese data fra et felt i et fast sted og flytter dataene til et felt i et annet fast sted på skjermen. Hvis feltet beveger seg, stopper verktøyet opp. RPA er mer enn et skjermskrapingsverktøy. Skjermskraping er som individuelle makroer, skrevet av enkeltpersoner. RPA derimot kan kobles til presentasjonslaget i HTML, på klientserveren, mainframe-applikasjoner og ulike API ved hjelp av innebygd applikasjonsstøtte. RPA har også en fordel i forhold til automatisert arbeidsflyt, ettersom den kan automatisere en hel prosess fra start til slutt, mens automatisert arbeidsflyt som regel gjøres på en mindre del av en prosess.
Det er viktig å huske på at selv om RPA kan automatisere strømlinjeformede, gjentakende og regelbaserte prosesser, er ikke programvaren i stand til å håndtere unntak eller ta avgjørelser som den ikke er programmert til å gjøre. Fremtiden til RPA vil derfor være at det kombineres med kunstig intelligens som for eksempel maskinlæring for å håndtere mer komplekse oppgaver og ta beslutninger selv [9, s.271] [13]. Dermed vil RPA-verktøyet kunne klare å lære av sine feil og tilpasse seg for å håndtere unntaksoppgaver [9, s.271].

2.3 Teknologier for utvikling av RPA

Det fins mange ulike teknologier og verktøy som kan brukes for å utvikle RPA-roboter. Ifølge The Forrester Wave, er Blue Prism, UiPath og Automation Anywhere anerkjent som ledende teknologier innen RPA-utvikling (se figur 2.1 under) [17]. Det var også disse tre RPA-aktørene Skatteetaten vurderte for automatisering av sine prosesser, der valget falt på Blue Prism. Alle tre teknologiene tilbyr dra-og-slipp utvikling, der man bruker bokser med ulik funksjonalitet. Resultatet blir en kode som ligner flytdiagram kjent fra programvareutvikling (se figur 2.3 på side 10). Videre i denne seksjonen vil alle tre teknologiene bli beskrevet i korte trekk og sammenlignet, med hovedfokus på Blue Prism.

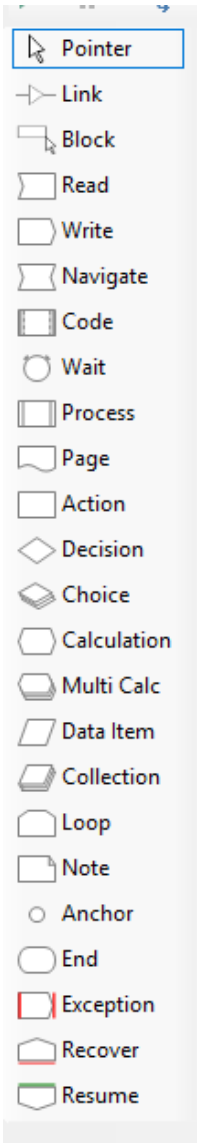


Figur 2.1: Figur over ledende aktører innen RPA, fra The Forrester Wave

Kilde: [17]

Blue Prism

Blue Prism Group er et britisk selskap som ble startet opp i 2001 [18]. Blue Prism Group selger lisenser og opplæringsmateriell for deres programvare til bedrifter, som får bruke programvaren slik de selv ønsker. Siden det er nødvendig med lisens for å bruke Blue Prism, er det ikke mulig å trene seg opp i å bruke programmet på egenhånd uten lisens. Verktøyet er bygget på det veletablerte Microsoft .NET-rammeverket.

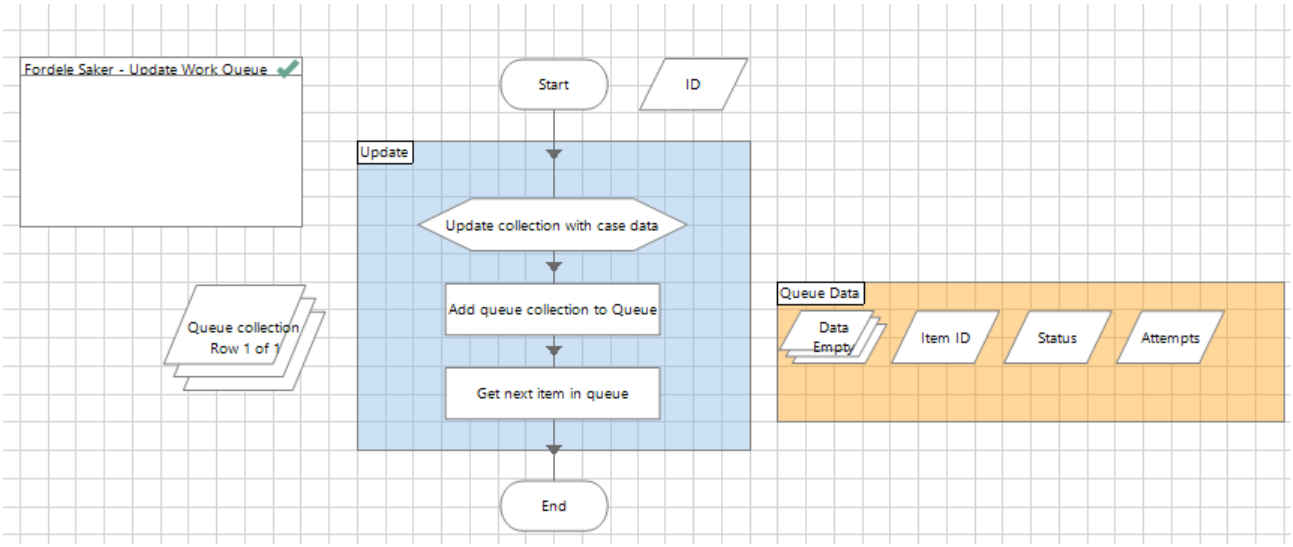


Figur 2.2: De ulike boksene i Blue Prism

Som nevnt tidligere bruker teknologien dra-og-slipp bokser som utviklingsverktøy, og det skal ikke mye programmeringserfaring til for å kjenne igjen boksene. De ulike boksene som er tilgjengelig i Blue Prism er vist i figuren til venstre. En "Decision"-boks tilsvarer en typisk if-setning, der man skriver inn uttrykk som enten er sant eller usant. Likeså er "Data Item" det samme som en variabel. Blue Prism har også unntakshåndtering, med "Exception", "Recover" og "Resume" som fungerer som en typisk "Try-Catch" fra Java-programmering.

Blue Prism bruker også prinsipper fra objektorientering, på den måten at det er et skille mellom "Processes" for selve prosessflyten, og "Virtual Business Object" (VBO) for samhandling med eksterne programmer. VBOene utvikles i "Business Object Studio" som bruker Blue Prisms spioneringsverktøy for å gjenkjenne skjermbilder fra ulike program og hente ut informasjon. VBO fungerer altså som et mellomlag mellom prosessen, som inneholder all logikk for roboten, og programmet roboten samhandler med. For å kjøre og bruke et VBO, må det kalles på i en prosess via en "Action"-boks slik man i tradisjonell kode kaller på en klasse eller metode. Denne oppdelingen legger grunnlaget for at prosesser og metoder kan gjenbrukes flere ganger.

Når det gjelder sikkerhet, scoret Blue Prism høyest av alle RPA-teknologiene ifølge The Forrester Wave [17]. UiPath og Automation Anywhere følger med delt andreplass innen sikkerhet.



Figur 2.3: Eksempel på kode i Blue Prism

Automation Anywhere

Automation Anywhere er et amerikansk RPA-selskap grunnlagt i 2003 [19]. I likhet med Blue Prism, benytter Automation Anywhere seg av lisenser, der prisen vil avhenge av oppgavens størrelse. Utviklingen med å kombinere RPA-teknologi med kunstig intelligens og maskinlæring, samt støtte for statistikkanalyse av data, er hvor Automation Anywhere skiller seg ut. Programvaren tilbyr per dags dato teknologi som kombinerer dette i en robot [20].

UiPath

UiPath ble grunnlagt i 2005 i Romania [21]. I motsetning til Blue Prism, har UiPath en åpen plattform og tilbyr en gratis versjon i tilegg til en versjon laget for større selskaper som krever betaling. Dette vil si at det er mulighet for å utvide platformen med andre funksjonaliteter og tredjeparti applikasjoner etter eget ønske. Et annet punkt som skiller UiPath fra både Blue Prism og Automation Anywhere, er at UiPath benytter seg av en web-basert arkitektur for å distribuere en robot til en prosess, mens Blue Prism og Automation Anywhere bruker klassisk klient-server arkitektur.

Selve verktøyet er bygget på Microsofts Workflow Foundation (også bygget opp av .NET rammeverket) og er svært brukervennlig [21]. Dette gir en løsere struktur for å utvikle kode enn Blue Prism og Automation Anywhere, som gjør det enklere å lære, men det krever mer selvdisiplin dersom flere utviklere skal arbeide på samme prosess.

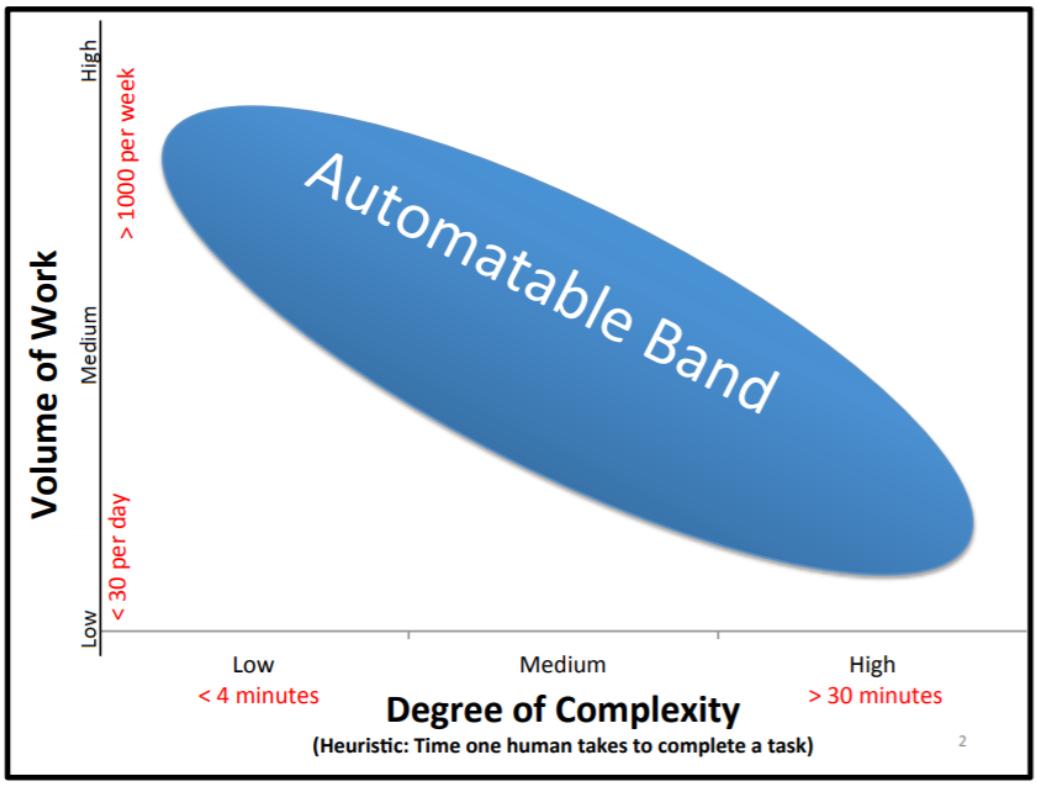
2.4 Tidligere RPA-prosjekter

Det har vært en økende interesse for RPA-teknologi de siste årene. I 2017 utførte Deloitte en undersøkelse der 400 bedrifter fra hele verden deltok. Det ble anslått at 73% av bedriftene ønsket å benytte seg av RPA-teknologi innen 2020 [22, s.2-3]. Innen nøyaktighet, tid og fleksibilitet, møtte eller overskred RPA forventningene hos 85% av bedriftene som allerede hadde implementert RPA [22, s.4]. Samtidig viser samme undersøkelse at 63% av bedriftene hadde høyere forventninger til implementasjontid, og 37% av bedriftene fikk ikke oppfylt forventingene til implementasjonskostnader [22, s.4]. Det kommer også frem at bare 3% klarer å videreskalere RPA til over 50 roboter i produksjon [22, s.7]. Disse tallene viser at det eksisterer fallgruver når det gjelder implementering av RPA. De største utfordringene som ble pekt på i rapporten var standardisering av prosesser, integrasjon og fleksibilitet, støtte fra IT-avdelinger, urealistiske forventninger hos interessenter og innvirkning på ansatte [22, s.14].
Videre tar vi for oss to gjennomførte prosjekter der RPA-teknologi har blitt implementert, og ser på effektene av implementeringen samt hvilke faktorer førte til at det ble en suksess. I begge prosjektene ble Blue Prism brukt som teknologi.

Telefónica O2

Telefónica O2 er britisk datterselskap av det spanske telekommunikasjonsselskapet Telefonica Group [23, s.5]. Selskapet var i 2018 den nest største teleoperatøren i Storbritannia [24]. I 2010, ble Telefónica interessert i å undersøke om RPA-teknologi kunne bidra til å kutte kostnader. Telefónica trente opp tre av sine egne ansatte i bruk av Blue Prism, som innen tre månender kunne automatisere prosesser fra ende-til-ende [23, s.8]. Innen 2015 hadde de automatisert 15 kjerneprosesser, som representerte om lag 35% av alle administrative transaksjoner [23]. Dette kuttet kostnader tilsvarende flere hundre fulltidsansatte, som ble omfordelt til andre tjenesteområder [23, s.8].
Gjennom deres vellykkede implementasjon av RPA, identifiserte Telefónica hvilke type oppgaver som egnet seg for automatisering. Fra tidligere erfaringer med tjenesteutsetting, visste de at prosesser med et høyt antall repeterende oppgaver gir størst mulighet til å redusere kostnader [23, s.9]. I følge Telefónica var eldre prosesser er også å foretrekke fordi de er godt dokumenterte, stabile, forutsigbare og kostnadene er kjent. Ut i fra disse faktorene, utarbeidet selskapet en modell for hvilke oppgaver som egner seg best for RPA-automasjon der de sammenligner antall saker per tid med tid brukt per sak. Resultatet ble "The Automatable Band"

vist i figuren nedenfor.



Figur 2.4: Telefónica O2s modell over saker som egner seg til RPA-automasjon

Kilde: [23]

Telefónica O2 konkluderte med at hvis det skal være lønnsomt å automatisere enkle oppgaver, må det være et høyt antall oppgaver. De presiserer at kompliserte oppgaver fortsatt kan bli lønnsomme dersom det utføres rundt 30 slike oppgaver per dag.

Xchanging

Xchanging er en leverandør av teknologibasert forretningsbehandling og innkjøpstjenester til kunder i mange bransjer [25]. Gjennom teknologi og innovasjon har Xchanging som mål å gjøre administrative tjenester bedre, raskere og mer kosteffektivt for sine kunder [26, s.6]. Xchanging ønsket å undersøke om RPA egnet seg i deres forsikringsvirksomhet. De observerte at de hadde en enorm mengde administrative oppgaver med repeterende data, der mange av dem ble utført manuelt [26, s.9]. Slik man ser i mange bedrifter ble informasjon hentet fra ulike kilder ble videre input til et annet system, eller brukt til å generere rapporter. I likhet med Telefóncia,

kom Xchanging fram til at slike regelbaserte oppgaver med høy grad av standardisering og et høyt antall transaksjoner var optimalt for RPA. De hadde likevel utfordringer i starten med å vite nøyaktig hvor mange transaksjoner som var nok til å gi gevinst [26, s.9]. Innen mai 2015, hadde Xchanging automatisert 10 prosesser. I en prosess der det tidligere tok flere dager for en person å behandle 500 saker, kunne en riktig programmert robot behandle sakene på omkring en halvtime, uten feil [26, s.12]. De observerte i tilegg at de automatiserte prosessene hadde en suksessrate på 93%, som er høyere enn deres opprinnelige mål på 80%. Dette resultatet skyldtes kontinuerlig forbedring av prosessene [26, s.13].

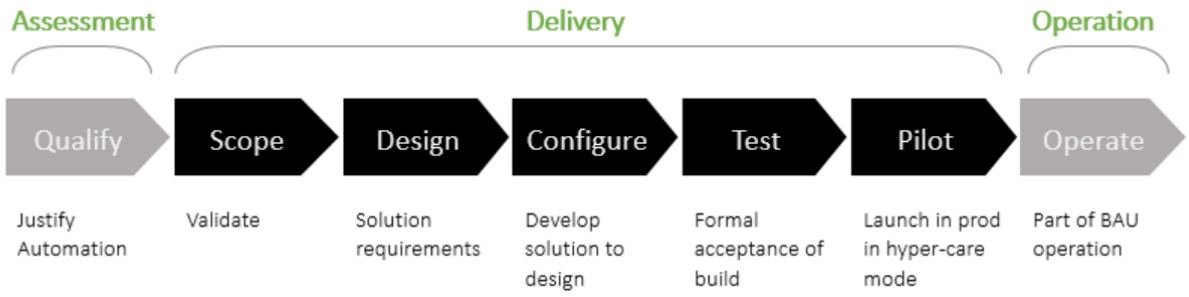
2.5 Utviklingsprosess

Skatteetaten deler gjerne RPA-utviklingsprosessen i tre faser:

1. **Evaluering**
   Kartelegge og evaluere behovet for RPA. Analysere oppgavens kompleksitet og tidsbruk, og undersøke om automatisering av prosessen er berettiget.

2. **Design, utvikling, testing og implementasjon**
   Prosessen kartlegges i detalj sammen med den som er ekspert på prosessen, som viser hvordan oppgaven løses i dag. Brukes som utgangspunkt for selve utviklingen. Justeringer av automatiseringen og testing foregår kontinuerlig, før det godkjennes i QA prosess i testmiljø at roboten kan settes i produksjon.

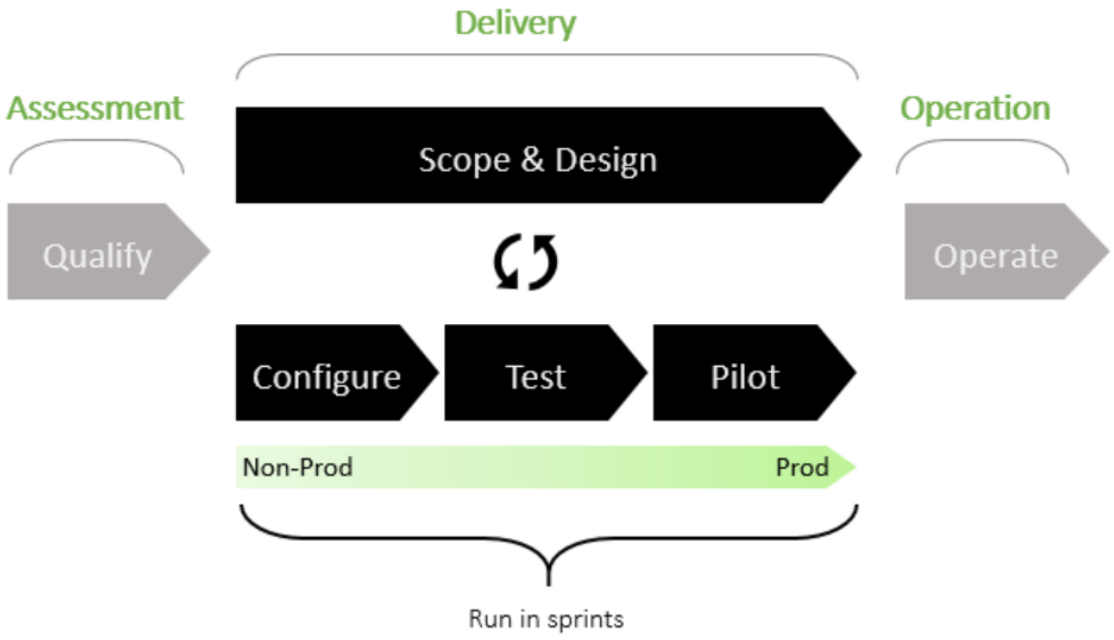
3. **Monitorering og forbedring**
   Prosessen overvåkes for å sikre at automatiseringen opererer korrekt. Håndtering av unntak og tekniske feil som kan oppstå. Generell vedlikehold og forbedring med input fra feil og unntakshåndtering.



Figur 2.5: Vannfallsmetode for RPA-utvikling, hentet fra Deloitte

Kilde: [27]

De tre fasene Skatteetatens benytter, gjenspeiles i Deloittes figur vist over. I begge tilfellene er det tydelig at prosessen foregår sekvensielt, der et steg må bli fullført før neste steg kan starte. Dette er også kjent som vannfallsmetoden. Steget for design, utvikling, testing og implementasjon ("Delivery" i figur 2.5) kan derimot gjennomføres iterativt i sprinter, slik figur 2.6 viser:



Figur 2.6: Iterativ metode i RPA-utvikling, hentet fra Deloitte

Kilde: [27]

Et av prinsippene bak iterativ utvikling er å være tilpasningsdyktig fremfor å følge en plan [28]. Endringer kan forekomme etter tilbakemeldinger fra brukere eller kunder, og det er viktig å kunne respondere på disse. Det finnes en mengde ulike iterative metoder for systemutvikling. Ulike elementer ved metodene, gjør at de egner seg for ulike typer prosjekter. Iterative metoder egner seg ofte for større, komplekse prosjekter der man ikke nødvendigvis har god oversikt fra start.

Kanban og Scrum er to eksempler på iterative metoder. Kanban er en kontinuerlig metode som har fokus på å unngå flaskehalser under utviklingen, og egner seg godt for både små og større team [29]. For å få oversikt over hvilke oppgaver som skal løses benyttes en Kanban-tavle. Tavlen visualiserer arbeidsoppgavene som skal løses, hvem som arbeider på oppgaven, og om arbeid på en oppgave pågår, er stanset eller ferdig. Til forskjell fra Scrum, som også er en iterativ metode med lignende tavle kalt "Scrum board", begrenser Kanban antall oppgaver som kan havne i hver kolonne. Dersom man blir arbeidsledig er det ikke lov til å legge til flere oppgaver, men man skal hjelpe de andre teammedlemmene å bli ferdig med sine oppgaver. I Scrum er det vanlig å bruke mye tid på å rangere forventet arbeidsmengde som hver av oppgavene krever, dermed vil utformingen av en slik tavle kreve mer planlegging. Til forskjell fra Kanban som er en kontinuerlig metode, foregår utviklingen i Scrum i sprinter. Sprintene planlegges ved å definere spesifikke roller, artefakter og seremonier [28]. For å utnytte Scrum på best mulig måte, må sprintene være nøye planlagt.

Som nevnt tidligere er det ulike utviklingsmetoder som passer til ulike prosjekt. Prosjektets omfang, tidsramme og antall teammedlemmer er variabler som har innvirkning på hvilken metode som passer. Alle prosjekt er forskjellig, og det kan også være en idé å velge ut elementer fra ulike metoder for å tilpasse sitt behov. Er man et stort team vil det være nødvendig med strammere rammer enn for eksempel et team på to eller tre personer. Uavhengig av teamets størrelse er det viktig å ha et bevist forhold til hvilken utviklingsmetode man velger å følge for å sikre fremgang i prosjektet og passe på at delmål blir møtt.

Kapittel 3

Valg av teknologi og metode

3.1 Blue Prism

Et av premissene for gjennomførelsen av prosjektet var at utviklingen skulle foregå i Blue Prism. Dette var ikke et valg som ble gjort av utviklerne, men som var forhåndsbestemt av Skatteetaten. For å implementere roboten i det allerede eksisterende systemet til Skatteetaten, var det også naturlig å utvikle i Blue Prism. Dette med tanke på både lisenser, kjøring og vedlikehold.

Det er tre hovedårsaker til at Skatteetaten har valgt å bruke Blue Prism som verktøy.

1. **Prismodell for lisensiering**
   Blue Prism hadde en lisensmodell som passet Skatteetaten godt med tanke på utviklerlisenser, antall lisenser og bindingstid. Som nevnt i kapittel 2.3, har Automation Anywhere en lignende lisensmodell, men de har til nå vært mest fremtredende i USA og ikke hatt et stort fokus på det europeiske markedet.

2. **Sikkerhet**
   Skatteetaten forvalter store mengder informasjon, som inneholder sensitiv informasjon. Dette stiller høye krav til sikkerhet når en robot skal behandle saker. Blue Prism er anerkjent for sin gode sikkerhet og har viktige sikkerhetsmekanismer som følger med programvaren, som for eksempel Credential Manager (se vedlegg C.5.2). Dette er en tilleggsfunksjonalitet som UiPath tar betalt for i tilegg til deres originallisens.

3. **Struktur**

   Blue Prism har en strammere struktur for hvordan man lager objekter og bygger koden. Dette kan gjøre det utfordrene for utviklere uten programmeringskunnskap, men når flere utviklere arbeider på samme prosess vil det ikke være nødvendig med like stor grad av selvdisiplin ettersom Blue Prism vil sette begrensninger. UiPath har en løsere struktur, og selv om dette vil gjøre det enklere for Ola Nordmann å programmere, setter det også høyere krav til selvdisiplin. Dersom utvikleren ikke har denne disiplinen vil det gjøre det vanskeligere for andre å bygge videre på og vedlikeholde koden. Skatteetaten hadde et ønske om større grad av standardisering, og Blue Prism var da et passende valg.

I forhold til valg av teknologi, er det ingen direkte sammenheng mellom Skatteetatens årsaker og kravene som ble satt opp i visjonsdokumentet. Visjonsdokumentet beskriver stabilitet, ytelse, pålitelighet, unntakshåndtering og enkelt vedlikehold som krav til roboten. Dersom valget skulle blitt tatt av studentene basert på disse kravene, ville det vært naturlig å også vurdere andre teknologier.

3.1.1 Konfigurasjon

Her beskrives alle nødvendige krav som må innfris før konfigurasjon av Blue Prism, samt valgfrie konfigurasjoner av Blue Prisms funksjonaliteter som utviklerne har benyttet seg av under utvikling av robotene.

Lisens og programvare

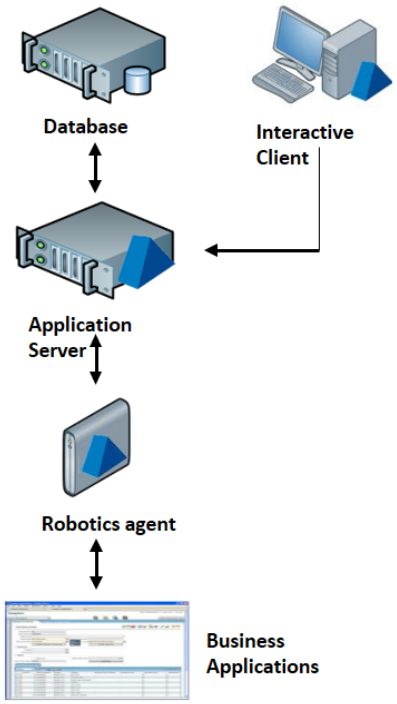
For å få tilgang til Blue Prims programvare må man kjøpe lisens. Blue Prism tilbyr en startpakke som inneholder prosess -og objektstudio, produksjonslisens, testlisens og Control room. Etter å ha fått tilgang til Blue Prism programvaren, er det krav som stilles til et robotmiljø som er designet for å støtte robotene i deres utførelse av automatiserte prosesser. I robotmiljøet er applikasjonsserver, database, klientdatamaskin og robotmaskin nødvendige komponenter.

Applikasjonsserveren er et blandet rammeverk av programvare som tillater opprettelse av både webapplikasjoner og et servermiljø til å kjøre dem [30]. Applikasjonsserveren fungerer som et mellomledd mellom database, robotmaskinene og klientmaskinene. Den vil lagre informasjon og logger fra prosessene på databasen, og sørger for at robotmaskinen har nødvendig informasjon til å utføre en prosess [31]. Applikasjonsserveren vil også kontrollere at bare klienter med riktige

tillatelser kan aksessere databasen [31].

Robotmaskinen er en virtuell maskin som er satt opp som en robotbruker og samhandler med annen programvare på samme måte som et menneske ville ha gjort.

Selve utviklingen og testingen av prosessen samt kontroll av robotmaskinens kjøretid initieres av en interaktiv klientdatamaskin, som brukes av en utvikler eller prosesskontroller. Prosesskontroller igangsetter og observerer kjøringen av roboten.



Figur 3.1: Figur over komponentene og deres kommunikasjon i robotmiljøet.

Kilde: [31]

Arbeidskø (Work Queue)

Arbeidskøen er en funksjonalitet som Blue Prism tilbyr i deres programvare. En prosess kan koble seg opp mot en eller flere arbeidskøer, legge inn saker i køen og deretter plukke dem opp en etter en for å behandle dem. Dette gjør det mulig å dele arbeidsoppgaver mellom flere roboter, der de kan plukke opp forskjellige saker fra arbeidskøen for behandling. Arbeidskøen gjør det i tilegg mulig å registrere resultatene av saksbehandlingen og se hvor lang tid det tar å behandle en sak. Saker som blir behandlet korrekt og uten feil, vil bli markert som fullført.

Dersom det skjer en feil i saksbehandlingen og saken ikke blir fullført, vil den blir markert som unntak. Det er mulig å legge ved en tekstlig beskrivelse av unntaket.
En prosesskontroller overvåker og vedlikeholder arbeidskøen fra Control Room, som er der prosesser blir kjørt fra (Merk: Det er mulig å kjøre prosessen fra selve utviklingsmiljøet til prosessen, men dette regnes som debugging i Blue Prism). Kunden vil motta en rapport som blir utformet som en Excel-fil med informasjonen som ligger i arbeidskøen. Rapporten inneholder blant annet informasjon om Business Exceptions, der som er kontrollerte unntak der saken går til manuell behandling.

Tilbakemelding via rapporten er viktig for videre vedlikehold og gjør det enkelt for kunden å oppdatere eventuelle data som roboten bruker underveis. På den måten kan roboten behandle en sak den tidligere ikke hadde ressursene til å behandle. Dersom mange saker feiler, vil prosesskontroller opplyse om dette til utviklerne, som må gå inn i koden og gjøre nødvendige endringer. En fordel med statusrapportene som genereres i arbeidskøen er dermed at man slipper å gå inn i selve koden for å få status over behandlede saker.

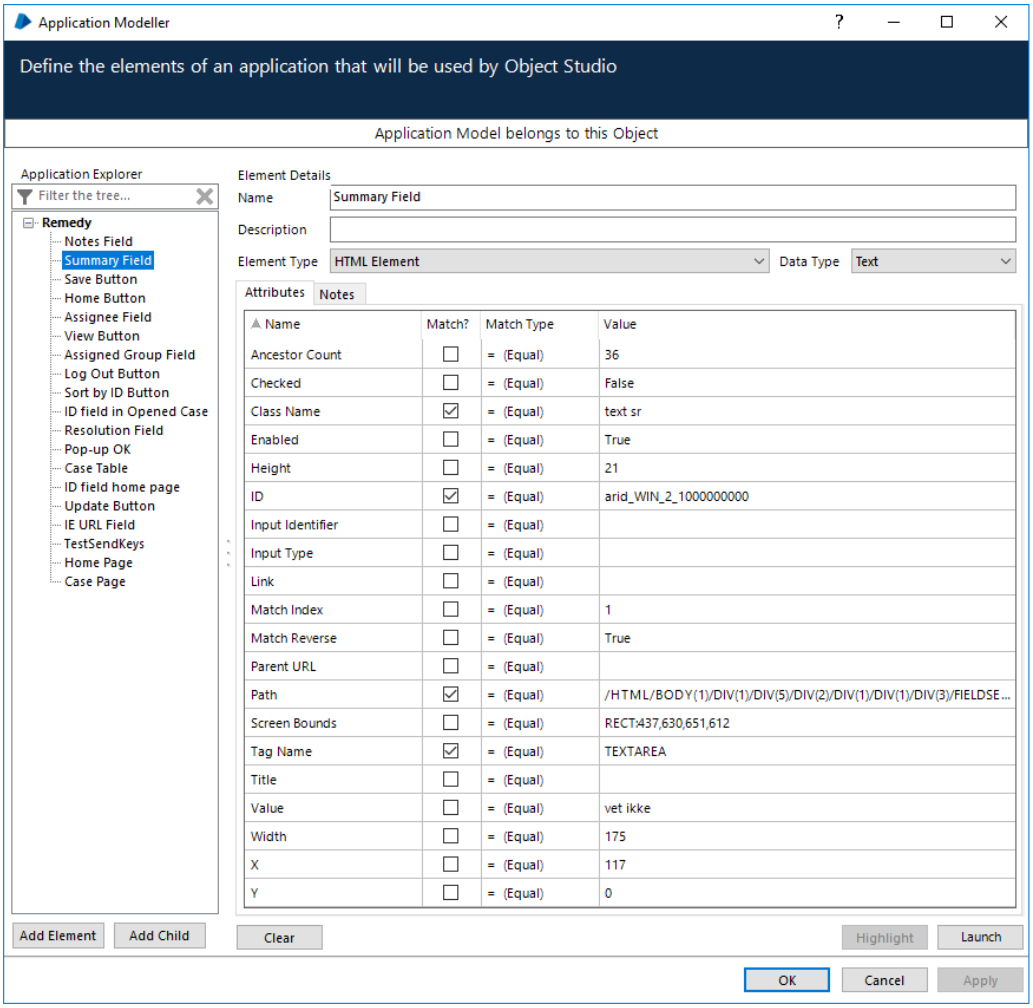
Queue Contents Clear Filters Show Positions in Queue

	Item Key	Priority	Status	Tags	Resource	Attempt	Total Work Time
	INC000000273209	0		Exception: Business Exception	RIW0RPA-APP11	1	00:10
	INC000000273208	0	3. Linje sak		RIW0RPA-APP11	1	00:25
	INC000000273207	0	3. Linje sak		RIW0RPA-APP11	1	00:24
	INC000000273206	0		Exception: Business Exception	RIW0RPA-APP11	1	00:15
	INC000000273205	0		Exception: Business Exception	RIW0RPA-APP11	1	00:10
	INC000000273204	0			RIW0RPA-APP11	1	00:42
	INC000000273203	0			RIW0RPA-APP11	1	00:37
	INC000000273201	0		Exception: Timeout Exception	RIW0RPA-APP11	1	01:05
	INC000000273109	0			RIW0RPA-APP11	1	00:32
	INC000000273108	0			RIW0RPA-APP11	1	00:33
	INC000000273103	0			RIW0RPA-APP11	1	00:33
	INC000000273001	0			RIW0RPA-APP11	1	00:35

Figur 3.2: Arbeidskøen etter en kjøring i Remedys testmiljø. Grønn hake betyr fullført sak og flagg betyr feil under utførelsen. Det er flere attributter i arbeidskøen, men de er kuttet for visningens skyld.

Spioneringsverktøy (spying)

For at roboten skal finne feltene den skal samhandle med, benytter man spioneringsverktøyet i Blue Prism. Verktøyet baserer seg på skjermskraping-teknologi. Før det kan spioneres på elementer må applikasjonen være modellert i "Application Modeller". Der opprettes så elementer for knapper og felt man ønsker å spionere på. Det er flere ulike moduser for spionering som egner seg for ulike typer applikasjoner. Utviklerne valgte å bruke Internet Explorer (IE) HTML Mode for spionering i Remedy, ettersom dette er modusen som egner seg best for spionering av web-applikasjoner i IE [32]. IE HTML Mode samhandler med brukergrensesnittet ved hjelp av det øverste HTML-laget. Etter å ha spionert et felt, huker man av bokser man ønsker skal brukes til å kjenne igjen feltet eller knappen.



Figur 3.3: Eksempel på spionering av et HTML-felt og attributtene man kan velge for gjenkjenning av feltet.

Miljøvariabler (Environment Variables)

En annen funksjonalitet som Blue Prism tilbyr, er miljøvariabler. Det som skiller disse fra vanlige variabler er at verdien blir spesifisert utenfor koden. Slik kan variabler endres og skreddersys når som helst av kunden via prosesskontroller. Ved å bruke en miljøvariabel kan AT Økonomi blant annet bestemme filstier og antall saker roboten skal jobbe med. Miljøvariabler gjør det også enklere når prosessen skal overføres fra testmiljø til produksjonsmiljø. Ved å benytte seg av en boolsk miljøvariabel, kan man for eksempel spesifisere om roboten jobber i test- eller produksjonsmiljø.

3.2 Vaktliste i Excel

For at Remedy-roboten skal vite hvem som skal tildeles en sak og hva saken skal kategorisere som, valgte utviklerne å bruke en vaktliste i form av en Excel-fil. Begrunnelsen for dette valget er at AT Økonomi benyttet seg av en slik vaktliste når de skulle tildele saker manuelt i Remedy. Vaktlisten for Remedy-roboten er en videreutvikling av den opprinnelige vaktlisten til AT Økonomi, der formatet har blitt tilpasset RPA med mulighet for skalering. Formatet på Excel-filen ble utformet av utviklerne i samarbeid med representanter fra AT Økonomi, både for å ta hensyn til robotens evne til å lese inn data og for at kunden som skal fylle filen med nødvendig informasjon og eventuelt oppskalere til andre avdelinger. En fordel med å bruke Excel for utforming av vaktlisten, er at det allerede eksisterer en Excel-VBO med ferdiglagede metoder, som muliggjør samhandling med Excel uten å måtte bruke spioneringsverktøyet. Roboten kan da hente ut data fra Excel-filen svært raskt uten å åpne selve filen. Dette vil redusere tiden og muligheten for feil, ettersom spionering er mer ustabilt.

3.3 Testing

RPA-prosesser i Blue Prism kan ikke testes ved å lage egne, kjørbare tester slik det er vanlig å gjøre i klassiske systemutviklingsprosjekt. Testing utføres ved å kjøre roboten kontinuerlig og se etter feil som oppstår under kjøring. For å kvalitetsikre roboten, har Skatteetaten egne rutiner for hvilke tester roboten må gjennom før produksjonssetting. Dette innebærer at all utvikling skal foregå i testmiljø, helt til den godkjennes under QA-prosess i testmiljø. Her lager kunden relevante testsaker og et testskript, som viser forventet resultat av kjøringen. Etter kjøring

plottes det faktiske resultatet inn i skriptet, og sammenlignes med de forventede verdiene. Etter QA-prosess i testmiljø gjennomføres prøveproduksjon ved å bruke samme metode, bare i produksjonsmiljøet. Dette er for å sikre at roboten jobber likt i produksjon og i test. Dersom roboten arbeider som den skal i prøveproduksjon, settes roboten til å behandle en eller flere reelle saker. Blir roboten godkjent etter prøveproduksjon og QA prosess i testmiljø, er den klar for å produksjonsettes.

3.4 Utviklingsmetode

Teamet har valgt å bruke en hybrid mellom sekvensiell vannfallmetode og iterativ metode (se kapittel 2.5 og figur 2.6). Avgjørelsen begrunnes med at utviklingsprosessen for RPA har elementer fra begge metodene, ettersom selve hovedfasene må utføres sekvensielt. Evalueringen av prosessen må gjennomføres og godkjennes av kunden før design og utvikling av roboten kan starte. Produktet må testes med kunden og ferdigstilles før prosessen kan settes ut i produksjon. Problemet med å kun benytte seg av vannfallmetoden er at den er lite fleksibel når det kommer til endringer underveis, og i et større prosjekt som dette er det nødvendig med kontinuerlige forbedringer i utviklingen av roboten.
Med en iterativ metode følger rom for endring og samhandling underveis. I utviklingsfasen var er det helt nødvendig for utviklerne å fortsette tett kommunikasjon med kunden etter startfasen. For å kunne levere et kvalitetsprodukt er tilbakemeldinger fra kunden essensielt. Det er vanlig at kunden kommer med nye ideer eller forbedringer underveis i utviklingen, og man må ofte gå tilbake og endre på både dokumentasjon og kode.
Det var spesielt viktig med en iterativ utviklingsmetode for dette prosjektet da ingen av teammedlemmene hadde erfaring med RPA fra før. Det var nødvendig å ha mulighet til å endre på koden og produktets funksjonaliteter underveis for å oppnå beste praksis og et produkt som dekker kundens krav.

Fra tidligere systemutviklingsprosjekt har teammedlemmene erfaring med Scrum, som tidlig ble vurdert som mulig utviklingsmetode. I dette prosjektet ville det blitt en utfordring ettersom mangel på erfaring med RPA-utvikling, ville gjøre det vanskelig å planlegge sprintene. I tillegg hadde teamets størrelse betydning for at Scrum ikke ble valgt som metode. Teamet ble enig om å benytte seg av en iterativ prosess som ikke var like rigid, og med mindre fokus på arktefakter og roller.

Ut fra dette falt valget på å bruke prinsipper ved Kanban som utviklingsmetode. For teamet var oppdelingen og visualiseringen av oppgavene viktig, uten det store fokuset på sprint planlegging og seremonier.

3.4.1 Organisering

For å organisere arbeidet, har teamet valgt å bruke nettsiden Trello som gjør det enkelt å holde oversikt over det store bildet og de mindre oppgavene i sanntid på en nettside. Teamet valgte nettside istedenfor en fysisk tavle fordi nettsiden er raskere å sette opp, og gjør det mulig å vite hva andre medlemmer arbeider med uten å være i samme rom som tavlen.
Teamet deler opp arbeidsoppgavene i mindre deler og skriver de inn slik at man enkelt ser hva som må gjøres. Man kan også tildele oppgaver til en person som da har ansvar for at den spesifikke arbeidsoppgaven fullføres. Teamets Kanban-tavle på Trello består av fire faner; "Må gjøres", "Pågår", "Ferdig" og "Stanset". Arbeidsoppgavene under må gjøres, dras til de andre fanene etter hvert som de jobbes på og blir utført. Oppgavene kan også markeres med farger, slik at det er mulig å kategorisere oppgavene ytterligere. Ettersom teamet arbeider på en Kanban-inspirert måte, er det er kun mulig å ha fire oppgaver under "Pågår" samtidig.

Samhandlingsverktøy

AT Økonomi er lokalisert i Lillehammer og utviklingsteamet i Trondheim, derfor foregikk kommunikasjonen ved hjelp av samhandlingsverktøyet Skype for Business. Skype gjorde det mulig for AT Økonomi å dele sin skjerm og vise prosessen steg for steg under kartleggingsmøter. Teamet tok også opptak av viktige møter, som var nyttig å bruke underveis i utviklingen av robotene. Bortsett fra Skype ble Outlook brukt for å sende e-post, kalle inn til møter og dele kalender. Selve bacheloroppgaven er skrevet i LaTeX med samskrivingsverktøyet Overleaf, slik at teamet kunne skrive på hovedrapporten samtidig.

Arbeids- og rollefordeling

Teamet består av tre utviklere, som har holdt en flat struktur. Dette på grunn av likt kompetansenivå, få teammedlemmer og at arbeidet i all hovedsak har foregått i samme rom. Det har vært viktig for teamet å ha klart definerte roller, der alle vet hva de har ansvar for. Når alle jobber mot et klart definert mål, og vet hva arbeidoppgavene sine er, går ledelsen av seg selv. Teamet skrev under på en samarbeidsavtale i løpet av den første uken, der alle ble tildelt

en administrativ rolle. Camilla fikk rollen som referent, Kimia som møtekoordinator og Nora som dokumentansvarlig. I tillegg valgte teamet å dele opp arbeidsoppgavene på den måten at to personer programmerer, og en person jobber med dokumentasjon gjennom store deler av prosjektet.

Teamet har hatt en åpen og inkluderende kultur, der det er rom for å stille spørsmål og komme med ideer. Uenigheter har ofte blitt løst ved at det er to som er enige, og sistemann føyer seg etter flertallet.

Kapittel 4

Resultater

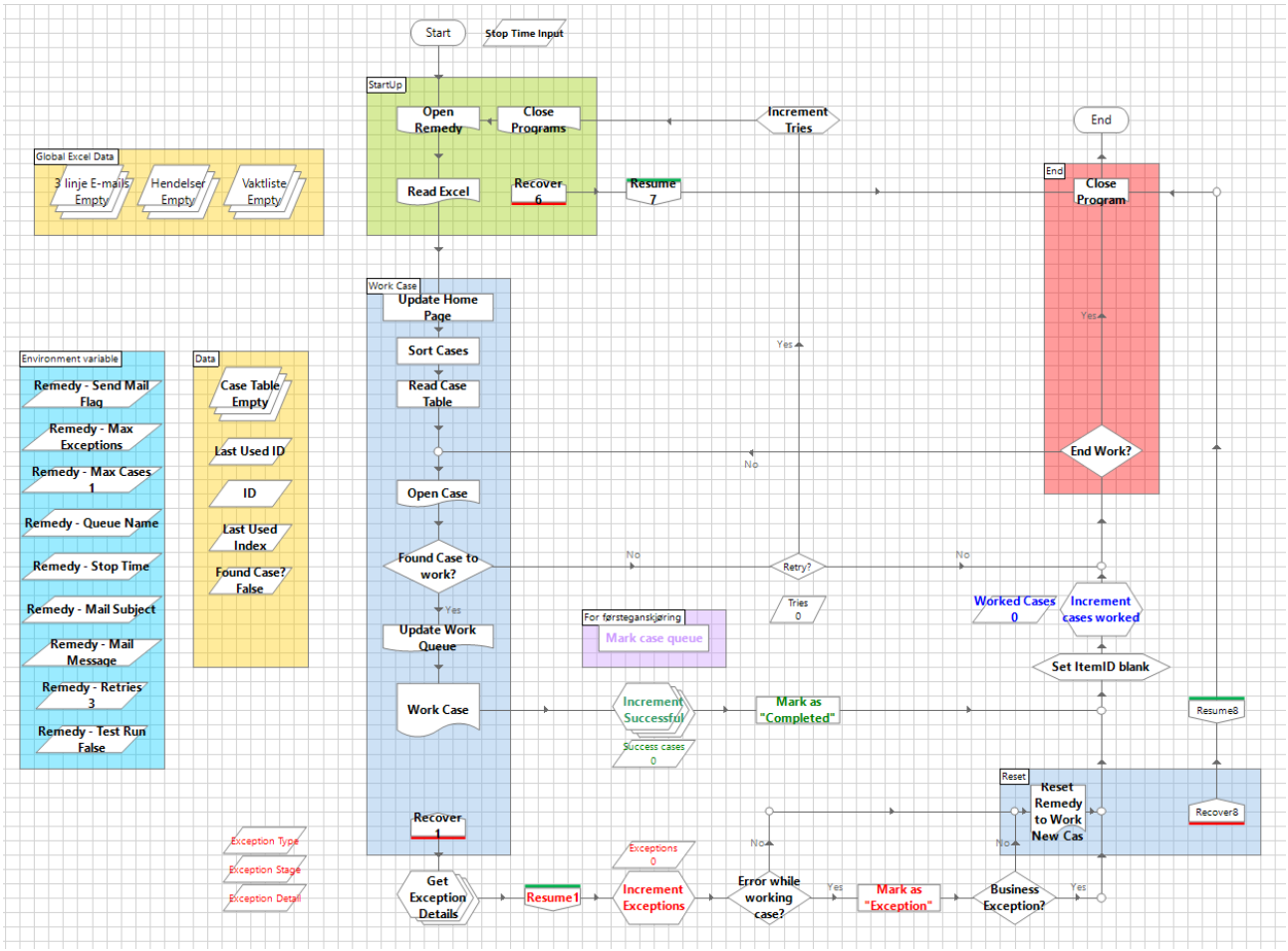
4.1 Vitenskapelige resultater

4.1.1 Produkt

Produktet i prosjektet er to RPA-roboter som utfører oppgaver for AT Økonomi i Skatteetaten. Ved utforming av hovedflyten til robotene er det tatt utgangspunkt i en mal utviklerne fikk tilsendt fra RPA-teamet i Skatteetaten. Hovedflyten kan ses i figur 4.1 og 4.2 på side 26 og 27.

Remedy-roboten er godkjent for produksjon og planlagt produksjonssatt mot slutten av mai 2019. Roboten består av en prosess med to VBOer. Én VBO for navigering i Remedy, og én for håndtering av popup-vindu. I tillegg bruker roboten to eksisterende VBOer, MS Excel VBO og MS Outlook Email VBO. Prosessen starter ved å åpne Internet Explorer, navigere seg til Remedys nettside der roboten allerede er logget inn via SSO. Deretter finner roboten Excel-vaktlisten fra et delt filområde og leser inn data fra arket som blir lagret i tabeller i Blue Prism. Dette skjer innenfor boksen "Global Excel Data". Videre vil roboten oppdatere siden, sortere alle sakene på INC-nummer, og lese inn sakene som ligger inne på Remedy. Deretter går den systematisk gjennom hver sak.
Roboten skal ikke behandle en sak den tidligere har behandlet. For å vite hvilken sak den skal behandle sjekker derfor roboten om INC-nummeret til saken ligger i arbeidskøen fra før. Dersom det allerede ligger i køen, vil roboten gå videre til neste neste sak. Dersom roboten finner en ubehandlet sak vil den åpne saken. Det første som sjekkes er om saken allerede er tildelt en saksbehandler, ettersom det er mulig at en ansatt har vært inne og manuelt tildelt saken før

roboten. Da vil roboten lukke saken og navigere til hovedsiden. Dersom saken er ubehandlet vil roboten hente ut informasjon om saken fra aktuelle felt, og lete etter triggerord i notatfeltet som forteller hvilken type sak den jobber med. Dette finner den ut fra Hendelser-arket i Excel-filen. Samsvarer triggerordene, vil roboten hente ut riktig verdi for sammendrag, hente ut saksbehandler som er på vakt for den gitte sakstypen, og skrive informasjonen inn i Remedy. Til slutt lagres saken, saken flagges som ferdig i arbeidskøen og roboten navigerer tilbake til hovedsiden. Slik fortsetter den helt til det ikke er flere ubehandlede saker igjen. Når roboten har iterert gjennom hele tabellen vil roboten avslutte Remedy og lukke Internet Explorer før den på nytt åpner programmene og gjentar de samme stegene. Antall ganger roboten vil prøve på nytt er lagt inn som miljøvariabel, og spesifiseres av kunden.



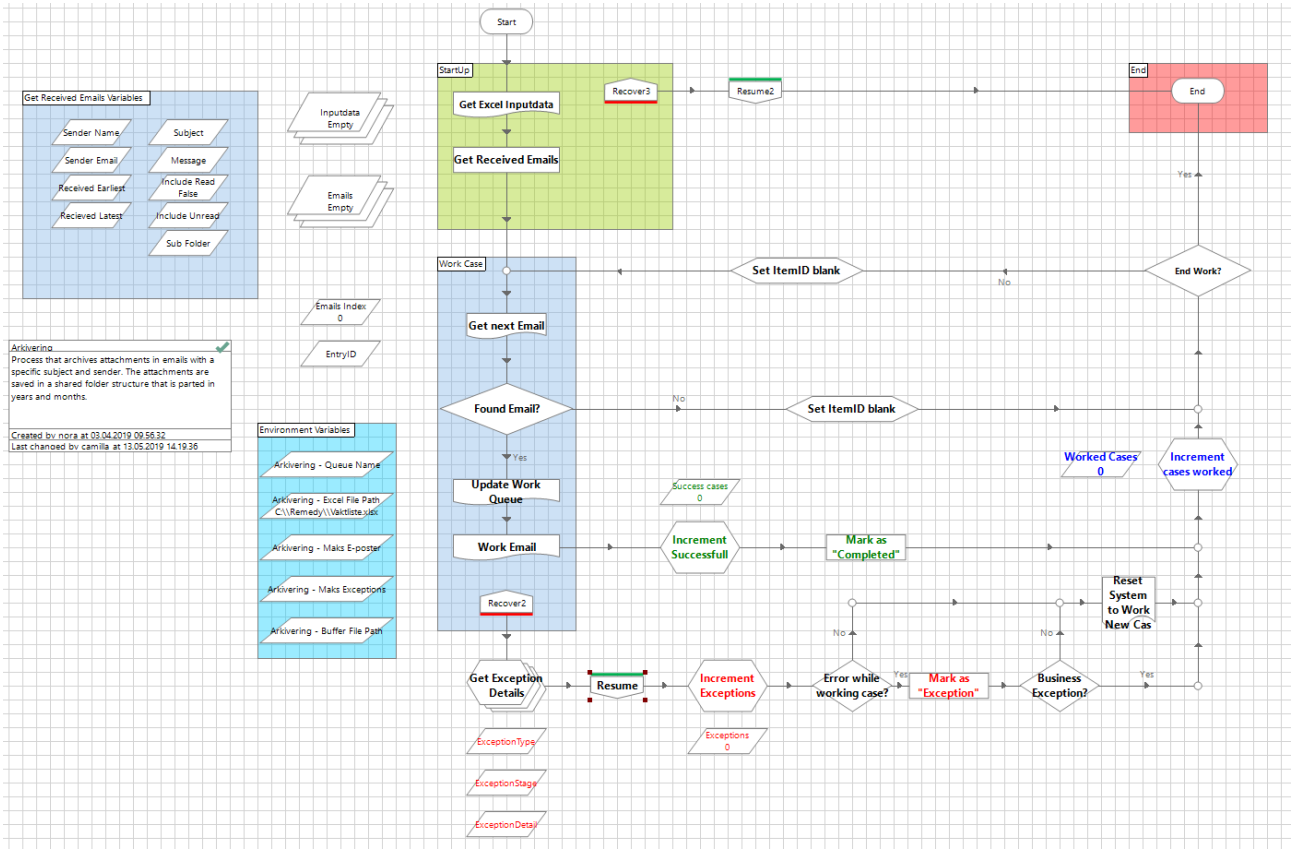
Figur 4.1: Hovedflyt i prosessen for Remedy-robot

Arkiv-roboten er kvalitetstestet i samarbeid med kunden, og har blitt godkjent for prøveproduksjon. Roboten er ikke helt ferdig for produksjon, men skal arbeides videre på av RPA-teamet i Skatteetaten. Hovedflyten til arkiv-roboten er veldig lik hovedflyten til Remedy-roboten (vist i

figur 4.1), ettersom begge prosessene benytter samme mal. Robotens oppgave er å lagre bilag som blir sendt til en e-post postkasse. Roboten leser inn alle uåpnede e-poster i den gitte postkasse og bruker verdier fra Excel-fil for å sjekke om kombinasjonen av avsender og emne gjør at e-posten skal behandles. Dersom en e-post skal behandles vil roboten lagre bilaget i et bufferområde for så å lese inn hele teksten i filen. Deretter leter den igjennom teksten etter aktuelle nøkkelord som blant annet hva saken gjelder, bilagsintervall og dato. Den sjekker også om bilaget er av typen "feillinje".

Dersom det er et "feillinje"-bilag skal roboten videresende e-posten til for videre behandling, og setter seg selv som kopi. Neste gang roboten leter etter nye e-poster i postkassen vil den arkivere e-posten som er sendt fra seg selv, og denne fungerer som en slags "kvittering" på at e-posten er videresendt til DFØ.

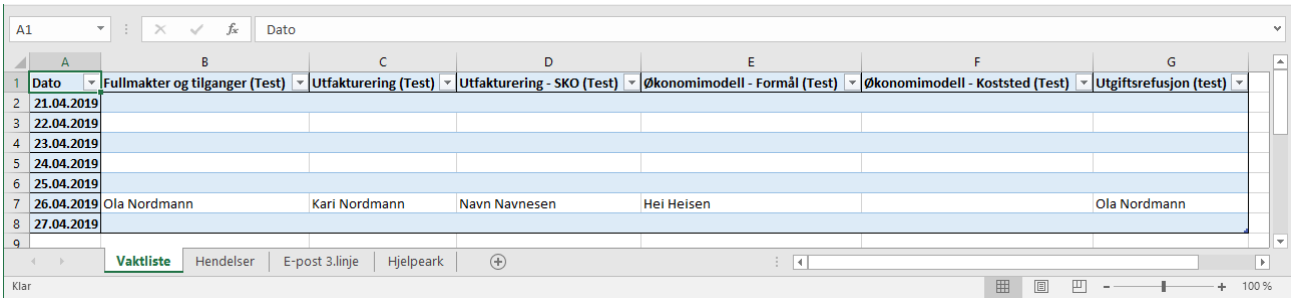
Dersom bilaget ikke er en "feillinje" vil roboten bruke dato og filsti fra Excel-arket til å finne ut hvor bilaget skal arkiveres. Filen lagres på riktig sted navngis med bilagsintervallet. Når dette er gjort vil den markere e-posten som lest og arkivere den.



Figur 4.2: Hovedflyt i prosessen for arkiv-robot

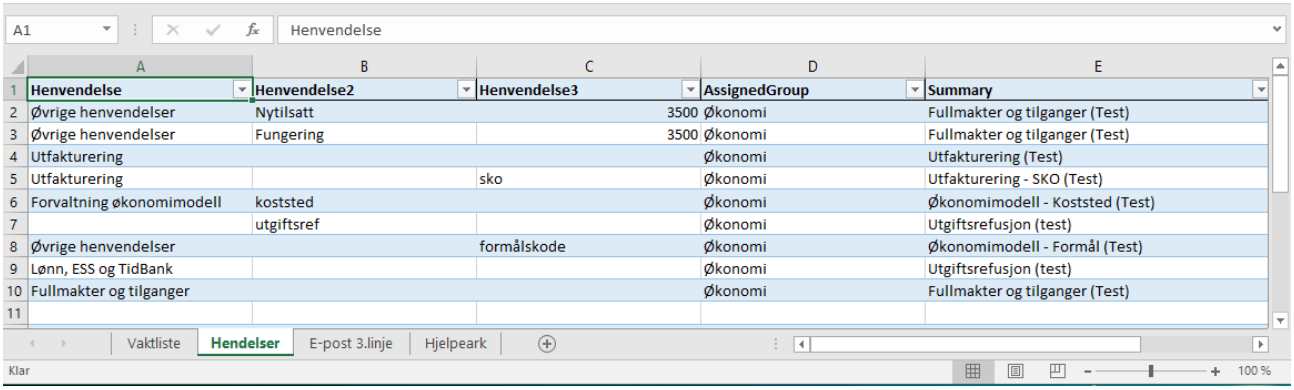
Excel-vaktliste

Vaktlisten til Remedy-roboten er en Excel-fil som ligger tilgjengelig for roboten i et delt filområde med begrenset tilgang i Skatteetaten. Vaktlisten ble utformet i samarbeid med kunden, som må holde den vedlike etter endt utvikling. Filen inneholder informasjon om hvilke triggerord som gjør at saken plasseres under bestemte kategorier og hvilke saksbehandlere som er satt av til å arbeide med hvilke saker. I tilegg vil arket inneholde mailadressene til de som er på vakt, som brukes i utsending av mail ved 3.linje sak.



Dato	Fullmakter og tilganger (Test)	Utfakturering (Test)	Utfakturering - SKO (Test)	Økonomimodell - Formål (Test)	Økonomimodell - Koststed (Test)	Utgiftsrefusjon (test)
21.04.2019						
22.04.2019						
23.04.2019						
24.04.2019						
25.04.2019						
26.04.2019	Ola Nordmann	Kari Nordmann	Navn Navnesen	Hei Heisen		Ola Nordmann
27.04.2019						

Figur 4.3: Vaktliste-arket i en Excel-eksempelfil med riktig formatering



Henvendelse	Henvendelse2	Henvendelse3	AssignedGroup	Summary
Øvrige henvendelser	Nytilsatt	3500	Økonomi	Fullmakter og tilganger (Test)
Øvrige henvendelser	Fungering	3500	Økonomi	Fullmakter og tilganger (Test)
Utfakturering			Økonomi	Utfakturering (Test)
Utfakturering		sko	Økonomi	Utfakturering - SKO (Test)
Forvaltning økonomimodell	koststed		Økonomi	Økonomimodell - Koststed (Test)
	utgiftsref		Økonomi	Utgiftsrefusjon (test)
Øvrige henvendelser		formålskode	Økonomi	Økonomimodell - Formål (Test)
Lønn, ESS og TidBank			Økonomi	Utgiftsrefusjon (test)
Fullmakter og tilganger			Økonomi	Fullmakter og tilganger (Test)

Figur 4.4: Hendelser-arket i en Excel-eksempelfil med riktig formatering

Det er viktig at arkene i Excel heter "Vaktliste", "Hendelser" og "E-post 3.linje", ettersom disse navnene er hardkodet for at roboten skal hente ut data fra riktig ark. Antall og navn på kolonnene i "Hendelser" må også være identiske med eksempel-figuren. Det er derimot ingen begrensninger på antall rader og kolonner i Vaktliste-arket, så lenge navnet på kolonnene samsvare med navnet under kolonnen "Summary" i Hendelser-arket.

4.1.2 Økonomi

Remedy-robot

Antall saker som tildeles i Remedy hvert år er på rundt 2 200 saker. I dag bruker saksbehandlerne omtrent tre minutter på å fordele hver sak. Dette koster skatteetaten tilnærmet 51 000 kr hvert år. Kostnadene kommer av lønn til de ansatte som løser oppgavene.
Roboten bruker 30 sekunder på hver oppgave. Dersom den behandler 2 200 slike saker i året vil det koste Skatteetaten totalt 166 000 kr de første 12 månedene. Dette inkluderer kostnader for utvikling, implementering og vedlikehold av roboten. For akkurat denne oppgaven, vil det altså være dyrere å automatisere enn å utføre oppgavene manuelt.
For at denne roboten skal være lønnsom må antall oppgaver per år være 6600.

Arkiv-robot

Skatteetaten begynte med elektronisk arkivering av bilag i 2018. Antallet oppgaver i 2018 var 3000, og det ble satt av 238 timer til å utføre disse. Dette tilsvarer 5 minutter per oppgave. Med de samme ratene som er brukt for Remedy-roboten tilsvarer dette cirka 116 000 kr.
Roboten bruker i gjennomsnitt 0,5 sekunder å behandle én e-post. Dersom roboten jobber med 3000 e-poster vil det koste Skatteetaten 156 000 kr, som er dyrere enn å løse oppgavene manuelt. Her har teamet antatt at kostnadene for lisenser, utvikling og implementering er 0 kr, ettersom det er naturlig at Skatteetaten ville brukt samme lisenser og robot som Remedy-roboten for å utnytte den best mulig. Remedy-roboten løser oppgavene raskt, og er inaktiv resten av tiden. På denne tiden kan den løse arkiv-saker.
For at denne roboten skal være lønnsom må antall oppgaver per år være 4500.

4.2 Ingeniørfaglige resultater

I begynnelsen av prosjektet ble mål for robotene beskrevet i visjonsdokument (se vedlegg A). I dette kapittelet vil vi kort ta for oss målene som fortsatt er relevante og beskrive status for hver av dem. Grad av oppnåelse rangeres fra 1 til 6, der 1 er lav grad av oppnåelse og 6 er høy grad av oppnåelse.

4.2.1 Mål

1. **Løser oppgaven raskere enn et menneske**

   En saksbehandler i AT Økonomi bruker omtrent tre minutter på å fordele en sak i Remedy. Roboten bruker omtrent 30 sekunder, som vil si $\frac{1}{6}$ av tiden. Arkiv-roboten behandler en e-post på 0,5 sekunder, sammenlignet med manuell behandling som tar i gjennomsnitt fem minutter. Dette er $\frac{1}{600}$ av tiden.

   Grad av oppnåelse: 6

2. **Løser oppgaven mer nøyaktig enn et menneske**

   De siste avgjørende testkjøringene ga samme resultat ved hver kjøring. Roboten løser oppgaven mer nøyaktig fordi den er konsekvent i utførelsen av oppgaven. Dersom roboten får lik input 100 ganger skal den gi samme resultat 100 ganger. Roboten vil ikke løse oppgaven dersom den ikke finner de riktige verdiene i Excel-arket.

   Grad av oppnåelse: 5

3. **Gir en robust løsning**

   Alle popup-vinduer, ulike utfall og hendelser som utviklerne har blitt gjort oppmerksom på, enten via informasjon fra kunden eller etter egen erfaring med programmet, har blitt tatt hensyn til. Robotene vil ikke håndtere utfall den ikke er programmert til å håndtere.

   Grad av oppnåelse: 4

4. **Gir høyere kvalitet**

   Roboten øker kvaliteten på utførelsen av oppgavene da det forekommer færre feil. Dette gjør at behovet for manuelle inngrep og dobbeltarbeid reduseres, som videre fører til redusert behandlingstid og økt arbeidskapasitet.

   Grad av oppnåelse: 5

5. **Samhandler med eksisterende systemer**

   Som nevnt i tidligere i kapittel 2.2.1 jobber RPA *oppå* andre program på samme måte som et menneske ville gjort. For Excel, Outlook og filbehandling bruker roboten ferdige metoder i Blue Prism, som gjør at roboten ikke trenger å åpne programmene for å samhandle med dem. For samhandling med Internet Explorer og Remedy ble Blue Prisms spioneringsverktøy brukt slik at roboten visste hvilke knapper og felter den skulle samhandle med.

   Grad av oppnåelse: 6

6. **Løsningen er 80-100% automatisert**

   Ettersom roboten ikke er satt i produksjon ved prosjektslutt, baseres graden av automasjon på testene som er gjennomført med kunden. Ut ifra varians- og volumtestene er robotene 100% automatisert. I praksis er det stor sannsynlighet for at løsningen ikke blir 100% automatisert før roboten har vært i produksjon en stund.

   Grad av oppnåelse: 5

7. **Sporbarhet ved feil**

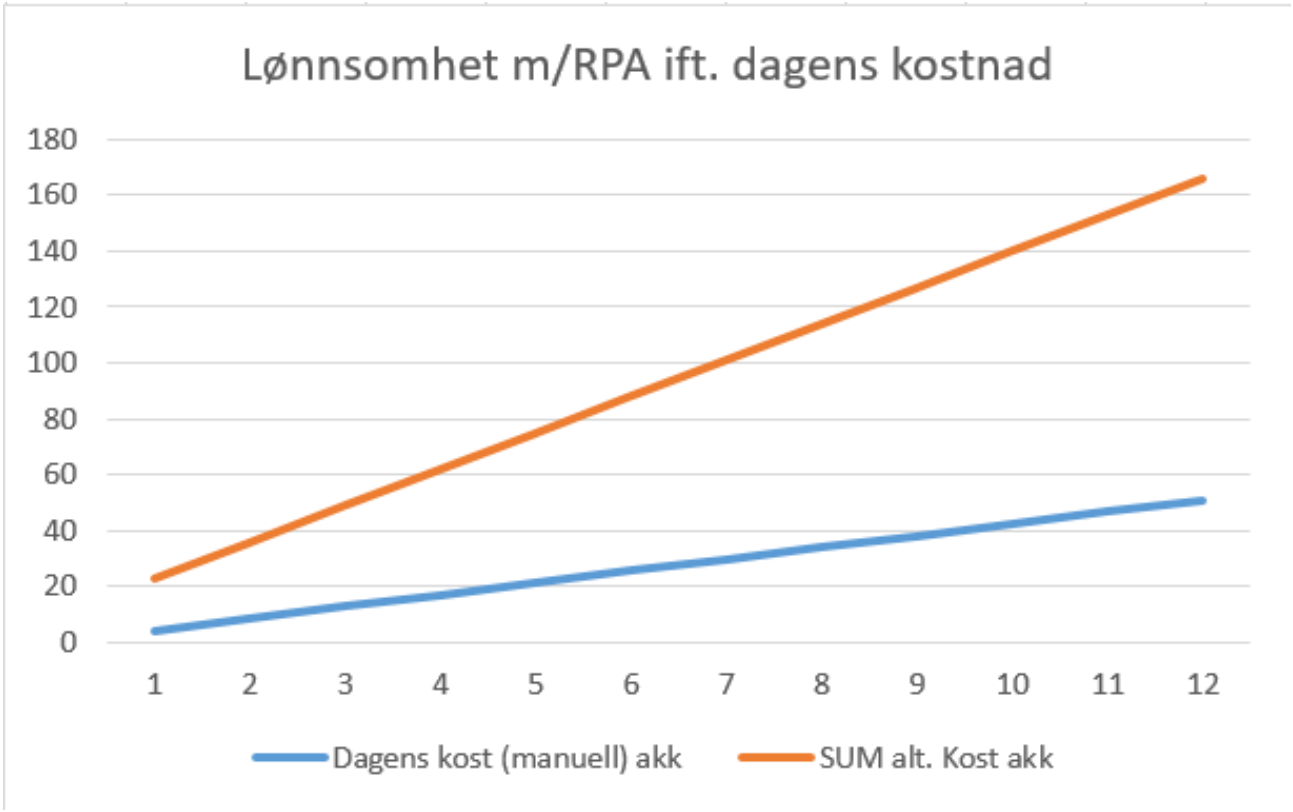
   Begge robotene bruker arbeidskø, som viser status på behandlede saker. Hver sak får i tillegg en beskrivelse som tydelig skal formidle hvilke eventuelle feil som har oppstått. Kunden har for begge robotene sagt seg fornøyd med hvilke tilbakemeldinger som blir gitt i loggen, og at disse er spesifikke nok til at kunden kan gjøre de endringer som er nødvendig i Excel-arket for å unngå at samme feil skjer på nytt.

   Grad av oppnåelse: 6

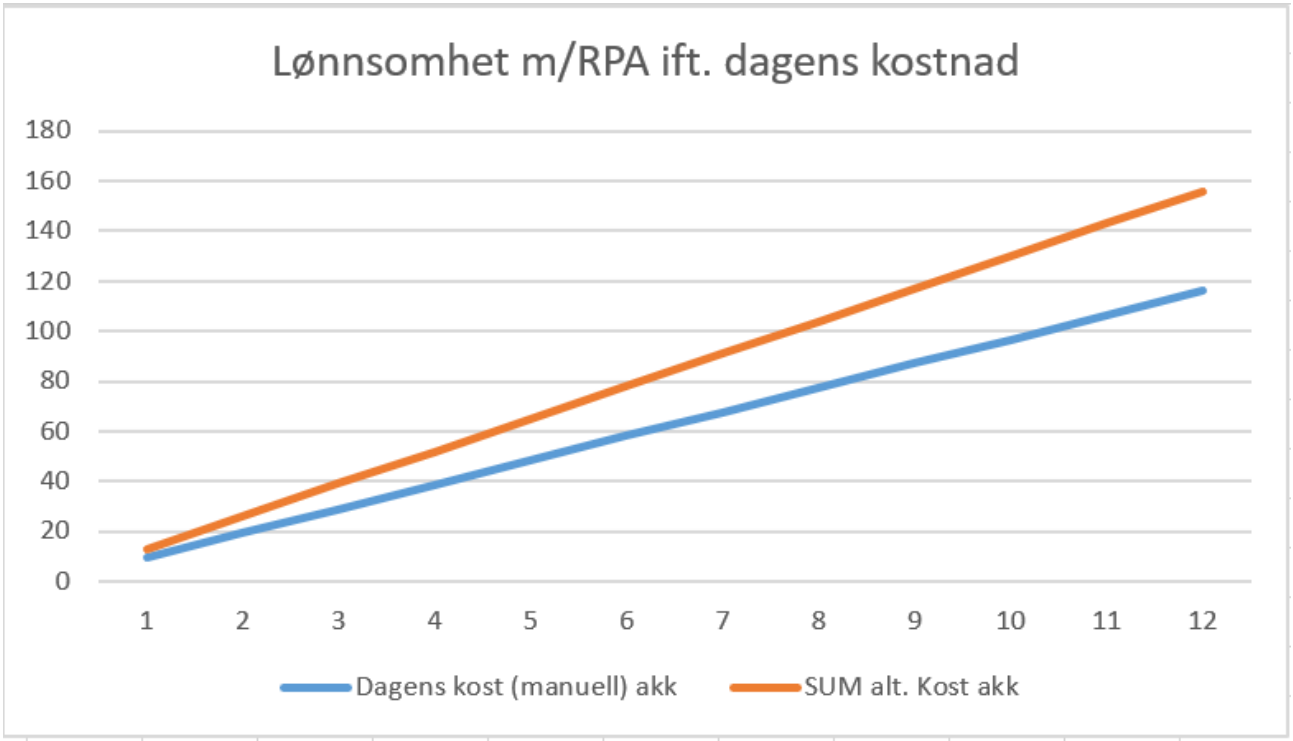
8. **Gir økonomisk gevinst**

   De økonomiske resultatene er dokumentert i kapittel 4.1.2. De viser at implementering av Remedy-robot vil være 115 000 kr dyrere enn å fortsette å utføre oppgavene manuelt. Arkiv-roboten blir 40 000 kr dyrere med RPA-implementasjon. Resultatet vises via grafen på figurene 4.5 og 4.6 nedenfor.

   Grad av oppnåelse: 1



Figur 4.5: Beregning av økonomisk lønnsomhet av Remedy-robot



Figur 4.6: Beregning av økonomisk lønnsomhet for arkiv-robot

4.2.2 Testing

Begge robotene ble kontinuerlig uformelt testet i testmiljøet under utviklingen. Av mer formelle tester var arkiv-roboten kun gjennom QA-prosess i testmiljø, og ble godkjent 13.05.19. Test-skript er ikke tilgjengelig, men kunden sendte test-mail til utvikler og påså via et Skype-møte at bilagene ble arkivert på riktig filsted, med riktig navn, og at feillinje ble håndtert riktig. Remedy-roboten ble godkjent i QA-prosess i testmiljøet 25.03.19. Etter fire prøveproduksjonsmøter i produksjonsmiljøet, ble Remedy-roboten godkjent 29.04.19 og forventes å være i produksjon i løpet av mai. Figur 4.7 viser resultat av test QA-prosess i testmiljø for Remedy-robot. Navnene til ansatte er anonymisert grunnet taushetsplikt.

Forventet resultat	Dato 25.03.2019 i vaktliste		RPA Resultat		
Summary	Assignee+	Distribuere 3.linje mail	Summary	Assignee+	Distribuere 3.linje mail
Andre henvendelser-Økonomi	Ola Nordmann	Nei	Andre henvendelser-Økonomi	Ola Nordmann	Nei
Avstemming-Økonomi	Ola Nordmann	Nei	Avstemming-Økonomi	Ola Nordmann	Nei
Budsjettering-Økonomi	Kari Nordmann	Nei	Budsjettering-Økonomi	Kari Nordmann	Nei
Forvaltning økonomimodell-Økonomi	Navn Navnesen	Nei	Forvaltning økonomimodell-Økonomi	Navn Navnesen	Nei
Inngående fakturaer-Økonomi		Nei	Inngående fakturaer-Økonomi		Nei
Omposteringer/manuelle bilag-Økonomi	Hei Heisen	Nei	Omposteringer/manuelle bilag-Økonomi	Hei Heisen	Nei
Rapporter-Økonomi	John Smith	Nei	Rapporter-Økonomi	John Smith	Nei
Utfakturering-Økonomi	Jane Doe	Nei	Utfakturering-Økonomi	Jane Doe	Nei
Vet ikke??	Per Nordmann	Nei	Vet ikke??	Per Nordmann	Nei
Vet ikke??		Nei	Vet ikke??		Nei
Avstemming-3. linje Økonomi		Ja	Avstemming-3. linje Økonomi		Ja
Forvaltning økonomimodell-3. linje Økonomi		Ja	Forvaltning økonomimodell-3. linje Økonomi		Ja
Inngående fakturaer-3. linje Økonomi		Ja	Inngående fakturaer-3. linje Økonomi		Ja

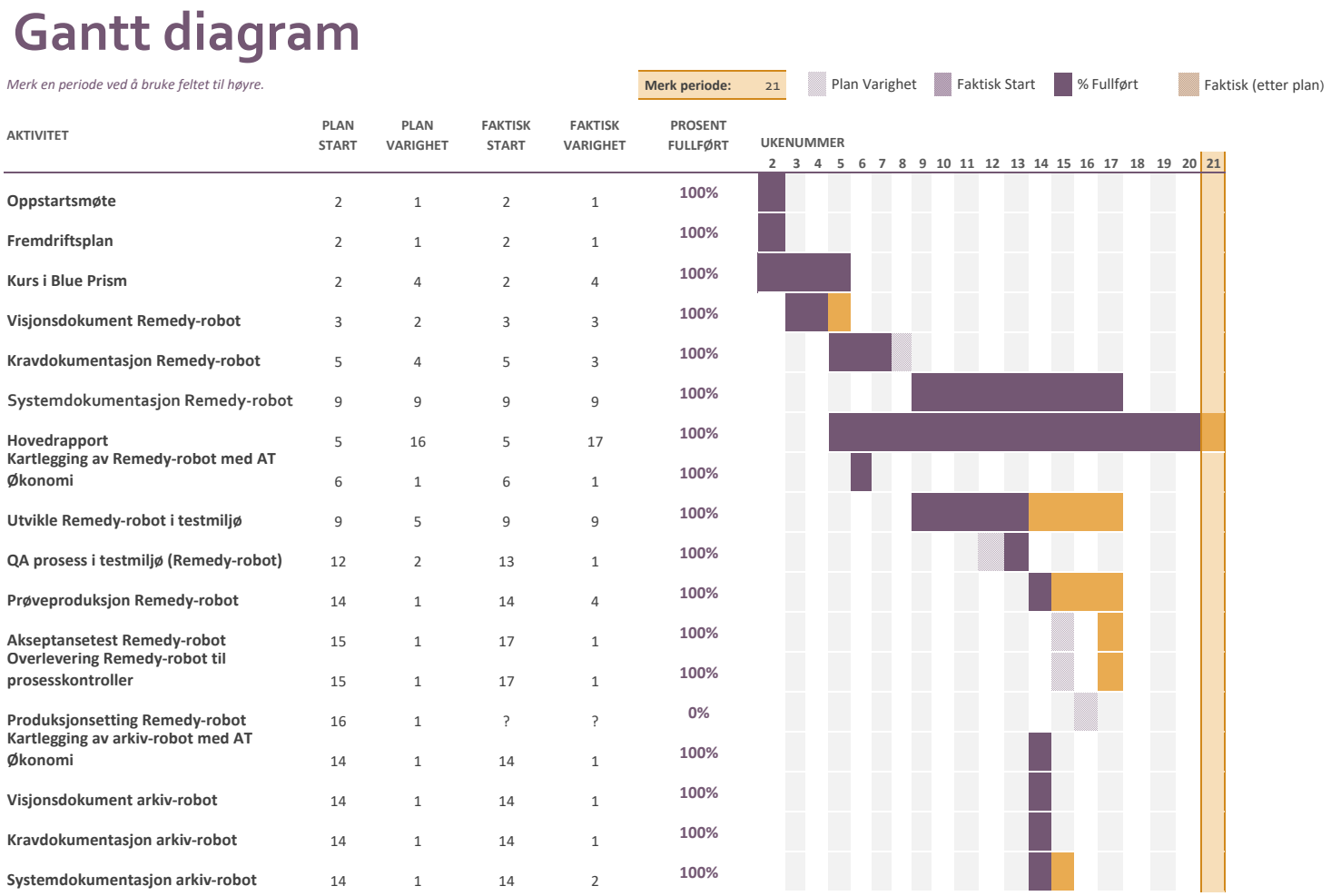
Figur 4.7: Testskript fra QA prosess i testmiljø for Remedy-robot 25.03.19

4.3 Administrative resultater

I dette delkapittelet refereres det til prosjekthåndboken som er vedlagt i innleveringen av bachelooppgaven til NTNU. Prosjekthåndboken inneholder timelister for teammedlemmene under prosjektet, møteinnkallinger og møtereferat, samt Gantt-diagram.

4.3.1 Fremtidsplan

Fremtidsplanen og faktisk varighet på de ulike aktivitetene finnes i figur 4.8 og presenteres via et Gantt-diagram. Diagrammet ble oppdatert hver uke for å holde oversikt over hvilke aktiviteter som til enhver tid skulle prioriteres. De aller fleste aktivitetene ble satt i gang og fullført til planlagt tid, med unntak av et par aktiviteter.



Figur 4.8: Gantt-diagram ved prosjektslutt

Visjonsdokument tok en uke lenger enn planlagt å ferdigstille, mens det tok en uke mindre å ferdigstille kravdokumentasjon (se vedlegg B). QA prosess i testmiljø av Remedy-roboten startet en uke etter planen, men tok også kortere tid enn planlagt å fullføre og dermed var teamet likevel i rute. Samtidig som QA prosess i testmiljø ble ferdig, ble punktene som gjelder arkiv-roboten lagt til i Gantt-diagrammet. Prøveproduksjon og utvikling av Remedy-robot i testmiljø foregikk tre uker lengre enn planlagt. Dermed ble også overlevering til prosesskontroller forskjøvet med likt antall uker. Utvikling av arkiv-roboten ble påbegynt en uke etter planlagt start. Deretter ble den satt på pause i tre uker før det igjen ble påbegynt i uke 18 og godkjent gjennom QA prosess i testmiljø i uke 19.

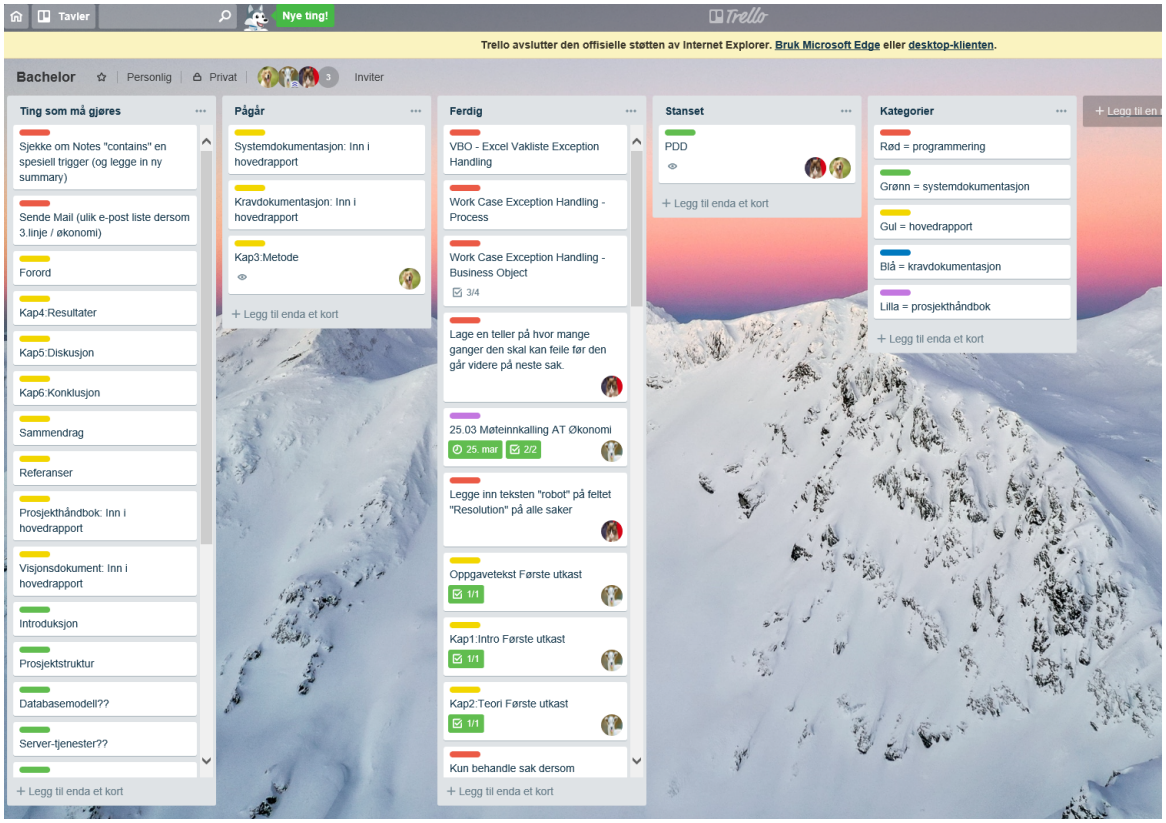
4.3.2 Timeregnskap

Hver uke skrev team-medlemmene timelister og spesifiserte hvilke aktiviteter (gjenspeilet i Gantt-diagrammet) som ble jobbet med. Totalt antall timer brukt på de ulike aktivitetene er vist i tabellen nedenfor:

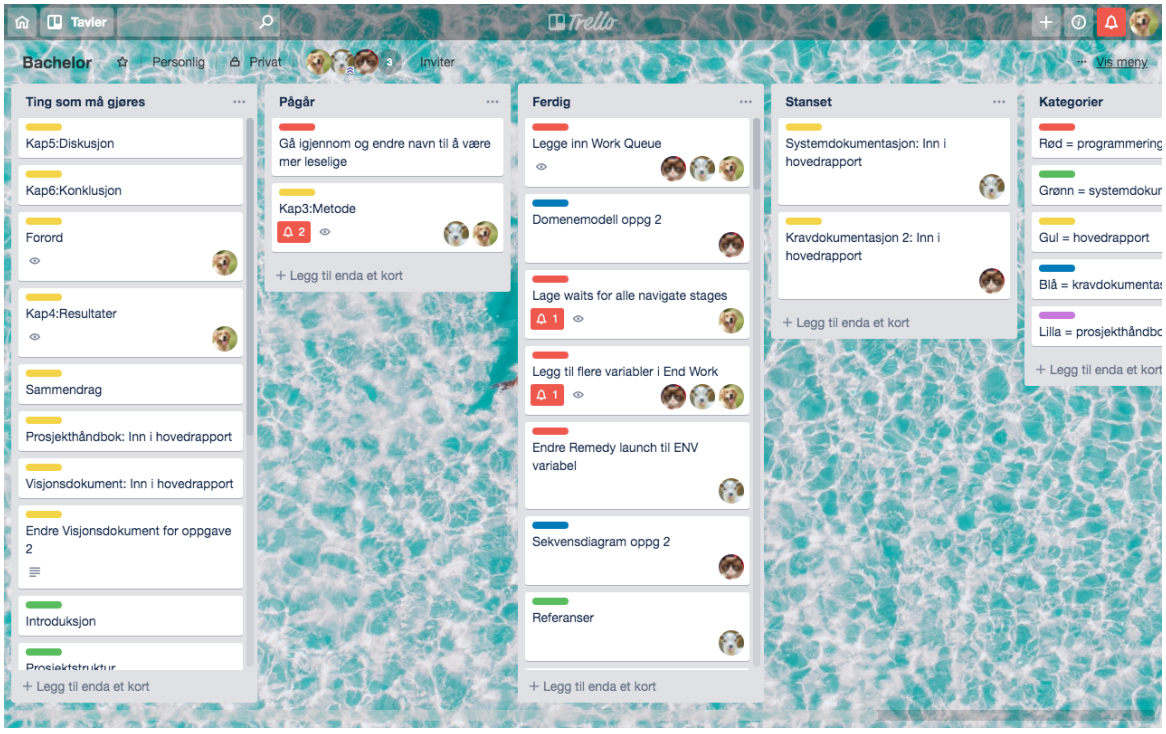
Aktivitet	**Timebruk**
Oppstartsmøte	13,5
Fremdriftsplan	17
Kurs i Blue Prism	151,5
Kartlegging av Remedy-robot AT Økonomi	12
Visjonsdokument Remedy-robot	47,5
Kravdokumentasjon Remedy-robot	29
Systemdokumentasjon Remedy-robot	32,5
Hovedrapport	639.5
Utvikle Remedy-robot i testmiljø	361
Prosjekthåndbok	144
Obligatorisk undervisning	25,5
Prøveproduksjon	25,5
Akseptansetest	3
Kartlegging av arkiv-robot AT Økonomi	4
Visjonsdokument arkiv-robot	8
Kravdokumentasjon arkiv-robot	7
Systemdokumentasjon arkiv-robot	22
Utvikle arkiv-robot i testmiljø	47
QA prosess 2 i testmiljø	1
Totalt	**1560**

4.3.3 Metode

Teamet brukte elementer fra Kanban som utviklingsmetode. Bildene viser hvordan Kanban-tavlen så ut i uke 13 og uke 18. Resultatet var at utviklerne holdt oversikt over hva de andre på teamet arbeidet med til en hver tid, hva som var ferdigstilt og hva som måtte gjøres.



Figur 4.9: Forsiden på Trello i uke 13



Figur 4.10: Forsiden på Trello i uke 18

Kapittel 5

Diskusjon

I dette kapitlet drøftes det hvordan resultatene kan forstås i forhold til problemstillingen og hvorfor resultatene har blitt som de har blitt. Profesjonsetiske retningslinjer

5.1 Vitenskapelige resultater

5.1.1 Produkt

Selve programmeringen som har blitt gjort for å utvikle roboten er først og fremst et resultat av kurset utviklerne gjennomførte i begynnelsen av prosjektet. Kurset inneholdt grunnleggende informasjon om hvordan en robot utvikles i Blue Prism. Her kom studentens programmeringsgrunnlag til nytte. Kunnskapen utviklerne har tilegnet seg gjennom studiet har gitt dem et godt grunnlag for å lære nye språk og teknologier. I tillegg til kurset har utviklerne benyttet Blue Prisms hjemmeside for å finne svar på spørsmål underveis. RPA-teamet i Skatteetaten har også vært behjelpelige når utviklerne har hatt utfordringer de ikke har funnet svar på andre steder.
For at flyten skulle bli best mulig var det viktig for utviklerene å få tilbakemelding på programmeringen. Frem til uke 14 jobbet teamet stort sett på egenhånd, uten særlig tilbakemelding på flyten i prosessen. RPA-teamet var tilgjengelig for å svare på spørsmål per e-post og Skype, men det ble ikke gjennomført kodevurdering før i uke 14.

I uke 14 ble første prøveproduksjon gjennomført. Kunden og utviklerne mente på dette tidspunktet at roboten utførte oppgaven på riktig måte og at den var god nok til å settes i produksjon. Det viste seg derimot at prosessen ikke var utviklet etter beste praksis. Tilbakemeldingene

teamet fikk under første prøveproduksjon var helt nødvendige for å oppnå et kvalitetsprodukt. Selv om utviklerne hadde Blue Prism kurset tilgjengelig viste det seg, at det fortsatt var elementære regler for beste praksis som teamet ikke hadde lært. I ettertid ser utviklerne at det ville vært nyttig med en kodegjennomgang på et tidligere tidspunkt i utviklingsprosessen. Dette for å spare tid for alle involverte.

Tilbakemeldingene som ble gitt av RPA-teamet under første prøveproduksjon var blant annet at arbeidskø må implementeres (mer om dette i neste delkapittel), det må legges inn flere ventesteg og de ulike typene unntak i Blue Prism må håndteres på ulike måter.
Ventestegene fungerer på den måten at roboten arbeider videre så fort feltet den skal samhandle med finnes i skjermbildet. Derfor er det hensiktsmessig å legge inn ventesteg på for eksempel 30 sekunder, sammenlignet med at Remedy restartes hver gang roboten jobber "for fort". Da venter roboten på at nettsiden er lastet inn før den samhandler med felter og knapper. Dersom roboten jobber fortere enn programvaren tillater vil den kunne ligge foran skjermbilde, og ikke finne de elementene den skal samhandle med. Skulle dette skje, vil roboten lukke alle programmene, restarte, og prøve på nytt.
I tillegg ble det tipset om å legge inn flere kriterer for at roboten skal stoppe. Disse kriteriene var maks antall saker, maks antall unntak og tidsfrist for utførelse av arbeidet, alle lagt til som miljøvariabler. Stoppkriteriene gir større kontroll over når roboten ikke skal kjøre og hvordan den skal avslutte arbeidet. Utviklerne hadde også valgt å plassere logikk i VBOene, noe som ikke var anbefalt med tanke på objektorientering og gjenbruk av prosesser. VBOene skal være forbeholdt samhandling og ekstrahering av data, ikke logikk for å prosessere data.

Arbeidskø

Opprinnelig valgte teamet å bruke en tom txt-fil for å ta vare på IDen til saken behandlet sist. Når roboten skulle velge neste sak å behandle, leste den verdien fra txt-filen, og sammenlignet denne med neste sak i Remedy-køen. Roboten ville kun behandle saken dersom den hadde høyere ID enn IDen på txt-filen. På den måten ville den aldri behandle samme sak to ganger, som var et krav fra kunden. Denne løsningen fungerte bra i testmiljøet, men under første prøveproduksjon ble det oppdaget at roboten ville velge feil sak dersom det kom inn en ny sak med lavere ID-nummer underveis i kjøringen.
I tillegg oppfylte ikke denne løsningen kravet til AT Økonomi om tilbakemelding og sporbarhet ved feil. Det ble derfor besluttet under første prøveproduksjon at teamet skulle implementere

en arbeidskø.
Ved utvikling av arkiv-roboten ble arbeidskø implementert fra starten, etter erfaringene som ble gjort med Remedy-roboten.

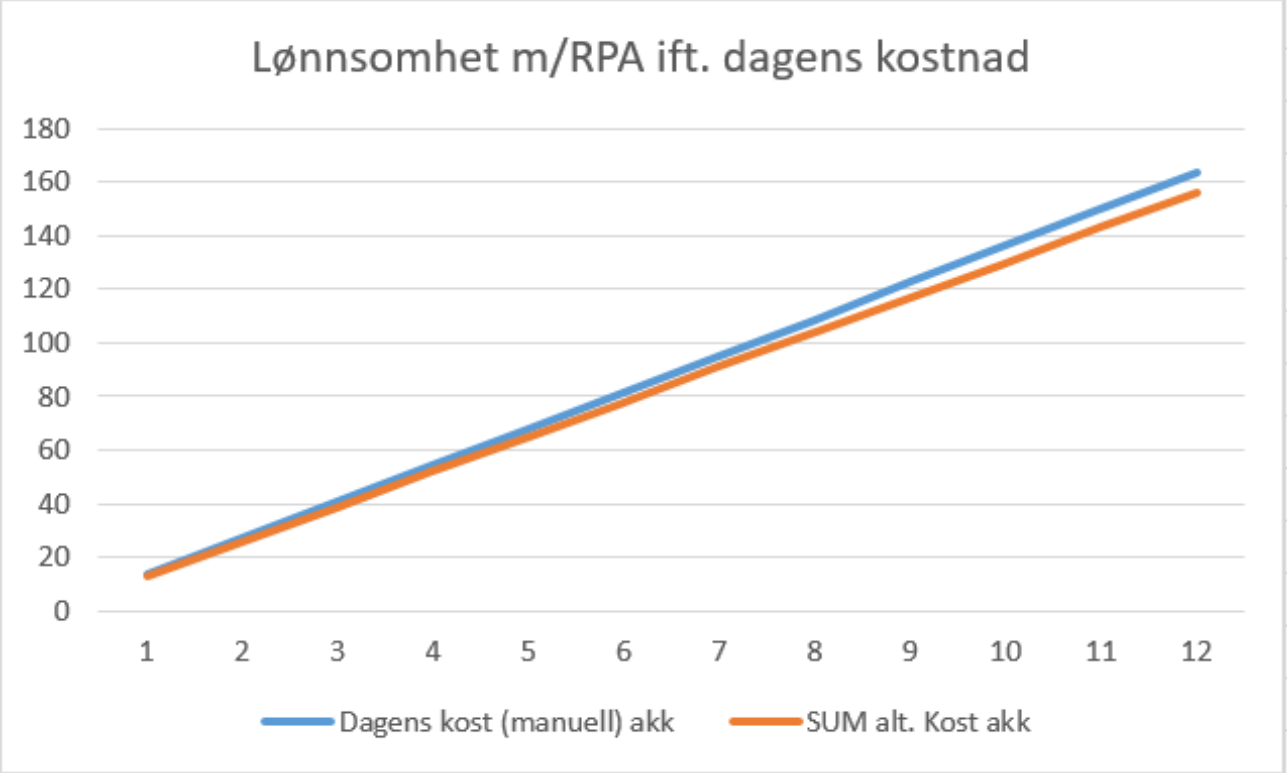
Excel vaktliste

Det var viktig for kunden at Excel-arket ble utformet på en slik måte at det enkelt kan skaleres. Ettersom flere avdelinger i Skatteetaten bruker Remedy, er det en sannsynlighet for at roboten kan brukes av flere. Med dagens løsning med vaktliste i Excel, er dette mulig ved å legge til nye kolonner med triggerord og tilhørende kategorier roboten skal bruke. Det er essensielt at utformingen av Excel-filen ikke endres på for at roboten skal behandle alle sakene. Dersom det er skrivefeil, vil ikke roboten klare å sette navnene lik hverandre, og vil da markere saken som feil og legge ved en kort forklaring på hva som har gått galt i arbeidskøen. I samarbeid med AT Økonomi ble Excel-filen til Remedy-roboten utformet med begrensninger for hva som kan fylles inn i vaktlisten. Navnene på de ansatte som skal inn i timelisten fylles blant annet inn ved å velge fra ferdig utfylte navn fra en nedtrekksmeny. Dette begrenser feil i utformingen av Excel-filen.

5.1.2 Økonomi

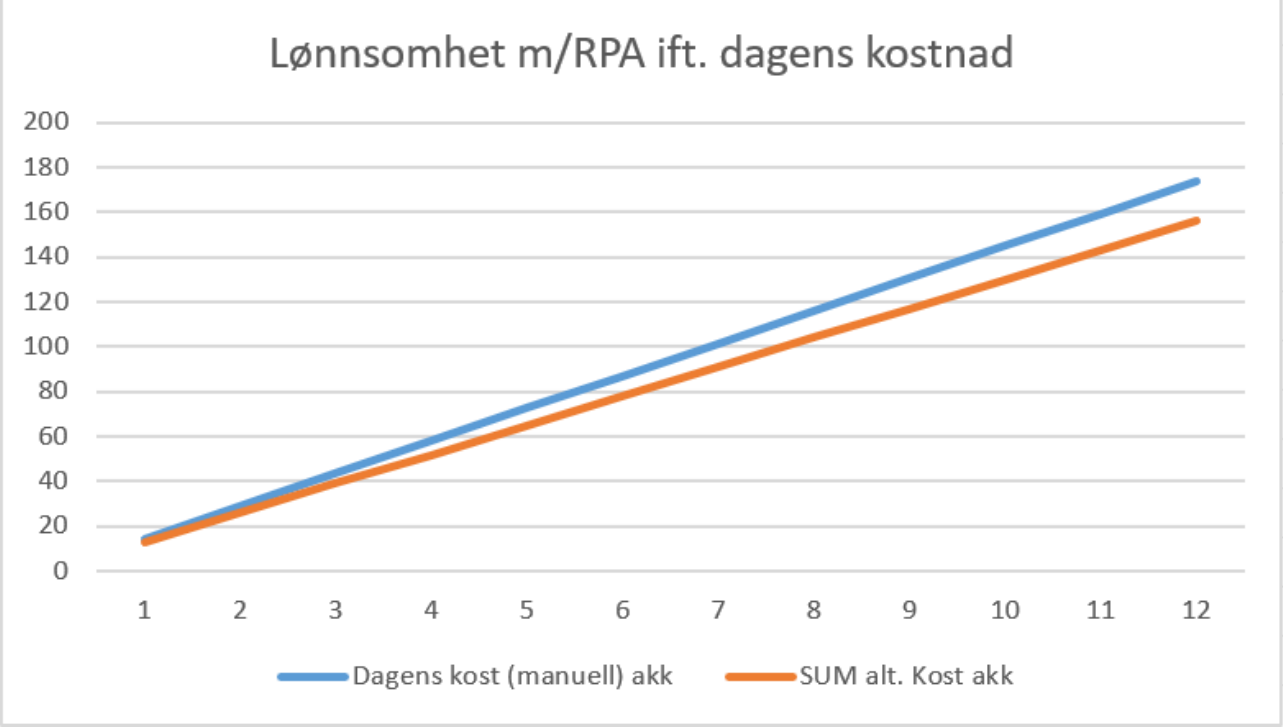
For å kunne svare på problemstillingen ville utviklerne undersøke om oppgaven ville gi økonomisk gevinst og valgte å analysere lønnsomheten til begge robotene. Teamet benyttet seg av samme mal som Skatteetaten bruker ved samme undersøkelse; en Business Case. Dette er en Excel-mal som sammenligner dagens manuelle kostnader med forventede kostnader ved automatisering av oppgaven med RPA-robot. Roboten må gi en økonomisk gevinst innen 12 måneder for at Skatteetaten ser på en robot som lønnsom. Ettersom Skatteetaten ønsket å få studenter til å robotisere en oppgave uavhengig om det var lønnsomt, ble ikke denne undersøkelsen gjort på forhånd.
Resultatet av analysen viste at hverken Remedy-roboten og arkiv-roboten ville være lønnsomme (se figur 4.5 og 4.6). Kostnadene for utviklerlisens, servere, vedlikehold og drift blir høyere enn å utføre oppgaven manuelt. Årsakene til dette er at det er for få saker som behandles i året eller at oppgavene er for små. Manuell behandlingstid av oppgavene var målt til tre og fem minutter. Ifølge Telefónicas modell over saker som egner seg for RPA-automasjon (figur 2.4), bør det være rundt 1000 saker i uken hvis saken tar under fire minutter å utføre manuelt. Med

dette i betraktning, ville teamet undersøke når RPA-robotene ville bli økonomisk lønnsomme dersom det var et høyere antall saker. Ved å tredoble antall saker til Remedy-roboten til 6600 saker i året, fikk teamet dette resultatet:



Figur 5.1: Beregning av økonomisk lønnsomhet for Remedy-robot med 6600 saker

Figuren viser at roboten vil bli lønnsom etter en måned. Hvis man antar at en seksjon i Skatteetaten løser 2 200 saker i året slik AT Økonomi gjør, vil det være økonomisk gunstig dersom roboten Remedy-roboten blir skalert til å ta hånd om saker fra minst tre seksjoner (gitt at roboten kjører på samme maskin og lisens). Da vil kostnadene for RPA-roboten ligge på rundt 156 000 kroner etter ett år, som er 10 000 kroner rimeligere enn å utføre oppgavene manuelt. Den samme undersøkelsen ble gjort på arkiv-roboten, og ved å øke antall oppgaver til 4500 fikk teamet følgende graf:



Figur 5.2: Beregning av økonomisk lønnsomhet for arkiv-robot med 4500 saker

I begge tilfellene ser vi at ved å øke antall oppgavesom robotene skal behandle vil man kunne oppnå en økonomisk gevinst. Fokuset på skalering, som teamet har hatt igjennom prosjektet, har dermed vist seg å kunne være en avgjørende faktor for om robotene som ble utviklet vil ende opp med å bli lønnsomme.

5.1.3 Mål

1. **Løser oppgaven raskere enn et menneske**

   Årsaken til at begge robotene jobber raskere enn et menneske er først og fremst at roboten allerede klikker seg rundt i skjermbilder mye raskere enn et menneske klarer. I tillegg gjør de innebygde metodene for å samhandle med Excel og Outlook at programmene ikke behøver å åpnes på samme måte som et menneske ville gjort. Ved hjelp av disse metodene jobber roboten like fort som en tradisjonell kodesnutt med løkker, kalkulasjoner og tabelloppslag. Prosessflyten er også optimalisert med tanke å løse oppgaven på en rask og effektiv måte. Blant annet blir alle verdiene fra Excel-filen lest inn i Blue Prims egne tabeller i begynnelsen av kjøringen, for å unngå å åpne Excel flere ganger for å sjekke sak mot saksbehandler.

Remedy-roboten jobber saktere enn arkiv-roboten grunnet tregheter i Internet Explorer og Remedy. Dette løses ved hjelp av ventesteg, som nevnt i 5.1.1.

2. **Løser oppgaven mer nøyaktig enn et menneske**
   Et menneske vil kunne gjøre egne vurderinger og tolkninger ved behandling av saker, og det er dermed ikke gitt at samme input vil gi samme resultat 10 av 10 ganger. Roboten gjør derimot ikke egne vurderinger, men følger en fast prosessflyt og behandler saker på samme grunnlag hver gang. Roboten vil arbeide på samme måte uansett tid på døgnet uten å bli sulten, sliten eller trøtt. For eksempel vil roboten bestandig lese av rute F3 i Excel dersom det er dette den har fått beskjed om. Utfordringen er at man som utvikler er nødt til å tenke gjennom og sikre at alle scenarier håndteres for at roboten skal arbeide på en korrekt måte. For eksempel vil ikke en robot vite at St.Petersburg og Sankt Petersburg er det samme dersom den ikke har blitt fortalt det.

3. **Gir en robust løsning**
   I samarbeid med kunden, skal utviklerne sørge for at roboten håndterer alle mulige situasjoner som kan oppstå. Her er det stort fokus på popup-vinduer som kan dukke opp under kjøring. RPA-robotene i dette prosjektet benytter ikke maskinlæring, og vil dermed gjøre samme feil gang på gang dersom den ikke har blitt programmert til å håndtere hendelsen. For å finne frem til alle mulige situasjoner som kan oppstå brukte utviklerne egen erfaring med Remedy, og innspill fra kunden som forklarte hvilke situasjoner de har opplevd. Dersom roboten er programmert til å håndtere alle mulige utfall vil roboten arbeide på en selvstendig måte, ettersom det ikke krever menneskelig innblanding. Skulle det oppstå noe roboten ikke er programmert til å håndtere skal den markere saken som ikke behandlet i arbeidskøen, restarte Internet Explorer og Remedy, og fortsette til neste sak. Det var viktig å sikre at roboten aldri vil havne i en evig løkke, og dermed er også en maksgrense for antall unntak og antall saker behandlet stoppkriterier.

4. **Gir høyere kvalitet**
   Roboten vil øke kvaliteten på tjenestene ved å minimere manuelle inngrep, feil og dobbeltarbeid. Roboten vil ikke lese av feil felt i Excel-arket, skrive inn feil verdier, tolke tekst eller klikke på feil knapper. Utviklerne har full kontroll over robotens handlinger og hvilke valg den vil ta underveis. I praksis påvirkes robotens kvalitet også av kvaliteten på Excel-arket og stabiliteten i Remedy.

Ulempen oppstår dersom det skulle være feil i vaktlisten. Et menneske vil kunne rette på feilen underveis, og vite at det for eksempel er skrevet feil person på feil dag. Dette vil ikke roboten ta hensyn til, og dermed tildele saken til den personen som står under riktig kategori. Derfor er det viktig at den er programmert til å behandle alle typer saker som kan oppstå.

5. **Samhandler med eksisterende systemer**

   For å få roboten til å kjenne igjen elementer i Remedy brukes IE HTML Mode som spioneringsverktøy. Først valgte utviklerne å bruke ulike moduser alt ettersom hva som så ut til å fungere. Dette viste seg fort å føre til ustabiliteter, og utviklerne måtte skaffe seg bedre kunnskap om de ulike modusene. I løpet av prosjektet var Remedy nede en kort periode. Ved neste kjøring kjente ikke roboten igjen feltene som hadde blitt spionert. Denne hendelsen viste en svakhet hos roboten, som var kritisk og viktig å finne en løsning på. Utviklerne fikk tips av RPA-teamet om å fjerne flere avkrysninger i boksene som bestemmer hvordan et felt skal gjenkjennes (se figur 3.3). Som eksempel kan URLen til et felt endre seg, og egner seg derfor ikke som attributt for gjenkjenning av HTML-elementer. Etter å ha huket av riktige attributter løste problemet med ustabilitet seg.

   Teamet lærte etter denne hendelsen flere nyttige måter å bruke spioneringsverktøyet på. En av disse var å velge HTML-elementer med indeks for å bevege seg opp eller ned i en tabell. Tidligere brukte utviklerne tastetrykk som var mer ustabilt. En bedre løsning var dynamisk HTML-sti som endrer indeksen i tabellen ut i fra hvilket element man ønsker. Utviklerne lærte også at sammenligningstypen "Wildcard" er nyttig i tilfeller der for eksempel deler verdien til et felt er fast, men andre deler varierer. Et eksempel er dersom man skal finne et element kalt ElementX, og man godtar elementet uansett hvilken verdi X har.

   Andre viktige avgjørelser som ble gjort for å få roboten til å samhandle godt med Remedy var å bruke ventesteg. I noen tilfeller var det også nødvendig med "Dummy waits", som ikke venter på et bestemt felt, men venter et gitt antall sekunder før roboten fortsetter. Grunnen til dette var at roboten i noen tilfeller fant HTML-elementer før siden var fullstendig lastet inn.

6. **Løsningen er 80-100% automatisert**

   En robot er 100% automatisert dersom alle relevante saker behandles av roboten uten menneskelig innblanding. Dette krever at Excel-arket innholder alle mulige kombinasjoner

av triggerord. Selv om robotene er 100% automatiserte etter gjennomkjøring av testsaker, kan det oppstå utfordringer når roboten er produksjonssatt som ikke har kommet frem under testing. Skatteetatens RPA-team vet av erfaring at roboter sjelden er 100% automatisert. I realiteten oppnår man ofte autmasjonsgrad på 45-85%. Det er derfor mer realistisk å anta at automasjonsgraden er cirka 80%. Dersom teamet hadde hatt mulighet til å følge opp roboten etter produksjonssetting og rettet på årsaken til at eventuelle feil oppstår, ville man etterhvert kunne se en høyere automasjonsgrad.

7. **Sporbarhet ved feil**

   Arbeidskøen gjør det enkelt for prosesskontrolleren, utviklerne og kunden å spore feil. Etter hver kjøring i produksjon mottar kunden en rapport av loggen fra arbeidskøen, slik at de kan følge med på hvor mange saker den den fullfører og feiler. Ved spesielle unntak legges det i tillegg ved en kommentar som beskriver hvilken feil som har oppstått og hvilke tiltak kunden må gjøre for å unngå at samme feil skjer igjen.

   Prosesskontroller vil også ha tilgang til en mer detaljert logg fra hver kjøring. En slik logg viser nøyaktig hvilke beslutninger som til en hver tid er tatt i programmet. Denne loggen er av lite nytte for kunden, men er viktig for utviklerene under testing for å kunne spore feil som har oppstått i koden eller Blue Prism.

8. **Gir økonomisk gevinst**

   Årsaken til at dette målet ikke ble oppnådd er at utviklerne ikke gjorde en grundig teoretisk vurdering av om oppgaven ville være lønnsom å automatisere i forkant av utviklingen. Dersom utviklerne hadde gjort denne analysen ville de sett at automatisering av oppgavene ikke vil være lønnsomt, og ville derfor ikke satt det som et mål i visjonsdokumentet. Under 5.1.2 diskuteres årsaken til at robotene ikke er lønnsomme.

5.1.4 Testing

Testing mot testmiljøet i Remedy ble utført kontinuerlig gjennom hele utviklingsprosessen. Også når prosessen blir kjørt i debug-modus, vil den kjøre gjennom prosessen i testmiljøet i Remedy slik at roboten ikke behandler reelle saker under test. Det at kunden kunne opprette reelle testsaker var også nødvendig for utviklerene, slik at de kunne programmere roboten på best mulig vis. For at det skal være hensiktsmessig å sette opp et testmiljø, er det avgjørende at testmiljøet er så likt produksjonsmiljøet som mulig. Den eneste forskjellen mellom test - og

produksjonmiljøet i Remedy var at produksjonsmiljøet bruker SSO, der det i testmiljøet måtte logges inn med brukernavn og passord. På grunn av at miljøene var såpass like, var det ikke krevende å klargjøre roboten til prøveproduksjon.

5.2 Administrative resultater

5.2.1 Framtidsplan

Gantt-diagrammet som ble oppdatert hver uke var et nyttig diagram for utviklerne, både for å ha en plan på hva som skulle være ferdig til hvilken tid, og for å vise veileder og RPA-teamet status på prosjektet. Prøveproduksjonen er den aktiviteten som viste seg å ta lengst tid i forhold til hva som var planlagt. I tillegg havnet prøveproduksjonen rundt påske, som førte til at det tok flere uker enn det hadde vært behov for. I uke 14 tok teamet sammen med RPA-teamet, AT Økonomi og veileder beslutningen om å utvikle en sekundærrobot. Denne beslutningen ble ikke tatt tidligere fordi det var vanskelig å forutse hvor tidkrevende det var for teamet å utvikle Remedy-roboten. Ettersom teamet på dette tidspunktet kun var omtrent halvveis i antall timer, var det en bra løsning å utvikle en robot til.

5.2.2 Timeregnskap

Teamet brukte vesentlig kortere tid på utvikling av arkiv-roboten i forhold til Remedy-roboten. Det er flere årsaker til dette, blant annet at arkiv-roboten er mindre kompleks enn Remedy-roboten. Den krever mindre samhandling med andre programmer og bruker blant annet ikke spioneringsverktøyet i Blue Prism. Arkiv-roboten samhandler kun med Outlook og Excel. Erfaringen utviklerne har fått gjennom programmeringen av Remedy-roboten spilte også en stor rolle i den kraftige reduksjonen av tiden. Evnen til å tenke prosess umiddelbart, vite om eksisterende metoder, og hva som er beste praksis er viktige faktorer for hvor lang tid det vil ta å utvikle en robot. Dokumentasjonen for arkiv-roboten tok også kortere tid å skrive ettersom mye var likt for begge robotene, og teamet kunne gjenbruke eksisterende kode.

5.2.3 Metode

Ettersom ingen av teammedlemmene hadde erfaring med RPA-utvikling, var det utfordrende å anta hvor lang tid oppgavene ville ta og hva som var mulig å utrette med teknologien som var

tilgjengelig på forhånd. Derfor var det nødvendig å arbeide iterativt i utviklingsfasen for å ha muligheten til å endre for eksempel oppgavens krav.

Elementene fra Kanban som teamet valgte å bruke som utviklingsmetode bidro til å ha struktur og oversikt over arbeidsoppgavene. Teamet kunne kommet tidligere i gang med bruken av Kanban-tavle, men så ikke nytten av det helt i begynnelsen, fordi det ikke var mange oppgaver å fordele, i tillegg til at arbeidsfordelingen foregikk muntlig på kontoret. Ettersom teamet fant det utfordrene å dele opp oppgavene i mindre deloppgaver i begynnelsen, ble ikke Kanban-tavlen i Trello tatt i bruk før utviklingen av roboten startet. Årsaken til at dette ikke ble gjort, var at utviklerne arbeidet på samme kontor til samme tid hver eneste dag. Dermed foregikk mye av kommunikasjonen muntlig, og det var ikke behov for å detaljerte arbeidsoppgaver på Kanban-tavlen. Underveis i prosjektet ble Trello oppdatert omtrent daglig.

Hver uke hadde teamet statusmøte med vår hovedkontaktperson fra RPA-teamet. Dette bidro til at teamet sjelden stod fast ved problemer, og at både RPA-teamet og utviklere var oppdatert på prosjektet status. Utviklerne hadde stort sett spørsmål hver eneste uke, som enten ble besvart under møtet eller på mail etter møtet. Selv om det i begynnelsen av prosjektet følte eksessivt for utviklerne å ha møte hver eneste uke, viste det seg å være en god løsning for begge.

Når det kommer til arbeidsdeling ble det fort enighet om at to personer skulle jobbe med utvikling og en med hovedrapport. Dette både for å komme i gang med hovedrapport tidlig og arbeide med den parallelt, men også fordi det kun er én person som kan redigere inne i en prosess eller VBO av gangen. Dette viste seg å fungere godt, og mye av tiden ble brukt på denne måten. Videre delte de to som programmerte ofte inn slik at en person jobbet i prosessen og en i et VBO.
Blue Prisms begrensning av antall utviklere i en prosess førte også til deler av utviklingen foregikk med parprogrammering, spesielt på slutten der mye av arbeidet foregikk i hovedprosessen. Det var i tilegg hjelpsomt når man skulle programmere mer komplisert logikk, fordi det var praktisk å forklare hverandre hva man tenkte og se ting fra ulike perspektiver.

5.2.4 Gruppearbeid

Teammedlemmene kjente hverandre godt før prosjektets start og hadde dermed ikke et stort fokus på teambuilding. I starten av prosjektet utarbeidet teamet en arbeidskontrakt som la grunnmuren for gruppedynamikken. Dette gjorde at teamet tidlig ble enige om mål og forventninger, som har vært avgjørende for å holde fokus og jobbe målrettet. Det har vært lav terskel for å komme med ideer og teammedlemmene har opprettholdt god kommunikasjon gjennom hele prosjektet. Dersom prosjektet skulle vært gjennomført på nytt ville teamet valgt samme metode og arbeids- og rolle fordeling. Alle har fått mulighet til å arbeide med forskjellige oppgaver, både innenfor programmering og dokumentskriving. Arbeidet har blitt fordelt på en rettferdig og effektiv måte, og alle har fått erfaring med de ulike oppgavene. Dette gjenspeiles også i timelistene. Møteinnkallingene har stort sett blitt skrevet av Kimia, referatene av Camilla og timelistene av Nora. Dette har fungert bra og gjort at alle har vært bevisst på hva de har ansvar for og gjort eller fulgt opp det som må gjøres.

Kapittel 6

Konklusjon og videre arbeid

Basert på resultatene er det hensiktsmessig å implementere RPA-roboter for å utføre manuelle oppgaver dersom visse forutsetninger er tilstede. Både Remedy og arkiv-roboten løste oppgavene adskillig raskere enn et menneske, og med høy grad av automasjon. Samhandlingen med eksisterende systemer foregikk på samme måte som for en ansatt, uten å forstyrre underliggende datasystem eller infrastruktur. Kvaliteten er jevnere og oppgaven *kan* utføres mer nøyaktig. Ingen av robotene ga økonomisk gevinst for Skatteetaten, da antall oppgaver per år var for få til at det var lønnsomt å utvikle og implementere en RPA-robot.

Valg av prosess for automatisering og antall oppgaver ble oppdaget som avgjørende forutsetninger for at det skal være hensiktsmessig med RPA-teknologi. RPA er per dags dato best egnet for standardiserte, regelbaserte og repeterende oppgaver, en påstand som bekreftes gjennom dette prosjektet. For at det skal være hensiktsmessig å automatisere enkle prosesser der saksbehandlere vanligvis bruker kort tid per oppgave, må det være et stort volum av oppgaver som skal behandles. Skalerbarhet er dermed viktig for at robotene som er utviklet i prosjektet skal være lønnsomme i det lengre løp.

Det er også flere faktorer som spiller inn for om implementering av robot i et eksisterende system blir vellykket. Grundig opplæring i Blue Prism, og jevn kontakt med erfarne konsulenter, muliggjør utvikling av et produkt som følger beste praksis, som er enkelt å drifte og som lever opp til Skatteetatens krav for utforming av RPA-robot.
Gjennom prosjektet har utviklerne erfart betydningen av god kommunikasjon med kunden og RPA-team for implementasjonen av robot. Et godt samarbeid med saksbehandlerne som

vanligvis utfører oppgaven manuelt er avgjørende, ettersom utviklerne er avhengig av forstå stegene i prosessen som skal automatiseres. Likså kan det være viktig for saksbehandlerne å forstå robotens muligheter og begrensninger, for at kunden og utviklere sammen kan komme frem til de aller beste løsningene.
Det er viktig å tenke på at produksjonsmiljøet kan se og oppføre seg ulikt testmiljø, og det er lurt å være forberedt på mye nytt arbeid etter første prøveproduksjon. En annen viktig faktor for vellykket implementasjon, er at roboten får tilganger og tillatelser til å utføre arbeid på lik linje med en ansatt. I store selskap kan dette ta lang tid og må planlegges godt i forveien.

Videre arbeid vil være å produksjonssette robotene, og observere de i produksjon slik at det er mulig å rette opp eventuelle feil som oppstår og komme med forbedringer. Som Xchanging erfarte i deres prosjekt med RPA, er kontinuerlig forbedring essensielt for å sikre at robotene holder høy kvalitet etter selve implementeringen. Deretter er det naturlig å foreta en dypere analyse av effektene ved innføring av RPA etter at robotene har vært i produksjon over en lengre periode. Det vil også være interessant å se om løsningen kan skaleres til flere avdelinger og om det i praksis vil føre til at robotene blir lønnsomme, slik vi har konkludert.

Referanser

[1] S. Roger, "Hvorfor digitaliseringsminister?" IKT-Norge, 2019, [Accessed: 20.03.19]. [Online]. Available: https://www.ikt-norge.no/nyheter/hvorfor-digitaliseringsminister/

[2] J. T. Sanner, "Vekst og arbeidsplasser i teknologiens tidsalder," Den Norske regjeringen, 2017, [Accessed: 26.02.19]. [Online]. Available: https://www.regjeringen.no/no/aktuelt/vekst-og-arbeidsplasser-i-teknologiens-tidsalder/id2536213/

[3] K. og moderniseringsdepartementet, "Helhet, kvalitet og effektivitet," Den Norske regjeringen, 2018, [Accessed: 23.04.19]. [Online]. Available: https://www.regjeringen.no/contentassets/e6079ad76a1041578d57561128aa07f8/helhet_kvalitet_effektivitet.pdf

[4] "Samfunnsoppdrag og strategi," Skatteetaten, 2018, [Accessed: 26.02.19]. [Online]. Available: https://www.skatteetaten.no/om-skatteetaten/om-oss/samfunnsoppdrag-strategi/

[5] P. Strøm and J. Resvoll, "I dag er det slutt på dette – hvis du ikke reserverer deg," NRK, 2014, [Accessed: 26.02.19]. [Online]. Available: https://www.nrk.no/troms/slutt-pa-skatteoppgjoret-i-posten-1.11767246

[6] "Økonomi," Skatteetaten, hentet fra Skatteetatens intranett.

[7] L. P. Willcocks, M. Lacity, and A. Craig, "The IT function and Robotic Process Automation," 2015.

[8] M. ALBERTH and M. MATTERN, "Understanding robotic process automation (RPA)," *AUTOMATION*, p. 54, 2017.

[9] W. M. van der Aalst, M. Bichler, and A. Heinzl, "Robotic process automation," 2018.

[10] P. B. Andersen, "Automatisering," Store norske leksikon, 2018, [Accessed: 23.04.19]. [Online]. Available: https://snl.no/automatisering

[11] J. Varis, "Automating processes in web-interfaces with Robotic Process Automation," 2018.

[12] "Hva er Robotisk prosessautomatisering (RPA)?" Deloitte, [Accessed: 12.05.19]. [Online]. Available: https://www2.deloitte.com/no/no/pages/technology/articles/hva-er-rpa-.html

[13] N. Ostdick, "The Evolution of Robotic Process Automation (RPA): Past, Present, and Future," UIPath, 2016, [Accessed: 26.02.19]. [Online]. Available: https://www.uipath.com/blog/the-evolution-of-rpa-past-present-and-future

[14] D. Glez-Peña, A. Lourenço, H. López-Fernández, M. Reboiro-Jato, and F. Fdez-Riverola, "Web scraping technologies in an API world," *Briefings in bioinformatics*, vol. 15, no. 5, pp. 788–797, 2013.

[15] E. Vargiu and M. Urru, "Exploiting web scraping in a collaborative filtering-based approach to web advertising." *Artif. Intell. Research*, vol. 2, no. 1, pp. 44–54, 2013.

[16] "Workflow," Merriam Webster, 2019, [Accessed: 05.03.19]. [Online]. Available: https://www.merriam-webster.com/dictionary/workflow

[17] C. Le Clair, W. McKeon-White, and D. Lynch, "The Forrester Wave™: Robotic Process Automation, Q2 2018," The Forrester Wave, 2018, [Accessed: 07.05.19]. [Online]. Available: https://samfundsdesign.dk/siteassets/media/downloads/pdf/the_forrester_wave_rpa_2018_uipath_rpa_leader.pdf?fbclid=IwAR1Ha691Hr7tKPlcnS_QxIJMS5BEkceI1S3BTFw6G7iq3RT7DAgUoSg5TRo

[18] "Meet the Team," Blue Prism Group, 2019, [Accessed: 21.03.19]. [Online]. Available: https://www.blueprism.com/who-we-are/team

[19] "Automation Anywhere," Crunchbase, April 2018, [Accessed: 29.04.19]. [Online]. Available: https://www.crunchbase.com/organization/automation-anywhere

[20] "What is the Digital Workforce?" Automation Anywhere Inc., 2019, [Accessed: 29.04.19]. [Online]. Available: https://www.automationanywhere.com/digital-workforce

[21] C. Le Clair, A. Cullen, and M. King, "The Forrester Wave™: Robotic Process Automation, Q1 2017," The Forrester Wave, 2017, [Accessed: 07.05.19]. [Online]. Available: http://www.bluvaultsolutions.com/wp-content/uploads/2017/11/Robotics.pdf

[22] J. Watson and D. Wright, "The robots are ready. Are you?" Deloitte, 2017, [Accessed: 12.05.19]. [Online]. Available: https://www2.deloitte.com/cn/en/pages/strategy-operations/articles/the-robots-are-ready.html

[23] M. Lacity, L. P. Willcocks, and A. Craig, "Robotic Process Automation at Telefonica O2," 2015, the London School of Economics and Political Science.

[24] "Market share held by mobile operators in the United Kingdom (UK) 2018, by subscriber," Statista, 2018, [Accessed: 02.05.19]. [Online]. Available: https://www.statista.com/statistics/375986/market-share-held-by-mobile-phone-operators-united-kingdom-uk/

[25] "About us," Xchanging, [Accessed: 03.05.19]. [Online]. Available: http://www.xchanging.com/about-us

[26] L. P. Willcocks, M. Lacity, and A. Craig, "Robotic Process Automation at Xchanging," 2015, The London School of Economics and Political Science.

[27] M. Azim, "A Hybrid Agile approach to delivering RPA," Deloitte, April 2018, [Accessed: 01.04.19]. [Online]. Available: http://blog.deloitte.com.au/hybrid-agile-approach-delivering-rpa/?fbclid=IwAR3QNWjOyibNCcEDKVmIEbOTO49RHLNxbcjNT66UuilTp2Lag89fg2DvnT4

[28] N. Tesdal, "Agil utvikling," 2018, Institutt for datateknologi og informatikk, NTNU.

[29] D. Radigan, "What is Kanban?" Atlassian, [Accessed: 07.05.19]. [Online]. Available: Tilgjengeligfra:https://www.atlassian.com/agile/kanban

[30] R. M.-C. Bobby Hellard, "What is an application server?" ITPro UK, Desember 2018, [Accessed: 10.05.19]. [Online]. Available: https://www.itpro.co.uk/strategy/29643/what-is-an-application-server

[31] K. for Skatteetaten, "Robotic Operating Environment," 2018.

[32] "How do I spy my application in Blue Prism?" Blue Prism Group, 2017, [Accessed: 15.05.19]. [Online]. Available: https://portal.blueprism.com/customer-support/support-center#/path/Automation-Design/Application-Integration/Active-Accessibility-Spy-Mode/1195399032/How-do-I-spy-my-application-in-Blue-Prism.htm

Vedlegg A

Visjonsdokument for Remedy-robot

Innhold

A.1 Innledning

Hensikten med visjonsdokumentet er å få en oversikt over overordnede mål og krav for prosjektet, samt analysere interessentene og deres behov. I tillegg fokuseres det på hvorforbehovene finnes, slik at de kan oppfylles på best mulig vis. RPA roboten som skal utvikles er ønsket og eid av seksjon for Økonomi i Skatteetaten avdeling Administrative Tjenester. Utviklerne er studentene Kimia Abtahi, Camilla Haaheim Larsen og Nora Othilie Ringdal. Prosjektet er gitt som bacheloroppgave for tredje klasse Dataingeniør ved NTNU, Trondheim.

A.2 Sammendrag problem og produkt

A.2.1 Problemsammendrag

Problem med	tidkrevende og repeterende fordeling av økonomiske dokumenter
berører	ansatte i seksjon for økonomi i Skatteetaten
der resultatet av dette er	ineffektivitet og variert kvalitet.
En vellykket løsning vil	arbeide raskere og mer effektivt, samt håndtere mulige utfall selvstendig og gi jevn kvalitet.

A.2.2 Produktsammendrag

For	seksjon for økonomi i Skatteetaten
som	har behov for automatisering av oppgaver som i dag utføres manuelt.
RPA Roboten	er en virtuell robot
som	selvstendig løser oppgaver raskere og mer nøyaktig enn et menneske, ved hjelp av skjermbilder
I motsetning til	ansatte som vanligvis gjør arbeidet manuelt
har vårt produkt	mulighet til å jobbe døgnet rundt, minimere feil og arbeide raskere.

A.3 Overordnet beskrivelse av interessenter og brukere

A.3.1 Oppsummering interessenter

Navn	Utdypende beskrivelse	Rolle under utviklingen
Oppdragsgiver Skatteetaten	Representert av en ansatt i Skatteetaten. Har rolle som prosjekteier og sponsor.	- Finansiere prosjektet
RPA-team Skatteetaten	Representert av tre ansatte i IT-avdelingen i Skatteetaten. Har rolle som RPA-eksperter.	- Veilede utvikling av robot. Fra kurs til ferdig produkt - Være med på å godkjenne produktet - Produksjonssetting av robot
Veileder NTNU	Veileder representerer NTNU. Skal gi utviklerne tilbakemelding på dokumenter og fremdrift underveis i prosjektet.	- Veilede utviklerne gjennom utviklingsprosessen - Gi tilbakemelding på dokumentasjon - Gjøre en totalvurdering av prosjektgjennomføring og sluttleveranser - Sette sluttkarakter
Utviklere	Studenter ved NTNU. Har ansvar for å utvikle og implementere roboten.	- Sikre at kravene for produktet blir møtt. - Sikre god kommunikasjon mellom alle interessenter - Utarbeide vitenskapelig rapport ut fra prosjektet.
Kunde Skatteetaten	Ansatte ved Administrative Tjenester seksjon Økonomi representerer kunden i prosjektet. De har behov for produktet og skal benytte seg av det når det er ferdigstilt.	- Videreformidle informasjon om det ønskede produkt slik at det kan tilpasses etter deres behov. - Beskrive hvordan oppgaven som roboten skal utføre, gjøres i dag. - Sette kravene for produktet. - Utarbeide tester på varians og volum. - Godkjenne sluttproduktet.

A.3.2 Oppsummering brukere

Navn	Utdypende beskrivelse	Rolle under utviklingen	Representert av
AT Økonomi	Det er sakene ved denne seksjonen roboten vil jobbe med. De ansatte vil dermed få tildelt saker av roboten som de videre skal behandle.	Er med på kartlegging av oppgaven som må beskrives i detalj ved hjelp av skjermbilder. De vil også ha ansvar for å godkjenne QA prosess i testmiljø og prøveproduksjon før produktet produksjonssettes. De har også ansvar for å holde Excel-arket oppdatert.	Ansatte hos AT Økonomi i Skatteetaten.

A.3.3 Brukermiljøet

Det skal utvikles en RPA robot som skal erstatte de ansatte i seksjon for økonomi sitt behov for å tildele interne saker til ansatte for saksbehandling. Roboten skal passe inn i et brukermiljø der det per dags dato er mennesker som utfører oppgaven. Ettersom roboten skal erstatte mennesker, må den samhandle med de eksisterende systemene. Den vil løse oppgavene på samme måte som de ansatte, men ved hjelp av skjermbilder. Roboten skal logge seg inn i systemet Remedy, åpne opp saken og lese vaktlister fra Microsoft Excel for å finne en ansatt på vakt som den kan tildele saken til. Dersom saken er en 3.linje sak, skal det sendes en e-post til alle ansatte som er oppført i vaktlisten via Microsoft Outlook.

Når roboten er tatt i bruk, vil det være de ansatte i seksjon økonomi som varsler prosesskontroller når det er behov for å kjøre roboten. Prosesskontroller har ansvar for den produksjonssatte roboten, og må igangsette denne ved behov.

A.3.4 Sammendrag av brukernes behov

Behov	Prioritet	Påvirker	Dagens løsning	Foreslått løsning
Høyest mulig grad av automasjon	Høy	Økt kapasitet	Ansatte i Skatteetaten utfører oppgavene manuelt	En 80%-100% automatisert løsning
Jevn kvalitet	Middels	Kvaliteten av arbeidet som blir gjort	Mennesker utfører oppgaven med ulik kvalitet	Roboten utfører oppgaven på lik måte hver gang, noe som gir lik kvalitet hver gang.
Sporbarhet ved feil	Lav	Kan spore opp årsak til feil	Vanskelig å finne ut av hvor feilen er blitt gjort når noe går galt	Rapport vil jevnlig bli sendt ut til brukere med oversikt over eventuelle feil som er blitt gjort, med tilhørende feilmelding utarbeidet av kunde og utviklere.
Økonomisk gevinst	Middels	Økonomien til Skatteetaten	Ansatte i Skatteetaten skal ha lønn for å utføre oppgavene	En sikker robot som utfører oppgavene uten feil, noe som vil gjøre at prosesskontrollør og vedlikehold blir en så liten kostnad som mulig.
Forbedre brukeropplevelsen	Lav	Ansatte i Skatteetaten	Det kan oppleves tidkrevende og ineffektivt av de ansatte å gjøre en slik repeterende oppgave manuelt.	Implementere en robot som er enkel å samarbeide med og som tilfører flyt i arbeidsprosessen.
Effektivitet	Lav	Reduserer tidsbruk for opprettelse av dokumenter	Manuell utførelse som tar tid.	Robot som kan utføre oppgaver raskere enn et menneske.

A.3.5 Alternativer til vårt produkt

Et alternativ til vårt produkt er å endre på rutiner eller arbeidsprosessen for å få utførelsen av den spesifikke oppgaven til å foregå så effektivt som mulig. Ved å endre og optimalisere rutiner vil mindre tid bli brukt på å utføre oppgaven. Simplifisering av arbeidsprosessen vil føre til mindre feil og kvaliteten vil kunne bli jevnere. Det kan også være relevant å ta opp spørsmålet om dette er en oppgave som må bli gjort.

A.4 Produktoversikt

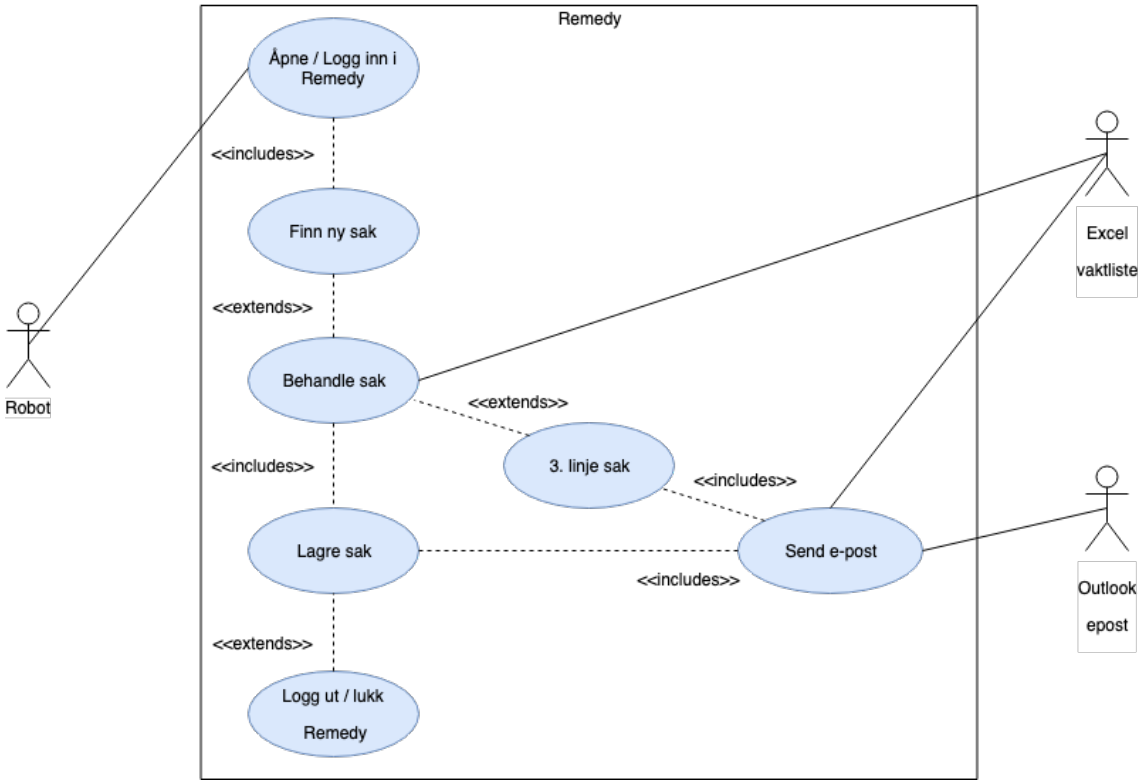
A.4.1 Produktets rolle i brukermiljøet

Robotens rolle i brukermiljøet er at den selvstendig skal samhandle med ulike program og bli en del av systemet rundt oppgaven som utføres av mennesker. Den skal selv åpne opp Remedy og fordele saker til ansatte. Den blir satt i gang av prosesskontroller, som også passer på at roboten fungerer som den skal og videre drift.

A.4.2 Forutsetninger og avhengigheter

For å kunne programmere roboten etter kundens behov, er utviklerne avhengige av å få en-detaljert innføring i hva roboten skal gjøre. Ettersom roboten skal bruke skjermbilder til å utføreoppgavene må ansatte hos seksjon for økonomi vise nøyaktig hvilke knapper som trykkes på og når, hva som skal fylles inn hvor og hvor roboten skal hente informasjon. For at roboten skal kunne samhandle med de ulike programmene, forutsettes det at de er installert både på datamaskinen som programmerer og maskinen som skal kjøre roboten.

A.5 Produktets funksjonelle egenskaper



Figur A.1: Use Case

Åpne/Logge inn i Remedy

Når roboten kjører i Blue Prism, er den allerede logget inn i Skatteetaten interne system. Skatteetaten benytter egentlig SSO slik at man automatisk logges inn når man åpner Remedy, men siden roboten må aksesseres via et eksternt skrivebord krever Remedy brukernavn og passord. Når roboten først er innlogget i Remedy, vil den forbli det helt til roboten logger seg ut eller krysser ned Internet Explorer som Remedy kjøres i.

Finne ny sak

I Remedy får man opp en liste over saker, som skal eller har blitt tildelt en ansatt. Roboten skal gå igjennom listen og finne saker som ikke har blitt tildelt ansatt enda, og begynne med den siste saken som har kommet inn i listen. Roboten skal ikke behandle saker som har blitt behandlet allerede, verken av roboten selv eller saker som har blitt manuelt overstyrt. Dersom

roboten ikke finner en ny sak, vil den vente en gitt tidsperiode før den sjekker listen på nytt. Den vil prøve å sjekke listen et gitt antall ganger før den vil gi seg og avslutte Remedy.

Behandle sak

Hvis roboten har funnet en sak som oppfyller kravene nevnt over, skal den klikke og åpne denne saken. Der skal den sammenligne tittelen med sammendraget. Stemmer ikke disse overens, skal roboten endre sammendragets kategori til å bli lik tittelen. Videre skal roboten sjekke vaktlisten i et lagret Microsoft Excel-regneark og undersøke om saken kan få tildelt en ansatt som er på vakt. Navnet på den ansatte fylles inn i feltet for "Assginee". Er det ingen ansatte, skal ikke roboten gjøre noe med "Assignee"-feltet. Da skal det bli stående blankt.

3.linje sak

Dersom tittelen på saken som roboten behandler er "3.linje sak", må roboten handle på en annen måte enn ved andre saker. Roboten må da hente ut alle ansatte som er satt opp på vaktliste fra en Excel-fil.

Sende e-post

Ved 3.linje saker må roboten sende ut en e-post til alle ansatte som er ført opp i vaktlisten i Excel. Roboten må åpne opp ny e-post, skrive at det er en 3.linje sak i sakslisten som må behandles snarest og sende denne e-posten til de ansatte på vakt gjennom Microsoft Outlook.

Lagre sak

Når roboten er ferdig med å behandle en sak, skal den lagre saken slik at den kommer tilbake til Remedys hovedside. Har den tildelt en sak til en ansatt, vil det nå være synlig i sakslisten.

Logge ut/lukke Remedy

her må vi sjekke litt når roboten er ferdig med å sjekke om det er nye saker i listen.

A.6 Ikke-funksjonelle egenskaper og andre krav

A.6.1 Stabilitet

Roboten skal arbeide selvstendig og håndtere flest mulige feil som oppstår av seg selv, slik at man unngår mye vedlikehold. Produktet skal utføre oppgaver i størst mulig grad uten manuell innblanding. Da er det viktig at den ikke stopper opp på grunn av mangel på feilhåndteringer.

A.6.2 Ytelse

Kundens behov at roboten utfører arbeidet på en effektiv og korrekt måte, derfor er det viktig at roboten arbeider med høy ytelse og nøyaktighet.

A.6.3 Pålitelighet

Skatteetaten er en norsk statlig etat og har dermed ansvar for store mengder viktige dokumenter. Dette setter høye krav til pålitelighet og korrekt utførelse av oppgaver, ettersom feil kan gi store konsekvenser. Påliteligheten til roboten sikres ved grundige tester av roboten, både av utviklerne og Skatteetaten, som har tidligere erfaring med RPA.

A.6.4 Unntakshåndtering

Dersom programmene roboten samhandler med ikke svarer eller andre feil oppstår under kjøring, må roboten håndtere disse feilmeldingene og komme seg på rett spor igjen. Dette vil sikre høy grad av automasjon. For at roboten i størst mulig grad skal bli autonom, må den ha metoder for å takle uforutsette hendelser og gi tilbakemeldinger på hva som har skjedd. Dette er også med på å gi jevnere kvalitet.

A.6.5 Enkel å vedlikeholde

For at roboten skal gi en økonomisk gevinst til Skatteetaten må den kreve minimalt med vedlikehold. Dersom den ikke gjør det vil det ikke være lønnsomt for Skatteetaten å utvikle en robot for denne oppgaven.

A.7 Implementasjon begrensninger

Ettersom ingen av utviklerne har erfaring med RPA eller Blue Prism kan det være det oppstår utfordringer underveis i implementasjonen. For å møte kravene som er satt med hensyn til implementasjonen, skal utviklerne gjennomføre et kurs i Blue Prism før de spesifikke oppgavene som roboten skal løse presenteres. I tillegg vil ansatte i IT-avdelingen som har erfaring med RPA bistå med veiledning.

Vedlegg B

Kravdokumentasjon for Remedy-robot

Innhold

B.1 Introduksjon

B.1.1 Hensikt

Hensikten med dokumentet er å gi en detaljert beskrivelse av RPA-roboten som vi skal utvikle for Skatteetaten og spesifisere kravene for prosjektet slik at både RPA-team, veileder (Majid Rouhani), kunden og utviklerne er sikre på at kravene er forstått av begge parter. Dokumentet inneholder krav til funksjoner, data og egenskaper, slik at funksjonaliteten og implementasjonen er klar. Dette vises gjennom user stories, domenemodell og sekvensdiagram

B.1.2 Omfang

Det skal utvikles en RPA-robot som programmeres ved hjelp av automatsjonsprogramvaren Blue Prism. Roboten skal behandle og fordele saker i saksbehandlingsprogrammet Remedy. Sakene legges inn via Brukerportalen på Skatteetatens intranett. I dag utføres denne oppgaven manuelt av ansatte i seksjonen AT Økonomi i Skatteetaten.

B.2 User-stories

B.2.1 US 1

Som bruker av skatteetatens systemer
Ønsker jeg å raskt få svar på mine henvendelser
Slik at jeg kan fortsette med annet arbeid.

Krav:

- Roboten jobber kontinuerlig
- Roboten tar ikke for seg allerede behandlede saker

B.2.2 US 2

Som ansatt i seksjon for økonomi
Ønsker jeg å automatisere arbeidsoppgaver med robot
Slik at jeg kan utføre andre, mer krevende oppgaver.

Krav:

- Roboten må være 100% automatisert

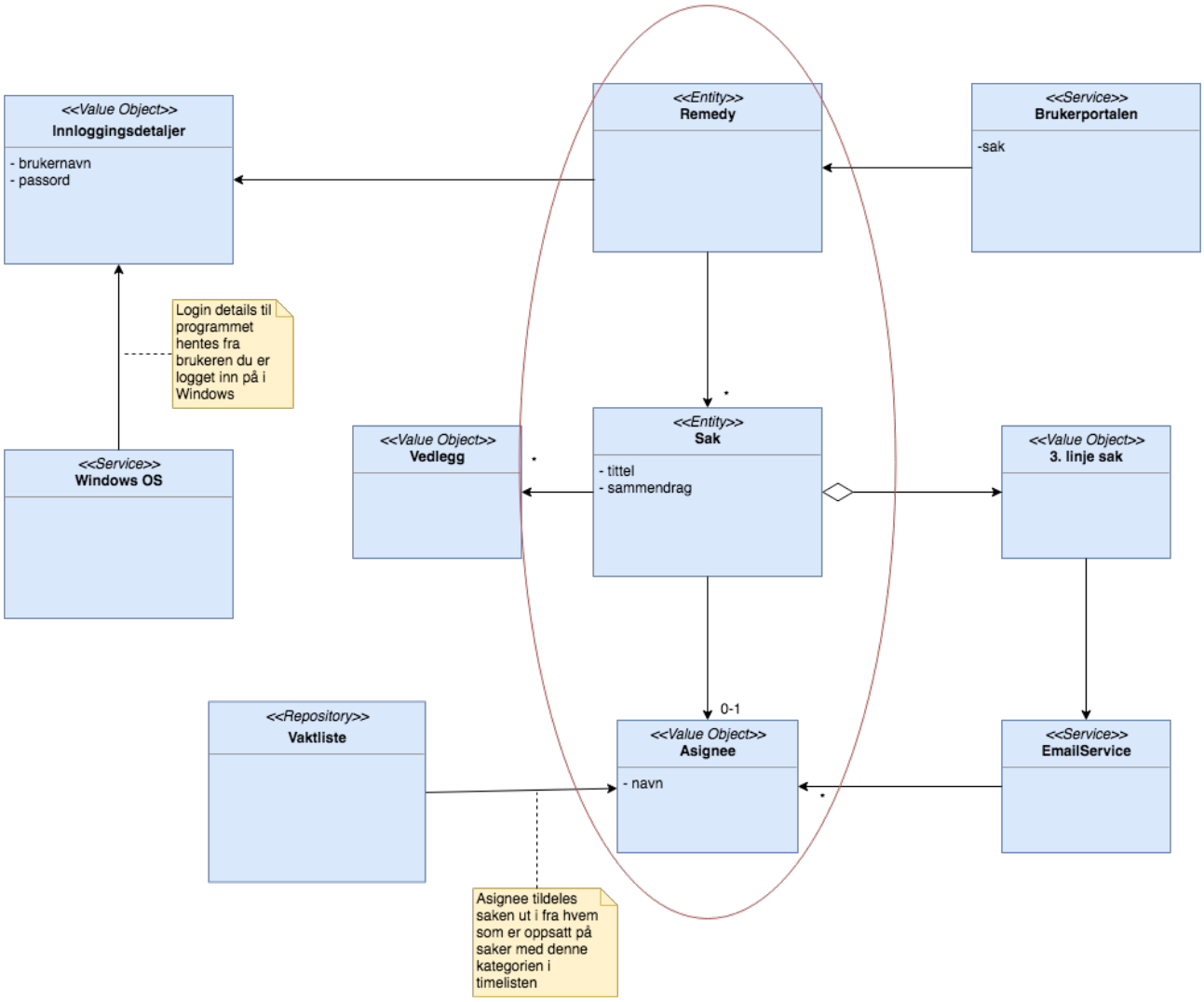
B.2.3 US 3

Som ansatt i seksjon for økonomi
Ønsker jeg å få tildelt saker jeg har ansvar for
Slik at de som har satt opp en sak får raskest og best mulig hjelp/svar.

Krav:

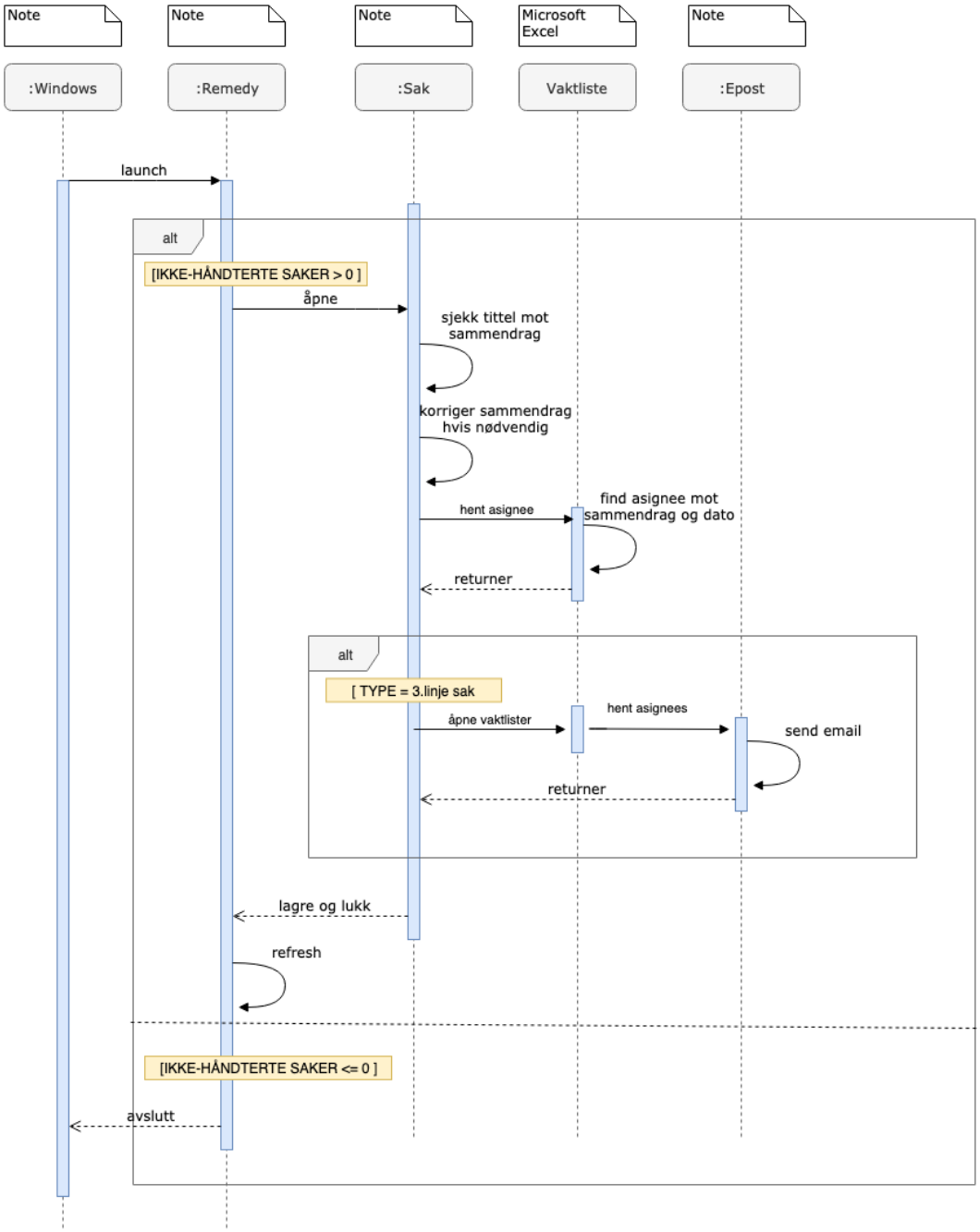
- Timelistene må ha fast format
- Roboten må kunne sette saker på riktig person ut fra dato og type sak, ved hjelp av timelisten

B.3 Domenemodell



Figur B.1: Domenemodell primæroppgave

B.4 Sekvensdiagram



Figur B.2: Sekvensdiagram primæroppgave

Vedlegg C

Systemdokumentasjon for Remedy-robot

Innhold

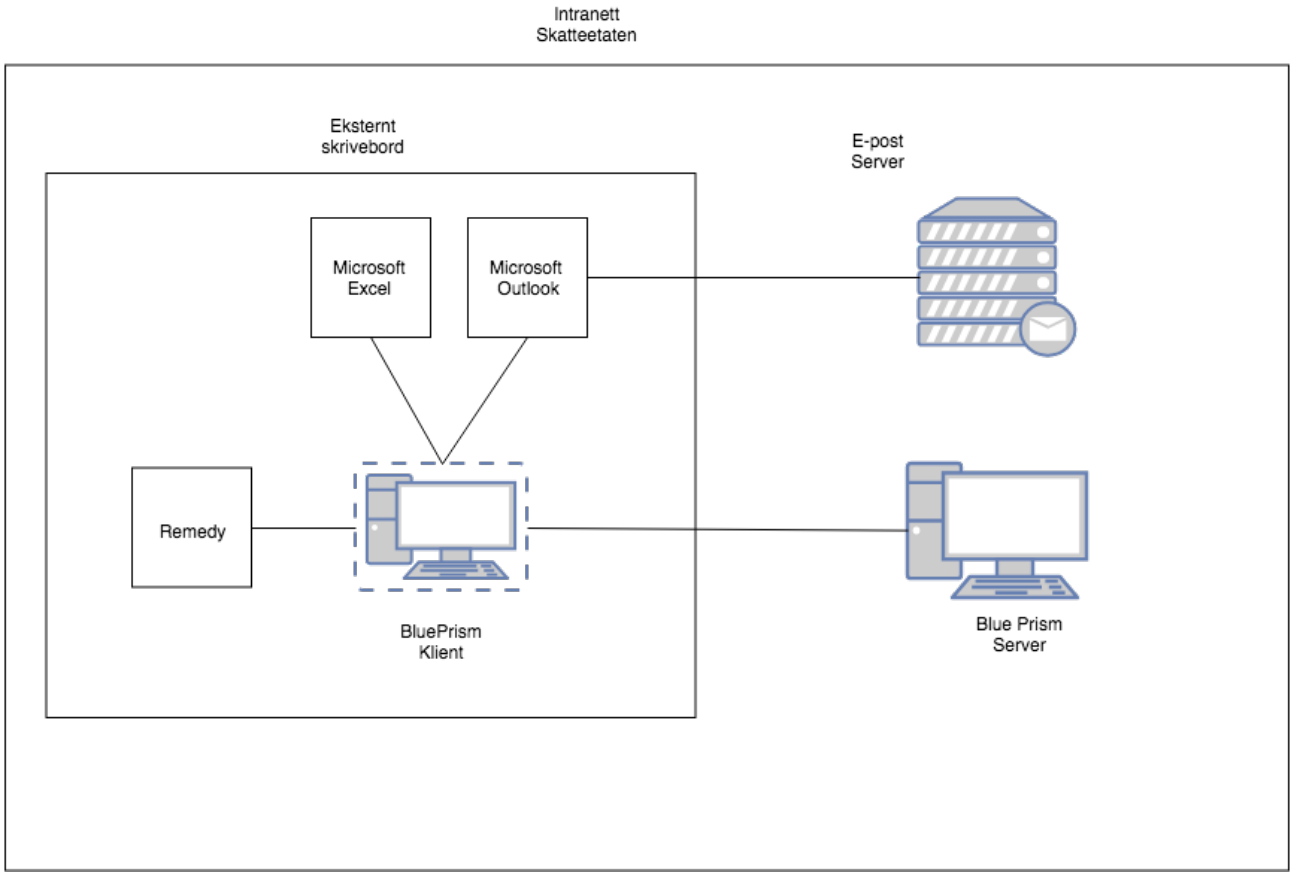
C.1 Introduksjon

C.1.1 Hensikt

Dette dokumentet skal fungere som en detaljert beskrivelse av RPA-roboten som skal fordele saker i Remedy for Skatteetaten. Det skal vise hvordan systemet er strukturert og hvordan de ulike komponentene samhandler med hverandre. I tillegg vil dokumentet inneholde dokumentasjon på selve koden.

C.2 Arkitektur

C.2.1 Arkitekturskisse



Figur C.1: Arkitektur

C.2.2 Tekstlig beskrivelse

1. For å kunne benytte seg av RPA-roboten, må klienten først være logget inn på en datamaskin i Skatteetatens intranett. Det vil si at klienten må ha brukernavn og passord i Skatteetatens systemer. Denne datamaskinen vil fungere som Blue Prism Server.

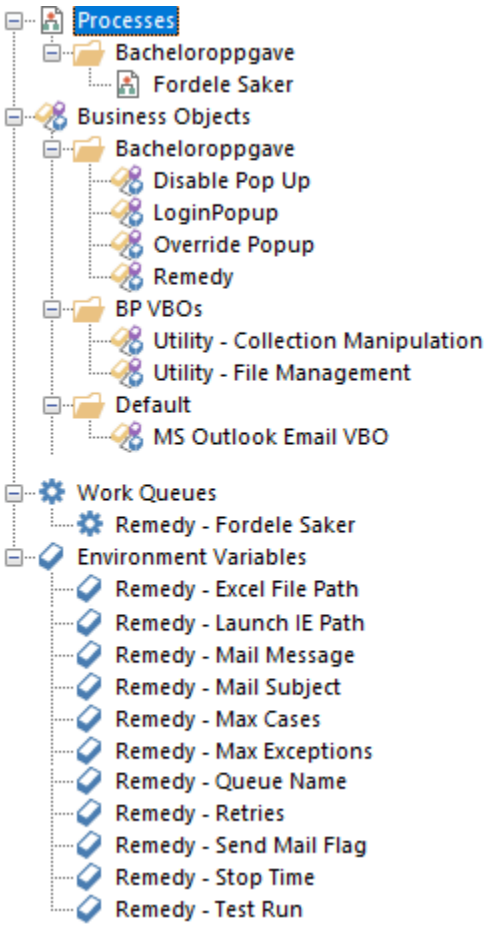
2. Fra Blue Prism Server må klienten deretter koble seg opp mot et eksternt skrivebord som har Blue Prism installert. Dette vil fungere som Blue Prism Klient. Autorisasjonen skjer ved innlogging med samme brukernavn og passord på steg 1.

3. Blue Prism Klienten må så logge inn på Blue Prism med eget brukernavn og passord. Herfra kan en klient kjøre RPA-roboten. Dette vil gjøres av en prosesskontroller, som er en ansatt i Skatteetaten som får ansvar for igangsetting av og vedlikehold av RPA-roboten.

4. Fra Blue Prism Klient vil RPA-roboten selv logge seg inn på Remedy, med SSO.

5. Excel-arket med vaktlisten ligger lokalt på Blue Prism Klienten. Får roboten behov for å sende mail, vil den bruke Microsoft-Outlook til å sende mail, som den er logget inn ved med SSO i Blue Prism Klient.

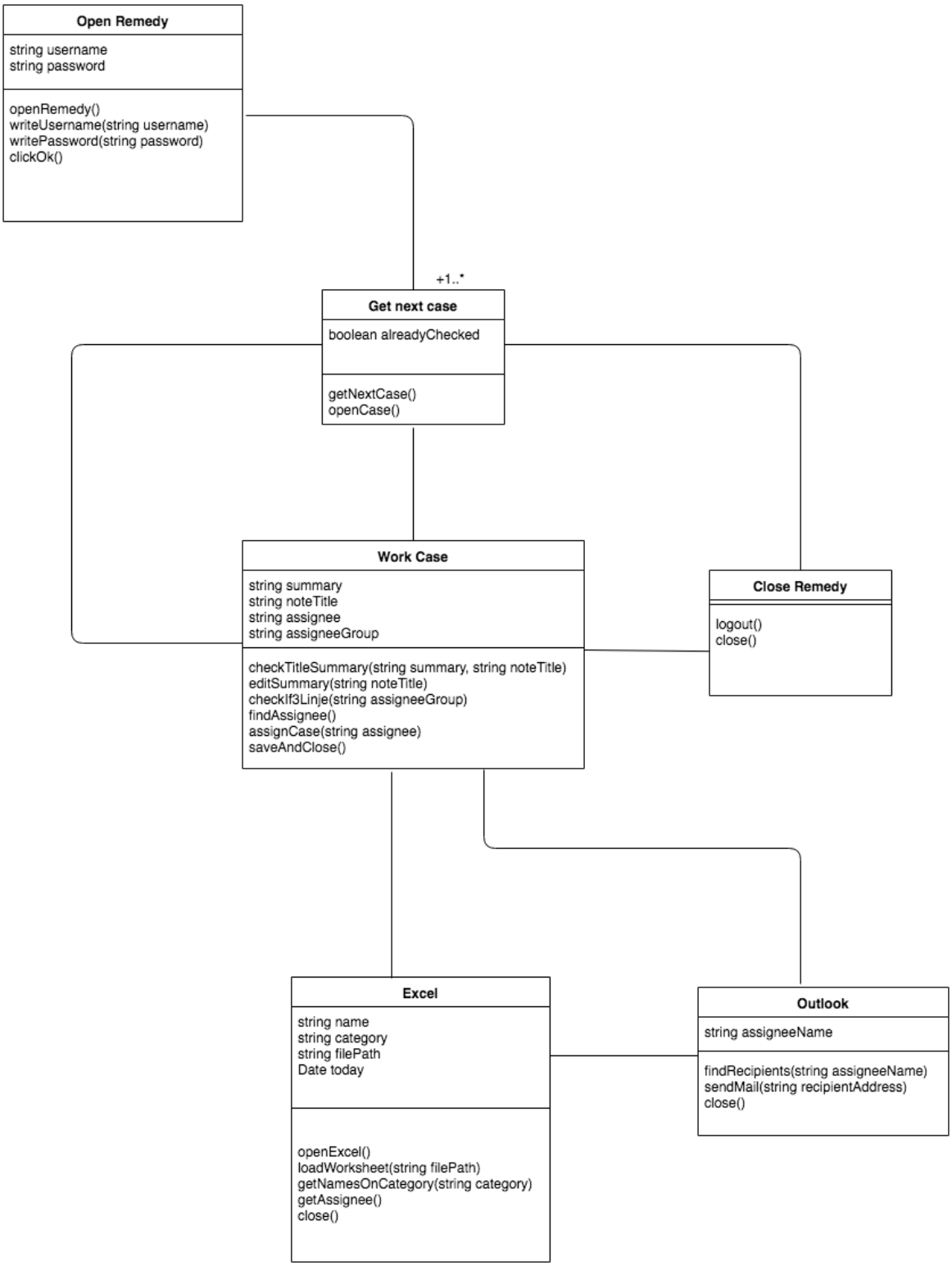
C.3 Prosjektstruktur



Figur C.2: Prosjektstruktur av Remedy-roboten

Figuren viser alle avhengigheter som er nødvendig for å kjøre roboten. LoginPopup er bare nødvendig i testmiljøet, og vil ikke brukes når roboten skal kjøres i produksjon ettersom roboten logger seg inn automatisk med SSO.

C.4 Klassediagram



Figur C.3: Klassediagram

Siden dette ikke er et tradisjonelt systemutviklingsprosjekt med ordinær programmering med klasser, er ikke et nøyaktig klassediagram nødvendig i denne sammenhengen. Likevel ble det lagd en grov skisse av systemet basert på klassediagram i systemutvikling for å få en overordnet oversikt over systemet. Med utgangspunkt i det som vil bli robotens hovedside, er klassene i dette klassediagrammet de viktigste og mest grunnleggende komponentene i systemet og hvordan disse samhandler med hverandre. Klassene i dette diagrammet er "subpages" i Blue Prism. Metodene som er beskrevet i klassene er ikke faktiske metoder som blir utført, men beskrivelser av hendelser innenfor den klassen/siden de er den del av.
Dette klassediagrammet ble brukt som utgangspunkt før selve utviklingen av roboten slik at alle teammedlemmene hadde lik forståelse av hovedkomponentene i systemet. Ettersom det ikke er en nøyaktig fremstilling av selve koden, har ikke denne blitt oppdatert etter utviklingen var ferdigstilt.

C.5 Sikkerhet

For å sikre god beskyttelse mot uautorisert tilgang, blir det benyttet flere forskjellige former for sikkerhetsteknologi, både via sikkerhetsrutiner i Skatteetaten og teknologi i Blue Prism.

C.5.1 Skatteetaten

Skatteetaten har en sentral rolle i det norske samfunnet, og dette medfører at forvaltning av store mengder informasjon om virksomheter og enkeltpersoner. Dette stiller høye krav til sikkerhet for ansatte i Skatteetaten og til de digitale tjeneste som er i bruk. Som vist i arkitekturskissen (se Figur C.1), foregår all aktivitet med og av roboten innenfor Skatteetatens intranett. Roboten vil ha autorisasjon via en brukerident til Skatteetatens intranett. Det vil si at en RPA-robot vil ha de samme rettighetene og tilgangene som et menneske med brukerident i Skatteetatens systemer, og den er beskyttet innenfor Skatteetatens sikkerhetsprotokoller og teknologi.

C.5.2 Blue Prism

Blue Prism tilbyr funksjonalitet for å både sikre mot uautorisert tilgang til roboten og til de programmene Blue Prism samhandler med. Den viktigste funksjonaliteten er hvordan Blue Prism lagrer og krypterer legitimasjon i Credential Manager.

Credential Manager

Ettersom roboten samhandler med andre programmer slik et menneske gjør, må den blant annet logge seg inn i programmene på samme måte. Med andre ord, må roboten lagre innloggingslegitimasjon som den selv skal bruke. Da er det viktig at sikkerheten ivaretas. RPA-roboten må logge seg inn på Remedy selv, og dermed er lagring av legitimasjon relevant for denne oppgaven. En enkel og usikker måte å lagre brukernavnet og passordet til Remedy ville vært å hardkode det inn i "Data Items" i selve prosessen som roboten da aksesserte. Problemet med dette er at alle prosessene som er i samme utviklingsmiljø er åpne. Med andre ord kan alle som har tilgang til utviklingsmiljøet også ha tilgang til å gå inn i prosessen og lese legitimasjonen. Slik informasjon må derfor krypteres i Credential Manager.

Credential Manager er en del av "System" i Blue Prism, der man kan opprette en Credential med innloggingsdetaljer for et program. Slik legitimasjon lagres i Blue Prisms egne database, men blir kryptert på en slik måte at bare brukere som er autoriserte kan hente det. Når legitimasjonen opprettes i Credential Manager, bestemmes autoriseringen av følgende aksessrettigheter:

- **Process** - Bare spesifiserte prosesser kan hente ut innloggingsdetaljene. Hvis legitimasjonen skal brukes av en subprosess eller fra VBO, må foreldreprosessen ha autorisasjon for at det skal være mulig å hente ut informasjonen.
- **Resource** - Bare spesifiserte maskiner, som i dette tilfellet vil være eksterne skrivebord, er autorisert til å kjøre og hente ut legitimasjon.
- **User Role** - Bare spesifiserte brukere som bruker maskinen er autorisert til å bruke prosessen og hente ut informasjonen. Dette vil stoppe brukere som prøver å hente ut informasjon fra en maskin som er "offentlig" og prosesser som er definert for en spesifikk prosess.

Kryptering

For å lagre, kryptere og hente legitimasjon i Blue Prisms database, må det opprettes et krypteringsskjema som vil generere en krypteringsnøkkel. Man kan velge hvor denne nøkkelen skal lagres. Krypteringsnøkkelen kan enten lagres i Blue Prisms database eller i Blue Prims

applikasjonsserver. Når legitimasjonen lagret i databasen skal hentes ut vil maskinen aksessere nøkkelen direkte fra databasen for å dekryptere. Hvis krypteringsnøkkelen lagres i applikasjonsserveren, vil maskinen sende legitimasjonen i klartekst til serveren, som med krypteringsnøkkelen vil kryptere informasjonen før det sendes videre til lagring i databasen. Med dette valget vil applikasjonsserveren fungere som et mellomledd mellom maskin og database, mens hvis det kun er databasen som blir brukt vil det være en direkte kobling mellom maksin og database. Blue Prism anbefaler å bruke en applikasjonsserver, men siden teamet ikke har mulighet til å konfigurere Skatteetatens applikasjonsserver, falt valget på å bruke databasen, som gir tilfredsstillende beskyttelse for denne oppgaven.

Når krypteringsnøkkelen skal opprettes, er det også mulig å velge krypteringsalgoritmen som skal benyttes. Her falt valget på den sterkeste algoritmen som ble tilbudt; AES-256 bit. AES står for Advanced Encryption Standard og brukes som standard av blant annet amerikanske myndigheter [1].

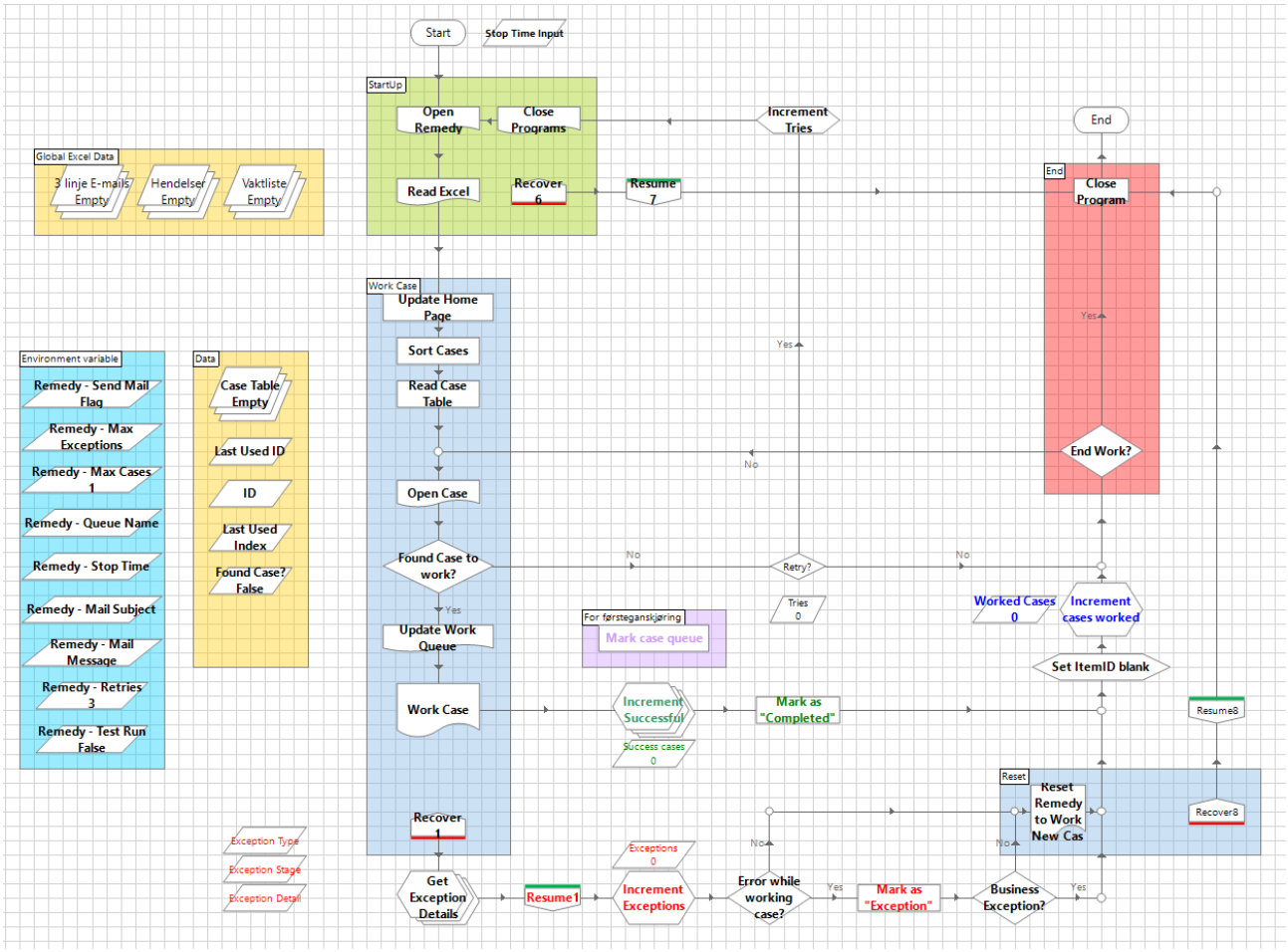
C.6 Installasjon og kjøring

For å kjøre roboten i produksjonsmiljø, må prosessen først eksporteres fra testmiljø til produksjonsmiljø. Da må prosessen utgis som en release-pakke som inneholder alle prosesser, VBO arbeidskøer og miljøvariabler som blir brukt. Blue Prism har egen funksjonalitet som pakker alle avhengighetene i form av en .bprelease-fil, som er en .xml-fil skreddersydd for Blue Prism. Når pakken er klar, kan den eksporteres og sendes til prosesskontroller som importerer den i produksjonsmiljøet. Derfra vil prosesskontroller koble seg opp mot en ekstern server, og deretter kjøre prosessen. Prosesskontroller kan enten koble opp og kjøre prosessen selv, eller sette opp en tidsplan der prosessen skal kjøre på visse dager og bestemte klokkeslett.

C.7 Dokumentasjon av kildekode

For å gi et overordnet bilde av funksjonaliteten er skjermbilder av de tre viktigste prosessene fra Remedy-roboten vedlagt.

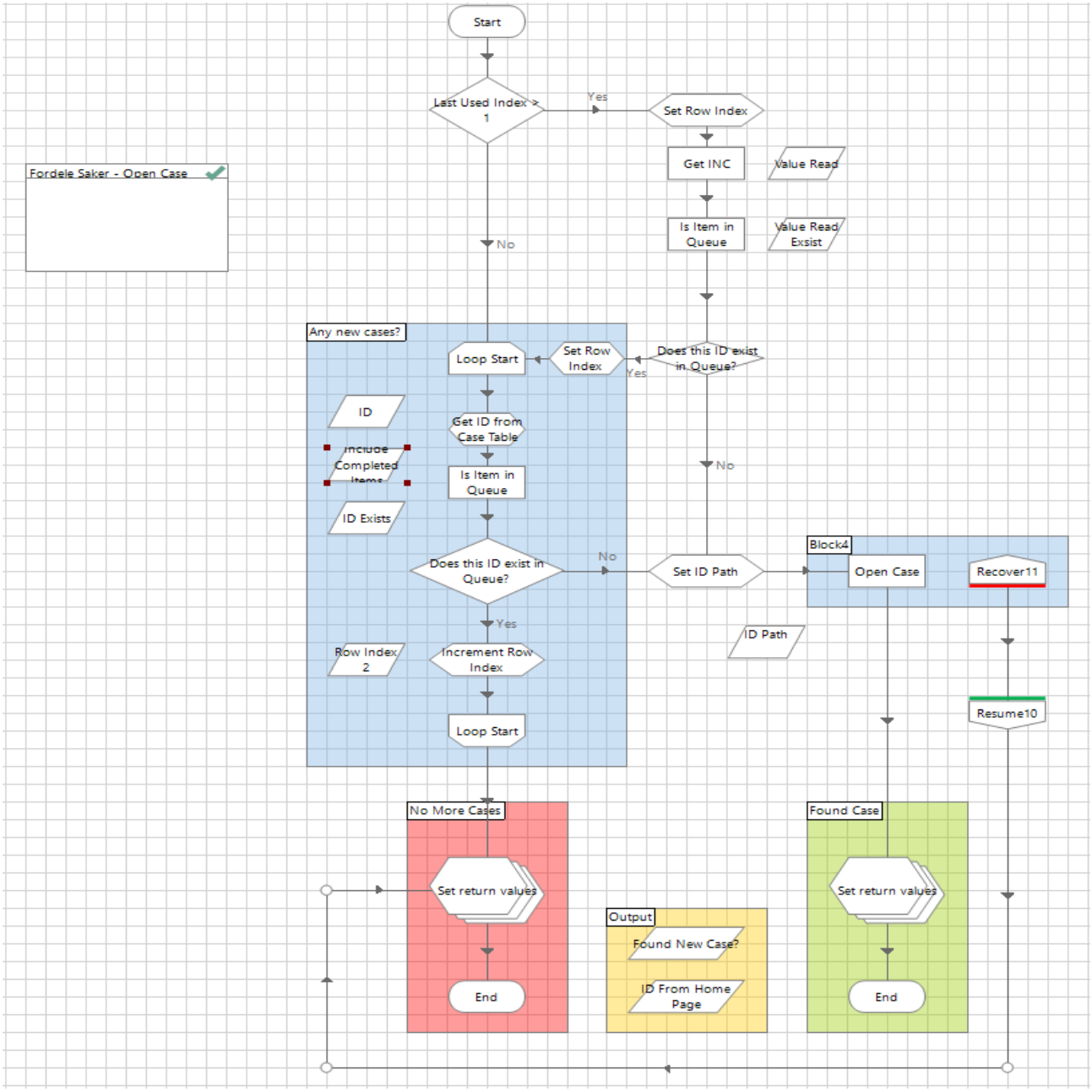
C.7.1 Hovedside (Main Page)



Figur C.4: Hovedsiden i Remedy-robot

Hovedsiden tar for seg Remedy-robotens generelle flyt, globale variabler og miljøvariabler samt unntakshåndtering om en sak feiler. Hovedsiden inneholder svært lite logikk, med unntak av å markere status til en sak og inkrementere antall behandlede saker og forsøk.

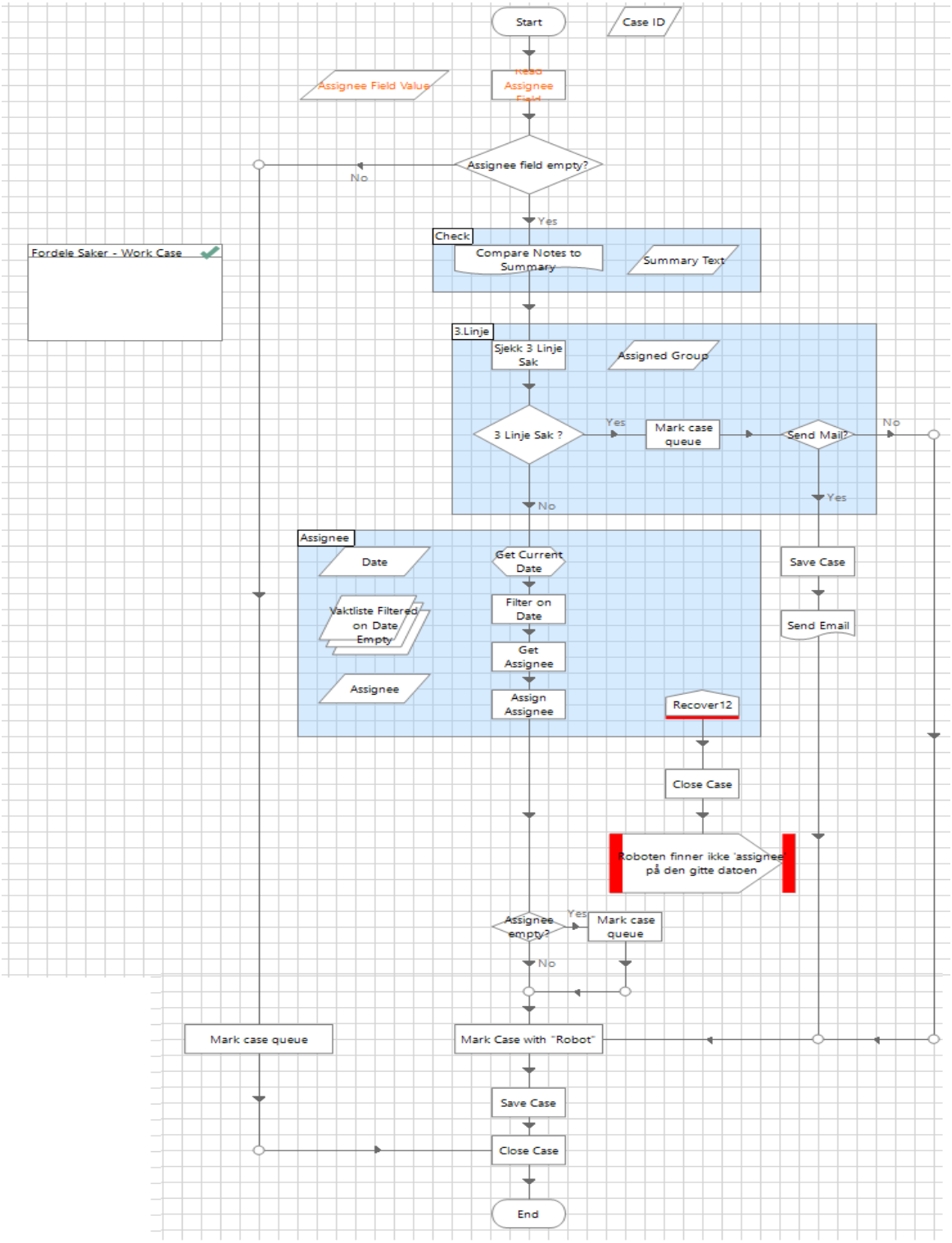
C.7.2 Åpne sak (Open Case)



Figur C.5: Side med logikk for å åpen en sak i Remedy.

"Åpne sak" sjekker om den valgte saken fra sakstabellen allerede eksisterer i arbeidskøen ved å sjekke om saksIDen ligger i køen. Hvis den ikke gjør det, vil den sette HTML-stien til riktig verdi og åpne saken. Dersom saken eksisterer i køen, vil den gå gjennom sakstabellen fra starten av og sjekke om den finner en sak som ikke er behandlet allerede. Finner den en ubehandlet sak, vil den åpnes- Finner den ingen, vil den avslutte.

C.7.3 Behandle sak (Work Case)



Figur C.6: Side med logikk for å behandle (fordele) sak i Remedy

Remedy-roboten sjekker først om saken allerede er tildelt en saksbehandler. I det tilfellet, vil

den bare lukke saken. Hvis den ikke er tildelt, vil den sjekke "Notes"-feltet og sammenligne dette med informasjon fra Excel-filen. Så sjekker roboten om saken er en 3.linje sak. Da vil den sende mail om dette til aktuelle saksbehandlere. Dersom saken ikke er 3.linje, vil den hente ut hvilken saksbehandler saken skal tildeles til ved å filtrere vaklisten på dato og type sak. Videre vil saken lagres og markert som "Robot" før den lukkes.

C.8 Testing

Utviklerne har kun hatt tilgang et testmiljø innad i Blue Prism, der all utvikling av roboten ble gjort. I tilegg fikk utviklerne en egen sakskø i Remedy som bare utviklerne og kunde hadde tilgang til. I denne sakskøen kunne utviklerne og kunde legge inn testsaker som roboten kunne arbeide på. Når utviklerne mente at roboten tildelte testsakene på en tilfredstillende måte, ble kunden kalt inn til QA-prosess i testmiljø i Gantt-diagram. Kunde lager et testskript med forventet resultat i forkant av kjøring, som sammenlignes med det faktiske resultatet etter kjøring. Her viste utviklerne hvordan roboten arbeidet på testsakene i sanntid mens kunde observerer. Dersom kunden er mener roboten presterer slik den skal, vil det bli kalt inn til prøveproduksjon. Da vil prosessen overføres fra testmiljø til produksjonsmiljø. Først vil roboten bli kjørt på testsaker i produksjonmiljø, før den får prøve seg på faktiske saker som ligger inne i Remedy. Løser roboten sakene i henhold til kundens krav, vil den bli godkjent, og den er klar til å settes i produksjon.

C.9 Referanser

[1] "Cryptology," Encyclopædia Britannica, Inc., August 2016, [Accessed: 29.03.19]. [Online]. Available: https://www.britannica.com/topic/cryptology/Cryptanalysis

Vedlegg D

Visjonsdokument for Arkivrobot

Innhold

D.1 Innledning

Hensikten med visjonsdokumentet er å få en oversikt over overordnede mål og krav for prosjektet, samt analysere interessentene og deres behov. I tillegg fokuseres det på hvorfor behovene finnes, slik at de kan oppfylles på best mulig vis. RPA roboten som skal utvikles er ønsket og eid av økonomiavdelingen i Skatteetaten. Utviklerne er studentene Kimia Abtahi, Camilla Haaheim Larsen og Nora Othilie Ringdal. Prosjektet er gitt som bacheloroppgave for 3.klasse Dataingeniør ved NTNU, Trondheim.

D.2 Sammendrag problem og produkt

Se kapittel A.2.

D.3 Overordnet beskrivelse av interessenter og brukere

Se kapittel A.3 for oppsummering av interessenter og brukere.

D.3.1 Brukermiljøet

Det skal utvikles en RPA robot som skal erstatte de ansatte i økonomiavdelingens behov for å arkivere mail med vedlegg angående bilag i en spesiell mappestruktur som er felles. Roboten skal passe inn i et brukermiljø der det per dags dato er mennesker som utfører oppgaven. Ettersom roboten skal erstatte mennesker, må den samhandle med de eksisterende systemene. Den vil løse oppgavene på samme måte som de ansatte, men ved hjelp av skjermbilder.
Roboten skal åpne Microsoft Outlook og navigere til en angitt "Økonomi postkasse". Der skal gå igjennom innboksen og lete et mail for å finne et spesielt emnefelt eller avsender. Finner den det emnefeltet, skal den åpne vedlegget i mailen i Notepad og hente ut relevant bilagsinformasjon. Deretter skal roboten åpne Microsoft Utforsker og lagre denne filen i riktig filstruktur på fellesområdet.
Når roboten er tatt i bruk, vil det være de ansatte i økonomiavdelingen som varsler prosesskontroller når det er behov for å kjøre roboten. Prosesskontroller har ansvar for den produksjonssatte roboten, og må igangsette denne ved behov.

D.3.2 Sammendrag av brukernes behov

Se kapittel A.3.4.

D.3.3 Alternativer til vårt produkt

Se kapittel A.3.5.

D.4 Produktoversikt

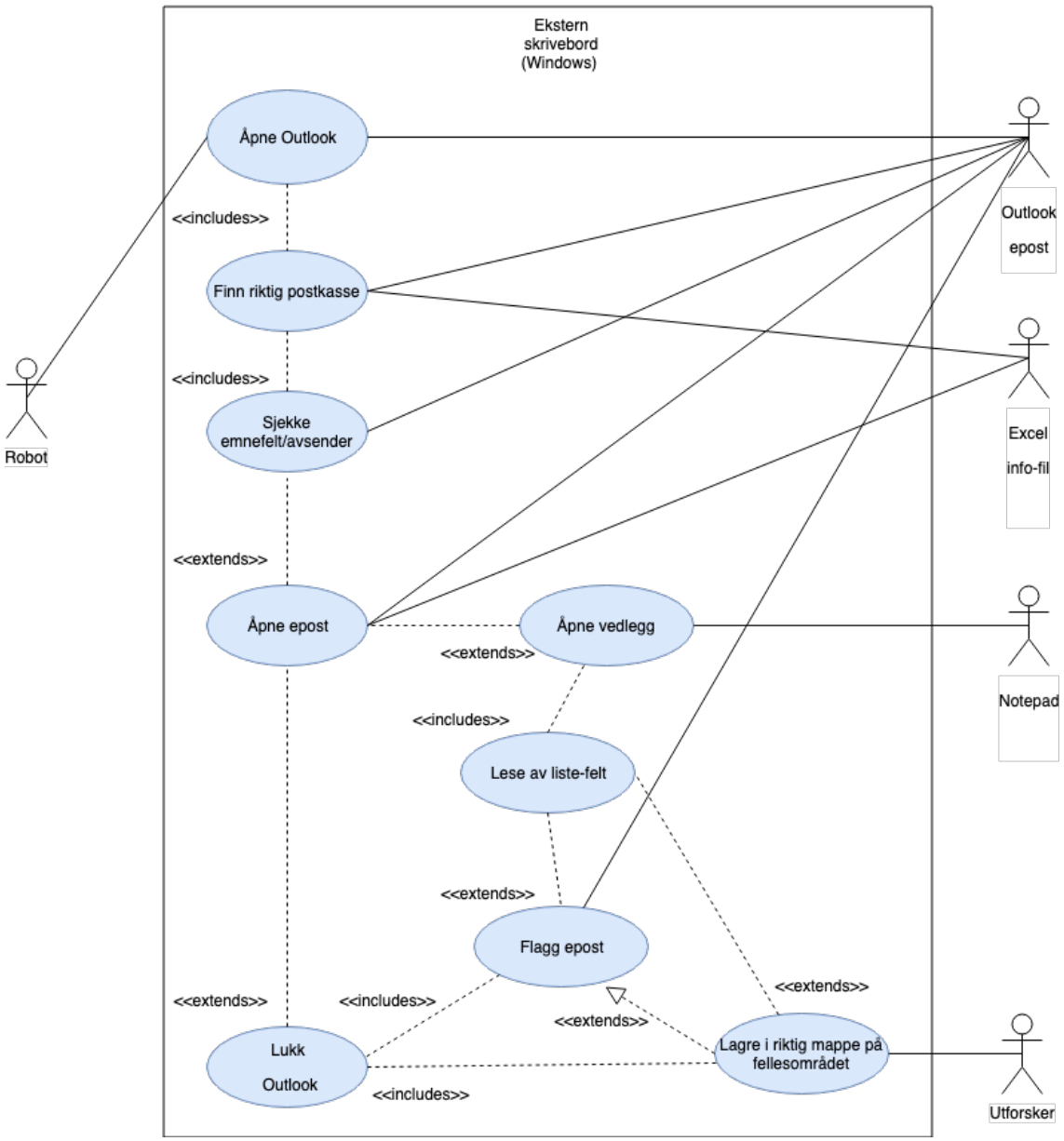
D.4.1 Produktets rolle i brukermiljøet

Robotens rolle i brukermiljøet er at den selvstendig skal samhandle med ulike program og bli en del av systemet rundt oppgaven som utføres av mennesker. Den skal bruke Outlook til å behandle mail som kommer til kundens postkasse.

D.4.2 Forutsetninger og avhengigheter

For å kunne programmere roboten etter kundens behov, er utviklerne avhengige av å få en detaljert innføring i hva roboten skal gjøre. Ettersom roboten skal bruke skjermbilder til å utføre oppgavene må økonomiavdelingen vise oss nøyaktig hvilke knapper som trykkes på og når, hva som skal fylles inn hvor og hvor roboten skal hente informasjon. For at roboten skal kunne samhandle med de ulike programmene, forutsettes det at de er installert både på datamaskinen som programmerer og maskinen som skal kjøre roboten.

D.5 Produktets funksjonelle egenskaper



Figur D.1: Use Case Arkivrobot

Åpne Microsoft Outlook

Skatteetaten benytter seg av SSO, så når roboten kjøres i Blue Prism vil det allerede være koblet til Intranettet. Det betyr at roboten ikke behøver å logge seg inn på Outlook for å åpne det, ettersom den vil allerede være pålogget med den brukeren som logget seg inn på det eksterne skrivebordet. Roboten vil fortsatt være pålogget Outlook selv om Outlook lukkes.

Finn riktig postkasse

Roboten må navigere seg til riktig postkasse slik at den kan finne epostene som arkiveres. Da må den hente inn navnet på postkassen fra Excel-filen, og sammenligne det med postkasser i Outlook for å finne frem til den riktige. Hvis postkassen ikke finnes eller det er en skrivefeil, vil roboten avslutte prosessen med en feilmelding.

Sjekk emnefelt/avsender

Roboten skal gå gjennom innboksen og sjekke om emnefeltet og avsenderen for eposten matcher med input fra Excel-filen. Dersom emnefeltet og/eller avsenderen stemmer overens med input, skal roboten gå videre til steg 5.4. Hvis ikke, skal roboten gå videre til neste epost, helt til det ikke er flere eposter som den ikke har sjekket eller behandlet. Da skal roboten videre til steg 5.

Åpne epost

Dersom emnefeltet og/eller avsender er riktig i forhold til input fra Excel-filen, skal roboten åpne denne e-posten.

Åpne vedlegg

Når den riktige e-posten er åpnet, skal roboten lagre vedlegget i et bufferområde og lese av teksten i vedlegget som lagres i et data objekt.

Lese av liste-felt

Dersom bilaget er feilliste skal roboten videresende e-posten til DFØ og sette seg selv som kopi. Neste gang roboten kjører vil den finne e-posten den sendte til seg selv og arkivere denne i mappe kalt "feilliste".

Flagg epost

Dersom bilaget er av typen feilliste skal eposten flagges.

Lagre i riktig mappe på fellesområdet

Ved informasjon fra bilaget som bilagsintervall, type bilag og dato skal roboten lagre bilaget på riktig filområde med riktig filnavn.

Lukk Outlook

Roboten skal lukke outlook når den er ferdig med å arbeiede med alle e-postene.

D.6 Ikke-funksjonelle egenskaper og andre krav

Se kapittel A.6.

D.7 Implementasjon begrensninger

Denne oppgaven er gitt som sekundæroppgave av Skatteetaten, med den betydning at det regnes som en bonusoppgave etter ferdigstillingen av primæroppgaven. Det vil si at det er ikke er et krav fra Skatteetaten at roboten skal implementeres og ferdigstilles, men at teamet arbeider med løsningen så langt som det lar seg gjøre og leverer dette. Dette er gjort ettersom denne oppgaven er gitt sent i bachelorløpet og det ikke er en garanti fra teamets side at løsningen leveres før bacheloroppgaven må leveres i mai. Teamet hadde ikke erfaring med RPA-utvikling før primæroppgaven og det var dermed vanskelig å forutsi hvor lang tid utviklingen av primæroppgaven ville ta. Ettersom store deler av primæroppgaven ble ferdig tidligere enn forutsatt, ble det avtalt at en ny oppgave skulle gis.

Vedlegg E

Kravdokumentasjon for Arkivrobot

Innhold

E.1 Introduksjon

E.1.1 Hensikt

Hensikten med dokumentet er å gi en detaljert beskrivelse av RPA- roboten som vi skal utvikle for Skatteetaten og spesifisere kravene for prosjektet slik at både RPA-teamet, veileder (Majid Rouhani), kunden og utviklerne er sikre på at kravene er forstått av begge parter. Dokumentet inneholder krav til funksjoner, data og egenskaper, slik at funksjonaliteten og implementasjonen er klar. Dette vises gjennom user stories, domenemodell og sekvensdiagram.

E.1.2 Omfang

Det skal utvikles en RPA-robot som programmeres ved hjelp av automatsjonsprogramvaren Blue Prism. Roboten skal arkivere bilag som blir sendt til AT Økonomi sin Outlook postkasse. I dag utføres denne oppgaven manuelt av ansatte i av avdelingen AT Økonomi i Skatteetaten.

E.2 User-stories

E.2.1 US 1

Som ansatt i økonomiavdelingen
Ønsker jeg å automatisere arbeidsoppgaver med robot
Slik at jeg kan utføre andre, mer krevende oppgaver
Krav:

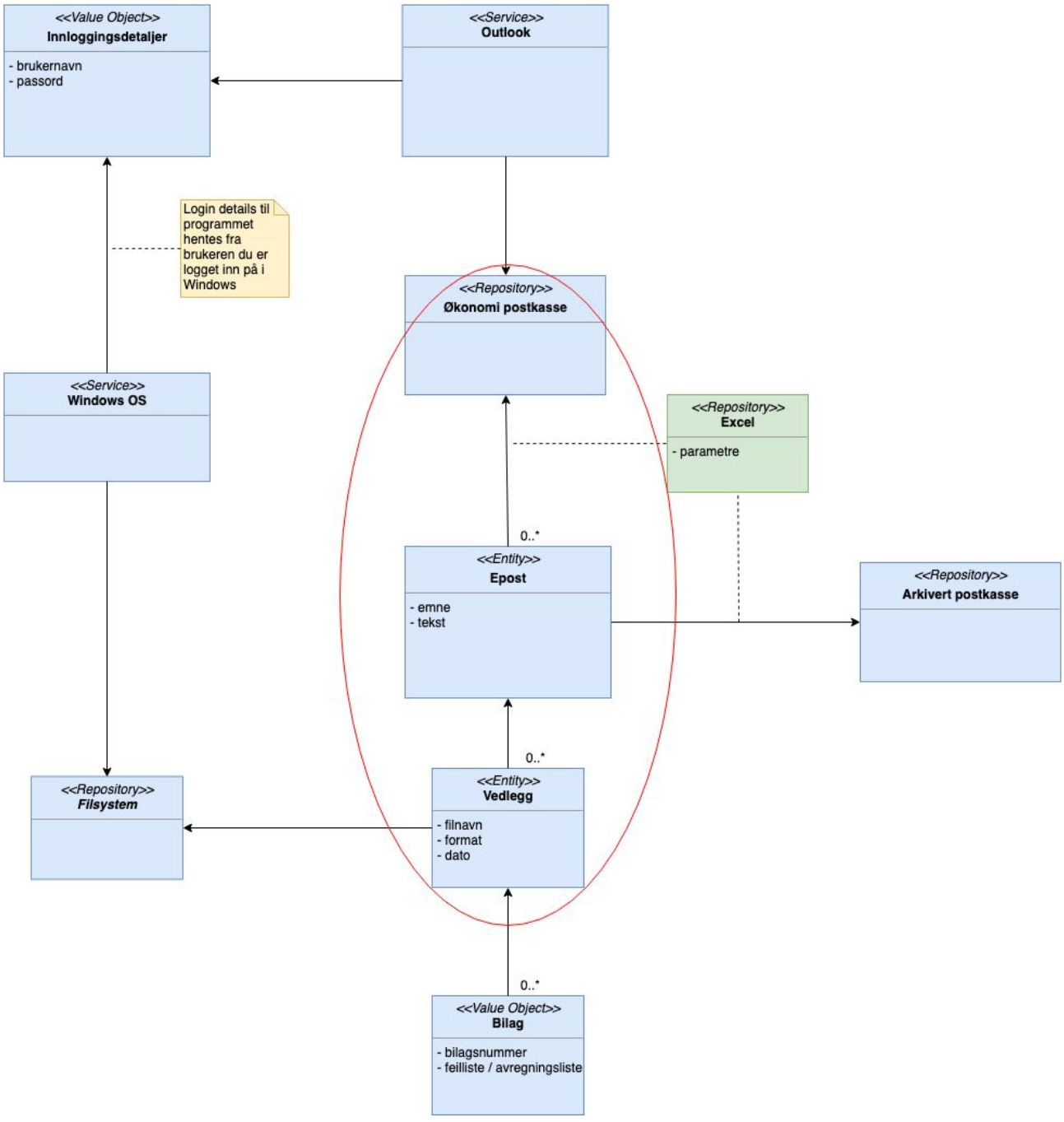
- Roboten må være 80-100% automatisert

E.2.2 US 2

Som ansatt i økonomiavdelingen
Ønsker jeg å ha muligheten til å se hva roboten har jobbet med
Slik at jeg føler meg komfortabel med å la en robot gjøre arbeidet
Krav:

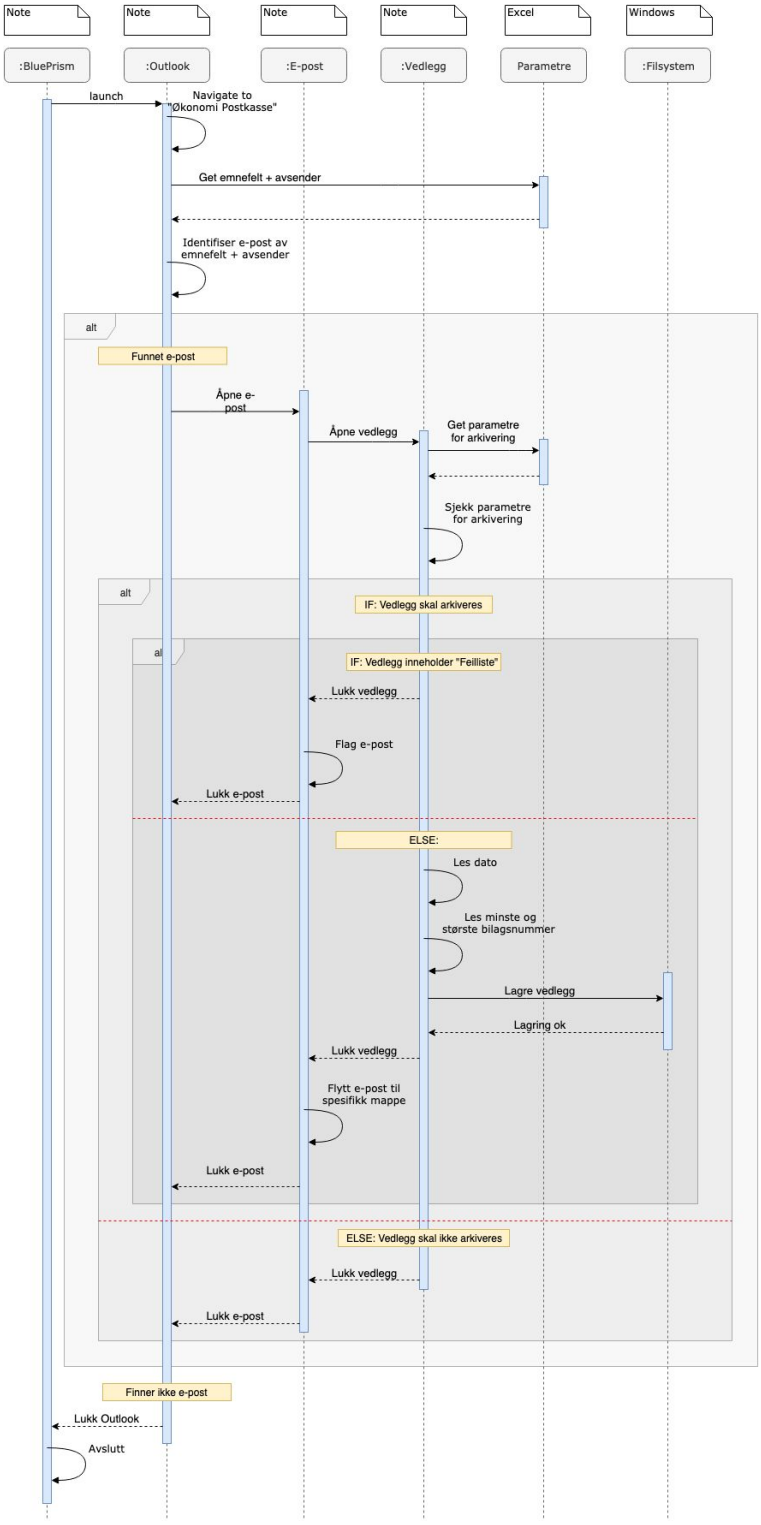
- Gi grundig tilbakemelding til AT Økonomi om hva roboten har gjort
- Ha mulighet til å se om roboten har videresendt e-post til DFØ.

E.3 Domenemodell



Figur E.1: Domenemodell arkivrobot

E.4 Sekvensdiagram



Figur E.2: Sekvensdiagram arkivrobot

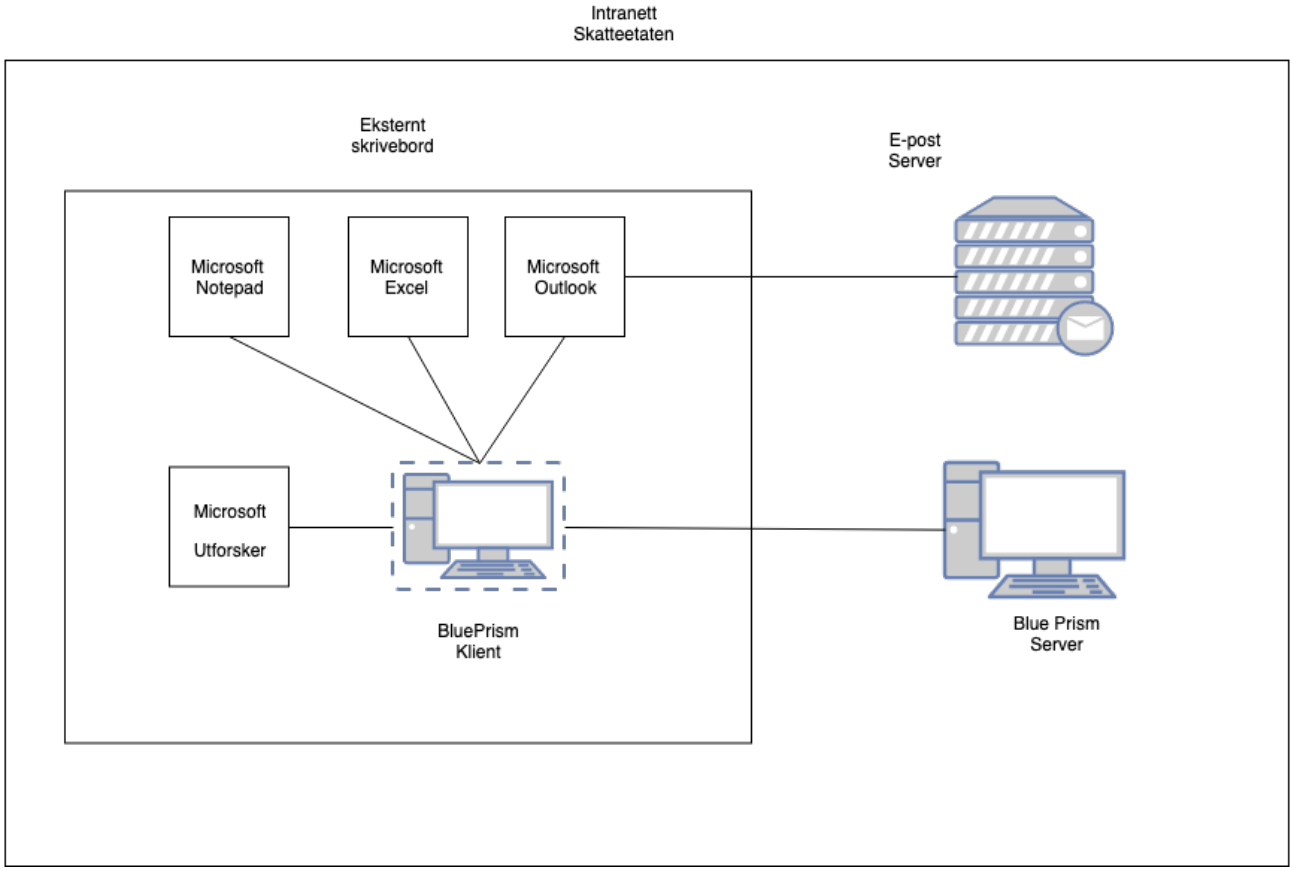
Vedlegg F

Systemdokumentasjon for Arkiv-robot

Innhold

F.1 Introduksjon

F.2 Arkitektur



Figur F.1: Arkitektur av arkiv-robot

1. For å kunne benytte seg av arkiv-roboten, må klienten først være logget inn på en datamaskin i Skatteetatens Intranett. Det vil si at klienten må ha brukernavn og passord i Skatteetatens systemer. Denne datamaskinen vil fungere som Blue Prism Server.

2. Fra Blue Prism Server må klienter deretter koble seg opp mot et eksternt skrivebord som har Blue Prism installert. Dette vil fungere som Blue Prism Klient. Autorisasjonen skjer ved innlogging med samme brukernavn og passord på steg 1.

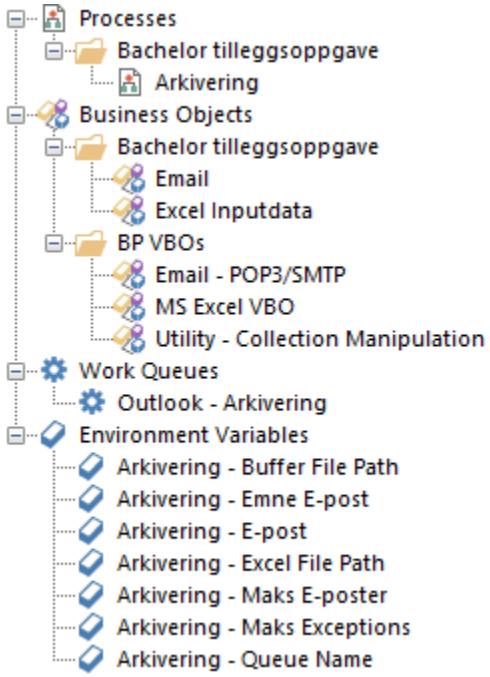
3. Blue Prism Klienten må så logge inn på Blue Prism med eget brukernavn og passord. Herfra kan en klient kjøre RPA-roboten. Dette vil gjøres av en prosesskontroller, som er en ansatt i Skatteetaten som får ansvar for igangsetting av og vedlikehold av RPA-roboten.

4. Fra Blue Prism Klient vil RPA-roboten allerede være innlogget i Outlook, med SSO. Når Outlook skal hente og sende e-post, vil den være i kontakt med en e-post server.

5. Excel-arket med vaktlisten ligger lokalt på Blue Prism Klienten.

6. Bilags-vedlegget lagres i en felles mappeområde som Blue Prism Klienten har tilgang til via Microsoft Utforsker.

F.3 Prosjektstruktur



Figur F.2: Prosjektstrukturen til arkiv-roboten

F.4 Klassediagram

Utviklerne hadde ved oppstart av arkiv-roboten mer erfaring med RPA-utvikling, og valgte å bruke hovedsiden til Remedy-roboten som mal. Dermed ble utformingen av et klassediagram sett på som overflødig.

F.5 Sikkerhet

Se kapittel C.5.

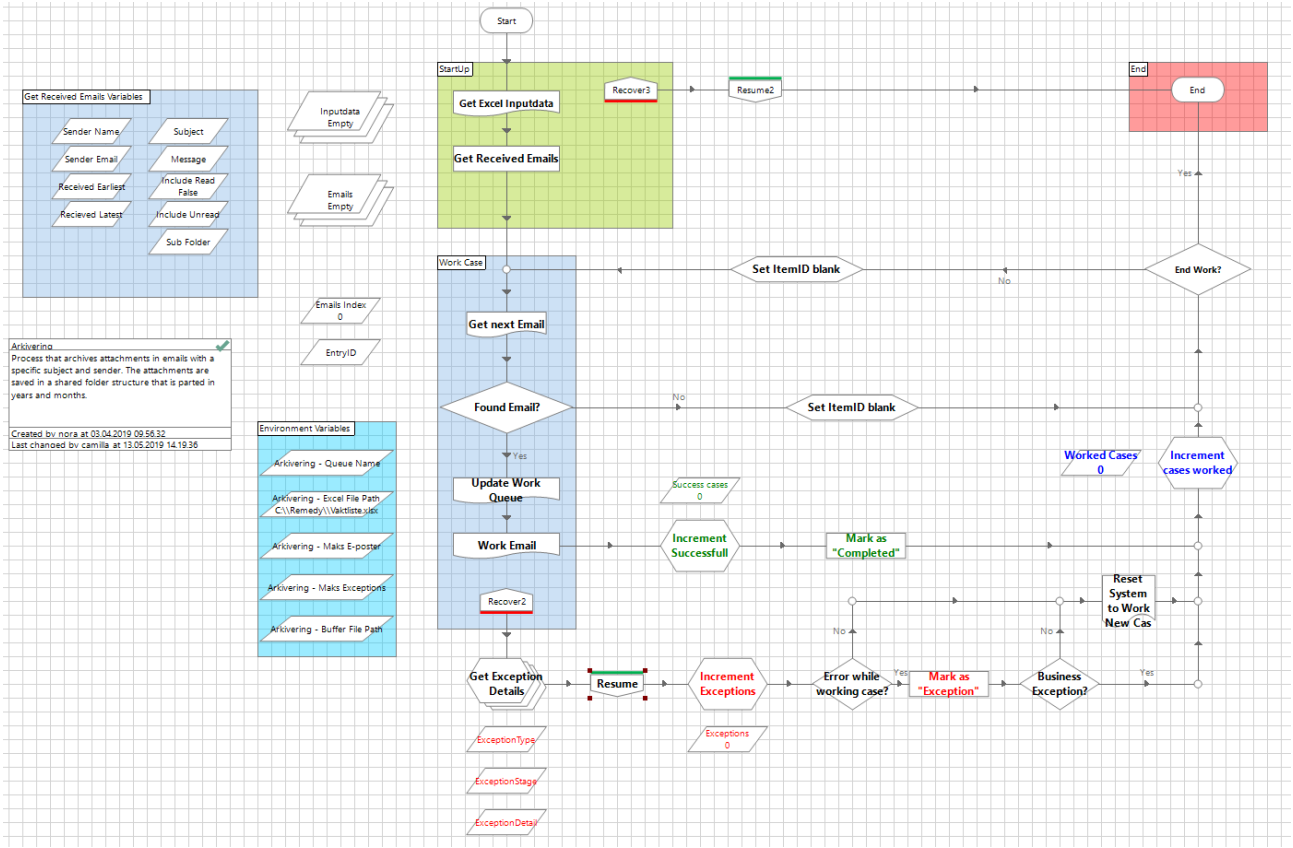
F.6 Installasjon og kjøring

Se kapittel C.6.

F.7 Dokumentasjon av kildekode

Viser overordnet funksjonalitet for de tre viktigste delen av kildekoden. Fullstendig kildekode ligger i zip-fil som leveres til NTNU.

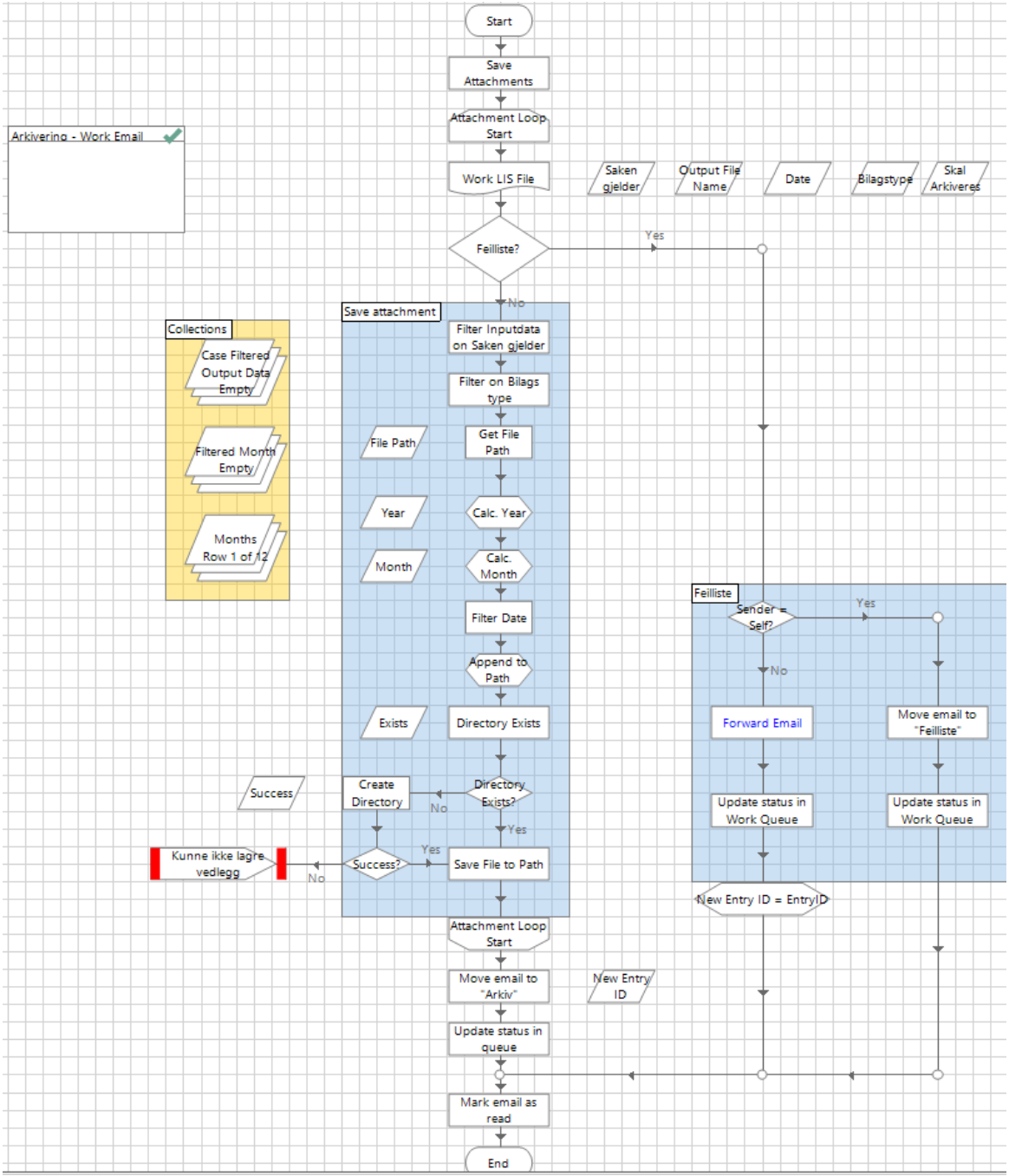
F.7.1 Hovedsiden (Main Page)



Figur F.3: Hovedsiden i arkiv-robot

Hovedsiden tar for seg arkiv-robotens generelle flyt, globale variabler og miljøvariabler samt unntakshåndtering om en sak feiler. Hovedsiden inneholder svært lite logikk, med unntak av å markere status til en sak og inkrementere antall behandlede saker og forsøk.

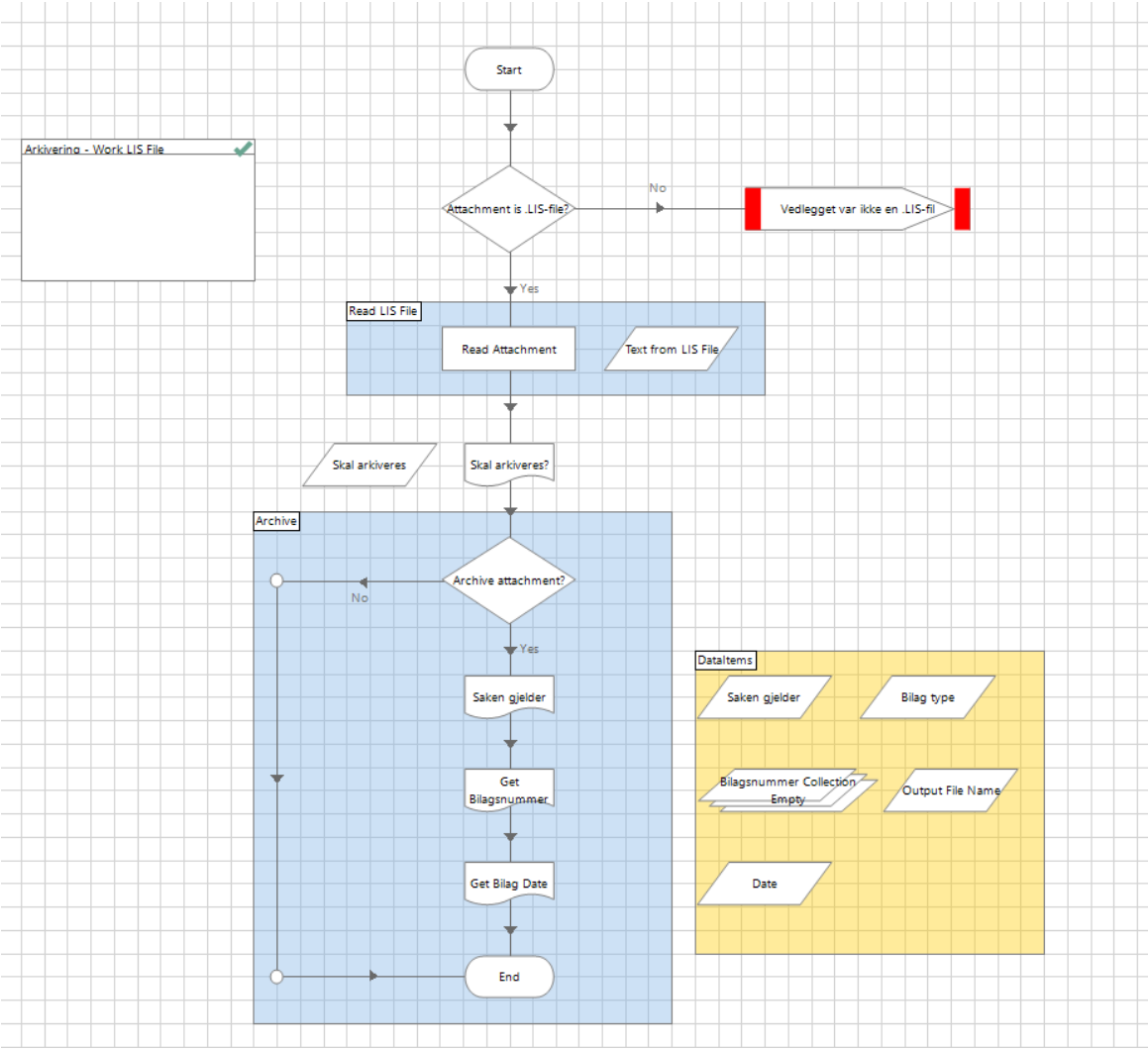
F.7.2 Behandle E-post (Work E-mail)



Figur F.4: Side med logikk for å behandle e-post

Arkiv-roboten behandler e-postene ved å først lagre vedleggene i e-posten. Deretter henter den ut nødvendig informasjon fra vedleggene, som er av typen .lis. Så vil roboten sjekke om bilagsvedlegget er av typen feilliste. Hvis det stemmer, vil den sende en mail med vedlegget til DFØ. Hvis det ikke er feilliste, vil roboten filtrere informasjonen den har hentet ut med Excel-filen slik at den kan finne ut hvor i mappestrukturen filen kan lagres. Hvis det ikke har oppstått unntak og vedlegget er blitt lagret på riktig sted, vil roboten flytte e-posten til "Arkiv"-innboksen i Outlook og merke saken som vellykket i arbeidskøen.

F.7.3 Behandle bilag (Work LIS-file)



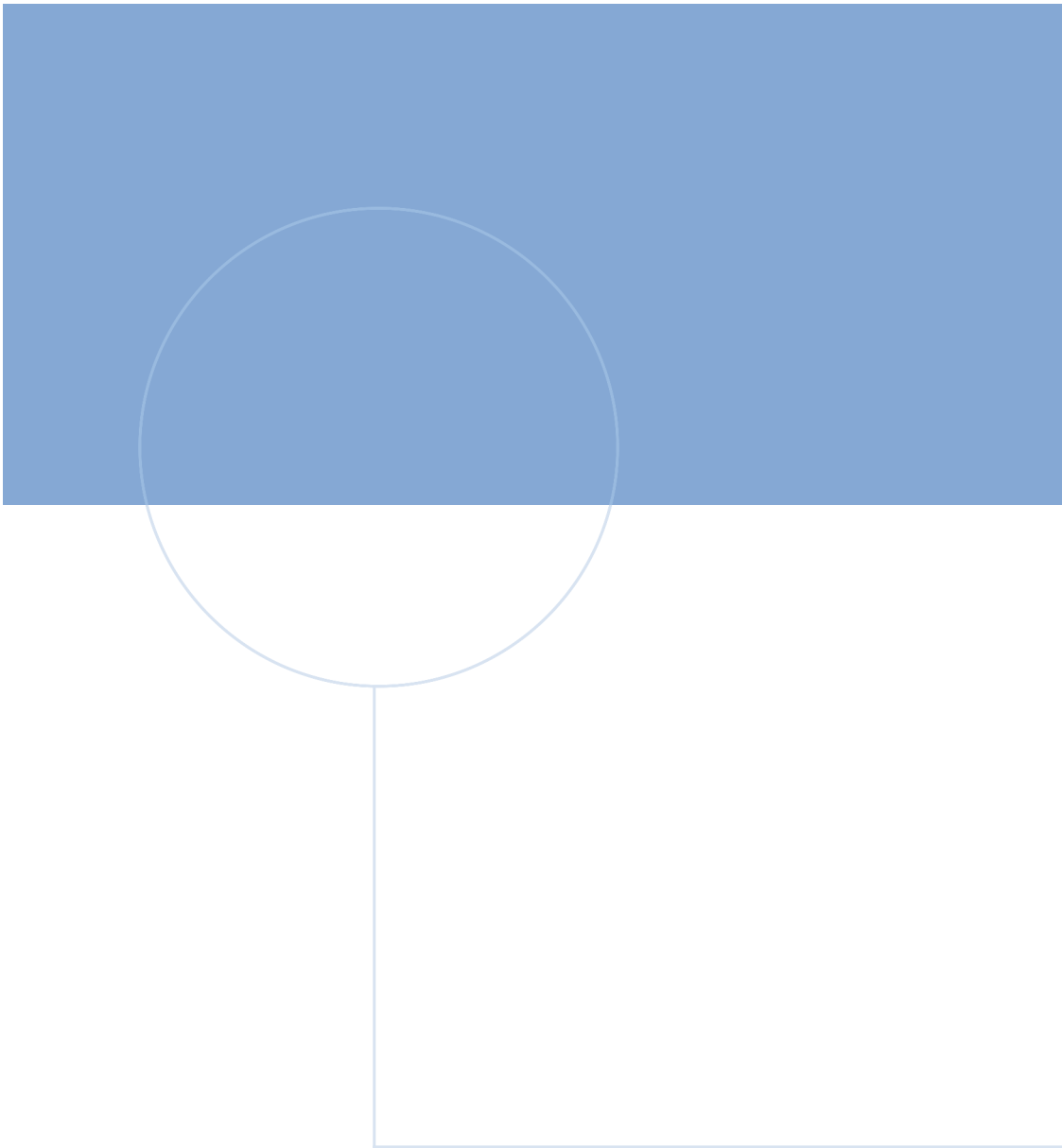
Figur F.5: Side med logikk for å behandle bilag

Arkiv-roboten behandler bilaget som er vedlagt i e-posten. Først sjekker den om vedlegget

er en .LIS fil, leser innholdet, og sjekker om innholdet inneholder ordet "feillinje". Dersom det er en feillinje markeres "Skal arkiveres" som false, og roboten hopper over all videre innlesning. Dersom det ikke er en feillinje så skal bilaget arkiveres og den leser da også inn hva saken gjelder, bilagsnummer og dato.

F.8 Testing

Se kapittel C.8.



NTNU
Kunnskap for en bedre verden

Skatteetaten