Sundsvalls kommun

Regler och riktlinjer vid RPA-utveckling

En handbok för UiPath-utvecklare

Dokumentinformation

Titel	Regler och riktlinjer för RPA-utveckling
Skapat av	Claes Nyborg
Datum	2020-03-22
Godkänt av	Mikael Nordlander
Version	1.0

Förändringshistorik

Version	Datum	Status och eventuell förändringsorsak	Utfärdare
0.1	2020-04-14	Första förslaget.	Claes Nyborg
0.2	2020-04-16	Justeringar i avsnitten, 2.1.4, 2.1.5, 2.1.6 och 2.2 samt formateringsändringar i hela dokumentet.	Claes Nyborg
0.3	2020-04-20	Döpt om dokumentet med tydligare begrepp. Tillägg av avsnitt 1.1 regler och riktlinjer. Justeringar i hela dokumentet för vad som är regler kontra riktlinjer samt ändringar av formuleringar i hela dokumentet. Större ändringar i avsnitt 2.6 källkodshantering.	Claes Nyborg
0.9	2020-04-24	Döpt om dokumentet för att spegla systerdokumentet i referens [1]. Justeringar i avsnitten 1.2, 2.1.6.3, 2.5, 3. Justering av rubriknivåer i avsnitt 4. Adderat avsnitt 4.3 kö-ärenden.	Claes Nyborg
1.0	2021-03-22	Justerat namngivningsstandard i avsnitt 2.1.1. Justerat namngivningsstandard så orchestrator-entiteter inte behöver ha avdelning i namnen i avsnitt 2.1.6. Tagit bort avsnitt om miljöer under avsnitt 2.1.6. Justerat rubrik och innehåll i avsnitt 2.1.6.1 för att handla om maskiner istället för robotar. Ändringar i hantering av personuppgifter i både avsnitt 4.4, 4.3 och 3.	Claes Nyborg

Relaterade dokument

Ref	Version	Datum	Benämning
1	0.5	2020-04-22	Modell för RPA-utveckling

Innehållsförteckning

1 Inledning

Detta dokument är tänkt att vara ett stöd för framförallt utvecklare av automatiseringar i UiPath men också till viss del för administratörer av plattformen, mer om det nedan. När alla utvecklare använder samma praxis blir underhållet och förvaltningen av automatiseringarna lättare och det blir även lättare att samarbeta. Dokumentet sätter också en standard som ökar säkerheten rörande automatiseringar, gör automatiseringarna stabilare och återanvändning lättare.

Dokumentet är också, som nämnt ovan, i vissa fall tänkt som stöd för administratörer av UiPath–plattformen främst genom att sätta en tydlig och beskrivande namnstandard för entiteter i orchestratorn, se avsnitt 2.1.6 Orchestrator. Detta för att underlätta och förtydliga arbetet som utförs i Orchestratorn.

Detta dokument bygger på grundriktlinjerna som finns uppsatta i UiPaths dokumentation för både Studio och Orcherstrator, se avsnitten 1.2.1 och 1.2.2. Till dessa grundriktlinjer har mer allmänt kända utvecklarriktlinjer adderats samt även lokalt förekommande praxis.

1.1 Regler och riktlinjer

Detta dokument innehåller både *regler*, där mer specifika och obligatoriska ramar sätts för handling, samt även *riktlinjer*, som inte är exakta handlingsramar utan mer generella förhållningssätt. Främst innehåller dokumentet regler men inte alla är bindande utan av mer rekommenderande karaktär utifrån de fastställda riktlinjerna. De regler som är obligatoriska har angivits som ”ska” medan rekommenderade handlingssätt men som är inte obligatoriska har angivits som ”bör”. Regler som inte har angivits som varken ”ska” eller ”bör” ska ses som rekommenderade men inte tvingande regler, dessa kan t.ex. ha angivits som ”kan”, ”gärna” eller annat.

1.2 Källor

Lägg till vad på varje länk som hänvisas till.

1.2.1 Källor från dokumentationen för UiPath Studio

- **Workflow Design** (https://docs.uipath.com/studio/docs/workflow-design): Naming Conventions, Comments and Annotation. Datum: 2020-04-23.
- **Projekt Organization** (https://docs.uipath.com/studio/docs/project-organization): Alla avsnitt. Datum: 2020-04-23.
- **Automation Lifecycle** (https://docs.uipath.com/studio/docs/automation-lifecycle): Test, Release, Monitoring. Datum: 2020-04-23.

1.2.2 Källor från dokumentationen för UiPath Orchestrator

- **Automation Best Practices** (https://docs.uipath.com/orchestrator/docs/automation-best-practices): Robots, Environments, Queues. Datum: 2020-04-23.

- **Deployment and Configuration Considerations** (https://docs.uipath.com/orchestrator/docs/deployment-and-configuration-considerations): User and Robot Permissions. Datum: 2020-04-23.

2 Kodningsstandarder

2.1 Namnkonvention

Beskrivande och meningsfulla namn ska alltid användas för workflow-filer, aktiviteter, argument och variabler för att förklara deras syfte och användning i projektet.

Projekten i sig ska också ha meningsfulla beskrivningar, eftersom dessa visas i Orchestratorns användargränssnitt. Att ha en tydlig namngivningskonvention i Orchestratorn för t.ex. miljöer (environments), tillgångar (assets) och köer (queues) underlättar användandet av Orchestratorn.

Endast argumentnamn är skiftlägeskänsliga, men för att förbättra läsbarheten, ska variabler och andra entiteter också anpassa sig till samma namnkonvention.

Alla arbetsflöden bortsett från Main-flödet (som finns i ramverken) ska innehålla ett verb som beskriver vad arbetsflödet gör,
t.ex. *GetTransactionData*, *ProcessTransation*, *TakeScreenshot*.

2.1.1 Variabler

- En variabel bör endast användas för ett syfte.
- Minimera omfattningen för varje variabel.
- Håll aktiviteter som använder samma variabel/er så nära varandra som möjligt i flödet.
- Variabler ska **alltid** ha meningsfulla namn. Variabelns namn ska beskriva exakt den entiteten variabeln representerar. Ange med ord vad variabeln representerar.
- Vid namngivning ska **kamelnotation med stor första bokstav** (eng. upper Camelcase) användas för variabler. För kamelnotation med stor första bokstav används sammansatta ord, inga andra tecken mellan orden, där varje ord börjar med en stor bokstav. T.ex *TransactionNumber*, *FilePath*, *ReportName* osv.
- Längden på variabelns namn bör vara mellan 6 och 20 tecken lång. Om du känner att 20 tecken inte räcker, överväg att förkorta längre ord. Kortare (t.ex. *i*, *index*, *file*, *row*) variabelnamn kan användas när de finns i ett lokalt sammanhang (scope).
- Variabler bör inte namnsättas utifrån ungersk notering, dvs. variabelnamn ska **inte** ha prefix som anger variabeltyp (t.ex. str, int, bool, osv). Detta gör namnen svårlästa och samt att variabeltypen är lätt att se när variabeln dyker upp i UiPath.
- Bool-variabler: Dessa variabler bör ha namn som kan vara sant eller falskt. Du kan t.ex. använda prefixet **Is** eller **Är** följt av namnet, t.ex. *IsRed*, ÄrHittad etc. eller *Exists* eller motsvarande efter namnet, t.ex. *ApplicationExists*. Använd alltid positiva namn, negativa namn, t.ex. *NotFound*, ska undvikas.

2.1.2 Argument

För argument gäller samma regler och riktlinjer som för variabler men med nedanstående skillnader:

- Varje argument ska ha ett prefix beroende på riktning: **in**, **out** eller **io** följt av understreck ("_"). Exempel: *in_Config*, *out_int_InvoiceNumber*, *io_RetryNumber*, *in_Employees*.
- Man kan använda standardvärden för argument antingen för att testa enskilda flödesfiler eller som standardvärde i återanvändbara flödesfiler/komponenter, när inget värde ges till argumenten. Standardvärdet ska specificeras i beskrivningen i den återanvändbara flödesfilen.

2.1.3 Aktiviteter

Aktivitetsnamn ska tydligt återspegla de åtgärder som vidtagits, t.ex. *Click "Spara" Button*. Behåll delen på titeln som beskriver åtgärden (Click, Type Into, Element Exists osv). I de fall en aktivitet kastar ett fel (exception) kommer källan till felet innehålla aktivitetsnamnet. Detta gör att ett beskrivande namn för varje aktivitet underlättar felsökning och förståelse av felet. Även aktiviteter som har ett standardnamn, som Assign, If, For Each, Sequence osv. ska namnet bytas på och för dessa behöver man vara mer noggrann eftersom namnen inte ändras automatiskt – dessa måste man alltid byta namn på manuellt.

2.1.4 Flödesfiler (Workflow)

- Namnen ska skrivas med kamelnotation med stor första bokstav.
- Namnet på flödesfiler ska börja med ett prefix som innehåller applikationsnamnet. Exempel för fil som jobbar i applikationen Raindance: ***Raindance**_Login* eller ***Raindance**_ExtractClientReport*. Flödesfiler som jobbar i samma applikation eller system bör grupperas i en mapp med applikationsnamnet under projektets huvudmapp. Om det finns många filer för en applikation, bör vidare kategorisering med hjälp av undermappar användas.
- När ett mallramverk (eng. Framework) används – är filerna för mallramverket redan skapade och har standardiserade (inklusive Main.xaml) namn – de ska inte ändras.
- När ett mallramverk används ska testflödes-filer (Test_Framework-filerna som kör tester) namnges med prefixet **Test_**. Dessa ska placeras i mappen Test_Framework i projektets huvudmapp.
- För att underlätta läsförståelsen, använd gärna nummerprefix för att betona ordningen för körandet (invoke) av de olika flödena i projektet, där utgångspunkten alltid är Process (i REFrameWork).

2.1.5 Projekt och Delprojekt (Projects i UiPath)

- Namnen ska skrivas med kamelnotation med stor första bokstav.
- Om det är relevant, gruppera avdelningsvis hjälp av ett prefix: T.ex. ***HR_***, ***Ekonomi_***.

- Om processen utgörs av delprocesser (flera projekt/paket för samma affärsprocess, t.ex. vid användande av dispatcher- och perfomerflöden), ska affärsprocessens namn anges först och därefter delprocessens namn som prefix: T.ex. *Timanställda_Registrera* och *Timanställda_Avsluta*, eller *RegistreraTimanställda_Perfomer*.

2.1.6 Orchestrator

En konsekvent namngivningsstrategi med beskrivande namn ska användas även i Orchestratorn. Avdelning anges på många enheter eftersom dessa alltid är synliga på hela delorchestratorn (tenant).

2.1.6.1 Maskiner

Maskiner bör ha en beskrivning som anger vart maskinen används, t.ex. avdelning eller processer som använder den.

2.1.6.2 Tillgångar (Assets)

Tillgångar utöver behörigheter ska namnges enligt följande:

- Tillgångar kan namnges med **[Avdelning]_** som prefix
- Tillgångar som gäller en specifik affärsprocess, [Avdelning]_[Affärsprocessens namn]_[Tillgångens (asset) namn], t.ex. *HR_RegistreraTimanställda_URLÖversättningsTabell*.
- Tillgångar som används eller kan komma att användas av flera processer, [Avdelning]_[Tillgångens (asset) namn], t.ex. *HR_URLÖversättningsTabell*.

Behörigheter (credentials) ska namnges enligt följande:

- Behörigheter ska alltid ha ett namn som innehåller **[Avdelning]** (valfritt), **[Affärsprocessens namn]**, **[Behörighets namn]**.
- De ska ha **Cred_** som prefix före de övriga delnamnen.
- Behörigheternas utgångperiod ska anges i dagar, om ingen utgångsperiod finns ska **NA** ersätta utgångsperioden.
- Behörigheter som gäller en specifik affärsprocess, Cred_[Utgångsperiod i dagar]_[Avdelning]_[Affärsprocessens namn]_[Behörighetens namn], t.ex. *Cred_180_HR_RegistreraTimanställda_AnvändareRaindance*.
- Behörigheter som används eller kan komma att användas av flera processer, Cred_[Utgångsperiod i dagar]_[Avdelning]_[Behörighetens namn], t.ex. *Cred_NA_HR_AnvändareRaindance*.

2.1.6.3 Köer (Queues)

Köer ska namnes enligt följande:

- Vad kön ska heta ska vara beskrivande för vilken typ av ärenden den innehåller (vad utgör ett ärende).
- Köer kan namnges med **[Avdelning]_** som prefix.
- [Avdelning]_[Affärsprocessens namn]_[Könamn], t.ex. *RegistreraTimanställda_NyaMedarbetareQueue*.

2.2 Inställningar för värden

Vid automatisering behövs ofta inställningar för variabler och andra värden. Inställningarna kan kategoriseras i följande grupper:

- Inställningar för värden som **aldrig** ändras. Exempel på dessa är statiska selektorer (selectors), eller etiketter (labels) i applikationer/system. Dessa bör hårdkodas i arbetsflödena - och där det är relevant - i variabler. Det ska inte finnas någon framtida nytta av att ha dessa lättjusterade.
- Inställningar för värden som är **osannolika** att de ändras men används på flera ställen, eller inställningar som är viktiga men inte avsedda att bytas av någon annan än en utvecklare. För enkelt kunna ändra och även öka läsbarheten av flödet, ska dessa inställningar lagras i en inställningsfil. Exempel på dessa är loggmeddelanden, loggfält, fil- eller mappsökströngar och väntetider. Konfigurationsfilen bör ha beskrivande namn för nycklarna som motsvarar värden (t.ex. av "ReportID" istället för det verkliga värdet "12.361.223").
- Inställningar som **sannolikt** kommer ändras från en miljö till en annan, ändras i framtiden, eller ska vara tillgängliga för verksamheten. I denna kategori finns t.ex. applikationssökströngar, webbadresser, könamn, behörigheter, epost-adresser osv. Dessa inställningar ska lagras som tillgångar (assets) i Orchestratorn. Den största fördelen med dessa är att värdena kan ändras utan att modifiera koden, även av verksamheten, så att det som utvecklas i utvecklingsmiljö är lätt att migrera utan ändringar till test- och sedan produktionsmiljö.
- Inställningar som verksamheten ska ha möjlighet att ändra på men som av någon anledning inte passar som tillgångar i Orchestratorn. Dessa kan lagras i extern inställningsfil i en skyddad mapp, t.ex. en nätverksplats som enbart vissa personer har tillgång till. Denna typ av inställningar ska begränsas eftersom filen eller värden kan modifieras eller tas bort och är starkt påverkad av en mänsklig osäkerhetsfaktor. Värdena bör ha en formatgranskning inbyggd i filen. Det ska alltid finnas en backup på en ”ren” och inte ifylld version av filen.
- Inställningar som har olika värden för olika robotar. Dessa ska lagras som tillgångar i Orchestratorn med olika värden per robot.

Generellt sett ska den slutliga lösningen vara justerbar vid behov för att möjliggöra variationer och förändringar i indata utan ingripande från utvecklare.

2.3 Behörigheter

Bestämmelser rörande behörigheter finns utförligare beskrivet i referens [1]. Som summerad upprepning ska behörigheter och rättigheter för en robot alltid vara så snäva och precisa som möjligt. Ett risktänk ska alltid finnas för vad som eventuellt kan hända om någon obehörig får tillgång till robotens behörigheter.

2.3.1 Robotbehörigheter

Användarbehörigheter för att Orchestratorn ska kunna logga in på Windows på en robot ska enbart lagras som användarbehörigheter på roboten i Orchestratorn. Lösenordet är då krypterat med 256 bitars AES-krypteringsalgoritm och kan inte visas efter det har satts första gången. Det är **kraftigt** rekommenderat att utvecklare inte bör känna till lösenordet i produktion efter att roboten har blivit stabiliserad i produktionsmiljön, om det inte är absolut ofrånkomligt. Lösenordet kan sättas av annan än utvecklaren eller ändras i efterhand via t.ex. en robotprocess för lösenordsbyten.

2.3.2 Applikations-/systembehörigheter

Applikations-/systembehörigheter ska **aldrig** lagras i arbetsflödesfiler eller inställningsfiler, varken som klartext eller som en ’Get Password’-aktivitet. Det finns två sätt dessa behörigheter får lagras:

- Behörighetstillgångar (Credential assets) i Orchestratorn: Dessa lagras säkert på SQL Servern, med 256 bitars AES. När lösenordet har angetts kan det inte visas igen. De hämtas i flödet med hjälp ’Get Credential’-aktiviteten och Orchestratorn returnerar användarnamn som sträng och ett lösenord som säker sträng (secure string). I produktionsmiljön ska behörigheterna alltid lagras på detta sätt. I test är det önskvärt att förvara behörigheterna på detta sätt och i andra hand enligt nedan.
- Om användning av behörighetstillgångar i Orchestratorn inte är möjligt, ska dessa lagras i Windows Autentiseringshanteraren (Credential Store). Lagringen sker då på lokalt på maskinen som roboten kör på vilket innebär att man behöver lägga till behörigheten på varje maskin som ska köra processen.

Omfattningen (scope) för behörighetsrelaterade variabler ska begränsas så mycket som möjligt och bara vara tillgängliga där de verkligen behövs i arbetsflödet.

2.3.3 Säkra strängar (Secure string)

Lösenordet från ’Get Credentials’-aktiviteten returneras som en säker sträng (Secure string). Detta är en särskild klass i .NET Framework som representerar text som behandlas konfidentiellt. Lösenordet är inte i klartext i minnet, utan i förvrängd form (dock inte krypterat). En säker sträng tas även bort från minnet så fort omfattningen är variabeln finns är passerad, till skillnad från vanliga strängar. När en säker sträng används för att t.ex. logga in i ett system ska en ’Type Secure Text’-aktivitet användas för normala applikationer eller ’Send Keys Secure’-aktivitet i terminalapplikationer.

2.4 Felhantering

Två typer av fel eller undantag kan hända när du kör en automatiserad process, ett förutsägbart fel eller helt oväntat fel. Utifrån detta kan felet/undantaget hanteras antingen genom att explicita åtgärder utförs för att hantera felet inom det specifika arbetsflödet där det uppkommer, eller genom att eskalera felet till en högre nivå i processen.

- **Förutsägbara fel** hanteras t.ex. genom att placera känsliga aktivitetssteg inuti i en 'Try Catch'-aktivitet eller i ett Retry-block där felet kan hanteras/motverkas.
- På den högsta nivån, ska huvudprocessdiagrammet ha breda korrigerande åtgärder för att ta itu med alla **generiska och oförutsägbara fel** och för ökad stabilitet. Båda ramverksmallarna som nämns i avsnitt 3 har redan denna felhanteringsmekanism inbyggd.

När ett fel inte kan hanteras ska det vara så lätt som möjligt att förstå vart och varför ett fel har uppstått och då är den förklarande namngivningen av aktiviteterna viktig.

2.5 Rent och snyggt

Innan det slutförda UiPath-projektet laddas upp till Orchestratorn bör projektet och alla inkluderade flöden gås igenom och rensas upp. Ta t.ex. bort oanvända variabler och avaktiverade (disabled) aktiviteter. Kolla även så namnen på aktiviteter är beskrivande och unika och att varje arbetsflödesfil (workflow) har en beskrivande notering.

Personuppgifter ska aldrig finnas i ett UiPath-projekt, säkerställ detta med en genomgång och rensning av data i variabler, kommentarer och noteringar, aktivitetsnamn osv.

Kolla också att namnet på UiPath-projektet är beskrivande och följer namnstandarden eftersom detta syns i Orchestratorn. Projektets namn är satt till det initiala projektnamnet, men det kan ändras i projektets inställningar i UiPath.

Beskrivningen av projektet syns även det i Orchestratorn och bör därför också vara beskrivande och tydligt.

2.6 Källkodshantering

För säker lagring, ändringshistorik samt för att lätt kunna samarbeta med andra utvecklare ska källkodshantering användas i Studion.

- En förvaringsplats (repository) ska skapas för varje projekt i UiPath.
- Utvecklingen ska inte ske på huvudgrenen (master branch) utan i egen gren (eng. branch) för utveckling.
- Vid utveckling av stora projekt eller då flera utvecklare är inblandade av annan anledning kan stödgrenar (supporting branches) skapas från utvecklings-grenen för utveckling av ett specifikt syfte, t.ex. ett flöde, buggfix, eller något annat. Denna typ

av stödgrenar ska sammanföras (merge) med utvecklings-grenen då de är färdigutvecklade.
- När utvecklingen av hela projektet är färdigt och koden i utvecklings-grenen är testad ska den sammanföras med huvudgrenen och sedan sättas (publish) i produktion.
- Om något mindre fel finns i produktionskoden, den som finns i huvudgrenen, kan en snabbfix-gren (hotfix) skapas direkt från huvudgrenen och sedan sammanföras efter testning direkt till huvudgrenen och sedan sättas i produktion.
- Ett beskrivande ändringsmeddelande (commit message) ska anges vid varje ändring (eng. commit). Det är rekommenderat att hålla ändringarna så små som möjligt eftersom det då är det lättare att hålla reda på vilka ändringar som är nygjorda och därmed skriva ett bra ändringsmeddelande.
- Åtminstone en gång per dag ska ändringarna laddas upp (eng. push) till förvaringsplatsen (repository) från Studion men gärna oftare.

3 Mallramverk

En automatisering ska alltid utvecklas med grund i ett mallramverk eftersom detta underlättar och förbättrar utvecklingen på flera sätt. I mallramverken finns färdig högnivåhantering av fel, en miniminivå av loggning, justerbara processinställningar i extern Excel-fil samt ärendehantering för Orchestrator-köer.

Så ofta som möjligt ska köer i Orchestratorn användas för ärenden som processen ska hantera. Detta för att kunna se historik för varje enskilt ärende, se allmän statistik, automatisk omprövning av misslyckade ärenden samt att enkelt manuellt kunna starta om misslyckade ärenden. Kö-ärenden (queue items) som skapas i Orchestratorn ska **inte** i onödan innehålla direkta personuppgifter (data som på ett eller annat sätt kan härledas till en specifik person) för att minimera för spridning av personuppgifter utan anledning. I största möjliga mån ska inga personuppgifter användas, i andra hand kan indirekta personuppgifter användas (t.ex. ärendenummer till ärende där personuppgifter finns), i tredje hand kan maskerade eller pseudonymiserade personuppgifter användas och om inget av dessa fungerar kan direkta personuppgifter användas.

Excel-filer kan också användas för köhantering men ärendena bör då läsas in i en eller flera köer i Orchestratorn innan de ska handläggas av processen.

3.1 REFrameWork

REFrameWork är den rekommenderade mallen att använda. Här finns allt som är positivt med ett ramverk utan att det är för stort eller komplicerat.

3.2 Enhanced REFrameWork

Detta ramverk kan användas för stora och komplicerade processer som behöver extra stabilitet. Här är varje del av högnivåhanteringen ett eget ramverk i sig. För de flesta processer är detta ramverk onödigt komplicerat.

4 Design av lösning

4.1 Noteringar och kommentarer

Noteringar och kommentarer används för att beskriva vad som händer och varför där det behövs. Åtminstone varje arbetsflödesfil ska ha en inledande beskrivande notering med text om:

- Vad flödet gör.
- Vad flödet behöver för ingående tillstånd (precondition).
- Vilket avslutande tillstånd (post condition) som flödet har skapat.

Ha alltid i åtanke att en annan utvecklare bör kunna öppna projektet eller flödet och lätt kunna förstå vad flödet gör och varför.

4.2 Projektuppdelning

Det är rekommenderat att bryta upp projektet i mindre arbetsflöden, gärna så små som möjligt. Flödena kan då testas oberoende av varandra samt återanvändas på ett modulärt sätt. Projektet ska delas upp utifrån applikationsspecifika flöden, som bara hanterar applikationerna samt flöden som innehåller affärslogik.

4.2.1 Återanvändning

Eftersom ett projekt ofta har hela flöden eller delar av flöden som utför samma steg men på olika ställen och/eller med olika värden bör dessa flöden brytas ut för att kunna återanvändas.

Det finns inget självklart sätt att bryta upp flöden men en tumregel är det generella ordspråket för utveckling, ”don’t repeat yourself”. Detta kan vara svårt att följa i vissa fall men är en bra tumregel att sikta mot. Om man ofta kopierar och klistrar in samma steg från ett flöde till ett annat, är detta ett tydligt tecken att stegen bör brytas ut till ett eget flöde.

4.2.2 Applikationsflöden och affärslogikflöden

Som nämnt ovan ska affärslogik separeras från flöden för applikationshantering. Applikationshanteringen, dvs. automatiseringens hantering av gränssnitt ska vara generell och processoberoende för att kunna återanvändas. Ett flöde för applikationshantering ska även enbart utföra **en** specifik handling/funktion i ett gränssnitt (t.ex. ”öppna kund”, ”läs kundinformation”, ”ändra kundinformation”) och **inte** flera handlingar/funktioner (t.ex. ”läs och ändra kundinformation”). På detta sätt kan dessa flöden återanvändas av olika processer oberoende av vad som ska uträttas av processen i stort och vilken affärslogik som används i den specifika processen.

Affärslogiken läggs i egna flöden som är specifika för processen. Ett exempel hur affärslogiken kan se ut är följande steg: ”öppna kund”, ”läs kundinformation”, om kunden är myndig kör då flödet ”ändra kundinformation” annars avsluta ärendet.

4.2.3 Förvaring och användning

Flöden som har skapats för möjlighet till återanvändning i andra projekt ska skapas och sparas som ett bibliotek (library). Detta gäller främst för flöden som hanterar applikationer men även andra flöden som kan tänkas vara intressanta att återanvända i andra projekt (t.ex.”konvertera personnummer till ålder”). Biblioteket ska sedan laddas upp till Orchestratorn. Från Orchestratorn kan biblioteken sedan användas i olika projekt och alla projekten får automatiskt uppdateringar av biblioteket om det finns någon. Om applikationen har uppdaterats och automatiseringarna har slutat fungera behöver man därmed bara justera hanteringen av applikationen på ett ställe, i biblioteket, för att alla automatiseringarna ska få det nya biblioteket från Orchestratorn och fungera igen.

4.3 Kö-ärenden

Som tidigare nämnt ska kö-ärenden som skapas i Orchestratorn **inte** i onödan innehålla direkta personuppgifter (data som på ett eller annat sätt kan härledas till en specifik person). Hänvisa istället till t.ex. ett ärendenummer som finns i annat system (t.ex. e-tjänst) där personuppgifterna som roboten ska hantera finns. Eller också bör personuppgifterna maskeras eller pseudonymiseras.

Om ärendena som automatiseringen ska hantera inte innehåller personuppgifter kan och bör dessa sparas direkt på kö-ärendena i Orchestratorn eftersom utvecklare, verksamheten och eventuellt andra intressenter med behörighet då kan se och följa varje ärende i sin helhet direkt i Orchestratorns webbgränssnitt. Data på kö-ärenden kan då också justeras, t.ex. innan nytt försök på ärenden där data på ärendena var felaktig och gjorde att automatiseringens hantering av ärendet misslyckades.

4.4 Loggning

Loggning ska utföras med ’Logga Meddelande’-aktiviteten (Log Message) eftersom man med den aktiviteten kan välja vilken nivå som loggningen ska ske på (Trace, Info, Warning, Error).

Meddelandet bör vara så beskrivande och specifikt som möjligt för att underlätta felhantering och debugging. Mer loggning ger en tydligare bild av vad som händer under processen och ger därför en tydligare bild av vad som har gått fel och varför. Åtminstone i början av varje flöde ska loggning ske men gärna i slutet av flödet också, speciellt om flödet är omfattande. Fel som fångas ska också loggas samt om data hämtas från någon extern källa. Använd mest lämpad loggningsnivå utifrån situationen (Trace, Info, Warning, Error), där info är standardnivå och trace är detaljnivå för att logga specifika steg inom ett flöde. Warning används t.ex. vid avvikande situationer men som inte är rena fel eller kritiska för processens fortsatta hantering av ärendet.

Direkta personuppgifter (data som på ett eller annat sätt kan härledas till en specifik person) ska **inte** i onödan finnas i loggarna av samma anledning som det inte ska finnas direkta personuppgifter i kö-ärenden i onödan, nämligen för att minimera spridning av personuppgifter i organisationen. Som vid övrig användning av personuppgifter bör även eventuella personuppgifter som används i loggar maskeras eller pseudonymiseras.