# Att mäta och kommunicera affärsvärde i team som använder lättrörliga och lean- metoder

---

Max Weijola

**Kandidatarbete**
Handledare: Jeanette Heidenberg
Institutionen för informationsteknologi
Åbo Akademi

**Sammanfattning**

Vid mjukvaruutveckling kan kommunikationen mellan kund och utvecklare vara svår och missförstånd är vanliga, vilket kan påverka den slutgiltiga produkten negativt. Speciellt kommunikation av affärsvärde är en stor utmaning men ger i gengäld hög utdelning i form av djupt samarbete och förståelse. Ömsesidigt samarbete och kommunikation resulterar i högre värde för kunden samtidigt som utvecklingsteamet kan förbättra sina interna processer.

Lättrörliga metoder ställer ytterligare krav på kundrelationen eftersom prioritering av funktioner till nästa iteration är grundläggande för arbetssättet. Prioriteringen kan göras på basen av olika kriterier, t.ex. risk, värde och kostnad. Denna uppsats undersöker den befintliga litteraturen och presenterar verktyg för olika former av prioritering och ekonomisk analys av krav och kravsamlingar.

# Innehåll

# Kapitel 1

# Introduktion

Lättrörliga metoder har rönt stora framsteg inom mjukvaruutvecklingen tack vare inbyggda egenskaper för att kunna reagera på ständigt skiftande krav. Styrkan ligger i att kunna prioritera kvarvarande arbetsmängd för att uppnå största möjliga effektivitet och kvalitet på programvaran. Denna egenskap ställer dock höga krav på samarbetet mellan utvecklarteamet och kunden:

> ...choosing the highest priority delivery really helps to reduce work in process, but you need support and commitment from the business to do this. - *Alan de-Ste-Croix* [6]

Kommunikationen blir således den viktigaste faktorn i värdeutdelningen och det ligger i båda parters intresse att effektivera förståelsen för prioritering och affärsvärde. Problematiken kommer från båda parter och att undvika missförstånd är en utmaning. Kunden kan ha svårt att kommunicera önskemål på grund av avsaknad av teknisk kunskap eller ofullständig helhetsbild. Programmeringsteamet kan på motsvarande sätt ha svårt att förstå kundens problemdomän och arbetssätt. Genom att introducera olika tekniker och arbetssätt kan utvecklingsföretag arbeta aktivt med att öka förståelsen för sina kunders värderingar.

Affärsvärde är ett begrepp som ofta används men betydelsen kan vara varierande. I och med firandet av 10-års jubileumet för *Agile Manifesto* identifierades affärsvärde som ett av de lättrörliga metodernas nuvarande problem [8]. Mike Cohn ger följande beskrivning av problemet (fritt översatt); “ produktägare får ofta rådet att prioritera funktionalitet utgående från affärsvärde, men vad är affärsvärde? ” [5]

Målsättningen är att genom befintlig litteratur diskutera för- och nackdelar med kontinuerlig kundkontakt. Därutöver kommer befintliga tekniker för att mäta affärsvärde att behandlas. Slutligen tas användningen av affärsvärde upp vid val och prioritering av krav.

# Kapitel 2

# Utvecklingsmetoder

## 2.1 Tidiga utvecklingsmetoder

Mjukvaruutveckling ärvde i ett tidigt skede många egenskaper från andra ingenjörsmässiga fält som byggnadsindustri och arkitektur. Detta arbetssätt innebar en sedvanlig uppdelning i krav och kontrakt, analys och implementation. Den tekniska utvecklingen var så snabb och tidskrävande att själva mjukvaruprocessen inte studerades i lika hög grad. På grund av hårdvarans höga pris var det endast stora aktörer som t.ex. statliga verk, militärer och stora bolag som hade råd att beställa mjukvara. Kunder i denna kategori präglas ofta av formalitet och byråkrati och lägger stor tillit till skriftliga kontrakt vilket resulterade i noggrant utarbetad kravdokumentation. Problematiken i detta arbetssätt ligger inte i att dokumentera och implementera krav utan i svårigheten för kunden att entydigt kunna uttrycka vad programmet bör göra. För att exemplifiera detta kan nämnas en undersökning från The Standsh Group 1994, var den största enskilda faktorn för nerlagda mjukvaruprojekt var brist på information från användarna (ingår i [10]. En annan undersökning gjord av ESPITI ( European Software Process Improvement Training Initiative) år 1995 visade att ca. en tredjedel av alla fel i levererad programvara berodde på problem med krav (ingår i [10]).

### 2.1.1 Vattenfallsmodellen

Den första egentliga processmodellen som identifierats för mjukvaruutveckling är den av Winston Royce [14] beskrivna (men icke namngivna) vattenfallsmodellen. Royce skriver att kärnan i all mjukvaruutveckling är någon form av analys följt av programmering. För att stödja dessa kärnsteg krävs oftast även kravspecificering före analysen, design före programmeringen och testning före överlämnande till kunden. Begreppet *vattenfallsmodell* har senare uppkommit eftersom dessa steg oftast utfördes i ordning efter varandra utan möjlighet att gå tillbaka till en högre nivå likt vatten som rinner nedåt från en högre ansats till en lägre. Problematiken som Royce beskriver, och varför detta arbetssätt sällan kan fungera kostnadseffektivt är när någonting oväntat dyker upp under utvecklingsarbetet och flera steg måste göras om. T.ex om ett oväntat fel uppstår under testningen måste i värsta fall programmet designas på nytt vilket medför att alla steg från och med kravhanteringen måste repeteras. Denna form av oelasticitet kan lätt få utvecklingskostnaderna att eskalera. För att undvika liknande scenarion föreslog Royce att utvecklingen borde göras i två steg [14]. Först skulle en enklare version av programvaran utvecklas under cirka en fjärdedel av totala projekttiden. Detta leder till att potentiella problem som t.ex. tidskrav och datalagringsbegränsningar upptäcks tidigt och kan testas och korrigeras mera verklighetstroget än vad simuleringar kan. Den första versionen av programvaran fungerar sedan som utgångsläge i utvecklingsfaserna för det slutgiltiga programmet.

### 2.1.2 Tidiga iterativa modeller

I ett försök att adressera vattenfallsmodellens begränsningar introducerades olika iterativa modeller för att bättre kunna hantera risker i stora projekt. En av de mera kända metoderna är spiralmodellen [4] som försöker kombinera iterativ utveckling med riskhantering och budgetuppföljning. Kritik har dock framförts emot spiralmodellen bland annat för att den är komplex och användningsområdena kan vara begränsade [16].

## 2.2 Lättrörliga metoder

För att bättre kunna reagera på ständigt ändrande kravspecifikationer och minska de ekonomiska riskerna för kunden växte diverse lättrörliga metoder fram runt millennieskiftet. Som svensk översättning återfinns även *rörliga metoder* och *agila metoder*, från engelskans *Agile Methods*. Namnet vill reflektera att lättrörliga metoder vill minska dokumentation och administration som kan försinka utvecklingsarbetet. Lättrörlighet anspelar också på möjligheten att snabbt kunna reagera på ändrade krav samt att team inte bör vara för stora i medlemsantal. Principerna för lättrörlig utveckling utkristalliserades och skrevs

ner av representanter från olika lättrörliga metoder i februari 2001. Resultatet blev dokumentet *Agile Manifesto* (se bilaga A) som alla parter, trots sina olika bakgrunder, gick med på att underteckna [1].

Manifestet beskriver fyra huvudprinciper som omprioriterar vissa tidigare vedertagna element i utvecklingsprocessen. Som första princip framhävs individer och interaktioner som viktigare element än processer och verktyg för att effektivera kommunikationen. Vidare prioriteras fungerande programvara över omfattande dokumentation vilket kan tolkas som ett försök att tackla problemet med projekt som avstannar vid specificeringsskedet. Som tredje princip värderas kundsamarbete framför kontraktsförhandling. Genom denna princip vill lättrörliga metoder drastiskt tona ner vikten över uttömmande kravspecifikation och kontrakt för att istället snabbare börja utveckla programvaran. Slutligen värdesätts anpassning framför att följa en plan. Denna sista princip tacklar problematiken med höga kostnader vid ändrade krav i utvecklingen genom att aktivt arbeta med möjligheten att krav kommer att ändras under arbetets gång.

### 2.2.1 Extreme Programming

Extreme Programming (XP) är en av de första lättrörliga utvecklingsprocesserna, framtagen av Kent Beck [2] och identifierar i stort sett samma problem som Royce [14]. När Royce föreslår en tudelad utveckling i en prototyp som leder till färdig produkt, försöker XP tackla problemet genom att dela upp arbetet i ett stort antal små iterationer. Iterationerna koncentreras alltid på vad som är viktigt för tillfället istället för att titta för långt in i framtiden. Beck hävdar vidare att genom att ta beslut om kritiska delar i ett projekt och inleda kodningen i ett så tidigt skede som möjligt går det att spara på kostnader senare.

XP är uppbyggd av ett antal tekniker vilka tillsammans försöker hjälpa utvecklare att bättre kunna reagera på ändrande krav under arbetets gång. Sammanlagt omfattar XP tolv olika tekniker [3] och innefattar; små leveranser, metafor, enkel design, tester, gemensamt ägarskap, omstrukturering av kod, parprogrammering, kontinuerlig integration, 40-timmars arbetsvecka, och gemensam kodstandard [3]. De två intressantaste teknikerna i detta sammanhang är dock planeringsspelet och kund på plats vilka presenteras närmare i kapitel tre.

### 2.2.2 Lean Metoder

Lean är en mjukvaruprocess förespråkad av bland andra Mary och Tom Poppendieck utvecklad för att kunna skapa högkvalitativ programvara snabbt, kontinuerligt och pålitligt [12]. Lean är en processmodell som härstammar från Toyotas utvecklingsmetod *Lean Production* tillämpad på mjukvaruutveckling. Den viktigaste av de sju grundprinciperna inom lean är förhindrande av slöseri vilken samtidigt fungerar som en röd tråd. Principen går ut på att försöka eliminera möjligast många steg och artefakter i utvecklingsprocessen som inte skapar

värde för kunden. De övriga principerna stödjer den tidigarenämnda och består av; effektivera inlärning, göra beslut så sent som möjligt, leverera så snabbt som möjligt, bemyndiga teamet, bygga in integritet samt slutligen att se helheten [13].

### 2.2.3 Scrum

SCRUM är ett ramverk som ofta används för projektledning i utvecklingsteam som använder lättrörliga metoder och består av roller, artefakter, regler och tidsramar [15]. De tre rollerna inom scrum som Schwaber och Sutherland beskriver är klart definierade vilket ger medlemmarna stor handlingskraft. Till scrummästarens ansvar hör att teamet följer alla värderingar och regler som hör till scrum. Scrumteamet består av programmerare och personer som ansvarar för att omvandla krav till levererbara inkrement. Den sista rollen är produktägaren som har hand om behovslistan för produkten (eng. *Product Backlog*) och kommunicerar med kunden, rollen tas upp noggrannare i kapitel tre. Förutom rollerna är den så kallade sprinten kärnan i ramverket och består av en förutbestämd tidsenhet, t.ex. en månad, under vilken teamet utvecklar ett levererbart inkrement av programvaran [15]. Sprinten stöds bl.a. av korta dagliga möten samt ett längre planeringtillfälle i början och en tillbakablick efter avslutad sprint. Det faktum av att *Backlog*:en aldrig ses som färdig vittnar om scrums lättrörlighet; endast de högst prioriterade kraven är beskrivna i detalj och kan utvecklas.

# Kapitel 3

# Kundkontakt och förståelse för affärsvärde

## 3.1 Begreppet affärsvärde

Affärsvärde (eng *Business Value, BV*) är ett begrepp som kan ha varierande betydelse beroende på sammanhang. I denna text kommer affärsvärde att användas enligt Jeff Pattons definition som någonting som skapar ett mervärde hos organisationen som betalar för den utvecklade mjukvaran [11] . Mervärdet kan beskrivas som ökade intäkter, kostnadsinbesparingar eller förbättrade tjänster. Ett av problemen som Patton identifierar med begreppet affärsvärde är dess icke påtagliga natur. Varje slutanvändare och intressent uppfattar affärsvärde som en produkt leverar med en personlig vinkling och egna intressen [11]. Mike Cohn beskriver affärsvärde vid kravhanteringen på följande vis: “How much money will the organization make or save by having the new features included in the theme. ” [5].

## 3.2 Kontakten till kunden

Som tidigare nämnts är en fungerande kommunikation mellan kund och utvecklare viktig för lyckad utgång av utvecklingsprojekt. I tidig mjukvaruutveckling skedde merparten av kundkontakten under kravinsamlingen och utarbetandet av kontrakt. På grund av svårigheten att fullständigt förstå en problemdomän tidigt under ett projekt och tolkningsmöjligheter identifierades ett behov av att involvera kunden till en högre grad. Royce föreslår redan 1987 att kundkontakten bör vara integrerad på ett formellt sätt så att kunden är engagerad i ett tidigare skede än vid överlämnandet [14]. Ett annat förslag där kunden är mera involverad i planeringssteget och vid brainstorming inför ett projekt ger Leffingwell [10]. Här beskrivs ett kravmöte med deltagare från olika intressegrupper. En balans bör upprätthållas mellan intressenterna genom en opartisk mötesledare som rekommendationsvis inte ingår i utvecklingsteamet. De lättrörliga metoderna fortsätter att understryka vikten av kontinuerlig kundkontakt och presenterar olika tekniker, roller och principer för att maximera utbytet. Kundens tid kan dock tänkas vara en begränsad resurs, som Kent Beck skriver; "slutanvändare till systemet som utvecklas är är för värdefulla för att ställas till utvecklingsteamets förfogande" (fri översättning ur [3]).

### 3.2.1 Produktägaren

Inom scrum har produktägaren ansvaret för kundkontakten och att förstå problemdomänen för att kunna prioritera den kvarvarande kravlistan (eng *Product Backlog*). Denna uppgift är kritisk för en lyckad utgång av hela projektet och det ligger på produktägarens ansvar att göra produktens framgång till ett gemensamt intresse för teamet.

> The product owner has gone from someone who could blame development if a project failed to someone who is responsible for the success or failure of the project. - Ken H. Judy [7]

Enligt klassisk scrum bör produktägaren alltid vara en enda person för att ansvarsfördelningen skall vara så enkel som möjlig och beslut klara och enhälliga [7, 15]. Om denna uppgift blir för stor och ansvarsfull för en person så kan arbetsgrupper och experter ge råd och hjälpa till att förstå problemdomänen men alla beslut om t.ex prioritering borde fattas av produktägaren. Variationer har föreslagits till produktägarrollen för att t.ex. bättre kunna hantera stora projekt i form av prouktägarteam [6] eller för att bättre kunna kommunicera värde till scrumteamet i form av kollektivt produktägarskap [7].

#### Produktägarteam

Ett förslag på hanteringen av produktägarrollen i väldigt stora projekt är så kallade produktägarteam [6]. Teamen består då av olika experter från varierande

områden och har som uppgift att översätta kundens behov till implementerbara krav. Under själva implementationen fungerar produktägarteamet som en representant för kunden vid frågor från utvecklarna.

#### Kollektivt produktägarskap

För team som nyligen inkorporerat lättrörliga metoder kan uppdelningen i produktvision och tekniskt genomförande vara förhållandevis strikt [7]. En dylik inställning där programmerare utför implementationen utan engagemang i produkten som helhet kanske inte alltid resulterar i ett optimalt arbetsflöde. Ett förslag på hur detta problem kan adresseras är den vinkling på produktägaren och samarbetet inom scrumprojekt som föreslås av Judy och Krummins-Beens som kollektivt produktägarskap [7]. Termen bör inte förväxlas med den liknande XP tekniken *Collective Ownership* [3] som avser gemensamt ägandeskap av källkod.

Kollektivt produtägandeskap är en till synes svår balansgång för produktägaren. Dels måste denne ta fullt ansvar för produkten och kravprioritering men samtidigt kunna involvera scrumteamet till den grad att alla känner ett gemensamt ansvar för slutresultatet.

> The team is passionate about the product they are building and feel personally accountable to the product's success - *Ken H. Judy och Ilio Krumins-Beens* [7]

### 3.2.2 Tekniker inom XP

Av de tolv varierande teknikerna i XP finns det två som kan tänkas ha en mer framträdande roll i kontakten till kunden och för förståelsen av affärsvärde. Beck definierar den bakomliggande problematiken som en kamp där affärs- och tekniska intressen är drivna av olika prioriteringar och eftersträvar olika mål [3]. Om någondera part erhåller för stort inflytande i utvecklingen lider oftast kvaliteten på arbetet.

#### Planeringsspelet

Ett återkommande moment i början på varje iteration är planeringsspelet [3] vars mål är att försöka skapa en symbios mellan motpolerna som utgörs av affärs- och tekniska intressenter. Målet är att maximera nyttan och värdet av den utvecklade mjukvaran. Beck beskriver uppdelningen som att affärsintressen oftast representeras av kunden eller expertanvändare med god insikt i produktens syfte. De tekniska intressena representeras av utvecklingsteamet och andra involverade i implementationen av produkten. Spelets pjäser är sk. *Story Cards*, kort med beskrivningar av programfunktioner. Dessa kort skrivs av

affärsintressenterna som har en vision om vad den slutgiltiga produkten bör göra. Det är sedan programmerarnas ansvar att uppskatta arbetsmängden som kommer att gå åt att implementera varje kort.

När alla *Story Cards* är skrivna prioriteras dessa av affärsintressenter i tre steg; funktionskritiska, värdefulla och till sist mindre viktiga. På basen av detta är det igen utvecklarnas uppgift att prioritera korten i tre olika klasser av riskfylldhet med avseende på tidsåtgång på basen av tidigare erfarenhet. När således intressenterna har fått uttrycka sina egna tankar väljer företagsrepresentanterna en mängd programfunktioner och längden på iterationen bestäms av programmerarna. Alternativt kan leveransdatum bestämmas av kunden och programmerarna väljer vilka funktioner som kan implementeras inom tidsrymden. Planeringsspelet innehåller även en fas under själva iterationen där justeringar kan förhandlas ifall tidsbrist uppstår eller nya programfunktioner behöver inkoroporeras redan i pågående iteration [3].

### Kund på plats

En värdefull men samtidigt svårgenomförbar teknik är att ha en kund eller motsvarande expertanvändare på plats med utvecklingsteamet [3]. I sin renaste form skulle kunden sitta full tid i samma rum som utvecklarna för att kontinuerligt kunna svara på frågor om önskad funktionalitet och prioritering. Beck beskriver problematiken som att kundrepresentanten troligtvis skapar ett större värde i sin egen organisation än genom att sitta med utvecklingsteamet hela tiden [3]. De negativa aspekterna som utvecklarna måste acceptera om kontinuerlig kundkontakt inte är möjlig är bl.a. noggrannare och långsiktigare planering vilket både är tidskrävande och ökar riskerna [3].

# Kapitel 4

# Affärsvärde som grund för prioritering

På grund av det hårdnande konkurrensklimatet inom mjukvaruutveckling är lyckade projekt och återkommande nöjda kunder eftertraktade för alla utvecklingsföretag. Misslyckade mjukvaruprojekt har blivit mindre accepterade av samtliga parter och potentiella kunder ställer allt högre krav.

> Companies need assurance that they pay for software that corresponds to their real needs even if they are not capable to define them correctly. - *Oualid Ktata and Ghislain Levesque*

Förutom kollektivt produktägandeskap vilket diskuterades i föregående kapitel finns det andra tillvägagångssätt för utvecklingsteam att förstå problemdomänen och skapa programvara som tillfredsställer kundens behov. Ett tillvägagångssätt är att monetärt uppskatta affärsvärde för olika funktioner för att kunna göra objektivare och bättre motiverade val i prioriteringssituationer [5, 11, 13].

## 4.1 Prioritering av krav

En fas där kompromisser ofta måste göras mellan utvecklare och kund är vid val av vilka krav som bör implementeras i kommande iteration.

#### Hundradollarstestet

Dean Leffingwell föreslår hundradollarstestet som en teknik för att välja vilka krav eller funktioner som bör prioriteras i ett tidigt skede efter kravinsamlingen [10]. Alla intressenter tilldelas en lika stor summa imaginär valuta att spendera på krav och funktionalitet från en dokumenterad samling. Med denna teknik tilldelas varje idé ett värde som reflekterar den kollektiva uppfattningen och kan användas som bas för beslutsfattande. Leffingwell påpekar dock att denna teknik endast kan användas tillförlitligt en gång inom samma grupp eftersom deltagare blir påverkade efter att ha sett resultaten en gång.

#### Kategorisering

En annan teknik föreslagen av Leffingwell är kategorisering enligt ”kritisk, viktig, användbar”. Tekniken går ut på att ge intressenter en tredjedel röstsedlar av vardera kategori för att rösta och prioritera identifierade krav. På detta sätt kan inte alla krav bli klassade som kritiska utan högst en tredjedel. Klassificeringen *kritisk* betyder att systemet inte anses vara möjligt att levereras utan denna funktion. *Viktig* avser funktioner som tillför mervärde och utan dessa kan missnöje bland användare uppstå. *Användbara* funktioner är sådana som underlättar användningen och kan skapa mervärde och en bättre användarupplevelse [10]

## 4.2 Finansiell analys som grund för prioritering

Finansiella modeller har länge använts för att uppskatta lönsamheten för projekt inom diverse branscher. Kontinuerlig användning under mjukvaruutveckling för att kommunicera krav och funktionalitet ur ett monetärt perspektiv har dock historiskt sett varit begränsat [13]. Rätt använda kan ekonomiska modeller dock underlätta arbetet för utvecklare genom att introducera en gemensam (monetär) enhet, värderad enligt kundens behov, att basera beslut på.

Eftersom uppskattningarna endast är ungefärliga är det svårt att analysera och behandla enskilda krav. Tidsramen för uppskattning varierar även i storlek, Poppendiecks föreslår en femårsperiod [13] medan Cohn föreslår en tvåårsperiod uppdelad i kvartal [5]. Projekt och produkter som helheter kan uppskattas t.ex. med en produktmodell [13] som diskuteras i nästa kapitel. Krav och mål inom en produkt kan vara för små att uppskattas enskilt och kan t.ex. behandlas som kravgrupperingar (eng *Themes*) [5] eller i en applikationsmodell [13].

### 4.2.1 Tid är pengar

För att kunna uppskatta vikten av snabb leverans, eller i motsats, kostnaden för förseningar har Mary och Tom Poppendieck föreslagit en ekonomisk produktmodell (eng *Product model*) applicerad på lean metoder [13]. Modellen bygger på en uppskattad resultaträkning under en femårsperiod för en produkt under utveckling. Modellen bör vara tillräckligt enkel och allmän för att alla delaktiga skall finna den trovärdig. Variabler som föreslås för uppskattning är t.ex. antal sålda licenser, pris, marknadsandel samt kostnader såsom distribution, marknadsföring och underhåll.

Resultat av strategiska beslut såsom att reducera funktionalitet eller öka utvecklingsbudgeten kan uppskattas t.ex. genom att be kundens marknadsföringsavdelning korrigera berörda variabler. Den största kostnaden har visat sig vara att inte kunna leverera produkten inom utsatt tid och växer dramatiskt i jämförelse med endast uteblivna intäkter. Detta kan bland annat bero på minskad marknadsandel och övriga faktorer som inte utvecklare alltid är medvetna om [13]. Ett konkret förslag som Mary Poppendieck ger är att tilldela varje projekt en bokförare som, i samråd med övriga ekonomiska intressenter, kan skapa och upprätthålla en resultaträkning för produkten. Därigenom underlättas prioritering och beslutsfattande genom en form av ekonomisk förankring.

### 4.2.2 Applikationsmodellen

Även vid mindre utvecklingsprojekt där inte en hel produkt utvecklas kan det vara lönsamt att kunna prioritera på basen av ekonomiska faktorer. Problemen som programvaran ämnar lösa ges uppskattade monetära värden för att erhålla en bättre förståelse för kundens ekonomiska intressen. Poppendiecks föreslår en såkallad applikationsmodell (eng *Application model*) där problemen som försöker lösas tilldelas egna kolumner i en tabell och kan därigenom jämföras mot företagets ursprungliga inkomster [13]. Genom att uppskatta de ekonomiska resultaten för varje mål i en skild kolumn kan dessa dels prioriteras sinsemellan men också adderas ihop och ett totalt adderat mervärde på t.ex. månatlig basis kan uppskattas.

# Kapitel 5

# Diskussion

I detta arbete har lättrörliga- och lean metoder diskuterats, bland annat egenskapen att reagera och anpassa arbetet enligt ändrande krav är viktig i en turbulent affärsvärld. Trots många positiva egenskaper som adresserar diverse problem inom mjukvaruutvecklingen är inte lättrörliga metoder alltid det bästa valet. Ktata och Lévesque föreslår val av utvecklingsmetod beroende på omgivningens och kundens benägenhet till förändringar. I stabila omgivningar krävs inte snabb leverans eftersom användningsområdet inte ändrar [9]. Valet av utvecklingsmetod är således inte alltid självklar och användningen tycks kunna vara en blandning av flera olika metoder. I många texter förespråkas även en stegvis anpassning till lättrörliga metoder, t.ex Beck avråder från en omedelbar övergång till XP och föreslår istället inkorporering av tekniker stegvis [2].

Affärsvärde är ett välanvänt begrepp inom litteraturen. Begreppets icke påtagliga natur [11] och svårigheten att entydigt kunna definieras vid olika projekt innebär att en viss försiktighet bör iakttas hos utvecklare som vill använda affärsvärde under utvecklingsarbetet. Med dessa reservationer verkar dock affärsvärde kunna fungera som en bra influens vid mjukvaruutveckling. En vinkling på målet som affärsvärde står för, att kunna leverera värdeskapande och användbar mjukvara, diskuterades i och med kollektivt produktägarskap. Denna utvidgning av produktägarrollen framställs i litteraturen som en möjlighet till att förbättra utvecklarnas förmåga till värdeskapande och innovation. Att skapa ett djupt samarbete och delade mål som kollektivt produktägandeskap förespråkar innebär dock att en vilja för samarbete och vision finns hos alla individer i organisationen vilket kan vara sällsynt. Faktorer som kan påverka denna vilja kan tänkas vara t.ex. kortvariga anställningar och förändringar i organisationen.

Beck identifierar motsatta strävanden hos affärsintressenter och tekniska in-

tressenter [3] . Utan att använda begreppet affärsvärde föreslås istället ett *planeringsspel* för att identifiera och prioritera mål som programmet bör lösa. Planeringsspelet är en teknik som kan tänkas fungera bra om kunden med säkerhet kan specificera produkten. Detta har dock historiskt sett visat sig vara svårt och planeringsspelet ger begränsat utrymme för utvecklarna att föreslå mål och funktioner. Samarbete och kontinuerlig kundkontakt beskriver dock Beck som en viktig del i utvecklingsarbetet i form av *kund på plats*, vilket kan vara svårt att genomföra i praktiken.

Användningen av affärsvärde som grund vid prioritering och beslutsfattande uppfattas som en intressant teknik. Litteraturen ger förslag på olika tekniker för att välja och prioritera krav och funktioner ur ett monetärt perspektiv [5, 13]. Användning i rätt sammanhang och med klara definitioner kan kravgrupperingar och funktioner med monetära värden lättare prioriteras och diskuteras mellan kunder och utvecklare. Eftersom alla värden är uppskattade finns risker för feluppskattningar. Framtagandet av monetära värden ställer krav på kunden att tillhandahålla uppskattningar. Hanteringen av uppskattningarna föreslår Mary Poppendieck att skötas genom att tilldela varje team en bokförare. Mervärdet som uppskattat affärsvärde erbjuder bör alltså överstiga de medförda kostnaderna hos båda berörda parter för att vara motiverat.

Alla lättrörliga metoder presenterar tekniker och tillvägagångssätt för att skapa bättre mjukvara och förbättra utvecklingsarbetet. Dessa förfaranden skiljer sig på vissa plan medan de kan likna varandra mycket vid en djupare granskning. Förståelsen för en produkts mål får ses som en viktig del vid värdeskapande utvecklingsarbete. Affärsvärde och finansiell prioritering av krav framställs i positiv anda i befintlig litteratur men kan antagligen vidareutvecklas genom fortsatt forskning.

# Bilaga A

# Agile Manifesto

## A.1 Agile Manifesto

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

**Individuals and interactions** over processes and tools
**Working software** over comprehensive documentation
**Customer collaboration** over contract negotiation
**Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

## A.2 Manifest för Agil systemutveckling

Vi finner bättre sätt att utveckla programvara genom att utveckla själva och hjälpa andra att utveckla. Genom detta arbete har vi kommit att värdesätta:

**Individer och interaktioner** framför processer och verktyg
**Fungerande programvara** framför omfattande dokumentation
**Kundsamarbete** framför kontraktsförhandling
**Anpassning till förändring** framför att följa en plan

Det vill säga, medan det finns värde i punkterna till höger, värdesätter vi punkterna till vänster mer.

# Litteraturförteckning


[1] Beck, Beedle, vanBennekum, Cockburn, Cunningham, Fowler, Grenning, Highsmith, Hunt, Jeffries, Kern, Marick, Martin, Mellor, Schwaber, Sutherland, and Thomas. Agile manifesto. 2011.

[2] Kent Beck. Embracing change with extreme programming. *Computer*, 32:70–77, October 1999.

[3] Kent Beck. *Extreme programming explained: embrace change*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000.

[4] B Boehm. A spiral model of software development and enhancement. *SIGSOFT Softw. Eng. Notes*, 11:14–24, August 1986.

[5] Mike Cohn. *Agile Estimating and Planning*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.

[6] Alan de Ste Croix and Alan Easton. The product owner team. In *Proceedings of the Agile 2008*, pages 274–279, Washington, DC, USA, 2008. IEEE Computer Society.

[7] Ken H. Judy and Ilio Krumins-Beens. Great scrums need great product owners: Unbounded collaboration and collective product ownership. In *Proceedings of the Proceedings of the 41st Annual Hawaii International Conference on System Sciences*, HICSS '08, pages 462–, Washington, DC, USA, 2008. IEEE Computer Society.

[8] Philippe Kruchten. The elephants in the agile room. 2011.

[9] Oualid Ktata and Ghislain Lévesque. Agile development: issues and avenues requiring a substantial enhancement of the business perspective in large projects. In *Proceedings of the 2nd Canadian Conference on Computer Science and Software Engineering*, C3S2E '09, pages 59–66, New York, NY, USA, 2009. ACM.

[10] Dean Leffingwell and Don Widrig. *Managing software requirements: a unified approach*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000.

[11] Jeff Patton. Ambiguous business value harms software products. *IEEE Softw.*, 25:50–51, January 2008.

[12] Mary Poppendieck. Lean software development. In *Companion to the proceedings of the 29th International Conference on Software Engineering*, ICSE COMPANION '07, pages 165–166, Washington, DC, USA, 2007. IEEE Computer Society.

[13] Mary Poppendieck and Tom Poppendieck. *Lean Software Development: An Agile Toolkit*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.

[14] W. W. Royce. Managing the development of large software systems: concepts and techniques. In *Proceedings of the 9th international conference on Software Engineering*, ICSE '87, pages 328–338, Los Alamitos, CA, USA, 1987. IEEE Computer Society Press.

[15] Ken Schwaber and Jeff Sutherland. The scrum guide. 2010.

[16] Claes Wohlin. *Introduktion till programvaruutveckling*. Studentlitteratur AB, 2005.