

MOKKA RPA

Projektmodel for implementering af RPA

Oktober 2018

Udarbejdet af

René Ingemann Andersen
IT-Konsulent

Mokka RPA

Indholdsfortegnelse

1. Forord	4
1.1. Metode	5
1.2. Afgræsning	6
1.3. Terminologi	7
1.4. Læsevejledning	8
2.1. RPA-Projekt	10
2.2. RPA-Program	11
2.3. Tre typer af RPA-organisering	11
2.4. Oversigt over modellens faser og komponenter	13
3. Faser	18
3.1. Plan	18
3.2. Design	20
3.3. Kode	23
3.4. Test	25
3.5. Release	28
3.6. Stabilisering I <i>stabiliseringsfasen</i> er der behov for udvidet support og overvågning af RPA-processen. I løbet af <i>stabiliseringsfasen</i> vil der opstå færre og færre driftsfejl og fejlrettelser efterhånden, som RPA-processen bliver mere modnet. <i>Stabiliseringsfasen</i> er typisk 30 dage. I den periode har RPA-teamet fortsat ansvaret for at tjekke, at RPA-processen virker, og RPA-teamet er i tæt dialog med brugerne for at tjekke resultaterne af hver kørsel.	30
3.7. Drift	32
3.8. Monitor I <i>monitorfasen</i> skal driften løbende overvåge logfiler for fejl og advarsler. Brugere skal holde øje med transaktionsloggen og IT-systemerne, så de fortsat ser korrekte ud. Kundens ledelse skal holde øje med ledelsesrapporten – går tingene, som forventet.	33
4. Komponenter	35
4.1. Product Backlog	35
4.2. Automatiseringsscreening	36
4.3. Kanban tavle	37
4.4. Done	41
4.5. Testplan	41
4.6. Opgavebeskrivelse	42
4.7. Løsningsbeskrivelse	43
4.8. Konfigurationsstyring	44
4.9. Udvikling	45
4.10. Driftsvejledning	46
4.11. Testaktiviteter	47
4.12. Gennemførelse	49
4.13. Rapportering	49
4.14. Organisation	50
4.15. Accepttest	51

4.16. Overvågning	52
4.17. Fejlrettelser	52
5. Teknikker	54
5.1. Walkthrough	54
5.2. Inspektionsreview	54
5.3. Instruktionsdækning	56
5.4. Kodestandart	57
5.5. Test-Driven Development	58
6. Møder	60
6.1. Retrospektiv	60
6.2. Stand-up	61
6.3. Interview og gennemgang af arbejdsgang	61

1. Forord

I dag implementerer mange virksomheder Robotic Process Automation (RPA) i et stort omfang. RPA kan være med til at effektivisere processer, frigive ressourcer, øge kvalitet og være medvirkende til, at der opstår færre fejl. Men det er vigtigt, at man som virksomhed er opmærksom på, at RPA også har sine begrænsninger, og at mange af implementeringerne mislykkes helt eller delvist.

Det sker ofte på grund af, at:

- Virksomheder skal have hurtige resultater og forretningsgevinster, så planlægning og kvalitetskontroller bliver ikke implementeret.
- Når virksomheden endelig har fået robotterne i produktion, så forventes det, at alt er godt. Intet kunne være længere fra sandheden. Der er behov for en hel styringsmodel for implementeringen.

RPA er ikke bare lige sådan at implementere. RPA-teknologien er nem at benytte, men teknologien skaber også forandringer, der skal håndteres for at realisere det fulde potentiale ved RPA-teknologi. Det er nødvendigt med en struktureret og prioriteret indsats for at lykkes med implementeringen.

Derudover er det væsentligt at holde for øje, at et RPA-projekt også er et IT-projekt - og som alle andre IT-projekter og forandringer - så er det helt essentielt at kontrollere og styre den iboende risiko, der er forbundet med forandring og fornyelse, for at kunne levere RPA-projekter med succes. RPA-projekter er kendetegnet ved at være af kortere varighed end almindelige IT-projekter, men man skal gennem de samme faser, som man skal med et traditionelt IT-projekt:

- Analyse
- Design
- Udvikling
- Test
- Produktion
- Overvågning og support

Min tanke er derfor at prøve at udvikle en projektmodel, som tager udgangspunkt i de moderne udviklingsformer – i dette tilfælde Agile DevOps-processer - som jeg vil forsøge at lægge henover de nødvendige RPA-processer. Modellen tager udgangspunkt i erfaringer fra snesevis af RPA-projekter, der i nogle tilfælde har bidraget med dyrt købte erfaringer og i andre tilfælde med succesfulde oplevelser.

Erfaring viser nemlig, at mange virksomheder har generelle principper og visioner, som guider deres udvikling af initiativet, men kun få af dem har en nedskrevet styringsmodel, som følges - en definering og afklaring omkring RPA-strategien er et af de vigtigste punkter for en succesfuld implementering af RPA. Hvor RPA forankres, og hvordan organisationen derudover tilpasses for at inddrage RPA, afhænger blandt andet af virksomhedens opbygning og kultur, hvilket også vil blive berørt i introduktionen til projektmodellen.

Denne projektmodel egner sig dermed til at skabe et solidt fundament og skal ses som et defineret arbejdsredskab til implementering af RPA-løsninger.

Formål

Formålet er at konceptualisere en ny projektmodel gennem inddragelse af udefra kommende inputs, undersøgelser vedrørende implementeringen af RPA samt indsamling af Best Practice. Projektmodellen vil løbende blive revideret og afprøvet på virkelige projekter, og der vil også blive indsamlet viden herfra.

Projektmodellen skal være et praktisk arbejdsredskab for RPA-projektdeltagere, så man ved hjælp af skabeloner og forklaring til anvendelse af skabeloner kan implementere RPA mere struktureret.

Projektmodellen stilles til rådighed som open source, så alle er velkommen til at kopiere og arbejde videre med skabeloner og dokumenter. Det ville dog være rart med feedback på nye ting, som kan indarbejdes i projektmodellen til alles interesse.

Nu er rejsen startet mod at finde en projektmodel til RPA-projekter, der kan hæve succesraten og samtidig øge kvaliteten, når vi udvikler nye RPA-processer.

1.1. Metode

Projektmodellen tager udgangspunkt i kendte og succesfulde IT-projektmodeller. Som udgangspunkt er følgende valgt:

- DevOps
- Test-Driven Development (TDD)
- Agil projektstyring (Scrum og Kanban)

Devops

Formålet med at inddrage DevOps er, at der skal være et tæt samarbejde mellem udviklingen af RPA-processerne og de driftsaktiviteter, som sker i forbindelse med afviklingen af disse RPA-processer. Begge parter skal være enige om dette. Det vil fremme kommunikationen og samarbejdet samt sikre bedre kvalitet gennem strukturerede processer.

Test-Driven Development

TDD hjælper udvikleren med at indtænke test fra starten. Der er fokus på test, inden koden skrives. Desuden vil TDD give sikkerhed for, at ændringer ikke har ødelagt kendt funktionalitet eller reintroduceret tidligere fejl.

Agil projektstyring

I agile projekter handler det om at udnytte det naturlige flow i et projekt, hvor man løbende bliver klogere. Komplexiteten er ofte så høj, at man ikke kan gennemskue løsningen på forhånd. Det bygger på gennemprøvet tankegang, som samtidig er god at kombinere med viden om RPA-udvikling og viden om den organisation, man arbejder i for i sidste ende at skabe sig et godt udgangspunkt for arbejdet med RPA.

1.2. Afgræsning

Det er ikke meningen, at projektmodellen skal omfatte alle emner, der er relevante i forbindelse med implementering af RPA-projekter. Aspekter vedrørende ledelse og projektledelse, som er omfattet af eksisterende og gennemprøvede metoder, holdes uden for denne projektmodel.

Projektmodellen omfatter ikke specialistteknikker, der bruges i forbindelse med konkrete RPA-værktøjer, fx hvordan man bedst laver TDD i det valgte værktøj. Pilotprojekt forløb og anskaffelsesforløb ligger heller ikke inden for selve projektmodellens område.

1.3. Terminologi

I dette afsnit vil der blive præsenteret og præciseret en række af de termer, som vil blive brugt løbende og som har betydning for forståelsen af projektmodellen.

Bruger defineres som den person eller gruppe, som er ansvarlig for resultatet af den færdige RPA-proces. Det vil sige, dem som er ansvarlige for at RPA-processen overholder de forretningsregler, som findes for en korrekt løsning af opgaven.

Kunde repræsenterer den person eller gruppe, som har bestilt arbejdet, og som vil få udbytte af de endelige resultater.

Arbejdsgang er de tastetryk og kontroller, som en medarbejder indtaster og læser fra et eller flere IT-systemer. Arbejdsgang beskriver hele processen med at udføre den pågældende forretningsopgave.

Arbejdsopgave er de opgaver, som skal løses af RPA-teamet.

Produkt anvendes til at beskrive, alt det et projekt skal skabe eller ændre. Produktet er fx en automatiseret arbejdsgang eller en opgavebeskrivelse, som beskriver arbejdsgangen.

RPA-projekt er en omlægning af en manuel arbejdsgang til en automatiseret arbejdsgang. Et RPA-projekt har en livscyklus, som er den vej og den rækkefølge, som gennem de forskellige faser fører frem til det endelige produkt.

RPA-program er en samling af RPA-projekter, hvis samlede resultater medfører en række udbyttegivende omlægninger af manuelle arbejdsgange.

RPA-proces er en arbejdsgang, som er omlagt til automatisering og sat i produktion.

KPI er en forkortelse af Key Performance Indicators. KPI'er er en styringsmekanisme, der hjælper en virksomhed med at vurdere, hvor godt det går med at nå de opsatte mål. KPI'er giver information om, hvordan RPA-processerne performer i forhold til målsætningerne. Det er vigtigt, at KPI'er er målbare, det kan fx være et antal automatisk betalte fakturaer og det antal fakturaer, som fortsat skal manuelt behandles. Hvis antallet af de manuelt behandlede fakturaer er for højt, så er investeringen i omlægning til RPA måske slået fejl.

Programkommissorium udarbejdes af virksomheds-/programledelsen inden RPA-projektets start og udgør den oprindelige motivation, baggrund og det idégrundlag, som ledelsen skal vurdere projektets eksistens ud fra. Der skal være henvisninger til vigtige strategier og politikker, som alle RPA-projekter skal overholde, fx GDPR- eller testpolitikker. Programkommissorium skal ses som begrundelsen for projektet, som kan være resultatet af en forskellige ting, fx ny forretningsmodel, en ny strategi, reaktionen på ændret lovgivning, anbefalinger fra en rapport eller en revisionsgennemgang.

Systemtest er at teste en færdigudviklet RPA-proces for at verificere, at den overholder specificerede krav, og dermed sandsynligvis vil kunne overtage en manuel arbejdsgang.

Accepttest er en formel test i forhold til brugerbehov, krav og forretningsprocesser udført for at fastlægge, om en RPA-proces opfylder acceptkriterierne, og som gør det muligt for brugerne at afgøre, om RPA-processen kan accepteres og sættes i produktion.

Center of Excellence (CoE) er et center, hvor virksomheden har samlet alle eksperter. Centret kan hjælpe med at standardisere og genbruge virksomhedens erfaringer og Best practice. CoE gør det muligt at styre alle ressourcer mere effektivt for at sikre kvalitet på tværs af RPA-projekter og opbygge en kultur, der fokuserer på kvalitet og samarbejde.

1.4. Læsevejledning

Dette dokument er udformet som en komplet oversigt over RPA-projektmodellen.

I introduktionen til projektmodellen præsenteres de grundlæggende principper, og hvordan de hænger sammen indbyrdes.

Faseafsnittet beskriver procesmodellen og forklarer, hvordan man bedst styrer de enkelte faser ved at kombinere og anvende komponenter og teknikker.

Afsnittet om komponenter forklarer og beskriver de vigtigste elementer i projektmodellen, fx *Kanban tavle*, *konfigurationsstyring*, og hvordan disse elementer indgår i projektmodellen. Disse

komponenter repræsenterer "råstofferne" til god implementering, inklusive kvalitets- og risikostyring.

Sidst forklares om nogle af de teknikker, som skal beherskes i et RPA-projekt. Afsnittet er mest målrettet de udviklere og testere, der skal arbejde med teknikkerne.

Ord skrevet med *kursiv* er enten faser, komponenter, teknikker eller møder.

2. Introduktion til Projektmodel

Denne projektmodel henvender sig til personer, som vil få en rolle i et RPA-projekt, eller som ønsker at bidrage til, hvordan et RPA-projekt kan forløbe.

2.1. RPA-Projekt

Nedenstående tegning viser de faser et RPA-projekt skal gennem i projektmodellen. Faserne er sat op som et DevOps-forløb:



Plan: Det vurderes om arbejdsgangen med fordel kan omlægges.

Design: Her beskrives hvordan arbejdsgangen skal omlægges.

Kode: Her udvikles koden som omlægger arbejdsgangen.

Test: Her testes det om koden virker.

Release: Her flyttes koden fra Test-miljøet til produktionsmiljøet.

Stabilisering: Her holdes ekstra øje med den omlagte arbejdsgang.

Drift: Her er arbejdsgangen omlagt til normal drift.

Monitor: Her overvåges og gives notifikationer, hvis der opstår problemer med driften af den omlagte arbejdsgang.

Afslutningsvis beskrives de møder, som afholdes i forbindelse med et RPA-projektforløb.

2.2. RPA-Program

Projektmodellen skal understøtte en tvær-organisational organisation, som er nødvendig for at producere foruddefinerede resultater på et foruddefineret tidspunkt ved hjælp af foruddefinerede ressourcer.

Derfor er et RPA-program karakteriseret ved følgende:

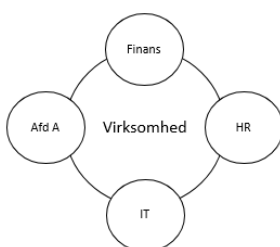
- Definerede og målbare forretningsprodukter.
- Et sammenhængende sæt aktiviteter, det er nødvendige for at levere forretningsprodukterne.
- En defineret ressourcemængde.
- En organisationsstruktur med definerede ansvarsområder til ledelse af programmet
- Et RPA-program indeholder et eller flere projekter. Et projekt kan være enkeltstående, eller det kan være en omlægning af en række af beslægtede arbejdsgange.
- Et projekt har en livscyklus, som er den vej og den rækkefølge, som gennem forskellige aktiviteter fører frem til det endelige produkt – omlægning af en manuel arbejdsgang til en automatiseret arbejdsgang.
- Et projekts livscyklus omfatter faserne fra analyse af egnethed og udformning af produktet gennem test og levering til drift. Denne projektmodel dækker hele projektets livscyklus plus en del af den forberedelse, der ligger før projektet samt især support og vedligeholdelse, der ligger efter projektet.

2.3. Tre typer af RPA-organisering

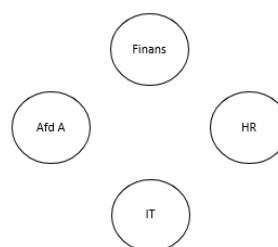
Den centrale organisering



Den hybride organisering



Den decentrale organisering



Projektmodellen skal kunne anvendes som grundlag for et **Center of Excellence** (CoE). Det betyder, at projektmodellen understøtter tre forskellige typer af RPA-organiseringer:

Den centrale organisering, hvor alle RPA-teamets deltagere er samlet det samme sted, og alle samarbejder på alle arbejdsopgaver uanset hvilken afdeling, de kommer fra.

Fordele: Det er lettere at sikre kontroller, standarder, vagtordninger og kvalitetssikring.

Ulemper: Teamet arbejder med de arbejdsopgaver, som har højest prioritet. Dermed kan fordelingen af opgaver måske tilfalde én afdeling mere end en anden afdeling. Desuden kan det være vanskeligt at samle RPA-teamet, hvis de fortsat arbejder på opgaver i deres egen afdeling.

Den hybride organisering, hvor hver afdeling har sit eget RPA-team. Projektlederen er fortsat ansvarlig for alle teams, men der findes flere Kanban tavler - et for hvert RPA-team.

Fordele: Deltagerne kan både arbejde med RPA-arbejdsopgaver og andre opgaver i afdelingen. Afdelingerne får så meget fokus, som de ønsker ved at allokere egen ressourcer. Få ressourcer = få arbejdsgange bliver automatiseret.

Ulemper: Styringen er meget vanskeligere. Det er svære at få vagtordninger, så en enkelt medarbejder ikke både skal udvikle nye arbejdsopgaver og holde fokus på driftsproblemer.

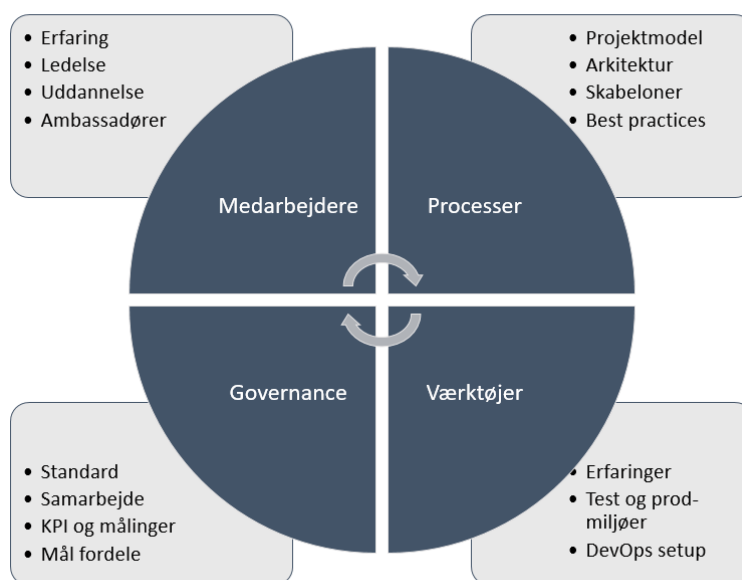
Den decentrale organisering, hvor hver afdeling har sit eget RPA-team, projektleder og definitioner af, hvordan man skal løse en opgave.

Fordele: Beslutninger og arbejdsgange kan være en del lettere, da afdelingslederen selv kan tage beslutninger og har ansvaret for resultaterne, gevinsterne og omkostningerne.

Ulemper: Ofte kommer der dårligere kvalitet samarbejde og manglende standarder, som ofte fører til kaos og projekter, som aldrig bliver gennemført.

Det er op til virksomheden at beslutte hvilken organisering, som de ønsker at gennemføre RPA-projekterne i, men projektmodellen er den samme uanset hvilken organisering, der vælges.

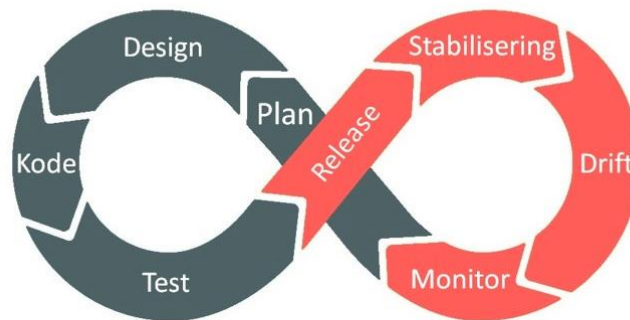
Når der skal tages beslutning om organisering, kan følgende metode anvendes til at se på hele organiseringen. Virksomheden skal tage beslutninger på alle fire områder for at få et vellykket RPA-projekt:



2.4. Oversigt over modellens faser og komponenter

Projektmodellen indeholder en fasebaseret metode. Faserne definerer de aktiviteter, der skal udføres i løbet af projektet. Desuden beskriver projektmodellen en række komponenter, som anvendes inden for de relevante aktiviteter i faserne. Endelig findes en del teknikker og møder, som understøtter projektmodellen.

Nedenstående figur viser komponenter, som er anbragt rundt om den centrale projektmodel.



Komponenter

Product Backlog, Automatiseringsscreening, Kanban tavle, Done

Teknikker

Møder

Retrospektiv, Stand-up

Faser

Projektmodellen består af otte afgrænsede faser, der omfatter aktiviteterne fra igangsættelse over styring og overvågning af fremdrift til idriftsættelse og support. Ved ethvert RPA-projekt skal der tages hensyn til hver af disse faser. I dette afsnit vil der være en kort introduktion til projektmodellens forskellige faser, yderligere uddybning vil være at finde senere hen (se afsnit 3).

Alle faser i projektmodellen er hele tiden aktive. Faserne kommer i samme rækkefølge som arbejdsgangene en efter en bliver omlagt.

Faser i projektmodellen

Plan: Den første fase er *planlægningsfasen*. Her udvælges og analyseres de arbejdsgange, som kan være relevante at omlægge.

Design: I *designfasen* nedbrydes arbejdsgangen i mindre arbejdsopgaver, og det defineres, hvordan opgaven bør løses. Desuden udarbejdes test cases og test data.

Kode: Her udvikles de enkelte mindre arbejdsopgaver, og koden testes. Desuden udarbejdes vedligeholdelses- og driftsdokumentation.

Test: Det samlede projekt testes fra start til slut. Der rettes fejl, og projektet testes igen.

Release: Projektet overdrages til driftsorganisationen, som er ansvarlig for at sætte projektet i produktion.

Stabilisering: Der er en periode, efter at et projekt eller en større ændring er sat i produktion, hvor man må forvente flere fejl og problemer.

Drift: I denne fase er der fokus på effektiv og løbende drift af de RPA-processer, som er sat i produktion.

Monitor: I denne fase skal RPA-processerne overvåges og supportes i drift. Desuden skal KPI'er for de enkelte RPA-processer overvåges.

Arbejdsgangene går fra fase til fase, men RPA-teamet arbejder hele tiden i alle faser for at flytte opgaverne (arbejdsgange) fra en fase til en anden. Det betyder, at man hele tiden er i *planfasen*, hvis kunderne kommer med nye idéer til arbejdsgange, som bør omlægges. Man venter ikke til et bestemt tidspunkt, som fx i Scrum, hvor man kun kigger på nye opgaver i forbindelse med Sprint planlægningen.

Komponenter

Projektmodellen ligeledes af en række forskellige *komponenter* – i alt 16 *komponenter*. *Komponenterne* skal ses som hjælpeværktøjer under de forskellige faser. Nedenstående afsnit giver en kort introduktion til de 16 *komponenter*, som indgår i projektmodellen. Senere hen vil *komponenterne* blive yderligere uddybet (se afsnit 4), hvor det beskrives, hvilken indflydelse en bestemt *komponent* har i projektmodellen, og afsnittet giver vejledninger til, hvordan de forskellige *komponenter* skal anvendes.

Komponenter i projektmodellen

Product Backlog: Er første skridt, når der modtages et kundeønske om omlægning til RPA, det kan fx være et Excel-regneark. Det er en prioriteret liste af mulige arbejdsgange.

Automatiseringsscreening: Identificerer automatiseringspotentialer i en arbejdsopgave - består både af forretningsmæssige- samt udviklingsmæssige potentialer.

Kanban tavle: Anvendes til at implementere agile arbejdsformer. *Kanban Tavlen* er med til at sikre, at medarbejdere holder fokus på det arbejde, der aktivt er i gang.

Done: Definitionen af *done* og benyttes til at vurdere, hvornår arbejdet er færdigt i de enkelte faser.

Testplan: Det er et dokument, der beskriver, hvordan den omlagte arbejdsgang skal testes, så det sikres, at arbejdsgangen behandler alle de funktioner og data, som forventet.

Opgavebeskrivelse: Et dokument, som indeholder skærmbilleder og tastetryk som dokumentation for, hvordan flowet er i arbejdsgangen.

Løsningsbeskrivelse: Her skal RPA-udvikleren gøre sig tanker om, hvordan arbejdsgangen bør implementeres.

Konfigurationsstyring: Det er gældende for alle faser i projektmodellen og er kontrol af dokumenter og kode i alle faser.

Udvikling: Med udgangspunkt i *opgavebeskrivelsen* og *løsningsbeskrivelsen* startes selve *udviklingen* af kode i det valgte RPA-værktøj. RPA-udvikleren kan tage udgangspunkt i en virksomhedsudviklet kodeskabelon eller starte forfra.

Driftsvejledning: Den danner grundlaget for, at driftsorganisationen er i stand til sikkert og stabilt at installere og driftsafvikle den pågældende RPA-proces.

Testaktiviteter: I testfasen gennemføres en systemtest. Systemtesten skal vise, om hver funktion er til stede og fungerer. Derudover udarbejdes en række testcases samt en testprocedure.

Gennemførelse: Her afvikles de testcases, som er planlagt i testproceduren.

Rapportering: Når testfasen har opfyldt sine kriterier, så kan testen, og dermed RPA-processen, godkendes af testeren. Her udarbejdes en testopsommeringsrapport.

Organisation: En gennemgående komponent i hele projektmodellen. Det er afgørende for projektets succes, at der bliver etableret en effektiv, organisatorisk struktur.

Accepttest: Formålet med denne test er at give brugerne tillid til, at den automatiserede RPA-proces kan overtage arbejdsopgaven.

Overvågning: Her skal driften holde øje med, om der skrives fejl, advarsler eller andet i RPA-processens logfiler.

Fejlrettelser: Hvis der sker afvigelser i forbindelse med driften af en RPA-proces, så skal der ske en registrering af en fejlrapport.

Teknikker og mødetyper er en væsentlig del af projektmodellen, men spiller ikke samme rolle som faser og komponenter. Derfor vil en yderligere beskrivelse af teknikker og møder først være at finde senere hen.

Forbindelser mellem faser, komponenter, teknikker og møder

Det er ofte svært at forstå de vigtigste relationer mellem faser, komponenter, teknikker og møder. I hvilke faser benyttes hvilke komponenter, teknikker og møder?

Nedenstående tabel kan være med til at give et overblik over disse forbindelser:

Fase	Komponent	Teknik	Møde
Overordnet styring			Retrospektiv, Stand-up
Plan	Automatiseringsscreen, Product Backlog, Kanban tavle, Done		Analysemøde
Design	Opgavebeskrivelse, Testplan, Kanban tavle, Done	Walkthrough, Inspektionsreview	Interview og gennemgang af arbejdsgang, Teknisk review, Kundepræsentation og review
Kode			
Test			
Release			
Stabilisering			
Drift			
Monitor			

3. Faser

Hver fase i projektmodellen skal ses som et vigtigt led i forsøget på at implementere RPA-processer med succes. I nedenstående afsnit beskrives hver fase nærmere.

3.1. Plan

Plan er den første fase i projektmodellen. Denne fase skal sikre, at forudsætningerne for omlægning er til stede. Fasen forudsætter eksistensen af en ledelsesbeslutning, som definerer de overordnede begrundelser for, at RPA-omlægningen skal gennemføres samt hvilke produkter, der bør omlægges og prioriteres, fx tidsbesparelser, kvalitetsforbedringer m.m. Omlægningen til RPA begynder så snart, denne fase er udført, og prioriteringen er så høj, at den er godkendt i henhold til styregruppens programkommissorium.

Der skal være et grundlæggende forretningsbehov til at igangsætte et projekt. Før omlægningen af arbejdsgangen begynder, og før der afsættes ressourcer, skal følgende spørgsmål kunne besvares: "*Har vi et levedygtigt og rentabelt projekt?*". Dette spørgsmål skal besvares gennem en *automatiseringsscreening*, så det kan sikres, at de afsatte ressourcer ikke går til spille.

Forudsætninger

Fasen forudsætter en række forudsætninger, sat af ledelsen, som kan forklare baggrunden for RPA-programmet, samt de forventede resultater er beskrevet, og roller i projektet er besat.

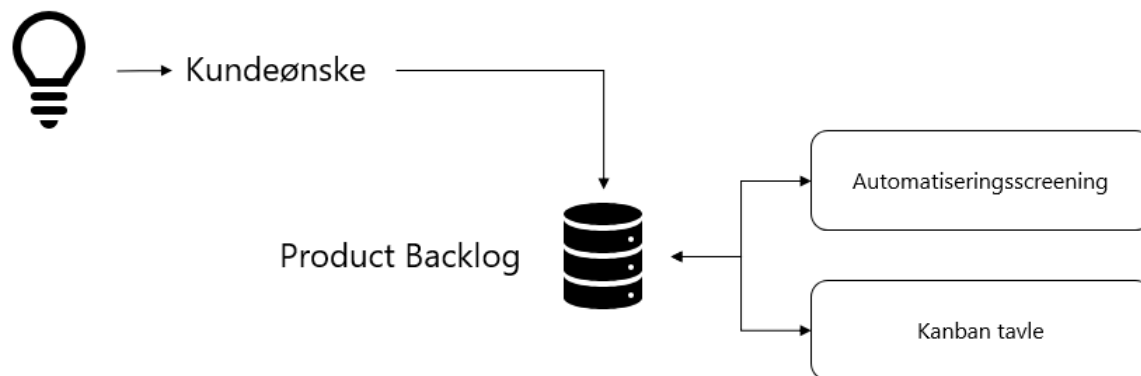
Der er følgende forudsætninger:

- Programkommissorium
- Definition af *done* for de enkelte faser
- Roller i projektet er besat

Der kan ikke foretages noget arbejde i projektet før programkommissorium, og "Done" er defineret, og de vigtigste roller er besat. Nogen skal "kickstarte" projektet og træffe de første beslutninger. Desuden forventes det, at den organisation, projektet arbejder i, bliver informeret om eksistensen og betydningen af det nye projekt.

Procesbeskrivelse

Nedenstående figur viser de processer, som findes i *planfasen*:



Arbejdet i fasen er bygget op omkring tre processer:

- Kunden opretter et omlægningsønske i *Product Backloggen*.
- *Automatiseringsscreening* anvendes til dels:
 - At foretage en forretningsscreening af forretningspotentialer ved omlægning af arbejdsgangen fra manuel til automatiseret.
 - At foretage en udviklingsscreening af de tekniske muligheder for omlægning af den pågældende arbejdsgang.
- *Kanban tavle* anvendes dels til:
 - At foretage en prioritering, som sikrer, at kun arbejdsgange med høj prioritering placeres på *Kanban tavlen*.
 - *Kanban tavlen* fortæller RPA-teamet, hvilke opgaver de skal arbejde med og sikrer, at RPA-teamet holder fokus på de vigtigste opgaver.

Ansvar

Ansvar for denne fase og tilhørende processer ligger hos projektlederen.

Informationsbehov

Produkt	Input/Output til processen	Beskrivelse
Programkommissorium	Input	Overordnet grundlag for alle beslutninger i projekterne.

Projektorganisation	Input	De enkelte roller er besat i henhold til roller for et RPA-team.
Kundeønske	Input	Signal til igangsætning af <i>planfasen</i> og tilhørende processer.
<i>Product Backlog</i>	Output	Beskrivelse og prioritering af kundens ønske.
<i>Kanban tavle</i>	Output	Hvis kundens ønske er prioriteret og der er plads på <i>Kanban tavlen</i> , flyttes opgave over på To-Do listen.

Nøglekriterier

- Adgang til og rådighed over medarbejdere, der forstår og kan udføre den nuværende manuelle arbejdsgang.
- Har de udvalgte medlemmer af RPA-teamet de fornødne færdigheder og den viden, som kræves, for at de kan udføre deres opgaver.

3.2. Design

Designfasen skal sikre, at forudsætningerne for at anvende flere ressourcer i *kodefasen* er til stede. Fasen forudsætter eksistensen af en arbejdsopgave på To-Do listen. I *designfasen* udarbejdes en detaljeret beskrivelse af, hvordan RPA-teamet kommer i mål med omlægningen.

I *planfasen* blev det vurderet, om arbejdsopgaven er et "*levedygtigt og rentabelt projekt*". I *designfasen* skal dette understøttes ved, at der udarbejdes et detaljeret designdokument, som i detaljer beskriver, hvordan arbejdsgangen skal omlægges. Opgavebeskrivelsen skal besvare spørgsmålet: "*Hvordan kan vi omlægge den pågældende arbejdsgang til en automatiseret arbejdsgang?*". Dette spørgsmål skal besvares gennem interview af brugere, udarbejdelse af opgavebeskrivelse, vurdering af tests samt review og godkendelse af opgavebeskrivelsen, så *kodefasen* kan igangsættes.

Når arbejdsopgaven flyttes fra To-Do listen til *designfasen* på

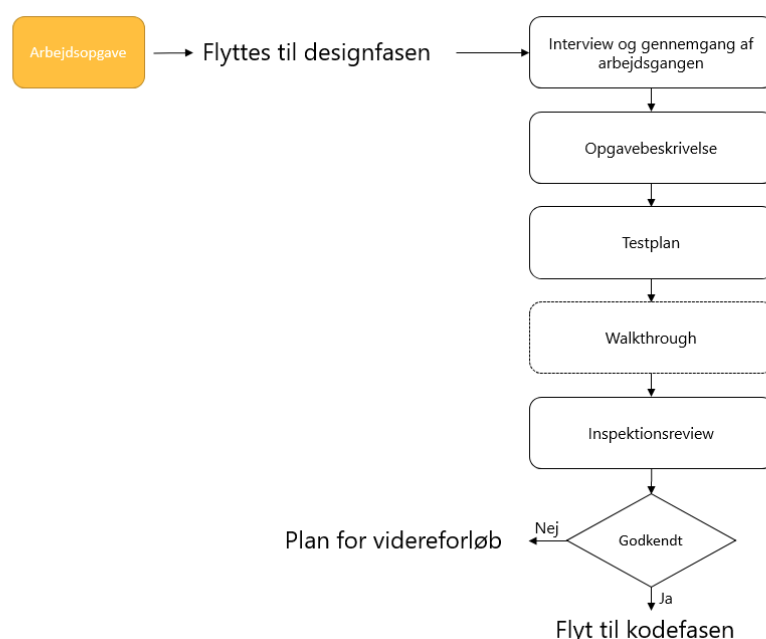
Kanban tavlen, er det besluttet, at RPA-teamet skal bruge sine ressourcer på at få arbejdsopgaven flyttet hele vejen gennem RPA-projektfaserne. Arbejdsopgaven er nu topprioritet på *Kanban tavlen* og kan kun stoppes, hvis grundlaget for arbejdsopgavens prioritet ikke længere er til stede.

Forudsætninger

Fasen forudsætter at der findes en arbejdsgang på *Kanban tavlens* To-Do liste.

Procesbeskrivelse

Nedenstående figur viser de processer, som findes i *designfasen*:



Arbejdet i fasen er bygget op omkring fem processer:

- Der afholdes et *interview og gennemgang af arbejdsgangen*-møde, hvor brugerne gennemgår den manuelle arbejdsgang for RPA-deltageren. Mødet skal bruges til at overføre viden fra brugerne og give RPA-deltageren mulighed for at stille opklarende spørgsmål. Målet er, at RPA-deltageren forstår opgaven, og selv vil være i stand til at kunne udføre den manuelle arbejdsgang ved hjælp af dokumentation skabt under mødet.
- *Opgavebeskrivelse* er udarbejdelse af et designdokument, som i detaljer beskriver arbejdsgangen helt ned til felter og skærm billeder i de enkelte IT-systemer.

- *Testplan* er en plan for, hvordan den automatiserede arbejdsgang skal testes, så det sikres, at denne vil virke i produktion.
- Et møde som *walkthrough* er ikke obligatorisk, men anbefales. *Walkthrough* er et review af *opgavebeskrivelse* og en *testplan*, hvor RPA-deltageren gennemgår dokumenter for én eller flere fra RPA-teamet. Dermed sikres overdragelse af viden samt kvalitetssikring af dokumenterne.
- Afslutningsvis afholdes mødet *inspektionsreview*, som er et review med kunden og brugerne for at sikre, at *opgavebeskrivelsen* og *testplan* opfylder kundens behov og opfylder det grundlag, som arbejdsopgaven blev startet på.

Ansvar

Ansvar for denne fase og tilhørende processer, ligger hos den RPA-deltager, som er ansvarlig for arbejdsopgaven.

Informationsbehov

Produkt	Input/Output til processen	Beskrivelse
Arbejdsopgave	Input	Arbejdsopgave fra <i>Product Backloggen</i>
Opgavebeskrivelse	Output	Designdokument, som beskriver hvordan arbejdsgangen løses.
Testplan	Output	<i>Testplanen</i> beskriver, hvordan den omlagte arbejdsgang skal testes.
Godkendelse	Output	Formel godkendelse fra kunden af <i>opgavebeskrivelse</i> og <i>testplan</i> .
Kanban tavle	Output	Ved godkendelse flyttes arbejdsopgaven til <i>kodefasen</i> .

Nøglekriterier

- Adgang til og rådighed over medarbejdere, der forstår og kan udføre den nuværende manuelle arbejdsgang.
- RPA-deltageren får adgang til de IT-systemer, hvor arbejdsgangen finder sted.

- RPA-deltageren skal i *testplanen* foretage en vurdering af, om der findes et egnet testmiljø til udvikling og test, eller om udviklingen og testen er nødsaget til at ske i produktionsmiljøet - i givet fald, hvordan dette kan ske med begrænset risiko.

3.3. Kode

I *kodedefasen* skal RPA-udvikleren udvikle arbejdsopgaven fra en manuel til en automatiseret via det RPA-værktøj, som virksomheden har valgt.

Grundlæggende principper

En vellykket udvikling bør være baseret på følgende principper:

- Alt udvikling bør ske i henhold til virksomhedens kodestandard.
- Udviklingen bør ske så tæt på Test-Driven Development som muligt.
- Udvikling bør tage højde for fejlsituationer og uforudsete hændelser og dermed skabe det nødvendige fundament for effektiv og stabil drift.
- Kode bør reviewes for at sikre ensartethed og kvalitet.

Følges disse principper, sikres en større sandsynlighed for et vellykket RPA-projekt.

Kontekst

Kodedefasen sigter mod at udvikle den kode, der er nødvendig for at opfylde *opgavebeskrivelsen*. Den følger efter *designfasen* og igangsættes ved godkendelse af *opgavebeskrivelsen*.

Forudsætninger

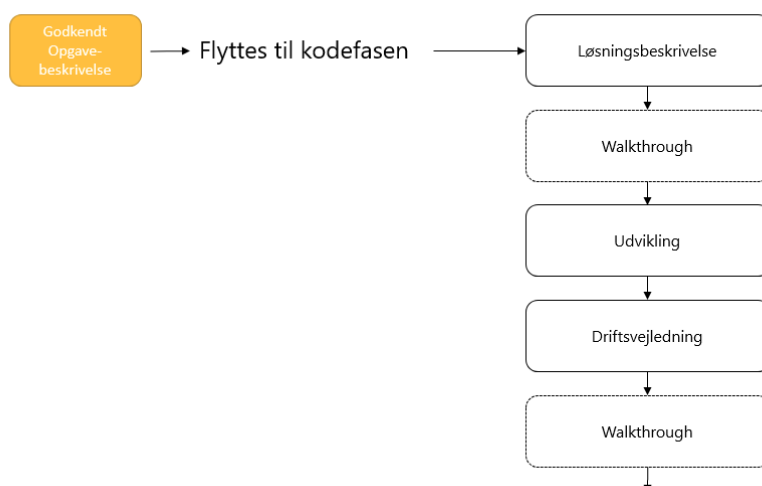
Der er følgende forudsætninger i *kodedefasen*:

- Godkendt *opgavebeskrivelse*

Desuden forventes det, at det IT-miljø, som skal anvendes ved udviklingen, er på plads. Derudover skal RPA-udvikleren have adgang og rettigheder til IT-miljøet, så der kan udføres de opgaver, som arbejdsgangen kræver.

Procesbeskrivelse

Nedenstående figur viser de processer, som findes i *kodedefasen*:



Arbejdet i fasen er bygget op omkring fem processer:

- *Løsningsbeskrivelse* udarbejdes på baggrund af den godkendte *opgavebeskrivelse*. *Løsningsbeskrivelsen* er et teknisk dokument, som beskriver hvordan opgaven skal løses rent teknisk.
- *Walkthrough* er et teknisk review af *løsningsbeskrivelsen* gennemgået af én eller flere af RPA-teamets deltagere. Deltagerne skal have det tekniske overblik for at kunne foretage en teknisk sparring med RPA-udvikleren.
- *Udvikling* er den komponent, der er selve kodningen i RPA-værktøjet. Det anbefales i vidst muligt omfang, at RPA-teamet anvender teknikken, Test-Driven Development, i forbindelse med udviklingen. Det giver en væsentlig bedre kode, og gør det lettere at vedligeholde kode fremover.
- *Driftsvejledning* er et dokument, som indeholder installations- og driftsinstrukser til driften.
- Mødet walkthrough er et teknisk review af *kode-og driftsvejledning*. På samme måde som *walkthrough* af *løsningsbeskrivelsen*, så kræves det, at RPA-deltagere har et teknisk niveau til et sådant review.

Ansvar

Ansvaret for denne fase og tilhørende processer ligger hos RPA-udvikleren.

Informationsbehov

Produkt	Input/Output til processen	Beskrivelse
Godkendt Opgavebeskrivelse	Input	<i>Opgavebeskrivelsen</i> indeholder en detaljeret gennemgang af den manuelle arbejdsgang.
Løsningsbeskrivelse	Output	<i>Løsningsbeskrivelsen</i> indeholder en teknisk nedbrydning af opgaven, så selve udviklingen først starter efter et bedre overblik.
Kode	Output	Koden, som bliver produceret i RPA-værktøjet.
Driftsvejledning	Output	Vejledning til driften, som hjælper dem med, hvordan RPA-processen skal installeres og driftes.

Nøglekriterier

- Det er vigtigt, at IT-miljøet i udvikling og produktion er ens, ellers kommer der rettelser i forbindelse med idriftsættelse og igen, når processen skal tilbage i udviklingsmiljøet for fejlrettelser.
- Det er vigtigt, at RPA-udviklerens rettigheder og rolle er identisk mellem udviklings- og produktionsmiljøet.
- RPA-udvikleren skal have fokus på at gøre processen stabil og kunne håndtere nedbrud, uden at data går tabt, eller data bliver forkert i de underliggende IT-systemer.
- RPA-udvikleren skal forsøge at lave sin kode så vedligeholdelsesvenlig som muligt og følge alle retningslinjer, fx kodestandarder, *konfigurationsstyring* m.m.

3.4. Test

Udgangspunktet for testen er en RPA-proces med en færdig og udviklertestet arbejdsgang. Målet er nu at teste, om RPA-processen er komplet og korrekt. Det gøres i praksis ved at gennemgå de enkelte funktioner, som RPA-processen er bygget op af og sikre sig, at der arbejdes med de rette data i RPA-processen.

Grundlæggende principper

Testfasen skal verificere RPA-processens adfærd, både ud fra funktionelle og ikke-funktionelle krav. Her søges efter fejl, men fokus er på, at RPA-processen kan bruges til formålet og på, at den opfylder kravene.

Testen bør gennemføres i et testmiljø, der ligner produktionsmiljøet.

Det er svært for den, som har udviklet RPA-processen at opdage fejl i sin egen software. For at finde disse fejl bør koden testes af en anden end den, der har udviklet den. Testen kan gennemføres af en testspecialist eller en anden ressource fra RPA-teamet.

Kontekst

Testfasen begynder, så snart kodefase er afsluttet, og testressourcen er ledig. Koden skal være meldt færdig fra udvikleren, og alle RPA-processens produkter skal være tjekket ind i *konfigurationsstyringen*.

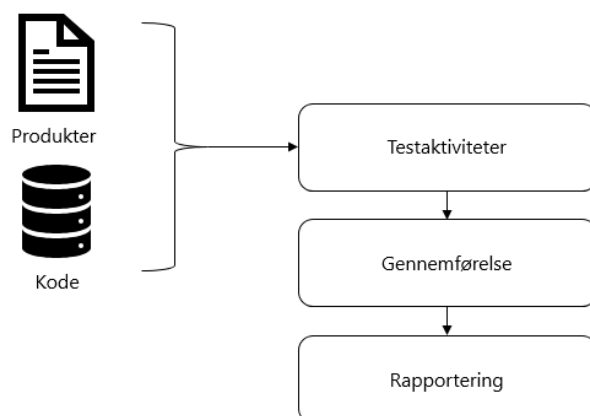
Forudsætninger

Der er følgende forudsætninger:

- Testmiljø er til rådighed, som ligner produktionsmiljøet.
- Tester er til rådighed.
- Udvikler har meldt kode og produkter klar til test.
- Tester har adgang og rettigheder til IT-systemerne i testmiljøet.

Procesbeskrivelse

Nedenstående figur viser de processer, som findes i *testfasen*:



Arbejdet i fasen er bygget op omkring tre processer:

- *Testaktivitet* er en komponent, som opdeler testen i testaktiviteter med udgangspunkt i *opgavebeskrivelse*, *løsningsbeskrivelsen* og kode. Virksomheden kan have udarbejdet overordnede testpolitikker, som også bør følges i forbindelse med udarbejdelse af testforløbet. Ved *testaktivitet* planlægges testen og hvilke testcases, der bør afvikles.
- Komponenten *gennemførelse* er den faktiske afvikling af alle testcases og sammenligning mod de forventede resultater. Der skrives eventuelt fejlrapporter, hvis der findes afvigelser.
- I Komponenten Afslutning vurderes det, om kriterierne for at afslutte testfasen er opnået. Der udarbejdes et slutdokument, og RPA-processen godkendes til videreforløb i *releasefasen*.

Ansvar

Ansvaret for denne fase og tilhørende processer ligger hos testeren.

Informationsbehov

Produkt	Input/Output til processen	Beskrivelse
Programkommissorium	Input	I programkommissorium beskrives krav til test og kriterier for godkendelser.
Opgavebeskrivelse	Input	Testeren kan her se den manuelle arbejdsgang.
Testplan	Input	Testeren kan her se de indledende overvejelser omkring testen.
Løsningsbeskrivelse	Input	Testeren kan her se de beslutninger om kodedesignet, RPA-udvikleren har truffet.
Kode	Input	Testeren kan her se, hvordan koden er struktureret.
Testcases	Output	Testcases til afprøvning om koden virker som forventet.

Testprocedure	Output	Afviklingsplan for hvordan de enkelte testcases skal afvikles.
Fejlrapport	Output	Dokumentation af en funden afvigelse i forhold til forventet resultat.
Slutrapport	Output	Dokumentation af forløbet i testfasen og eventuel godkendelse af RPA-processen.

Nøglekriterier

- Testeren bør have et detaljeret kendskab til test og testteknikker.
- Testeren bør have et detaljeret kendskab til RPA-problematikker for bedre at kunne teste og finde fejl i en RPA-proces.
- Testmiljøet bør indeholde realistiske data, så testen kan forløbe på de samme betingelser som i produktionsmiljøet.

3.5. Release

I *releasefasen* skal RPA-processen flyttes til produktionsmiljøet. RPA-processen overgår nu til driftsafvikling, og *releasefasen* er den første fase i driften af RPA-processen. I denne fase skal det sikres, at RPA-processen installeres korrekt i produktionsmiljøet.

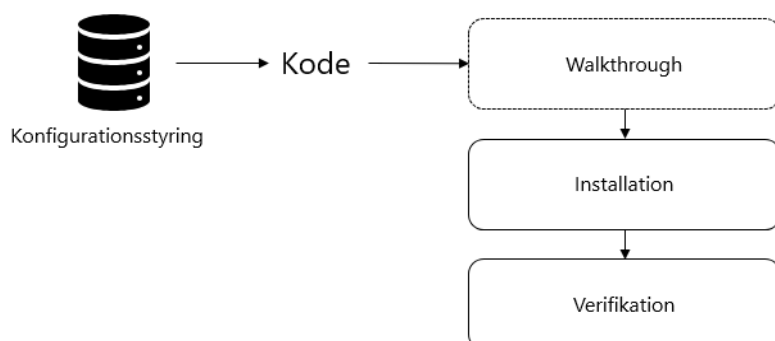
Forudsætninger

Der er følgende forudsætninger:

- RPA-processen skal være godkendt i *testfasen*.
- Produktionsmiljøet skal være etableret og klart.

Procesbeskrivelse

Nedenstående figur viser de processer, som findes i *releasefasen*:



Arbejdet i fasen er bygget op omkring tre processer:

- *Walkthrough* benyttes til at foretage et review af *driftsvejledningen* for at sikre, at denne vil kunne installeres, driftes og overvåges på tilfredsstillende vis i produktionsmiljøet. Dette er en valgfri proces.
- *Installation*, som er en del af driftsvejledningen, sikrer, at kode flyttes fra *konfigurationsstyringssystemet* til produktionsmiljøet. RPA-værktøjet kan indeholde metoder til dette, så hvordan dette gøres afhænger af virksomhedens RPA-værktøj valg.
- *Verifikation*, som også er en del af *driftsvejledningen*, sikrer, at den kode, som er installeret i produktionsmiljøet, er korrekt installeret. Dette kan eventuelt tjekkes manuelt, eller det kan være et afviklet script, der tjekker, om koden virker tilfredsstillende.

Ansvar

Ansvar for denne fase og tilhørende processer ligger hos en medarbejder med viden om RPA-infrastruktur.

Informationsbehov

Produkt	Input/Output til processen	Beskrivelse
Driftsvejledning	Input	Her kan RPA-infrastrukturmedarbejderen læse hvordan RPA-processen installeres og

		overvåges.
Kode	Input	Koden til den RPA-proces, som skal installeres i produktionsmiljøet.

Nøglekriterier

- *Konfigurationsstyring* af miljøerne er vigtig, så det ikke er nødvendigt at ændre i koden for at flytte RPA-processen fra testmiljøet til produktionsmiljøet.
- RPA-infrastrukturmedarbejderen skal have kendskab til det valgte RPA-værktøj - problemstillinger og arkitektur.

3.6. Stabilisering

I *stabiliseringsfasen* er der behov for udvidet support og overvågning af RPA-processen. I løbet af *stabiliseringsfasen* vil der opstå færre og færre driftsfejl og fejlrettelser efterhånden, som RPA-processen bliver mere modnet. *Stabiliseringsfasen* er typisk 30 dage. I den periode har RPA-teamet fortsat ansvaret for at tjekke, at RPA-processen virker, og RPA-teamet er i tæt dialog med brugerne for at tjekke resultaterne af hver kørsel.

I denne fase skal RPA-processen gøres så stabil, at driften og brugerne efterhånden overtager overvågningen af den daglige drift.

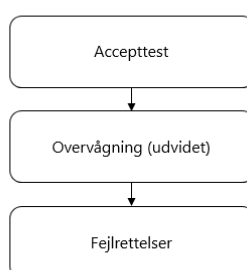
Forudsætninger

Der er følgende forudsætninger:

- RPA-processen er korrekt installeret i produktionsmiljøet.

Procesbeskrivelse

Nedenstående figur viser de processer, som findes i *stabiliseringsfasen*:



Arbejdet i fasen er bygget op omkring tre processer:

- *Accepttest* er RPA-teamets overdragelse af RPA-processen til brugerne. I *accepttest* skal RPA-udvikleren vise brugerne, at RPA-processen virker og vise brugerne, hvordan de holder øje med, om RPA-processen virker.
- *Overvågning* (udvidet - hvilket betyder, at det fortsat er RPA-teamets ansvar at tjekke disse logfiler og sikrer, at driften virker efter hensigten) er den daglige overvågning af logfiler, transaktionslogs og ledelsesrapportering.

Komponenten *fejlrettelser* er en proces til at fange, registrere, rette og idriftsætte fejl, som er opstået i forbindelse med den daglige drift af RPA-processen.

Ansvar

Ansvaret for denne fase og tilhørende processer ligger hos RPA-udvikleren.

Informationsbehov

Produkt	Input/Output til processen	Beskrivelse
Godkendelse	Output	Formel godkendelse fra Kunden til at sætte RPA-processen i produktion.
Fejlrapport	Output	Afvigelse registreret i forbindelse med driften.

Nøglekriterier

- I forbindelse med opstarten af RPA-processen, er det ofte en fordel, at processen kun behandler et begrænset antal transaktioner, fx kun at behandle 10 fakturaer pr. kørsel. Hvis noget går galt, så er de efterfølgende rettelserne ikke så store, som hvis RPA-processen har behandlet 1000 fakturaer.
- Det kan være en fordel at "pakke" ændringer af data i IT-systemerne ind, så programmet kan køre uden at ændre data. Det kan være godt i forbindelse med test i et produktionsmiljø.
- Hvis en RPA-proces fejler, så kan det være en fordel at tage et skærmdump af IT-systemets skærbillede på det tidspunkt, hvor fejlen opstod. Det gør det lettere for RPA-udvikleren at finde fejlen.

- Skriv i logfiler på "rigtige" steder i koden. Det hjælper også med fejlfinding af driftsproblemer.

3.7. Drift

Driftsfasen indeholder alle hardware og softwarekomponenter, som er nødvendig for, at RPA-processerne kan afvikles – uanset om det er i udvikling-, test- eller produktionsmiljøet.

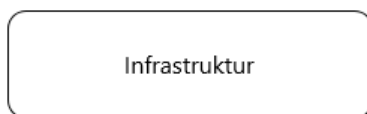
Forudsætninger

Der er følgende forudsætninger:

- Ingen

Procesbeskrivelse

Nedenstående figur viser de processer, som findes i *driftsfasen*:



Arbejdet i fasen er bygget op omkring en proces:

- Processen *infrastruktur* er vedligeholdelse af hardware og software til afvikling af RPA-processer. *Infrastrukturen* er forskellig afhængig af, hvilket RPA-værktøj virksomheden har valgt. Processen skal dog varetage processerne og supporten omkring følgende områder:
- Server management
- Netværksadministration
- Opgraderinger af software og hardware
- Backup og restore
- Ressourceovervågning
- Adgange og rettigheder

Ansvar

Ansvar for denne fase og tilhørende processer ligger hos medarbejderen med viden om RPA-infrastruktur.

Informationsbehov

Produkt	Input/Output til processen	Beskrivelse
---------	----------------------------	-------------

Nøglekriterier

- Ingen

3.8. Monitor

I *monitorfasen* skal driften løbende overvåge logfiler for fejl og advarsler. Brugere skal holde øje med transaktionsloggen og IT-systemerne, så de fortsat ser korrekte ud. Kundens ledelse skal holde øje med ledelsesrapporten – går tingene, som forventet.

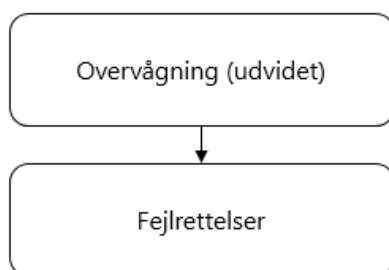
Forudsætninger

Der er følgende forudsætninger:

- RPA-processen er i produktion

Procesbeskrivelse

Nedenstående figur viser de processer, som findes i *monitorfasen*:



Arbejdet i fasen er bygget op omkring to processer:

- Komponenten *overvågning* er den daglige *overvågning* af logfiler, transaktionslogs og ledelsesrapportering. Det er driftsorganisationens ansvar at

tjekke systemlogfiler, brugerne skal tjekke transaktionslogfilen, og ledelsen hos kunden skal tjekke ledelsesrapporteringen.

- *Fejlrettelser* er en proces til at fange, registrere, rette og idriftsætte fejl, som er opstået i forbindelse med den daglig drift af RPA-processen.

Ansvar

Ansvaret for denne fase og tilhørende processer ligger hos driftsorganisationen og kunden.

Informationsbehov

Produkt	Input/Output til processen	Beskrivelse
Systemlogfiler	Input	RPA-processen skriver fejl og advarsler i systemlogfiler i forbindelse med afviklingen af den automatiserede arbejdsgang.
Transaktionslog	Input	Brugerne kan se hvilke transaktioner, der er blevet behandlet af RPA-processen samt status på, om transaktionen efterfølgende skal manuelt behandles.
Ledelsesrapportering	Input	Kundens ledelse er ansvarlig for løbende at tjekke, om det overordnede mål med omlægning af arbejdsgangen er opfyldt/opnået
Fejlrapport	Output	Afvigelse registreret i forbindelse med driften.

Nøglekriterier

- Det er vigtigt at RPA-processen løbende overvåges. RPA-processen skal anses, som en "medarbejder". Der skal være ledelse og kontrol, ellers løber tingene let ud i ingenting.

4. Komponenter

Hver *komponent* beskrives nærmere i dette afsnit. Det beskrives, hvilken indflydelse en bestemt *komponent* har i projektmodellen, og afsnittet giver vejledninger til, hvordan en *komponent* skal anvendes.

4.1. Product Backlog

Første skridt, når projektlederen modtager et kundeønske, er at oprette kundeønsket i en *Product Backlog*. En *Product Backlog* kan fx være et Excel-regneark eller et IT-system, som indeholder de arbejdsgange, der eventuelt skal løses i forbindelse med RPA-programmet.

Product Backloggen vil i løbet af planfasen udvikle sig til en raffineret og prioriteret liste af mulige arbejdsgange. Alle disse arbejdsgange er samlet i *Product Backloggen*, så alle interessenter kan se hvilke arbejdsgange, der er og deres prioritet.

***Product Backloggen* indeholder:**

- En prioriteret liste over manuelle arbejdsgange, som ønskes omlagt til automatisering.
- Resultaterne af *automatiseringsscreeningerne*.
- Synlige lister for alle interessenter.
- Mulighed for omprioritering, som løbende gøres af projektlederen.

Hver enkelt arbejdsgang i *Product Backloggen* kaldes en *Product Backlog*-arbejdsopgave - den laves ud fra de emner, som forretningen har et ønske om at automatisere.

***Product Backloggen* bør indeholde følgende informationer:**

- Entydig identifikation.
- Navn på arbejdsopgaven.

- Kort beskrivelse af arbejdsopgaven.
- Navn på den organisation, som bestiller arbejdsopgaven (fx afdelingsnavn).
- Navn på bestiller, som er ansvarlig for arbejdsopgaven (fx den som skal betale).
- Navne på medarbejdere, som kan udføre den manuelle arbejdsgang - dem som RPA-teamet kan kontakte for at få informationer om arbejdsgangen.
- Informationer fra *automatiseringsscreeningen*.
- Prioritet, som er beregnet ud fra *automatiseringsscreeningen*.
- Status på arbejdsopgaven (ny, prioritet, *Kanban*, produktion, afventer).

Arbejdsopgaver, som er i produktion bør fjernes fra *Product Backloggen* og eventuelt placeres i et andet Excel-regneark.

4.2. Automatiseringsscreening

En *automatiseringsscreening* identificerer automatiseringspotentialer i en arbejdsopgave. Screeningen skal både indeholde forretningsmæssige potentialer, samt hvor udviklingsegnet den pågældende arbejdsopgave er til RPA-omlægning.

Forretningsscreening bør indeholde følgende informationer:

- Hvilke systemer/applikationer tilgår arbejdsgangen?
- Hvor mange skærbilleder indeholder arbejdsgangen?
- Hvilke inputs og outputs har arbejdsgangen, fx læses en e-mail fra en indbakke?
- Tidsforbrug i minutter pr. transaktion.
- Antal transaktioner pr. måned.
- Hvor ofte skal arbejdsgangen afvikles, fx skal den køres hverdage inden kl. 9?
- Hvordan igangsættes arbejdsgangen? Er det manuelt besluttet eller på bestemte tidspunkter?
- Tages der kun regelbaseret beslutninger? Det vil sige, at en computer vil kunne kodes til at tage de samme beslutninger.
- Er der andre grunde til omlægningen? Fx øges kvaliteten.
- Forventes der større planlagte ændringer i de systemer/applikationer, som arbejdsgangen arbejder i inden for de næste 6 måneder?
- Nødvendige brugerrettigheder i de enkelte IT-systemer for at kunne udføre arbejdsgangen.

På baggrund af ovenstående, skal der udregnes en forretningsmæssig prioritering af projektet ved omlægning.

Udviklingsscreening bør indeholde følgende informationer:

- Er input struktureret, så et program vil kunne forstå og behandle disse inputs?
- Er output struktureret, så et program vil kunne forstå og danne disse outputs?
- Hvilke data inputs indgår i arbejdsgangen (pdf, billeder, Excel osv.)?
- Hvilke data outputs indgår i arbejdsgangen (pdf, billeder, e-mail osv.)?
- Kræver det indlæsning af indscannede dokumenter for at løse arbejdsopgaven?
- Kræver det indlæsning af håndskrevet tekst eller fritekst for at løse arbejdsopgaven?
- Hvilke teknologier skal anvendes for at tilgå systemerne (Citrix, Remote, Web, Desktop)?
- Er der et egnet testmiljø til rådighed til udvikling og test af arbejdsgangen?

På baggrund af ovenstående, skal der udregnes en udviklingsmæssig prioritering. Derefter kan man vælge at gange den forretningsmæssige prioritet med den udviklingsmæssige prioritet og dermed få en samlet prioritering.

Forretningsmæssig prioritering x udviklingsmæssig prioritering = den samlede prioritering.

4.3. Kanban tavle

Kanban er en metode, der anvendes til at implementere agile arbejdsformer - typisk i IT-projekter. *Kanban tavlen* skal sikre, at RPA-teamet kun er fokuseret på det arbejde, der aktivt er sat i gang. Når teamet afslutter en opgave, tages den næste prioriterede opgave fra To-Do listen.

Ved at bruge metoden *Kanban* er det hele teamets ansvar at sikre, at arbejdet bevæger sig problemfrit gennem alle faser.

Multi-tasking dræber effektiviteten. Jo flere arbejdsopgaver der er i gang på samme tid, jo mere skal der skiftes mellem arbejdsopgaverne. Hver gang der foretages et skift til en anden arbejdsopgave, blokerer dette for, at den foregående arbejdsopgave kan blive afsluttet. Det er derfor et centralt element i *Kanban* at begrænse mængden af igangværende arbejdsopgaver samtidig (WIP Work In Progress). WIP-grænser synliggør flaskehalse og understøtter teamets bestræbelser på at reducere

gennemløbstiden ved at have fokus på at færdiggøre arbejdsopgaven, fjerne forhindringer samt opbygge overlappende kompetencer.

Kanban principper:

- Visualiser workflow.
- Begrænse antal igangværende arbejdsopgaver (WIP).
- Måle tiden i de forskellige faser (cyklustid/gennemløbstid).
- Kontinuerlig læring og forbedring med henblik på at reducere cyklustid/gennemløbstid, forbedring af kvalitet m.m.
- Teamet laver selv arbejdsplaner og koordinerer indbyrdes, hvordan de vil løse arbejdsopgaverne. Teamet har i deres planlægning fokus på at levere på den kortest mulige tid - og hvis blokeringer opstår - at hele teamet forsøger at løse opgaven.

Eksempel

To-Do (5)	Design (3)	Kode (3)	Test (5)	Release (5)	Stabilisering (10)
Afklaring (2)					

På kolonnen med To-Do må teamet kun have fem nye arbejdsopgaver, da WIP er sat til fem ((5) ved To-Do overskriften). Hvis listen er under fem arbejdsopgaver, så tages opgaven med den højeste prioritet fra *Product Backloggen* og placeres på To-Do listen.

Når en deltager fra RPA-teamet tager opgaven, så flyttes den gule seddel fra To-Do kolonnen og til kolonnen med design - sådan flyttes arbejdsopgaven gennem alle faser på *Kanban tavlen*. De røde og gule farver forklares yderligere senere i afsnittet. Det er vigtigt at definere *done* for hver kolonne. *Done* betyder, hvornår en arbejdsopgave er færdig i den pågældende kolonne.

Eksempel

En opgave kan ikke flyttes fra *Product Backloggen* til To-Do listen før, der er sket en *automatiseringsscreening*. Arbejdsopgaven skal altså være helt færdig i den foregående fase, før den kan flyttes til en ny fase. Hvis et nyt kundeønske i *Product Backloggen* har højere prioritet end én af de arbejdsopgaver, som står i To-Do listen, så flyttes arbejdsopgaven fra *Product Backloggen* til To-Do listen, og opgaven med lavest prioritet fjernes fra To-Do listen (flyttes tilbage til *Product Backloggen*).





Igangværende opgaver

En arbejdsopgave, som er flyttet til fx *designfasen* (eller andre faser), kan ikke erstattes af en ny arbejdsopgave med højere prioritet - igangværende arbejdsopgaver skal altid afsluttes først.

Det har i praksis vist sig, at det kan være en god idé at lave et sted til de opgaver, som ikke hurtigt kan afklares. Det kunne fx være et spørgsmål, som ikke blev opdaget i *designfasen*, men som er fremkommet under kodning. Medarbejderen med viden til at besvare spørgsmålet er måske ikke til stede, så flyttes opgaven til afklaring på *Kanban tavlen*, og udvikleren tager en anden arbejdsopgave. Udvikleren kan først vende tilbage til den uafklarede arbejdsopgave, når den igangværende arbejdsopgave er færdig.

Eksempel

Hvis teamet ønsker review efter fx *design*, *kode* og/eller *testfaserne* for at sikre, at kvaliteten er i overensstemmelse med aftalerne, så kan man på *Kanban tavlen* opdele disse faser i to kolonner.

To-Do (5)	Design (3)		Kode (3)		Test (5)		Release (5)	Stabilisering (10)
	I gang	Review	I gang	Review	I gang	Review		
								
								
								
Afklaring (2)								
								

På tavlen bliver arbejdsopgaver markeret med en seddel, fx en gul seddel for omlægning af en arbejdsgang og en rød seddel for en fejl eller et driftsproblem i produktion.

På de gule sedler bør man have følgende informationer:

- Navnet på arbejdsopgaven.
- Navnet på kunden - afdelingen som bestiller opgaven.
- Prioritet, så man kan se hvilken prioritet, arbejdsopgaven har anvendes også til at sortere på To-Do listen, så man tager den med højest prioritet først).
- Identifikation af hvem, som er ansvarlig for udførelse af arbejdsopgaven (én fra RPA-teamet). Man kan opdele sedlen yderligere i fx faser, så man kan se, hvem der var ansvarlig i *design*-, *kode*-, *test*- og *release*fasen samt eventuelt tidspunkt, så man kan måle på tiden i de forskellige faser.

På de røde sedler bør man have følgende informationer:

- Navnet på den RPA-proces, som der er driftsproblemer med.
- Tidspunkt for hvornår, problemet skete.
- Prioritet, så man kan vælge den med højest prioritet, hvis der er flere driftsproblemer samtidig.

Som udgangspunkt har fejl og driftsproblemer i produktion højere prioritet end andre igangværende arbejdsopgaver og kan som den eneste undtagelse fjerne en RPA-deltager fra en igangværende arbejdsopgave.

RPA-teamet kan med fordel lave vagtordninger, hvor der allokeres en teamdeltager til at have driftsansvaret for den pågældende dag. Det er så denne teamdeltager, som behandler og løser driftsproblemet, eventuelt med støtte fra andre team deltagere. RPA-deltageren, som har vagtordningen, arbejder på andre arbejdsopgaver, hvis der ikke er driftsproblemer.

På denne måde sikres det, at:

- Viden overdrages, da alle kommer til at kende de forskellige automatiserede arbejdsgange.
- Alle ved, at kvalitet og standard er vigtige, da man hurtigt – og ofte under pres - skal sætte sig ind i andres løsninger og dokumentation.

- De øvrige RPA-deltagere kan fokusere og koncentrere sig om en enkelt arbejdsopgave uden at skulle være på vagt over, om der opstår en fejl fra produktionsmiljøet.

4.4. Done

Når en arbejdsopgave flyttes fra en fase til en anden fase, er det nødvendigt, at alle har den samme opfattelse af, hvad dette betyder. Det er her definitionen af *done* kommer ind. *Done* varierer væsentligt fra RPA-team til RPA-team. Derfor må deltagerne i teamet have en fælles forståelse af, hvad det betyder, at arbejdet er færdigt for at give gennemsigtighed. Definitionen af *done* og benyttes altså til at vurdere, hvornår arbejdet er færdigt i de enkelte faser.

I takt med at RPA-teamet opnår større viden og bliver mere modent, forventes det, at deres kriterier for definitionen bliver mere fokuseret på at levere højere kvalitet.

4.5. Testplan

En *testplan* er et dokument, der beskriver, hvordan den omlagte arbejdsgang skal testes, så det sikres, at arbejdsgangen behandler alle de funktioner og data, som forventet. *Testplanen* er grundlaget for den formelle *accepttest* af den omlagte arbejdsgang.

Testplanen skal beskrive omfanget, tilgangen, ressourcerne og tidsplanen for de planlagte testaktiviteter. Den identificerer blandt andet testelementer (de funktioner, der skal testes, testmiljøet, behov for testdata, hvilke testteknikker, der skal anvendes, hvornår testen kan starte, og hvornår den skal stoppe).

Det er muligt at have flere *testplaner* i projektet (fx integrationstestplan), men i denne projektmodel er testplanen tænkt, som en *accepttest*.

Testplanen er en løbende aktivitet, og den udføres i alle faser og aktiviteter, så længe den automatiserede arbejdsgang er under udvikling. Feedback fra andre aktiviteter anvendes til at erkende forandringer og eventuelt ændrede risici, så *testplanen* skal justeres.

***Testplanen* bør indeholde:**

- Introduktion med overblik over testplanens mål og begrænsninger.
- Testelementer - hvilke testelementer skal testes, og hvilke skal ikke testes og hvorfor?
- Fremgangsmåde for testen samt hvilke testteknikker der bør anvendes.
- Hvornår starter testen, og hvornår stopper testen?
- Testmiljø og testdata skal beskrives.
- Estimering af tidsforbrug og nødvendige ressourcer
- Risici - liste over de risici, der er blevet identificeret, og hvordan de bør omgås.
- Angiv de antagelser og afhængigheder, der er foretaget under udarbejdelsen af denne testplan.

4.6. Opgavebeskrivelse

Opgavebeskrivelsen er et dokument, som indeholder skærbilleder og tastetryk som dokumentation for, hvordan flowet er i arbejdsgangen.

Opgavebeskrivelsen skal være så detaljeret, så den kan overdrages til en RPA-udvikler som på baggrund af dokumentet, vil kunne udvikle den pågældende arbejdsgang i det valgte RPA-værktøj. Her anvender RPA-deltageren vejledninger, instruktioner og videooptagelsen fra *interview og gennemgang af arbejdsgang*-mødet. Det betyder også, at alle skærbilleder, klik, tastetryk, regler og undtagelser skal være beskrevet i *opgavebeskrivelsen*.

Opgavebeskrivelsen skal godkendes af kunden, inden arbejdsopgaven kan flyttes til *kodfasen*.

En opgavebeskrivelse bør indeholde følgende informationer:

- Alle skærbilleder.
- Alle klik og tastetryk for at betjene systemerne.
- Regler og undtagelser.
- Hvilke opgaver overtager RPA-processen, og hvilke opgaver overtages ikke?
- Håndtering af fejl og undtagelser.
- Eventuelle forslag til forandringer i forbindelse med en automatisering.
- Hvilke informationer skal bruges i forbindelse med RPA-processen for at give brugerne et overblik over, hvad RPA-processen har udført? Det kunne fx være en transaktionslog, hvor det noteres præcist hvilke processer, robotten har udført.

- Hvilke ledelsesinformationer skal komme ud af RPA-processen? Det kan fx være antallet af transaktioner, som er behandlet eller antallet af transaktioner, som efterfølgende skal manuelt behandles, således ledelsen kan følge med i udviklingen. Disse ledelsesinformationer danner grundlag for ledelsens vurdering af, om forretningspotentiallet er opnået.
- Hvordan og hvornår skal RPA-processen startes? Det kan fx være online startet, hvor medarbejderne tager beslutningen, eller afvikles batch på samme bestemte tidspunkter.
- Referencer til vejledninger, instruktion og videooptagelse.

4.7. Løsningsbeskrivelse

I *løsningsbeskrivelsen* skal RPA-udvikleren gøre sig nogle tanker om, hvordan arbejdsgangen bør implementeres.

***Løsningsbeskrivelsen* er et dokument, som bør indeholde tanker om implementering i forhold til følgende emner:**

- Hvilke fællesmoduler kan genbruges, hvilke nye fællesmoduler skal udvikles, hvilke moduler skal opgaven nedbrydes i?
- Hvordan beskytter vi IT-systemerne mod nedbrud i RPA-processen? Fx hvis en RPA-proces markerer en faktura, som betalt, og derefter sker et nedbrud af RPA-processen, inden betalingen af faktura faktisk er sket. Nu har vi en faktura, som alle tror er betalt, men der er ikke sendt en betaling til leverandøren.
- Hvordan sikres det, at RPA-koden kan genstartes efter nedbrud og starte det korrekte sted? Eller hvordan sikre det, at brugerne gøres opmærksomme, på at der er et problem?
- Hvordan sikres det, at RPA-processen kan håndtere de forventede transaktionsmængder?
- Det kan være en god idé at sætte grænser for, hvor mange transaktioner RPA-processen skal behandle pr. kørsel.
- Sikkerhed, rettigheder og logning skal have særlig fokus.
- Håndtering af både forretningsundtagelser og kodeundtagelser.

Forretningsundtagelser er transaktioner, som ikke kan behandles på grund af anderledes data, fx fordi transaktion er uden for grænseværdier. Kodeundtagelser er undtagelser, som opstår på grund af koden, fx at "Ok"-knappen ikke kan findes på skærbilledet.

- Hvordan notificeres driften, brugerne og RPA-teamet omkring driften og især ved fejl?
- Er der behov for, at processen skal kunne stoppes udefra, på en kontrolleret måde, inden den faktisk er færdig med alle transaktioner?
- Hvordan sikres en effektiv fejlfinding ved problemer i produktion? Det kan være ved logninger og Screenshot ved fejl.

4.8. Konfigurationsstyring

Konfigurationsstyring er gældende for alle faser i projektmodellen. *Konfigurationsstyring* er kontrol af dokumenter og kode i alle faser. Det er en disciplin, som sikrer præcis kontrol med projektets produkter, fordi RPA-teamet får mulighed for at:

- Specificere versioner af de produkter, der findes og er i brug og opbevare oplysninger om deres status og ændringshistorik.
- Kontrollere produktændringer ved at sikre, at ændringer kun foretages efter aftale.
- Udføre revision på dokumenterne for at sikre, at de indeholder de korrekte oplysninger.

Uden *konfigurationsstyring* har RPA-teamet kun lidt eller ingen kontrol over de produkter, der fremstilles - hvor de er, om de er ændret, og hvad den nyeste version er.

Baseline

En baseline er et øjebliksbillede af et produkt - fastfrosset på et givet tidspunkt til et bestemt formål. Når kunden har godkendt opgavebeskrivelsen, fastfryses den og bliver "baseline". Herefter kan en baseline bruges som et fast grundlag for udviklingen i senere faser.

Hvis produktet senere ændres, forbliver baselineversionen uændret og et nyt versionsnummer skal tildeles. Når denne ændrede version er færdig og har gennemgået kvalitetskontrol, etableres der en ny baseline for den pågældende version.

Gamle baselineversioner kasseres aldrig.

Konfigurationsstyringsmetoden skal altid tillade genskabelse af alle versioner af et givent produkt.

Release

En release er en komplet automatiseret arbejdsgang inklusiv tilhørende produkter, fx opgavebeskrivelse og driftsvejledning. For hvert produkt i en release, skal der være en baseline, så det er klart hvilke versioner af de forskellige dele, der skal inkluderes i en release. De fleste virksomheder har heldigvis ofte implementeret værktøjer til *konfigurationsstyring*.

Konfigurationsstyring bør indeholde:

- Dokumenter: Fx *Product Backlog*, *opgavebeskrivelse*, *løsningsbeskrivelse*, *driftsvejledninger* m.m.
- Kode: Alle moduler og fællesmoduler, som findes i *konfigurationsstyringen*.
- Tjekliste og skabeloner: Alle tjeklister til fx review, kodestandarder, Word-skabeloner m.m. skal også være under *konfigurationsstyring*.

Alle i RPA-teamet skal være ekstra opmærksomme på, at kode og dokumenter ikke indeholder brugernavne og adgangskoder når de tjekkes ind i *konfigurationsstyringen*.

4.9. Udvikling

Med udgangspunkt i *opgavebeskrivelsen* og *løsningsbeskrivelsen* startes selve *udviklingen* af kode i det valgte RPA-værktøj. RPA-udvikleren kan tage udgangspunkt i en virksomhedsudviklet kodeskabelon eller starte forfra.

Det anbefales, at udvikleren anvender teknikken, Test-Driven Development (TDD), som udviklingsmetode. Mange af RPA-værktøjerne kan ikke dette som udgangspunkt, men der kan ofte implementeres arbejdsmetoder, der ligner TDD i RPA-værktøjet alligevel.

Under *udviklingen* skal RPA-udvikleren have fokus på følgende forhold:

- Virksomhedens kodestandard for RPA-udvikling skal følges. Kodestandarden indeholder emner som: Navngivning af variabler, kommentarer, logging, brug af fællesmoduler, udvikling af nyt fællesmodul osv.
- Virksomhedens standard for *konfigurationsstyring* skal følges.

- Sikkerhed skal være i fokus både sikkerhed i form af begrænset rettigheder, brugernavne og adgangskoder, men også sikkerhed for at RPA-processen ikke "ødelægger" data og status i de underliggende IT-systemer.
- RPA-udvikleren skal have fokus at alle kodelinjer, som minimum er blevet dækket ved en test case (se afsnittet om *Instruktionsdækning* under *Teknikker*). Det har nemlig vist sig, at mange RPA-processer sættes i produktion, hvor flere kodelinjer aldrig har været testet. Det betyder, at den første test af kodelinjer sker i produktion!
- Udviklingsmiljøet stemmer overens med produktionsmiljøet. Tænk konfiguration, hvor der er forskelle, fx mapper og filstruktur.

4.10. Driftsvejledning

Driftsvejledningen danner grundlaget for, at driftsorganisationen er i stand til sikkert og stabilt at installere og driftsafvikle den pågældende RPA-proces.

En driftsvejledning bør indeholde følgende informationer:

- Formål med RPA-processen.
- IT-systemer, der indgår i processen.
- Sikkerhed og rettigheder.
- RPA-proces afhængigheder, fx til at andre RPA-processer er afsluttet.
- **Installationsinstruks:**
 - Hvilken RPA-proces skal installeres inkl. versionsnr. på alle produkter, som indgår i RPA-processen.
 - Ændringer i konfigurationsfiler
 - Afhængigheder til IT-miljø
 - Hvordan testes det, om installationen virker?
- **Driftsinstruks:**
 - Schedulering af RPA-processen. Må RPA-processen flyttes med. Er der faste deadlines?
 - Hvad skal overvåges?
 - Hvad skal der gøres ved fejl og succes?
 - Hvilke logs og dokumenter produceres af RPA-processen?
 - Må RPA-processen genstartes og i givet fald hvordan?

Kvaliteten af vejledningen skal være af en sådan beskaffenhed, at RPA-processen kan overdrages til driftsafdelingen, som har kendskab til RPA-problematikkerne, men som ikke har detaljeret kendskab til den pågældende RPA-proces.

Det er RPA-udviklerens ansvar at levere og løbende at opdatere vejledningen i forbindelse med ændringer i RPA-processen.

Installationsinstruksen skal sætte driftsafdelingen i stand til egenhændigt at foretage en installation og verifikation af RPA-processen i produktionsmiljøet.

Driftsinstruksen skal sætte driften i stand til egenhændigt at sikre en sikker og stabil driftsafvikling af RPA-processen. Det skal af driftsinstruksen fremgå, hvordan driften ser, om RPA-processen er afviklet korrekt eller ej. Og ved fejl skal driften vide, hvad der skal gøres, inden fejlen eventuelt skaleres til en vagtordning eller RPA-udvikleren.

4.11. Testaktiviteter

I testfasen gennemføres en systemtest. Systemtesten skal vise, om hver funktion er til stede og fungerer. Alle funktioner i RPA-processen skal testes. Spørgsmålet er, hvor dybt testen skal gå ned i dækningen af hver funktion. Det er umuligt at foretage udtømmende test og dermed finde alle fejl. Det skal derfor besluttes, hvilke funktioner der bør testes bedst.

Systemtesten består af en grundlæggende funktionel test udvidet med bevidste fejlsituationer, fx at stoppe processen midt i afviklingen. Funktioner, som ændrer data i de underliggende IT-systemer, bør have en mere dybdegående test.

Systemtesten deles op i de elementære testaktiviteter, som fx:

- Hent næste faktura fra indbakken.
- Download vedhæftet pdf-fil.
- Skan vedhæftet pdf-fil for fakturaoplysninger.
- Fremfind kunde i økonomisystemet.
- Tjek om faktura allerede er betalt.
- Opret betaling af faktura.

Systemtesten tester systematisk testaktiviteterne, og derved skabes det gode grundlag for den senere *accepttest* med deltagelse af brugerne.

Produkterne (*opgavebeskrivelse, løsningsbeskrivelse*, kode samt eventuelle vejledninger, som brugerne har udarbejdet om den manuelle arbejdsgang) er lige til at bruge i systemtesten. Det gælder fx til grænseværdier og ækvivalensklasser – kort sagt alle de

testteknikker, som danner en veldrevet test. På baggrund af produkterne udarbejdes testcases, som tester de enkelte *testaktiviteter*.

En testcase bør indeholde følgende informationer:

- Nummer: Et unikt nummer.
- Navn: Det kunne fx være "Find kunde i økonomisystem".
- Beskrivelse: Hvad vil vi teste under udførelsen?
- Betingelser: Er der nogle forudsætninger, som skal være til stede?
- Test Data / Input Data: Krav til data i testen.
- Trinnr.: Unikt trinnummer.
- Trin Handling: Hvad skal der gøres?
- Beskrivelse af teststeget og handlingen skal udføres af brugeren for at få resultatet af programmet.
- Forventet resultat: Hjælper testeren med at sammenligne resultatet med det faktiske resultat.
- Faktisk resultat: Giv plads til at testeren kan skrive det faktiske resultat.

Planen for afvikling af alle testcases samles i en testprocedure. Formålet med testproceduren er at specificere de trin, der skal løbes igennem for at afvikle alle testcases.

Testproceduren bør indeholde følgende informationer:

- Formål (hvilke testaktiviteter skal testes)
- Særlige krav (som er nødvendige for at afvikle proceduren)
- Procedure trin:
 - Log (hvordan skal resultaterne logges?).
 - Set-up (hvad skal der gøres, inden proceduren kan afvikles?).
 - Start (hvordan startes afviklingen af proceduren?).
 - Fremdrift (beskriv alle handlinger, der er nødvendige for at afvikle proceduren).
 - Måling (beskriv alle de målinger, der skal foretages undervejs).
 - Afbrydelse (beskriv hvordan testen stoppes hvis der sker noget uventet).
 - Genstart (beskriv hvordan testen genstartes efter en afbrydelse).
 - Stop (beskriv hvordan testen stoppes).
 - Miljø (beskriv nødvendige handlinger, der sætter miljøet klar til start igen).

- Nødplaner (beskriv handlinger, som kan håndtere fejlsituationer under afviklingen).

4.12. Gennemførelse

Testeren skal under denne komponent afvikle de testprocedurer, som er planlagt i testcases. Testeren skal kontrollere, at afviklingen sker i henhold til testproceduren, fx at de korrekte testdata er til rådighed.

Når testprocedure afvikles, skal testeren nøje kontrollere, at det er de forventede resultater. Det sker ved at sammenligne de faktiske resultater med det forventede resultat fra den enkelte testcase.

Hvis resultaterne stemmer overens, er testcasen godkendt. Hvis dette ikke er tilfældet, så skal testeren oprette en fejlrapport. Alle fejl, der findes i testfasen, skal registreres, så der kan ske en opfølgning.

En fejlrapport bør indeholde følgende informationer:

- Unik identifikation.
- Kort beskrivelse med reference til testaktiviteten med testcase identifikation.
- Input.
- Forventet resultat.
- Faktisk resultat.
- Defekter/fejl.
- Dato og tidspunkt.
- Procedure trin: Hvordan genskabes fejlen igen?
- Miljø: Hvor blev testcasen kørt?
- Tester: Navnet på testeren
- Prioritet: Hvad er konsekvensen af fejlen?

De fleste virksomheder har et fejlopfølgingsværktøj, som testeren kan anvende til at oprette og holde status på, om fejlene er rettet eller ej.

4.13. Rapportering

Når testfasen har opfyldt sine kriterier – ofte at alle kritiske fejl er rettet - så kan testen, og dermed RPA-processen, godkendes af testeren.

Der skal udarbejdes en testopssummeringsrapport, som anvendes til løbende *rapportering* og som *rapportering*, når testfasen afsluttes.

Testopssummeringsrapporten bør indeholde følgende informationer:

- Resume af testen.
- Variationer (afvigelser fra planer og procedurer og forklaring af, hvorfor der er afvigelser).
- Sammenhængende vurdering (ift. de aftalte kriterier. Ting, der ikke blev testet tilstrækkeligt fremhæves, og det forklares hvorfor).
- Testens resultater (antal fundne fejl, uløste fejl, løste fejl).
- Evaluering (testaktiviteterne – er de godkendte, eller fejlede de?).

4.14. Organisation

Organisation er en gennemgående komponent i hele projektmodellen. Det er afgørende for projektets succes, at der bliver etableret en effektiv, organisatorisk struktur. Alle projekter har behov for teknisk knowhow, ledelse, styring, kontrol og kommunikation.

Nærværende projektmodel beskriver en fremgangsmåde, der indeholder de processer, der er nødvendige for, at organisationen har en struktur, men endnu vigtigere er det, at projektet bemandes med de korrekte medarbejdere.

Nedenstående er en liste over de vigtige roller i et RPA-program:

- **Styregruppe:** Sikrer, at RPA-projekternes grundlag og forankring i virksomheden.
- **Projektleder:** Har ansvaret for den daglige ledelse af projekterne på vegne af styregruppen inden for de rammer, styregruppen har udstukket. Projektlederen kan godt have en anden rolle i projektet også. En projektleder kan fx også være Tester i projektet.
- **RPA-udvikler:** Designer, udvikler og tester af RPA-processerne.
- **RPA-infrastruktur:** Er ansvarlig for installation og de enkelte RPA-miljøer.
- **RPA-tester:** Er ansvarlig for systemtesten af RPA-processen og kan støtte RPA-udvikleren i forbindelse med *accepttesten*. RPA-testeren er en rolle, som godt kan varetages af fx en RPA-udvikler.

- **RPA-support:** Er first-line support for RPA-processerne i produktion. Rollen kan godt varetages af en RPA-udvikler, fx i forbindelse med vagtordninger.
- **RPA-business Analyst:** Hjælper forretningen med at finde og vurdere nye arbejdsgange, som eventuelt kan automatiseres.

4.15. Accepttest

Formålet med denne test er at give brugerne tillid til, at den automatiserede RPA-proces kan overtage arbejdsopgaven. Den skal gøre kunden sikker på, at idriftsættelse af RPA-processen kan godkendes med tryghed.

I *accepttesten* skal RPA-udvikleren vise, at RPA-processen kan leve op til kundens forventninger.

RPA-processen startes i produktionsmiljø med sikring af, at ingen data ændres, uden at brugerne har godkendt dette- ofte behandles kun én transaktion ad gangen.

Eksempel

En kunde skal have betaling på en faktura. RPA-processen ser således ud:

1. Fremfind kunde.
2. Hent faktura.
3. Opret betaling.
4. Pop-up vindue, *"Jeg er klar til at klikke på opret-knappen"*.

RPA-udvikleren og brugerne kan nu beslutte, at alt ser okay ud. Hvis ikke dette er tilfældet, så stoppes RPA-processen, og den har dermed ikke foretaget ændringer i IT-systemet.

Når testen afvikles, skal brugerne og RPA-udvikleren nøje kontrollere, at det er de forventede resultater, som sker på skærbilleder.

Hvis resultaterne ikke stemmer overens, så skal RPA-udvikleren oprette en fejlrapport. Alle fejl, der findes i *stabiliseringsfasen*, skal registreres, så der kan ske en opfølgning.

Når en eller flere transaktioner er behandlet uden problemer, og brugerne/RPA-udvikleren begynder at få tryghed ved RPA-processen, så kan pop-up vinduerne slås fra, men man vil ofte stadigvæk kun behandle én transaktion ad gangen.

Hvis RPA-processen fortsat virker efter forventningerne, så kan antallet af behandlede transaktioner sættes op.

Der skal udarbejdes en *accepttest* godkendelse, når testfasen afsluttes.

Accepttest godkendelsen bør indeholde følgende informationer:

- Resume af testen.
- Vurdering af testens resultater.
- Testens resultater (antal fundne fejl, uløste fejl, løste fejl).
- Evaluering (er RPA-processen godkendt eller fejlede den?).

Accepttest godkendelsen skal sendes til kunden.

Det kan være en idé i stabiliseringsfasen at sætte de behandlede transaktioner gradvist op, som RPA-processen bliver mere stabil, indtil man når det antal transaktioner, der ønskes ved en kørsel.

4.16. Overvågning

Under *overvågning* skal driftsorganisationen holde øje med, om der skrives fejl, advarsler eller andet i RPA-processens logfiler. Dette kan med fordel automatiseres, så der sendes en incidentrapport eller e-mail, hvis der er kommet fejl eller advarsler i logfilen.

Brugerne skal holde øje med transaktionsloggen, *"Hvilke transaktioner blev behandlet, hvilke skal fortsat manuelt behandles og hvorfor?"*.

Kundens ledelse og styregruppen er ansvarlig for at holde øje med ledelsesrapportering. Hvor mange RPA-processer er i produktion. Lever de op til forventningerne? Det kan fx være i forhold til hvor stor en procentdel, der automatisk kan behandles.

I *stabiliseringsfasen* (*overvågning* udvidet) er RPA-teamet ansvarlig for, at overvågningen sker. Dels skal RPA-teamet selv holde øje med logfiler, transaktionslog og ledelsesrapportering, men de skal også hjælpe driftsorganisationen og brugerne med at lave opfølgning på logfiler.

4.17. Fejlrettelser

Hvis der sker afvigelser i forbindelse med driften af en RPA-proces, så skal der ske en registrering af en fejlrapport.

Fejlrettelser sker, når RPA-processer er i produktion, så dette kan ske fra *stabilisering-og monitorfaserne*. Fejlrapporten skal oprettes i fejlopfølgningssværktøjet, men det er vigtigt, at den også prioriteres og kommer på *Kanban tavlen*.

Når fejlrapporten flyttes fra To-Do listen på *Kanban tavlen* til *designfasen*, så skal RPA-udvikleren tage stilling til følgende:

- Er der behov for drøftelser med brugerne omkring rettelse af fejlen?
- Skal eksisterende dokumenter i designfasen ændres?
- Hvordan kan fejlrettelsen testes, så det kan dokumenteres, at denne virker efter fejlrettelsen?

Når fejlrapporten flyttes fra *designfasen* til *kodefasen*, skal RPA-udvikleren tage stilling til følgende:

- Koden skal rettes - evt. ved at skrive en testcase, som fejler og derefter ændre koden, så testcasen virker.
- Skal eksisterende dokumenter i kodefasen ændres?

Når fejlrapporten flyttes fra *kodefasen* til *testfasen*, så skal RPA-testeren tage stilling til følgende:

- Hvordan tester jeg om rettelsen virker?
- Der skal ske en afvikling af RPA-processen for at tjekke, om rettelsen virker.

Når fejlrapporten flyttes til *releasefasen*, så skal RPA-infrastrukturmedarbejderen tage stilling til følgende:

- Er der ændringer i *driftsvejledningen*, som jeg skal være opmærksom på?
- Har jeg behov for review af fejlrettelsen med RPA-udvikleren?
- Flytning af koden til produktion.
- Evt. ændringer til overvågningen af logfiler.

Når fejlrapporten flyttes til *stabiliseringsfasen*, så skal RPA-udvikleren tage stilling til følgende:

- Ved mindre rettelser fjernes fejlrapporten fra *Kanban tavlen* efter første succesfulde kørsel af den nye kode.
- Ved større rettelser bør fejlrapporten blive stående på *Kanban tavlen* i hele stabiliseringsperioden.

5. Teknikker

Som støtte for projektmodellen anvendes nogle få *teknikker*. I nedenstående afsnit beskrives de anbefalede teknikker i detaljer.

5.1. Walkthrough

Formålet med et *walkthrough* er at se, om et produkt lever op til kravene.

Walkthrough kan både anvendes til at finde fejl og manglende opfyldelse af andre krav, fx dokumentationsstandarder.

En eller flere af RPA-teamets deltagere modtager en invitation fra ophavsmand til at deltage i et *walkthrough review*.

I invitation skal angives følgende informationer:

- Tid og sted.
- Varighed.
- Formålet.
- Link til materiale, som skal behandles ved reviewet.

Walkthrough review:

- Ledes af ophavsmand.
- Ophavsmand gennemgår materialet i henhold til dennes tanker for at opnå en fælles forståelse og indsamling af feedback.
- Deltagere kan eventuelt anvende tjeklister, fx ved gennemgang af dokumentationsstandard.
- Der er ingen dokumentationskrav, fx til referat af mødet. Ophavsmand indarbejder feedback i sit materiale.

Hovedformålet er at diskutere, træffe beslutninger, evaluere alternativer, finde fejl, løse tekniske problemer og at kontrollere overensstemmelse med krav og standarder.

5.2. Inspektionsreview

Inspektionsreview er et mere formelt review, hvor kunden i sidste ende skal være tryk ved at godkende opgavebeskrivelsen og testplanen. Formålet er at inspicere

opgavebeskrivelsen og testplanen så detaljeret, at det giver et væsentlig bidrag til, at det endelige produkt overholder de arbejdsgange, der er påkrævet.

Roller og ansvar

Et *inspektionsreview* indeholder følgende roller:

- **Manager:** Bestemmer udførelsen af reviewet, tildeler tid til brugerne og tager stilling til, om målet med reviewet er nået og i sidste ende, om dokumenterne kan godkendes. Manager er ofte beslutningstager hos kunden.
- **Moderator:** Leder reviewet af dokumenterne, herunder planlægning, afvikling af mødet og opfølgning efter mødet. Typisk er det projektlederen, som tager denne rolle.
- **Ophavsmand:** Forfatteren for de dokumenter, der skal reviews.
- **Reviewere:** Brugere, med specifik forretningsmæssig viden om den manuelle arbejdsgang, der, efter den nødvendige forberedelse, finder og beskriver observationer i de undersøgte dokumenter.
- **Referent:** Dokumenterer alle observationer, problemer og åbne punkter berørt under mødet.

Planlægning

Planlægningen starter med at ophavsmand henvender sig til projektleder og beder om at få gennemført en inspektion. Derefter overtager projektlederen ansvaret. Projektlederen henvender sig til manager og sikrer sig, at der er ressourcer og tid til afholdelse af et *inspektionsreview*.

Mødet indkaldes skriftligt og bør indeholde:

- Klart defineret formål.
- Hvilke personer skal deltage, og hvad er deres roller.
- Sted, tid og varighed.
- Hvordan reviewet forløber.
- Hvad deltagerne bør forberede inden mødet.
- Link til de nødvendige dokumenter.

Afholdelse

Moderator præsenterer dokumenterne afsnit for afsnit. Hvis alle er enige, kan der fortsættes. Hvis der er uenighed, har man som minimum konstateret en uklarhed.

Referenten markerer fejlen/uklarheden i selve dokumentet, og forsyner den med et nummer. Det er referencen til en liste, hvor hver fejl/uklarhed dokumenteres med den tekst, som reviewerne og ophavsmanden enes om.

Moderator er ansvarlig for, at der tages referat, og at mødet forløber sagligt, er effektivt med hensyn til fremdrift, og at det er præget af en god tone.

Opfølgning

Den vigtigste del af opfølgningen er, at ophavsmanden følger op på uklarheder og fejl fra listen. Moderator skal sørge for, at det gøres. Moderator skal også være overbevist om, at fejlene er rettet på betryggende vis. Hvis der er behov, indkaldes til et nyt *inspektionsreview* for at få dokumenterne godkendt.

5.3. Instruktionsdækning

Når en RPA-proces kører, eksekverer det et antal kodelinjer (instruktioner), som tilsammen udfører den arbejdsgang, som RPA-processen er pålagt.

Ved *instruktionsdækning* er man interesseret i at vide, hvor stor en del af kodelinjerne, som er dækket af testcases. Dette kaldes for testdækning, og det udregnes på følgende måde:

$$\text{Instruktionsdækning} = \text{Antal testede kodelinjer} * 100 / \text{Antal kodelinjer}$$

Det anbefales, at man som minimum har 100% instruktionsdækning. *Instruktionsdækning* er den svageste testdækning. Ved særlige kritiske steder i koden kan man i stedet benytte beslutningsdækning, betingelsesdækning eller multibetingelsesdækning i stedet.

Instruktionstest er en whitebox testdesignteknik, hvor testcases designs til at eksekvere kodelinjer.

Eksempel på kode:

1. Læs alder
2. Hvis alder ≥ 18 og alder < 67 så
 - a. Beregn voksenpris
3. Ellers hvis alder > 67 så
 - a. Beregn pensionistpris
4. Ellers
 - a. Beregn børnepris
5. Slut

(Der er bevidst en fejl i koden. Pensionist skulle være alder ≥ 67 . Nu vil en alder på 67 blive beregnet som børnepris).

Lad os lave en testcases til test af kodelinjerne:

Alder = 22. Denne testcase rammer kodelinjerne: 1, 2, 2a og 5.

Alder = 69. Denne testcase rammer kodelinjerne: 1, 2, 3, 3a og 5.

Alder = 8. Denne testcase rammer kodelinjerne: 1, 2, 3, 4, 4a og 5.

Instruktionsdækning = $8 * 100 / 8 = 100\%$.

Vi fik 100% instruktionsdækning, men vi fandt ikke fejlen. For at finde fejlen skal man anvende mere andre testdækningsmetodikker og/eller andre whitebox testdesigntechnikker. Grænseværdianalyse ville have givet nogle andre testcases, som fx:

Alder = 18. Denne testcase rammer kodelinjerne: 1, 2, 2a og 5.

Alder = 67. Denne testcase rammer kodelinjerne: 1, 2, 3, 4, 4a og 5.

Alder = 17. Denne testcase rammer kodelinjerne: 1, 2, 3, 4, 4a og 5.

Instruktionsdækning = $7 * 100 / 8 = 87,5\%$.

5.4. Kodestandard

Det er en fordel, at alle RPA-udviklere anvender den samme kodestandard. Virksomheden kan med fordel tage udgangspunkt i eksisterende kodestandarder.

De enkelte RPA-værktøjer stiller ofte en kodestandard til rådighed, som virksomheden kan benytte som udgangspunkt.

En kodestandard bør behandle følgende emner:

- Designprincipper
- Moduldesign og genbrug
- Navngivning af variable, funktioner og moduler
- Kommentarer
- Udviklingsmetodikker, fx citrix, web, desktop
- Versionsstyring
- Sikkerhed
- Bruger credentials
- Fejlhåndtering og logning
- Evt. brug af standard skabeloner
- Kvalitetssikring
- Tjeklister til review

5.5. Test-Driven Development

Test-Driven Development (TDD) skal ses som en udviklingsmetode. Koden bliver let at vedligeholde, og koden er godt organiseret. Udfordringen kan være, at man udvikler dobbelt, og det kan være svært at tænke TDD i starten af en proces.

Det anbefales dog, at RPA-udviklerne prøver TDD af.

I TDD starter RPA-udvikleren med at kode en RPA-testcase, som fejler, da der endnu ikke er kode, som testcasen kalder.

Eksempel

RPA-udvikleren skal udvikle en RPA-proces som skal tjekke om en fil er kommet i en mappe.

I RPA-værktøjet startes en ny kodefil som hedder FilMappe_test. Første kode i FilMappe_test er:

Kald FilMappe modul

Når RPA-værktøjet starter FilMappe_test, så vil testcasen fejle, da FilMappe modulet ikke findes. Man siger at testcasen er rød, da den er fejlet.

RPA-udvikleren må kun skrive funktionskode, når testcase koden er rød. RPA-udvikleren starter derfor en ny kodefil, som hedder FilMappe. FilMappe er tom. RPA-udvikleren må kun lige netop skrive, så meget kode, at testcasen virker.

Nu køres FilMappe_test igen, og nu fejler den ikke. Udviklingen er i grøn tilstand.

RPA-udvikleren skal derfor skrive en ny testcase, som sandsynligvis vil fejle fordi koden ikke er tilstede i FilMappe.

FilMappe_test udvides med:

Kald FilMappe med filnavn.pdf c:/downloades

Kørsel af FilMappe_test vil fejle fordi FilMappe ikke har kode til at modtage filnavn og mappenavn, så er testcasen rød.

FilMappe koden udvides med argumenter til at modtage filnavn og mappenavn.

Kørsel af FilMappe_test fejler ikke, og testcasen er derfor grøn.

FilMappe_test udvides med koden:

Kald FilMappe med filnavn.pdf c:/downloades og returkode

Hvis returkode er fundet, så skriv "Filnavn findes i mappen"

Ellers returkode er ikke fundet, så skriv "Filnavn findes ikke i mappen".

Ellers skriv "Et eller andet gik galt"

Kørsel af FilMappe_test vil fejle igen, og derfor kan RPA-udvikleren tilføje kode, så testcasen virker.

På den måde bliver RPA-udvikleren ved med at bygge testcases og dernæst kode, indtil hele RPA-processen er udviklet.

Det er vigtigt at forsøge at gøre testcases lette at køre igen senere.

Eksempel:

En RPA-proces har kørt fint i 3 måneder. RPA-udvikleren, som har udviklet RPA-processen, er på ferie og nu er RPA-processen stoppet med en fejl i koden.

Den nye RPA-udvikler finder fejlen, som kræver rettelser flere steder i kode.

Udvikleren er meget nervøs for at lave disse rettelser og har betænkeligheder ved,

hvilke andre fejl disse rettelser kan medføre. Ting, som virkede før, virker pludselig ikke. Dette er skrækscenariet for den nye RPA-udvikler. I denne situation kommer den tidligere RPA-udviklers testcases virkelig til sin ret.

Efter den nye RPA-udvikler har lavet en testcase (for at teste, at den nye fejl er rettet) og dernæst har udviklet koden, som fikser det, så kan RPA-udvikleren afvikle alle de tidligere testcases og se, at de alle sammen er grønne. Det betyder, at rettelsen ikke har skabt problemer andre steder.

RPA-udvikleren er nu rimeligt rolig ved at sende den rettede RPA-proces i produktion igen.

6. Møder

Projektmodellen indeholder nogle få påkrævede *møder*. Disse møder beskrives nærmere i dette afsnit.

6.1. Retrospektiv

Retrospektivmødet er en mulighed for RPA-teamet til at inspicere sig selv og udarbejde en plan for forbedringer, der kan effektueres hurtigst muligt.

Dette er et tidsbegrænset møde af 3 timers varighed og afholdes én gang om måneden.

Formålet med det retrospektive møde er at:

- Inspicere hvordan projektet er gået i forhold til personer, relationer, processer og værktøjer
- Identificere og ordne de større ting, der er gået godt samt mulige forbedringer
- Udarbejde en plan for implementering af forbedringer til, hvordan RPA-teamet arbejder.

Projektlederen opfordrer teamet til at forbedre deres udviklingsproces og praktikker for at gøre dem mere effektive og glade. Under hvert *retrospektivmøde* planlægger teamet måder at øge produktkvaliteten ved at tilpasse definitionen af *done* efter behov.

6.2. Stand-up

Stand-up er et møde, der er tidsbegrænset til 15 minutter, hvor RPA-teamet står op og synkroniserer Kanban tavle, aktiviteter og hjælper hinanden med at fjerne forhindringer.

Stand-up mødet bør afholdes samme sted og tid på fastlagte dage for at reducere kompleksiteten. Ofte kan RPA-teamet mødes hver dag, hvis der er aktiviteter nok til dette, men RPA-teamet skal som minimum mødes én gang per uge til *stand-up*.

Under mødet forklarer hver team deltager:

- Hvad har jeg opnået siden sidste møde?
- Hvad bliver færdigt før det næste møde?
- Hvilke hindringer er i vejen?
-

RPA-teamet bruger mødet til at vurdere fremskridt, hvordan fremdriftens tendens er mod færdiggørelse af igangværende arbejdsopgaver.

RPA-deltagerne kan mødes umiddelbart efter mødet for at hjælpe hinanden med arbejdsopgaverne. Man må ikke komme i løsningsstilstand under selve mødet.

RPA-teamet bør være i stand til at forklare projektlederen, hvordan de agter at arbejde sammen som et selvorganiserede team for at nå målet og skabe den forventede leverance.

Stand-up mødet er *ikke* et statusmøde, og mødet er kun for deltagere i RPA-teamet.

Stand-up mødet forbedrer kommunikation, eliminerer behovet for andre møder, fremhæver og fremmer en hurtig beslutningsproces, og forbedrer RPA-teamets vidensniveau. Dette møde er et centralt inspektions- og tilpasningsmøde.

6.3. Interview og gennemgang af arbejdsgang

Interview og gennemgang af arbejdsgang er et møde, hvor brugerne fortæller og viser, hvordan den manuelle arbejdsgang er. Det er vigtigt, at præsentationen af arbejdsgangen er så detaljeret som muligt - helst ved at brugerne har nogle konkrete opgaver, som løses på mødet, så RPA-deltageren kan se præcis, hvordan de enkelte opgaver løses.

Forberedelse

På mødet skal brugere med daglig/praktisk viden om udførsel af arbejdsgangen deltage. Desuden skal én fra RPA-teamet deltage. Det kan være en RPA-deltager, som har Business Analyst eller udvikler rollen.

Varigheden på mødet kan varigere afhængig af arbejdsgangens størrelse, men et interviewmøde må ikke have en varighed på mere end 3 timer. Ved behov for længere varighed anbefales det at tage mødet over flere dage.

Det er en fordel, hvis brugerne har nogle gode og repræsentative opgaver, som de samler sammen inden mødet. Disse opgaver kan brugerne, så benytte til at vise, hvordan de arbejder med systemerne.

Brugerne skal også indsamle alle relevante vejledninger og instruktioner, som kan give RPA-deltageren et indblik i hvordan arbejdsgangen er. Disse vejledninger og instruktioner kan med fordel sendes til RPA-deltageren inden mødet, så denne kan mulighed for at stille spørgsmål til dem under mødet.

Afholdelse

Mødet styres af RPA-deltageren, som beder brugerne fortælle lidt om arbejdsgangen. Det anbefales at RPA-deltageren optager hele mødet på video, da dette giver et langt bedre og hurtigere flow, end at RPA-deltager skal tage notater.

Arbejdsgangen vises ved at løse nogle konkrete opgaver. RPA-deltageren stiller afklarende spørgsmål. Det er vigtigt at RPA-deltageren kigger efter regler og undtagelser under præsentationen, så hele arbejdsgangen bliver klarlagt. RPA-deltageren kan desuden stille opklarende spørgsmål til eventuelt tilsendte vejledninger og instruktioner.

Hovedformålet er, at RPA-deltageren forstår og selv vil være i stand til at kunne udføre den manuelle arbejdsgang ved hjælp af dokumentationen skabt under mødet.