Agil softwareudvikling under den nye Udbudslov

En erhvervsøkonomisk og juridisk analyse af det agile udviklingsparadigme i et udbudsretligt perspektiv

af DIANA BØJESEN

Mens anvendelsen af en agil softwareudviklingsmetode i stigende grad anvendes og efterspørges i den private sektor, udfordres offentlige ordregivere af udbudsloven, der blandt andet ikke tillader foretagelse af væsentlige ændringer i aftalegrundlaget. Studier tyder på, at der i dag netop er behov for at kunne foretage ændringer som følge af eksponentiel teknologisk udvikling. Dette muliggør den agile udviklingsmetode gennem anvendelsen af en iterativ og inkrementel udviklingstilgang.

Denne afhandling skaber et teoretisk og tværfagligt fundament for anvendelsen af en agil udviklingsmetode i det offentlige. Den udleder indledningsvist en række underliggende forudsætninger for succesfuld anvendelse af denne ændringsdrevne agile tilgang, ud fra et softwareudviklingsmæssigt og et erhvervsøkonomisk perspektiv. Disse forudsætninger danner grundlag for den efterfølgende juridiske analyse, hvilken anskueliggør, i hvilket omfang det er muligt for en offentlig ordregiver at anvende en agil udviklingsmetode, når aftaleindgåelsen er underlagt den nye udbudslov.

Derudover belyser denne afhandling, hvilke relevante kontraktuelle overvejelser, den ordregivende myndighed skal tage i betragtning for at muliggøre en vellykket gennemførelse af et agilt udviklingsprojekt og for ikke at begrænse adgangen til at udnytte potentielle muligheder.

Som det mest essentielle, konkluderer og perspektiverer denne afhandling, hvordan ordregiver gennem indgåelse af den nye, mere fleksible udbudsprocedure, innovationspartnerskab, kan imødekomme behovet for udarbejdelsen af en kontrakt, der muliggør en konstant tilpasning af den udbudte opgave gennem hele projektets livscyklus.

1 ABSTRACT

This paper provides an analysis of agile software development from a legal perspective to determine, at what extent it is possible for a public contracting authority to use an agile software development method when outsourcing a development project that is subject to the new Danish Public Procurement Act[1].

The purpose of this analysis is to provide a theoretical and holistic foundation for applying an agile software approach in public IT procurement. For this reason, this paper initially deduces the underlying assumptions for agile software development, from the perspective of software development and business economics, which the legal analysis is founded on.

Furthermore, the paper elucidates the relevant contractual considerations the contracting authority must take into account, to enable a successful implementation of an agile development method.

The research draws attention to the exponentially evolving technology, which requires a flexible and chance-driven approach to software development. The agile approach contributes to this need by providing people-centric methodology, which consider iterative development, incremental delivery of working software, cooperation and changeability the most valuable assets of the development process.

[1] LOV nr. 1564

For this purpose, Scrum[2] utilizes a Product Backlog in preference to a specification of requirements. This Backlog is under constant alteration throughout the projects lifecycle, as this gives the software development team the ability to provide a product that is kept up-to-date with the evolving need of the contracting authority, comprising the end-users of the software product.

The most important finding of this paper is the necessity for the contract to enable this constant alteration. However, herein lies a challenging task as the Public Procurement Act restricts the ability to effect changes, as it requires compliance with the general procurement principles of i.a. equal treatment and transparency. This entails that it is not possible to alter any fundamental aspects of the original terms of the contracts. It is however possible to apply the new contract award procedure for innovation partnerships, allowing the contracting authority to define the subject matter of the contract on an alternative basis. This provides the possibility of envisaging the change requirements and change procedures in the tender documents in clear, precise and unambiguous clauses, whereby it becomes possible to make these necessary agile adjustments.

Furthermore, it is found that the flexibility of the agile method can be maintained under the Danish Public Procurement Act, resulting in a greater chance for the project to be carried out successfully and an opportunity for it to be completed ahead of schedule.

In addition to addressing the possibilities and consequences of agile adjustments, it is recommended:

- that the contract addresses the consequences of a premature completion, as this may pose a problem, in the case the event changes the economic balance of the contract in favour of the contractor in a manner, not provided for in the original contract
- that a proactive Risk Management strategy is implemented, allowing the contracting authority to exploit opportunities, hence taking advantage of the technological development that would otherwise have posed as a great threat to the project.
- that a representative of the contracting authority is included in the development process on a daily basis, as the innovative solution as a result of the agile development method relies on cooperation and Knowledge Management between the parties.

INDHOLDSFORTEGNELSE

[2] Scrum is an iterative and incremental agile software development framework and a management and control process for product development. [51]

2 INTRODUKTION TIL AFHANDLINGEN

2.1 INDLEDNING

Eksperter vurderer, at den teknologiske udvikling er eksponentielt excellerende og at den teknologi, vi beskæftiger os med i dag, vil udvikle sig til at blive 32 gange så avanceret på bare 5 år – og 64 gange i løbet af 6 år. [1] [2]

Dette tempo påvirker i sær den offentlige myndighed, der står overfor for udlicitering af et softwareudviklingsprojekt, idet en udbudsprocedure kan tage indtil flere år at gennemføre. Dette kan medføre, at den teknologiske udvikling kan nå at forælde de krav, der stilles ved påbegyndelsen af udbuddet inden det endelige produkt står færdigt.

Anskaffelse af enhver form for software tager udgangspunkt brugernes behov, hvilke kan være vanskelige at formulere, endda for brugerne selv, der måske ikke engang er bevidste om deres behov. Om ikke andet, er brugerne ikke klar over hvad de i fremtiden får brug for, hvilket i høj grad påvirkes af den teknologiske udvikling, der konstant skaber nye muligheder og dermed også nye behov.

Hvor den offentlige ordregiver ønsker det bedste produkt til den billigst pris[3], er spørgsmålet derfor, hvordan denne teknologiske udvikling bør anskues.

Ordregiver kan på den ene side vælge at danne sig et overblik over det nuværende behov, udbuddet har til formål at dække, og udbyde en traditionel fuldt ud specificeret opgave til en fast pris, og håbe på, at brugernes behov er de samme, når produktet i sidste ende står færdigt. Dertil at der ikke undervejs, i den potentielt flerårige lange proces, opstår nye relevante behov, der burde dækkes, samt at den aftalte pris i sidste ender ikke viser sig kunstig høj grunden den løbende teknologiske forbedring.

Som alternativ hertil, kan ordregiver anskue den teknologiske udvikling som en mulighed for at kunne tilbyde brugere, det bedst mulige produkt, ved netop at inddrage brugerne i udviklingsprocessen og lade udviklingen i teknologien og deres behov danne grundlaget for kravene til produktet, ved at omfavne og muliggøre ændringer heri.

Udfordringerne, der ligger i at udbyde et så fleksibelt produkt, at kunden endnu ikke har det fulde overblik herover, kræver en ganske anden tilgang til selve udviklingsprocessen, end der traditionelt set har været anvendt[4] - en agil proces. Men hvor private organisationer har fordelen ved den aftalefrihed, der ligger imellem private parter, står en offentlige ordregiver over for en udfordring i denne henseende. Offentlige udbud er notorisk kendte for at være ufleksible grundet Udbudsloven, idet der ikke må fortages "væsentlige ændringer" af en udbudt kontrakt[5], hvorfor udbudsproceduren ligeledes må revurderes for at omfavne sådanne anskaffelser.

Med dette udgangspunkt, vil denne afhandlingen kortlægge hvilke teoretiske forudsætninger den agile udviklingsmetode antager, og analysere hvilke muligheder, den offentlige sektor har for at følge med den private, når det kommer til at udnytte den teknologiske udvikling.

---

[3] Ordregiver skal, jf. LOV nr. 1564 af 15/12/2015 (Udbudsloven), § 162, tildele kontrakten på grundlag af det økonomisk mest fordelagtige tilbud på grundlag af et af følgende tildelingskriterier: Pris, omkostninger eller bedste forhold mellem pris og kvalitet.

[4] Den traditionelle udviklingsmetode, der henvises til, er den såkaldte plan-drevet tilgang, hvilken beskrives i afsnittet "Forskellige udviklingsparadigmer".

[5] Se hertil Udbudsloven samt Sag C-454/06, Pressetext (Pressetext-sagen), præmis 35-37 [92]

2.2 PROBLEMFORMULERING

I hvilket omfang er det muligt og relevant at anvende en agile udviklingsmetode i forbindelse med udlicitering af et softwareudviklingsprojekt, når kontraktforholdet er underlagt Udbudsloven?

- Hvilke teoretiske forudsætninger bør ordregiver i denne forbindelse tage højde for, i en softwareudviklings- og erhvervsøkonomisk diskurs?
- Hvilke kontraktuelle overvejelser bør den ordregivende myndighed, med udgangspunkt i disse forudsætninger, gøre sig for at muliggøre succesfuldt gennemførelse af et agilt udviklingsprojektet?

2.3 TEORI OG METODE

Formålet med denne afhandling er at skabe et overblik over hvilke juridiske udfordringer ordregiver stilles overfor, ved gennemførsel af et offentligt udbud af et softwareudviklingsprojekt efter et agilt udviklingsparadigme, samt at behandle hvorledes potentielle problemstiller bør adresseres kontraktuelt.

Første del af afhandlingen kortlægger ved anvendelse af en deduktiv metode, en række grundlæggende forudsætninger for succesfuld gennemførsel af et agilt udviklingsprojekt, fra en softwareudviklings- og et erhvervsøkonomisk perspektiv.

I det juridiske analyseafsnittet, Del II, hvilket reflekterer den agile udviklingsproces fra ordregivers perspektiv, vil disse forudsætninger efterfølgende analyseres ud fra et udbudsretsligt og kontraktuelt udgangspunkt, for at anskueliggøre hvilke muligheder og begrænsninger udbudsretten stiller for gennemførsel at et agilt udviklingsprojekt. Tillige inddrages de relevante juridiske hensyn, kontrakten skal adressere, for at muliggøre opfyldesen af de ovenfor nævnte forudsætninger.

Denne fremgangsmetode anvendes for at skabe et holistisk syn på den agile udviklingsmetode, idet denne forudsætter, at organisationen opfatter sig selv som værende en social konstruktion af dets projekter. [3] Som denne afhandling vil tydeliggøre, kræver dette, at leverandørens organisation integrerer de kommercielle overvejelser i de projektrelevante beslutninger og omvendt, hvilket tillige influerer behovet for ordregivers inddragelse i udviklingsprojektet.

En analyse fortaget på baggrund af disse betragtninger forudsætter anvendelse af system thinking, hvilket er en analytisk tilgang til en problemstilling, der fokuserer på hvorledes, elementet, der studeres, interagerer med de andre bestanddele af systemet. [4] [5] Frem for at isolere og analysere mindre dele af problemstillingen for sig, fungerer system thinking ved at udvide perspektivet til at betragte et større antal interaktion, der influere et problem. Herved er det muligt at analysere problemstillingen i et holistisk perspektiv og opnå en bredere forståelse herfor. Formålet med anvendelse af denne tilgang er, at hjælpe ordregiver, herunder Contract Manageren, samt de øvrige deltagere i det agile udviklingsteam, med at betragte projektet samt dets deltager som en helhed, frem for blot at anskue dette gennem vedkommendes egen rolle i projektet.

Denne tilgang vil assistere med at forstå den gensidige påvirkning de forskellige individer og roller i miljøet har på hinanden, og hvilken effekt denne påvirkning har på projektets udvikling og det mulighed for succes. Dertil vil denne forståelse fra et social konstruktivistisk perspektiv assistere med at skabe det sociale fællesskab og den gensidige afhængighed, denne succes forudsætter [3].

Med det formål, at skabe et praktisk anvendeligt grundlag for implementering af afhandlingens findings i offentlige myndigheder udbudsprocesser, inddrages udviklings-framework'et Scrum, der er den mest anvendte agile udviklings-metode. [6] Herved skabes der desuden mulighed for efterfølgende videnskabelig afprøvning af afhandlingens konklusion fra et praktisk perspektiv, hvilket denne afhandling dog ikke omfatter.

2.4 KILDER OG KILDEKRITIK

Der blev på et indledende plan, som inspiration til denne afhandling, fortaget et interview af Karen Bjernemose Rahbek, Koordinator for Rådgivningssekretariatet for IT-drift at Digitaliseringsstyrelsen, samt Thomas Schou Eistrup, Chefkonsulent for Digitaliseringsstyrelsen, vedrørende udviklingen af standardkontrakt K03, for IT-projekter baseret på en agil metode. Denne standartkontrakter er bemærkelsesværdigt udarbejdet som et agreed document på baggrund af adspurgte IT-udbydere og offentlige myndigheders erfaringer, og tager, som denne afhandlinger søger at gøre, ikke udgangspunkt i og stilling til videnskabelig litteratur. Dette interview anvendes derfor ikke direkte i afhandlingen, mens Standartkontrakten K03 samt vejledningen hertil udelukkende inddrages som fortolkningsbidrag til Udbudsloven hvor det findes relevant.

Til inspiration for afhandlingens problemstilling, blev der endvidere fortaget et interview af Simon Schulian, afdelingschef for Forretningskontrakter ved ATP, samt Birgitte Mau, Contract Manager ved ATP, vedrørende ATP's anvendelse af en agile udviklingsmetode ved udbud af softwareudviklingsprojekter. ATP anvender på nuværende tidspunkt ikke en agil tilgang i deres udviklingsprojekter. Ikke desto mindre bidrog disse interviews til forståelse for behovet herfor, og skabte desuden inspiration til den metodiske opbygning denne afhandling anvender.

Afhandlingen tager som nævnt udgangspunkt i videnskabelig litteratur, herunder bøger samt videnskabelige artikler, mens den juridiske analysen foretages med afsæt i national og international lovgivning, herunder Udbudsloven, direktiv nr. 2004/18/EF (Udbudsdirektivet) samt relevante forarbejder og retspraksis.

2.5 AFGRÆNSNING

Scopet for denne afhandling er at analysere muligheden i anvendelse en agil udviklingsmetode i forbindelse med offentlige anskaffelser på baggrund af et videnskabelige og teoretisk grundlag. Af denne grund uddybes alternativerne til den agile udviklingsmetode, herunder vandfald, Evo mf., samt hybrider mellem disse, ikke yderligere end der er behov for, for at give et klart billede af den agile udviklingsmetode, uanset at den traditionelle udviklingsmetode eller hybride metoder vil være mere relevant at anvende i visse henseender.

Eftersom der tages udgangspunkt i den nye Udbudslov, der trådte i kraft 1. januar 2016, er det i skrivende stund ikke muligt at inddrage retspraksis for anvendelse af denne samt eksempler på anvendelse af udbudsproceduren *innovationspartnerskaber*, hvilken inddrages i analyseafsnittet blandt løsningsmulighederne de udledte forudsætninger.

Af denne årsag, samt grundet afhandlings scope, afgrænses der fra inddragelse og analyse af praktiske eksempler, såsom udbudte udviklingsprojekter, herunder udbudsbekendtgørelse samt -materiale. En analyse af sådanne eksempler, på baggrund af afhandlingens findings, vil dog uanfægtet bidrage med en større indsigt i disse findings samt en praktisk og mere nuanceret vinkel herpå.

I samme tråd afgrænses denne afhandling fra at inddrage analyser af forskellige kontraktparadigmer samt eksempler på mulige juridiske formuering til anvendelse i kontrakten, idet sådanne ikke vurderes at have betydelig relevans for afhandlingens scope, men derimod at have større værdi i forbindelse med analyse af et praktisk grundlag, som det føromtalt.

Der tages desuden forbehold for, at der i softwareudviklings- samt erhvervsøkonomisk-regi, vil være flere relevante hensyn at inddrage ved gennemførsel af i agilt udviklingsprojekt, end de, der medtages i denne afhandling. De anvendte teorier og betragtninger er udvalgt i det omfang, det vurderes at have relevans for afhandlingens scope.

DEL I

3 AGIL SOFTWAREUDVIKLING

Det er kritisk for et samarbejde, at opnå en fælles forståelse for hvad den udbudte opgave omhandler, herunder hvilke behov, der skal opfyldes, hvad der skal udvikles og hvordan det skal udvikles. Dette vedrøre tillige de forventninger de involverede parter har til hinanden og deres deltagelse i det udbudte projekt samt, for denne afhandlings vedkommende – enighed om, hvad der forstås og forventes ved anvendelsen af begrebet *agil udvikling*.

3.1 HVAD ER AGILITET?

Ordet *agil* stammer fra det latinske *agilis*, afledt af ordet *agere*, hvilket betyder at bevæge, handle, gøre, udrette, administrere og at opnå. [7]

Agilitet referere blandt andet til evnen at være fleksibel og hurtigt at kunne tilpasse sig ændrede betingelser, samt til en persons fysiske kompetencer. Det handler om at kunne bevæge sig hurtigt og let, samt at kunne skifte retning og tage et skarpt sving. [8]

Dertil omhandler *mental agilitet* kognitiv og psykologisk omstillingsparathed, samt at have et hurtig, alert og årvågent sind, at have et ressourcefuldt og tilpasseligt karakter samt at være fleksibel. [9] Desuden vil en mental agil person være intellektuelt præcis, logisk og i stand til at tænke hurtigt og kreativ under pres, og klar til at handle og træffe konklusioner hurtigt.

Begrebet agilitet avendes, som denne afhandling vil vise, desuden inden for softwareudvikling og referer til en udviklingsmetodologi [10], der er centreret omkring værdien i at omfavne løbende ændringsmuligheder og af anerkendelse af mennesket som den primære drivkræft for et projekts succes. [11]

I denne sammenhæng definerer Preiss [12] agilitet som:

> *"A comprehensive response to the business challenges of profiting from the rapidly changing, continually fragmenting global markets for high quality, high performance, customer configured goods and services. Thus agility is dynamic, content specific, aggressively change embracing and growth oriented. Agility is a comprehensive response to new competitive forces that have undermined the dominance of the mass-production system"*

Begrebet agil anvendes i denne afhandling desuden i et erhvervsøkonomisk perspektiv, til at beskrive både projekthåndteringen i softwareudvikling, en organisations omstillingsparathed og forandringsledelse, samt evnen til at agere prompte på uforudsete udfordringer i forsøget på at tilpasse sig den teknologiske udvikling, der påvirker markedsbehovet.

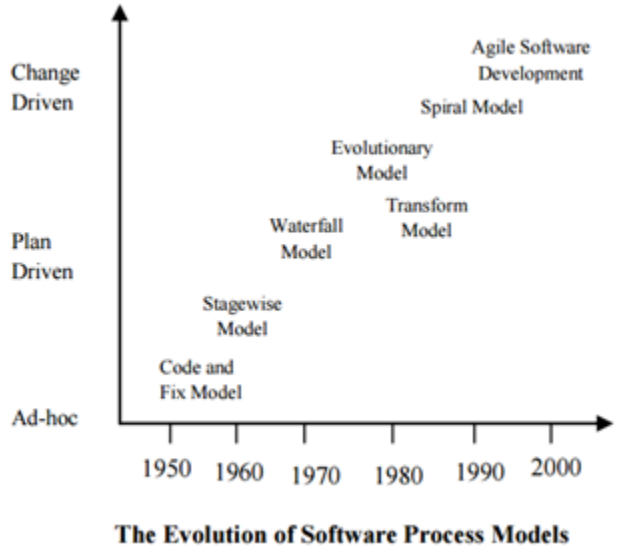
Ens for disse definitioner er et fokus på fleksibilitet, ændringsmuligheder og omstillingsparathed, hvilket i en verden præget af teknologisk udvikling, bemærkelsesværdigt kræver en stort grad af menneskelige ressourcer.

3.2 HISTORISKE UDVIKLING AF SOFTWARE UDVIKLINGSMETODOLOGIER

Mens den teknologiske udvikling er eksponentielt excellerende, halveres software-effektiviten, ifølge *May's Law*, hver 18 måneder [13, p. 67], hvilket ligger et stort pres på udviklingsmetoden og selve organisationen. Softwareudvikling er et felt, kendetegnet ved opgaver og krav, præget af en høj grad af variabilitet [10], samt

en brede vifte af forskellige metodologier og værktøjer, indeholdende både fordele og ulemper, alt afhængt af den enkelte udviklingsopgave og organisationsmiljøet.

Debatten om hvordan software udvikling bør organiseres for at levere hurtigere, bedre og billigere løsninger har været diskuteret i softwarekredse i årtier, og evolutionen ses opsummeret i figur 1, af Singh & Chana [14].



Figur 1, af Singh & Chana [11]

Tilgangen til softwareudvikling har som figur 1 illustrerer, udviklet sig fra at have et *plan-drevet* fokus til et mere *ændrings-drevet fokus*. Som eksempel på førstenævnte, kan den såkaldte vandfalds-model nævnes, hvilken er blandt de traditionelle udviklingsmodeller, med en *BRUF-tilgang* (Big Requirements Up Front), hvilket involverer, at samtlige krav til produktet defineres før projektets opstart. Det ændrings-drevne fokus vedrøre, som illustreret den agile metode.

Den agile udviklingsmodel har sine rødder tilbage i begrebet *lean thinking*, hvilket udsprang af *The Toyota Way*, der er et system, designet som værktøj til personer, der ønsker kontinuerligt at forbedre deres arbejde. [15, p. 36] Tankegang bag lean thinking underbygges af to essentielle grundsten - *kontinuerlig forbedring* og *respekt for individet*, hvilket tydeligt præger nutidens opfattelses af agilitet. Vigtigt er det dog at bemærke, at lean produktfremstilling, i modsætningen til agil udvikling, bifalder gentagelser og misbilliger variabilitet. Agilitet opererer i modsætning hertil med produktudvikling i et dynamisk udviklingsmiljø, hvor innovation og kreativitet er grundelementer for effektiv produktudvikling, hvilket muliggøres af netop variabilitet og hæmmes ved fokus på gentagelser. [16]

Iterative and Incremental Development (IID) har været anvendt siden 1950'erne [17] og er til stadighed en fundamental tankegang bage agile softwareudvikling. Denne tankegang blev dog ikke implementeret i den gennerelle plan-drevne tilgang, hvilke ikke formåede at bidrage med nok fleksibilitet til dynamisk at kunne tilpasses udviklingsprocessen og imødekomme omskiftelige krav. [18]

Det var først i slutningen af det 20. århundrede, begrebet *RAD* (Rapid Application Development) blev introduceret som en modreaktion herpå. Begrebet blev i starten af 1990'erne dog anvendt hensynsløst, uanset at selve udviklingsmetoden ikke ændrede sig fra et plan-drevet fokus, og uden at hverken organisationens ledelseskultur eller –adfærd blev tilpasset, hvorfor RAD som resultat blev associeret med fiasko. Den reelle hensigt bag RAD blev først indlejret i den agile tankegang gennem Barry Boehm's Spiral-model samt i Tom

Gilb's evolutionær model (Evo), [18] hvilken ifølge forfatterne selv, "strukturelt umuliggør projekt fiaskoer" [19]. Spiral-modellen og Evo indeholder begge karakteristika, der minder om den agile tilgang, men lider begge under den begrænsning, at processerne ikke er i stand til at håndtere ændringen af krav i sene udviklingsfaser, hvilket netop er blandt årsagerne til den agile metodes succes.

I 1999 blev den agile *eXtreme Programming-metode* (XP) udviklet, hvilken hovedsageligt fokuserer på de værdier og fremgangsmetoder, der i dag associeres med de teknisk aspekter af softwareprogrammering. Adskillige af disse aspekter anvendes i dag til generisk produktudvikling. [16]

Der opstod dog først et overordnet generisk navn for disse agile metoder, da der med *Det agile Manifes*t blev skabt en kollektiv og ensartet forståelse for, hvordan konceptet *agilitet* skulle fortolkes. Siden da er der blevet udviklet yderligere agile modeller, herunder Lean Software Development samt Kanban mf.

DET AGILE MANIFEST

Efter en årrække præget af fiasko, hvor IT-projekter og softwareprojekter i særdeleshed, havde fået ry for altid at blive forsinket og for at overskride budgettet, samledes 17 softwareudviklings-eksperter som en græsrodsbevægelse. Disse eksperter dannede The Agile Alliance, for at diskutere og udarbejde et alternativ til de plan-drevne, heavyweight udviklingsmetoder, der var skyld i denne negative opfattelse af branchen. [20] Som resultat heraf udsprang "Det agile Manifest".

Dette var første gang, en fælles forståelse for, hvordan agil softwareudvikling blev formuleret og dokumenteret, hvilket tog form af en formel manifestation. Det Agile Manifest skaber grundlaget for en iterativ og inkrementel tilgang til softwareudvikling med mennesket i centrum, hvilket opnås mest optimalt ved at fokusere på værdierne: [20]

- Individer og interaktion frem for processer og værktøjer
- Working software frem for omfattende dokumentation
- Kundesamarbejde frem for kontraktforhandling
- Forandringstilpasning frem for fastholdelse af en plan

I disse kerneværdier ligger blandt andet opfattelsen, at det ikke er hverken værktøjer eller processer i sig selv, der udvikler godt software, men ene og alene mennesker. [10] Dette ses blandt andet ved, at selve software udviklingsprocessen påvirkes, hvis blot det unikke individ, der fungere som bidragyder i et team, fjernes.

Dertil opfattes dokumentation som værende værdiløst uden et fungerende stykke software, også kaldt *working software*. Det Agile Manifest argumenterer for, at indtil et stykke kodet software fungerer som forventet, er der intet empiriske bevis for fremskridt. [10] Dokumenter og planer kan assistere udviklingen, men det er softwaren i sig selv og ikke dokumentation der er beviset dets værdi og succes.

I tråd med denne værdi, omhandler den tredje ligeledes dokumentation, her i forbindelse med kontraktforhandling, idet der ifølge Det Agile Manifest bør fokuseres på samarbejde frem for forhandlinger. Som nedenstående analyse i Del II, af hvorledes et agilt udviklingsprojekt skal håndteres juridisk, vil illustrere, skaber denne værdisætning visse udfordringer, især i forhold til Udbudsloven, hvilken kræver ligebehandling og gennemsigtighed i kontrakten, jf. Udbudsloven § 2.

Det skal dog bemærkes at disse værdier ikke påstår, at der ikke må anvendes nogle former for processer, værktøjer, dokumentation, kontraktforhandlinger og planer, men at disse bør være sekundære i forhold til individet, det udviklede stykke software, samarbejdet samt evnen og viljen til at kassere en plan for at tilpasse sig uforudsete ændringer.

Sidste centrale værdi, forandringstilpasning frem for fastholdelse af en plan, er bemærkelsesværdigt en reaktiv strategi, hvilket kan skabe udfordringer, såfremt den agile tilgang ikke er en integreret del af organisationen, og hvis ikke parterne er fuldt ud enig i denne strategi.[6] [10]

Det Agile Manifest fremstiller foruden disse værdisætninger, 12 principper for anvendelsen af den agile udviklingsmetode, der i dag anerkendes som værende af grundlæggende betydning for den agile udviklingsmetode: [21]

1) Højeste prioritet er at tilfredsstille kunden gennem tidlig og kontinuerlig levering working software.
2) Skiftende krav, selv sent i udvikling bør imødekommes. De agile processer skal sikre, at ændringerne er til kundens konkurrencefordel.
3) Jo hyppigere levering af working software desto bedre.
4) Kunderepræsentant og udviklere skal arbejde sammen dagligt gennem hele projektet.
5) Projekter skal opbygges omkring motiverede personer, og der skal skabes det rette miljø og den nødvendige støtte. Hav tillid til at teamet kan klare opgaven.
6) Den mest effektive kommunikationsform ifm. softwareudvikling er ansigt-til-ansigt samtale.
7) Working software er det primære værktøj til vurdering af fremdrift.
8) Agile processer fremme bæredygtig udvikling. Systemejere, udviklere og brugere bør til enhver tid kunne opretholde et konstant tempo.
9) Kontinuerlig fokus på teknisk ekspertise og godt design fremmer agilitet.
10) Enkelhed - kunsten at maksimere mængden af ikke-udført-arbejder - er afgørende.
11) De bedste arkitekturer, krav og designs udvikles af selvorganiserende teams.
12) Med jævne mellemrum bør teamet revurdere sin egen effektivitet og justere sin adfærd i overensstemmelse hermed.

3.3 FORSKELLIGE UDVIKLINGSPARADIGMER

Inden for softwareudvikling er der utallige udviklingsmetodologier, også kaldt Systems Development Life Cycle (SDLC), hvilke anvendes til at beskrive den udvalgte udviklingsmetode, herunder de forskellige opgaver og aktiviteter, der skal gennemføres i løbet af processen. Begrebet SDLC kan i forbindelse med plandrevne metodologier anvendes til at beskrive de tydeligt opdelte stadier, et projekt skal gennemgå i udviklingsforløbet, deriblandt planlægning, udvikling, testning og implementering af et softwaresystem. Dette står i kontrast til de agile udviklingsmetoder, der alene definerer en proces med iteration, hvor design, konstruktion og implementering af forskellige dele af systemet kan forekomme samtidig. [10]

Som nævnt ovenfor har tilgangen til softwareudvikling gennemgået en radikal udvikling gennem det seneste århundrede, og praktikere har udviklet adskille udviklingsmetoder til håndtering af det komplekse stykke arbejde, der uanfægtet høre til softwareudvikling.

Den plan-drevne udviklingstilgang refereres også til som the *heavyweight methods*, hvilken omfatter adskille udviklingsmetoder, hvoraf de mest anvendte er vandfalds-modellen, der siges at være den ældste og den oprindelig SDLC, the Unified Process (UP) samt Spiral-modellen. I henhold til den agile tilgang, der i branchen omtales som *lightweight methods*, findes der ligeledes talrige metoder, herunder: Extreme Programming, Scrum, Dynamic System Development Method, Feature Driven Development samt Adaptive Software Development.

[6] Dette perspektiv behandles i afsnittet ”Den projektorienteret organisation”

3.3.1 En forudsigelig versus en fleksibel tilgang

Forskelligheden i disse udviklingsmetoder vanskeliggør en direkte sammenligning. Ikke desto mindre er det indledende muligt, at karakterisere nogle væsentlige faktorer, hvorved de forskellige tilgange til softwareudvikling adskiller sig. Disse vedrøre: [10]

- Anvendelseskarakteristika - primære projektmål, projektets størrelse og anvendelse miljø
- Management-karakteristika - kunderelationer, planlægning og kontrol samt projektkommunikation
- Tekniske specifikationer - tilgange til kravdefinition, udvikling og test
- Personalekarakteristika - kundekarakteristika, udviklernes kompetencer og organisationskulturen

3.3.1.1 Den plan-drevne tilgang

De plan-drevne metoder bygger på en rationaliseret tilgang til udvikling, baseret på ingeniørvidenskab, hvor omfattende planlægning, kodificerede processer, og stringent gentagelse inkorporeres for at gøre udvikling til en effektiv og forudsigelig aktivitet. [22] Som en fundamental forskel fra den agile tilgang, antages problemer i den plan-drevne tilgang, at være fuldt ud specificerbare, og tilhængere af denne hævder, at der findes optimale og forudsigelige løsninger for hvert problem. [23]

Disse traditionelle metoder bygger på en lineær fremgangmetode fra projektundfangelse til vedligeholdelse, og er karakteriseret ved at anvende en på-forhånd fastsat kravspecifikation (BRUF) og en række sekventielle faser. Dette er en proces-orienteret tilgang til softwareudvikling, hvor fokus er på at eliminere variationer, ved at identificere fejl, mens der foretages løbende målinger og raffinering af processer. [24] Målet er forudsigelighed, stabilitet og høj sikkerhed.

Organisationer, der har succes med at anvende denne metodologi, er kendetegnet ved at have en mekanisk karakter, og at favorisere en bureaukratisk opbygning, hvor der primært fokuseres på realisering af optimerede, standardiserede og repeterede processer, gennem en kommando- og kontrolstruktur. [10] Dog nedbrydes denne forudsigelighed og stabilitet, når organisationen stilles overfor nye og hidtil usete udfordringer og store uforudsete ændringer, hvilket resultere i, at projektet pådrages betydelige udgifter. [10]

Cho [25, p. 341] beskriver den plan-drevne tilgang som *ligetil, metodisk og struktureret af natur, med en evne til at skabe forudsigelighed, stabilitet og stor sikkerhed*, men kritisere tilgangen for at være ineffektiv til at tilpasse sig, når den udsættes for hurtigt skiftende forretningsmæssige krav og muligheder, samt at have en tendens til at overskride budgettet og at skabe forsinkelser. Desuden argumenterer Cho for, at det generelt er vanskeligt at skabe en komplet og udspecificeret kravsspecifikation før projektets begyndelse.

Dette understøttes af Boehm:

> *"Plan-driven methods work best when developers can determine the requirements in advance . . . and when the requirements remain relatively stable, with change rates on the order of one percent per month."* -- Barry Boehm [23, p. 4]

Den traditionelle tilgang tager udgangspunkt i en række sekventielle trin, herunder kravspecificering, udvikling af en "løsning", testning samt slutteligt implementering, også kaldt *the big-bang*. [26] Disse metoder har næppe været kendt for deres succesrate, og er gennem tiden blevet mindre og mindre populære, på trods af deres fokus er på forudsigelighed og effektivitet, idet de ligeledes kritiseres for at være bureaukratisk gennem hele projektets livscyklus, og for at sænke selve udviklingsprocessen.

Gilb og Johannesen [27], udviklerne bag Evo, kritisere desuden vandfalds-modellen[7] for at:

- Udskyde risikobegrænsning til sene stadier

[7] Udtrykket "vandfald" anvendes i praksis desuden som et generisk begreb for alle sekventielle software udviklingsmetode [26]

- Udskyde dokumentation-baseret verifikation til sene stadier
- Fastsætte ufleksible krav for tidligt i cyklussen, idet ændring af kravene er opfattes negativt
- Operationelle problemer opdages for sent i processen (leveringskontrol)
- Indeholde langvarige modifikationscyklusser og meget omarbejde
- Næsten udelukkende at fokusere på funktionalitet - ikke på kvalitetsegenskab

Vandfalds-modellen repræsenterer en ukompliceret opfattelse af et projekts livscyklus, hvilket dog ikke er fælles for alle plan-baserede tilgange. Dette er ligeledes heller ikke passende for alle former for softwareudviklingsprojekter, og modellen virker af samme årsag bedst i tilfælde, hvor kravene til softwaren er letforståelige, som fx et styresystem, eller ved anvendelse af *Contract-based Software Development*, hvor alt fra kravsspecifikationer og design kan forudbestemmes og danne grundlaget for udviklingen. [28]

3.3.1.2 Den agile tilgang

Organisationer har i dag i stigende grad brug for at kunne agere effektivt og tilpasse strukturer, strategier og politikker til nye miljø, for at kunne levere informationssystemer og teknologier, der konstant udvikler sig, for at imødekomme skiftende krav og behov fra forbrugerne. [29] En grundlæggende forskel mellem den plan-drevne og den agile tilgang er, at sidstnævnte, som tidligere nævnt, anvender en iterativ udviklingsproces, hvilket gør, at udviklingsteamet kan drage nytte af de erfaringer de har gjort sig, ved at observere og teste allerede fungerende versioner af softwaren. Alle aktiviteter i et agilt projekt kører i iterationer, også kaldt *time-boxes*, for at holde fokus på og at målerette ressourceforbruget, og derved at skabe forudsigelighed. [22] Progressionen betragtes og måles efter hver iteration, hvorved fejl løbende rettes. Den agile tilgang søger at fremme adaptiv planlægning, evolutionær udvikling, tidlig levering, løbende forbedringer, og muliggør hurtig og fleksibel respons, ved at benytte principperne for kontinuerlig designforbedringer og testning baseret på hurtig feedback. [22]

I modsætningen til den plan-drevne tilgang, fokuserer de agile metodologier på udfordringen ved en uforudsigelig verden, og på at drage nytte af hver enkelt individ og teamets unikke styrker. Herved sættes der lid til personer og deres kreativitet frem for processer. [30]

I agil softwareudvikling spiller kunden ligeledes en vigtig rolle, og det anbefales der inddrages en kunderepræsentant i samarbejder med udviklingsteamet, for blandt andet at sikre, at misforståelser i forhold til kundens krav kan adresseres øjeblikkelig eller helt undgås. Desuden muliggør dette, at uopsættelige problemstillinger kan reduceres til et minimum.[8] [31]

Ifølge the 2011 CHAOS Manifesto [32], er der en tre gange så høj succesrate (42% versus 14%) for softwareudviklingsprojekter gennem en agile tilgang frem for plan-baserede tilgang samt langt færre overskridelser i forhold til tid og budget. The Standish Group udtrykker endda:

> *"The agile process is the universal remedy for software development project failure."* [32, p. 25]

I en kvantitativ analyse fortaget af Analysis.Net Research [33] gav 73 % af de adspurgte udtryk for, at den agile tilgang sikrer en hurtigere gennemførsel af projektet, og 92%, for at en agile udviklingsmetode forbedrede evnen til at håndtere ændringer i prioriteter gennem projektets livscyklus. Derudover fremhæver rapporten, at nedbrydelse af projektet i mindre dele resulterede i en større succesrate.

3.3.1.2.1 Begrænsninger ved en agil tilgang

Disse forskelligartede metodologier til softwareudvikling henvender sig, som ovenstående afsnit redegør for, til forskellige projekttyper, hvorfor den agile metode i nogle situationer er uhensigtsmæssig at anvende. Manglende forståelse for de ofte uformulerede antagelser, der ligger til grunde for den agile tilgang kan føre

[8] Dette perspektiv vedrøre Risk Management, hvilket behandles i afsnittet "Risikohåndtering".

til et omkostningsfuldt og ineffektivt udviklingsforløb. Det er i denne forbindelse ligeledes væsentligt at identificerer hvilke begrænsninger, den agile metode indeholder.

Med udgangspunkt i det føromtalte Agile Manifest og dets 12 principper [21], fremsætter Turk et al. [34] 14 antagelser for anvendelsen af agile processer, og perspektiverer deres betydning i forhold til hvilke begrænsninger, disse resultere i.[9] Forholdet mellem disse antagelser og begrænsninger fremgår af figur 2.

Assumptions	Agile Process Limitations					
	Personnel Limitations			Product Limitations		
	Limited support for distributed development environments	Limited support for subcontracting	Limited support for development involving large teams	Limited support for building reusable artifacts	Limited support for developing safety-critical software	Limited support for developing large, complex software
Customer Interaction Assumption	X	X	X			
Team Communication Assumption	X	X	X			
Face-to-Face Assumption	X	X	X			
Changing Requirements Assumption		X				
Documentation Assumption	X	X	X	X	X	X
Quality Assurance Assumption				X	X	X
Iteration Assumption						X
Application-Specific Development Assumption				X		
Continuous Redesign Assumption				X	X	X

Figure 4: Limitations of Agile Processes and Related Assumptions

Figur 2, af Turk et al. [34]

De agile antagelser, hvilke fremgår af ovenstående figur, begrænser, ifølge Turk et al. [34], organisationen i at anvende *distribuerede udviklingsmiljøer*, hvilket vil siger miljøer, hvor udviklerene i et team og/eller kunden ikke er placeret på geografisk samme sted. Dette kan skabe kommunikationsudfordringer og kræve særligt understøttende værktøjer, da det gøre folk ude af stand til at interagere på samme tid og sted. [34] Begrænsningen opstår hvor der i udviklingsprocessen ikke er taget højde for antagelsen om co-location, hvilken interaktionsmetode, agile udviklere anser som den mest effektive. [21]

Antagelserne bag den agile udviklingsmetoder skaber ifølge Turk et al. [34] *begrænsede støtte til underleverandører*, hvilket skaber problemer i forhold til outsourcing af softwareudvikling, idet sådanne ofte baseres på udspecificerede kravsspecifikationer, der detaljeret beskriver, hvad der kræves af underleverandøren.

[9] Disse antagelser uddybes ikke særskilt, idet essensen heraf i høj grad fremgår af Agile Manifests principper [21], og deres betydning desuden tydeliggøres gennem fremstilling af de medfølgende begrænsninger.

Uanset at underleverandøren ønsker at følge en iterativ og inkremental udviklingsproces, kræver kontraktforholdet muligvis, at leverancen er prædiktiv, herunder at antallet af iterationer og projektleverancer, forbundet med hver iteration eksempelvis skal angives på forhånd. [34] Hertil kan et underleverandørforhold desuden resultere i at muligheden for geografisk centralisering af projektets medlemmer begrænses yderligere.

Hvor den agile tilgang generelt ønsker at reducere mængden af dokumentation til et absolut minimum, forøges behovet for dokumentation ved anvendelse af underleverandører. Såfremt underleverandørkontrakten indeholder en på-forhånd fastsat kravsspecifikation, vil enhver ændring heri kunne forventes at påvirke prisen, da der er fundamentale forskelle på traditionelle underleverandørkontrakter, og "agile udviklingskontrakter", idet sidstnævnte ligefrem forudsætter ændringer. [34] Der kan således desuden opstå begrænsninger vedrørene antagelsen om skiftende krav, hvilken argumentere for, at krav altid vil udvikle sig over tid på grund af før omtalte ændringer i eksempelvis teknologi og kundebehov.

De agile antagelser begrænser tillige *støtten til udvikling der omfatter store teams*, idet disse antager, at software mest effektivt udvikles og håndteres i mindre selvorganiserede teams. [34] Sådanne teams menes ikke at være udfordret af de samme koordinerings-, styring- og kommunikationsmekanismer som større teams med mange sub-teams af specialister, som potentielt er geografisk fordelt ud over et uhensigtsmæssigt stort område. Store udviklingsteams, der håndtere udvikling af store softwaresystemer kan med fordel anvende mere traditionelle metoder til håndtering af *management-in-the-large*[10]. [34]

Udviklingen af *store og komplekse softwaresystemer* differentierer sig fra udviklingen af enkelte programmer, og involverer det, der kaldes *programming-in-the-large*[11]. Sådanne systemer består af uendeligt mange koder og små programmer, også kaldt moduler, hvilke kan udvikles af flere forskellige teams. Udviklingen af komplekse softwaresystemer indebærer indviklede sammenhænge mellem forskellige dele af systemet for at sikre dataintegritet. Herved opstår der en begrænsning for anvendelse af en inkrementel tilgang, grundet den tætte kobling og integration mellem modulerne i sådanne komplekse softwaresystemer, der skal sikre at alle dele af systemet interagerer pålideligt og efter hensigten. [34] [25] Det kan kræve en stor mængde menneskelige ressourcer, at løse problemer i forbindelse med udvikling af store softwaresystemer, hvilket igen kræver flere samspil og kommunikationslinjer mellem teamets/teamenes medlemmer, en højere grad af ledelsesmæssig kontrol, samt mere formaliserede processer end den agile metode bidrager med. I sådanne tilfælde kan det være hensigtsmæssigt at strukturere det overordnede projekt i henhold til en plan-drevet tilgang, og implementere en hybrid metode, hvilket beskrives i afsnittet "Hybride udviklingsmetoder". Denne kan som eksempel opstilles nogle ydre rammer for, at sikre de moduler, enkelte sub-teams udvikler efter en agile metode, er kompatible i sidste ende.

Den agile tilgang forudsætter, ifølge Turk et al. [34], *applikations-specifikt udvikling*, hvorved fokus udelukkende placeres på det enkelte behov, frem for udvikling af et mere generelt system, der lettere kan tilpasses fremtidige behov, idet sådanne har en tendens til at forøge kompleksiteten. Heraf opstår problemstillingen, at skulle skelne mellem et applikations-specifikt udviklingsmiljø og et miljø for udviklingen af *genanvendelige artefakter*. Sidstnævnte omhandler et fokus på genanvendelse af et segment af en kildekode, i sin helhed eller dele heraf, for at kunne tilføje nye funktioner med kun få eller ingen ændringer. Ved anvendelsen af genan-

[10] Management-in-the-large kræver blandt andet vedligeholdelse af flere kommunikationskanaler og en anden grad af koordination og kontrol. For yderligere information se da eksempelvis "Business Process Management in the Large" af Houy [104]

[11] Programming-in-the-large er et begreb, der dækker over en række udviklingsprocesser dedikeret til håndtering af komplekse udviklingsprojekter. Se eksempelvis "Programming-in-the-Large Versus Programming-in-the-Small" af DeRemer, et al. [102]

vendelige artefakter inddrages et overordnet og langsigtet perspektiv i udvikling, frem for blot et fokus på udviklingen af den enkelte applikation. [12] [35]

Der er tre måder, hvorved genanvendelighed kan inkorporeres i agile softwareudvikling: "Component Based Development", "Refactoring to Design Patterns" samt "Reusable Architectures". [14] Antagelsen om *kontinuerlig re-design* vedrøre både det overordnede projekt-scope og desuden det, der kaldes refactoring[13], hvilket opstår når en ændring af den interne struktur af et system re-designes og stadig bevarer den strukturelle og konceptuelle integritet, og dermed systemets eksterne funktionalitet. [36] Refactoring er en metode hvorpå det er muligt at forbedre eksisterende softwareartefakter vedrørende både design og forståelighed uden at ændre softwarens eksterne adfærd. [37] [38]. Refactoring er især effektiv, når store ændringer kan opdeles i mindre skridt, men Turk et el. [34] argumenterer dog for, at der kan opstå begrænsninger med hensyn til komplekse udviklingsprocesser, hvis der er behov for at ændre kritiske arkitekturelle aspekter, som spiller en central rolle for grundlæggende system. Mohammad et al. [31] kritisere i samme tråd den agile metoder for at mangle langtidsplanlægning med hensyn til muligheden for refactoring. Dette har betydning for projektets slutprodukt, hvor der kan opstå udfordringer, når forskellige dele af softwaren samles endeligt i sidste ende.

Såfremt udviklingen af moduler fordeles mellem flere sub-teams, er det altafgørende, at de krævede interaktioner omhyggeligt planlægges og dokumenteres. Systemudvikling vil være markant vanskeligere at gennemføre, hvis en given ændring har indflydelse på tværs af modul-grænser, idet samtlige påvirkede moduler således vil kræve re-udvikling. [39] Af denne grund er det væsentlig at være opmærksom på, i hvilke dele af programmet, der vil ske hyppige udvidelser og omstruktureringer. Udvikling af store softwaresystemer, skal derfor udformes i moduler, med lav kobling og høj kohæsion[14], hvorved disse er lette at udskifte i tilfælde af ændringer andetsteds i systemet. [39] [40]

KRITIK AF DEN AGILE TILGANG

Agile udvikling kaldes blandt kritikere "gammel vin på nye flasker", idet lignende praksis har været anvendt i softwareudvikling siden 1960'erne, [41] og er under tiden desuden blevet forvekslet med en ad hoc-tilgang, fordi designet udvikles løbende i modsætning til "up front". Dog bygger den agile tilgang i virkeligheden på et mantra af kvalitet i design og ærligheden af "working codes" samt effektiviteten af samarbejde mellem mennesker. [24]

Derudover er den agile udviklingsmetoder blevet kritiseret af flere praktikere og akademikere med fokus på følgende aspekter:

Agile udviklingsmetoder kritiseres for at mangler videnskabelig evidens, samt for ikke i optimal grad, at fokusere at på arkitekturen, hvilket fører til uhensigtsmæssige design-beslutninger. [42] [43]

Vijayasarathy [44] argumenterer for, at den agile tilgang indeholde problemer, når det kommer til praksis, blandt andet vedrørende den væsentligt reducering i af mængden af dokumentation, og det faktum, at selve koden skal fungere som dokumentation.

---

[12] Genanvendelige moduler og klasser kan forøge både produktiviteten, påledeligheden og vedligeholdelsen af et softwareprodukt, og reducerer implementeringstid, da forudgående testning af artefakten ofte allerede har elemineret programfejl. Sådanne fejl kaldes *bugs*, hvilke resulterer i, at programmet agerer anderledes end tilsigtet. [14]

[13] Refactoring af Product Backlog'en (hvilken anvendes i Scrum-regi, der beskrives afsnittet "Scrum") mellem hver iteration, er reelt set, hvad der i den agile tilgang fungerer som *integrated change control.* [97] For yderligere information vedr. Project Integration Management, se da eksempelvis "The PMBOK Guide" [96]

[14] Der er talte om høj kobling mellem moduler, når ændring i den ene kræver ændring i den anden. Opdeling af moduler med høj kohæsion(sammenhæng) vil fører til højere kobling.

Ifølge Uikey et al. [45] skaber den uformelle kommunikation mellem interessenter og udviklere til tider problemer såsom manglende evne til at håndtere systemets kompleksitet og hurtig-omskiftelige krav. Tillige kritiseres den agile metode for at låse udviklingsteamet ved ikke at tillader udskiftning af deltagere.

Cho [25] fremlægger flere fordele ved agil softwareudvikling såsom; kort udviklingscyklus, høj kundetilfredshed, få programfejl samt kort tilpasningstid til hurtigt-omskiftende forretningsmæssige krav, men han giver ligeledes udtryk for, at en organisation kan være tilbageholden med at skifte til et agilt paradigme af flere grunde. Heriblandt argumenteres der for, at projektet bliver stærkt afhængig af implicit viden, som resultat af den væsentligt reduktion i mængden af dokumentation; at metoden er ikke blevet tilstrækkeligt testet for sikkerhedskritiske projekter; samt at agile metoder kun kan lykkes med talentfulde personer, der favoriserer mange frihedsgrader.

Princippet om kundeinteraktion og co-location kan vise sig som endnu en svaghed ved den agile udviklingstilgang. Agile tilhængere anser ofte dette punkt som en styrke, men i praksis kan det vise sig at være en stor udfordring, hvis ikke kunden er i stand til at afsætte tid til at samarbejde med udviklerne, hvilket kan være tilfældet, hvis nøglepersonen er en ledere på et højt niveau. [31] Det er i særdeleshed problematisk at koordinere udviklingen optimalt mellem flere teams på et stort projekt, hvis disse er placeret på forskellige kontinenter rundt om i verden.

Som dette afsnit tydeliggør, er den agile udviklingstilgang ikke anvendelig i alle udviklingsprojekter og -miljøer. Altafgørende er det derfor, at disse kritikpunkter ikke negligeres, og at de antagelser, der ikke stemmer overens med det pågældende udviklingsmiljø, bliver adresseret og ændret i overensstemmelse hermed. [34]

3.3.1.3 Hybride udviklingsmetoder

Som alternativ til de to udviklingsparadigmer, er det som tidligere nævnt, en mulighed at anvende en hybrid tilgang. Der dog ikke meget videnskabeligt litteratur på området, hvilket kan være en følge af de utallige kombinationsmuligheder, der i sagens natur er mellem de to arketyper.

Udfordringen ved anvendelsen af hybride udviklingsmetoder er at finde en balance mellem de svagheder, der findes ved hver tilgang samt deres styrker og kompetencer. Herunder den højere kundetilfredshed, de lavere defektrater, den hurtigere udviklingstid og tilgangen til hurtigt skiftende krav, der kan opnås ved anvendelsen af den agile udviklingsmetode, samt den forudsigelighed, stabilitet og høje sikkerhed, der kendetegner en plan-drevne tilgange.

Der er her essentielt at skelne mellem agilitet på team-niveau og på virksomhedsniveau. Bill Curtis, senior vicedirektør og chefforsker for softwareanalysefirmaet CAST, beskriver behovet for en kombination af de to udviklingsparadigmer, i forbindelse med udviklingen af virksomhedsapplikation[15] således:

> *"What the mix of the waterfall method does is give attention to the overall architecture of the system at the start, and then after that you can get the benefit of the rapid feedback on what you are developing by doing it in short cycle releases."* [46]

Ved gennemførsel af store og komplekse softwareprojekter, kan anvendelsen af en agil udviklingsmetode være en udfordring i forbindelse med sammenkobling af iterative moduler mellem flere udviklingsteams, hvis ikke de følger det samme up-front arkitekturdesign og scope. Disse kan med fordel tage afsæt i en plan-drevet tilgang, hvilken vil danne grundlag for de krav, hvert agilt team skal opnå, mens den agile udviklingsproces ligeledes skal foregå imod et på forhånd fastlagt mål for projektet. [47]

[15] Som eksempel på en virksomhedsapplikation kan nævnes Microsofts "Business Solutions CRM". For mere information om virksomhedsapplikationer, se da "Developing for the Enterprise" [98]

Selvom der ikke anvendes en 100% agil udviklingsmetode, kan en af fordelene, Curtis omtaler, være den såkaldte *Fail fast, Fail Often*-tankegang, der kan skabe gennemsigtighed i udviklingen af et produkt. [48] Denne filosofi kan virke ulogisk, idet fiasko ofte forbindes med noget negativ, men konstant og kontinuerlig testning af produktet kan gøre det muligt for udviklerne at rette op på fejl, så snart de udvikles. Herved er det ligeledes muligt for kunden at identificere misforståelse eller problemer før de eskalerer. Alternativt, opdages problemstillinger i værste fald først efter endeligt release af hele projektet, hvorved fejlen kan vise sig uoprettelig.

3.4 OPSUMMERING

En organisation kan beskæftige sig med flere af de omtalte, fundamentalt forskellige, softwareopgaver på en gang, og som Del I har givet udtryk for, er den ene udviklingsmetode ikke overordnet set bedre end den anden, men derimod hver i sær bedst anvendelig i forskellige situationer. Håndteringen af en sådan forskelligartet software-portefølje vil derfor kræve en vifte af forskellige udviklingsparadigmer, udvalgt på baggrund af den enkelte opgaves art og størrelse.

Den plan-baserede tilgang egner sig bedst til forudsigelige projekter, hvor kunden har et klart billede af det ønskede produkt og kan udspecificere kravene hertil; og hvor en plan kan udformes indledende og forventes fulgt uden behov for fundamentale eller pludselige ændringer.

Den agile tilgang bør derimod anvendes til ”mindre” udviklingsprojekter, der kræver en kontant ajourførelse, hvor det er muligt at udnytte værdien i et selvorganiseret og tværfagligt udviklingsteam. Dertil kan sidstnævnte hybride tilgang i overvejende grad, være bedst egnet i forbindelse med udvikling af store og komplekse udviklingsprojekter, hvor det er fordelagtigt at kombinere styrker fra begge verdener.

Uanset det anvendte udviklingsparadigme, er det essentielt, at være opmærksom på de potentielle begrænsninger, der følger med.

4 SCRUM

Den mest populære af agile metode er i dag Scrum, og bygger på et empirisk grundlag. [49] Blandt andre agile metoder kan XP, Crystal, og Kanban nævnes.

Scrum er hverken en projektledelsesmetode i traditionel forstand eller en software udviklingsmetode, men nærmere et management framework til softwareudvikling. Scrum kan dog anvendes af enkelte udviklingsteams som et ledelsesværktøj. Desuden kan det danne grundlag for selve organisationsstrukturen, der kan være opbygget omkring Scrums principper og teknikker. [6]

Scrum anvender en iterativ, inkrementel tilgang til udvikling for at optimere forudsigelighed og for at kontrollere risici, i overensstemmelse med Det Agile Manifest, bygget op omkring mindre, selvorganiserede og krydsfunktionelle teams, indeholdende en Scrum Master, en Product Owner(kunderepræsentant) og fem udviklere. Foruden at definere disse roller, indeholder Scrum framework’et afholdes af seks forskellige møder samt anvendelse af 12 artefakter, med udgangspunkt i, at udviklere og kunden mødes dagligt, for ikke at separere det strategiske niveau med det taktiske og operationelle niveau. [6]

Product Owner’ens rolle askiller sig fra en traditionel Project Manageres idet han både har det finansielle ansvar og en dyp forståelse for kundens virksomhed og behov. Han skal agere som en produktvisionær og levere produktideer til teamet samt opmuntre og inspirere til at udvikle bedst muligt. [6]

Scrum opdeler udviklingsprocessen i iterationer, såkaldte sprints, der er tidsbegrænset udviklingscyklusser, der anbefales at vare to-fire uger. Herved opstilles der konstant tidsbegrænsninger, hvori teamet står overfor nye, nøje udvalgte udfordringer og opgaver, der skal implementeres i produktet. Dette sker i henhold til

”kravsspecifikationen”, der til forskel fra en plan-drevet metode, udvikles og tilrettes løbende, ved anvendelse af en såkaldt *Product Backlog*. [6] I Scrum, er Product Backlog’en det absolut vigtigste værktøj, da denne indeholde utrolig detaljeret oplysninger, og skitserer alle de krav, der er til produkter. Under det, der kaldes et Sprint Planning-møde, udvælger Product Owner’en og udviklingsteamet sammen, ud fra Backlog’en, hvilke krav og funktioner, der skal gennemføres i den kommende Sprint.

Hvor en kravsspecifikation, i et plan-baseret tilgang, normalt arrangerer kravene i to blokke *funktionelle* og *ikke-funktionelle* krav, beskrives de mere funktionelle specifikationer i agile projekter som korte *user stories* hvorefter de ikke-funktionelle elementer specificeres som *fit criterion*, for at sikre enighed om, hvornår en given opgave er færdiggjort.[16] [6] Disse user stories formuleres i korte og præcise sætninger, med udgangspunkt i spørgsmålet ”Hvilken funktion/mål ønsker brugeren, for at opnå, hvilket formål?”. [50] Hvis der er behov for ydere detaljering, er det muligt at samle flere user stories i et *theme*, og behovene kan desuden beskrives overordnet som et *epic*.[17] [6] Formålet med anvendelsen af user stories er at definere det bagvedliggende behov, samt at skabe forståelse for formålet herfor, hvilket ligger til grunde for vurdering af den fornødne tilstand eller kapacitet, der kræves.

Grundet den inkrementelle udvikling, implementeres og testes et fungerende stykke software, efter hvert eneste sprint, og skal konstant leve op til de standarter, guidelines og krav, der stilles til projektet. [6] Arbejdet udføres i et Scrumforløb således meget disciplineret, og enhver fejl, mangel eller handling vil straks efter hvert eneste sprint synliggøres, i overensstemmelse med førnævnte Fail fast, Fail Often-tankegang.

Kravene til det kommende sprint indføres i Product Backlog’en på baggrund af den feedback, teamet har modtaget på det allerede udviklet software. Udviklingen i et kommende sprint baseres endvidere på yderligere krav, udvalgt under Sprint Planning-mødet, samt organisationens behov og strategi, hvilke, som tydeliggjort ovenfor, kan ændre sig med tiden. En Product Backlog indeholder således alle de krav og behov, der stilles til produktet, og er den eneste kilde til at foretage ændringer i kundens krav. Backlog’en er på intet tidspunkt komplet, men dynamisk og udvikler sig i takt med produktet, fra kun at indeholde de absolut nødvendige krav, til at omfatte lister med alle de egenskaber, funktioner, krav, forbedringer og rettelser, der ligger til grund for udviklingen af produktet i fremtiden. [51]

Scrum-teamets arbejdsopgaver for det forestående sprint trækkes direkte fra Product Backlog’en frem for fra en, med tiden, gammel og potentiel forældet kravsspecifikation som anvendt i et plan-baseret forløb. Denne styres af Product Owner’en, der har til ansvar at styre udviklingen i den rigtige retning, og at sørge for, at teamet udvikler de rigtige funktionaliteter, i den rigtige rækkefølge. [6]

Det er i praksis muligt at anvende to forskellige former for Baclogs alt afhængig af, om samtlige teammedlemmer er *co-located* eller ej. Det anbefales, at der anvendes et fysisk manifestation af en Product Backlog, så som et task board med post-its, hvorpå der kan nedfældes user stories mv. Alternativt er der udviklet forskellige agile ledelses-værktøjer, så som ScrumWork Pro, det er en interaktiv cloud-løsning, som søger at imitere en fysisk tavle. [52]

Da alt udvikling og samtlige referencer kontrolleres via Backlog’en, er humlen at denne konstant opdateres, og at det er muligt at støtte og beskrive, hvad der kræves af udviklingsteamet, før påbegyndelsen af det kommende sprint.

[16] Ved anvendelse af en Product Backlog kan der dog opstå uenigheder om netop aspektet ”hvornår et krav kan accepteres som værende opfyldt”. [91] Denne problemstilling refereres til som ”*the definition of done”*. Da problemstillingen hovedsageligt vedrøre et en juridisk diskussion, behandles denne i afsnitet ”Anvendelse af en Product Backlog” i Del I.
[17] Et epic er en ”større historie” der dækker over en samling a user stories, der på nærværende stadie endnu ikke er blevet brudt op i enkelte implementerbare funktionsbehov. [101]

4.1 AGILE REQUIREMENTS DEFINITION AND MANAGEMENT (RDM)

RDM er et begreb, der dækker over processen med at *udtænke, dokumentere, analysere, prioritere*, og *nå til enighed* om krav til softwaren, der skal udvikles, og derefter processen med at *kontrollere, styre* og *føre tilsyn* med ændringer og risici. [53] For at opnå den mest effektive anvendelse, bør RDM implementeres som en kontinuerlig proces, og bistå udviklingen gennem hele projektets livscyklus, og derfor være en velintegreret proces i anvendelsen af Product Backlog'en. Det er desuden essentielt at krav og user stories vægte således, at ressourcerne er afsat sådan, at højprioritets- samt højrisiko-krav adresseres først. [53] Der bør ligeledes opstilles en række grundlæggende krav, der kan agere som målbare standarter for succes, hvorved både kunden og udviklere er indforstået og enige i, hvad der kræves af produktet.[18] [53]

VÆRDIEN AF RDM

Forskning viser, at 70% af alle softwarefejl opstår I forbindelse med kravspecificering og arkitektur-design, men først opdages i senere faser, samt at jo senere i udviklingslivscyklussen, en fejl konstateres, desto mere bekostelige er den at rette op på. [54] En fejl, fundet på beta testings-stadiet vil kræve 15 timer at rette, relativt til fem timer, hvis fejlen var konstateret i kodningsstadiet, mens omkostningerne ved at reparere en fejl fundet efter produkt-release er op til 30 gange så høje som i kravspecificerings-fasen. [54] [55] Dertil estimeres det, at uventede nedetid i 40% af tiden skyldes softwarefejl, hvilket ifølge Gartner [56] i gennemsnitligt koster $100.000 per time for forretningskritiske systemer.

Disse analyser, er fortaget på traditionelle plan-baserede softwareprojekter, og en agil udviklingsmetode vil, ved hjælp af en dedikeret RDM-strategi, kunne minimere sandsynligheden for at disse fejl opdages for sent og bliver byrdefulde for organisationen. Dog kræver dette, at Product Backlogs holdes ajour med detaljerede og præcis-formulerede oplysninger ved hjælp af RDM, der gør det muligt at anvende Backlog'en som det værktøj, den er tiltænkt. Jo flere sprint, et projekt gennemløber, desto flere og mere komplekse detaljer kommer den til at indeholde, hvilket fremtidige sprint beror på.

Dette kræver, at Product Owner'en i samarbejde med udviklingsteamet klart kan formulere, hvad der skal udvikles, og definere af hvilken kvalitet, førend udviklerene har behov for informationen. Dertil er det vigtigt at være opmærksom på, at risikoen stiger i takt med at afstanden mellem definering af krav og udviklingen på baggrund deraf forøges, grundet risiko for at miste vitale teammedlemmer, viden og generel teamtilgængelighed. [57]

5 AGIL SOFTWAREUDVIKLING FRA ET ERHVERVSØKONOMISK PERSPEKTIV

Management-tendenserne for softwareudvikling er på international blevet analysere af blandt andet The Standish Group. Den årlige rapport, CHAOS Report, viste i 2014 [58], at hele 31,1% af de analyserede projekter blev annulleret før, de nogensinde var færdig; at 52,7% af projekterne ville koste op mod 189% af deres oprindelige estimat; samt at kun 16,2% af projekterne overordnet set kunne karakteriseret som succesfulde. Ved definitionen "succesfuld" blev i nærværende rapport anvendt som beskrivelse for et projekt, gennemført til tiden og inden for budgettet, selvom disse langt fra blev gennemført i overensstemmelse med de oprindelige krav til egenskaber og funktioner. [58] Blandt de vigtigste årsager til dette nedslående resultat var, at mange projekter skulle genstartes indtil flere gange.

Der findes ikke et entydigt svar på, hvorfor softwareprojekter i adskillige tilfælde svigter. Fiasko kan forekomme på ethvert tidspunkt i et projekts cyklus og af flere årsager. Forskning afslører dog *Scope management* som det største management-relaterede årsag til projektfiasko, hvilket var tilfældet i 82% af projekterne, efterfulgt af dårlig *project management* og *change management*. [59]

[18] Se desuden nedenstående juridiske diskussion vedrørende "the definition of done" i afsnittet "Anvendelse af en Product Backlog".

I en analyse, fandt Taylor [59] at kun 130 af 1027 observerede softwareprojekter blev gennemført med succes, deriblandt var kun 2,3% af udviklingskontrakter succesfulde, selvom halvdelen af de analyserede kontrakter faldt i denne kategori. Problemerne vedrørte hovedsageligt *kravspecificerings-fasen* og *uklar målsætning og krav*, efterfulgt af *manglende business commitment* samt *ændringer vedr. forretningskrav*. Til sammenligning angav The CHAOS Report [58] *manglende bruger-input, ufuldstændige krav og specifikationer* samt *ændringer i krav og specifikationer* som de største årsager til projektfiasko.

Hertil vedrørter de mest kritiske succesfaktorer *klare, detaljerede krav, klare kontrolforanstaltninger for projektændringer* samt *brugerinddragelse* og *støtte fra direktionen*, mens de vigtigste færdigheder blev rangeret som *kommunikation, lederskab og motivation, organisatoriske færdigheder, planlægningsfærdigheder* og *team building-færdigheder*, og slutteligt *IT-viden*. [58] [59]

Der er delte meninger om, hvorledes disse udfordringer skal håndteres. Mens nogle forfattere mener, det er tilstrækkeligt at adressere problemstillingerne vedrørende målsætninger og krav, er det ifølge Taylor [59], af afgørende betydning for projektets succes, at have den rigtige Project Manager. Han argumenterer ligeledes for, at store og komplekse projekter kræver yderligere støtte til nogle af lederens ansvarsområder, mens alle deltagere i projektet skal føler et ansvarlig og ejerskab for projektets succes. Manglende støtte og engagement fra kundens siden, samt fokus på omkostninger og tid frem for kvalitet og troværdighed, danner, ifølge Taylor [59], netop grundlaget for fiasko, hvorfor det kræves, at kunden accepterer lige ansvar for projektets succes/fiasko, og udgør en aktiv støtte. Project manageren skal således implementere en effektiv kommunikationsproces og sigte mod at lede og motivere hele teamet frem for at presse på og true med sanktioner.

Eftersom 76,3% af de adspurgte ledere i Taylors forskning [59] kunne rapportere, at intet softwareprojekt efter deres erfaring nogensinde var blevet leveret i henhold til de oprindelige specifikationer, anbefaledes det tillige, at der i den indledende planlægning tages hensyn til klientens potentielt skiftende forretningsmæssige behov. Desuden tilrådes det at tidsplanen for et softwareprojekter begrænset så meget som muligt, for at skabe mindst mulig forsinkelse mellem kravsspecifikation og levering af en fungerende stykke software, hvilket The CHAOS Report [58] argumenterer for, kræver en iterativ proces med design, prototype, udvikling, testning og implementering af små elementer.[19]

Disse forskningsresultater influeres af det faktum, at intet softwareprojekt er ens og kan sammenlignes fuldstændigt, hvorfor der ikke kan udledes et entydigt svar på, hvorledes et softwareprojekt bør tilgås, og hvilken udviklingsmodel, der bør anvendes. Dog bidrager disse forskningsresultater med et indblik i de mest kritiske faktorer, der bør adresseres. I sær i forbindelse med udviklingsprojekter, er årsagerne til projektfiasko relevante at tage i betragtning, idet kun 2,3% af disse projekter blev gennemført med succes, mens drift og vedligeholdelse, til sammenligning var succesfulde i 18.2% tilfældene, og datakonverteringsprojekter i hele 79.5%. [59]

Reinertsen har i samme tråd fremsat 12 faktorer, der ifølge ham, forhindre succesfuld produktudvikling: [60, p. 4]

- *"Failure to correctly quantify economics*
- *Blindness to queues*
- *Worship of efficiency*
- *Hostility to variability*
- *Worship of conformance*
- *Institutionalization of large batch sizes*

[1919] Denne proces omtales i IT-kredse som *growing* software, i modsætning til begrebet *udvikling* af software. Growing af softwaren kræver, at 1) brugeren engageres tidligere, 2) hver komponent har en ejer, og 3) forventningerne er realistisk indstillet. [58] På trods af denne sondring, vil begrebet *udvikling* dog stadig anvendes i denne afhandling, for ikke senere at skabe forvirring for den ikke softwareudviklings-kyndige læser.

- *Underutilization of cadence*
- *Managing timelines instead of queues*
- *Absence of WIP (Work In Progress) constraints*
- *Inflexibility*
- *Noneconomic flow control*
- *Centralized control"*

Reinersten [60] argumenterer for, at disse negative opfattelser forstærker hinanden, og skaber en risikoaversion, der drive innovation ud af udviklingsprocessen, og et fokus på reduktion af variabilitet frem for rentabilitet.

Uanset hvilken metode, der anvendes til udvikling af et nyt og brugerdefineret software, udfordres parterne af omskifteligheden i *hvad*, der skal leveres, grundet umuligheden i at vide hvad, der endeligt er behov for. Det er essentielt for gennemførsel af et succesfuldt projektets, at acceptere, at denne uformåenhed er et systematisk og nødvendigt onde. [6] Accepten af det simple faktum, at det er en nødvendighed at indgå en et samarbejde om udvikling af et produkt, hvilket på forhånd ikke er muligt at beskrive detaljeret, er i virkeligheden det første skridt på vejen mod et succesfuldt projekt. [60]

Såfremt produktet består af en innovativ løsning, der har til formål at opfylde et bestemt behov, er det ikke muligt, på forhånd at udspecificere kravene til det endelige produkt. [6] Her består værdien i selve udviklingen frem for produktet, hvilket en plan-drevet tilgang ikke kan anvendes håndtere. Denne kan ikke tage højde for de mange uforudsete faktorer, der naturligt vil manifestere sig i takt med at løsningen udvikler sig.

Et interessant resultat af The Standish Group [61] vedrører anvendelsen af en BRUF-tilgang i den plandrevne tilgang. Ud af de analyserede, succesfulde projekter, der anvendte en traditionel udviklingsmetodologi, blev hele 45% af alle de på forhånd-definerede krav aldrig brugt, mens 19% sjældent blev brugt efterfølgende. Dette betyder, at kunden har investeret penge i udvikling af software, der viser sig at være hverken relevant eller nyttigt.

Ved anvendelse af en agil tilgang er det muligt at minimere tid og ressourcer brugt på egenskaber, som viser sig at være overflødig for produktet. Dette skyldes at der ikke opstille en opfattende kravs-liste, men at opmærksomheden henføres til det reelle behov og udviklingen deri. Potentielt kan udviklingstiden af denne grund forkortes ved ikke at investere tid i udvikling af i alt 64% unødvendig software[20], hvorfor produktet kan vise sig at være færdigudviklet på kortere tid end anticiperet.

5.1 AGILE PROJECT MANAGEMENT

I det agile udviklingsmiljø, er Project Management-fokus flyttet fra *planlægnings*- til *udførelsesfasen*. [3] Det er i projektudførelsen, de vigtige beslutninger træffes, da disse er altafgørende for projektets succes eller fiasko. I modsætning til et plan-drevet procesforløb, fungere områderne projektdefinition og planlægning i den agile tilgang som støttefunktion, hvorfor selve projektet vil ikke være afhængige af disse fra en start.

Der er ifølge Chin [3] tre dimensioner, der drive behovet for Agile Project Management: *brug af unik ekspertise, projektuvished* (herunder *intern uvished og ekstern uvished)* samt *uopsættelighed.*

Hvis der i et udviklingsprojekt er specielt behov for et specifikt team og deres kompetencer eller et enkelt individs ekspertise, er dette et tegn på, at der er behov for en agil udviklingsmetode. Til forskel fra klassisk projektledelse, hvor ressourcerne i en vis grad er udskiftelige, er der her ingen erstatning for ekspertens unikke ekspertise. [3] Udvikling efter en agile metode kræver et netop så talentfuldt et team, idet målet er

[20] De 64% dækker her over de 45% krav, der aldrig blev brugt efterfølgende plus de 19%, der kun sjældent blev brugt.

udvikling af fuldt funktionelt software på blot få uger, hvorved enhver kvalitetsmæssig mangel vil vise sig allerede ved første iteration og release.

Intern uvished omhandler de faktorer, der kan influeres eller kontrolleres af Product Owner'en, så som scope, tidsplaner og interne omkostninger. [3] Derimod består de eksterne faktorer, der er uden for Product Owner'ens kontrolsfære, så som de branche-forhold, der påvirker projektet, herunder konkurrencen på markedet, den teknologisk udvikling, samt strategiske beslutninger på et højt niveau i organisationen.

Ved anvendelse af en agil metode, kan selve udviklingsparadigmet skabe en følelse af manglende kontrol, idet det ikke er muligt for kunden samt dennes interessenter på forhånd at vide, præcis hvilken værdi de får for deres penge, samt i hvor stor en grad, eksempelvis den teknologiske udvikling vil påvirke projektet. Dog bevirker den hurtige levering af første udgave af softwaren, indeholdende de vigtigste og højst prioriterede krav, et hurtigere return on investment. [6] Således minimeres risikoen for at projektet ender med at skulle annulleres, mens konsekvenserne som følge af de eksterne faktorer ligeledes minimeres grundet projektets fleksibilitet.

Chin [3] argumenterer for at indvirkningen af uvished på et projekt er en funktion af projektets uopsættelighed, samt at jo større faktoren for uopsættelighed er, desto større effekt vil usikkerheden have på projektet. Sidste dimension omhandler således faktoren for uopsætteligheden. Denne afhænger af, hvor presserende det er, at få projektet færdigt til tiden, hvilket igen afhænger af virksomhedens størrelse og konsekvensen af fiaskoen. Et fejlslået projekt kan resultere i konkurs for en lille virksomhed, mens det ikke nødvendigvis har samme effekt på en stor organisation. Jo større uvished og uopsættelighed et projekt er påvirket af, desto større vil behovet for Agile Project Management således være.

KOMPRIMERING AF TIDSPLANEN

En problemstilling, der ofte præger et projekt er, hvorvidt ændringer påvirker projektets overordnet tidsfrist. For at forkorte varigheden af et projekt uden at ændre projektets scope, kan teamet anvende flere komprimeringsteknikker, heriblandt at *crashe* tidsplanen, hvorved omkostninger og konsekvenser af potentielle kompromisser analyseres for at afgøre, hvordan den størst mulige komprimering af tid opnås til gengæld for den mindst mulige inkrementelle omkostning. [62]

Endnu et værktøj er *fast tracking* hvilket betyder, at opgaver, der normalt udføres sekventielt, udføres parallelt, i såkaldt *overlapping stage phases*. Herved påbegyndes en senere fase, førend den tidligere afsluttes, så projektet samlet set afslutte hurtigere. Anvendelsen af overlappende opgaver på denne måde resultere dog ofte i omarbejde og øget risici, idet, det ikke er muligt at fundere den senere påbegyndte fase på feedbacken af den forudgående fase. [62]

Dertil findes det mere agile komprimeringsværktøj, *iterative relationship*, hvor den efterstående fase planlægges, mens den forudgående fase udføres. [63] Dette er ikke blot et komprimeringsværktøj i Scrum, men fundamentet for selve opbygningen af projektets livscyklus. Kravene for det kommende sprint prioriteres og udspecificeres, som tidligere nævnt, på et sprint planning-møde, mens feedback fra brugerne ligeledes inkorporeres. Herved har udviklerne størst mulighed for at levere præcis det produkt, kunden efterspørger på kortest mulig tid.

5.2 DEN PROJEKTORIENTERET ORGANISATION

Beslutningen om at adoptere en agilt metode involverer, ud over ovennævnte betragtninger, desuden overvejelser om organisationskultur samt hvilket udviklingsparadigme, der på nuværende tidspunkt anvendes. Et paradigmeskrift, kræver en omstrukturering af udviklingsorganisationen, hvis denne hidtil har fulgt et traditionelt plan-drevet paradigme, hvilket påvirker både ledelsesfilosofien, arbejdstilgangen og arbejdsmetoder-

ne. For at opnå succes med anvendelsen af en agile udviklingsmetode, er det derfor essentielt at organisationen er moden i et agilt regi, og at beslutningsstrukturen følger det implementerede paradigme. [21]

Matrix Management er en velanset model til håndtering af flere projekter på tværs af større og multifunktionelle organisationer. [3] Dog søger denne model per definition at adskille forretningsmæssig- og projektmæssig beslutningsprocesser, hvorved et team stilles overfor to managers, hvilket ikke er optimalt i et agile miljø. Her er netop det modsatte er ønsket, i forsøget på at integrere projekterne som kernen i virksomheden. En sådan opdeling vil fordre en differentiering mellem opfyldning af og udførelse af centrale projekter, frem for at skabe en balance mellem opfyldelse af projektets kortsigtede og organisationens langsigtede mål.

I modsætning til Matrix Management-strukturen, er målet i en *projektorienteret organisationsstruktur*, i overensstemmelse med Det Agile Manifest, at opbygge en organisation omkring dets essentielle projekter, hvorved det er muligt at investere i virksomhedens unikke eksperter. [3]

> *"Agile is not only about technology or project management. Agile is a mindset embracing ALL activities in a company from top management to the youngest trainee. The agile mindset is crucial to be competitive"* – Michael Holm, President and CEO, Systematic. [64]

Herved nås de kommercielle målsætninger netop ved gennemførelse af succesfulde projekter, mens de mange separate og til tider modstridende målsætninger, der kan opstå som følge af Matrix Modellen, undgås. Endvidere hæmmes projektfremskridt ikke som resultat af skiftende krav, hvilket ligeledes er et problem ved anvendelse af Matrix modellen.

> *"If your projects are your business and your key players are the heart and soul of your project, then it stands to reason that your key players are your business." - Chin, Gary* [3, p. 35]

Ved gennemførslen af en Risk Management proces[22], er det essentielt at de forretningsmæssige overvejelser integreres i projektetbeslutningen. Et projekt kan gennem en traditionel Project Management-betragtning anses som succesfuld, såfremt scopet blot leveres til tiden og inden for budget, selvom produktet faktisk er en fiasko. [3] Dette skyldes at projektet ikke tilpasses den forretningsmæssige realitet, men alene har til formål at opfylde de tidligere bestemte og potentielt forældede krav. En sådan tilgang vil være katastrofal for en projektorienteret organisation, hvorfor en sådan orientering kræver et fokus på at balancere distribuering af risici, allokering af ressourcer mellem projekter, samt fordeling af profit mellem projekt- og forretningsenheden. [65]

Beslutningsprocessen i et agilt forløb bør på denne baggrund inddrage aspekter af alle projektets dimensioner, heriblandt markedsanalyse; produktionsomkostninger; en forskydning af tidsplanen for at nå det oprindelige scope, hvis det stadig er anvendelig; tilgængeligheden for teknisk support til den første udgivelse og opgradering; det overordnede finansielle perspektiv; samt andre relevante kommercielle hensyn. Såfremt der er tale om store risici, kan denne fremgangsmetode indebære en stor forøgelse af de samlede omkostninger,

[21] Modenhedsniveauet testes gennem en modenhedsmodel for at giver et indtryk af hvor, det den største udfordring befinder sig, og dermed hvor den største udvikling bør ske. Der findes adskillige modenhedsmodeller for software udviklingsprocesser, der muliggøre en klassificeringen af udviklingsorganisationers processer. Fælles for disse modeller er at de opdeler organisationens modenhed i flere stadier, for på den måde at kunne sammenligne organisationer imellem samt at for at tilskynde udvikling og forbedringer, samt konkurrence og diversificering. [94] Mest kendt er nok Capability Maturity Model Integration (CMMI), der bidrager med et såkaldt road map for procesforbedring. Grundet et bredt anvendelsesområde, støtter CMMI dog ikke i tilstrækkelig grad organisationer, der anvender agile udviklingsprocesser [103], hvorfor denne model ikke vil uddybes yderligere.

[22] Risk Management ifm. den agile udviklingstilgang behandles nedenfor i afsnittet " Risikohåndtering".

en forskydning af tidsplanen og potentielt en ændring af det overordnede scope. Ikke desto mindre kan disse foranstaltninger være at foretrække, såfremt den samlede cost/benefit-analyse understøtter det.[23] [3]

5.3 RISIKOHÅNDTERING

Risk Management er et tilbagevendende aspekt af et hvert udviklingsprojekt, uanset om denne følger en plan-drevet eller en agil tilgang. Det er essentielt at kunne skelne mellem en risiko og et problem, idet en risiko kan influeres, mens et problem, vedrøre en hændelse, der med sikkerhed vil opstå. [66] Dertil spiller risikoens begrænset varighed den rolle, idet risikoen kun er tilstede, indtil et bestemt stadie er overstået eller en bestemt hændelse indtræder. [67] Det er naturligvis vigtigt at tage hånd om både problemer og risici, men vigtigst af alt er at bemærke, at risici kan forbygges.

Begrebet risiko defineres og opfattes forskelligt i forskellige brancher [68], og kan referere til både negative samt positive omstændigheder. [66] [69] Eksempelvis definerer ISO [69] en risiko som *"the effect of uncertainty on objectives"*. Dertil udtrykkes risikoen ofte som en kombination af *konsekvensen* af en *hændelse* og den tilhørende *sandsynligheden* herfor. [70] [71]. Begrebet risiko, vil i denne afhandling følge denne definition, men for gennemsigtighedens skyld, alene refererer til en trussel, mens den positive pendant hertil vil betegnes som en *mulighed.* Sidstnævnte behandles nedenfor i afsnittet " Risk Intelligence".

Konsekvensen af en risiko kan være eksempelvis tid, penge, markedsandele eller lignende. Tabets størrelse, kombineret med sandsynligheden for at risikoen indtræffer, definerer risikoens alvorsgrad, og derved hvorledes flere risici skal prioriteres inter partes. [68] Dog er det værd at bemærke, at en sådan en matematisk kalkulation af risici, på baggrund af en sandsynlighedsberegning, ikke skelner mellem risici, der involvere potentielt store konsekvenser, forbundet med en lille sandsynlighed for indtrædelse, og hændelser der forekommer relativt ofte, men involvere en forholdsmæssig lille konsekvens. [68] Det er essentielt at kunne differentiere imellem disse, idet de kræver forskellige Risk Management-tilgange, hvilket kræver en baggrundsviden for risikoens karakter, og en opmærksomhed på uvished fremfor blot sandsynligheden.[24]

Risk Management omhandler processer, metoder og værkøjer til håndtering af risici, [72] og involverer 1) en identifikation af projektets risici; 2) en analyse for at afgøre hver risikos alvorsgrad; 3) at disse prioriteres i forhold til deres alvorsgrad; samt 4) at der udvikles rette strategier for de højst prioriterede risici. [67] Risk Management kræver desuden løbende monitorering og sikring af, at de udvalgte strategier faktisk reducerer sandsynligheden for at risikoen indtræffer.

En Risk Manageent tilgang, udviklet til et flere-årigt plan-drevet projekt, er dog ikke umiddelbar anvendelig på et agilt projekt, hvor udviklingsprocessen forkortes til et enkelt sprint. Udfordringen ved den agile tilgang er derfor at identificere risici og iværksætte en risikoprocedure tids nok til at kunne reagere optimalt, således at følgevirkningen ikke får katastrofale konsekvenser for projektet succes. Som det vil fremgå af følgende afsnit, spiller det selvorganiserede team her, der centrale rolle.

5.3.1 Det selvorganiserede udviklingsteam

I forbindelse med Risk Management er det et altovervejende faktum, at agil udvikling forudsætter et samarbejde og et fælles ansvar for risici og succes, og ikke, som et plan-drevet projekt, er indgået som et traditionelt aftager-leverandørforhold. Denne konstellation bevirker at potentielle risici påvirker begge parter og

[23] Der er dog begrænsede muligheder herfor, såfremt projektet er underlagt Udbudsloven. Ændringsmulighederne i denne forbindelse, behandles i afhandlingens det juridiske analyseafsnit " Ændringsmuligheder og kontrolforanstaltninger for projektændringer",

[24] For en yderligere uddybelse af denne diskussion, se da Aven [68].

ikke blot leverandøren, hvorfor kunden, repræsenteret af Product Owner'en, pålægges et stor ansvar for at holde Product Backlog'en ajour, idet denne danner grundlag for risiko-identifikationen.

I Scrum anses individet, i overensstemmelse med det Agile Manifest, som det største aktiv, og udviklingen overlades, som nævnt, til et selvorganiseret teamet, inklusiv Product Owner'en, der som følge selv organiserer deres opgaver igennem hele forløbet.

Ved anvendelsen af et selvorganiseret team, opnår hver enkelt teammedlem mest mulig frihed og ansvar, grundet den overbevisning, at vedkommende således bedst muligt udnytter sin ekspertise og kreativitet, og på samme tid har mulighed for løbende at udvikle de nødvendige sine kompetencer yderligere. [6] Hver enkelt teammedlem tildeles dog ikke uendelig frihed, men selve tilgang bygger på en anerkendelse af *"meaningful interaction rules, discipline, personal responsability, thinking together, assisting eachother, and not using your knowledge simply to shine personally."* [6, p. 12]

Overordnet set antages det, at folk har en fundamental interesse i at bidrage med deres ideer for at forbedre ting eller for at udvikle nye, hvis de gives mulighed herfor. Det forudsættes, at de enkelte individer i et selvorganiseret team er i stand til at dele viden og anvende den i forskellige situationer, samt er i stand til at samarbejde i forskellige konstellationer i udviklingsforløbet. Baggrunden herfor er det faktum, at individer producerer mere effektivt og kreativt, hvis de ikke bare har forståelse for *hvad* de udvikler, men ligeledes en forståelse hvor *hvorfor.*[25] [6]

Hvor et udviklingsteam i et agilt projekt er 100% selvorganiseret har Product Owner'en ikke indflydelse på, hvor mange ressourcer, der henføres til projektet og den enkelt opgave. Det er overladt til teamet selv, at administrere den påtaget opgave. Dog foreskriver teorier vedrørende agile udvikling, herunder SCRUM, at dette ikke bør opfattes som et problem, da selvorganiserede teams performer bedre, og eftersom projektet ikke bør estimeres i henhold til tid og indsats men på selve udfaldet af denne indsats. [6] Det er således op til teamet selv at afsætte nok tid og ressourcer til at nå det aftalte mål inden for hver sprint.

Forud for et sprint, bør der i forbindelse med sprintplanlægningen udføres en Risk Management proces for det forelæggende sprint, og jo større fokus der tillægges RDM-processen, desto lettere er det at identificere og håndtere risici effektivt. [6] Risk Management-opgaven varetages af teammedlemmerne, der antages at være nærmest til at vurdere de involverede risici, herunder sandsynligheden for indtræden samt konsekvenserne heraf. [67] Risk Management processen bør adopteres på en sådan måde, at alle medlemmer af teamet er i stand til at udføre den hurtigt og effektivt. Smith & Pichler [67] anbefaler følgende "struktureret brainstorm"-fremgangsmetode: 1) *Evaluér* Product Backlog-elementerne; 2) udfør *risiko-identifikation, -analyse* og *–prioritering;* 3) udfør *risiko respons aktiviteter* ved at kortlægge de identificerede risici ift Backlog'en; 4) udvælg de dele af Backlog'en, der relatere sig til de mest kritiske risici som et *work item*[26]; samt 5) udled målet for sprintet.

Med udgangspunkt i ovenstående overvejelser, i forbindelse med en projektorienteret organisation, er det essentielt, at teamet tildeles den nødvendige beslutningsmandat. Denne decentraliseret Risk Management-strategi kræver endvidere, at der i organisationen skabes en kultur, der gør det muligt for den enkelte softwareudvikler at besside de nødvendige kompetence, til at kunne medtage kommercielle hensyn i denne proces. Såfremt disse forudsætninger ikke efterleves, hæmmer det selve potentialet ved det agile udviklingsparadigme, der således bliver en risiko i sig selv.

[25] Denne antagelse grunder i princippet bag Knowledge Management, hvilket behandles i afsnittet "Agile Innovation Management".
[26] "Work Item" er et begreb anvendte Project Managers for en given arbejdsopgave, skal udføres på et projekt. [106]

Dertil kan et enkelte teammedlems manglende kompetencer for både udvikling og Risk Management, vise sig bekostelig for projektet, hvilket ligeledes er tilfældet for Product Owner'ens evner og mulighed for at udføre den tilstrækkelige RDM-indsats.

5.3.2 Proaktiv Risk Management

Risk Management i et agilt regi, bør indeholde et proaktivt aspekt og fokusere på at undgå at risici udvikler sig til reelle kriser, og desuden på selve håndteringen af Risk Management. Udførelsen bør sker på en sådan måde, at den ikke blokerer potentialer og sætter projektets mulighed for succes i fare. [66] Der bør fokuseres på at udvikle langsigtede strategier, om skaber mulighed for at reducere potentielle risici og ikke mindst deres effekt, idet pointen bag implementering af Risk Management må være, at omkostningerne herved og forebyggelse af risici, er relativt mindre end en reaktiv håndtering af en risiko, efter denne har udviklet sig til en reel krise. [72]

5.3.2.1 Risk Intelligence

I nær tilknytning til Proaktiv Risk Management findes *Risk Intelligence*, hvilket omhandler modenhedsniveauet i henhold til risikohåndtering. [72] Denne tilgang vedrører selve organisationskulturen, og omhandler udnyttelse af risici frem for eleminering heraf, med udgangspunkt i de potentielle muligheder en risiko kan medføre. I denne forbindelse er der overordnet set fem forskelige responsmuligheder til en risiko: *undgåelse, reduktion, fordeling, accept* og *udnyttelse*. Valget af respons beror på en cost/benefit-analyse af sandsynligheden og effekten af den enkelt risiko. [72] Her er det selvsagt sidstnævnte respons, der åbner op for potentielle fordele.

Som eksempel herpå kan selve anvendelsen af en agil udviklings metode ses som en *udnyttelses-respons,* idet fokus placeres på at imødegå og udnytte nye teknologiske trends og viden løbende i udviklingsperioden. Som kontrast hertil skabes der, med anvendelse af en på forhånd udspecificeret kravspecifikation, i et plandrevet forløb, risiko for, at flere af de opstillede krav med tiden vil være forældet inden produktet implementeres.

Et fokus på den mulige udnyttelses-respons, og dermed potentialet for at optimere og skabe værdi, kaldes *Opportunity Management*, og er en integreret del af en proaktiv tilgang til Risk Management. Denne tilgang påvirker organisationens strategiske beslutninger, og kan potentielt øge produktiviteten, samt forbedre den finansiel gennemsigtighed. [73]

I denne forbindelse vedrøre begrebet mulighed *"all mature chances that are ripe for decision"* [74, p. 13] og vedrøre, som risici, sandsynligheden for at en hændelse vil indtræffe med en vis konsekvens. Muligheder bør gennemgå samme management proces som risici, herunder identifikation, analyse, planlægning og håndtering af muligheden. [74] Vigvigtigst af alt, bør Opportunity Management søge at gøre organisationen i stand til at identificere og reagere på de potentielle muligheder, de præsenteres for, hvilket kræver en fleksibel udviklingsproces så som den agile.

5.3.2.2 Contingency plan

Hvor der kræves ændringer i projektets overordnede plan, kan en såkaldt *contingency plan* anvendes i både traditionelle og agile projekter. Forskellen er dog stor i forhold til hoverledes en sådan plan anvendes. Contingency-strategier i traditionelle projekter søger ofte den korteste vej omkring udfordringen, mens strategien i agile projekter fokuserer på at finde alternative ruter mod målet. [3] I et agilt miljø fremlægges der indledende en foreløbig plan for projektets forløb, hvilken ikke forventes efterfulgt i hele projektets livscyklus. Den anvendes blot til at guide teamet i den rigtige retningen, indtil der foretages kursændringer som følge af uforudsete problemstillinger. Der er her forskel på begreberne *uforudset* og *uventet*, idet udfordringer og ændringer *forventes* i den agile udviklingsproces, selvom de ikke på forhånd kan forudses. [3]

Agile projekter skaber proaktivt mulighed for effektivt at kunne navigere reaktivt i takt med projektets udvikling. Dette er gældende både hvad angår reaktioner over for muligheder og risici. Denne reaktive ageren opstår som følge af den høje usikkerhed, de forholdsmæssige korte og detaljerede planlægningshorisonter kan føre med sig. Dette kan dog imødekommes ved hjælp af proaktive foranstaltninger, der muliggøre udnyttelse af klare forventning om fremtidige ændringer, vedrørende eksempelvis ændrede kundepræferencer og -behov eller nye teknologisk viden. [3]

Den inkrementelle levering og testning af produktet bidrager ligeledes med, at problemer og fejl, som ovennævnt, konstateres langt tidligere i projektet, hvilket resultere i langt lavere omkostninger, end hvis de først var blevet identificeret efter endelig levering og implementering. Fleksibiliteten i den agile tilgang gør det derfor ikke muligt at spotte potentielle muligheder og at adressere disse, men kan ligeledes minimere konsekvenserne af visse risici.

5.3.3 Agile Innovation Management

Den agile udviklingstilgang beror på behovet for løbende at kunne udnytte nye potentialer og udvikle innovative løsningsmuligheder. Innovation omhandler succesfuld vidensdannelse, indlæring og vidensdeling, gennem udvikling og udnyttelse af viden. Fra opfattelse af et behov eller identifikation af en mulighed, til idégenerering, udviklingen af et ”produkt” og implementering for at skabe værdi for en bruger. [75] Innovation er en proces, og kan derfor håndteres, herunder organiseres, kontrolleres, planlægges, implementeres, systematiseres og måles på.

Studier [76] viser, at innovation og performance i en organisation i stor grad påvirkes positivt af effektiv *Knowledge Management*, der kan opdeles i tre parametre: *vidensdannelse*, *vidensdeling* samt *reaktionsevne over for viden.* Heraf kan kun netop *reaktionsevnen over for viden*, med sikkerhed kan konkluderes at have en direkte indvirkning på finansielle outcome, der ligeledes positivt påvirkede performance.

Darroch [76] argumenterer for, at organisationer, der anvender *incemental innovation*, hvor et eksisterende produkt løbende forbedres og tilføres nye funktioner, ofte er mere tilbøjelige til at have en veludviklet Knowledge Management-adfærd og effektiv anvendelse af andre tilgængelige ressourcer. Desuden argumenteres der for, at innovative organisationer per definition har en god reaktionsevnen over for ny viden, idet innovation grundlæggende er en reaktion på ny viden. Organisationer, der anvender en agil udviklingsmetode har grundet en inkrementel tilgang til udvikling af software, derfor bedre forudsætninger for at være innovative end ”plan-baserede” organisationer. [76]

Denne opfattelse stemmer overens med definition af en agile organisation som værende én, der muliggør både effektiv håndtering og anvendelse af viden. [77] Dove [77] placere i sit studie stor vægt på den indbyrdes afhængighed mellem Knowledge Management og organisationens *ændringsfærdigheder*, som vedrører evnen til anvende alle former for viden effektivt, herunder både reaktivt og proaktivt. Netop disse parametre anses som værende grundlæggende kvalifikationer, der muliggør agilitet, hvilket ifølge Dove [77] ikke bør opfattes som et mål eller en strategi. Han mener derimod at agilitet i dag bør accepteres som en fundamentalt nødvendighed for enhver virksomheds eksistensgrundlag, grundet hurtig teknologisk udvikling, hurtig udvikling af ny viden, og fornyede mekanismer. [77]

Dertil er det relevant at have sammenspillet mellem Knowledge Management, ændringsfærdigheder og organisatorisk læring in mente, eftersom ændringer og værdiskabelse først opstår, når nogen har lært noget nyt. [77] I Agile organisationer vedrører Knowledge Management, ifølge Dove [77], tre aspekter: *læring*, *anvendelse* og *formål*, i netop denne rækkefølge, eftersom denne er altafgørende for at viden kan implementeres og skabe værdi og udbytte. Netop innovation er resultatet af de ændringer, der opstår som positiv følge af implementering af ny viden. [77]

Det specielle ved den agile tilgang er samarbejdet, mellem kunden og leverandøren, hvorfor der er både mulighed samt et grundlæggende behov for, at viden deles på tværs af organisationerne. Kunden har behov for

udviklingsteamets ekspertise vedrørende selve softwareudvikling, men også deres kreative anskuelse af løsningsmuligheder og håndtering af den teknologiske viden. Ligeledes er udviklerne afhængige af kunderepræsentantens interne viden om dennes organisation og grundlæggende behov. Den innovative løsning, kunden efterspørger, dannes således i sammenspil mellem disse roller, på baggrund af deres vidensdeling og fælles idégenerering.

5.4 DELKONKLUSION

Ud fra et softwareudviklings- og et erhvervsøkonomisk perspektiv, kan følgende principper udledes som forudsætninger for succesfuldt anvendelse af en agil udviklingsmetode.

Det er således en forudsætning, at:

- der opstilles klare mål og krav for projektet,
- kunden og leverandøren opnår et fælles ejerskab for projektets succes/fiasko,
- brugerne af softwaren inddrages i udviklingsprocessen gennem testning af produktet,
- der anvendes en Product Backlog frem for en kravsspecifikation, hvilken håndteres vha. en struktureret Requirements Definition and Management-proces,
- det er muligt at foretage ændringer/tilpasninger igennem projektets forløb,
- der opstilles klare kontrolforanstaltninger for projektændringer,
- teamet, herunder både udviklere og kunde, interagere ansigt-til-ansigt og er co-located,
- fokus ligger på working software frem for omfattende dokumentation,
- begge organisationer er moden til at håndter et agile udviklingsparadigme, og at leverandøren anser projektet som værende en central kerne i virksomheden,
- udviklingen sker i et selvorganiseret team, og at der gives autonomi til at medlemmerne kan udnytte deres kompetencer til fulde, samt til at foretage Risk Management-processen hurtigt og effektivt,
- der implementeres en langsigtet og proaktiv Risk Management-strategi, der skaber mulighed for at udføre enhver passende respons på en risiko,
- der integreres et Knowledge Management-fokus, samt
- det er muligt at afslutte samarbejdet efter enhver iteration, såfremt kunden finder det udviklede produkt endeligt acceptabelt tidligere end forventet.

DEL II

6 AGIL SOFTWAREUDVIKLING I DEN OFFENTLIGE SEKTOR

Ovenstående findings anses som værende forudsætningerne for gennemførelse af et optimalt agilt softwareudviklingsprojekt. Disse teoretiske betragtninger tager dog på intet tidspunkt højde for de juridiske rammer, herunder udbudsretslige begrænsninger, hvilke regulere forholdet mellem parterne.

Hvor Contract Manageren repræsenterer ordregiver, er det vedkommendes altovervejende ansvar at varetage ordregivers interesser. Contract Manageren har i forbindelse med udlicitering og gennemførelsen af et agilt udviklingsprojekt adskillige hensyn at tage, herunder overholdelse af udbudsretten; hensigten med indgåelse af samarbejdet og struktureringen heraf; samt de bagvedliggende forudsætninger for gennemførelse af et agilt udviklingsprojekt, for at fremme projektets overordnede muligheder for succes.

For at analysere hvorledes de ovenfor udledte forudsætninger bør adresseres kontraktuelt, med udgangspunkts i udbudsretten, inddeles disse i 10 hovedpunkter, på baggrund af deres respektive bagvedlæggende hensyn.

6.1 UDBUDSRETTEN

Hvor der fortages anskaffelser i den offentlige sektor, er ordregiver underlagt EU's generelle principper om ligebehandling, gennemsigtighed og proportionalitet samt procedureregler mv. i Udbudsdirektivet der er blevet implementeret i dansk ret gennem Udbudsloven.

Grundlæggende skal den offentlige ordregiver, ved anskaffelser over Udbudslovens fastsatte tærskelværdier, anvende en af de i lovens § 55 fremsatte udbudsprocedure, og følge særskilte regler for udvælgelse af leverandører, tilbudsvurdering og kontrakttildeling mv.. Ligeledes skal den ordregivende myndighed sondre mellem indkøb med og uden grænseoverskridende interesse.

Særligt essentielt for nærværende afhandling er ordregivers forpligtelse i forbindelse med udarbejdelse af tekniske specifikationer, herunder mulighederne for at fortage ændring i kontraktens løbetid, grundet det ændringsbehov, et agilt udviklingsprojekt, som det fremgår i Del I, påkræver. Desuden skabes der med den nye Udbudslov mulighed for anvendelse af en ny udbudsprocedure for *innovationspartnerskaber*, jf. §§ 73-79, hvilken muliggør et længerevarende tæt samarbejde med en leverandør om nyudvikling og videreudvikling af innovative produkter og services.

Det skal bemærkes, at udbudsretten alene regulerer forløbet frem til kontraktindgåelse, og dermed ikke det efterfølgende aftaleforhold, der udspringer af en afsluttet kontraktindgåelse. [78] Dog medfører principperne om ligebehandling og gennemsigtighed at ordregiver ikke efterfølgende kan foretage ændringer i et udbudssat aftalegrundlag, jf. § 24, stk. 1, nr. 37.

FREMGANGSMETODE

Ovenstående forudsætninger for anvendelse af et agilt udviklingsparadigme, kan overordnet set have indflydelse på fire forskellige udbudsretslige områder, herunder udbudsproceduren, udbudsbekendtgørelsen, udbudsmaterialet samt selve kontrakten. Således vil disse 10 forudsætninger analyseres i henhold til den del af udbudsretten, der er mest relevant i forhold til eventuelle problemstillinger og potentielle værdiskabende muligheder.

UDBUDSPROCEDUREN

Udbudsloven opstiller i kapitel 7 en udtømmende liste af udbudsprocedurer, der kan anvendes i forbindelse med udlicitering af offentlige opgaver.

Vedrørende proceduren *innovationspartnerskaber*, i henhold til §§ 73-79, skal ordregiver sondringen mellem hvorvidt udviklingsprojekt opfattes som indkøb af en tjenesteydelse eller udvikling af en vare, hvilket har konsekvenser for bl.a. fastsættelse af kontaktens genstand og anvendelsesområde samt leverandørens juridiske forpligtelse, herunder henholdsvis obligation de moyens og obligation de résultat[27].

Dertil kan det på baggrund af denne sondring diskuteres, hvorvidt en partnerskabskontrakt, jf. §§ 73-79, faktisk omfattes af Udbudslovens anvendelsesområde i henhold til kapitel 2, såfremt kontraktens genstand opfattes en vare,[28] idet en offentlig vareindkøbskontrakt, jf. § 24, stk. 1, nr. 26, defineres som offentlige kontrakter, der vedrører *køb*, *leasing* eller *leje* af varer, og uden at inddrage et udviklingsmæssigt perspektivet. Hvorledes kontraktens genstand bør defineres i nærværende situation, behandles i punkt 3, "Anvendelse af en Product Backlog".

UDBUDSBEKENDTGØRELSEN

Udbudsbekendtgørelsen er obligatorisk efter Udbudslovens § 128, og har til formål at skabe grundlag for potentielle leverandørers vurdering af, om opgaven er relevant at byde på. Oplysninger og krav, fremsat i udbudsbekendtgørelsen er bindene, og skal stemme overens med udbudsmaterialet, der som udgangspunkt er et supplement til udbudsbekendtgørelsen.

UDBUDSMATERIALE

Udbudsmaterialet skal indeholde informationer om blandt andet krav til tilbudsgiver i forhold til både udvælgelses- og tildelingskriterier. Der er efter den nye Udbudslov pligt til at beskrive evalueringsmetoden[29] i udbudsmaterialet, hvilken, som endnu en ændring, nu skal offentliggøres samtidigt med udbudsbekendtgørelsen, jf. § 132.

Tildelingen at kontrakten skal ske på baggrund af det økonomisk mest fordelagtige tilbud, efter tildelingskriterierne i § 162, herunder pris, omkostninger, eller bedste forhold mellem pris og kvalitet. Sidstnævnte skaber mulighed for at tildele kontakten til en leverandør på grundlag af blandt andet underkriterier som kvalitative, miljømæssige og sociale aspekter. Denne ændring åbner op for muligheden for at medtage innovationrelevante karakteristika i betragtningen, såfremt et sådan underkriterie er direkte forbundet med kontraktens genstand, i henhold til § 163, dog under hensyn til § 164.

Evaluering på baggrund af sådanne underkriterier, hvilke jf. Udbudslovens § 162, Stk. 4, kan danne det fulde evalueringsgrundlag, er særligt relevant i nærværende afhandling, eftersom *kreativitet*, *effektivitet* samt *arbejdsindsats*, ifølge Opelt et al. [6], danner det bedste grundlag for evalueringen af produktets værdi samt det potentielt bedste match for en effektivt og succesfuld udviklingsproces.

UDBUDSKONTRAKTEN

[27] Denne sondring vedrøre bl.a. fordeling af ansvar for kontraktbrud, herunder bevisbyrden i erstatningsansvar. [99]
[28] Som eksempel herpå ses Digitaliseringstyrelsens udbudsbekendtgørelse, vedrørende "Etablering af partnerskab med henblik på tilvejebringelse af en ny national digital identitets- og signaturløsning til afløsning af det eksisterende NemID", s. 8, hvor der argumenteres for at "Etableringen af partnerskabet er ikke en gensidig aftale om anskaffelse af varer eller tjenesteydelser og er derfor ikke omfattet af Udbudsdirektivet." Det skal bemærkes at proceduren innovationspartnerskab ikke anvendes i nærværende udbud, men derimod begrænset udbud, herunder udbud efter forhandling, konkurrencepræget dialog. Se nærmere Bilag 1 - Udbudsbekendtgørelse 118-215097.
[29] Evalueringsmetoden er ifølge udkastet til lovbemærkningerne, "den systematik, som ordregiveren benytter i evalueringen af tilbuddene med henblik på at identificere det økonomisk mest fordelagtige tilbud."

Selve den offentlige kontrakt er en del af udbudsmaterielet, og skal derfor offentliggøres sammen med udbudsbekendtgørelsen. Udbudslovens kapitel 15 regulerer kontraktens gennemførsel, herunder for anvendelsen af underleverandører, ændringer af kontrakten samt ophør heraf.

Udbudsloven regulerer ikke direkte kontraktvilkårene, men disse skal som resten af udbudsmaterialet følge de udbudsretlige principper om ligebehandling, gennemsigtighed samt proportionalitet i forhold til kontraktens genstand. Der må desuden ikke forhandles om de grundlæggende elementer i kontrakten.

6.2 JURIDISK ANALYSE AF FORUDSÆTNINGERNE FOR GENNEMFØRSEL AF AGILE SOFTWAREUDVIKLING

1. KLARE MÅL OG KRAV FOR PROJEKTET

Denne forudsætning stemmer fuldt overens med formålet med gennemførsel af udbud og de udbudsretslige principper. Definering af klare og udspecificerede krav danner et gennemsigtigt grundlag aftaleindgåelsen.

2. ET FÆLLES EJERSKAB FOR PROJEKTETS SUCCES/FIASKO SAMT BRUGERINDDRAGELSE

Det fælles ejerskab og den fælles deltagelse i projektudviklingen er en fundamental forudsætning for at opnå det fulde potentiale ved den agile udviklingen. Et fælles ejerskab kan både struktureres i kontrakten og i selve udbudsproceduren. Derudover kan det fælles ejerskab motiveres gennem relationel kontraktteori, hvilket er specielt relevant i overensstemmelse med Det Agile Manifest og princippet om ”Kundesamarbejde frem for kontraktforhandling”.

Udbudsloven § 73-79 skaber mulighed for at en ordregivende myndighed kan indgå i et offentlig-private innovationspartnerskab, (OPI), der, i modsætning til andre procedurer, ikke indgås som et aftager-leverandørforhold, men derimod, hvilket navnet antyder, som et udviklingspartnerskab.

Denne proceduren opdeles i tre faser: udbuddet, innovationsforløbet og et eventuelt efterfølgende indkøb af den udviklede ydelse, og er kendetegnet ved at skabe åbenhed, gensidig tillid og en høj grad af brugerinddragelse, idet den private part opnår adgang til ordregivers viden om brugerne og adgang til at teste løsningen i praksis undervejs i udviklingsforløbet. [79] Udbudsproceduren, innovationspartnerskaber, stemmer derfor fuldt overens med nærværende forudsætning.

Ved anvendelse af denne procedure udbudssættes der ikke en på forhånd kendt vare eller ydelse, med derimod en problemstilling og et behov, hvorfor kravene til selve løsningen på indgåelsestidspunktet er ukendte og kravsspecifikationen derfor kan anses som værende ”åben”[30]. [80]

Bag udgangspunktet for nærværende forudsætning ligger desuden et behov for at sikre, at det er muligt at opnå et godt partnerskab, idet begge parter er afhængige af hinandens indsats. Til vurderingen af tilbudsgiverne på et kvalificeret grundlag, kan der i vurderingsfasen inddrages såkaldte *code camps,* hvorved ordregiver kan danne sig et indtryk af det kommende samarbejde. [81] [82] [83] Såfremt der ikke udbydes en traditionel kravsspecifikation, er det ikke muligt for tilbudsgiverne at byde ind med et endeligt løsningsforslag. Ved anvendelse af code camps kan udvalgte tilbudsgiver i stedet vurderes ud fra deres håndtering af en fiktiv opgave. Herved er det muligt, at lade udviklingsteamets IT-kompetencer, samarbejdsevner og modenhed i forhold til den udvalgte opgave og eksempelvis den agile udviklingsmetode, samt selve matchet mellem ordregiver og leverandør indgå i tilbudsvurderingen som et underkriterie.

Der er således talte om en yderst fleksibel og meget passende udbudsprocedure i forhold til behovet for agil udvikling. Der er dog kun tale om et fuldt fælles ejerskab, såfremt partner optræder som økonomisk ligevær-

[30] Potentielle udfordringer ved anvendelse af en ”åben” kravsspecifikation behandles nedenfor.

dige frem for situationen, hvor den offentlige part delvist betaler honorar for den private parts deltagelse i udviklingspartnerskabet. Det skal bemærkes at Udbudsloven ikke tager stilling til denne sondring og betydningen at parternes indbyrdes investering af ressourcer i samarbejde. Som eksempel på hvorfor denne sondring er relevant, henvises der til de fem forskellige OPI modelaftaler, udarbejdet af Rønne & Lundgren Advokatfirma og Kammeradvokaten for OPI-lab[31].

SHARED PAIN/GAIN[32]

Etablering af et innovationspartnerskab samt indsætningen af mekanismer, regler og strukturer i kontrakten, der fremmer opbygningen af et samarbejde, gennemsigtighed og tillid vil assistere parterne i at opnå succes med samarbejdet og et fælles ejerskab.

En bred vifte af evidensbaseret forskning [84] [85] [86] [87] indikerer dog, at anvendelsen af positive og negative incitament (sanktioner) relateret til performance, samt bonusser er direkte skadeligt for et kontraktuelt forhold. Disse studier påviser at anvendelsen af sådanne incitamenter fører til øget gaming, en reduktion i gennemsigtighed og kvalitet, en svækkelse af systemet samt andre dysfunktioner. [88] Endvidere fremmer incitamenter og sanktioner en konkurrencepræget kultur mellem kontrahenterne frem for at fostre et samarbejde.

Som alternativ hertil foreslår Arbogast et al. [88] at parterne deler risici og belønninger mellem sig gennem en *shared pain/gain* model, og at risici allokeres til den part, der er nærmest til at håndtere denne. Dette betyder eksempelvis, at risici vedrørende projektets krav gerne placeres hos ordregiver, mens implementerings- og tekniske risici placeres i hænderne på leverandøren.

Denne tilgang vil i bedste fald skabe en øget harmonisering af motivationer for parterne, idet begge både har noget at vinde og at tabe. [88] Det er herved desuden muligt at forbedre den opfattede retfærdighed mellem parterne og styrke deres relation og samarbejdsevne. Denne filosofi er ifølge Arbogast et al. [88] kernen til en win-win samarbejde, og den forøget tillid vil kunne fremme yderligere kommercielle gevinster. Det skal dog bemærkes, at denne tilgang kan misbruges til at skyde skylden på den modsatte part og i stedet fokusere på egennytte, såfremt begge parter ikke motiveres af værdien af et langsigtet fremtidigt samarbejde, som den relationelle kontraktteori, hvilken behandles nedenfor, beskriver.

I denne forbindelse kan der refereres til anvendelsen af en *target-cost* model [88], hvor projektets samlede omfang og detaljer så godt som muligt indledende identificeres, i fællesskab, med det formål at opstille det, der kaldes *original target-cost*. Den endelige forskel mellem de estimerede target-cost og de faktiske omkostninger deles i sidste ende mellem parterne, uanset om der er blevet under- eller overestimeret. I denne model er der desuden fastlagt mekanismer for ændringer igennem projektforløbet.

LID TIL RELATIONEL KONTRAKTTEORI

Såfremt ordregiver ikke anvender udbudsformen innovationspartnerskab, hvor begge parter stilles økonomisk lige, men hvor ordregiver betaler honorar for leverandørens deltagelse i udviklingspartnerskabet, eller en alternativ udbudsprocedure så som offentligt eller begrænset udbud, vil det være nødvendigt at motivere paterne til et fælles ejerskab gennem relationel kontraktteori[33]. Dette skyldes den ”åbne kravsspecifikation”,

[31] OPI-Lab og Living Lab Denmark står bag de fem OPI-modelaftaler, der er udarbejdet af Rønne & Lundgren Advokatfirma i samarbejde med advokatfirmaet Poul Schmith / Kammeradvokaten og OPI-Labs juridiske arbejdsgruppe bestående af repræsentanter fra Syddansk Universitet og Region Syddanmark, http://opiguide.dk/vaerktoejer/.
[32] En shared pain/gain model kaldes også *shared risk/reward system*, og handler om hvorledes risici og gevinst deles mellem aftaleparterne.
[33] Relationelle kontraktforhold vedr. bl.a. muligheden for at opbygge og opretholde en udbytterigt samarbejde, grundfæstnet i normer, tillid og gensidighed frem for omfattende dokumentation og reguleringer. For yderlig information om

hvilket er kendetegnet ved situationen, hvor kontraktens genstand kan karakterers som værende kompleks, umuligt at beskrive samt besværligt at udvikle, såsom innovativ softwareudvikling. [89] Der er her tale om en *ufuldstændig kontrakt*, hvor det hverken er muligt eller fordelagtigt at udspecificere præcis, hvad der skal udvikles.

Kontraker for komplekse anskaffelser er af nødvendighed ufuldstændige, og anvendes hvor det er svært at specificere alle udvekslingsreglerne for transaktionen, og hvor hverken ordregiver eller leverandør kan forudsige alle de scenarier, som potentielt kan påvirke de samlede omkostninger og produktets udvikling. [90] Af samme grund kan komplekse kontraktforhold plages af incitamentsstruktur, der som udgangspunkt støtter egennytte og ikke-samarbejdsvillig adfærd.

Formålet med at investere i et relationelle forhold, er i denne forbindelse, at motivere parterne til at agere i modpartens interesse for at styrke samarbejdet, og på den måde at værne om værdien af et langsigtet samarbejde. Dog modstrider dette hensyn, forudsætningen, vedrørende agile udvikling, om at projektet skal kunne afsluttes efter enhver iteration, såfremt parterne finder at produktet tidligere end først antaget, i tilstrækkelig grad, opfylder ordregivers behov.[34]

Således menes det ikke at være hensigtsmæssigt at anvende enhver anden udbudsprocedure end innovationspartnerskab med økonomisk ligestilles af parterne, idet opretholdelsen af det fælles ejerskab, den agile udviklingsmetodologi forudsætter, langt hen ad vejen vil bero på et uhensigtsmæssigt relationelt grundlag.

3. ANVENDELSE AF EN PRODUCT BACKLOG

Denne forudsætning vedrører det vigtigste agile værktøj, og skaber selve den fleksibilitet, der danner grundlag til foretagelsen af *ændringer i medfør kontrakten*, hvilket beskrives under punkt 4, "Ændringsmuligheder og kontrolforanstaltninger for projektændringer".

Hvor der traditionelt set anvendes en kravsspecifikation, leverandøren skal opfylde uden inddragelse af ordregiver, udvikles Product Backlog'en i et agilt projekt derimod løbende i fællesskab, hvorfor kontraktens genstand konstant er i forandring. Dette er såfremt det udbudte projekt udbydes som en vare frem for en tjeneste ydelse. I så fald vil den endelige og altomfattende Product Backlog reelt set være, hvad der svare til en kravsspecifikation, hvilken ordregiver selv sagt ikke har overblik over ved projektets begyndelse. Dette udgør et problem i udbudsretlig forstand, eftersom der, jf. Udbudsloven § 178, ikke må foretages ændringer af kontraktens grundlæggende elementer.[35] Udbydes projektet derimod som en tjenesteydelse, er det ud over kontraktens grundlæggende elementer, herunder udvælgelses- og tildelingskriterier, simplificeret set selve behovene, der skal opfyldes. Disse kan eksempelvis opgøres i "epics" og være med til at udgøre kontraktens genstand. Til støtte for denne opfattelse, argumenterer Opelt et al. [6], som det fremgik af afsnittet "Agil softwareudvikling fra et erhvervsøkonomisk perspektiv", for at værdien i det agile udviklingsprojekt består af selve udviklingen frem for produktet.

Anvendelsen af en Product Backlog og sondringen mellem at udbyde en vare eller en tjenesteydelse er således essentiel i forhold til både udbudsbekendtgørelsen og -materielet, og skaber grundlaget for i hvilken grad og på hvilken måde ændringer skal foretages.

ACCEPT

---

relationel kontraktteori, se da eksempelvis: Macneil, "*Relational contract: What we do and do not know*" [105] samt Barnet, *"Conflicting Visions: A Critique of Ian Macneil's Relational Theory of Contract"* [100]

[34] Yderligere analyse af forudsætningen om muligheden for tidlig ophør af kontraktforholdet behandles under punkt 10, "Projektets ophør".

[35] Ændringsmulighederne i denne henseende behandles under punkt 4, "Ændringsmuligheder og kontrolforanstaltninger for projektændringer".

Anvendelsen af en Product Backlog frem for en traditionel kravsspecifikation bevirker tillige, at proceduren vedrørende accept skal håndteres anderledes. Eftersom kravene til det endelige produkt kun delvist eller slet ikke er defineret og uddybet i detaljer inden påbegyndelse af projektet, er det essentielt at kontrakten tager stilling til og regulerer hvorledes og på hvilket grundlag, det samlede projekt og delleverancerne accepteres. Desuden skal denne regulere proceduren for afhjælpning, såfremt produktet ikke kan accepteres, samt hvorledes tvister skal afgøres. Ikke mindst er det altafgørende, at opnå enighed om hvornår og hvordan kravene til den kommende iteration udvikles og hvem, der skal stå for denne proces. Klarhed om disse perspektiver i udviklingen af et juridisk rammeværktøj, vil assistere parterne i at opnå det fulde potentiale af deres samarbejde, frem for at skulle fokusere på uoverensstemmelser, misforståelser og skuffede forventninger, på baggrund af netop juridiske tvetydigheder. [88]

Efter hver iteration leveres en funktionsdygtig, og potentiel implementerbar tilvækst, hvilken løbende skal testes og accepteres. Dette kan forgå gennem en automatiseret proces, eftersom kriterierne for accepten, forinden iterationen er blevet udviklet i fællesskab. [88] Dette simplificerer accept-processen og minimerer risikoen for udviklingen af problemer vedrørende det, der i Scrum-regi [91] refereres til som *the definition of done* – hvad skal der til for at en delleverance er acceptabel og kan defineres som værende ”færdig”?

Kontrakten skal desuden regulerer, i hvilket omfang de relevante personer, herunder blandt andet brugerne af softwaren, skal deltage i accept-processen gennem eksempelvis *user acceptance tests*, for at identificere så mange problemer og muligheder, så tidligt som muligt.

DEFINITION OF DONE

Accepten af delleverancer afhænger af overensstemmelsen med en forud aftalt liste af kriterier, testet gennem eksempelvis user acceptance tests og nogle ikke-funktionelle test. Disse kriterier kan vedrøre kodning, integration, funktionalitet, performance, anvendelighed samt levering af den nødvendige dokumentation. [88] Opfyldelse af disse danner grundlag for beregningen af *velocity* (mængden af aftalte user stories, implememteret, testet og leveret), hvilken er afgørende for hvornår der er tale om misligholdelse.

Dog skal det bemærkes, at listen alene bør opstille minimumskrav frem for et statement-of-work, idet udviklingen bør bero på samarbejde, fælles informationsudveksling og idégenerering, som Det Agile Manifest prædiker, for ikke at demotivere teamarbejde. Denne opfattelse deles af Arbogast et al. [88], der argumentere for, at et sådan fokus vil føre til unødvendig opmærksomhed på forhandlinger og på hvorvidt der er overensstemmelse med *the quality plan*, frem for fokus på samarbejdet om at udvikle nyttigt software.

GARANTIER

En typisk problemstilling i såvel traditionelle som agile udviklingsprojekter vedrører horvidt slutproduktet efterlever de funktionelle krav det udvikles i henhold til. Eftersom den agile metode ikke opstiller specifikationer forud for projektets opstart, men disse udvikles inkrementelt i samarbejde med leverandøren, skal der tages tydeligt stilling til *hvornår* relevante garantiperioder skal starte. Enhver udviklingskontrakt bør i sagens natur udformes i overensstemmelse med den enkelte udviklingsopgave, og det kan derfor i nogle tilfælde være belejligt, at en garanti løber fra accepten af hver iterativ leverance. Dertil argumenterer Opelt et al. [6] dog for, at kontrakten mest hensigtsmæssigt kun bør indeholde én eksplicit og utvetydig startdato for garantiperioden.

Problemstillingerne vedrørende garantier formindskes dog af den iterative udviklingsproces, grundet løbende levering, testning og accept af fungerende software samt den fælles beslutningsproces, hvilket resultere i en formindsket risikoprofil. [88] Denne effekt forstærkes yderligere, såfremt der anvendes en automatiseret accept-testnings procedure, der formindsker risikoen.

4. ÆNDRINGSMULIGHEDER OG KONTROLFORANSTALTNINGER FOR PROJEKT-ÆNDRINGER

Der er her tale om ændringer i softwareudviklingsmæssigt forstand, hvilket vedrører, i hvor vidt en strækning den "kravsspecifikation", der udvikles på baggrund af, er endelig eller mulig at tilpasse undervejs i projektets livscyklus. Agil udvikling kræver, at der ikke fastsættes flere krav end de, der et abdolut nødvendige for at gennemføre første sprint og udvikle et fungerende stykke software. Det er således en forudsætning, at det er muligt at udvikle, herunder udvide og slette, kravene til softwaren løbende.

Ændringsforetagelse i projektets løbetid vedrøre udbudsmaterialet, herunder kontrakten, og mulighederne herfor reguleres i Udbudslovens §§ 178-184. Det altovervejende udgangspunkt er at der ikke må foretages ændringer i det udbudssatte, herunder kontraktens grundlæggende elementer, kontraktens overordnede karakter og dens anvendelsesområde, jf. § 178, idet sådanne ændringer fordrejer grundlaget for tildelingen af kontrakten, hvilket kræver en ny udbudsprocedure.

Alt afhængigt at den valgt udbudsprocedure, fortages ændringer på forskelligt grundlag. Såfremt den ordregivende myndighed vælger at anvende proceduren, innovationspartnerskab, i henhold til § 73-79, anvendes der, som omtalt, en ufuldstændig og "åben kravsspecifikation". Dette bevirker, at det til forskel fra de andre udbudsprocedure, ikke er en traditionel kravsspecifikation, der foretages ændringer i og på baggrund af. Dertil danner sondringen mellem om det er en vare eller en tjenesteydelse der udbydes, som nævnt, forskellige krav til udspecificering af kontraktens genstand.

Dette alternative aftalegrundlaget bevirker, at ikke alle af de efterfølgende omtalte regler vedrørende ændringsforetagelse er lige simple at fortolke i henhold til et anvendelsen af et innovationspartnerskab. Som udgangspunkt skaber Product Backlog'en i denne henseende blot en overordnet ramme for projektet, hvilken kan tilpasses undervejs. [80] Det gøres dog i Udbudsloven ikke klart, på hvilken måde en sådan ramme kan eller skal opstilles; i hvilket omfang tildeling af en partnerskabskontrakt kræver en traditionel kravsspecifikation; samt hvilke konsekvenser en ufuldstændig kravsspecifikation har i forhold til ændringshåndtering i overensstemmelse med Udbudsloven.

Af denne grund, samt det faktum, at dette er en ny procedure, og ikke nødvendigvis den hellige gral, analyseres de generelle regler vedrørende ændringer herefter i forhold til nærværende forudsætningen.

Der skelnes i forbindelse med ændringer, mellem *væsentlige ændringer af grundlæggende elementer*, jf. § 178, stk. 2, og *ændringer i medfør af kontrakten*, jf. § 179.

I henhold til førstnævnte, betragtes en ændring, eller værdien og karakteren af flere kumulerede ændringer i samme kontrakt, for værende væsentlig, blandt andet, såfremt den er afgørende forskellig fra bestemmelserne i den oprindelige aftale, hvilket afspejler, at det var parternes hensigt at genforhandle aftalens grundlæggende, jf. Pressetext-sagen, præmis 34. [92] Dog betragtes en ændring ikke for væsentlig, såfremt begge de i Udbudslovens § 180 opremsede forudsætninger er opfyldte, dog på betingelse af, at denne/disse ikke ændre kontraktens overordnede karakter, jf. § 180, stk. 3. Desuden når de kumulative betingelser i § 183 opfyldes, hvilken giver adgang til at fortage uforudsete ændringer for op mod 50 % at den oprindelige kontraktværdi per ændring.

Ved medtagelse af et hensyn til ordregivers påpasselighed i § 183, stk. 1, nr. 1, skabes der med den nye Udbudslov en mindre restriktiv adgang til at foretage ændringer i kontraktens løbetid. Denne adgang er særligt relevant i forbindelse med agil udvikling, især såfremt kontraktens genstand karakteriseres som en vare, idet ordregiver af sagens natur ikke er i stand til at forudse visse omstændigheder.

Dog bør den påpasselig ordregiver tage hensyn til det faktum, at der i et agilt udviklingsforløb forventes et behov for ændringer, selvom detaljerne på indgåelsestidspunktet ikke kan forudses. I denne forbindelse skaber § 179, vedrørende ændringer *i medfør kontrakten*, mulighed for, at ordregiver kan udbudssætte potentiel-

le ændringsbehov og -foranstaltninger gennem klare, præcise og entydige klausuler. Således gennemsigtiggør ordregiver allerede på tidpunktet for udbudssætningen, i hvilket omfang det forventes, at der vil blive fortaget ændringer og arten af sådanne ændringer.

K03[36] [93] opdeler kravene, herunder både funktionelle og non-funktionelle krav, i absolutte og øvrige krav. Begrebet absolutte krav dækker her over krav, der er uundværlige for opfyldelsen af ordregivers forretningsmæssige mål og behov. Alt afhængigt af det valgt kontraktparadigme, kan de øvrige krav struktureres som det, der i Scrum-regi kaldes epics, og afspejle ethvert tænkeligt overordnet behov, en ordregive må kunne forestille sig. Disse behov, samt ændringsforanstaltningerne herfor, kan i overensstemmelse med § 179, udbudssættes gennem klare procedurer for "Agile tilpasninger"[37]. Herved kan disse behov ændres (i medfør kontrakten) i forhold til prioritering og udskiftes, uden at der kræves en nye udbudsprocedure, eftersom tilføjelse af nye krav forudsætter, at krav af samme omfang slettes. [93]

Sådanne forventede ændringsbehov skal opfattes som optioner og, jf. § 30 medregnes ved beregning af den oprindelige kontraktens værdi. Disse kan som eksempel fylde hele 80% af den samlede kontrakt, og agere som et slags rammelement, såfremt de uundværlige (absolutte) krav alene udgør 20%. Dertil skaber § 183, som nævnt, mulighed for, at der kan foretages yderligere ændringer for op til 50 % af den samlede kontraktværdi, inklusiv optioner, per ændring, såfremt ændringen ikke vedrøre kontraktens karakter, hvilket gør kontrakten yderst fleksibel.

Det kan dog diskuteres i hvilket økonomisk omfang, kontrakten bør kunne opstille så fleksible ændringsmuligheder, idet § 179 alene kræver, at der ikke må kunne foretages ændringer på kontraktens overordnede karakter. Tillige bør ordregive have in mente, ændringer ikke må fordreje aftalens økonomiske balance til fordel for leverandøren på en måde, som ikke var fastsat i de oprindelige udbudsbetingelser, jf. Pressetextsagen, præmis 37 [92].

En ændring i de grundlæggende elementer anses i § 178 som værende væsentlig, såfremt kontraktens karakter ændres, eller der sker betydelig udvidelse af anvendelsesområdet, hvorfor det er essentielt for parterne at opnå klarhed over, hvad disse begreber dækker over. Hverken Udbudsloven eller Udbudsdirektivet giver dog en klar definition af "kontraktens karakter" eller "kontraktens anvendelsesområde", mens den engelske anvendte terminologi "the overall nature of the contract", ligeledes ikke er særlig forklarende. Formålet med at lade definitionen af disse bero på en analyse må være, at overlade den ordregivende myndighed et skøn i henhold til det enkelte indkøb og valgte procedure.

Ud fra Udbudslovens anvendelse af begrebet "kontrakten karakter" og "anvendelsesområde", kan der argumenteres for, at disse har tilknytning til begrebet "kontraktens genstand", der ligeledes ikke udbydes fyldestgørende. Det kan dog af Udbudsloven § 41, stk. 1, nr. 1 udledes, at de tekniske specifikationer, hvilke jf. § 40, fastsætter de egenskaber, som kræves af den udbudte tjeneste eller vare, danner grundlaget for identificeringen af kontraktens genstand. Ligeledes bør kontraktens genstand, jf. Udbudsdirektivets artikel 29, anføres

[36] Der tages i K03 ikke stilling til sondringen mellem udbydelse af en vare eller en tjenesteydelse, idet denne standartkontrakt udelukkende skal fungere som en skabelon og derfor tilpasses det enkelte udbuds behov.

[37] Den grundlæggende ide bag den agile udviklingsmetode tager afsæt i en konstant re-prioritering af Product Backlog elementerne og en adaptive og iterativ planlægningsproces. Derfor er det yderst essentielt, at kontrakten ikke opbygges og formuleres, samt håndterer Change Management på en traditionel vis, idet dette vil skade fleksibiliteten i den agile metodologi. [88] Kontrakten skal adressere og gennemsigtiggør ændringsforanstaltningerne, jf. § 179, og dermed omstændighederne for et fleksibelt projektet-scope. Dog bør kontrakten undgå, at opstille obligatorisk krav til change management, ændringsanmodninger og særlige komplekse ændringsprocesser, men i stedet opstille retningslinjer, der kan fungere som et værktøj til at forsimple ændringsproceduren, så ændringer kan ske ofte og i samarbejde mellem ordregiver og leverandør. [88]

ved en beskrivelse af ordregivers behov og de karakteristika, der kræves af de varer, bygge- og anlægsarbejder eller tjenesteydelser, der skal indkøbes, og en fastlæggelse af kontrakttildelingskriterierne.

Således menes kontraktens karakter og anvendelsesområde til en vis udstrækning at vedrøre kontraktens genstand, herunder *de egenskaber som kræves af den udbudte tjeneste eller vare samt beskrivelsen af ordregivers behov og de karakteristika, der kræves af indkøbet samt en fastlæggelse af kontrakttildelingskriterierne,* i det omfang disse oplysninger har være genstand for udbudssætning.

PRIS VERSUS SCOPE

Det er essentielt at afklare hvad, der netop forstås ved begreberne "kontraktens karakter", "genstand" og "anvendelsesområde", idet den fornødne fleksibilitet i en kontrakt, der regulere et agilt udviklingsprojekt, opnås ved at lade eet af to parametre være variabel: pris versus scope[38]. [6]

Som eksempler på velansete agile kontraktparadigmer kan der i denne forbindelse nævnes: *flexible-scope progressive* kontrakter, der indeholder stor fleksibilitet vedr. ændringer i scopet*; target-cost* kontraker, der bygger på et shared pain/gain aspekt, og overordnet definerer et fleksibelt scope for projektet; *variable-price, variable-scope progressive* kontrakter, hvor scopet og prisen ikke er defineret længere frem end første iteration; og *time and material* kontrakter. [88] Desuden anvender *Agile fixed-price* kontrakter, under et fastlagt budget, Product Bakclog'en som scope, hvilken er fleksibel. [6]

Uanset det valgte kontraktparadigme, hvilke grundet denne afhandlings begrænsede omfang ikke vil blive behandlet yderligere, er det i denne forbindelse vigtigt, utvetydigt, at tage stilling til hvorledes pris og scope fastlægges samt konsekvensen heraf. Eksempelvis i forhold til hvorledes ændringer kontrolleres og implementeres, samt i forhold til omkostningsstrukturen og ansvaret for disse ændringer. Opelt et al. [6] anbefaler her implementering af en *scope gonernance proces,* hvilken regulere netop disse perspektiver.

5. CO-LOCATION

Teorien om agil softwareudvikling forudsætter, at det er muligt udviklingen kan ske med udviklere og ordregiver på samme geografiske placering grundet værdien i et tæt daglige samarbejde. Denne forudsætning vedrører de grundlæggende udbudsretslige principper i henhold til § 2, herunder principperne om ligebehandling, gennemsigtighed og proportionalitet, eftersom et offentligt udbud, jf. § 2, stk. 2, ikke kunstigt må begrænse konkurrencen.

Problemstillingen vedrørende kravet om co-location påvirker princippet om ligebehandling, idet en forudsætning om at leverandøren er geografisk tæt placeret på ordregiver, påvirker, at udbuddet potentielt ikke har en grænseoverskridende interesse, eftersom det ikke er muligt for ellers potentielle udenlandsk leverandører at opfylde denne betingelse. Desuden kan det ligeledes afskære leverandører fra en anden landsdel fra at vinde opgaven.

Ordregiver vurderes ikke at kunne forsvare et så strengt krav i hensyn til proportionalitetsprincippet, idet Scrum-teorien argumenterer dog for, at det er muligt at anvende en cloud-løsningsudgave af en fysiske tavle, for at imødekomme vanskelighederne med at facilitere denne forudsætning, selvom udfaldet er mindre hensigtsmæssigt.

[38] I den internationale litteratur anvendes termen *scope,* der i Udbudsdirektivet er den direkte oversættelses af *anvendelsesområde*. Det skal dog bemærkes, at det ikke er muligt at ligge den ovenfor udbudsretslige, udledte definition af anvendelsesområde, til grunde for hvad der menes i generel forstand og dermed i disse efterfølgende kontrakter. Det er ligeledes ikke muligt at adoptere en hvilken som helst definition af begrebet, anvendt uden for Udbudsdirektivet til analyse heraf. Begrebet *scope* anvendes derfor herefter, hvor der refereres til litteraturen uden for Udbudsdirektivets anvendelsesområde.

6. SELVORGANISEREDE TEAMS

Denne forudsætning reguleres ikke af Udbudsloven. Dog dækker udbudslovens anvendelsesområde kun over udbudte opgaver, med en værdi på over 998.019 kr. eksklusive moms for offentlige vareindkøbskontrakter og tjenesteydelseskontrakter, jf. § 6, stk. 1, nr. 2.[39] Herved forudsættes, at det udbudte udviklingsprojekter er af en vis størrelse, hvilket som tidligere nævnt, kan skabe problemer i forhold til, hvorvidt det er muligt, for et enkelt team, at administrere opgaven.

7. KNOWLEDGE MANAGEMENT-FOKUS

Behovet for integreret Knowledge Management vedrøre ikke direkte udbudsloven, men såfremt der dannes et partnerskab i forbindelse med udbudsproceduren innovationspartner, skaber dette mulighed for, at der kan opstå vidensdeling på tvært af samarbejdet.

Dertil skal det bemærkes, at en uhensigtsmæssig kontrakt kan forhindre muligheden for vidensdeling og organisatorisk læring, samt øge risici og minimere projektets chancer for succes. [88] Det er derfor afgørende at koncipisten både selv forstår at anvende system thinking samt at kontrakten udformes således, at den er anvendelig i daglig brug og tilskynder system thinking blandt udviklingsteamets medlemmer. Således vil kontrakten reflektere det komplekse system, den influerer frem for blot at være et produkt af juristens silomentalitet. [88]

8. WORKING SOFTWARE FREM FOR OMFATTENDE DOKUMENTATION

Denne forudsætning kræver, som tidligere nævnt, ikke, at projektet på intet tidspunkt anvender dokumentation, men at der fokuseres på produktet frem for dokumenter om produktet. Som Turk et al. [34] beskriver det, ligger der til grunde for dette argument, en antagelse om, at udvikling af omfattende, relativt komplet og ensartet dokumentation af softwaremodeller er kontraproduktiv.

Dog skaber kritikken af Cho [25] vedrørende afhængigheden af implicit viden, som resultat af reduktionen i dokumentationen, et problem i udbudsretligt regi, grundet den ringe overdragelighed implicitte viden medføre

Såfremt et udbuddet ikke indeholde et efterfølgende køb af det udviklede produkter, er der behov for, at produktet efterfølgende kan udbudssættes. Dette nødvendiggør en vis kompleksitet i de dokumenterede informationer om produktet. Dertil kræver Udbudsloven, at der skabes ligestilling mellem alle potentielle leverandører, hvorfor leverandøren, der har stået for udviklingen af produktet, ikke må stilles bedre end andre, hvis denne ønsker at byde på denne efterfølgende kontrakt og den videre drift, jf. ligebehandlingsprincippet. Således skal alle relevante informationer om produktet kunne dokumenteres og udbudssættes, hvilket skaber behov for et løbende fokus på dette perspektiv i projektets livscyklus.

Såfremt kontrakttildelingen ud over udviklingen af produktet, tillige omfatter efterfølgende køb heraf, er det ligeledes nødvendigt, at der fokuseres på denne form for dokumentation i forbindelse med udviklingen, idet ordregiver ikke må placere sig i en situation, hvor der unødvendigt kun er én leverandøre, der kan varetage opgaven, eksempelvis driften af produktet. Det skal være muligt at fortage transition fra en leverandør til en anden, uden at pålægge en potentiel leverandør unødvendige omkostninger i denne forbindelse, samt uden at fordrejer konkurrencesituationen til fordel for den nuværende leverandør, jf. ligebehandlingsprincippet.

9. RISK MANAGEMENT

Offentlige myndigheder behandler personfølsomme data, hvorfor kontrakten skal sikre, at persondataloven overholdes. Dette bevirker, at teamet kan ikke opnå fuldstændig autonomi i projektet, men skal følge en Le-

[39] For offentlige anskaffelser, hvis værdi overstiger 500.000 kr., men er mindre end de aktuelle tærskelværdier i Udbudsloven, er ordregiver underlangt Tilbudsloven, jf. dennes § 15 a.

gal Risk Management-strategi[40], der grundet den enkelte risikos natur, kan være mere omfattende, end hvad der kan håndteres på projektniveau.

Der er dog rig mulighed for at tildele teammedlemmerne en række beslutningsbeføjelser, uden det vedrører Udbudsloven, men kontrakten bør klart definere teammedlemmernes rollerne og deres beføjelsers rækkevide, samt deres ansvarsområder.

I denne forbindelse bør ansvaret for *projektforsinkelser* tillægges særskilt opmærksomhed, både vedrørende overskridelser af tidsfristen for det samlede projekt og vedrørende løbende frister for delleverancer i henhold til en fastlagt tidsplan. Eftersom agile udviklingsprojekter involverer både leverandørens udviklingsteam, en Scrum-master, samt en Product Owner, kan forsinkelser skylde begge kontraktparter, hvorfor kontrakten skal tage stilling til hvornår den enkelte part ifalder ansvar samt under hvilke omstændigheder, der kan fritages herfor.

Ved indgåelse af et traditionel aftager-leverandørforhold overlades alt ansvaret til leverandøren, idet han skal sørge for at opfylde samtlige betingelser i kravsspecifikationen. I modsætning hertil, skal ordregiver ved indgåelse af et partnerskab om agil udvikling, udvælge kontraktparadigme og udforme blandt andet ændringsklausluer således, at risikoen for at projektet ender med at blive markant dyrere end forventet formindskes, samt således, at ansvaret herfor fordeles mellem parterne, i overensstemmelse med overvejelserne i dette afsnits punkt 2-4.

Risikofordelingen og -delingen, i forbindelse med eksempelvis anvendelsen af en shared pain/gain model, kan ligeledes have indflydelse på parternes ansvar, og reducere selve sandsynligheden for at en problemstilling opstår, idet begge parter har et incitament til at efterse modpartens interesse, og herved afværge potentielle problemer.

Allokeres besparelser og yderligere omkostninger mellem parterne, skal dette ligeledes reguleres i kontrakten, og der skal som tidligere fremlagt, opstilles retningslinjer for hvorledes performance måles og værdiansættes, og hvordan ansvaret herfor fordeles.

10. PROJEKTETS OPHØR

Samarbejdet skal ideelt set kunne endes efter enhver iteration, eftersom ordregiver hypotetisk allerede efter første iteration har fået leveret et fungerende udgave af det ønskede software. [88] Første udgave opfylder i sagens natur kun de absolutte mest kritiske krav, men jo længere i forløbet, udviklingen befinder sig, desto større sandsynlighed er der for, at ordregiver modtager et endeligt fungerende og acceptabelt produkt.

Muligheden for opsigelse af kontrakten, før den oprindelige planlagte periode, er i høj grad præget af den fleksibilitet, der kontraktuelt skabes, jf. ovenstående punkt 3., idet de forventede behov, med tiden ændres og helt eller delvist kan vise sig overflødige.

Dette perspektiv skal klart defineres i udbudskontrakten for at skabe gennemsigtighed og ligebehandling mellem de potentielle leverandører. Hvis ikke dette sker, vil en sådan ændring i omstændighederne, kunne defineres som værende væsentlig, jf. Udbudsloven § 178, stk. 2, nr. 1, samt Pressetext-sagen, præmis 35. [92]

Det er i denne forbindelse desuden relevant at tage stilling til, hvorledes et sådan ophør af kontraktforholdet skal forløbe, samt hvilket konsekvenser ophøret har for eksempelvis bod og vederlaget, idet aftalens økonomiske balance ikke må ændres til fordel for leverandøren på en måde, som ikke var fastsat i de oprindelige udbudsbetingelser, jf. Udbudsloven § 178, stk. 2, nr. 2, samt Pressetext-sagen, præmis 37. [92]

[40] For yderligere information om Legal Risk Management, se da ”Proactive Management and Proactive Law” kapitel 5, af R. F. Henschel [72]

7 KONKLUSION

Denne afhandling har analyseret i hvilket omfang det er muligt og relevant at anvende en agile udviklingsmetode i forbindelse med udlicitering af et softwareudviklingsprojekt, når kontraktforholdet er underlagt Udbudsloven. Dertil er det blevet anskueliggjort, hvilke teoretiske forudsætninger ordregiver bør tage højde for, fra et softwareudviklings- og erhvervsøkonomisk perspektiv, samt hvilke relevante juridiske hensyn, kontrakten skal adressere, for at muliggøre succesfuldt anvendelsen af en agile udviklingsmetode i et udliciteret softwareprojekt.

Første del af afhandlingen søger at skabe klarhed over det teoretisk fundament, for anvendelsen af en agil udviklingsmetode, herunder at udlede hvilke forudsætninger den agile metode opstiller.

Den agile udviklingsmetode er, i modsætning til en plan-drevet tilgang, udviklet som en proaktiv løsning til håndtering af den teknologiske udvikling, der udfordrer både ordregiver og leverandør. Den tager udgangspunkt i et ændrings-drevet behov, og søger at skabe fleksibilitet igennem projektets livscyklus. Den agile metode anser det enkelte individs ekspertise og kompetencer som det største aktiv i imødekommelsen af disse udfordringer, og sætter lid til værdien af samarbejde og hurtig levering og testning af fungerende software.

Som denne afhandling tydeliggør, er den agile udviklingstilgang ikke anvendelig i alle udviklingsprojekter og -miljøer, og den ene udviklingsmetode kan ikke overordnet karakterisere som værende bedre end den anden, men derimod bedst anvendelig i forskellige situationer.

Det kan dog konkluderes, at den agile metode er den bedst egnede til håndtering af uforudsigelige og omskiftelige krav i forbindelse med udviklingen af software, såfremt leverandørens organisation er moden hertil og formår at påtage sig en projektorienteret organisationsstruktur. Dette kræver at det strategiske niveau og det taktiske og operationelle niveau forenes, hvilket påvirker både ledelsesfilosofien, arbejdstilgangen og arbejdsmetoderne.

Ligeledes kræves det, at ordregiver er moden i et agilt henseende, eftersom den agile metode forudsætter, at der mellem ordregiver og leverandør er et fælles ejerskab for projektets succes/fiasko, idet projektets chancer for succes beror på begge parters indsats, samarbejdsevne og -vilje.

Inden for agile softwareudvikling, er Scrum i dag det mest anvendte agile management framework, og opstiller nogle værktøjer og forudsætninger for gennemførsel af et agil udviklingsprojekt. I Scrum sker udviklingen inkrementelt og i iterationer med udgangspunkt i en Product Backlog frem for en på-forhånd udspecificeret kravsspecifikation. Denne Backlog indeholder indledende kun informationer om de absolut essentielle krav og behov, angivet i såkaldte user stories. Den tilpasses konstant i projektforløbet, i takt med at ordregiver opnår en større indsigt i sit egentlige behov og de teknologiske muligheder. Ved projektets afslutning indeholder Project Backlog'en samtlige informationer om ændringer, fortaget undervejs, og om de endelige krav, produktet opfylder.

Anvendelse af en Product Backlog kræver et intens fokus på Requirements Definition and Management (RDM), idet forsømmelse heraf, gør udviklingsteamet ude af stand til at opfylde ordregivers behov og levere et potentielt implementerbart stykke software efter hver iteration. Formålet med den inkrementelle udvikling og løbende testning af softwaren er, at tillade så mange tilpasninger i udvikling, som det viser sig, ordregiver har behov for, og på samme tid at gøre det muligt for ordregiver at modtage et færdigudviklet produkt tidligere, end det ville være muligt ved anvendelse af en plan-drevet udviklingsmetode.

Udviklingen skal i henhold til Scrum-teorien ske i et mindre selvorganiseret team, hvilket bør omfatte en kunderepræsentant, kaldet en Product Owner. Vedkommende har ansvar for RDM-processen, og fungerer som ordregivers adgang og input til vidensdeling og fælles idégenerering på tværs af parterne, hvilket ifølge

Knowledge Management-teorien er altafgørende for innovativ udvikling. Af denne grund anbefaler Scrum-teorien, at leverandør og ordregiver er co-located, så det er muligt at drage fordel af en daglig kontakt.

Den agile udvikling kræver desuden at risici både håndteres og opfattes anderledes, end det er tilfældet i traditionelle, plan-drevne udviklingsprojekter. Denne udviklingstilgang søger at inddrage en langsigtet, proaktiv Risk Management-strategi, og gøre udviklingsteamet i stand til, hurtigt at spotte og reagere på potentielle muligheder, således at det er muligt at udnytte potentialet i en given risiko. Af denne grund forudsætter den agile tilgang, at det selvorganiserede udviklingsteam tildeles den fornødne autonomi og beslutningsmandat til selv at kunne foretage Risk Management-proceduren hurtigt og effektivt. Dette skyldes en overbevisning om, at teamet, såfremt organisationen formår at integrere projektet som en centrale kerne i organisationen, er nærmest til at identificere og prioritere risici, samt at udvikle den nødvendige respons-strategi hertil. Til grunde for denne decentraliseret Risk Management-strategi ligger desuden det økonomiske hensyn, at jo senere i projektets livscyklus en given risiko eller fejl opdages, desto større er omkostningerne ved at rette op herpå, hvorfor målet er at disse identificeres så tidligt, som det er muligt.

Anden del af denne afhandling behandler de udbudsretslige og kontraktuelle hensyn, ordregiver skal tage ved udlicitering af et agilt softwareprojekt, på baggrund af de, i Del I, udledte forudsætninger.

Som denne afhandling tydeliggør, har ordregiver bedst mulighed for at anvende en agil udviklingsmetode, såfremt udbudsproceduren, innovationspartnerskab anvendes. Ved denne procedure inddragelses ordregiver i udviklingsprocessen, idet projektet oprettes som et partnerskab frem for en aftager-leverandørforhold. Begge parters opnår herved ejerskab over projektet og kan influere dets chancer for succes, hvilket ligeledes påvirker fordelingen af risici og det indbyrdes ansvar.

Udbudsproceduren, innovationspartnerskab gør det muligt at udbudssætte kontrakten med udgangspunkt i den altafgørende Product Backlog, hvilken skaber den fornødne fleksibilitet, den agile udviklingsmetode kræver. Udbudsloven regulerer i høj grad parternes mulighed for at fortage ændringer gennem projektets løbetid, og baggrunden herfor er sondringen mellem hvorvidt, de agile tilpasninger resulterer i ændringer *af* kontrakten eller i *medfør af* kontrakten, idet kun sidstnævnte er tilladt. Altafgørende er det i denne sammenhæng, at kontrakten klart og entydigt definerer projektets forventede ændringsbehov og foranstaltningerne herfor, eftersom ændringer ikke må fordreje det oprindelige grundlag for kontrakttildelingen.

Vurderingen af ændringens karakter afhænger blandt andet af hvorledes kontraktens genstand defineres, herunder som værende en vare eller en tjenesteydelse.

Der argumenteres i denne afhandling for, at denne bør defineres som en tjenesteydelse, grundet det ufuldstændige kontraktgrundlag, partnerskabets indgås på, samt det faktum, at værdien af den udbudssatte opgave hovedsageligt beror på selve udviklingen af den innovative løsning. Dertil kræves det ikke af omtalte udbudsprocedure, at innovationssamarbejdet i det hele taget omfatter køb af det endelige produkt, hvilket ligeledes støtter op om denne opfattelse. Såfremt kontraktens genstand defineres således, kan kontrakttildelingen efter den nye Udbudslov desuden i højere grad bero på en vurdering af underkriterier vedrørende kvalitative og sociale aspekter. Disse kan indeholde innovation-relevante karakteristika, og gøre det muligt for ordregiver at tildele kontrakten på grundlag af udviklingsteamets IT-kompetencer, samarbejdsevner og modenhed i forhold til den udvalgte opgave og udviklingsmetode.

Idet, den agile metode anvender inkrementel udvikling af produktet og løbende implementering, er det relevant for ordregiver at tage stilling til accept-processen for både delleverancer og det endelig produkt. Desuden hvilket tidspunkter og på hvilket grundlag garantier skal være gældende. Kontrakten bør tillige adressere, hvorledes ophør af kontrakten, som følge færdigudvikling tidligere end forventet, påvirker den aftalte pris, idet aftalens økonomiske balance ikke må ændres til fordel for leverandøren på en måde, som ikke var fastsat i de oprindelige udbudsbetingelser.

Med udgangspunkt i disse ovennævnte kommercielle og udviklingsmæssige findings, kan det konkluderes, at det ikke blot er muligt for en offentlig ordregiver at udlicitere en software udviklingsopgave med afsæt i en agil udviklingsmetode. Det er ligeledes yders relevant.

8 REFERENCER

[1] R. Kurzweil, The Age of Spiritual Machines, Ray Kurzweil, 1999.

[2] R. Kurzweil, "The Law of Accelerating Returns," kurzweilai.net, 2001.

[3] G. Chin, Agile Project Management: How to Succeed in the Face of Changing Project Requirements, AMACOM, 2004.

[4] D. Aronson, "Overwiew og systemthinking," thinking.net.

[5] P. Senge, The Fift Discipline, Currency, 2006.

[6] A. Opelt, B. Gloger, W. Pfarl and R. Mittermayr, Agile Contracts: Creating and Managing Successful Projects with Scrum, John Wiley & Sons, Inc, 2013.

[7] Den Danske Ordbog, "Agil," [Online]. Available: http://ordnet.dk/ddo/ordbog?query=agil. [Accessed 30. jan. 2016].

[8] Vocabulary.com, "Agile," [Online]. Available: http://www.vocabulary.com/dictionary/agile. [Accessed 30. jan. 2016].

[9] Mind Fitness Training Institute, "What is mind fitness and why do i need it?," [Online]. Available: http://www.mind-fitness-training.org/tr_what_fitmind.html. [Accessed 30. jan. 2016].

[10] B. Boehm and R. Turner, Balancing Agility and Discipline: A Guide for the Perplexed, Boston, MA: Addison-Wesley, 2004.

[11] J. Highsmith and A. Cockburn, "Agile Software Development: The Business of Innovation," *IEEE Computer*, pp. 120-127, 2001.

[12] K. Preiss, "Agility – the Origins, the Vision and the Reality," International Conference on Agile Manufacturing, 2005.

[13] W. Luk and G. A. Constantinides, Transforming Reconfigurable Systems, Imperial College Press, 2015.

[14] S. Singh and I. Chana, "Enabling Reusability in Agile Software Development," *International Journal of Computer Applications vol 50*, pp. 33-40, 2012.

[15] J. K. Liker, The 14 Principles of the Toyota Way: An Executive Summary of the Culture Behind TPS, University of Michigan, 2004.

[16] P. Measey and Radtac, Agile Foundations, Principles, practices and frameworks, Radtac Ltd., 2015.

[17] C. Larman and V. R. Basili, "Iterative and Incremental Development: A Brief History," *IEEE Computer Society*, pp. 47-56, 2003.

[18] J. Erickson, K. Lyytinen and K. Siau, "Agile Modeling, Agile Software Develpment, and Extreme Programming," *Journal of Database Management, vol. 16*, p. 88–100, 2005.

[19] T. Gilb and K. Gilb, "Evolutionary Project Management," 2015. [Online]. Available: http://gilb.com/Project-Management. [Accessed 30. jan. 2016].

[20] The Agile Alliance, "Manifesto for Agile Software Development," sep. 2015. [Online]. Available: http://www.agilemanifesto.org/. [Accessed 30. jan. 2016].

[21] The Agile Alliance, "Principles behind the Agile Manifesto," sep. 2015. [Online]. Available: http://www.agilemanifesto.org/principles.html. [Accessed 30 jan. 2016].

[22] S. Nerur, R. Mahapatra and G. Mangalaraj, "Challenges of migrating to agile methodologies," *Communications of the ACM*, pp. 72-78, Maj 2005.

[23] B. Boehm, "Get Ready for Agile Methods, with Care," *IEEE Computer, vol. 35*, pp. 64-69, 2002.

[24] A. Cockburn and J. Highsmith, "Agile software development: The business of innovation," *IEEE Computer*, pp. 120-122, September 2001.

[25] J. Cho, "A Hybrid Software Development Method For Large-Scale Projects: Rational Unified Process With Scrum," *Information Systems, Volume X*, pp. 240-348, 2009.

[26] M. A. Awad, "A Comparison between Agile and Traditional Software Development Methodologies," The University of Western Australia, 2005.

[27] T. Gilb and T. Johansen, "From Waterfall to Evolutionary Development (Evo): How we rapidly created faster, more user-friendly, and more productive software products for a competitive multi-national market," *INCOSE International Symposium, vol 15*, p. 1676–1686, 2005.

[28] R. Mitchell and J. McKim, Design by Contract, by Example, Addison-Wesley, 2001.

[29] D. P. Truex, R. Baskerville and H. Klein, "Growing systems in emergent organizations," *Communications of the ACM, Vol. 42 No. 8*, pp. 117-123, 1999.

[30] A. Cockburn and J. Highsmith, "Agile software development: the people factor," *IEEE Computer*, pp. 131-133, 2001.

[31] D. A. H. Mohammad, D. T. Alwada'n and D. J. ". Ababneh, "Agile Software Methodologies: Strength and Weakness," *International Journal of Engineering Science and Technology (IJEST) vol 5*, pp. 455-459, 2013.

[32] The Standish Group, "The CHAOS Manifesto," 2011.

[33] Net Research Analysis, "8th Annual State of Agile Rapport," VersionOne, 2013.

[34] D. Turk, R. France and B. Rumpe, "Assumptions Underlying Agile Software Development process," *Journal of Database Management vol 16*, pp. 62-87, 2005.

[35] V. Basili, G. Caldiera and R. D., "Experience Factory," in *Encyclopedia of Software Engineering*, Wiley & Sons, 1994, pp. 469-476.

[36] R. S. Pressman, Software Engineering", 7th edition, McGraw Hill Education, 2009.

[37] M. Fowler, Refactoring: Improving the Design of Existing Code, Addison-Wesley, 1999.

[38] A. I. Binstock, "In Praise Of Small Code," *Information Week*, p. 41, 2011.

[39] F. P. Brooks jr., "The Tar Pit," in *Mythical Man-Month, The: Essays on Software Engineering, Anniversary Edition*, Addison-Wesley, 1982.

[40] J. Nordfalk, "Videregående OOP," in *Videregåendeprogrammering i Java*, GLOBE, 2003, p. Kapitel 15.

[41] H. Merisalo-Rantanen, T. Tuure and R. Matti, "Is extreme programming just old wine in new bottles: a comparison of two cases," *Journal of Database Management vol 16*, pp. 41-61, 2005.

[42] P. Mcbreen, Questioning Extreme Programming, MA, USA: Pearson Education, 2003.

[43] M. Stephens and D. Rosenberg, Extreme Programming Refactored: The Case Against XP, Apress, 2003.

[44] L. R. Vijayasarathy, "Agile Software Development: A survey of early adopters," *Journal of Information Technology Management, vol. XIX*, pp. 1-8, 2008.

[45] N. Uikey, U. Suman and A. K. Ramani, "A Documented Approach in Agile Software Development," *International Journal of Software Engineering (IJSE)*, pp. 13-22, 2011.

[46] M. Finnegan, "Agile and waterfall – is a hybrid approach best for enterprise app development?," 2014. [Online]. Available: http://www.computerworlduk.com/it-management/agile-waterfall-is-hybrid-approach-best-for-enterprise-app-development-3572875/. [Accessed 30. jan. 2016].

[47] C. Torode, "Agile best practices can combine waterfall and Scrum," 2010. [Online]. Available: http://searchcio.techtarget.com/news/1516059/Agile-best-practices-can-combine-waterfall-and-Scrum. [Accessed 30. jan. 2016].

[48] J. Shore, "Fail Fast," *IEEE Computer Society*, pp. 21-25, 2004.

[49] VersionOne, "State of Agile," VersionOne, 2015.

[50] M. Cohn, User Stories Applied: For Agile Software Development, Addison-Wesley Professional, 2004.

[51] K. Schwaber and J. Sutherland, "The Scrum Guide™," Scrum.Org and ScrumInc, 2014.

[52] Admin, "The Scrum Backlog," 2016. [Online]. Available: http://scrummethodology.com/?s=backlog. [Accessed 30. jan. 2016].

[53] G. Murphy, "Better Requirements Definition Management is Better for Business," *Mothods & Tools*, 2010.

[54] RTI, "The Economic Impacts of Inadequate Infrastructure for Software Testing," NIST, 2002.

[55] IBM Software Group, "Minimizing code defects to improve software quality and lower development costs.," IBM, 2008.

[56] Gartner, "From Concept to Production," *Software Changes and Configuration Management*, 2008.

[57] J. Moccia, "Agile Requirements Definition and Management," 2012. [Online]. Available: https://www.scrumalliance.org/community/articles/2012/february/agile-requirements-definition-and-management. [Accessed 30. jan. 2016].

[58] The Standish Group, "CHAOS Report," Project Smart, 2014.

[59] A. Taylor, "IT projects: sink or swim," *Computer Bulletin*, pp. 24-26, 2000.

[60] D. G. Reinertsen, The Principles of Product Development Flow: Second Generation Lean Product Development, Celeritas Publishing, 2009.

[61] The Standish Group, "CHAOS Report," 2004.

[62] W. R. Duncan, A Guide to the Project Management Body og Knowledge, PMI Publishing Division, 1996.

[63] K. Heldman and V. Mangano, PMP Project Management Professional Exam Review Guide, Wiley Publishing, 2009.

[64] M. Holm and R. C. Jakobsen, "Building an Agile Enterprise with Lean Culture," [Online]. Available: https://systematic.com/media/282239/Building_an_Agile_Enterprise_with_Lean_Culture.pdf. [Accessed 30. jan. 2016].

[65] D. Lowe and R. Leiringer, Commercial Management of Projects: Defining the Discipline, Wiley-Blackwell, 2006.

[66] K. Sorsa, T. Salmi-Tolonen and R. F. Henschel, "Proactive Contracting and Risk Management," in *Proactive Management and Proactive Business Law, A Handbook*, Tampereen Yliopistopaino Oy - Juvenes Print, 2011.

[67] P. Smith and R. Pichler, "Agile Risk/Agile Reward," *ProQuest Science Journal*, pp. 50-53, 2005.

[68] T. Aven, "On how to define, understand and describe risk," *Reliability Engineering & System Safety, vol. 95,* p. 623–631, 2010.

[69] ISO, *Risk management-vocabulary. Guide 73: 2009,* 2009.

[70] AS/NZS, *Risk Management Standard, AS/NZS 4360: 2004,* Jointly published by Standards Australia International Ltd., 2004.

[71] ISO, *Risk management—principles and guidelines, ISO 31000:2009,* 2009.

[72] R. F. Henschel, "Applications: Proactive Risk Management," in *Proactive Management and Proactive Business Law, A Handbook*, Tampereen Yliopistopaino Oy - Juvenes Print, 2011.

[73] K. Sorsa, "Applications: Proactive Contracting in Value Chain Management," in *Proactive Management and Proactive Business Law, A Handbook*, Tampereen Yliopistopaino Oy - Juvenes Print, 2011.

[74] P. Junge, "Strategic Opportunity Management," HTWG Konstanz, 2009.

[75] A. Camacho and E. Garcia, "Innovation Management," in *Proactive Management and Proactive Business Law, A Handbook*, Tampereen Yliopistopaino Oy - Juvenes Print, 2011.

[76] J. Darroch, "Knowledge management, innovation and firm performance," *Journal of Knowledge Management, vol 9,*, pp. 101-115, 1997.

[77] R. Dove, "Knowledge management, response ability, and the agile enterprise," *Journal of Knowledge Management*, pp. 18-35, 1999.

[78] R. Jurčík, "Changes to agreements related to public contracts in the EU and Czech," in *Recent Researches in Law Science and Finances*, WSEAS Press, 2013, pp. 15-19.

[79] Udbudsportalen, "Offentlig-privat innovation," 2016. [Online]. Available: http://www.udbudsportalen.dk/Strategi-og-Politik/Modeller-for-OPS/Offentlig-privat-innovation/. [Accessed 30. jan. 2016].

[80] KL, "Seks modeller for offentlig-privat samarbejde - En guide til kommunerne," Udbudsportalen, 2010.

[81] A. N. Westergaard, "Øget adgang til de mere fleksible udbudsformer ”udbud med forhandling” og ”konkurrencepræget dialog”," 2015. [Online]. Available: http://kammeradvokaten.dk/nyheder/2015/5/udbud-af-it-bliver-det-nemmere-med-den-nye-udbudslov/. [Accessed 30. jan. 2016].

[82] J. Bendsen, "Code Camp – et koncept til vurdering af agile teams," 2015. [Online]. Available: https://home.lundogbendsen.dk/ff20150814-post/. [Accessed 30, jan. 2016].

[83] A. Knudsen, "Få hjælp til at orkestrere den agile transformation, Den agile transformation hos STAR," 2015. [Online]. Available: http://www.changegroup.dk/agil-transformation/. [Accessed 30 jan. 2016].

[84] J. Pfeffer and R. Suttin, Hard Facts, Dangerous Half-Truths And Total Nonsense: Profiting from Evidence-Based Management, Harvard Business School Press, 2006.

[85] R. D. Austin, Measuring and Managing Performance in Organizations, Dorset House Publishing Co., 1996.

[86] A. Kohn, Punished by Rewards, Houghton Mifflin, 1993.

[87] F. Herzberg, One More Time: How Do You Motivate Employees?, Harvard Business, 1987.

[88] T. Arbogast, C. Larman and B. Vodde, "Contracts," in *Practices for Scaling Lean & Agile Development: Practices for Scaling Lean & Agile Development:*, Addison-Wesley, 2012, p. 499ff.

[89] P. Aghion, N. Bloom and J. Van Reenen, "Incomplete contracts and the internal organization of firms," *Journal of Law, Economics, and Organization* , p. 37–63, 2014.

[90] T. L. Brown, M. Potoski and D. Van Slyke, Complex contracting: Government contracting in the wake of the U.S. Coast Guard's Deepwater Program, Cambridge University Press, 2013.

[91] The Agile Alliance, "Definition Of Done," 2013. [Online]. Available: http://guide.agilealliance.org/guide/definition-of-done.html. [Accessed 30 jan. 2016].

[92] *EU-Domstolens dom i sag C-454/06, Pressetext Nachrichtenagentur mod Østrig*, 19. juni 2008.

[93] Digitaliseringsstyrelsen, "K03 Standardkontrakt for agile it-projekter," 2. sep. 2014. [Online]. Available: http://www.digst.dk/Styring/Standardkontrakter/K03-Standardkontrakt-for-agile-projekter.aspx. [Accessed 30. jan. 2016].

[94] A. P. G. Yin, "Scrum Maturity Model," 2011.

[95] L. Andersen and P. Madsen, Aftaler og Mellemmænd, Karnov Group Denmark A/S, 2006.

[96] Project Management Institute, A Guide to the Project Management Body of Knowledge (PMBOK Guide), Project Management Institute, vol 5, 2013.

[97] cvsnead, "Structured Agility: Agile Project Integration Management," 2013. [Online]. Available: https://theadaptivepm.wordpress.com/2013/04/01/sdlc-and-agile-project-integration-management/. [Accessed 30. jan. 2016].

[98] Microsoft Developer Network, "What is an Enterprise Application?," 2016. [Online]. Available: https://msdn.microsoft.com/en-us/library/aa267045(v=vs.60).aspx. [Accessed 30. jan. 2016].

[99] D. Alessi, "The Distinction between Obligations de Résultat and Obligations de Moyens and the Enforceability of Promises," *European Review of Private Law, Issue 5*, p. 657–692, 2005.

[100] R. E. Barnet, "Conflicting Visions: A Critique of Ian Macneil's Relational Theory of Contract," Georgetown Law Library, 1992.

[101] M. Cohn, "User Stories, Epics and Themes," 2011. [Online]. Available: https://www.mountaingoatsoftware.com/blog/stories-epics-and-themes. [Accessed 30. jan. 2016].

[102] F. DeRemer and H. Kron, "Programming-in-the-Large Versus Programming-in-the-Small," *IEEE Trans*, pp. 114 - 121, 1975.

[103] S. Godfrey, "What is CMMI?," NASA presentation, 2008.

[104] C. Houy, "Business Process Management in the Large," *Business & Information Systems Engineering*, pp. 385-388, 2011.

[105] I. R. Macneil, "Relational contract: What we do and do not know," *Wisconsin Law Review*, p. 483–525, 1985.

[106] J. D. Meier, J. Taylor, P. Bansode, A. Mackman and K. Jones, "Work Items Explained," 2007. [Online]. Available: https://msdn.microsoft.com/en-us/library/bb668962.aspx. [Accessed 30. jan. 2016].