UPPSALA
UNIVERSITET

Skatteverket ADA

Machine Learning: Skanna ID-handlingar

Slutrapport, *Programvaruprojekt*, 15 HP, VT-2021

Författare: Lukas Hjernquist, Olle Gardell

Handledare: Jonas Sjöström

UPPSALA
UNIVERSITET

Sammanfattning

Projektet är att skapa en applikation som skall integreras med en AI-modell för avläsningar av ID-handlingar inom Skatteverket. På varje ID-handling finns en "Machine Readable Zone" (MRZ-kod) som är skapad för att kunna läsa av ID:s elektroniskt. Syftet med applikationen är att kunna säkerställa att modellen läser av MRZ-koden korrekt, samt vid felaktigheter kunna identifiera vilka fel som uppkommit och återge informationen till modellen. Detta är ett viktigt steg i maskininlärning av AI-modeller.

Genom att använda en iterativ och adaptiv utvecklingsmetodik och litteratursökning av tidigare forskningsbidrag inom ämnen som substrängsmatchning, maskininlärning och syntetiska testdata har man kunnat utveckla en slutprodukt som kan bearbeta boxmodeller. Applikationen tar in en boxmodell som skapas av programmet Tesseract. Boxmodellen innehåller alla tecken på en avläst ID-handling tillsammans med koordinater vart på bilden tecknen finns. Applikationen konverterar tecknen till en sträng, som tillsammans med en korrekt kod, som hämtas från en textfil jämförs för att återge tre boxmodeller med en full-, korrekta- samt felaktiga tecken identifierade.

Nyckeln för att lösa substängmatchning med indexering härleds till en interaktion mellan Vladimir Levenshteins distansalgoritm och Peter Weiners substrängsalgoritm, vilket ger Edvard M Mocreights "Algoritm M". Vid implementation utgör algoritmen en iterativ trädalgoritm som jämför substrängar med varandra för att identifiera skillnader och likheter.

Nyckelord: Maskininlärning, AI-modell, MRZ, Tesseract, Skatteverket, Vladimir Levenshtein Distans Algoritm, Algoritm M

UPPSALA
UNIVERSITET

Abstract

The project is to create an application that will be integrated with an AI model for readings of ID documents within the Swedish Tax Agency. Each ID document has a “Machine Readable Zone” (MRZ code) that is created to be able to read IDs electronically. The purpose of the application is to be able to ensure that the model reads the MRZ-code correctly, and in the event of errors to be able to identify which errors have occurred and reproduce the information to the model. This is an important step in machine learning of AI models.

By using an iterative and adaptive development methodology and literature search of previous research contributions in subjects such as substrate matching, machine learning and synthetic test data, it has been possible to develop an end product that can process box models. The application includes a box model created by the program Tesseract. The box model contains all the characters on a read ID document together with coordinates where in the image the characters are. The application converts the characters into a string, which together with a correct code, which is retrieved from a text file, is compared to reproduce three box models with a full, correct and incorrect character identified.

The key to solving submatch matching with indexing is derived from an interaction between Vladimir Levenshtein's distance algorithm and Peter Weiner's substrings algorithm, which gives Edvard M Mocreight's "Algorithm M". During implementation, the algorithm is an iterative tree algorithm that compares sub-strings with each other to identify differences and similarities.

Keywords: Machine learning, AI model, MRZ, Tesseract, Swedish Tax Agency, Vladimir Levenshtein Distance Algorithm, Algorithm M

UPPSALA
UNIVERSITET

Förord

Denna rapport är en sammanställning av ett programvaruprojekt utfört av två studenter från Uppsala universitet våren 2021. Arbetet har genomförts av Lukas Hjernquist och Olle Gardell och utger 15 poäng av totalt 180 i programmet Systemvetenskap med inriktning mot Programvaruutveckling.

Ett speciellt tack riktas till vår handledare Jonas Sjöström, docent vid Institutionen för informatik och media Uppsala universitet för allt stöd och feedback under arbetets gång. Vi vill även rikta tack till följande personer, som har ägnat tid åt oss genom möten och förseelse av information:

- Staffan Lyttkens, Samordnare, Team Ada Skatteverket
- Alexander Bertvig, Utvecklare, Team Ada Skatteverket
- Göran Kimell, Sektionschef IT, Skatteverket

Visby den 17 maj 2021

Olle Gardell

Lukas Hjernquist

Innehållsförteckning

Begreppslista:

MRZ - Machine Readable Zone, är en del av ett formulär som är avsett för maskinavläsning.

OCR - (Optical Character Recognition) fungerar som ett referensnummer.

Tesseract - Motor för optisk teckenigenkänning för olika operativsystem.

Tess4J – Bibliotek för att implementera tesseract i Java applikationer

1. Introduktion

Som inledning kommer vi att presentera vilken organisation vi kommer att göra vårt projekt för och vad projektet rent praktiskt kommer att gå ut på. Vi kommer även att redovisa vilka avgränsningar som vi fått från uppdragsgivare och dem vi själva sätter för att tydligare definiera projektet. Som en avslutning på introduktionen delar vi även med oss av våra förväntningar på oss själva, organisationen och projektet, samt vilka förväntade bidrag varje part kommer bistå med.

1.1. Uppdragsgivare

Den statliga förvaltningsmyndigheten Skatteverket hanterar många frågor inom hela Skatteförvaltningens område. Det inkluderar bland annat taxering av fastigheter, skatteregistrering för enskilda firmor och bolag, folkbokföring och ansökningar för namnbyte och id-kortsservice. Det är den sist nämnda som vårt arbete kommer vara inriktat på.

1.2. Praktiskt uppdrag

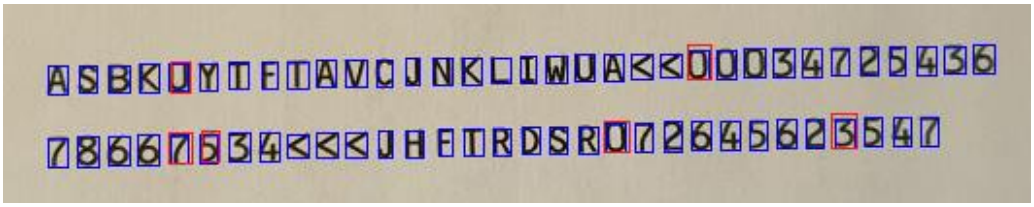
Skatteverket kommer inom snar framtid att utöka sin digitala funktionalitet genom att möjliggöra scanning av id-handlingar. Det som är intressant inom detta projekt är området på id-handlingen som en maskin ska läsa av, vilket beskriver personen vars id som har skannats. Detta område kallas för *Machine Readable Zone* som ofta förkortas till **MRZ**, vilket vi kommer nämna ofta framöver i rapporten när vi pratar om kod relaterad till denna. För att den nya funktionaliteten skall kunna tillämpas kommer organisationen ta hjälp av ett digitalt verktyg som finns inom området för artificiell intelligens, maskininlärning. De problem som finns kopplade till maskininlärning är erfarenhetsgraden för uppgiften. Vilket betyder att desto högre erfarenhet maskinen har, desto färre fel gör den i sina antagningar. Här kommer vårt mål vara att hjälpa till att skapa träningsdata som sedan skall köras i flera träningsiterationer, som i sin tur ska öka maskinens erfarenhetgrad.

Vi skall på uppdrag av Skatteverket skapa en applikation som integrerar med deras system för avläsning av ID-handlingar. Applikationen skall kunna ta in en avläst MRZ-boxfil från en ID-handling genom *optisk teckenavläsning* (ofta förkortat **OCR** efter engelskans *Optical Character Recognition*). Sedan skall den hämta en korrekt sträng av samma **MRZ** från ett textdokument som utgör ett facit för att jämföra dem båda strängarna och identifiera om brister uppkommit på den avlästa boxfilen. Avläsningen sker i programmet Tesseract. Tesseract är en motor för optisk teckenigenkänning för olika operativsystem.

Korrekt sträng kan se ut som följande:

ASBKUYTFTAVCJNKLIWUA<<00034725436

78667534<<<JHFTRDSR072645623547

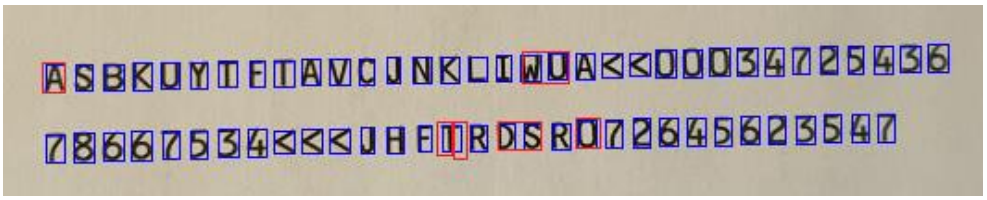


(Fig 1. Exempel på MRZ-kod från en ID-handling som är avläst i Tesseract)

Krav på brister som skall kontrolleras är:

- Antal tecken stämmer med facit men felaktigt OCR
- Antal tecken stämmer inte med facit, dubbla tecken tolkas som 1 tecken. Det kan även förekomma att fler än 2 tecken tolkas som ett tecken.

- 1 tecken tolkas som flera tecken se ”T” på nedersta raden



(Fig 2. Exempel på MRZ-kod från en ID-handling som är avläst i Tesseract med identifierade felavläsningar)

Man behöver kunna styra applikationen så att den kan lämna ifrån sig information med bara de felaktigheter som man upptäckt i givna substrängar. Applikationen har som input 1 boxfil + 1 textfacit som resultat ska den producera ny boxfil med filtrerat/korrigerat innehåll.

1.3. Avgränsningar

För att tydliggöra uppdraget kommer vi att definiera de avgränsningar som har gjorts.

Vi har för vårt uppdrag blivit avgränsade till egna verktyg och resurser. Vi kommer inte ta del av Skatteverkets system eller testdata. Detta därför att minska tidsåtgång till inlärning för deras system, som är en högre tröskel att ta sig över. Det finns dessutom en säkerhetsrisk med att släppa in oss i deras system utan att ha genomfört en gedigen bakgrundskoll på oss. Vilket i sig också skulle ta resurser ifrån själva uppdraget som lever under en kort och begränsad tid.

Uppdraget som vi har tilldelats rör sig inom området för AI (artificiell intelligens). Däremot är området för AI stort och täcker in fler delar än vad som är aktuellt för oss. Eftersom vi kommer arbete mot ett system som använder maskininlärning (eng. *machine learning*), är resterande AI-element inte relevanta för oss. För maskininlärningens del arbetar vi med bildigenkänning och där är fokus på avläsning av MRZ-kod. Utifrån detta kommer vi rikta oss mot att kvalitetssäkra de avläsningar som har gjorts för att kontrollera deras riktighet och noggrannhet.

Förväntningar

- Vi förväntar oss att vi ska skapa och leverera något som kan vara gynnsamt för Skatteverket att använda.
- Våra förväntningar på arbetsgivaren är att kunna få det stöd och öppna dialoger genom avstämningsmöten för att kunna skapa önskad funktionalitet i slutgiltig produkt. Vi förväntar oss även att kompetens som krävs för genomförande i detta projekt finns inom uppdragsgivarens organisation.
- Till följd av att projektarbetet har ett begränsat tidsfönster och tid för utveckling av produkt är ännu mer begränsad, så har Skatteverkets kontaktpersoner inga krav på att vi ska hinna bli färdiga med vårt tilldelade uppdrag. Däremot har de en förhoppning om att vi ska skapa något som de kan ta nytta av.
- På vår examinator/universitetet förväntar vi oss att få konstruktiv kritik och feedback löpande under projektets gång inom rimliga tidsramar.

1.4. Författarens eller delförfattarnas bidrag

Gruppmedlemmarnas medverkan i arbetet är att formulera den rapport som ska redogöra för arbetets process och resultat. Rapporten som vi skapar innehåller således vad vi ska göra, vad vi gjort och slutliga reflektioner på resultat och arbetsprocess. I arbetets gång kommer vi bägge projektmedlemmar vara likvärdigt delaktiga i författande av rapport samt utveckling av produkt. Detta med nackdel av att vi blir lite mindre effektiva, är för att säkerställa att arbetet alltid fortsätter framåt även vid bortfall av gruppmedlemmar.

Från uppdragsgivarens sida är bidraget från arbetslaget ADA uppdelat. Staffan Lyttkens har en central roll som samordnare mellan oss och Skatteverket, medan Aleksander Bertwig bidrar med teknisk kompetens. Båda bidrar som uppdragsgivare och produktägare, med designrelaterad kunskap. De anger i vilken riktning produkten bör utvecklas och vilket “utseende” den bör få.

2. Problematisering och forskningsanknytning

I detta kapitel kommer vi reflektera över vilka problem vi kan komma att ställas inför i vårt arbete. Bland de problem som vi identifierar letar vi sedan efter redan befintliga lösningar i förankrade forsknings artiklar och publicerade tekniska lösningar. Vi vill även genomföra problematiseringen med grund i forskningslitteratur som rör kvalitetssäkring. Detta för att kunna uppnå en högre kvalité i systemet vi utvecklar såväl som att leverera ett värdefullt och betydelsefullt system till beställaren, Skatteverket.

2.1. Problematisering

Vid utveckling av AI-modeller krävs det att modellen skall kunna identifiera funktionalitet och "tänka" själv hur den skall hanteras. För att uppnå detta stadie behöver man lära modellen vad som är korrekt och fel vid utförande av uppgift. Detta görs genom maskininlärning, då man genom testdata låter modellen genomföra tänkt funktionalitet och kontrollerar att den fungerar korrekt. Vid felaktigheter skall modellen informeras om vad som gått fel och hur det skall kunna korrigeras till korrekt data för att inte utföra samma felaktighet igen. Detta är ett kritiskt steg inom AI innan lansering för att systemet modellen skall implementeras i skall kunna hålla sin integritet.

I vårt uppdrag mot Skatteverket genomförs ett projekt att skapa en AI-modell som läser av MRZ-koden längst ned på id-handlingar och pass. Problemet som uppstått vid avläsning är att ord misstolkas eller tolkas som flera. Vilket leder till att modellen behöver genomgå en inlärningsperiod med testdata för olika fel och misstolkningar som kan uppstå för att kunna identifiera dem och lära sig åtgärda och hantera dem.

Det praktiska uppdraget vi skall genomföra är ett exempel på problemklassen Kvalitetssäkring i maskininlärning.

2.2. Liknande system / utredningar / uppdrag

Vi har genom att söka på nätet för att hitta artiklar och forskningsrapporter från flera olika källor som beskriver uppdrag som liknar vårt. För en enkelt och snabbt bilda en uppfattning om uppdragets ämnesområde användes en sökmotor för att hitta artiklar som beskriver kända för- och nackdelar kring arbeten med bildavläsning. Sökord som nyttjades i sökmotorn var; *machine learning image recognition*, *machine learning OCR* och *machine learning MRZ*. Det är de artiklar som vi ansåg värdefulla för att bilda en klok förståelse för ämnet som vi beskriver i detta avsnitt. I nästa avsnitt blir vi mer djupgående i ämnet och beskriver där vilken forskningslitteratur vi hittade och ansåg var värdefull för uppdraget.

Vid sökning på machine learning (ML) och bildavläsning återfinns en artikel från Infrrd. I denna artikel redovisas några fundamentala skillnader mellan tidigare bildavläsnings system och de nya som utnyttjar AI (och maskininlärning) för att lösa specifika uppgifter. De äldre systemen, i ett exempel från ett finansföretag som Infrrd samarbetade med, hade endast företagets system en träffsäkerhet på 60% när det kom till avläsningar. Detta eftersom det äldre systemen jobbade efter mallar för att rekognosera vilken typ av data som ska läsas av. Däremot kunde vanliga dokument som skannades variera i format och utseende, från datorskriven och handskriven text till avläsning av bild, dessa variationer är för komplexa för att fångas upp av en mall och kunde ibland leda till systemet kraschar. För att hantera dessa brister i de gamla systemen började man undersöka hur artificiell intelligens kunde hjälpa till att öka systemens noggrannhet. Artificiell intelligens och därigenom maskininlärning kan till skillnad från det gamla systemet bearbeta data dynamiskt, och därtill hantera komplexitet bättre. Detta var något som kunde utnyttjas för att hantera den initiala avläsningen av dokument och bilder. Företaget kunde således utnyttja maskininlärning för att öka deras OCR systems effekt och noggrannhet. (Infrrd, 2020)

I en bloggartikel från Nanonets, diskuteras och redogörs de vilka typer av avläsningar som utförs på id-handlingar. Vanligtvis används nyckel-värdeparning vid uthämtning av data från id-handlingar. Det betyder att det finns fasta fält som, namn, födelsedatum och nationalitet. Givetvis behövs ett flertal olika mallar för att möjliggöra hantering av de olika utseenden id-handlingar kan ha. Tekniken är den samma oavsett vilken mall som används. Hitta ett fält (nyckeln) och hämta ut värdet kopplat till nyckeln. I vårt aktuella fall är det MRZ-koden längst ner i id-handlingen som är av intresse. Vilket man i artikeln, inte anser är lika vanligt att man hämtar ut. Det betyder att det finns en begränsad mängd lösningar och algoritmer för denna typ av problem. I artikeln lyfts även svårigheter och utmaningar med att skanna av id-handlingar, vilket bekräftar de problem som vi själva och Skatteverket har identifierat. Det vill säga, id-handlingar kan ha skannats i fel vinklar, tagits med låg upplösning av pixlar eller i dåligt ljus. Det är i dessa tillfällen som man med fördel antingen kan nyttja bildredigerings program för att försöka förbättra bildkvalitén eller skapa algoritmer som tränas på att tolka rätt när tecken sättsamman eller feltolkas. (Nanonets. 2021)

(Fig 3. Exempelbild på hur en ID-handling ser ut med MRZ-kod längst ned)

2.3. Relaterad forskning / litteratur

Vi har i litteraturgenomgången identifierat fyra huvudsakliga områden att utgå ifrån: *machine learning, testning med syntetisk data, substringsearch algorithm* och utvärdering. De tre huvudområdena är de tillika sökorden vi använde för att hitta litteraturen som var relevant för oss i sökningar på Google Scholar. För området utvärdering har vi i stort ett omfattande bibliotek av litteratur från tidigare och nuvarande kurs. Vilket gör att vi känner oss bekväma och trygga i att nyttja de redan kända, och nytillkomna, källorna för att kunna bygga en modell och utvärdera detta projekts designprocess och resultat.

Machine Learning:

Taiwo Oladipupo Ayodele. (2010). Types of Machine Learning Algorithms

Artikeln avhandlar vilka olika typer av algoritmer som används inom maskininlärning. Typerna består utav övervakad, oövervakad, semi-övervakad, förstärkning, transduktion och lära sig för att lära. Övervakningen handlar om det finns exempel på vad output skall bli att bistå eller inte. I Förstärkning utgår algoritmen från en policy, varje aktion har en påverkan som ger feedback till algoritmen om det är korrekt. Transduktion liknar övervakad men skillnaden är att nya outputs förutspås genom träning av inputs och outputs. Lära sig för att lära utgår från att algoritmen skall lära sig själv genom tidigare erfarenheter.

Houssem Ben Braieka, Foutse Khomha. (2020). On Testing Machine Learning Programs

Maskininlärning programmens karaktär gör det svårt att resonera om deras beteende. Nyligen har forskare börjat utveckla testtekniker för att hjälpa utvecklare att upptäcka fel och testa maskininlärning-program. I det här dokumentet beskrivs den generiska processen för en ML-programskapning, från dataförberedelse till distribution av modell i produktion, och förklarar de viktigaste källorna till fel i ett maskininlärning-program. Därefter granskas testtekniker som föreslås i litteraturen för att hjälpa till att upptäcka dessa fel både på modell- och implementeringsnivåer. Man förklarar kontexten i vilken de kan användas samt deras förväntade resultat. Luckor identifierar också i litteraturen relaterade till testning av maskininlärning-program och föreslår framtida forskningsinstruktioner för det vetenskapliga samfundet. Genom att konsolidera framstegen med att testa maskininlärning-program i ett enda dokument hoppas författarna kunna utrusta maskininlärning- och programvaruteknik med ett referensdokument som framtidsforskningsinsatser kan byggas på.

Testning med syntetisk data:

Mark A. Whiting, Jereme Haack, Carrie Varley. (2008). Creating Realistic, Scenario-Based Synthetic Data for Test and Evaluation of Information Analytics Software

I artikeln presenteras NVAC Threat Stream Generator-projektets metod och verktyg för att skapa realistiska, syntetiska datamängder för testning och utvärdering av programvara för informationsanalys. Författarna anser att dem hjälper till att utveckla utvärderingsprocesser genom att skapa data med känd marksannhet, där ett scenario skapas och kartlägger element av ett hot från scenariot till data. Sedan testas resultatet i tävlingar där utvecklare och informationsanalytiker försöker "hitta hotet" i data. Feedback från tävlingarna har varit mycket värdefullt för att hjälpa dem att utveckla sin process för dataskapande. Verktygsbyggare rapporterar att tävlingarna hjälper dem att utveckla bättre programvara och även feedback från informationsanalytiker har varit uppmuntrande.

Tin Kam Ho, Henry S. Baird. (1995). Evaluation of OCR Accuracy Using Synthetic Data

I artikeln utvärderas testning av OCR-modell genom systematiskt felaktiga skapta testdata. Datan skapades i en modell med parametrar som suddighet, pixelstorlek och tröskelvärden. Dom kom fram till att suddighet och tröskelvärden hade mycket större inverkan på noggrannheten i modellen, medan pixelstorleken endast påverkade när något av dem andra två låg över god marginal av prestanda.

Substringsearch algorithm:

Edward M. Mocreight. (1976). A Space-Economical Suffix Tree Construction Algorithm

Denna artikel från 1976 presenterar hur man genom en binär trädalgorithm kan konstruera en metod som jämför strängar för att hitta skillnader, genom att dela upp substängar i olika grenar för att sedan jämföra alla möjliga skillnader som går att hitta i grenarna.

Nuvarande/tidigare kurslitteratur för utvärdering:

DeLone, W. H., & McLean, E. R. (1992). Information Systems Success: The Quest for the Dependent Variable. Information Systems Research.

Artikeln syftar till att skapa klarhet bland alla de studier som hade genomförts, under tidigare decennium, på vilka faktorer som spelar roll för framgången hos ett IS system. En del av den klarheten fås genom att identifiera sex stora kategorier:

Systemkvalitet: hur väl systemet fungerar prestandamässigt.

Informationskvalitet: kvaliteten på data som systemet levererar till användaren. Hur användbar och korrekt den är.

Användning: Kvaliteten på systemet i användning.

Användartillfredsställning: Hur nöjd användaren är med att använda systemet.

Individuell påverkan: Effekten på användaren som informationen har som systemet levererar.

Organisatorisk påverkan: Vilken effekt för organisationen informationen har som systemet levererar.

Genom att utnyttja dessa kategorier skulle man kunna konceptuellt och empiriskt mäta framgången hos ett IS system. Det är även med stöd av denna litteratur som vi hoppas kunna göra de prioriteringar i design av systemet som vi tror ger mest kvalité och därigenom mest tillfredsställes till beställaren (eng. user satisfaction).

Hevner, Alan R. (2007) "A Three Cycle View of Design Science Research," Scandinavian Journal of Information Systems: Vol. 19: Iss. 2, Article 4.

Artikeln vill förtydliga vilka cykler som lever i designforskning. Tre cyklar identifieras som, relevanscykeln, designcykeln och "beviskrafts" cykeln (eng. Rigor cycel). I den första cykeln undersöker man vilka problem och möjligheter som finns i den miljön/område man avser att forska i. Där i ingår specificering av krav och fälttester. Den andra cykeln är där olika artefakter designas och utvärderas. Detta genomförs i flera iterationer med flera designförslag, för att säkerställa bästa möjliga lösningen. Sista cykeln är där man vill fastställa om den forskningsdesign man bedrivit bidrar till att utöka vår kollektiva kunskap för ämnet, eller om man har genomfört en variant av något som redan har gjorts.

Heinz K. Klein and Michael D. Myers. (1999). A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems Stable

Artikeln syftar till att svara på frågor kring hur man kan säkerställa kvalitén i deduktiv forskning. En forskningsmetod som hade en ökande trend när artikeln skrevs. I artikeln ges förslag på principer som bör följas vid skapande och utvärdering av fältstudier inom området för informationssystem.

Stefan Cronholm, Göran Goldkuhl.(2003) Six Generic Types of Information Systems Evaluation

Artikeln syftar till att ge ramverk för olika utvärderingsmetoder. I denna återges kategorier av utvärderingar så som formativa eller summativa. Vardera kategorier medför säregna krav och måttstockar. Förutom tidigare nämnda kategorier kan tre strategier för utvärdering identifieras, målbaserad, målfri och kravbaserad utvärdering. Det är framför allt en av dessa strategier vi är intresserade av att nyttja när vi senare kommer granska vårt arbete.

2.4. Implikationer för uppdraget

Här sammanfattar vi litteraturgenomgången och beskriver kortfattat hur litteraturen influerat vår process och vår lösning. Vi har alltså för att tydliggöra skapat en tabell som ska visa på var vi anser att litteraturen, sin vi beskrivit tidigare, kommer vara relevant för vårt arbete.

Projektsteg	***Litteratur***	***Implikation***
Designprocess	DeLone, W. H., & McLean, E. R. (1992). Information Systems Success: The Quest for the Dependent Variable. Information Systems Research.	Genom att utgå från de sex kvalitetskategorierna definiera vilka kvaliteter vi vill lägga vikt på i utvecklingsprocessen.
	Taiwo Oladipupo Ayodele. *(2010)* Types of Machine Learning Algorithms	Vi har att tillgå exempel på outputkrav inom projektet, vilket leder till att med hjälp av artikel kan vi utforma en övervakad typ av algoritm för modellen.
	Baskerville, R., Kaul, M., & Storey, V. (2011). Unpacking the duality of design scienc	Vi kan använda oss av litteraturen för att få en ökad förståelse och struktur för att design systemet (detta avser nivå 1 i Three-Level Design framework
	Mark A. Whiting, Jereme Haack, Carrie Varley. (2008). Creating Realistic, Scenario-Based Synthetic Data for Test and Evaluation of Information Analytics Software	I stycket “Lessons Learned” finns en punktlista som man kan bistå för att enklare skapa syntetiska data.
Programvara	Edward M. Mocreight. (1976). A Space-Economical Suffix Tree Construction Algorithm	Vi kommer att använda algoritmen för att skapa metoden som skall kontrollera avläst MRZ-sträng med en korrekt sträng från en textfil.
Utvärdering	Cronholm, S., & Goldkuhl, G. (2003).	Vi kommer nyttja ramverket för målbaserad utvärdering när vi granskar vårat arbete.

	Six Generic Types of Information Systems Evaluation	
	Baskerville, R., Kaul, M., & Storey, V. (2011). Unpacking the duality of design science	Vi kan använda oss av litteraturen för att få en ökad förståelse och struktur vid utvärdering och tillämpning av forskning till design (detta avser nivå 2 i Three-Level Design framework)
	DeLone, W. H., & McLean, E. R. (1992). Information Systems Success: The Quest for the Dependent Variable. Information Systems Research.	Vi tittade på dessa sex kategorier i början av designprocessen. I utvärderingsfasen vill vi se hur väl vårt system lever upp till dem.

(Tab 1. Lista på litteratur som influerar vår process och lösningar)

2.5. Förväntade kunskapsbidrag

Utifrån Baskerville et al (2011) tror vi att vårt arbete kommer närmast ett mer praktiskt än akademiskt scenarier vilket motsvarar *Designing with research.* (Baskerville, R., Kaul, M., & Storey, V. 2011). Här ser vi att vår design av systemet sker med stöd i befintlig forskning för att undvika, annars vanliga, designbrister. Vår design av systemet har alltså fundamentalt stöd från fastställd forskning i syfte att skapa en produkt av hög kvalité. Arbetet och designen av produkten syftar därför inte till att öka en förståelse för designprocessen, inte heller för forskningsprocessen. Vår förhoppning är däremot att arbetet ska kunna bidra med någon form av kunskap till design och/eller forskningsområdet inom ämnet. Den kunskapen kan vara om hur vår designprocess såg ut och till vilken grad vi levererar en produkt av god kvalité. Kunskapen skulle även kunna vara den om hur väl det går att skapa syntetiska testdata och genomföra tester på aktuell testdata. Tester som ska generera en produkt som sedan kan integreras i ett system som inte vi, som utvecklare, har behörighet till.

Ågerfalk (2014) belyser vikten av empirisk kontra teoretisk erfarenhet. Vårt arbete har som utgångspunkt att styrka praktisk och empirisk erfarenhet. Men det betyder inte att vi inte kan, genom att studera vårt arbete, utvinna dess teoretiska innebörd. Precis som Ågerfalk hänvisar till så finns det många exempel inom områden som utvinner teoretiska kunskapsbidrag genom empirisk erfarenhet. Ett av dom medicin, då man genom att studera hur patienter reagerar till olika mediciner kan skapa en teoretisk kunskapsbank.

Arbetet vi genomför kommer med största sannolikhet att ligga inom ramarna för *routine design*, Gregor & Hevner (2013). Eftersom vi arbetar med att tillämpa redan befintlig forskning i en designprocess som syftar till att skapa en redan "uppfunnen" produkt. Vi använder oss alltså av redan befintliga lösningar för att lösa redan kända problem. Enligt författarna är denna typ av forskning på gränsen till att ens kunna anses vara kunskapsbidragande. Däremot finns det fall där nya upptäckter sker även inom dessa ramar av forskning, även om det är mer sällan än ofta. Den här diskussionen kring det förväntade kunskapsbidraget skiljer sig inte avsevärt från den som

fördes med hänvisning till Baskerville et al (2011). Däremot finns det enstaka olikheter. Gregor & Hevner verkar förmedla en inställning om att en så kallad *routine design*, endast bidrar i få fall till nytt kunskapsbidrag. Även om vårt arbete, med stor rimlighet, inte kommer medföra avsevärda upptäckter, kan vi möjligtvis komma med nya perspektiv på designprocessen som vi har genomfört eller på den produkt vi utvecklat.

3. Forsknings- och utvecklingsprocessen

I detta kapitel kommer vi redogöra för hur vi har genomfört vår forsknings- och utvecklingsprocess. Vi kommer ge en överblick på projektet och reflektera över några av de val vi gjort för arbetsmetoder och tekniska verktyg. Vi kommer även att diskutera vilken typ av data som vi kommer arbeta med och vad den förväntas medföra till arbetets resultat. Slutligen diskuterar vi vilken forskningsetik vi arbetar efter, vilka rättigheter vi har från Skatteverket och vilka antaganden vi gör i uppdraget.

3.1. Projektöversikt

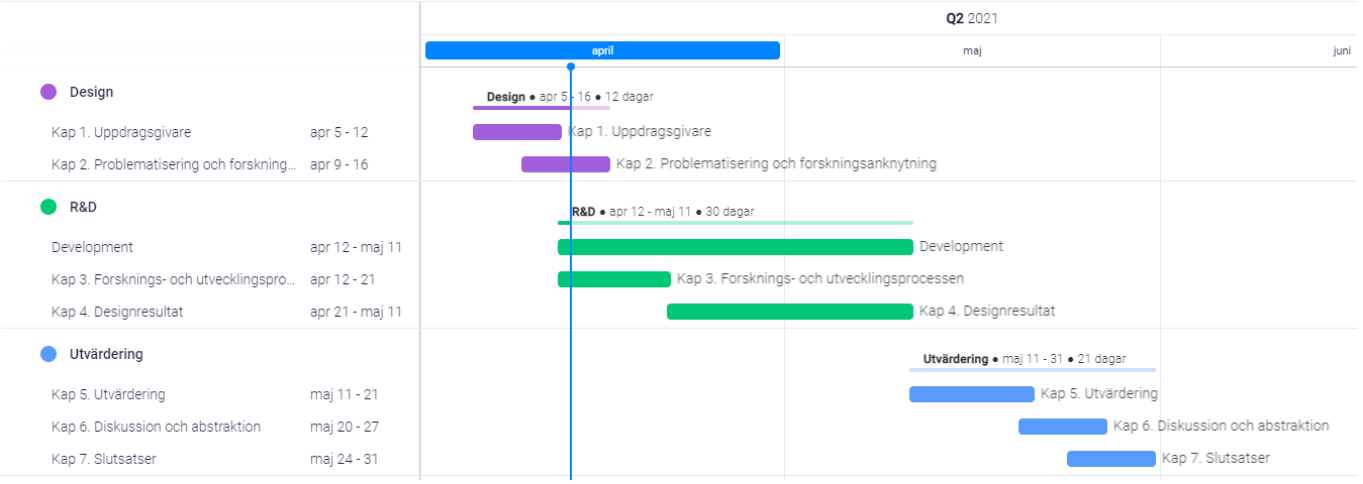
För att effektivt kunna visa en överblick av projektet och dess olika steg så har vi valt att skapa ett gantt schema (fig. 4). Schemat visar en tidslinje över hela projektets gång med dem olika steg som bestämts som liggande staplar när dem utspelar sig. Detta ger en tydlig bild av hur projektets framsteg förhåller sig till tidsplanen. Vi har valt att hålla oss till relativt överblickande steg för detta i sambandet mindre projekt, men det går att gå ned på väldigt små detaljsteg för att kunna följa upp daglig status. (Projektledning. 2018).

Gantt schema (fig. 4) beskriver tydligt:

- Vad som ska göras
- Tidsåtgång som varje arbetsuppgift bör ta
- Vem/vilka som åtar sig varje uppgift
- Vilka olika arbetsuppgifter som överlappar
- En proximering av när varje del av projektet förväntas genomföras samt vara färdigt

Enligt Harvey Maylors artikel *Beyond the Gantt Chart: Project Management Moving on* anses gantt scheman skapa problem inom projekt som att det uppmuntrar till ett-steg-i-taget inom planering och kan få anställda att emotsätta sig att utmana schemat för att skapa eget momentum i sitt arbete. Det gör även att projektledaren behöver kontrollera projektet mer än att låta team-medlemmarna skapa ansvarskänsla för tidsplaner. (Harvey Maylors. 2001).

Vi anser att detta påverkar större och mer omfattande projekt. För vårt mindre projekt så ger gantt schemat oss en klar överblick, och skapar tydlighet för alla inblandade. Samt eftersom vi är våra egna projektledare kommer ansvarskänsla per automatik.



(Fig 4. Gantt schema med översikt av vårt projekt)

3.2. Metodstöd för forskning och utveckling

Vad gäller vår metodik för utveckling i vårt uppdrag så tar vi stöd från befintliga och väletablerade arbetsprocesser och metoder. Vi ser att ett självklart val är en agil-arbetsprocess där Scrum tar en central roll i vårt arbete. Även Skatteverket jobbar enligt en variant av Scrum-modellen så där finns ytterligare skäl till att vi försöker jobba likvärdigt vid utveckling. Vår arbetsprocess kommer att vara agil eftersom vi planerar vårt arbete i mindre block som avses slutföras under kortare tidsrymder.

För att tydliggöra hur vi ämnar ta stöd från forskning och befintliga metoder kommer vi ge exempel på arbetsprocessen vi följer, tillika blir exemplen ett intyg på att vi jobbar mot litteraturen. Det första exemplet är att vi jobbar, under två veckor tillika en sprint, med att bygga en forsknings- och metodgrund inför design av systemet. Detta kopplar vi direkt till relevanscykeln (eng. *Relevance Cycle*), Hevner (2007). Det är inom ramarna för denna sprint som vi formulerar och specificerar kraven från beställaren, Skatteverket. Vi skapar även här underlag som ska ge oss stöd och möjliggöra en högre kvalité i de system vi kommer börja designa i nästa sprint. Den nästkommande sprinten som tillika blir början av nästa steg i designforskningens forsknings cyklar, designcykeln (eng. *Design Cycle*). Inom ramarna för designcykeln ämnar vi att genom iterativa utvecklingscyklar skapa olika typer av design på systemet. Efter att en design skapats görs en utvärdering och jämförelse mot de andra designerna i syfte att utforma den, för uppgiften, mest optimala designen på systemet. Sist följer en process, iteration eller sprint, där vi knyter an den design som vi skapat till forskning för att antingen bidra med ny, eller befästa redan ackumulerad kunskap.

Ytterligare ett exempel kan ges på att vi följer en agil arbetsprocess därtill en variant av Scrum-modellen. Vi har fastställt och avsatt tid för att veckovis ha möten med våra kontaktpersoner på Skatteverket. Detta i syfte för att genom hela arbetsgång säkerställa att vi arbetar mot ett gemensamt mål där slutresultatet är att vi levererar en produkt som Skatteverket är nöjda med och kan nyttja i sin verksamhet.

3.3. Datainsamling

Vår datainsamling eller som Goldkuhl (2019) föredrar, genererade data, består av syntetiska id-handlingar. Vad Goldkuhl vill framhäva är att det är en väsentlig skillnad på datainsamling och att generera den. Där är datainsamling då man med lätthet hämtar in data från någon eller något, medan vid generering av data måste man de-facto skapa data. Det sistnämnd är vad vi ämnade att göra eftersom vi inte hade tillgång till de testdata som Skatteverket har.

Följaktligen behövde vi generera data. De syntetiska id-handlingarna skulle återspegla tre olika typ-fall av fel; två tecken tolkas som ett, ett tecken tolkas som ett annat, ett tecken tolkas som flera tecken. Dessa typer av syntetiska data skapade vi genom att studera id-handlingar med MRZ-kod, för att uppnå en högre naturtrogenhet. Vi ville sedan efterlikna olika scenarier där id-handlingen inte är i perfekt skick för att avläsas. Detta återskapar vi genom att fysiskt eller digitalt förvränga de id-handlingar som vi skapat. Förvrängningarna kommer ta skepnad i fysiska kaffefläckar, fysiskt tillgjorda förslitningar (så delar av tecken saknas), digitalt konstruerade suddigheter eller försämring av pixelkvalité.

För att skapa enklare och bättre syntetiska testdata valde vi att följa dessa punkter från (Mark A. Whiting, Jereme Haack, Carrie Varley. 2008):

- Utveckla dina första testdata för hand. Du kommer att fåen uppskattning av både skapelseprocessen och vilkaverktyg du behöver för att öka processen.
- Involvera informationsanalytiker i början av dinprocesser. Detta kommer att ge dig ovärderlig hjälpi att skapa trovärdig testdata och också tillhandahållatrovärdighet när andra informationsanalytiker granskardata.
- Bestäm vilka informationskomponenter du villfokusera på i din utvärdering: verktyg, människor, uppgifter,processer eller resultat.

- Tänk på att designfasen för skapande av testdata kanta längre tid än genererings fasen. Komplexascenarier resulterar ofta i mycket komplex testdata.

Istället för informationsanalytiker lät vi utvecklare från uppdragsgivaren som jobbar med AI-modellen granska vår syntetiska testdata med feedback för hur den ställde sig mot riktigt data i form av komplexitet vid felaktigheter som bör uppstå.

3.4. Dataanalys

En utförlig dataanalys har inte varit aktuell för uppdraget vi fått. Vi har diskuterat med både samordnare och utvecklare på Skatteverket, i syfte att få en korrekt uppfattning om testdatats förväntade utseende. Vi kunde genom att läsa artiklarna från Infrrd (2020) och Nanonets (2021) få en fördjupad förståelse i testdatats förväntade syfte och problematik relaterad till bildavläsnings processer. Vi ville alltså genom litteraturen skapa en större förståelse för hur man arbetar med ML i uppdrag som liknar de vi blivit tilldelade. Därefter behövde vi hitta litteratur som stöd till den algoritm vi nyttjar för att validera substrängar. Det var här Mocreight (1976) blev för oss relevant med en trädalgoritm som än idag levererar i såväl komplexitetsklass mot stora O, det vill säga prestanda, och även i funktion.

3.5. Forskningsetik och sekretess

Vi kommer förhålla oss till Vetenskapsrådets riktlinjer om god forskningssed. (Vetenskapsrådet. 2017). Personer vi samarbetar med inom projektet har alltid rätt till att hålla sig anonyma. Innan publicering av denna rapport kommer ansvarig från Skatteverket kunna ta del av den för att godkänna publicering alternativt förändringar för att kunna fylla den sekretess som lagen kräver från en statlig organisation som Skatteverket. Skulle vi behöva utföra intervjuer kommer respondenten i fråga informeras om Vetenskapsrådets rekommendationer vid början av intervjuer.

Respondenten har rätt till att:

- ...vara anonym.
- ...avbryta när den vill.
- ...ta delar av resultat innan publicering.
- ...ta bort vissa delar pga. Sekretess.

3.6. IP-rättigheter

Vi har i möte med Skatteverket (2021-03-30*)* slutit ett muntligt avtal att hantera all fysisk kod och dokumentation som skapas genom detta projekt som öppen källkod. Vilket innebär att alla parter har rätt att nyttja funktionaliteten av applikationen efter avslutat projekt. Vi kommer inte ta del av organisationens källkoder och system utan bygger en artefakt utifrån syntetiska testdata, således uppstår ingen komplikation med sekretesslagen.

3.7. Antaganden

Våra förutsättningar vid utvecklingen av denna applikation att:

- Systemet som skall integrera klarar funktionalitet från Java 11 och äldre versioner.
- Innan integrering kommer den nya boxfilen och bilden att granskas manuellt och eventuellt justeras innan de används för att träna förbättrad OCRB-modell av personal på Skatteverket. Eftersom vi inte har tillgång till deras system och möjlighet utföra testerna själva på verkliga data.

4. Designresultat

I detta avsnitt kommer vi att redogöra i detalj för hur vår utvecklingsprocess gick till samt presentera resultaten ifrån slutet av kapitlet. Redogörelsen kommer bland annat utgöras av hur vi, i grupp, jobbade för att bedriva en utvecklingsprocess som var präglad av effektivitet och kvalité. Vi vill även beskriva vilka delar av litteraturen som har varit av särskild betydelse för vår utvecklingsprocess och det slutgiltiga resultatet. Vi kommer även ge en kortare presentation av processens resultat, med syfte att ge en djupare reflektion i nästkommande kapitel, *5. Utvärdering*.

4.1. Arbetsprocessen

Arbetsmetoder

Vi genomför arbetet inom ramarna för SCRUM-metoden men med huvudsakligt inslag av XP-metodiken. Att vi inriktar oss mer specifikt mot XP beror på att vi är en liten grupp som arbetar självständigt med uppdraget. Vidare medför vår gruppstorlek att vi kan friare välja bland vilka metoder vi ska nyttja, där XP medför flera enkla och effektiva verktyg till vår process, Altexsoft (2021). Arbetsprocessen vi bedriver behöver alltså inte ta hänsyn till alla de metoder som finns inom de agila-ramverken, eftersom många av de metoderna är avsedda att hantera samverkan i större personalkonstellationer än vårt team. Vi nyttjar generöst verktyget par programmering för att uppnå en högre kvalité med en potentiell kostnad på effektiviteten i utvecklingsprocessen. Vi ansåg däremot att det fanns mer nytta i att två hjärnor jobbade parallellt för att undvika enklare fartgupp och kompetensbortfall. Ett arbete som vi båda deltog likvärdigt i gav oss med all förmodan en redundans, däremot möjliggjorde detta att vi alltid hade en rörelse framåt. Att vår arbetsprocess alltid rörde sig framåt gav oss många gånger olika lösningar på de problem vi möte längs vägen. Ett problem som man kan ha, och som vi hade, var att alla lösningar inte alltid var optimala att arbeta vidare ifrån. För att hantera sådana situationer blev versionshantering med verktyget, git och bitbucket, ett för oss fundamentalt nödvändigt verktyg för processen vi bedrev. Ett verktyg som är väletablerat inom systemutvecklingsprocesser, Mann (2021) och Azarian (2013). Genom versionshanteringen kunde en programmerare göra avsteg från den pågående utbyggnaden för att testa andra lösningar. Att vid sidan av att testa andra lösningar gjorde att vi kontinuerligt utforskade nya lösningars potential till att vara bättre än de som vi annars hade ansett, för tillfället, vara mest optimala. Genom detta arbetssätt lyckades vi ofta ta steg ansatsvis framåt vilket gav oss en högre hastighet igenom utvecklingsprocessen. Arbetssättet gav oss alltså hastighet men också med resultat i en tillfredställande riktning.

En annan del av kvalitésäkringen i arbetet var, vilket också bekräftar anammandet av de agila arbetssätt, att vi hade regelbunden kontakt med uppdragsgivaren, Skatteverket. Genom att vi, i många fall, hade möte veckovis med våra kontaktpersoner kunde vi regelbundet kontrollera att vi utvecklade något som möte deras förväntningar och krav. Vi kunde alltså vid avstämningarna diskutera våra lösningar, problem och frågor. Detta gjorde att vi, efter varje avstämning, fick större klarhet av vår uppgift. Skatteverket kunde då också se hur väl vi förmådde och hann med att implementera uppgifterna de gav oss. Något som i sin tur gjorde att de, i rimlig takt, kunde stegra komplexiteten i systemet vi utvecklade.

4.2. Arkitekturbeskrivningar

I detta avsnitt beskrivs viktiga, och kritiska, delar av vår applikations arkitektur och design. Avsnittet innehåller alltså vilka design principer som vår applikation eftersträvar. Utöver detta tydliggörs applikationens centrala logik genom flödesscheman och beskriver även mer i detalj de avgörande algoritmer vi använt för att lösa applikationens primära funktion.

4.2.1 SOLID Designprincip

Genom att följa dem 5 design principer som SOLID står för kunde vi leverera ett system som är enklare att utveckla, förstå, underhålla och utökas. Principerna intruducerades av Robert C. Martin i sin artikel *"Design Principles and Design Patterns"* år 2000. (Martin. R, 2000)

Principerna står för (se bilagor för exempel på respektive princip):

- **S** *(Single responsibility principle)*
  Alla metoder och klasser är uppdelade att endast innehålla ett ansvar. Där metoder har haft fler har vi brutit ut och delat upp i olika metoder i klassen som i stället anropas.
  (bilaga 8.2)

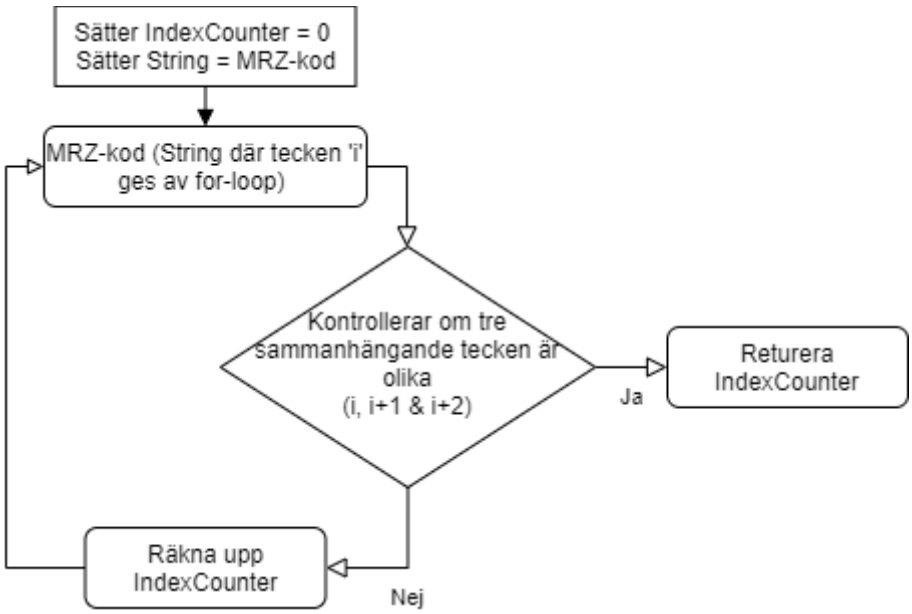
- **O** *(Open closed principle)*
  Principen står för att applikationen skall vara öppen för förlängning men stängd för förändring.
  (bilaga 8.3)

- **L** *(Liskov Substitution Principle)*
  Ingen klass i artefakten ärver av någon annan klass som är implementerad av en 3e klass.
  (bilaga 8.4)

- **I** *(Interface Segregation Principle)*
  Varje subklass som implementerar ett interface eller en abstrakt klass ärver inga metoder som den inte använder.
  (bilaga 8.5)

- **D** *(Dependency Inversion Principle)*
  Inga hög-nivå klasser är låsta till specifika instanser av låg-nivåklasser.
  (bilaga 8.6)

4.2.2 Flödesscheman för central logik i systemet och viktiga algoritmer



(Fig 5. Flödesschema för central logik i systemet)

4.2.2.1 Vladimir Levenshtein Distance Algorithm

Nyckeln till att lösa uppdraget vi ålagts är att kunna jämföra två strängar med tecken för att definiera vilka skillnader som finns och vart. Det här var något som matematikern Vladimir Leibenstein redan 1965 kämpade med. Han skapade då sin "Distance Algorithm" för att jämföra två olika ord och beräkna avståndet för att ändra det ena ordet till det andra. (CueLogic. 2017)

Definition

Matematiskt ges Levenshtein Distance mellan två strängar a, b (längd | a | respektive | b |) av leva, b (| a |, | b |) där:

$$\operatorname{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \operatorname{lev}_{a,b}(i-1,j) + 1 \\ \operatorname{lev}_{a,b}(i,j-1) + 1 \\ \operatorname{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

(Fig 6. The Levenshtein Algorithm)

1 (ai ≠ bi) är indikatorfunktionen lika med 0 när ai ≠ bi och lika med 1 annars, och leva, b (i, j) är avståndet mellan de första i-tecknen i a och de första j-tecknen i b.

Observera att det första elementet i minimum motsvarar radering (från a till b), det andra till insättning och det tredje som matchar eller inte matchar, beroende på om respektive symboler är detsamma. (Exempel på implementation se bilaga 1).

Med hjälp av denna algoritm kan man identifiera var den korrekta strängen börjar för att kunna ta bort irrelevanta tecken som inte ingår i MRZ-koden.

4.2.2.2 Algorithm M

Genom att implementera Levenshteins algoritm löstes endast en del av problemet. Man kan nu extrahera en korrekt sträng från en annan med så få steg som möjligt. Men om strängen man skall extrahera in är helt korrekt blir algoritmen ofullständig. Detta problem presenterades 1965 av Edvard M Mocreight i sin artikel A Space-Economical Suffix Tree Construction Algorithm. (Edvard M Mocreight. 1965).

Professor Peter Weiner från Santa Monica College hade även han upptäckt detta problem och skapat en algoritm för att implementera en funktion som försöker matcha understrängar mot huvudsträngar i alla möjliga inriktningar. (Weiner, P. 1965). Problemet med algoritmen var att det blev fort långsam och ineffektiv.

I sin artikel presenterade Edvard M Mocreight en ny algoritm som han kallade “Algoritm M”. Algoritmen ger samma resultat som Weiners linjära mönstermatchnings algoritm, men genom att integrera Levenshteins algoritm för att genom så få steg som möjligt matcha strängar och understrängar kunna använda 25 procent mindre datautrymme samt effektivisera implementationen. Algoritmen ger även en indexering för varje steg i algoritmen.

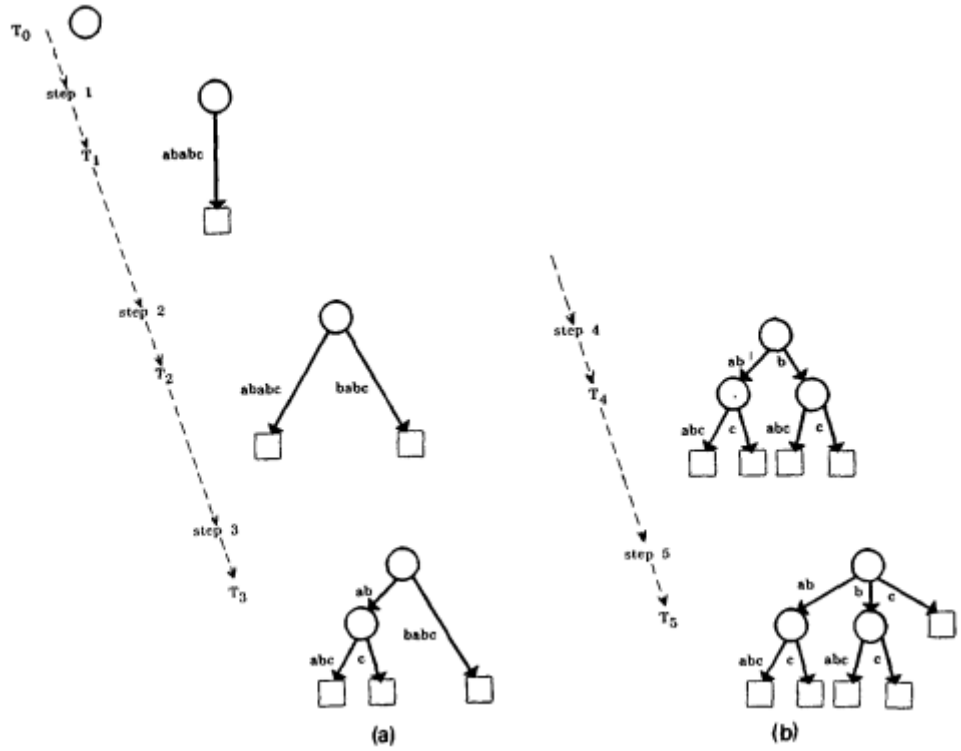
Definition

Algoritm M börjar med ett tomt träd till och går in i banor som motsvararsuffix av S en i taget, från längsta till kortaste.

Trädet T motsvarar vår exempelsträng S (ababe) skulle konstrueras av algoritmen i stegen som visas i Figur 1, ett steg per suffix av S:

Vi definierar att det är suffixet för S som börjar vid teckenposition i. (Position 1 är definierad som den längst till vänster för S, så suf(1) är S.) Under steg z infogar algoritmen en väg som motsvarar strängen suf_1 i trädet $T(i-1)$ för att producera trädet $T(1)$.Vi definierar head som det längsta prefixet för suf(i), som också är ett prefix för suf(j), för vissa j < i. Likaså, head(1) är det längsta prefixet för suf(t) vars utökade lokus existerar inom trädet T(i-1). Vi definierar tail, som suf(2) - head. I vårt exempel är suf(3) = abe,head(3) = ab och tail(3) = e.

Begränsning av S(1) försäkrar oss om att tail inte är tom. Att infoga suf(t) in i trädet T(2-1), gör att det utsträckta headen i T(2-1) hittas, en ny icke-terminal noden är konstruerad för att dela upp den inkommande bågen och bli platsen för head, om det behövs, och slutligen är en ny bågmärkt svans konstrueras från den icke-terminala noden till en ny terminal nod.



(Fig 7. Exempel på trädalgoritm)

Ponera exempel på steg 3, som omvandlar T2 till T3 i figur 1. Algoritmen måste infoga suf(3). Genom att spåra denna sträng inom T2 ser den att head(3) är ab, att den utvidgade platsen för ab är terminal noden längst till vänster och att dess inkommande båge (märkta babe) måste delas. Algoritmen delar upp bågen i två delar, märkta ab och abe, genom att infoga en ny icke-terminal nod. En ny båge märkt e, eller tail3, läggs sedan till från den icke terminala noden till en ny terminalnod.

Implementation

Genom att genom ett användargränssnitt ta in först en korrekt sträng med tecken och sedan en boxmodell som består av avlästa tecken med koordinater från ett id så kan vi först lägga den datan i en Hashmap för att sedan endast hämta tecknen från vår Hashmap och göra till en sträng. Om vi skicka båda strängar till vår implementerade version av “Algorithm A” kan vi utvinna de tre efterfrågade boxmodeller som output.

För att först visa hela MRZ-koden som output, hämtar index på var första substrängarna matchar med if-statser som kontrollerar om just den första delen av MRZ-koden skulle kunna vara felaktigt avläst. Vår applikation är satt till att klara av att hantera om någon eller alla av de 5 första tecknen skulle vara felaktig och ändå kunna hämta korrekt MRZ-kod. Annars anser vi att avläsningen är i för dåligt skick för att kunna analyseras.

När vi har ett startindex så räknar vi efter hur många tecken som skall existera i en korrekt sträng och lägger ihop med startindex, vilket ger oss ett slutindex. Det skapas en ny Hashmap där vi sparar alla tecken ihop med koordinater mellan startindex och slutindex som sedan skickas till en print-klass som skapar en box-fil med rätt koordinater för endast MRZ-koden.

För att kunna ge boxfiler med endast korrekta- och inkorrekta tecken så använder vi oss av den skapade Hashmapen med endast MRZ-koden. Vi kontrollerar varje tecken mot facit för att kunna spara ner antingen alla korrekta eller felaktiga tecken i egna Hashmaps som sedan skapar respektive boxfiler till användaren genom att print-klassen sparar ner en box-fil.

4.2.3 Implementationsspecifik översikt

Här ämnar vi att ge en så specifik bild av de verktyg, ramverk, språk, plattformar mm som vi använt för att utveckla det system som vi presenterat. Syftet med denna specificering är att underlätta eventuella och framtida återskapande av detta system som helhet eller delar av det.

För att börja någonstans så börjar vi där ändå all programmering börjar, i IDE:n. Vi nyttjade ‘IntelliJ IDEA Community Edition’, där krav, från Skatteverket, på programmeringsspråk var Java SDK 11. Kort ska det också tydliggöras att både Windows och Apples operativsystem har nyttjats under utvecklingsprocessen utan att ha stört denna. Vi använde bitbucket och Git som versionshanterare för att synkronisera arbetet mellan de olika datorer som använts. Vi använde gantt-scheman (fig.4) och Trello för att skapa tydlig struktur i arbetsfördelning och tidsåtgång till respektive delar av arbetet. Då Corona-pandemin störde fysiskt sociala interaktioner ersatte vi dessa med digitala chatt- och samtalsrum som Discord och Skype. Primärt använde vi Discord för kommunikation men också för att skärmdela vid par-programmeringen. Skype är ett digitalt kommunikations-verktyg som för nuvarande är fastställd inom Skatteverkets organisation. Således ägde möten med Skatteverket rum igenom detta verktyg.

5. Utvärdering

Nu kommer vi att summera och utvärdera det arbete som har genomförts i den designprocessen som vi har bedrivit. Här ingår en genomgång, samt argumentation för vilka kvaliteter som vår lösning har. Argumentationen kommer grunda sig i den litteratur som vi redan hänvisat till i kapitel *2. Problematisering och forskningsanknytning*. Vi kommer alltså ta stöd i forskningslitteratur för att påvisa betydelsen av kvalitéer som vi har prioriterat i vår lösning. Vi tar även stöd i forskning för att beprövat, metodiskt och ur olika synvinklar utvärdera olika delar av vår designprocess. Utvärderingen görs givetvis med direkt koppling till uppdragsformuleringen för att skapa en så tydlig bild som möjligt för lösningens relevans till Skatteverket. Alltså vi kommer i detta kapitel att gå igenom olika perspektiv för utvärdering och på så sätt ge en tydlig bild över vilka kvalitéer vi anser att vår lösning har.

5.1. Demonstrativ utvärdering

Avstämningsmöten av vårt arbete har genomförts i slutet på varje sprint. Varje sprint var inledningsvis en vecka där mötena låg på torsdag. Detta var sant för större del av utvecklingsprocessen förutom sista sprinten som utgjorts av två veckor. Vid varje möte sker en redovisning av hur arbetet gått i veckan för att sedan visa en demonstration av den nya funktionaliteten i applikationen. Efter demo så ges feedback från kunden (Skatteverket) på vad som är funktionellt med deras system och vad som behövs justeras för att möta deras kriterier. Som utgångspunkt för att enklare kunna stämma av vad som fyller funktionella kriterier används de sex kategorier från *"Information Systems Success: The Quest for the Dependent Variable":*

- Systemkvalitet
- Informationskvalitet
- Användning
- Användartillfredställning
- Individuell påverkan
- Organisatorisk påverkan

All funktionalitet behöver inte klara alla sex kategorier, men det är en lämplig utgångspunkt för att diskutera vilka som är väsentliga. Mötet avslutas med att en lista med utbyggnader och förändringar tas fram, samt prioriteringar för varje nytt tillägg definieras inför nästa sprint.

5.2. Experimentell utvärdering

En viktig del av utvärderingen är att vi har testat den produkt vi har utvecklat. Det skulle vara ytterst oseriöst att skriva en utvärdering om en produkt som inte har genomgått några tester eller simuleringar överhuvudtaget. Med det sagt kan fortfarande de tester vi har genomfört vara bristfälliga eller otillräckliga för att kunna garantera kvalité i alla avseenden. Således är vår förhoppning att följande avsnitt ska ge liknande projekt i framtiden stöd vid experimentella utvärderingar. Vi kommer nu redogöra här för vilka tester vi ansåg vara lämpliga att göra och vilka brister som fanns i dem. En kortare, rent spekulativ, diskussion kommer även föras kring vilka uteblivna tester som hade kunnat genomföras för att ge högre evidens för produktens kvalité.

En viktig del för att lösa uppdraget var att skapa ett system som skulle kunna, vid iteration genom box-filen, indexera samtliga tecken. Denna del var i sig viktig för att vi skulle kunna hämta ut enbart, eller delar av, MRZ-koden från en större samling tecken. Den är del är också viktig att den blir rätt för att vi ska kunna hävda att vårt system uppfyller externa kvalitetsegenskaper som korrekthet (eng. c*orrectness*) och träffsäkerhet (eng. *accuracy*), se McConnell (2004).

Inledningsvis började vi därför med att endast lösa filtreringen så att vi endast behövde hantera själva MRZ-koden. För att underlätta detta jobbade vi mot ett facit som matchade box-filen. Det fanns alltså inga felläsningar i box-filen till en början. Men när vi kom till det stadie där vi kunde indexera, och tillika hämta ut, MRZ-koden kunde vi påbörja enklare tester på vår lösning.

Här är en lista på vad vi ville kontrollera att systemet klarade av att hantera:

- Systemets algoritmer skall klara av att indexera, och välja ut, endast och hela MRZ-koden även då flera tecken läggs till före eller i MRZ-koden.
- Systemets algoritmer ska klara av att i flera olika fall där tecken är felaktiga före och i MRZ-koden fortfarande indexera, och välja ut, endast och hela MRZ-koden.
- Systemets algoritmer ska klara av att fortfarande indexera, och välja ut, MRZ-koden även om liknande tecken-mönster som MRZ-koden börjar med förekommer tidigare i teckensamligen.

Listan var målen vi ville uppfylla och varje iteration genom PDCA-cykeln (eng. Plan-Do-Check-Act) tog oss närmare förverkligandet av dessa. Iterationscykeln, PDCA, är ett viktigt ramverk som har möjlighet att öka kvalitén i en produkt. Detta uppnås genom att man upprepat kontrollerar och experimenterar med metoder för att förbättra och bygga en bättre produkt. (Basili & Shapiro. 1994). En process som också bygger mycket på att man har kontinuerlig feedback. Vi har tillämpat ett sådant iterativt-arbetssätt väl och ett tecken på det är hur effektivt och snabbt vi kunde ta oss till ett fungerande system. Vi jobbade alltså kontinuerligt med ett tydligt ramverk, vad ska vi göra - vi gör det – vi testar och kontrollerar – vi tillämpar lösningen. Utöver att ett flertal sådana iterationer rullar på, så tog vi regelbunden kontakt med uppdragsgivaren, Skatteverket. Det gav oss ytterligare feedback inför planeringen på hur systemet som växte fram skulle se ut efter nästkommande iterationer.

5.3. Tolkande utvärdering

Som tidigare nämnts så utgår vi från de sex kvalitets kategorier som skapats av DeLone och McLean. Vi har valt att lägga fokus på system-, informations-kvalitet och organisatorisk påverkan. Eftersom applikationen inte implementeras i aktuellt system i detta projekt har vi mindre påverkan på hur användbarhet och användartillfredsställning möts i systemet vid drift. Den del vi i projektet kan bistå med integreras i system-, informations-kvalitet då om dessa möter kvalitetskraven från uppdragsgivare skapas per automatik en viss användbarhet samt användartillfredsställning. (DeLone & McLean, 1992).

För att uppnå hög systemkvalitet har fokus varit att skapa en applikation som är robust och ger hög prestanda. Vi har följt SOLID-principen där möjlighet funnits, för att göra applikationen enklare att utveckla, förstå, underhålla och utökas. Däremot anser vi att jobba mot interfaces inte blev aktuellt då vi ansåg att det inte tillförde något som höjer kvaliteten vid en så specifik applikation där det även finns chans till förändringar på input/output vid implementation. (Martin. 2000).

Att skapa en metod för att jämföra substrängar genom “Algoritm M” blev en bidragande orsak för att kunna garantera både system- och informations-kvalitet. Metoden är rekursiv och anropas för att hitta alla möjliga liknande substrängar. Vi fastställt, genom program tester att metoden är felsäker tills dem 5 första tecknen i MRZ-koden är felaktiga. Då det anses att avläsningen är i för dåligt skick för att kunna användas. Vilket är en godkänd acceptansnivå i dialog med uppdragsgivaren. Algoritmen har en komplicitetsklass O(n2) eftersom vid varje rekursivt anrop så delas substrängar upp i två olika nya substrängar och anropas rekursivt igen. Detta ger ett tidskomplex på O(n2) pga. om man dubblar n så ökar tiden med 4, alltså n*n. (P Danziger. 2010)

5.4. Formativ utvärdering genom designprocessen

En formativ utvärdering har också varit av absolut betydelse genom designprocessen. Det har i stor utsträckning varit den formativa utvärderingen som har nyttjats i PDCA-cyklerna, och har således varit en delmängd av den experimentella utvärderingen. Detta avsnitt blir därför inte heller ett avsevärt långt avsnitt, men vissa aspekter bör ändå lyftas.

Inför det första steget i PDCA-cykeln, planering, har det varit viktigt för oss att begrunda och analysera vad vi tidigare implementerat, innan vi fortsätter utveckla nästa del av systemet. I designprocessen vagga fanns det inte mycket att utgå ifrån. Här var det enda stödet vi hade att luta oss på delar av den litteratur vi hittat. Därför var de tidiga implementationerna i systemet otydliga och ibland ledda av slumpen. Däremot, vart efter att klasser och metoder tog skepnad, kunde vi börja undersöka vad som var korrekt eller bristfälligt i implementationerna. Det fanns nu ett underlag att utvärdera och för varje nytt delmål gjorde vi en plan. Där planen var otydlig, och vägen till lösningen var dunkel, fick vi leta i litteratur eller ta hjälp av dokumentation på internet för att hitta nästa steg framåt. Det var måhända ha varit främst under dessa omständigheter som den formativa utvärderingen blev vital. En vacklande implementering är i ett absolut nödvändigt behov av att utvärderas. Det var på så sätt vi kunde justera eller göra återtag på och designa om tidigare implementationer som vi senare hade upptäckt, eller ansåg, vara otillräckliga.

Den formativa utvärderingen skedde alltså löpande genom designprocessen och gav som resultat systemet en högre kvalité. Men högre kvalité menar vi i avseende på yttre kvalitetsegenskaper som korrekthet och träffsäkerhet, se McConnell (2004).

5.5. Pragmatisk utvärdering

Vår applikation skall implementeras när aktuell AI-modell skall tränas via maskininlärning. Vid tiden för detta projekt utvecklas deras modell kontinuerligt och detta uppdrag utförs med uppsåt att implementeras senare i deras system. Redan från planeringsstadiet har tanken varit att det GUI vi skapat inte kommer att användas, utan det är metoderna som löser problemet som är det aktuella. Det grafiska är endast skapat för att kunna demonstrera funktionerna för uppdragsgivaren.

Vår applikation, om den tags i bruk i någon form, kommer att fungera som ett kontrollprogram och skapa underlag till att träna deras ML. Skatteverkets system kommer redan automatiskt att kontrollera checksummor på de id-handlingar som har fotats och skickats in. Men där checksummor inte längre stämmer kommer filen att skickas till en handläggare som manuellt skriver av det dom ser (MRZ-koden). Det är sedan den manuella avskrivningen som kommer att matas in i vårt system, vilket ska ge direkt svar på om AI-modellen läst id-handlingen korrekt och/eller vilka fel som uppkommit.

5.6. Sammanfattande karaktärisering och reflektion

En sammanfattande mening som Cronholm & Goldkuhl (2003), avslutar sin artikel med är att, när det kommer till utvärderingsmetoder, anser dom att det går att blanda egg från olika korgar. Vad författarna menar med denna metafor, och förklarar i artikeln, är att olika situationer kräver olika typer av utvärderingar. Ibland är det även lämpligare för vissa situationer att kombinationerna utvärderingar för att nå så hög effekt som möjligt. Det är i ljuset av författarnas slutsats som vi har gjort en utvärdering av vårt system och den process som denna skapats genom. Vi kommer nu att sammanfattningsvis redogöra för i vilken grad vi har tillämpat ovanstående utvärderingar och vilka styrkor och brister som finns i dessa.

Som vi redan tidigare har nämnt går vissa av de utvärderingar vi gjort mycket hand i hand. Det vill säga att några utvärderingar går in i varandra och kan därför vara svåra att särskilja. Denna företeelse är framför allt märkbar för de demonstrativa, experimentella och formativa utvärderingarna vi gjorde. Här anser vi att den experimentella utvärderingen någonstans är basen i utvärderingen. Basen bygger vi sedan på komplexitet med de demonstrativa och formativa utvädringarna. På så sätt skapar vi en mer heltäckande utvärdering, där vår bild av systemet byggs upp av att flera olika perspektiv tas in, betraktas och tillämpas. Vi kan tydliggöra ytterligare hur vi nyttjade de tre utvärderingarna. Om vi skulle säga att PDCA-cykeln är vår sprints grundflöde. Så börjar första steget, planeringen, efter veckans möte med kunden (Skatteverket). Mötet i sig genererar den demonstrativa utvärderingen där vi analyserar, tillsammans med kunden, hur vår

implementering möter de sex kategorier från DeLone & McLean (1992). Mötena ger även oss de krav som kunden ställer på framtida implementationer eller systemet i helhet. Den demonstrativa utvärderingen blir således ett första underlag till den planering vi gör inför nästa utbyggnad. Den experimentella utvärderingen gav oss, till del, en bild av vilka brister och fel som fanns i systemet just då och producerar därigenom underlag till en formativ utvärdering. Den formativa utvärderingen gjorde att vi fick en uppfattning om vilken grad av kvalité systemet har uppnått. Vi kunde alltså se om systemet uppfyllde kraven på en tillfredställande nivå, eller om det fortfarande fanns betydande felaktigheter eller svagheter som behövde tas itu med. Den formativa utvärderingen kom på så sätt att bli del av nästkommande planering tillsammans med nästa demonstrativa utvärdering. Planeringen kunde därför, med hänsyn till utvärderingarna, ibland komma att ta mer fokus på att förbättra och vidareutveckla en tidigare sprints implementation.

Den tolkande utvärderingen var, kort sagt, ett sätt för oss att kontrollera och verifiera att vi arbetade mot den litteratur som vi i kapitel två la fram som underlag för systemet. Vi kunde genom denna utvärdering gå igenom systemet och undersöka hur väl vi hade tillämpat litteraturen i vår utvecklingsprocess.

Den pragmatiska utvärderingen skriver vi inte så mycket om och det är för att den hade liten, eller ingen betydelse för oss. Hur och om vårt system kommer att byggas ihop med Skatteverkets-system har vi ingen större insyn i. Därför var denna enbart en kortare spekulativ del av vår utvärdering.

Vi vill också nämna en av de forskningslitteraturer som vi presenterade i *Tab.1* men som sen inte visade sig vara relevant för oss att använda, Taiwo Oladipupo Ayodele. *(2010)*. Anledningen att vi presenterade författarens forskning som relevant var för att vi under de tidiga stadierna av designprocessen inte hade en helt klar bild av uppdraget och dess omfattning. Alltså då vi tidigt inte var helt säkra på i vilken utsträckning vi skulle validera, testa och arbeta mot ML gjorde vi en gardering där vi inkluderade författarens forskning. Däremot var litteraturen fortfarande värdefull för får designprocess då den skapar en bättre förståelse för ML-system och hur testning genomförs i området. Litteraturen skapade på så sätt en starkare kunskapsbas som med all förmodan gav oss bättre förutsättningar för att designa ett system av högre kvalité.

Våra utvärderingar som genomförts har varit av Typ 2 (målbaserad utvärdering på IT-systemet som sådant) som Cronholm & Goldkuhl definierar i sin artikel. Vi har valt att utgå från Typ 2 med åtanke av att vi har definierade mål från uppdragsgivaren vad systemet skall kunna uppfylla, men vi vill endast titta på systemet som det är och inte när det är implementerat i drift. Oftast genomförs en Typ 2 utvärdering när man vill skapa en bild om vilka teoretiska bidragsystemet utvunnit. Hade vi genomfört en målfri utvärdering hade vi möjligtvis kunnat få en mer abstrakt utvärdering med större överblick. Men genom att använda uppsatta mål kan vi enklare fokusera på hur systemet fungerar enligt definition, samt kunna enklare beskriva teoretiska bidrag. Se bilaga 8.7 för definition för målbaserad utvärdering av systemet som sådant.

Vi har genom att analyserat och utvärderat vårt arbete skapat en större överblick och kunskapssamling för att kunna diskutera aktuella teoretiska kunskapsutvinningar från projektet. Genom att utvärdera experimentellt, demonstrativt, formativt, tolkande och pragmatiskt ger det flera olika infallsvinklar för vidare diskussioner.

6. Diskussion och abstraktion

I detta avsnitt kommer vi föra en diskussion om vilka kunskapsbidrag vi kan komma med ifrån den genomförda processen. Kortare kommer vi också att diskutera vilken problemklass vårt uppdrag är ett exempel på.

De inledande designbesluten vi behövde ta var på sätt och vis de viktigaste. Eftersom det första vi behövde bygga var det som vi senare skulle vidareutveckla och skräddarsy, var det oerhört viktigt att vi hade rätt utgångspunkt. Vi behövde alltså tidigt skapa en uppfattning och förståelse för hur systemet skulle se ut och fungera när det var färdigt. Vilket i sig kan låta självklart, men det fanns inte så mycket utrymme för oss att hipp som happ implementera lösningar från internet eller våra huvuden. Framför allt är huvudkomponenterna i det nuvarande systemet tillräckligt avancerade för att kräva mer sofistikerade lösningar än de som vi skulle kunna tänkas konstruera inom en rimlig tid. Därför var bland de tidigaste besluten vi fattade att vi skulle experimentera fram lösningar utifrån en väletablerad algoritm (Mocreight´s “algoritm M”), som är en variant av Levenshteins distansalgoritm.

Efter att vi hade enats om den ytterst betydande utgångspunkten kunde vi också tidigt konstatera att lösningarna vi behövde komma fram till inte alltid kunde göras på egen hand. Det var i sin natur avancerade funktioner som vi behövde förstå för att kunna arbeta med och bygga vidare på. Det var helt enkelt lättare att jobba med två hjärnor i stället för en. Vad vi däremot också insåg var att två hjärnor ibland snubblar över varandra och följaktligen påvisades den kända metaforen “ju fler kockar, desto sämre soppa”. Men som spännande nog visade sig inte heller var riktigt sant. Ett flertal gånger kunde vi konstatera att båda utvecklarna, när de jobbat på olika håll för att lösa ett problem, hade bra lösningar men som var för sig inte var fullständiga. Däremot efter att vi beslutat oss för att fortsätta med den ena lösningen och mötes av nästa problem kunde vi mer ofta än sällan se att en kombination av lösningarna var det som skapade fulländade lösningar.

Det vi tar med oss från detta projekt är att inte uppfinna hjulet på nytt. Från början gick vi in med inställningen att bygga metoder för att jämföra strängar själva. Men relativt snabbt blev vi varse om hur stora problematiseringar det finns inom ämnet. Genom efterforskning kunde vi inse att vi definitivt inte var dem första att ställa inför dessa problem och att det fanns väldig mycket nyttigt att utvinna från tidigare forskningsartiklar. När man sedan identifierat just algoritmer som delvis löste våra problem var det inte heller med stor ansträngning man kunde hitta utvecklare som skapat java-baserade metoder för algoritmerna. Sedan var det bara för vår del att anpassa metoderna att integrera med våra variabler och just den informationen vi var ute efter. Hade vi inte gjort efterforskning och problemklassning så hade vi med all förmodan fortfarande suttit med en ofärdig applikation.

Precis som Baskerville, R., Kaul, M., & Storey, V. (2011) beskriver så har vi använt oss av nivå 1 och 2. Först har vi gjort research inom ämnet för att utveckla och bredda vår kunskapsbas (Designing with Research). Detta för att kunna nyttja tidigare designer och reflektioner m.a.o. inte skapa hjulet på nytt när det redan finns bra lösningar. Senare i processen, under utveckling av applikationen, har vi kunnat reflektera över våra designaktiviteter med utgångspunkt från våra egna erfarenheter (Research into Designing) för att kunna skapa en högre nivå av förståelse i designen.

7. Slutsatser

I detta avsnitt tänker vi sammanfatta vårt uppdrag med att redovisa det praktiska resultatet mot vilka projektmål som identifierades vid uppstart från uppdragsgivaren. Vi kommer även ge rekommendationer för fortsatt utveckling av applikationen för att underlätta vidare arbete.

Vidare diskuteras vilka forskningsbidrag som kan utvinnas från detta projekt och hur man skulle kunna vidareutveckla fördjupad forskning inom ämnen och problemformuleringar vid identifierat under projektets gång.

Avslutningsvis diskuterar vi även vilka metoder som använts och hur det fungerat, vad som kunde gått bättre och hur man skulle kunna strukturerar metoderna smidigare vid framtida liknande uppdrag.

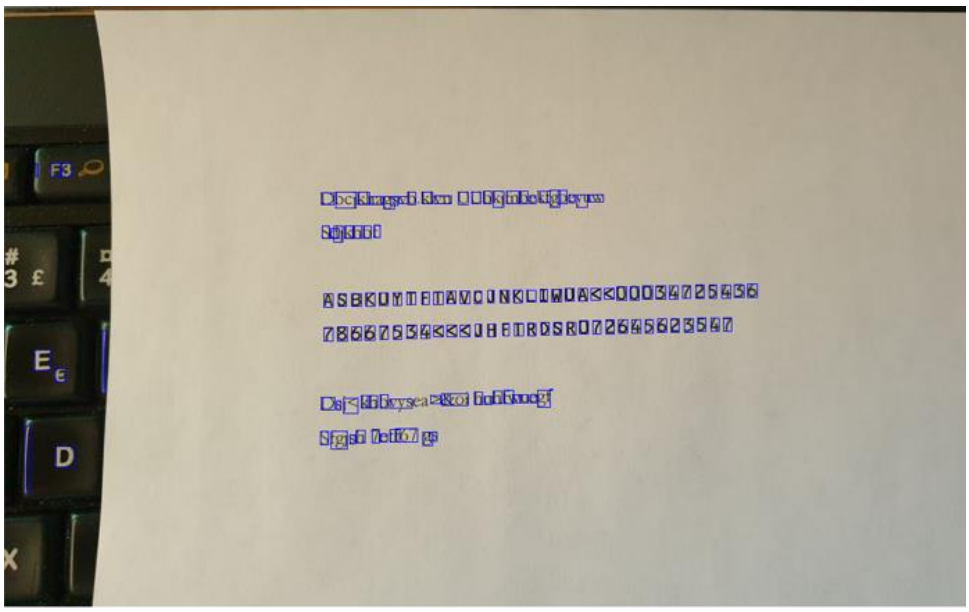
7.1. Slutförande av uppdraget

Vi anser att vid avslut av detta uppdrag kunnat leverera en applikation som fyller de ursprungliga krav som ställts av uppdragsgivaren. Applikationen kan ta in en boxmodell för att skapa tre nya boxmodeller med full MRZ-kod, alla korrekta och alla felaktiga tecken i aktuell MRZ-kod.

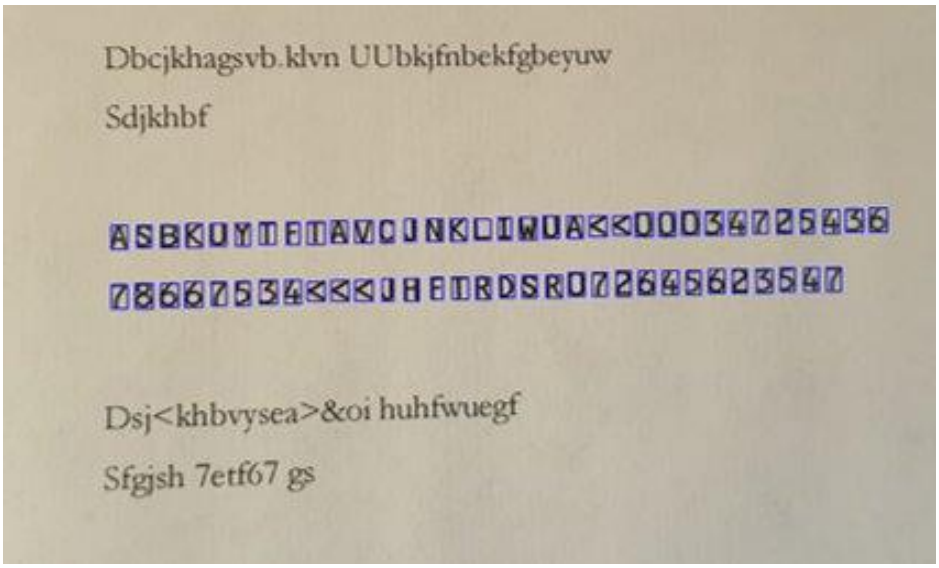
Krav på brister som skulle kontrolleras var:

- Antal tecken stämmer med facit men felaktigt OCR
- Antal tecken stämmer inte med facit, dubbla tecken tolkas som 1 tecken. Det kan även förekomma att fler än 2 tecken tolkas som ett tecken.
- 1 tecken tolkas som flera tecken se ”T” på nedersta raden

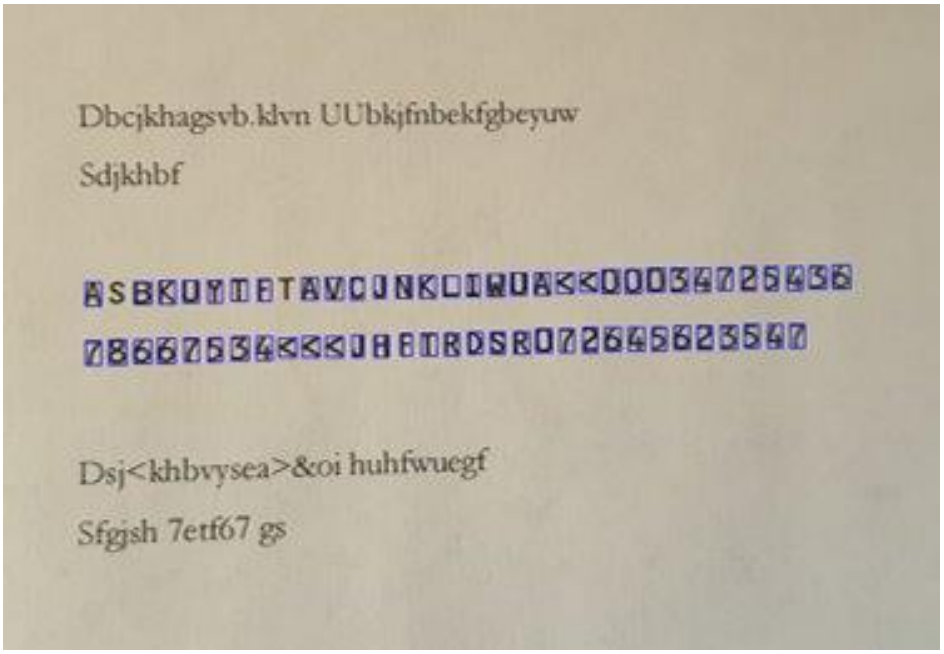
Vi kan med säkerhet intyga att vår applikation fyller de kontroller fullt ut, tills det blir fler än fem felaktiga tecken i rad vid start av MRZ-koden. Då anses bilder för dåligt skannad, men problemet är isolerat i en Try/Catch och användaren blir informerad om gällande fel och anvisas att scanna om bilden. Det är endast vid start av MRZ-kod som det inte får förekomma fler än 5 fel, i MRZ-koden klarar vår applikation med integrerad algoritm M alla fel som kan uppstå.



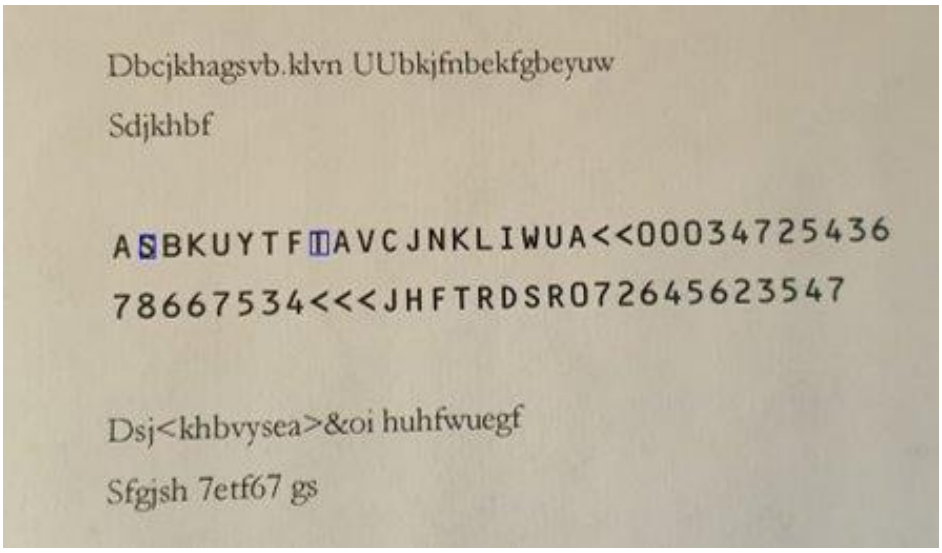
(Fig 8. Identifierade tecken på id-handling av Tess4J)



(Fig 9. Fullständig MRZ-kod)



(Fig 10. Korrekta tecken identifierade)



(Fig 11. Felaktiga tecken identifierade)

I samråd med uppdragsgivaren anses det projektmål som angavs vid uppstart avklarade och nya mål sattes. Dem nya målen bestod utav att integrera Tess4J i applikationen pga. att istället för att behöva läsa av bilder utanför applikationen och sedan läsa in boxmodell, kunna utföra det steget initialt i applikationen. Vi hann med att implementera Tess4J biblioteket, samt att skapa funktioner för att läsa av bilder som skapar boxmodeller, men vi stötte på komplikationer med vilka koordinater som skapades i de nya boxmodellerna. Applikationen funderar med den nya funktionaliteten men det boxar som sparas i de tre nya boxmodeller hamnar inte i rätt koordinater på bilden igen.

7.2. Fortsatt praktiskt arbete

För fortsatt arbete efter överlämnade av applikationen rekommenderar vi uppdragsgivaren att:

- Kontrollera vilka koordinater som blir felaktiga vid utskrift till nya boxmodellen. Två misstänkta fel är:
  - Att y-koordinaterna på tecknet som blir avläst ihopblandade med varandra, vilket resulterar i att toppen på rutan som skall innehålla tecknet blir botten i stället.
  - Att övre och undre raden på MRZ-koden blivit inverterade, m.a.o. att koordinaterna för den övre radens rutor sparats som dem undre tecknen och tvärt om.
- Skapa metod för att kunna ta in den korrekta koden som skall jämföras med som en input direkt i applikationen i stället som för nuvarande då man behöver ange en text-fil som inne håller koden.
- Under samspråk med uppdragsgivaren konstaterades att en felmarginal med fem tecken är en godkänd nivå att ligga på för applikationen, men det går med säkerhet att utveckla denna felmarginal för en mer robust applikation med högre integritet.

Dessa två förslag är även utmärkta i dokumenterad kod för gällande metoder med uppsåt att underlätta framtida arbete.

7.3. Forskningsbidrag

Vi diskuterade tidigare (i avsnitt 2.5 och 6) hur vårt arbete skulle kunna bidra till forskningsvärldens kunskapsbibliotek. Vi nämnde att, och i enlighet med Baskerville et al (2011), vi arbetade mer praktiskt än akademiskt vilket betyder att vi framför allt efterlever *Designing with research.* Det betyder alltså att den utvecklingsprocess vi redogjort för i rapporten främst har tagit

stöd i befintliga forskningskunskaper och inte aktivt har ansträngt sig för att skapa nya. Trots detta är däremot detta en process som i sig kan skapa empiriskt och teoretiskt underlag utifrån den praktiska erfarenhet som man får av att genomföra en design- och utvecklingsprocess.

Det är i ton med föregående stycke som vi tror att, vi i framtiden, kommer ha skapat ett mervärde i det arbete vi gör nu. Då vi även tidigare diskuterade Ågerfalk (2014) och hur det kan finnas ett betydande värde i praktiska och empiriska erfarenheter. Så tror vi alltså att detta stärker vår motivering till en fortsatt bearbetning av den data denna process har genererat. Vi anser att det, till del, redan kan finnas erfarenheter i rapporten som skulle kunna vara till hjälp och praktiskt stöd i liknande arbeten. Däremot tror vi att med fortsatt bearbetning och utökad med mer rigorösa undersökningar att detta arbete kan ligga som en stabil grund inför fortsatt forskning.

7.4. Fortsatt forskning

I enlighet med föregående avsnitts avslutande ord, och som fortsättning på dem, tror vi att vi har identifierat en intressant och viktig del av vår utvecklingsprocess. Vi har diskuterat många intressanta lärdomar och erfarenheter igenom rapporten men vad vi anser som är extra värt att titta närmare på är samspelet mellan utvecklare. Upprepade gånger har vi i fältanteckningar, muntliga konversationer, presentationer och denna rapport påpekat hur viktigt parprogrammeringen har varit för att vi skulle slutföra uppgiften vi fått. Parprogrammeringen var således ett nyckelelement i den framgång vi hade i att tackla olika problemen som dök upp längs vägen. Det fanns givetvis fler metoder som vi tog stöd av för att lösa uppgiften och parprogrammering är långt ifrån en singulär och omnipotent lösning. Däremot kan vi inte heller frånskriva par programmeringens betydelse för designprocessens framgång.

Vi tror därför att en framtida frågeställning i stil med ‘När upphör par programmering att vara effektiv?’ kan vara lämplig för ett forskningsarbete i en C-uppsats. Här skulle målet vara att mer utförligt undersöka när par programmering är som mest tillika minst effektiv under utvecklingsfasen. Till detta kommer denna rapport och tillhörande fältanteckningar utgöra ett empiriskt underlag. Vi kommer även behöva komplettera detta underlag med till exempel enkät-undersökningar och ytterligare litteratursökningar. Därigenom tror vi att vi kan dra rimliga slutsatser och formulera viktiga och nödvändiga kunskaper som kan ha värde för andra.

7.5. Metodreflektion

Att redan innan projektstart upprätta ett gantt-schema har gett en tydlig överblick hur projektet ligger till tidsmässigt och tid vi har kvar för att kunna utveckla applikationen. Det vi kan ta med oss till framtida projekt är att oftare uppdatera tidsplanen, möjligtvis tillsammans med veckomötet då man kan lägga in veckans mål och skapa tidsplanen för vad varje mål bör ta. Kan diskuteras om detta är en nödvändighet i detta mindre projekt, men vid större bör denna metod ha en högre prioritet då utvecklare kan tydligt se vem som jobbar på vad och när varje mål bör bli färdigt.

Som vi reflekterat över i tidigare avsnitt så har par-programmeringen fungerat väl tills ett läge då avancerad problematik uppstått för att lösa problemen, då vi valt att dela upp oss för att enklare kunna genomför framsteg genom “trial and error”-metoder. Sedan har vi presenterat olika förslag på lösningar eller delar av lösningar för varandra, vilket oftast lett till att kunna sammanföra olika metoder från varandras lösningar för en optimal lösning.

Vikten av versionshantering blev väldigt uppenbar och av stor vikt då vi valde att dela upp oss för att genomföra “trial and error”-metoder. Det gjorde det att vi med enkelhet alltid kunde gå tillbaka till tidigare etablerad korrekt kod som beslutats som senaste godkända version när lösningar inte fungerat som tänkt. Att även direkt dokumentera varje metod i koden gjorde att vi enkelt kunde förstå varandras metoder och vilka parametrar som behövdes tas in och returneras för att upprätthålla funktionalitet i applikationen.

Över lag anser vi att arbeta genom en anpassad agil arbetsmetod varit ett mycket givande koncept. Att utöva forskning och programmera applikationen en vecka i tagen för att sedan genomföra demos för uppdragsgivaren varje torsdag har varit lagom för detta mindre projekt. De mål som satts för varje vecka har kunnat genomförts i stora drag och då problem uppstått har det även där varit givande att kunna ha ett möte varje vecka för att presentera förslag på olika lösningsorienterade vägar att fortsätta på. Det har gjort att slutprodukten enklare har kunnat styras från uppdragsgivaren till högre belåtenhet.

Källförteckning

Altexsoft. (2021). *Extreme Prograrmming: Values, Principles, and Practices*.

URL: altexsoft.com/blog/business/extreme-programming-values-principles-and-practices/

Azarian, I. Segue Technologies (2013). *A Review of Software Version Control: Systems, Benefits, and Why it Matters*. URL: seguetech.com/a-review-of-software-version-control-systems-benefits-and-why-it-matters/

Baskerville, R., Kaul, M., & Storey, V. (2011). *Unpacking the duality of design science. In ICIS 2011 Proceedings.*

Cronholm, S., & Goldkuhl, G. (2003). Strategies for information systems evaluation-six generic types. *Electronic Journal of Information Systems Evaluation*, *6*(2), 65–74.

Cuelogic Insights. (2017). *The Levenshtein Algorithm*

URL: https://www.cuelogic.com/blog/the-levenshtein-algorithm

Danziger P. (2010). *Big o notation*

URL: https://www.cs.ryerson.ca/~mth210/Handouts/PD/bigO.pdf

Edward M. Mocreight. (1976). *A Space-Economical Suffix Tree Construction Algorithm*

URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.130.8022&rep=rep1&type=pdf

Goldkuhl, G. (2019). The generation of qualitative data in information systems research: the diversity of empirical research methods. *Communications of the Association for Information Systems*, *44*, 572–599.

Gregor, S., & Hevner, A. R. (2013). Positioning and Presenting Design Science Research for Maximum Impact. *Mis Quarterly*, *37*(2), 337–355.

Harvey Maylors. (2001). *Beyond the Gantt Chart:Project ManagementMoving on*

URL: https://www.sciencedirect.com/science/article/pii/S0263237300000748

Heinz K. Klein and Michael D. Myers. (1999). *A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems Stable*

URL: https://www.jstor.org/stable/249410

Hevner, Alan R. (2007) *A Three Cycle View of Design Science Research, Scandinavian Journal of Information Systems: Vol. 19: Iss. 2, Article 4.*

URL: http://aisel.aisnet.org/sjis/vol19/iss2/4

Houssem Ben Braieka, Foutse Khomha. (2020). *On Testing Machine Learning Programs*

URL: https://arxiv.org/pdf/1812.02257.pdf

Infrrd. (2020). *How AI & Deep Learning Algorithms Deliver OCR Accuracy*

URL:https://www.infrrd.ai/blog/how-ai-and-deep-learning-algorithms-deliver-ocr-accuracy-for-business

Mann, J. Gearset (2021). *The benefits of version control driven development.* (Läst: 2021-04-23)
URL: docs.gearset.com/articles/the-benefits-of-version-control-driven-development

Mark A. Whiting, Jereme Haack, Carrie Varley. (2008). *Creating Realistic, Scenario-Based Synthetic Data for Test and Evaluation of Information Analytics Software* URL: http://beliv.cs.univie.ac.at/papers/beliv2008/whiting_creating_beliv08.pdf

Martin. R. (2000). *Design Principles and Design Patterns*

URL:
https://web.archive.org/web/20150906155800/http://www.objectmentor.com/resources/articles/Principles_and_Patterns.pdf

McConnell. S. (2004). Code Complete Second Edition. sida 463

Nanonets. (2021). How to easily OCR passport and ID cards

URL: https://nanonets.com/blog/ocr-for-passports-and-id-cards/

Projektledning. (2018). *GANTT Schema – Visualisera din projektplanering.*

URL: https://projektledning.se/gantt-schema/

S. H. Kan, V. R. Basili & L. N. Shapiro. IBM Systems Journal, VOL 33, NO 1, (1994). *How information quality leads to localized capabilities and customer service performance*.

Taiwo Oladipupo Ayodele. *(2010). Types of Machine Learning Algorithms*

URL: https://pdfs.semanticscholar.org/c4ae/802491724aee021f31f02327b9671cead3dc.pdf

Tin Kam Ho, Henry S. Baird. (1995). *Evaluation of OCR Accuracy Using Synthetic Data*

URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.41.9491

Venable, J., Pries-Heje, J., & Baskerville, R. (2016). FEDS: a framework for evaluation in design science research. *European Journal of Information Systems*, *25*(1), 77–89.

Vetenskapsrådet. (2017). *God forskningssed*.

Weiner, P. (1965). *Linear pattern matching algomthms.*

URL:
https://www.researchgate.net/publication/229067733_Linear_Pattern_Matching_Algorithm

Ågerfalk, P. J. (2014). Insufficient theoretical contribution: a conclusive rationale for rejection? *European Journal of Information Systems*, *23*(6), 593–599. https://doi.org/10.1057/ejis.2014.35

8. Bilagor

8.1. Exempel Leibenstein Distance Algorithm

H		O	N	D	A	
H	Y	U	N	D	A	I

H	O		N	D	A	
H	Y	U	N	D	A	I

insertion
substitution
deletion

8.2. Målbaserad utvärdering av systemet som sådant

Main perspective	Depends on the character of the goals
What to achieve knowledge about	Has the IT-system fulfilled the desired business goals? The IT-systems' potential positive and negative consequences for the business. What is the presumed contribution of the IT-system?
Data sources	IT-system, goal descriptions, requirement specifications, descriptions of the IT-system
Deductive or inductive	Deductive
Who is participating	Evaluator expert
When to chose this type	When a clearly focused evaluation is wanted, when there are fewer resources at hand, when there are no users available