Extrem Programmering

Extrem Læring?

En specialeafhandling indenfor Humanistisk Datalogi

Mads Peter Bach
Institut for Kommunikation
Aalborg Universitet

2006

Indholdsfortegnelse

Forord

I like to think (and the sooner the better!) of a cybernetic meadow where mammals and computers live together in mutually programming harmony like pure water touching clear sky.

I like to think (right now please!) of a cybernetic forest filled with pines and electronics where deer stroll peacefully past computers as if they were flowers with spinning blossoms.

I like to think (it has to be!) of a cybernetic ecology where we are free of our labors and joined back to nature, returned to our mammal brothers and sisters, and all watched over by machines of loving grace.

Richard Brautigan, 1967

Rejsen fra de første tanker som fostrede dette speciale, til det færdige produkt har været lang, spændende og under tiden også frustrerende. Jeg håber at du, læseren, efter du har læst dette speciale vil sidde tilbage med en forståelse af hvorfor jeg syntes at lige netop dette emne var spændende og relevant at underkaste en undersøgelse ud fra en humanistisk datalogisk betragtning. Allerede for snart 40 år siden beskrev Richard Brautigan en systemudviklingsmæssig utopi, hvor pattedyr og computere lever sammen i en kybernetisk harmoni og gensidigt programmerer hinanden. Vi er måske ikke nået til denne gensidige programmering, men udviklingen af computersystemer er alligevel af stor betydning for de fleste i dagens samfund, da en stor del af vores liv efterhånden er afhængige af computere i alverdens

afskygninger. Derfor mener jeg, at vi bør forsøge at nå til en klarhed omkring hvordan vores systemudviklingsmetoder fungerer, og hvordan vi agerer og lærer under vores anvendelse af disse metoder. En sådan klarhed vil, efter min, mening føre til udviklingen af bedre systemer, og dermed forhåbentlig til en bedre tilværelse for os alle. Og hvem ved? Måske når vi en dag til en kybernetisk eng, hvor vi kan leve i harmoni med vores computersystemer.

For hjælp og støtte under udarbejdelsen af dette speciale vil jeg gerne takke:
Min vejleder, Ellen Christiansen, for støtte og inspiration, ikke bare under skrivningen af dette speciale, men også tidligere i mit uddannelsesforløb.
Joakim Hjort, som hjalp med at tilvejebringe de empiriske data.
Mine kollegaer hos HUM-IKT, for forståelse og overbærenhed alle de gange jeg har omtalt dette speciale.
Min gode ven Anders Rhod Gregersen for mange frugtbare samtaler – også om alt andet end specialet, samt langtidsudlån af litteratur (jeg lover at du får dine bøger tilbage nu).

Aalborg, den 31. marts 2006

Digitally signed by Mads Peter Bach
Reason: I am the author of this document
Location: Aalborg, Danmark
Date: 2006.03.31 15:32:37 +02'00'

Mads Peter Bach

Specialet er sat med Optima, punktstørrelse 12 og fylder 127.355 tegn uden indholdsfortegnelse, bilag m.v., svarende til 53 normalsider.

Abstract

THIS masters thesis examines an important field, the field of software development, in the technology driven world of today where computers running software systems have achieved a critical role. The everyday lives of people are affected, when software is developed that doesn't fit its intended purpose. I posit that the fitness of software is very much affected by the way it is developed, and that improvements in this fitness won't come from better technology, but can be achieved when the software development process is improved. Specifically, this thesis examines the values and basic principles of the development methodology known as Extreme Programming, one of the new agile development methodologies, from a learning perspective by viewing the developers as members of a Community of Practice (CoP), as described by Etienne Wenger.

In the thesis, I attempt to determine how the perspective can contribute to our understanding of the software development process, when Extreme Programming (XP) is implemented, and if it can show aspects of the XP methodology that can be improved to facilitate improvement in the learning process.

I base my conclusion upon reflections on XP as a development methodology, and Communities of Practice as a description of how people learn in practice, and data collected by questioning experienced software developers, who use Extreme Programming to guide their development process.

I come to the conclusion that software developers using XP can viewed as a community of Practice, there is an excellent mapping between the values of XP and CoP, and that the CoP viewpoint does point towards a number of areas in the XP methodology that can probably be improved, by better facilitating the learning process, and hopefully thereby improving the fitness of software developed using the methodology.

Indledning og problemformulering

"The most important thing any teacher has to learn, not to be learned in any school of education I ever heard of, can be expressed in seven words: Learning is not the product of teaching. Learning is the product of the activity of learners."

John Caldwell Holt

LÆRING og undervisning bliver af nogen betragtet som to sider af samme sag. Man ser også holdningen, at de bedste betingelser for læring findes i en stereotyp undervisningssituation, hvor hvert enkelt lærende individ opnår sin læring i ensomhed. Sat op mod en betragtning om mennesket som et socialt væsen (Wenger 1998, s. 4), forekommer denne fokus på undervisningen af individer som malplaceret. Med tanken om, at læring som en social proces ikke nødvendigvis foregår på skolebænken, kan vi måske i højere grad dreje fokus over på den læring der foregår i det almindelige arbejdsliv, i et arbejdsfællesskab.

En måde at forstå et arbejdsfællesskab på er at betragte det som blandt andet konstitueret og propageret gennem den læring der foregår i denne praksis.

Et eksempel på et arbejdsfællesskab kunne være et fællesskab der producerer computersoftware. En af de væsentlige måder man kunne beskrive sådan et fællesskab på, ville være at beskrive den udviklingsmetode som blev anvendt, da den må antages at have en overordentlig stor indflydelse på praksis i et udviklingsforløb. Ligeledes må udviklingsmetoden antages at være et af de primære teoretiske

fundamenter og hjælperedskaber der anvendes i forbindelse med ændring og forbedring af udviklingsprocessen.

Igennem de senere år har der været stor interesse for en ny type af systemudviklingsmetoder. Disse metoder kaldes med en fælles betegnelse for adrætte systemudviklingsmetoder, og en af de mest kendte repræsentanter for disse kaldes Extreme Programming. Et kritikpunkt man kunne rejse overfor Extreme Programming som metodologi er, at store dele af litteraturen omkring den er meget anvendelsesorienteret. Der findes et antal artikler omkring anvendelsen af Extreme Programming som et undervisningsværktøj, i forbindelse med undervisning i systemudvikling, men den læring der foregår i disse forløb er primært beskrevet ud fra et undervisningssynspunkt og er ofte ikke sat ind i en forståelsesmæssig ramme.

Mit projekt i nærværende speciale bliver dermed:

En undersøgelse, ud fra social læringsteori, af den læring der bør og kan foregå i et systemudviklingsforløb, der er baseret på Extreme Programming, med henblik på at afklare om denne forståelse af læreprocesserne kan bidrage til en bedre implementering af XP som systemudviklingsmetodologi..

Dette vil jeg forsøge at afklare, dels ved at betragte teorien omkring XP gennem en optik dannet ud fra social læringsteori, dels ved kvantitativt og kvalitativt at indsamle erfaringer fra udøvere af Extreme Programming, og sætte disse erfaringer op mod mine teoretiske betragtninger.

Udvikling med Extreme Programming

Make it as simple as you can, but no simpler.

Albert Einstein

For at kunne foretage en analyse af den læring der foregår i et systemudviklingsforløb hvor Extreme Programming anvendes, er det nødvendigt først at gennemgå de elementer som konstituerer praksis i XP. Endvidere vil jeg gennemgå de værdier som ligger bag praksis i XP, da alle deltagerne i et systemudviklingsforløb skal tage disse værdier til sig, for at der kan blive tale om XP i praksis. Jeg vil dog begynde med først at skitsere baggrunden som XP opstod ud af, og hvordan denne baggrund adskiller sig fra traditionel systemudvikling.

Set historisk

EXTREME Programming (XP) er det man i dag betegner som en adræt system-udviklingsmetodologi. Andre eksempler på denne type af metodologier er *The Crystal Methods, Lean Development, Scrum* og *Evolutionary Development*[*], men XP er den mest kendte (Noble *et al* 2004, s. 1). Tidligere har denne type af systemudviklingsmetodologier også været kendt som letvægts systemudviklingsmetodologier, som en kontrast til de "tunge" traditionelle stive og dokumentorienterede metodologi, men ved et møde i Utah i 2001 blev mange af drivkræfterne bag de adrætte metodologier enige om et manifest for adræt systemudvikling, som beskriver det fælles

* Disse metodologier blev for alvor formuleret og populariseret i årene omkring årtusindeskiftet, men elementer af dem har været anerkendt og brugt langt tidligere.

værdigrundlag, som alle disse metodologier er baseret på:

> *" 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."*

(Beck *et al* 2001)

Adræt systemudvikling er et mere dækkende navn end letvægtssystemudvikling, da udvikling efter disse metodologier, ikke nødvendigvis kræver færre ressourcer end udvikling efter en traditionel model. Navnet "Adræt systemudvikling" understreger derimod, at disse metodologier i højere grad lægger vægt på en udviklingsproces, som ikke er fastlåst, men hvor mål og midler let kan tilpasses undervejs i forløbet. At dette kan være en afgørende fordel for en metodologi behøver man ikke kigge udenfor Danmarks grænser for at konstatere, selv om det er der disse metodologier er udviklet. I Danmark har vi glimrende eksempler på systemer, f.eks. virksomhedsportalen Virk. dk eller arbejdsformidlingens Amanda hvor udviklingsprocessen tilsyneladende ikke kunne tilpasse sig de ændrede krav der opstod undervejs i forløbet, med forsinkelser og budgetoverskridelser til følge (Hansen 2005). Ud fra en simpel betragtning kan man opstille fire kriterier for et succesfuldt systemudviklingsforløb:

1. Systemet bliver færdigt til tiden
2. Systemet koster ikke mere end der oprindeligt var planlagt

3. Systemet opfylder de kvalitetskrav der blev stillet
4. Systemet indeholder den aftalte funktionalitet

Hvis det undervejs i processen viser sig, at det ikke kan lade sig gøre at opfylde alle fire kriterier, så vil det i en traditionel udviklingsproces typisk være svært at ændre specielt det sidste kriterium, hvorfor man oftest ser at de første to kriterier ikke bliver opfyldt. Ifølge Barry Boehm (Vanderburg 2005, s. 542) stiger omkostningerne ved at foretage forandringer i løbet af en systemudviklingsproces jo længere fremskredet den er. Omkostningerne ved forandring stiger eksponentielt, og derfor er de traditionelle systemudviklingsmetoder baseret på, at forandringer skal foretages så tidligt som muligt i forløbet, for at holde omkostningerne nede. Boehm baserede primært sine undersøgelser på projekter der fulgte den traditionelle vandfaldsmodel. Adrætte processer ser derimod ud til at have en meget fladere omkostningsstigningskurve, og man kan derfor postulere, at Boehms undersøgelse i virkeligheden viste hvordan omkostningernes stigning er afhængig af længden på iterationerne i udviklingsforløbet (Vanderburg 2005, s. 542).Med en adræt systemudviklingsmetode er iterationerne korte, og der lægges op til, at i hvert fald tre af de fire kriterier kan tilpasses løbende (tid, pris og funktionalitet). Dermed bliver det muligt for kunden at afgøre hvordan der skal prioriteres, i stedet for at processen er fastlåst fra det øjeblik der er lavet en kravspecifikation, til systemet bliver testet.

Hvad er det så, der gør adrætte udviklingsmetoder bedre i stand til at tilpasse sig ændrede krav, i forhold til en traditionel faseopdelt udviklingsmetodologi? De to vigtigste grundsten er vægtning af fleksibilitet og en iterativ tilgang til udviklingen. Den efterfølgende beskrivelse af XP vil fokusere på hvordan XP implementerer disse.

XP som eksempel på en adræt metodologi

Tre aspekter der beskriver XP som metodologi er værdier, principper og udviklingspraksis.

Værdier

Extreme Programming er baseret på fire værdier, som udviklerne skal tage til sig, for kunne bringe de 13 hovedprincipper som XP er baseret på i anvendelse (Beck 2000, s. 29):

1. Kommunikation
2. Simpelhed
3. Tilbagemelding (feedback)
4. Mod

Kommunikation

Selve grundpillen i XP er god kommunikation. Uden den kan ingen af de praksiser som XP foreskriver anvendes med held. En af antagelserne bag XP er, at alle problemer i et projekt kan føres tilbage til mangelfuld kommunikation mellem de involverede parter (Beck 2000, s. 29-30). Dette forsøger XP at afhjælpe, ved at introducere en rolle som coach, som har til opgave at facilitere kommunikation ved at sørge for at ingen af parterne holder op med at kommunikere. Ved lægge vægt på de processer hvor kommunikation er uafvendelig, forsøger XP at minimere risikoen for at manglende kommunikation bliver et problem i processen. Disse processer er bl.a. par programmering, hvor kommunikationen foregår mellem de to udviklere der arbejder sammen, og planlægning sammen med den repræsentant for kunden der er til stede hos udviklerne.

Simpelhed

At prioritere simpelhed i et udviklingsforløb betyder ikke, at kun simple opgaver kan løses. Det betyder i højere grad, at udviklerne skal fokusere på de opgaver der skal løses lige nu og her, og ikke bruge en masse energi på at tænke over hvordan en given softwarekomponent måske skal kunne anvendes i fremtiden (Beck 2000, s. 30-31). Tanken er, at hvis der er brug for ændringer senere, så er det bedre at lave disse ændringer senere, i stedet for at forsøge at tage højde for noget der måske slet ikke bliver nødvendigt. Ved at følge denne strategi formindskes kompleksiteten af systemet, og XP undgår spild af ressourcer.

Tilbagemelding (feedback)

Der menes to forskellige ting med tilbagemelding i XP sammenhæng. Den første er, at når en situation ændrer sig, så har man som deltager i en XP proces pligt til at kommunikere denne ændring til de parter der berøres af den (Beck 2000, s. 31). Det kunne for eksempel være at et udviklerpar finder ud af, at de har undervurderet hvor lang tid der vil være nødvendig for at implementere en given funktionalitet. Dette melder de tilbage til projektledelsen og kunderepræsentanten hurtigst muligt, så planen kan ændres. Kunden kan for eksempel beslutte at udskyde implementeringen af denne funktionalitet til en senere iteration. Ligeledes er kunderepræsentanten forpligtet til snarest muligt at melde tilbage, når vedkommende opdager en del af systemet der ikke lever op til de opstillede krav. Det betyder også at test af systemet prioriteres høj, så udviklerne får disse tilbagemeldinger hurtigst muligt. Fra udviklernes side prioriteres test af systemet lige så højt, hvis ikke højere end udvikling af ny funktionalitet:

"(...) until the tests run, you're not done, and when the tests all

run, you're done."

(Beck 2000, s. 33)

Ved test forstår man her både formel testning af systemet, udført af udviklerne, ved hjælp af test cases, og brugernes løbende brug af simple udgaver af systemet, efterhånden som de bliver færdige (Beck 2000, s. 32). Dette, koblet med XP's iterative natur, gør at systemet løbende kan forandres under udvikling, efterhånden som brugerne forstår hvilke krav de i virkeligheden har til systemet.

Mod

At opfordre til at udviklerne skal have mod til at eksperimentere, og skal turde træffe beslutninger, uden at gennemløbe en formel proces kan virke som en noget risikabel opfordring. Hvordan kan man sikre at udviklerne træffer de rigtige valg, og har udviklerne i det hele taget de nødvendige forudsætninger for at træffe de rigtige valg? Her kommer de tre tidligere nævnte værdier ind i billedet. Når udviklerne kommunikerer om deres eksperimenter, holder dem så simple som det er nødvendigt, og får tilbagemeldinger både i form af automatiske tests, og fra brugerafprøvning, så kan eksperimenter foretages uden at stabiliteten af systemet kommer i fare (Beck 2000, s. 34). Et andet aspekt af mod, er modet til at forkaste resultatet af et eksperiment:

"(...) maybe you have three design alternatives. So, code a day's worth of each alternative, just to see how they feel. Toss the code and start over on the most promising design."

(Beck 2001, s. 33)

At man har mulighed for at forsøge at ændre systemet radikalt hele vejen gennem udviklingsprocessen, giver den fleksibilitet som adskiller XP og dermed adræt systemudvikling fra et traditionelt udviklingsforløb.

Principper i XP

Princippernes rolle i et XP forløb, er at hjælpe udviklerne med at træffe det bedst mulige valg, når de har en række valgmuligheder (Beck 2000, s. 37). Disse principper inkorporerer de fire værdier, men er mere direkte forbundet til udviklernes praksis.

Hovedprincipperne er (Beck 2000, s. 37):

- Hurtig feedback
- Antag simpelhed
- Forandring skal være trinvis
- Tag forandring til dig
- Udfør kvalitetsarbejde

I XP metodologien er også en række mindre essentielle principper, som hjælper udviklerne med at beslutte hvad der skal gøres i specifikke situationer. Hvis vi ser bort fra de principper som kun har ophandler den tekniske del af systemudvikling, så er de (Beck 2000, s. 39):

- Undervis i læring
- Beslut ud fra konkrete eksperimenter
- Kommuniker åbent og ærligt
- Arbejd med folks instinkter, i stedet for imod dem
- Accepteret ansvar

Hurtig tilbagemelding

Dette bygger over på tilbagemelding som værdi. I XP understreges at tilbagemeldingen skal komme så hurtigt som muligt for at være mest effektiv (Beck 2000, s. 37).

Antag simpelhed

Endnu en understregning af, at udviklerne skal antage at problemet der skal løses, kan løses simplest muligt. XP postulerer at dette vil være korrekt i stort set alle tilfælde, og at den tid man sparer ved at antage det, vil være mere end den ekstra tid man kommer til at bruge i de få tilfælde hvor det viser sig ikke at være så simpelt (Beck 2000, s. 38).

Forandring skal være trinvis

Dette er en af nøglerne til muligheden for at eksperimentere i XP. Tilrettelæg udviklingen så du altid kan gå et lille trin tilbage, i stedet for en stor forandring, der har krævet en masse ressourcer, og derfor som udviklerne derfor nødig vil opgive. Med små forandringer er der mindre der skal opgives, hvis forandringen ikke virker.

Tag forandring til dig

Når du står for at skulle vælge mellem alternativer, så er tanken i XP at det alternativ der giver dig flest forskellige muligheder efterfølgende oftest vil være det bedste (Beck 2000, s. 38).

Udfør kvalitetsarbejde

For at opnå den størst mulige tilfredshed blandt udviklerne skal man lade dem udføre kvalitetsarbejde, i stedet for at insistere på løsninger der er for hurtige eller billige, da dette vil føre til en dårlig spiral for projektet (Beck 2000, s. 38)

Undervis i læring

I stedet for eksempelvis at fastlægge stramme normer for hvordan tests skal udføres, og hvor mange der skal være, så skal man lade det være

op til udviklerne at finde ud af hvordan og hvor meget der er nødvendigt (Beck 2000, s. 39). Dette gælder også for andre aspekter af udviklingen som design og refaktorering.

Beslut ud fra konkrete eksperimenter

Hvis en beslutning træffes uden en test, så er der en risiko for at denne beslutning er forkert. Derfor bør beslutninger ikke træffes uden at afprøve de forskellige alternativer (Beck 2000, s. 40).

Kommuniker åbent og ærligt

Dette virker som en selvfølge, hvilket Kent Beck også kommer ind på (Beck 2000, s. 40), men det er ment som en understregning af det det er vitalt for udviklingsprocessen at både positive og negative erfaringer, nyheder o.s.v. kan fortælles uden at det får negative konsekvenser for budbringeren.

Arbejd med folks instinkter, i stedet for imod dem

Hermed menes der, at den bedste strategi oftest er, at følge det der virker som den bedste ide på kort sigt, i stedet for at forsøge at overbevise udviklere om fordelen ved at udføre noget på en mere besværlig måde, som må vil give bedre resultater på langt sigt (Beck 2000, s. 41). Dette kan umiddelbart lyde som en dårlig måde at opnå gode resultater på langt sigt, men koblet med blandt andet den store vægt der lægges på at alting skal testes, og at underviklerne skal have lov til at udføre kvalitetsarbejde (se også det næste punkt), så postuleres det at det ikke giver problemer på langt sigt.

Accepteret ansvar

Ansvar kan ikke pålægges, det skal accepteres af den enkelte person (Beck 2000, s. 41). Det betyder også at arbejdsopgaver ikke pålægges, men at man i stedet lader udviklerne vælge hvilke opgaver de ønsker at løse. Det betyder ikke, at upopulære opgaver efterlades uløst, da medlemmerne af et udviklingshold har ansvaret for at føre de beslutninger de træffer ud i livet. Hvis et udviklingshold bliver enige om at en opgave skal udføres, så skal et af holdets medlemmer påtage sig det. Hvis ingen vil det, så kan beslutningen om at opgaven skal udføres ikke tages.

Udvikling med XP

Systemudvikling med XP består basalt set af fire aktiviteter (Beck 2000, s. 44 ff):

1. At programmere
2. At teste
3. At lytte
4. At designe

Aktivitet nr. 3 virker ikke som en oplagt selvstændig aktivitet, og det er heller ikke sådan den skal betragtes under XP, men da kommunikation er vitalt for udviklingsprocessen nævnes den som en selvstændig aktivitet.

At programmere

Dette er naturligvis en af de vigtigste aktiviteter i et systemudviklingsforløb. Uden programmering kommer der ikke noget produkt ud af forløbet, og i dette adskiller XP sig ikke fra andre metodologier. Men et af målene

med at programmere i XP sammenhæng er også læring:

> *"Code also gives you a chance to communicate clearly and concisely. If you have an idea and explain it to me, I can easily misunderstand. If we code it together, though, I can see in the shape in the logic you write the precise shape of your ideas. Again, I see the shape of your ideas not as you see them in your head, but as they find expression to the outside world. This communication easily turns into learning."*

(Beck 2000, s. 44)

At teste

Automatiseret testning er vigtigt for systemudvikling, blandt andet for at kunne påvise at funktionalitet findes og fungerer som den skal. I mange traditionelle systemudviklingsforløb foregår testning manuelt, og foregår ikke løbede i forløbet, eller måske i en anden afdeling. XP understreger vigtigheden af løbende tests, og også test af andet end funktionalitet, som for eksempel at afviklingshastigheden er som forventet, og at standarder følges.

Automatiske tests er en af de ting der giver mulighed for forandring af systemet, da de, hvis de tester systemet grundigt nok, kan bruges til at påvise at systemet stadig fungerer som det skal efter en ændring.

Det postuleres også, at løbende testning øger udviklingshastigheden (Beck 2000, s. 46), fordi man ikke er nødt til at bruge så meget tid på fejlsøgning, når systemet ikke virker efter man har lavet en ændring.

Testning skal også afspejle de krav som kunden har til systemet, så kunden kan have tillid til at systemet virker som det er forventet.

At lytte

En vigtig pointe som understreges i XP (Beck 2000, s. 47), er at programmører ikke alene har tilstrækkelig viden om det problemfelt som systemet bliver udviklet for at løse. Derfor er de nødt til lytte til

og acceptere de svar de får, fra eksperterne indenfor feltet, som typisk er kunden. De er til gengæld også nødt til at fortælle kunden, hvad der er svært og let, så kunden kan træffe de rigtige valg med hensyn til prioritering af udviklingsressourcer.

At designe

For at et system ikke skal blive uoverskueligt, efterhånden som det vokser sig større, er det nødvendigt at træde et skridt tilbage fra kildekoden, og bruge tid på at omstrukturere logikken i systemet, så det bliver simplere og nemmere at forstå, og udviklingen kan fortsætte.

Udviklingspraksis

Den udviklingspraksis som XP beskriver strukturerer de fire aktiviteter, ud fra de værdier og principper som vi har gennemgået. Praksis består af følgende

- Planlægningsspillet
- Små lanceringer
- Systemet som metafor
- Simpelt design
- Testning
- Refaktorering (refactoring)
- Par programmering
- Fælles ejerskab
- Fortløbende integration
- Mindst muligt overarbejde
- Kunden tilstede
- Programmeringsstandarder

Jeg vil her kort gennemgå de dele af praksis der ikke tidligere er omtalt.

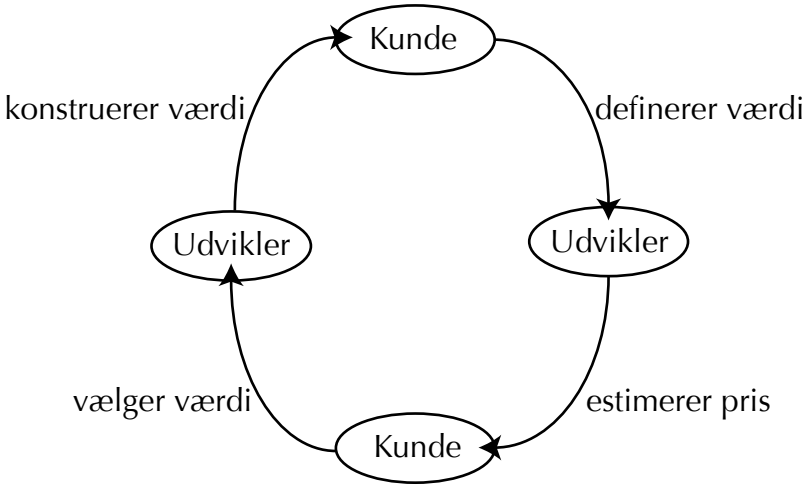
Planlægningsspillet

> *"Neither business consideration nor technical considerations should be paramount. Software development is always an evolving dialog between the possible and the desirable."*
>
> (Beck 2000, s. 55)

I XP understreges nødvendigheden af, at både kunden og udviklerne løbende får den information der er nødvendig, for at de kan træffe de rette valg. Kunden fortæller i hvilken rækkefølge de ønsker funktionaliteten implementeret, udviklerne estimerer hvor mange ressourcer der skal bruges for at implementere denne, og kunden har så mulighed for ændre sin prioritering både af rækkefølge og af den ønskede funktionalitet. I modsætning til mange andre udviklingsmetodologier er dette en løbende balanceringsproces, ikke en beslutning der foretages før udviklingen går i gang.

Et XP udviklingsforløb er dermed cyklisk, forstået på den måde at det ganske vist starter med en specifikation af krav, efterfulgt af implementering og test, men dette sker ikke kun en gang i løbet af et udviklingsforløb. Et XP forløb har mange iterationer, der alle er bygget over følgende model:

Figur 1, efter (Jeffries *et el* 2001, s. 15)



Her skal pris ikke forstås som en pris i kroner og ører, det er en pris i udviklingsressourcer (tid). Iterationer afvikles over en kort tidshorisont, så kunden ikke er låst fast af et første valg, men kan omprioritere undervejs, hvis det for eksempel viser sig at udviklingen ikke skrider frem med den forventede hastighed (Jeffries *et al* 2001, s.159). Kunden definerer hvad der skal udvikles ved hjælp af historier. Historier beskriver hvilken funktionalitet systemet skal have, med en varierende detaljeringsgrad. En historie kunne være:

> *"For at beregne saldoen på en konto lægges alle indlån sammen, og alle udlån trækkes fra".*

En historie bliver typisk til en enkelt softwarekomponent, som udvikles og testes af et par programmører i løbet af en iteration.
De detaljerede historier gør det muligt for udviklerne at komme med ret præcise estimater af hvor mange udviklingsressourcer en historie vil kræve at implementere.
Ud fra disse estimater prioriterer kunden hvilke rækkefølge historierne skal udvikles i. En iteration resulterer typisk i en lancering (release) af systemet med den funktionalitet der er tilføjet i løbet af iterationen. Ideen er, at en sådan lancering skal kunne tages i brug af kunden, så kunden umiddelbart kan konstatere at de får noget for deres penge, i stedet for først at se et helt færdigt system i slutningen af udviklingsprocessen (Jeffries *et al* 2001, s. 50). Det gør også at selv hvis den oprindelige tidsplan overskrides, så har kunden et fungerende system der kan tages i brug, slev om ikke al funktionalitet er implementeret.

Små lanceringer

Små lanceringer giver kun mulighed for små forsinkelser. Dette er et af de vigtige værktøjer, som er med til at holde et udviklingsforløb baseret

på XP på sporet. Disse små lanceringer giver kunden mulighed for at melde tilbage, og ændre planen for udviklingsforløbet.

Systemet som metafor

En praksis som skal hjælpe udviklerne i at forstå systemet, er en overordnet beskrivelse ved hjælp af metaforer. Et eksempel kunne være at brugeren oplever systemet som et skrivebord (Beck 2000, s. 56). Beskrivelsen ved hjælp af metafor er simpel, og er dermed nem for at holde in mente når der skal træffes beslutninger i løbet af udviklingen.

Refaktorering

Refaktorering er en cyklisk oprydningsproces i udviklingsforløbet. Før ny funktionalitet implementeres overvejer udviklerne og systemet kan ændres, så det bliver nemmere at tilføje den nye funktionalitet. Når udviklerne så har tilføjet den nye funktionalitet, skal de overveje om denne funktionalitet gør det muligt at gøre systemet simplere, uden at de eksisterende tests holder op med at fungere. Denne oprydning er med til at holde systemet overskueligt, og ikke risikobetonet, fordi de automatiske tests skal opdage uforudsete bivirkninger af refaktoreringen.

Par programmering

Par programmering er ikke nyt*, men mange støder første gang på det i forbindelse med XP, og det er derfor blevet et af XP's kendetegn. Par programmering foregår ved at to udviklere arbejder sammen ved en arbejdsstation, den ene "fører", og skriver den kildekode der skal realisere den ønskede funktionalitet, mens den anden tænker mere

* Edsgar Djikstra beskriver hvordan han anvendte en form for par programmering i forbindelse med en implementering af Algol60 (Djikstra 2001, s. 5)

overordnet og strategisk, og overvejer om det er den korrekte måde at komme til det ønskede mål. Når vedkommende føler at de valg der bliver truffet ikke er de rigtige, eller ikke forstår hvorfor disse valg træffes, så er spørger vedkommende ind til begrundelsen for valget, således at opnås en fælles forståelse, og der bliver mindre risiko for at en systemfejl introduceres i kildekoden. Programmering forvandles via par programmering fra en soloaktivitet, hvor kommunikation og social læring i bedste fald er sekundære aktiviteter, til en social proces, hvor kommunikationen og læring er i højsædet. Da to udviklere sjældent vil tænke ens, og have den samme viden og forudsætninger, så vil der hele tiden opstå situationer hvor deres perspektiver på løsningen af en opgave ikke er ens, og begge får gennem kommunikationen mulighed for at forandre deres egen forståelse af problemet og dets løsninger.

Fælles ejerskab

> *"I'm not afraid to change my own code. And it's all my own code."*
>
> (Jeffries *et al* 2001, s. 75)

I modsætning til både den meget formelle ide om at et stykke kildekode har en ejer, og at ejeren er den eneste der kan ændre dette stykke kode, og det anarki det kan opstå hvis alle uden videre ændrer i et hvilket som helst stykke kode, uden at føle noget ansvar for hvilke følger det har, prøver XP at ramme en gylden middelvej. I XP har alle ret til at ændre i al kildekode, og har faktisk pligt til at implementere de forbedringsmuligheder de opdager. Da alle har taget ansvaret for stabiliteten af hele systemet, bør dette ikke resultere i en ustabil kodebase. Her er de automatiske tests også med til at fjerne en stor del af risikoen, da en fejlbehæftet ændring, som beskadiger systemet bør udløse en testfejl. Ligeledes fører det fælles ejerskab af koden, og det fælles ansvar for systemets stabilitet, til en mindsket tendens til at

indføre utestet eller uoverskuelig kode, dels fordi andre har mulighed for at se den kode man introducerer, dels for ikke selv at komme ud for uoverskuelige kode fra de andre udviklere (Beck 2000, s. 99)

Fortløbende integration

Ved at integrere ændringer i den eksisterende kodebase løbende, forsøger XP at minimere problemer med, at den del af systemet der er under udvikling kommer til at afvige mere og mere fra det kørende system. Når den nye eller ændrede kode bliver integreret i den kodebase som det kørende system genereres ud fra, så skal de tests der køres på den eksisterende kodebase vise de problemer der eventuelt måtte opstå. En integration er ikke komplet, før alle tests afvikles uden fejl.

> *"Integrating one set of changes at a time works well because it is obvious who should fix a test that fails – we should, since we must have broken it, since the last pair left the tests at 100%. And if we can't get the tests to run at 100%, we should throw away what we did and start over"*

(Beck 2000, s. 60)

Her understreges igen hvor vigtig det er med små trinvise ændringer til systemet, så der ikke er spildt mange ressourcer, hvis det er nødvendigt at kassere et sæt af ændringer til kodebasen.

Mindst muligt overarbejde

> *"We worked really hard for a few weeks. (...) We looked upon the work we had done in the overtime period (...) We found poorly written code, we found untested code, and we found un-refactored code."*

(Jeffries *et al* 2001, s. 81)

Hvis overarbejde er nødvendigt, så peger det på at der er et alvorligt problem med udviklingsprojektet. Udviklere kan ikke i længden blive ved med at arbejde for meget, og stadig være kreative og omhyggelige.

Reglen i XP regi er derfor, at man ikke kan have overarbejde mere end en uge i træk. Hvis det virker nødvendigt for at nå de planlagte mål, så er det nødvendigt at ændre disse mål, i samråd med kunden. Hermed stræber XP dels efter at alle skal glæde sig til at komme tilbage på arbejde, i stedet for at være trætte og nedslidte på grund af for mange timer på arbejdet, dels efter at kvaliteten af arbejdet skal være god.

Kunden tilstede

En anden meget utraditionel praksis som XP foreskriver, er at en af de kommende brugere af systemet altid skal være til stede, så udviklerne kan konsultere med vedkommende når de har brug for input til at foretage en prioritering, eller det har brug for mere viden om et bestemt brugsscenarie. På den måde kan man hævde, at udvikling med XP kan minde om et kontinuerligt prototypeforløb, hvor udviklerne hele tiden får feedback på det de udvikler af de kommende brugere. Dette virker umiddelbart som om det kræver flere ressourcer fra kunden, i forhold til et traditionelt udviklingsforløb, og det kan derfor blive et problem, hvis kunden ikke opnår en forståelse af hvorfor det er nødvendigt, men systemet skulle gerne kunne tages gradvist i brug, efterhånden som de små kontinuerte lanceringer opnår mere og mere funktionalitet, og kunden vil dermed forhåbentlig kunne se at de ressourcer der bruges på at have en repræsentant til stede, hurtigt bliver tjent ind ved at have et kørende system der hele tiden får mere og mere funktionalitet.

Programmeringsstandarder

"I can always read my own code. But wait, it's all my own code."

(Jeffries *et al* 2001, s. 79)

At foreskrive standarder for hvordan systemet skal programmeres, kan umiddelbart virke som en kontrast til den vægt der ellers i XP lægges på

fleksibilitet og mulighed for forandring. Standarder er dog nødvendige, hvis det fælles ejerskab af kode og system skal blive en realitet, da det ellers ikke vil være muligt for alle udviklere at ændre i alle dele af hele systemet (Beck 2000, s. 61). XP lægger dog vægt på, at det er udviklerne der skal blive enige om hvilke standarder der er nødvendige, hvordan de skal implementeres, og at et af målene for de standarder man bliver enige om er, at de ikke må kræve en stor indsats at overholde. Dette falder godt i tråd med princippet om at arbejde med udviklernes instinkter, i stedet for imod dem.

Problematisering af XP som systemudviklingsmetodologi

Der bliver i dette speciale ikke taget stilling til hvornår XP er en hensigtsmæssig metodologi at anvende, men udviklerne bag XP påpeger selv en række tilfælde, hvor XP ikke er den mest hensigtsmæssige metodologi at anvende. Det drejer sig primært om virksomheder, hvor virksomhedskulturen ikke understøtter, eller vil være kritisk overfor dele af den praksis som gør XP effektiv. Det kunne for eksempel være en virksomhed, hvor det forventes af medarbejderne, at de har en lang arbejdsdag, for at vise deres engagement (Beck 2000, s. 156). Andre eksempler er miljøer, hvor det er nødvendigt at tage hensyn til eksisterende systemer der ikke kan ændres. Det kan føre til uhensigtsmæssig kompleksitet i systemet, og mindske eller eliminere den fleksibilitet, der er en af XP's forcer. XP er efter ophavsmændenes mening heller ikke vilkårligt skalerbar i antallet af programmører på et udviklingsprojekt. De skriver at ti absolut er muligt, men at tyve nok er for mange. Dette skaber en naturlig grænse for størrelsen af de systemer der kan udvikles af et enkelt udviklerhold (Beck 2000, s. 157). Indenfor disse rammer kan der dog stadig udvikles ganske store systemer, og flere udviklerhold kan samarbejde, hvis de systemer de udvikler kun er

koblet løst sammen. En anbefaling i forbindelse med implementering af større systemer er, at lade udviklerholdene knopskyde, således at man starter med et hold af 10 udviklere, som udvikler systemet op til den første lancering, for derefter at dele holdet op i to nye hold, og lade flere udvikle tilgå disse to hold. Efter den anden lancering gentages processen, og den forventes at kunne gentages 1-2 gange (Beck og Fowler 2000, s. 118).

Under en uformel samtale med en erfaren systemudvikler, der tidligere havde anvendt XP som metode, havde jeg lejlighed til at spørge om hvorfor hans virksomhed ikke længere anvendte XP på de projekter, som han er tilknyttet. Han nævnte årsager som, at der ikke var tilstrækkelig mulighed for at tilpasse eksisterende systemer, og at tilbagemeldingshastigheden på ændringer i systemet var for lav til at tilbagemeldingerne gjorde meget nytte. Ligeledes nævnte han, at de havde bedre mulighed for at lokalisere fejl uden XP, gennem at køre det system der var under udvikling i paralleldrift med det eksisterende, og sammenligne uddata fra de to systemer automatisk. I virksomheden bliver XP dog stadig benyttet, men stort set kun til udvikling af nye, mindre, systemer, og da deres forretningsprofil bevæger sig væk fra at levere den type af systemer, så bliver XP brugt mindre og mindre. Som et kuriosum kan det nævnes, at de systemer han arbejder på, primært er lønsystemer, hvilket er tankevækkende da XP netop blev grundlagt i forbindelse med udviklingen af et lønsystem hos Chrysler. Dette viser, at det nok i lige så høj grad er virksomhedskulturen som typen af systemer, der kan frembyde problemer for anvendelsen af XP som metodologi.

Også andre har påpeget svagheder i XP, og i særdeleshed den tendens der har været i nogle kredse til at se XP som løsningen på alle problemer

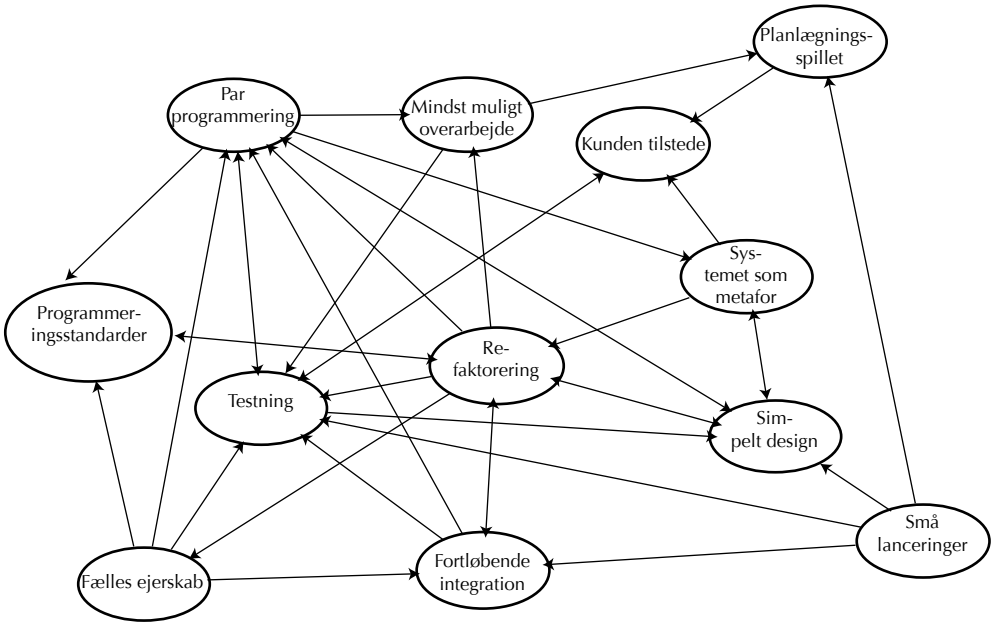
med udvikling af softwaresystemer. Se (Boehm og Turner 2003) samt (Stephens og Rosenberg 2003).

Man kan måske endda sige at den store popularitet som XP har nydt, på længere sigt kan komme den til skade, hvis den bliver anvendt i sammenhænge hvor den ikke er velegnet, og hvor man måske i stedet burde have brugt en traditionel metodologi, som f.eks. vandfaldsmodellen (Pressman 1997), eller en mere formaliseret ramme for sikring af kvalitet som f.eks. The Capability Maturity Model (Paulk *et al* 1993).

Opsummering

I dette afsnit er specielt de elementer der adskiller XP fra en mere traditionel udviklingsmetodologi blevet gennemgået.

En visualisering af de vigtigste elementer i XP's praksis, og deres indbyrdes afhængigheder kan se således ud:



Figur 2, inspireret af (Beck 2000, s. 70)

Figuren viser både envejs afhængigheder, "Fælles ejerskab afhænger af Programmeringsstandarder" og tovejs afhængigheder "Refaktorering

afhænger af Simpelt design, og Simpelt design afhænger af Refaktorering". Afhængigheder mellem elementerne afspejler steder hvor kommunikation foregår, eller hvor udviklernes handlerum modereres af det sociale praksisfællesskab. Det er således tydeligt, at XP adskiller sig fra traditionelle systemudviklingsmetodologier ved den vægt der ligger på det kommunikative og sociale aspekt, gennem XP's værdier, principper og praksis.

Læringsforståelse

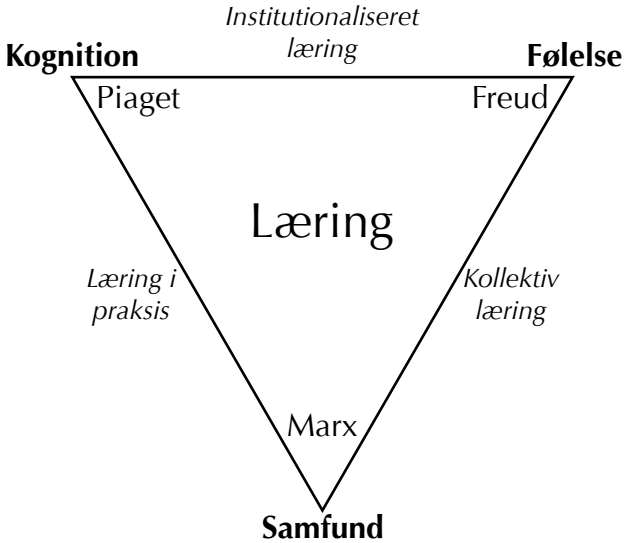
A learning experience is one of those things that says, "You know that thing you just did? Don't do that."

Douglas Adams

FØR jeg står med et værktøj, som kan anvendes til forståelse af læring i systemudvikling, er det nødvendigt at argumentere for hvordan læring skal forstås teoretisk i genstandsfeltet, systemudvikling med den adrætte systemudviklingsmetode Extreme Programming, og hvorfor denne forståelse af læring er den mest passende til lige netop dette formål.

En interessant måde at betragte det både dybe og brede felt af læringsteorier- og forståelser foreslås af Knud Illeris i bogen *"The Three Dimensions of Learning"* (Illeris 2002). Illeris forsøger i denne bog at sammentænke de vigtigste læringsforståelser, og forholde dem til hinanden. Som navnet på bogen antyder, så ser Illeris læring ud fra tre forskellige poler:

Figur 3 efter (Illeris 2002, s. 19 og 237)



Illeris postulerer, at læring er en integreret proces, der er konstitueret af to delprocesser, der gensidigt påvirker hinanden (Illeris 2002, s. 19). De to delprocesser er henholdsvis den lærendes interaktion med det omgivende miljø, og den interne proces i den lærende, som fører til læring.

Da den type af læring som jeg vil studere foregår i et praksisfællesskab, så virker det oplagt at tage udgangspunkt i at afsøge det område der fokuserer på den lærendes interaktion med det omgivende miljø, altså området mellem den kognitive og samfundsmæssige læringsforståelse, mere specifikt omkring læring i praksis. Jeg vil dog kort argumentere for hvorfor jeg ikke mener at polerne i nogen de tre dimensioner er den rette forståelsesramme, til vores formål, men at elementer fra disse dimensioner, specielt den kognitive vil være med til at supplere teorien om praksisfællesskaber, og vil slutteligt give os et godt forståelsesredskab.

Den kognitive læringsdimension

Illeris beskriver den kognitive læringsforståelse, med udgangspunkt i Piagets teorier om læring. Meget af det arbejde der er blevet gjort indenfor denne forståelse beskæftiger sig med læring set ud fra den udvikling mennesker går igennem, fra barn til voksen (Illeris 2002, s. 28-32). Dette aspekt er ikke interessant for denne sammenhæng, da de lærende individer som vi ønsker at forstå er voksne mennesker, der lærer i deres arbejdsliv. Et andet aspekt af denne forståelse er selve de mentale og psykologiske processer der finder sted mens læring foregår. Dette aspekt kunne være interessant at inddrage, men det ville kræve en anden indgangsvinkel til indsamlingen af empiri, og derfor vælger

jeg også at se bort fra dette aspekt. Man kan dog se David Kolbs teori om "Experiental Learning" som en videre udvikling blandt andet af Piagets læringsteori, og jeg vil senere vende tilbage til denne teori.

Den følelsesmæssige læringsdimension

Freud betegnes af Illeris som en af de banebrydende kræfter indenfor denne forståelse af læring, som blandt sine kerneområder blandt andet har motivationen for, og driften til at lære. Disse områder vil igen være vanskelige at sige meget om i forbindelse med vores genstandsfelt, men jeg vil vende tilbage til det motivationsmæssige aspekt senere.

Den samfundsmæssige læringsdimension

Det marxistiske syn på læring udspringer af den marxistiske forståelse af de psykologiske processer, som ikke værende lokaliseret i hvert enkelt individ, men i stedet lokaliseret i den sociale interaktion mellem individer og samfund (Illeris 2001, s. 19 og 235). Dette syn forekommer igen som værende noget vanskeligt at bringe i anvendelse overfor vores genstandsfelt, men en mindre ekstrem position vil passe bedre til vores anvendelse.

Dette leder mig frem til Etienne Wengers teori om læring som forankret i social praksis. Han tager afsæt i fire præmisser:

> *Vi er sociale skabninger.*
>
> *Kyndighed er et spørgsmål om kompetence i forbindelse med forehavender der værdsættes.*
>
> *Erkendelse er et resultat at deltage i sådanne aktiviteter*
>
> *Betydning – vores evne til at opleve verden og vores engagement med den som meningsfuld – er i sidste ende det som læring skal afføde.*
>
> (Wenger 1998, s. 4, min oversættelse)

Dette sætter ham i opposition til virksomhedsteoriens forståelse af det sociale fællesskab som et resultat af menneskehedens kamp for overlevelse (Hermansen 1996, s. 56-58), hvor Wenger nok mere ser vores overlevelse som et resultat af vores sociale fællesskab. Wenger ser heller ikke en modsætning mellem individets frihed og det kollektive (Wenger 1998, s. 147). Vi stræber efter fællesskab, det bliver ikke påtvunget os, og det er på denne måde vi engagerer os og danner vores sociale praksis. Jeg vil her gennemgå Wengers beskrivelse af praksisfællesskaber, for efterfølgende at operationalisere dette syn som et værktøj til analyse af teorien bag Extreme Programming, og udøveres erfaringer med XP.

Praksisfællesskaber

Forudsætninger for et praksisfællesskab

Wenger opererer med tre elementer som skal være til stede, for at man kan tale om at et praksisfællesskab findes:

1. Gensidigt engagement (mutual engagement). Hermed menes, at deltagerne er forbundet af et socialt fællesskab, et praksisfællesskab opstår ikke kun på grundlag af f.eks. en ensartet faglige profil.
2. Fælles forehavende (joint enterprises). Deltagerne i et praksisfællesskab tager ansvar for, og bidrager til at forfølge et mål de sammen definerer gennem deres gensidige handlinger og engagement.
3. Delt repertoire (shared repertoire). Her menes ikke at deltagerne i et praksisfællesskab har de samme færdigheder, men at de har en fælles diskurs omkring deres praksis, ligesom en sanger og

> en musiker kan have et fælles repertoire, uden at kunne bytte roller. Wenger bruger navnet repertoire, for at understrege dens "indøvede" karakter, og dens potentielle ugennemsigtighed for udenforstående.
>
> (Wenger 1998, s. 73 ff)

Disse tre elementer viser os, at et praksisfællesskab er knyttet tæt sammen, og ikke bare er et andet ord for team eller gruppe.
Wenger understreger også, at alle disse elementer, som konstituerer praksisfællesskabet ikke er statisk defineret en gang for alle, men til stadighed dynamisk opbygges gennem forhandling mellem deltagerne.
Netop forhandling er et nøglebegreb for forståelse af Wengers definition af praksisfællesskaber. Med forhandling menes der naturligvis ikke en formel forhandlingssituation, med kontrakter og lignende. Der menes at deltagerne påvirker hinanden, og implicit bliver enige om opbygningen af deres fælles virkelighed i praksisfællesskabet, selv om eksplicit kommunikation omkring denne opbygning ikke finder sted:

> *"The negotiation of meaning may involve language, but it is not limited to it. It includes our social relations as factors in the negotiation, but it does not necessarily involve a conversation or even direct interaction with other human beings."*
>
> (Wenger 1998, s. 53)

Forhandling af betydning

Selv om der tydeligvis findes en fælles verden, som vi alle oplever gennem vores sanser*, og som vi finder betydning i, så er denne betydning ikke noget som blot findes i verden, og som er ens for alle. Vi konstruerer selv vores egen betydning af alt hvad vi oplever, men

* Dette skal ikke forstås som den positivistiske tanke om, at intet eksisterer der ikke kan sanses, men blot som en anti-solipsisme.

denne betydning skabes ikke bare ud af den blå luft, den afhænger af hvordan verden og vores historie er:

> *"Our engagement in practice may have patterns, but it is the production of such patterns anew that gives rise to an experience of meaning (...) we produce meanings that extend, redirect, dismiss, reinterpret, modify or confirm – in a word, negotiate anew – the histories of meanings of which they are part. In this sense, living is a constant process of* negotiation of meaning*."*
>
> (Wenger 1998, s. 52-53)

Wenger bruger således begrebet forhandling af betydning meget generelt omkring vores oplevelse af verden, og vores skabelse af betydning. Denne skabelse af betydning er en stadig proces af et samspil mellem deltagelse og reifikation.

Deltagelse og reifikation

Ved deltagelse forstår Wenger en proces, som kun sociale aktører kan udføre. En computer eller en akvariefisk kan således ikke være en deltager i Wengers forstand, selv om akvariefisken er et levende individ. Deltagelse er også mere end kun handlinger i en praksis. Deltagere i et praksisfællesskab holder ikke op med at være deltagere når arbejdstiden ophører, det er en del af deres identitet (Wenger 1998, s. 56).

Reifikation er ifølge Wenger et nyttigt begreb til at beskrive hvordan vores engagement i verden skaber betydning for os. Reifikation betyder at omdanne til et objekt, og kan eksempelvis være at vi projicerer vores egen opfattede betydning ud i verden, og opfatter dem som havende deres egen eksistens. Et sæt af regler er et eksempel på en reifikation af et område af forhandlet betydning. Reifikation former således vores opfattelse af verden.

Reifikation og deltagelse er komplementære, uden deltagelse har reifikation ingen mening, og reifikation er kritisk for den

forhandlingsproces som deltagelse er (Wenger 1998, s. 61-62). Uden reifikation kan resultatet af forhandlingsprocessen ikke fastholdes, som når vi tager noter til et møde, og uden deltagelse kan reifikationen ikke fortolkes og forstås, som når vi holder et møde for at diskutere hvordan en ny regel skal forstås. Vekselvirkningen mellem deltagelse og reifikation tillader os således at udvikle og modificere rammerne for vores praksis, og dette er en vigtig forståelse for analysen af et praksisfællesskab. Praksisfællesskabet er ikke fastlagt udefra, men forandrer sig under deltagernes produktion og fortolkning af deres egen praksis. Disse analytiske kategorier kan således bruges til at forstå de processer der finder sted når et praksisfællesskab definerer og omfortolker sig selv.

Læring som historie

Deltagelse, og forhandling af betydning er begge processer der udspilles over tid, og praksis må derfor også forstås i et temporalt perspektiv. Ifølge Wenger, kan man derfor tænke på et praksisfællesskab som en fælles læringshistorie (Wenger 1998, s. 86). For at forstå et praksisfællesskab må man derfor forstå den interne dynamik, som udgør disse fælles læringshistorier.

En forudsætning for historie er hukommelse. Både deltagelse og reifikation kan tjene som former for hukommelse i et praksisfællesskab. Når en deltager forlader et praksisfællesskab forsvinder vedkommende fra deltagelsen, og bidrager ikke længere med sin fortolkning af praksisfællesskabet. De reifikationer som vedkommende har været med til at skabe vedbliver dog med at eksistere, da de har deres egen uafhængige eksistens, selvom de måske ikke giver mening uden deltagerens fortolkning. Deltagelse og reifikation er således ikke bundet til hinanden, selv om de måske er afhængige af hinanden for at give mening.

> *"We are connected to our histories through the forms of artifacts that are produced (…) and modified through the ages, and also through our experience of participation as our identities are formed (…) and transformed through mutual engagement in practice from generation to generation."*

(Wenger 1998, s. 89)

Ligesom både deltagelse og reifikation virker som en hukommelse, så giver de også begge mulighed for at glemme gennem generationsskift. Deltagere forlader praksisfællesskabet, nye kommer til, og hvis praksisfællesskabet eller består længe nok, så vil der en dag ikke længere være flere af de oprindelige deltagere tilbage, selvom praksisfællesskabet består. Reifikationer kan også have generationsskifter, for eksempel ved indførelsen af et computersystem, i stedet for en papirbaseret praksis. Begge former for generationsskifter kan transformere praksis, både gennem det der huskes og det der glemmes.

Læring i praksis

Læring forgår i alle former for praksis, men mange deltagere i praksisfællesskaber ser ikke læring som en del af deres dagligdag (Wenger 1998, s. 95). Ofte ses læring som noget de nye, potentielle, deltagere i fællesskabet gennemgår. Denne opfattelse gør sig gældende, blandt andet fordi læring ikke reificeres og ekspliciteres som mål*. Den læring der finder sted i et praksisfællesskab er ikke en indlæring af et statisk pensum, men derimod en kontinuert opbygning af betydning omkring den praksis der engageres i. Læreprocessen i fællesskabet konstitueres af følgende underprocesser:

- ✦ Udvikling af former for gensidig engagement
- ✦ Forståelse og tilpasning af det fælles forehavende

* Og i høj grad nok også fordi mange sætter lighedstegn mellem læring og undervisning.

- ✦ Udvikling af det delte repertoire

(Wenger 1998, s. 95)

Selvom læringen ses som en kontinuert proces, så understreger Wenger at læringsbegrebet ikke må trivialiseres, ved at betragte alt vi foretager os som læring. Den betydende læring er den, der påvirker de tre elementer som konstituerer praksisfællesskaber*. Praksisfællesskaber kan således ses som en emergent struktur, der opstår ud af den forhandling af betydning som deltagerne engagerer hinanden i. Dette resulterer en struktur som både er fleksibel og modstandstandsdygtig, og dermed tilpasningsdygtig. Fleksibiliteten er til stede fordi de tre konstituerende elementer gensidigt kan påvirke hinanden, så følgerne af en introduktion af noget nyt ikke er begrænset til det element hvor introduktionen foregår. Modstandsdygtigheden er et resultat af den investering som deltagerne har i fællesskabets tre elementer, som er blevet en del af deres egen identitet, og derfor ikke forandres uden grund.

Læring som bro mellem generationer af deltagere

Da medlemskabet i et praksisfællesskab, som nævnt tidligere, ikke er statisk, er en af de essentielle konstituerende processer, den der tillader nye medlemmer at blive en del af fællesskabet. Uden den ville praksisfællesskaber dø med deres medlemmer. Wenger hævder at praksis kan deles på tværs af generationer af deltagerne, netop fordi praksis er en social læringsproces (Wenger 1998, s. 99). Dermed bliver en afgørende faktor for bestandigheden af et praksisfællesskab at de eksisterende medlemmer giver de potentielle nye medlemmer mulighed for at deltage i praksis, og dermed læringsprocessen. Hvis de nye potentielle medlemmer isoleres, så knytter de ikke de sociale bånd som er nødvendige, dels for at de eksisterende medlemmer opfatter

* Gensidigt engagement, fælles forehavende og delt repertoire.

dem som potentielle ligeværdige, dels for at de selv opnår en følelse af at have del i fællesskabet*.

En interessant tese som Wenger fremsætter er, at praksisbaserede uddannelsesforløb som f.eks. mesterlære, er bedre til lære eleverne færdigheder, netop fordi læringen foregår i praksis, og der dermed er en bedre pasform mellem læringsprocessen og den efterfølgende udøvelsesproces. Dette kan synes som en triviel pointe, men hvis man ser på hvor mange uddannelsesforløb der har en stor grad af afkobling mellem uddannelse og praksis, så forekommer det ikke længere så trivielt.

Identitet og tilhørsforhold

Identitet

Wenger karakteriserer identitet som bestående af blandt følgende aspekter:

- ✦ Identitet som forhandlet oplevelse
- ✦ Identitet som medlemskab af et fællesskab
- ✦ Identitet som læringsbane

(Wenger 1998, s. 149)

Disse aspekter kan holdes op mod aspekter af praksis, som følger:

* Dette emne har Wenger udforsket dybere sammen med Jean Lave i bogen *"Situated learning – Legitimate peripheral participation"* (Lave og Wenger 1991), om end man kan indvende mod deres behandling af emnet, at de i udpræget grad ignorerer teoretisk og abstrakt viden, for kun at beskæftige sig med den praktiske og erfaringsbaserede viden og læring (Aboulafia og Nielsen 1997, s. 55-57)

Identitet som:	Praksis som:
Forhandling af selv-oplevelse gennem deltagelse og reifikation	Forhandling af betydning gennem deltagelse og reifikation
Medlemskab	Fællesskab
Læringsbane	Fælles læringshistorie

Identitet er ikke kun det samme som et reificeret selvbillede, identitet dannes også gennem vores deltagelse i fællesskaber.

> *"Who we are lies in the way we live day to day, not just in what we say or think about ourselves (…) Nor does identity consist solely of what other think or say about us. Identity in practice is defined socially"*

(Wenger 1998, s. 151)

Identitet gennem medlemskab af fællesskaber består af disse dimensioner:

- ✦ Gensidigt engagement
- ✦ Ansvarlighed overfor et forehavende
- ✦ Forhandling af repertoire

Det gensidige engagement danner vores identitet ved at lade os over være en del af det engagement med andre deltagere der skaber vores fællesskab.

Ansvarlighed overfor et forehavende medvirker til dannelsen af vores identitet gennem de måder vi er i stand til at bidrage til forehavendet. Bidragene er med til at forme vores verdenssyn, så der ofte er en overensstemmelse mellem det verdenssyn som forskellige bidragydere til et forehavende har (Wenger 1998, s. 153).

Gennem at virke i praksis, bliver vi i stand til at fortolke og bruge det repertoire som er en del af historien for denne praksis. Vi forstår historien, fordi vi selv er en del af den, og andre historier som vi ikke er en del af, former også vores identitet, idet de *ikke* er en del af vores identitet, men er en del af andres identiteter der omgiver os.

"In sum, membership in a community of practice translates into an identity as a form of competence."

(Wenger 1998, s. 153)

Identitet kan betragtes som en bane, blandt andet på grund af dens foranderlighed over tid. Da identitet dannes i sociale kontekster, er den mere kompleks end en simpel tidslinie, og bane-begrebet en forståelse for at identitet omfatter både fortid, nutid og fremtid. Det giver os også en måde at tale om den indvirkning som fællesskaber har på en individuel identitet, gennem varierende grader af deltagelse. Et karriereforløb kan ses som en del af den bane en identitet beskriver.

Wenger beskriver blandt andet følgende baner:

- Perifere baner – baner der ikke leder til fuld deltagelse, men alligevel bliver betydelige nok til at bidrage til en identitet.
- Indadgående baner – potentielle nye deltagere i et fællesskab, der nu befinder sig i en perifer position, men er på vej mod fuld deltagelse.
- Indre baner – fuld deltagelse i et fællesskab betyder ikke enden for påvirkning af identitet, da den konstant genforhandles.
- Udgående baner – at forlade et fællesskab er også identitetsformende, da identiteten også her genfortolkes og ses på nye måder.

(Wenger 1998, s. 154)

Tilhørsforhold

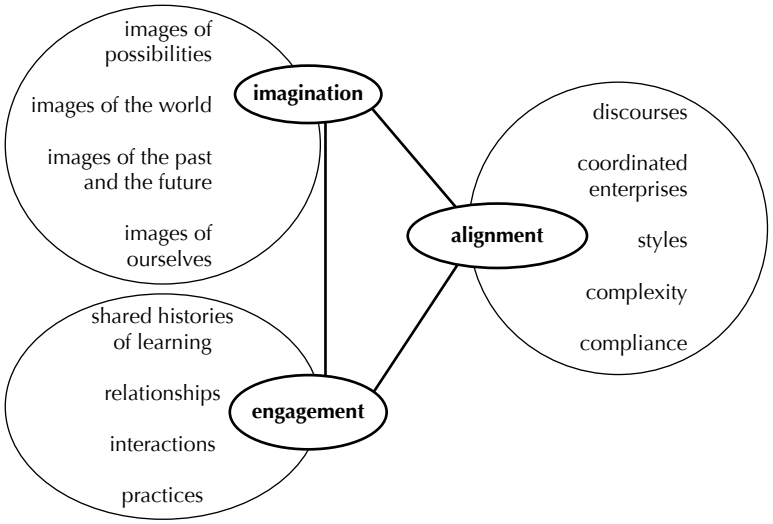
Vi har indtil nu talt om deltagelse (engagement) i et praksisfællesskab, som identitetsopbyggende. Tilknytning kan også have andre former, vi

kan med altså have andre tilhørsforhold til fællesskaber Wenger nævner to andre processer, som kan knytte os til fællesskaber:

> *"Imagination – creating images of the world and seeing connections through time and space by extrapolating from our own experience.*
>
> *Alignment – coordinating out energy and activities in order to fit within broader structures and contribute to broader enterprises."*

(Wenger 1998, s. 173-174)

Disse forskellige måder at skabe tilhørsforhold kan afbilledes således:



Figur 4, efter (Wenger 1998, s. 174)

I modsætningen til engagement, så involverer disse andre tilhørsforhold ikke en forhandling af betydning.

Fantasi (imagination), tillader os at række ud over tid og rum, og blandt andet forstå sammenhænge som ikke direkte kan afledes af vores eget liv, som for eksempel at se os selv som gamle mennesker.

Indsporing (alignment) kan være det at føle sig som en del af noget større, f.eks. at være stolt af ens arbejdsplads, være medlem af en politisk bevægelse og påtage sig eller blive pålagt at følge de normer som følger af dette,

Engagement, fantasi og indsporing er adskilte tilknytningsmåder, men

de udelukker ikke hinanden. Et fællesskab kan være konstitueret af forskellige andele af hver af disse, og dette kan resultere i fællesskaber af vidt forskellig karakter. Dette skal dog ikke forstås således, at en af disse tilknytningsmåder har bedre potentiale for læring end de andre, de har hver deres fordele og ulemper, og ved at kombinere dem kan et praksisfællesskab blive et læringsfællesskab (Wenger 1998, s. 183 og 187)

Identifikation og mulighed for forhandling

Identitet dannes i brydningsfeltet mellem deltagerens investering i tilhørsforhold, og evnen til at forhandle de betydninger der er vigtige i disse kontekster. Dette bringer naturligvis magten ind som en faktor, men Wenger understreger, at magt ikke kun skal forstås som konflikt mellem aktører, eller den enes dominans over den anden (Wenger 1998, s. 189).

Identifikation

Identifikation kan etableres gennem de tre former for tilknytning, som allerede er gennemgået. Vi har allerede set på hvordan engagement former vores identitet, men hvordan identifikation gennem fantasi, og identifikation gennem indsporing er værd at kigge nærmere på.

Identifikation gennem fantasi kan forbinde os på mange måder, og udvider dermed vores identitet voldsomt. Det kan for eksempel være gennem at identificere os med en reklame, eller identificere os som tilhængere af en bestemt tøjstil.

Identifikation gennem indsporing kan for eksempel opstå ved at følge en inspirerende leder.

Mulighed for forhandling

Mulighederne for at forhandle betydning er uafhængige af vores identifikation med en sammenhæng. Man kan betragte muligheder for at forhandle betydning som en betydningsøkonomi. I en sådan betydningsøkonomi afgøres mulighederne for forhandling af betydningsejerskab (Wenger 1998, s. 197). Betydningsejerskab skal ikke forstås som at en betydning kun kan have en ejer, betydningsejerskab kan deles, og dette kan udvide deltagelsen i produktionen af betydning.
Betydningsejerskab kan opnås gennem de tre former for tilknytning der tidligere er gennemgået. En typisk måde at opnå betydningsejerskab på, er gennem engagement, og dette ses ofte når nye deltagere knytter sig til et fællesskab. Hvis engagementet ikke er gensidigt, så er dette dog også en måde at forhindre forhandling om betydningsejerskabet, forstået på den måde, at deltagere hvis bidrag sjældent accepteres helt kan glide ud og blive ikke-deltagere.

Læringsfællesskaber

Et praksisfællesskab kan både facilitere muligheden for erhvervelse af kyndighed, og muligheden for at skabe kyndighed. Facilitering af erhvervelse af kyndighed sker, når nye deltagere i praksisfællesskabet får adgang til at erhverve sig kompetence, og mulighed for at gøre deltagelse til en del af deres identitet. Muligheden for at skabe kompetence opstår gennem det gensidige engagement omkring et delt forehavende, når der er et stærkt bånd af fælles kompetence, og respekt for erfaringens egenart, hvilket giver mulighed for at udforske nye indsigter uden at virke aparte på en negativ måde (Wenger 1998, s. 214)
Når dette kombineres med mulighed for tilknytning til fællesskabet gennem vekslende former, så opstår muligheden for andre typer af læring. Når det er eksempelvis er muligt at knytte an til fællesskabet

både gennem engagement og fantasi, så skabes der muligt for refleksiv praksis (Wenger 1998 s. 218 og Schön 1983). Disse kombinationer giver læringsfællesskabet mulighed for at ændre og forhandle sin egen struktur og betydningsøkonomi.

Problematisering af teorien om læring i praksisfællesskaber

Nu hvor jeg har gennemgået de vigtigste begreber i Wengers teori om læring i praksisfællesskaber, er det et passende tidspunkt at komme ind på områder som ikke har fokus i denne læringsforståelse, og derefter kort argumentere for hvorfor jeg ikke mener dette er et problem i vores tilfælde.

Wenger fokuserer næsten udelukkende på den uproblematiske og ikke-grænseoverskridende læring, selv om han påpeger, at denne kan være et af de tidspunkter hvor der kommer mest fokus på læring (Wenger 1998, s. 8). Han hævder dog, at det ikke nødvendigvis betyder at disse tidspunkter er dem hvor vi lærer mest eller dybest. Man kan også med god ret hævde, at set ud fra hvor den største del af læring sker, så er det mest interessante de tidspunkter hvor læringen ikke har fokus, som det er tilfældet for den "usynlige" læring der finder sted i praksis.

Ligeledes kan man fremhæve, at Wengers teori ikke fører til en forståelse af den interne læreproces i den lærende, og at dette kan være en svaghed ved at anvende Wengers teori som sit eneste redskab til forståelse af læring. Dette ville naturligvis være korrekt, og et stort problem, hvis man kun forsøgte at forstå læring, ved at kigge på den enkelte lærende, men da fokus i dette speciale netop ligger på læring i fællesskab, og ikke læring for den enkelte systemudvikler, så mener jeg heller ikke at dette udgør noget problem for min anvendelse af Wengers teori som eneste læringsforståelse. Jeg vil dog i analysen

inddrage enkeltelementer fra andre læringsforståelser, til at belyse enkelte delprocesser i systemudviklingsforløbet.

Opsummering

Nu hvor jeg har gennemgået de vigtigste begreber i Wengers teori om læring i praksisfællesskaber, og har overvejet nogle mulige problemer ved at anvende den som læringsforståelse, vil jeg ud fra disse begreber beskrive min tilgang til analyse af teorien bag, og erfaringer med praksisfællesskaber og læring i et Extreme Programming udviklingsforløb.

Overvejelser omkring analyse

MIN analyse af XP vil operere på to forskellige felter. Dels vil jeg analysere den tidligere gennemgåede teoretiske baggrund for XP, dels vil jeg analysere den indsamlede empiri, for ud fra den faktiske tilstedeværelse eller fravær af den praksis XP foreskriver, at se i hvilken grad min læsning af teorien støttes af empiri.

For at det kan give mening at foretage en analyse, ud fra et praksisfællesskabsmæssigt synspunkt, så bliver vi nødt til først at vise, at det vi analyserer rent faktisk bør fostre et praksisfællesskab. Dette vil jeg vise, ved at påvise, at de tre forudsætninger for eksistens af et praksisfællesskab, bør opstå ud af Extreme Programming som praksis.

Dernæst vil jeg ligeledes søge at påvise, at reifikation og deltagelse er centrale elementer i XP's praksis.

Herefter vil jeg undersøge om man i XP's praksis kan se en fælles læringshistorie og en læring i praksis

Jeg vil efterfølgende undersøge, hvordan der i XP lægges op til dannelse af identitet og tilhørsforhold til fællesskabet.

Sluttelig vil jeg undersøge identifikation og mulighed for forhandling og om XP dermed lægger op til dannelsen af et læringsfællesskab.

Når de ovenstående undersøgelser er gennemført, og jeg har

fundet ud af hvorvidt XP som metodologi faciliterer dannelsen af praksis- og læringsfællesskaber, og om praksisfællesskaber kan påvises i den indsamlede empiri, vil jeg afsluttende ud fra Wengers teorie om praksisfællesskaber komme med nogen bud på hvordan implementeringen af XP kan tilrettelægges, så der er de bedste muligheder for dannelsen af praksis- og læringsfællesskaber.

Overvejelser før indsamling af empiri

FOR at opnå et bredt datagrundlag, har jeg valgt at indsamle data primært via en kvantitativ metode, konkret et web-baseret spørgeskema. Jeg har støttet mig til bogen *"Valg af organisationssociologiske metoder"* (Andersen, 1990) i min udformning af spørgsmål og spørgeskema. Nogle af spørgsmålene er inspireret af (Williams 2000). Spørgeskemaet findes i bilag A.

Det web-baserede spørgeskema er valgt, dels for respondenterne ikke skal finde det besværligt at udfylde, dels for at lette den efterfølgende databehandling. Da respondenterne alle er professionelle systemudviklere, virker det som en rimelig antagelse, at formode at de er fortrolige med brug at Internettet, og at dette ikke vil være en hindrende faktor. Antallet af spørgsmål er ligeledes sat forholdsvis lavt, for ikke at afskrække potentielle respondenter.

Jeg vil søge kontakt til respondenterne gennem mit faglige netværk, hvilket formodentlig vil give et udvalg af respondenter fra flere forskellige udviklingsmiljøer og virksomheder. Det kan naturligvis gøre at respondenterne har en forskellig oplevelse af genstandsfeltet ud fra hvordan det er organiseret i lige netop deres virksomhed, og derfor svarer forskelligt,

Det fremgår klart af spørgeskemaet hvad de indsamlede data vil blive brugt til, og hvordan de vil blive behandlet. Ligeledes fremgår det, at data kun vil blive lagret anonymt. Som motiverende faktor for at

udfylde spørgeskemaet har jeg valgt at udlodde en lille belønning blandt respondenterne, men de indsamlede kontaktdata bliver lagret separat, og kan ikke knyttes til de afgivne svar. Dermed har jeg naturligvis afskåret mig fra en efterfølgende opfølgende kontakt, men det er min forhåbning, at dette vil være med til at sikre respondenternes oprigtighed.

Ikke alene lukkede spørgsmål vil blive brugt i spørgeskemaet, jeg vil også inkludere et antal åbne spørgsmål og muligheder for at respondenterne kan kommentere deres svar, for på denne måde også at opnå et kvalitativt indblik i respondenternes syn på Extreme Programming som systemudviklingsmetode, og for at den kvalitative respons kan opveje eventuelle mangler ved de kvantitative data og omvendt (Andersen 1990, s. 31 ff).

Da spørgsmålene blandt andet dækker en række komplicerede områder, er mange af spørgsmålene udarbejdet med andet end ja/nej svar, men er udformet som en tilkendegivelse, som respondenter kan erklære sig enig med i større eller mindre grad. Jeg håber således at få indsamlet et mere nuanceret datamateriale end ja/nej spørgsmål ville have givet mulighed for. Denne udformning giver dog også en risiko for at svarene bliver meget spredte, og derfor ikke bidrager med megen viden.

Opfølgning på indsamlet empiri

Ved slutningen af den periode hvor jeg samlede empiri via det web-baserede spørgeskema, stod det klart at antallet af respondenter ikke var nået op på det niveau som jeg havde regnet. Der var i løbet af en periode på 3 måneder kun kommet i alt 6 besvarelser. Dette rejser

naturligvis spørgsmålet om en kvantitativ analyse, baseret på denne meget lille mængde af data, kan siges at være valid. Jeg har valgt at gribe dette problem an på to måder, idet jeg dels vil supplere min egen empiri med empiri fra artikler i faglitteraturen omkring brugen af Extreme Programming, for at opnå en større bredde, dels primært vil fokusere på det kvalitative element i de indsamlede data. Denne fremgangsmåde er naturligvis ikke den ideelle, men jeg håber alligevel at kunne drage nytte af de indsamlede data, til at understøtte min analyse af teorien bag Extreme Programming, og til at pege på områder i implementeringen af XP, hvor lærings- og praksisfællesskaber kan understøttes.

Analyse

"Today's scientists have substituted mathematics for experiments, and they wander off through equation after equation, and eventually build a structure which has no relation to reality."

Nikola Tesla

SPØRGESKEMAUNDERSØGELSEN har resulteret i data fra programmører med forskellige baggrunde. Der er dog et antal fællesnævnere blandt respondenter som jeg her vil komme kort ind på. Det komplette opsummerede datamateriale findes i slutningen af dette speciale som bilag B.

De demografiske spørgsmål viser en klar tendens til at respondenterne er erfarne udviklere. 67% har arbejdet professionelt med systemudvikling i mere end 5 år. Den resterende gruppe af respondenter har mellem 1 og 5 års erfaring, og interessant nok har ingen under et års erfaring.
Ligeledes ser det ud til, at respondenterne har en ganske stærk tilknytning til deres arbejdsplads, idet hovedparten af dem (67%) har været ansat hos deres nuværende arbejdsgiver i mere end 5 år.
Når vi kommer til spørgsmålet om deres erfaring med par programmering så spredes svarene lidt mere, idet kun 33% har mere end 2 års erfaring med par programmering.

Eksistens af forudsætninger for praksisfællesskab

De tre elementer som forudsættes, for at et praksisfællesskab kan eksistere er:

- Gensidigt engagement
- Fælles forehavende
- Delt repertoire

Gensidigt engagement kan især ses forankret to steder i XP's praksis. Det første, og mest oplagte, er naturligvis par programmering. Hele par programmeringsprocessen er at engagere sig i en aktivitet med partneren, og aktivt tage stilling til hvilken vej det fælles projekt skal bevæge sig. Også "The Planning Game" med tilbagemelding og de forpligtelser der opstår mellem kunde og udviklere skaber et engagement mellem disse parter.

Gensidigt engagement kommer blandt andet til udtryk i empirien som følger:

> *"Det er som ved alt samarbejde, der skal en vilje til samarbejde for at det fungerer – er den vilje ikke til stede dur det selvfølgelig ikke med par-programmering. I det team jeg er i nu, er viljen til stede og det fungerer derfor godt."*

(Bilag B, s. 5)

Respondenten giver her udtryk for, at hvis ikke begge parter er indstillet på at arbejde sammen, at engagere hinanden med andre ord, så vil par programmeringen ikke fungere ordentligt. Som det fremgår af citatet, så gælder dette naturligvis ikke kun for par programmering, men for enhver form for engagement.

> *"Pair programmers put a positive form of "pair-pressure" on each other. The programmers admit to working harder and smarter on programs because they do not want to let their partner down. Also, when they meet with their partner they both work very intensively because they are highly motivated to complete the task at hand during the session. "Two people working together in a pair treat their shared time as more valuable. They tend to cut phone calls short; they don't check e-mail messages or favorite Web pages; they don't waste each others time."*

(Williams 2000, p. 39)

Her ser vi, at deltagerne sætter pris på deres arbejde sammen, og gør en indsats for at dette arbejde skal forløbe så godt som muligt.

Udviklingsprojektet i et XP forløb danner et naturligt fælles forehavende. Udviklerne skaber netop hen imod et mål (det færdige system) som de definerer og raffinerer i samspil med kunden. Udviklerne har også et andet fælles mål, nemlig stabiliteten af systemet undervejs, som de opnår ved hjælp af tests og den kontinuere integration af ny kode i det eksisterende system.

Det delte repertoire optræder gennem det fælles ejerskab af kildekoden, som forudsætter at udviklerne overholder de standarder som gruppen er blevet enige om. Det delte repertoire udtrykkes også gennem den vægtning er der af refaktorering, hvor udviklerne netop tager ejerskab af hele systemet, for at samlet set at kunne forenkle det.
Delt repertoire viser sig blandt andet i empirien ved følgende:

> *Par-programmering giver mere ensartet kode og fjerner de "uno-der" man ellers kan få. [den] Gør det også lettere at læse den fælles kode da der nu er åbenhed om kodestil.*
>
> (Bilag B, s. 3)

> *Et andet element er at gøre "min" kode til "vores" kode (...) Det kan være en svær omstilling for mange udviklere*
>
> (Bilag B, s. 6)

Respondenterne giver her udtryk for hvordan de tager det fælles ejerskab af kildekoden til sig, og at det kun kan udvikle sig til et fælles repertoire gennem en indsats fra udviklerne.

Jeg mener hermed at have påvist, at forudsætningerne for opståen af et praksisfællesskab er tilstede, i teorien bag Extreme Programming samt

i den indsamlede empiri. Jeg har dog ikke kunnet konstatere et udtryk for erkendelse af, at et fælles forehavende er til stede i de empiriske data, hvilket kan skyldes at det ikke var et tema som mine spørgsmål bragte på bane overfor respondenterne, og som de derfor ikke har omtalt. En anden mulig forklaring kan være, at det fælles forehavende er så indlysende for respondenterne, at de ikke kan se nogen grund til at omtale dette. Man kan ud fra de demografiske data argumentere for, at et fælles forehavende rent faktisk er til stede, da flertallet af respondenterne har haft den samme arbejdsgiver, og arbejdet med systemudvikling i mere end 5 år. Dette er dog ikke nogen garanti for at der er et fælles forehavende, men det kunne dog være et fingerpeg i den retning.

Deltagelse og reifikation

Vekselvirkningen mellem deltagelse og reifikation optræder i XP's praksis blandt andet i den måde historier anvendes til at planlægge udviklingsforløbet. Kundens interne forestillinger om hvad systemet skal kunne, transformeres gennem deltagelse, hvor kunden og udviklerne mødes, til en reificeret form, nemlig en nedskrevet historie. Vi kan se en manifestation af dette i empirien:

> *"XP er fedt til [udvikling?], blandt andet fordi man får en rigtig god kontakt til kunden, og det gør det nemmere at løse de rigtige problemer."*

(Bilag B, s. 6-7)

Respondenten udtaler sig her positivt omkring den gode indflydelse, som deltagelse vekslende med reifikation har, med hensyn til at udviklingsressourcerne bliver brugt rigtigt i forhold til kundens ønsker.

En anden del af XP's praksis som konstitueres af deltagelse og reifikation,

er selve programmeringen. Her omdanner programmør-parret i fællesskab deres forståelse af hvad der skal implementeres, og hvordan dette gøres bedst til en reifikation, selve kildekoden til systemet. Denne reifikation deles dernæst med de andre i gruppen (fællesskabet) gennem det fælles ejerskab af kildekoden, med deraf følgende mulighed for refaktorering. Dette kan vi spore i empirien her:

> *"Ideer bliver løbende evalueret og testet. Dårlige ideer bliver hurtigt forkastet [og] gode bliver forfremmet"*
>
> (Bilag B, s. 4)
>
> *"Extreme Programming er mere end blot par-programmering, det omfatter hele udviklingprocessen [sic]. Den absolutte styrke er den løbende evaluering (der er altid mindst 2 der har set på produktet)."*
>
> (Bilag B, s. 6)

Her ser vi tydeligt at respondenterne mener, at to udvikleres deltagelse og fælles reifikation er en af metodens styrker.

Reifikation finder også sted, når kunden udarbejder sine tests for accept af systemet, og dermed konkretiserer forventningerne til systemet på et andet plan, end det der er tilstede i den historie som udviklingen foretages ud fra.

Man kan sige, at en af de ting der adskiller XP fra traditionelle systemudviklingsmetodologier er, at der bruges så mange ressourcer på deltagelse ud over udviklernes egne rækker, og at de er spredt ud over hele udviklingsforløbet. I en traditionel metode kan man argumentere for, at deltagelse med inddragelse af kunden primært finder sted i den første del af forløbet, og at vægten herefter ligger på reifikation, i form af udvikling af software og udarbejdelse af dokumentation. Udarbejdelse af dokumentation kan naturligvis også rumme deltagelse, men rummer

et stort element af reifikation. Dette skal ikke forstås som en antydning af, at jeg betragter reifikation og traditionelle systemudviklingsmetoder som noget dårligt, men jeg mener XP udøver en interessant form for balance mellem deltagelse og reifikation, og deres gensidige moderation.

Fælles læringshistorie og læring i praksis

Muligheden for en fælles læringshistorie kan man i XP's praksis blandt andet se gennem den måde det anbefales at udvide en gruppe af udviklere på. Her lægges der vægt på vigtigheden af, at en udviklingsgruppe deles i to, stedet for at man introducerer en helt ny udviklingsgruppe til projektet. Dermed har de to udviklingsgrupper begge en fælles læringshistorie delt blandt de eksisterende medlemmer, og de nye medlemmer får mulighed for glidende at blive medlemmer af praksisfællesskabet, ved at blive en del af den fælles læringshistorie.

Læring i praksis konstitueres blandt andet gennem udvikling af det delte repertoire. Det kan i XP sammenhæng dels ses som den konstante udvidelse og refaktorering af systemets kildekode, men også gennem spredning af ny ekspertise:

> *"Another thing that was good in this project was the amount we learned from each other. Because of the way we swapped pairs and worked together, it is an excellent way of swapping knowledge and expertise. I think that this project was the perfect place to use XP, and it served us well. I can't imagine the project proceeding with as much success if we had used any other development methodology."*
>
> (Loftus og Radcliffe 2005, s. 314)

Hvis man ser forståelse og tilpasning af det fælles forehavende som eksemplificeret gennem den iterative udviklingsproces, hvor udviklerne og kunden hele tiden genforhandler målet med processen, og vejen

dertil, så ser vi at dette aspekt af læring i praksis er en af de vigtigste dele af praksis i XP, som dette citat også viser:

> *"(…)our initial estimates were very poor. (…) At our first attempt, we promised almost the whole system for one iteration (…) Our estimates did eventually improve in accuracy, but we didn't start to get them right until the third and final iteration."*
>
> (Loftus og Radcliffe 2005, s. 313)

XP ser således ud til i høj grad at fostre både en fælles læringshistorie og læring i praksis.

Identitet og tilhørsforhold

Identitet som forhandlet oplevelse, altså opbygget gennem vekselvirkningen af deltagelse og reifikation kan virke svær at decideret at lokalisere i XP's praksis, men andre aspekter af identitetsdannelsen træder straks tydeligere frem. Identitet gennem medlemskab af fællesskaber, specifikt ansvarlighed overfor et fælles forehavende. Ansvarligheden overfor det fælles forehavende træder blandt andet frem i det ansvar udviklerne tager for deres handlinger i forbindelse med udviklingen af systemet, specifikt prioriteringen af at reducere defekter. Alle respondenterne erklærede sig enige i at antallet af defekter rent faktisk bliver reduceret ved at benytte XP, og erfaringerne med solo-programmering er ikke gode blandt uerfarne udviklere:

> *"sometimes somebody worked by himself or herself to write a piece of code, and we would not find out until the next day. We therefore had no knowledge of how this code worked, and [it] led to a serious problem with bug fixing…"*
>
> (Loftus og Ratcliffe 2005, s. 313)

Forhandling af repertoire synes også at optræde i praksis:

> *"We often came up with different ideas about how the design should go and the result of arguing over which one was better often led to a truly superior hybrid design"*

(Williams 2000, s. 41)

Fra denne måde at anskue forhandling af repertoire kan man drage linier til Donald Schön's beskrivelse af Reflection in action (Schön 1983, s. 49-50). Til forskel fra den refleksion som Schön beskriver som implicit, så foregår refleksion i XP gennem deltagelse og reifikation mellem to partnere, hvilket jeg betragter som ganske usædvanligt.

Identitet som en læringsbane, og specielt det gode ved at lade nye medlemmer af fællesskabet få reel adgang til kompetence beskrives også i XP's praksis, her hvordan en erfaren programmør lærer fra en mindre erfaren gennem praksis, på trods af prækonceptioner omkring deres indbyrdes roller, hvilket kan betragtes som en måde at skifte læringsbane på.

> *"I was sitting with one of the least-experienced developers, working on some fairly straightforward task. Frankly, I was thinking to myself that with my great skill in Smalltalk, I would soon be teaching this young programmer how it's really done.*
>
> *We hadn't been programming more than a few minutes when the youngster asked me why I was doing what I was doing. Sure enough, I was off on a bad track. I went another way. Then the whippersnapper reminded me of the correct method name for whatever I was mistyping at the time.*
>
> *Pretty soon, he was suggesting what I should do next, meanwhile calling out my every formatting error and syntax mistake.*
>
> *I'm not entirely stupid. I noticed very quickly that this most junior of programmers was actually helping me! Me! Can you believe it? Me!"*

(Williams 2000, s. 48)

En vigtig faktor for om man bliver et fuldgyldigt medlem af et fællesskab, eller vedbliver med kun at have en perifer tilknytning, er ens motivation

for at blive et medlem af dette fællesskab. En af de vigtige faktorer i motivationen kan være interaktionen med erfarne praktikere (Lave og Wenger 1991, s. 110). I XP sammenhæng kan man derfor forestille at nytilkomne udviklere primært sættes i par med de mest erfarne udviklere. Man kunne frygte at dette ville forsinke udviklingsprojektet, da den erfarne ville blive nødt til at bruge mere tid på at lære den nye op, men det lader ikke til at være tilfældet:

> *"Time spent answering newbie questions is made up for in the new perspective they bring to old problems"*
>
> (Beck og Fowler 2000, s. 117)

Andre tilhørsforhold end deltagelse

Specielt tilknytning til et fællesskab gennem fantasi virker som en oplagt form for tilknytning der findes i en XP praksis. Jeg har tidligere nævnt hvordan reifikation er en central proces i et systemudviklingsforløb, da selve programmeringen kan forstås som en reifikationsproces, men også fantasi er en naturlig del af en udviklingsproces. Udvikling er ikke kun at producere funktionalitet ud fra en specifikation, men afhænger i lige så høj grad af evnen til at "tænke udenfor kassen", og indse at en bestemt løsning måske slet ikke er den rigtige, og at noget nyt og uprøvet er det rette valg. Her indtager systemudvikling måske lidt en speciel plads blandt praksisfelter, da det ligesom de kunstneriske praksisfelter kan gå direkte fra fantasi til reifikation. I systemudviklingens tilfælde, er det reificerede element endda en funktionel artefakt.

Tilknytning gennem indsporing kunne i XP's kontekst faktisk være en tilknytning til XP og "Manifesto for agile software development" (Beck *et al* 2001), hvor udviklerne bliver en del af fællesskabet ved at lade dets værdier være med til at danne deres identitet. Se (Elliot og Scacchi

2003) for et interessant blik på hvordan denne proces kan finde sted i et Free Software* udviklingsmiljø.

Identifikation og mulighed for forhandling af betydning

Jeg har allerede været inde på hvordan XP tilbyder forskellige tilknytningsformer. Disse tilknytningsformer er med til at skabe vores identifikation via vores medlemskab af forskellige fællesskaber, og den investering vi foretager i disse medlemskaber. Det andet aspekt af dannelse af identifikation, er den mulighed for forhandling af betydning den enkelte oplever i et fællesskab. Denne forhandling er udgangspunktet for en betydningsøkonomi, hvor det er muligt at opnå og dele betydningsejerskab. I XP's praksis ses muligheden for forhandling af betydning og dermed betydningsejerskab blandt andet gennem den planlægnings- og styringsproces der ligger til grundlag for hvordan iterationerne i XP afvikles.

> *"The planning game allows the developers and customer to discuss in simple, non-technical terms what the customer wants from the software… For example one of the stories was to implement a Site Map which we understood to be something quite different from what the customer wanted…"*

(Loftus og Ratcliffe 2005, s. 314)

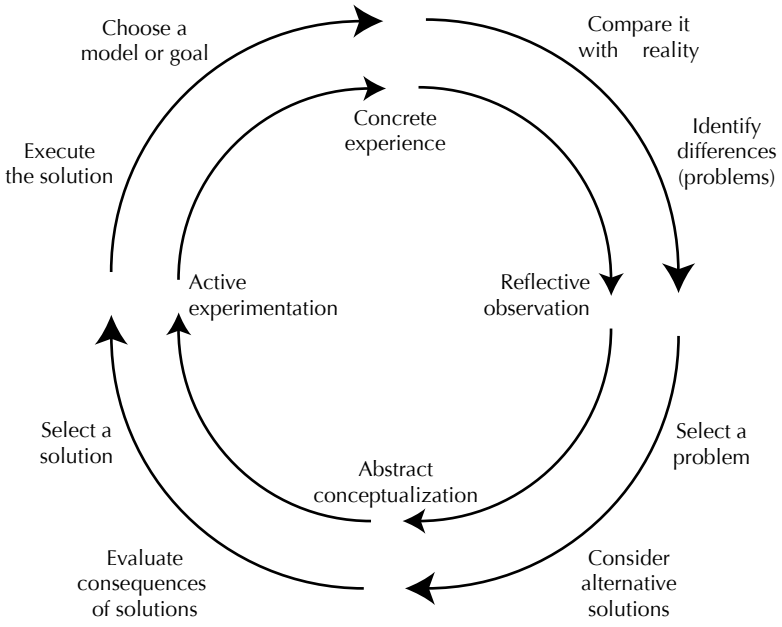
Man ser også et fingerpeg om vigtigheden af muligheden for betydningsejerskab gennem den vægt som respondenterne i spørgeskema undersøgelsen lægger på, at effektiv kommunikation er et vigtig aspekt af XP, og vigtigheden af en vilje til at engagere hinanden (Bilag B s. 3).

Forhandling af betydning og betydningsejerskab omfatter med XP's praksis altså både udviklere og kunden, og foregår kontinuert gennem hele udviklingsprocessen, hvilket igen kan ses som en betydende forskel

* Se http://www.gnu.org/philosophy/free-sw.html for en definition af hvad Free Software er, og de moralske værdier bag.

fra en traditionel systemudviklingsmetode.

En anden måde at anskue denne forhandlings- og tilpasningsproces på, er ved at se den som cyklisk i stedet for kontinuert. En cyklisk proces i praksis udspiller sig naturligvis også i den temporale dimension, men en understregning af at de samme temaer genudspilles igen og igen kan være en nyttigt forståelsesredskab. Et eksempel på denne måde at se på, beskrives i David Kolbs bog *Experential Learning: Experience as The Source of Learning and Development* (Kolb 1984), og kan visualiseres gennem denne figur:



Figur 5, lærings- og beslutningscirkel, efter (Kolb 1984, s. 33)

Som det ses, så er der et ganske stort sammenfald mellem denne måde at anskue læring og beslutning på, og den iterative, historiedrevne udviklingsstrategi, som XP gør brug af.

XP og læringsfællesskaber

Praksis i XP faciliterer både erhvervelse af kyndighed, og skabelse af kyndighed. Jeg har tidligere været inde på, hvordan nye udviklere får mulighed for at opnå kompetence, gennem samarbejdet med de erfarne

udviklere der allerede er fuldgyldige medlemmer af fællesskabet, og erkendelsen af, at dette er en kritisk faktor for succes med XP ses i spørgeskemaundersøgelsen ved at flertallet af respondenterne tilkendegiver deres enighed i at par programmering har gjort dem til dygtigere udviklere. Ingen af respondenterne er decideret uenige i at de er blevet dygtigere, men en tredjedel af dem forholder sig neutralt til spørgsmålet (Bilag B, s. 3). I spørgsmålet om hvorvidt dette skyldes samarbejdet med mere erfarne udviklere, svarer kun en tredjedel bekræftende, men ved at holde dette op mod de demografiske data, ses det at det er udviklerne med mindst erfaring der svarer bekræftende på dette. Det er derfor tænkeligt at fordelingen af svar ville have været betydelig anderledes, hvis flertallet af respondenter ikke havde været erfarne udviklere før de begyndte at anvende XP. Dette støttes af blandt andet af resultaterne i Williams og Upchurch 2001, hvor de undersøgte udviklere var studerende under uddannelse:

> *The collaborating students were queried about the reasons for their independence in an anonymous survey on the last day of class.*
>
> *74% wrote "between my partner and me, we could figure everything out."*
>
> *84% of the class agreed with the statement "I learned Active Server Pages faster and better because I was always working with a partner."*
>
> *We would attribute part of this result to enhanced problem solving in pairs, as described above, and part to enhanced pair learning. Pair-programming may "contextualize" the learning activity in a manner that allows the students to focus on the different knowledge types, and provide the feedback necessary to increase their ability to develop monitoring mechanisms for their own learning activities.*
>
> (Williams og Upchurch 2001, s. 330)

Det er også en mulighed, at uerfarne udviklere er mere opmærksomme på aspekterne i den læringsproces de gennemgår, da de i højere grad end erfarne udviklere tilegner sig færdigheder gennem udøvelsen af

praksis.

Facilitering af muligheden for skabelse af kyndighed i et læringsfællesskab ses klart i XP, blandt andet gennem den eksplicitte understøttelse af eksperimenter, og friheden til at kassere kildekode der ikke leder til det ønskede resultat. Dette ser ud til at afspejle et miljø, hvor det at fejle ikke primært ses som noget negativt, men snarere som en måde at opnå erfaring på. Gennem den opnåede erfaring har man derefter blandt andet mulighed for at genforhandle fællesskabets betydningsøkonomi, og ændre praksis i fællesskabet.

Ligeledes ser det ud til, at et læringsfællesskab baseret på XP, tilbyder muligheden for forhandling af betydning udenfor fællesskabet. Dette ses blandt andet gennem den metodiske fleksibilitet som XP lægger op til:

> *"We have been prescriptive here, telling you what to do to plan your software project. At the end of the day, however, you have to own your own plan and your own planning.*
> *One way to adapt XP planning to your own situation is to take the bits and pieces of it that make sense and mix them with what you are doing now"*
>
> (Beck og Fowler 2000, s. 129)

Dette er dog også en af de potentielle svagheder ved XP, som det også kommer til udtryk i empirien:

> *"Det 'sjove' ved at tage XP i brug er at det er svært at bruge fordi det er en meget løs [sic] defineret metode. Så hvis man ikke har været igennem et metodemæssig forløb hvor man har prøvet andre mere stringente metoder såsom vandfald o.lign. så er det meget svært at komme i gang. Vi var vant til at bruge vandfalds-metoden, så et skifte til XP var forholdsvis nemt"*
>
> (Bilag B, s. 6)

> *"Vores erfaring er, at XP bedst indføres lidt af gangen, så overgangen bliver mere glidende - f. eks. kan man starte med kode review. Sker det hele på en gang vil man opleve en stor modstand fra nogen, som ikke har set "lyset" men kun set forandringen.*
>
> *Det er vigtigt at man oplever det positive i XP og derefter bliver mere åben for at tage flere XP tiltag ind i udviklings processen - omstilling til en ny arbejdsform kan være svær, specielt for erfarne udviklere, der 'ved hvordan man gør'.*

(Bilag B, s. 6).

Det er interessant, hvordan disse to respondenter har delvist modsatrettede oplevelser af problemerne ved et skift til XP som udviklingsmetode. Den ene ser den manglende stringens i XP som et problem, men mener at det ikke er det store problem for erfarne udviklere, mens den anden oplever at omstillingen kan være svær, specielt for erfarne udviklere. Denne forskel er tilsyneladende ikke forårsaget af en forskel i erfaring mellem respondenterne, idet begge har mere end 5 års erfaring med systemudvikling. Det er dog muligt at forskellen kan bestå i de to respondenters respektive roller, hvis den første eksempelvis primært arbejder som udvikler, og den anden måske mere virker som projektleder. Dette giver spørgeskemaundersøgelsen dog ingen mulighed for at svare på.

Opsummering

Jeg har i denne analyse gennemgået teorien bag praksis i Extreme Programming gennem en optik dannet ud fra Etienne Wengers teori omkring praksisfællesskaber. Mine konklusioner omkring teorien har jeg støttet ved at inddrage dels min egen indsamlede empiri, dels empiri fra litteratur der beskæftiger sig med undersøgelse af XP i praksis.
I den efterfølgende konklusion vil jeg vurdere hvad man kan uddrage af denne analyse, og efterfølgende komme med bud på hvordan XP som metode kan styrkes gennem facilitering af læring.

Konklusion

En Steen kand ikke flyve.
I kand ikke flyve.
Ergo, er Morlille en Steen?

Ludvig Holberg, "Erasmus Montanus"

Efter at have undersøgt teorien bag Extreme Programming ud fra et læringsteoretisk synspunkt, mener jeg man kan konkludere, at XP som systemudviklingsmetodologi i høj grad giver mulighed for, og ansporer til, dannelsen af praksisfællesskaber bestående af både udviklere og repræsentanter for kunden. Jeg oplever endvidere denne inddragelse af kunden i fællesskabet som en betydende forskel, i forhold til forholdet mellem kunde og udvikler i en traditionel systemudviklingsmetodologi.

Jeg har ikke kunnet påvise en fuldstændig overensstemmelse mellem XP's praksis samt teori, og teorien omkring praksisfællesskaber, men jeg mener at have vist at der er en meget stor grad af overensstemmelse. Dermed mener jeg, at der ikke kan være nogen tvivl om, at læringen i et XP forløb med fordel kan forstås ud fra den læringsforståelse som Etienne Wenger beskriver. Dette bestyrkes jeg i, gennem den kontrast jeg oplever, i forhold til en overensstemmelse mellem traditionel systemudviklingsmetodologi og Wengers teori.

Kan vi så ud fra denne overensstemmelse bidrage til en bedre implementering af XP, gennem forbedrede muligheder for læring? Jeg mener vi kan, og jeg vil i det efterfølgende komme ind på hvordan man kan designe til praksis med henblik på læring.

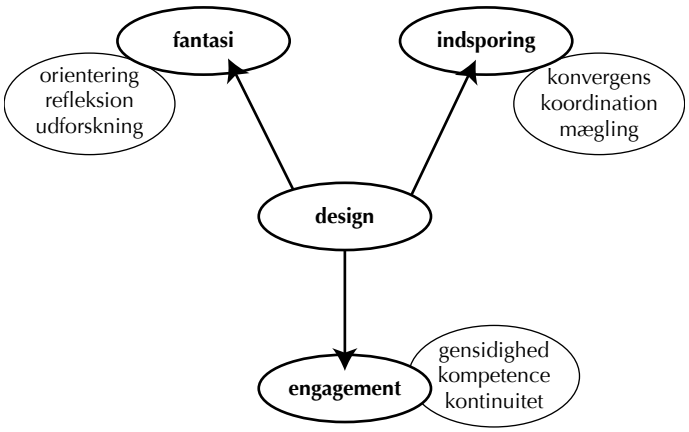
Design til læring

Læring kan ikke direkte designes, da læring opstår ud af praksis og de erfaringer der erhverves i praksis. Hvad man derimod kan, er at designe *til* læring, gennem design af sociale infrastrukturer som faciliterer læring (Wenger 1998, s. 225). På samme måde kan man ikke designe praksis, praksis vil opstå ud af de betingelser som de praktiserende bliver tilbudt eller pålagt.

Disse betingelser kan både reificerede af natur, så som værktøjer og regler, men kan også være at bringe de rette personer sammen, og dermed give mulighed for deltagelse og gensidigt engagement. (Wenger 1998, 232).

> *"There are two conversations going on. One inside the company. One with the market.*
>
> *In most cases, neither conversation is going very well. Almost invariably, the cause of failure can be traced to obsolete notions of command and control. (...)*
>
> *These two conversations want to talk to each other. They are speaking the same language. They recognize each other's voices.*
>
> *Smart companies will get out of the way and help the inevitable to happen sooner."*
>
> ("The Cluetrain Manifesto", Levine *et al* 1999)

Selv om dette citat primært er rettet mod kommunikationen med en forbruger, så mener jeg det er en fin parallel når man taler om design til læring. Man kan med andre ord sige, at når design skal fostre deltagelse og læring, så er mere design og mere reifikation ikke altid bedre. Rummet man kan designe kan ses som bestående af tre dimensioner, de samme som vi tidligere har brugt til at beskrive deltageres tilhørsforhold til et fællesskab.



Figur 1 (efter Wenger 1998, s. 237)

Ved at tilbyde faciliteter som støtter disse tre dimensioner kan man give mulighed for at tilknytning opstår, og læring finder sted.

At støtte engagement kan blandt andet ske ved at sørge for de rette fysiske og virtuelle rum. Par programmering fordrer eksempelvis arbejdspladser der er indrettet så to personer kan arbejde sammen ved samme computer (se Noble *et al* 2004, s. 5-6). Engagement kan ligeledes støttes, ved at lade deltagere i fællesskabet bruge deres dømmekraft, og blive evalueret, og dermed styrke deltagernes kompetence. Hukommelse i fællesskabet, både i form af reificerede artefakter og deltagelse kan være med til at skabe en kontinuitet, der støtter engagement (Wenger 1998, s. 238). Dette kan ligeledes være med til at facilitere potentielle deltageres vej ind i fællesskabet, ved at lade dem veksle mellem praksis sammen med eksisterende deltagere, og deres egen refleksion over praksis (Wenger 1998, s. 249-250).

Fantasi kan støttes ved at sørge for, at deltagerne har adgang til andre praksisfællesskaber, hvilket kan give dem inspiration til deres eget praksisfællesskab. At understøtte muligheden for at deltagere kan være knyttet perifert til flere praksisfællesskaber, kan give mulighed for genforhandling af forholdene mellem flere fællesskaber (Wenger 1998, s. 255). Ligeledes vil tid og rum til refleksion over egen praksis

og udforskning af alternativer kunne styrke deltagernes tilknytning til fællesskabet.

Fokus på deltagernes indsporing i forhold til fællesskabet, og deres forståelse af fællesskabets og virksomhedens værdigrundlag kan være medvirkende til at forbinde deltagernes praksis til en større sammenhæng. I en XP sammenhæng kunne det for eksempel være en codex der viste værdier deltagerne burde tage til sig, ud over dem som XP beskriver. En anden form for facilitering af indsporing kunne være at distribuere autoritet, og dermed give den enkelte deltager en større fornemmelse af kontrol over vedkommendes eget liv.

Som vi har set, så støtter Extreme Programming som metodologi allerede godt op mange af de elementer jeg har nævnt, men et forøget fokus på disse elementer, og målene med at støtte op om dem, vil efter min mening kunne hjælpe virksomheder, der har implementeret XP som deres systemudviklingsmetodologi.

Diskussion og perspektivering

PÅ samme måde som jeg indledte dette speciale med et af de digte jeg bliver ved med at vender tilbage til, så afslutter jeg specialet med dette maleri, "blå himmel", malet af den bornholmske billedkunster Birger Bendtsen i 1970. Når jeg betragter maleriet, fanges jeg altid af den sorte silhuet i forgrunden. Det har altid forekommet mig, at det er en skikkelse der bevæger sig ind i maleriet hen ad vejen. Himlen over skikkelsen er både lys og mørk, og signalerer måske at der venter både svære og gode tider forude. Den samme oplevelse har jeg haft undervejs i processen med dette speciale, og jeg vil her dels reflektere over nogen af disse oplevelser, og de spørgsmål de rejser, dels over selve konklusionen på specialet.

Det første spørgsmål der meldte sig for mig, var om de spørgsmål jeg havde stillet var værd at stille, og om de konklusioner jeg var

kommet frem til var interessante for andre end mig selv. Jeg mener at problemstillingen var, og er, interessant for personer der har ansvaret for en systemudviklingsproces, og som overvejer at indføre Extreme Programming som rammen omkring deres udviklingsproces. Er mit speciale af den grund rettet mod eksempelvis en projektleder, der står for at skulle indføre XP som metodologi i en virksomhed? Nej, det var ikke mit sigte, og specialet kredser nok for meget omkring en ret snæver teoretisk synsvinkel, til at en praktiker ville synes at vedkommende fik meget ud af at læse det. Mit speciale indeholder ikke en grydeklar opskrift på hvordan man bedst muligt implementerer XP, men jeg mener at man efter at have læst det vil have en forståelse for aspekter af XP som har med læring at gøre, og dermed har en mulighed for at forstå problemer relateret til læring i XP. Er svarene på de spørgsmål jeg stillede i min problemformulering, så dels relevante for forståelsen af XP, dels gyldige svar? Det vil jeg mene de er, dog med det forbehold at det kvantitative og kvalitative datagrundlag i undersøgelsen burde have været større, og at respondenterne burde være kommet fra en bredere vifte af virksomheder og udviklingsmiljøer. Ligeledes vil jeg fremhæve den begrænsede mængde af spørgsmål som en kilde til usikkerhed, som kunne have været undgået ved i større grad at have brugt en kvalitativ metode som for eksempel interviews eller observation. Det aktuelle datagrundlag var hvad jeg kunne opnå ud fra mit personlige netværk på det pågældende tidspunkt, med den metode jeg valgte.

I forlængelse af dette, kan det diskuteres om programmeringen af et system der kunne vise et web-baseret spørgeskema, lagre respondentens valg, og efterfølgende opsummere disse til brug i analysen, var en fornuftig brug af min tid. At bruge et eksisterende system var naturligvis en mulighed, men ved selv at udfærdige systemet, og have fuld

kontrol over hvordan data blev lagret, var jeg i stand til at garantere respondenterne den grad af fortrolighed, som jeg selv ville kræve for at medvirke i en tilsvarende undersøgelse.

Disse overvejelser omkring tidsforbrug leder mig videre til mit næste spørgsmål. Ville jeg en hypotetisk anden gang gennemføre specialeskrivningsprocessen på samme måde, og skrive speciale alene? Svaret på dette spørgsmål må blive nej. Jeg har en del gange i forløbet ønsket, at jeg havde en partner at vende ideer og problemstillinger med. En partner ville nok også have bidraget til en mindre fornemmelse af at sidde fast undervejs i processen. Dette forløb har stået i skrap kontrast til mit daglige arbejdsliv, hvor mine kollegaer i høj grad føles som et Community of Practice. Dog har det været en lærerig proces, hvor jeg mener at have opnået større selvindsigt, både på det personlige og faglige plan.

En af de ting jeg blev klar over i forløbet med at skrive dette speciale var, at min personlige erfaring med systemudvikling i større fællesskaber ikke er så stor som jeg kunne ønske. Man kan mene, at det ikke er nødvendigt for at kunne undersøge et område, og at min personlige erfaring ikke er så lille endda, men det vil være et af mine projekter for den nærmeste fremtid.

En interessant opfølgning på dette speciale kunne være at gennemføre etnografiske studier af et miljø hvor XP anvendes, for at opnå en større grad af viden om hvordan systemudviklere kommunikerer og lærer i denne proces. Man kan endda forestille sig, at der ville være færre vanskeligheder forbundet med et sådant studie, netop fordi udviklere der anvender XP er vant til tilstedeværelsen af andre i deres daglige arbejde,

i forbindelse med blandt andet par programmering, sammenlignet med udviklere der primært arbejder alene.

Et andet interessant projekt kunne være et studie af om XP eller en anden adræt systemudviklingsmetode kunne anvendes i et miljø, hvor udviklerne ikke er samlet geografisk. Et sådant miljø kunne for eksempel være et projekt der følger en Open Source model, og hvor udviklerne primært kommunikerer og interagerer over Internettet. Ville tekniske hjælpemidler der faciliterer interaktion, som for eksempel video-chat og skærmdeling, gøre det muligt at anvende XP i sådan et scenarie?

Ligeledes kunne det være interessant at gennemføre en analyse af XP ud fra et distribueret kognition synspunkt, og holde den op mod analysen af XP som praksisfællesskab.

Jeg kommer ikke til at udføre nogen disse opfølgende studier, men jeg vil inddrage de erkendelser jeg er kommet til omkring praksis, læring og fællesskab, i løbet af mit arbejde med dette speciale, både i mit arbejdsliv og i mit liv udenfor arbejdet.

Litteraturliste

Aboulafia, A. og Nielsen, J. L. (1997), *"Situated Learning" – Nogle Videnskabsteoretiske Synspunkter,* Aalborg Universitetsforlag

Abrahamsson, P.; Warsta, J.; Siponen, M. T.; og Ronkainen, J. (2003), *New directions on agile methods*. I *Proceedings of the 25th international Conference on Software Engineering*, IEEE Computer Society,

Andersen, Ib (red.) (1990), *Valg af organisationssociologiske metoder,* Samfundsliteratur

Beck, K. (2000), *Extreme Programming Explained*, Addison-Wesley

Beck, K.; Beedle, M.; van Bennekum, A.; Cockburn, A.; Cunningham, W.; Fowler, M.; Grenning, J.; Highsmith, J.; Hunt, A.; Jeffries, R.; Kern, J.; Marick, B.; Martin, R. C.; Mellor, S.; Schwaber, K.; Sutherland, J. og Thomas, D. (2001), *Manifesto for agile software development*. Tilgængelig på web-adressen http://agilemanifesto.org/ eller fra forfatteren til dette speciale.

Beck, K. og Fowler, M. (2000), *Planning Extreme Programming,* Addison-Wesley

Boehm, B. og Turner, R. (2003), *Balancing Agility and Discipline*, Addison-Wesley

Cockburn, A. (2001), *Agile Software Development*, Addison-Wesley

Dijkstra, E. (2001), *What led to "Notes on Structured Programming"*. Tilgængelig på web-adressen http://www.cs.utexas.edu/users/EWD/ewd13xx/EWD1308.PDF eller fra forfatteren til dette speciale.

Elliott, M. S. og Scacchi, W. (2003). *Free software developers as an occupational community*. I *Proceedings of the 2003 international ACM SIGGROUP Conference on Supporting Group Work. GROUP '03*, ACM Press

Hansen, S. (2005), *Virk.dk langt fra mål - optimisme blomstrer,* I ComputerWorld 12/08-2005.

Hermansen, M. (1996), *Læringens univers,* Forlaget Klim

Illeris, K. (2002), *The Three Dimensions of Learning*, Roskilde University Press/Niace

Jeffries, R.; Anderson, A. og Hendrickson, Chet (2001), *Extreme Programming Installed,* Addison-Wesley

Kolb, D. (1984), *Experiential Learning,* Prentice Hall

Lave, J. og Wenger, E. (1991), *Situated Learning,* Cambridge University Press

Levine, R.; Locke, C.; Searls, D.; Weinberger, D., *The Cluetrain Manifesto*. Tilgængelig på web-adressen http://www,cluetrain.org/#manifesto eller fra forfatteren til dette speciale.

Loftus, C. og Ratcliffe, M. (2005), *Extreme Programming Promotes Extreme Learning?,* I *Proceedings of the 10th Annual SIGCSE Conference on innovation and Technology in Computer Science Education, ITiCSE '05*, ACM Press

Newkirk, J. (2002), *Introduction to agile processes and extreme programming*. I *Proceedings of the 24th international Conference on Software Engineering*, ACM Press

Noble, J.; Marshall, S.; Marshall, S.; og Biddle, R. (2004), *Less Extreme Programming*. I *Proceedings of the Sixth Conference on Australian Computing Education - Volume 30* , Australian Computer Society

Paulk, M.C.; Curtle, B.; Chrissis, M. B.; Weber, C. V. (1993), *Capability Maturity Model for Software, Version 1.1,* Software Engineering Institute Technical Report

Pressman, R. (1997), *Software Engineering,* McGraw-Hill Book Compagny

Schneider, J. og Johnston, L. (2003), *eXtreme Programming at universities*. I *Proceedings of the 25th international Conference on Software Engineering,* IEEE Computer Society

Schön, D. (1983) *The Reflective Practitioner*, Basic Books

Stephens, M. og Rosenberg, D. (2003), *Extreme Programming Refactored: The Case Against XP*, Apress.

Vanderburg, G. (2005), *A simple model of agile software processes*. I *Proceedings of the 20th Annual ACM SIGPLAN Conference on Object Oriented Programming, Systems, Languages, and Applications*, *OOPSLA '05*, ACM Press

Wenger, E. (1998), *Communities of Practice*, Cambridge University Press

Williams, L. (2000), *The Collaborative Software Process,* Department of Computer Science, University of Utah

Williams, L. og Upchurch, R. L. (2001), *In support of student pair-programming*. I *Proceedings of the Thirty-Second SIGCSE Technical Symposium on Computer Science Education*, ACM Press

Bilag A

Undersøgelse om Extreme Programming og par-programmering

Tak fordi du vil deltage i en undersøgelse om dine erfaringer med Extreme Programming og par-programmering. Resultaterne af undersøgelsen vil blive brugt anonymiseret i et speciale på uddannelsen Humanistisk Datalogi på Aalborg universitet. Når specialet er blevet bedømt vil de ubehandlede resultater blive slettet.

Hvis du vil være med i lodtrækningen om et gavekort til et par flasker vin, bedes du udfylde feltet e-mail adresse, så jeg kan kontakte dig, hvis du bliver udtrukket. Dette er naturlig helt frivilligt, og din e-mail adresse vil ikke blive brugt til andet.

Med venlig hilsen, **Mads Peter Bach**.

E-mail adresse:	
1. Hvor længe har du arbejdet professionelt med programmering?	
2. Hvor længe har du været ansat hos din nuværende arbejdsgiver?	
3. Hvor længe har du par-programmeret?	
4. Jeg føler et med-ejerskab af al den kode vi producerer	
5. Hvornår arbejder du hellere alene?	• ☐ Når der skal designes • ☐ Når jeg skal sætte mig ind i noget svært stof • ☐ Når jeg skal sætte mig ind i nyt stof • ☐ Når der skal laves prototyper • ☐ Når min partner ikke er til stede, eller har travlt • ☐ Andet (kommenter venligst)

	Kommentar
6. Hvad er de vigtigste roller for den person som ikke sidder ved tastaturet?	• ☐ Review af kode undervejs • ☐ Review af design undervejs • ☐ At overveje næste delopgave • ☐ Andet (kommenter venligst)
	Kommentar
7. Jeg mener kommunikation er et af de vigtigste aspekter af Extreme Programming	
8. Par-programmering har gjort mig til en dygtigere udvikler	
	Kommentar
9. (Hvis du svarede bekræftende på det foregående spørgsmål) Hvorfor er du blevet dygtigere?	• ☐ I et par lærer man hurtigere end man gør alene • ☐ Arbejdet med mere erfarne udviklere • ☐ Jeg er blevet bedre til at formulere mine tanker og ideer • ☐ Andet (kommenter venligst)
	Kommentar
10. Forskelle i erfaring kan skabe problemer ved par-programmering	

11. Par-programmering hjælper med at udviske forskelle i erfaring	
12. Jeg lærer hurtigere om et emne når vi par-programmerer	
13. Min personlige tilfredshed ved par-programmering er større end ved solo-programmering	
14. Antallet af defekter i vores kode er mindre når vi par-programmerer	
15. Jeg par-programmerer ikke lige godt med alle	Kommentar
16. Hvis jeg selv kan bestemme min udviklingsmetode, vil jeg foretrække Extreme Programming	
17. Generelle kommentarer om Extreme Programming	Kommentar

Indsend

Bilag B

Undersøgelse om Extreme Programming og par-programmering

Hvor længe har du arbejdet professionelt med programmering?

- 67% Mere end 5 år
- 33% 1-5 år

Hvor længe har du været ansat hos din nuværende arbejdsgiver?

- 67% Mere end 5 år
- 33% 1-5 år

Hvor længe har du par-programmeret?

- 50% Mindre end 1 år
- 33% Mere end 2 år
- 17% 1-2 år

Jeg føler et med-ejerskab af al den kode vi producerer

- 67% Enig
- 33% Meget enig

Hvornår arbejder du hellere alene?

Når der skal designes

- 100% Nej

Når jeg skal sætte mig ind i noget svært stof

- 67% Nej
- 33% Ja

Når jeg skal sætte mig ind i nyt stof

- 83% Nej
- 17% Ja

Når der skal laves prototyper

- 100% Nej

Når min partner ikke er til stede, eller har travlt

- 67% Nej
- 33% Ja

Andet (kommenter venligst)

- 67% Ja
- 33% Nej

Kommentar:

"Skrivning af projektplaner og den slags. "

"Vi er ikke så mange, så derfor er vi nød til at have specialområder, som andre ikke dækker. Det er i de situationer man arbejder alene. Jeg dækker området udviklingsmiljø, teknisk design og arkitektur og arbejder derfor tit alene på de områder."

"Jeg er vant til at studere svært stof alene, fra universitetet."

"Jeg foretrækker at arbejde selv, med alt andet end design, kodning og test."

"Jeg er vant til gruppearbejde fra universitetet, så det passer mig fint at arbejde i par det meste af tiden."

Hvad er de vigtigste roller for den person som ikke sidder ved tastaturet?

Review af kode undervejs

- 67% Ja
- 33% Nej

Review af design undervejs

- 83% Ja
- 17% Nej

At overveje næste delopgave

- 100% Nej

Andet (kommenter venligst)

- 67% Ja
- 33% Nej

Kommentar:

"At holde overblik, men også komme med forslag og "coache" undervejs. Det er nemmere at holde fokus og overblik, når man ikke sidder ved tastaturet."

"At gribe ind overfor ting man ikke forstår, så koden bliver ved med at være forståelig."

"At bevare det kølige overblik."

"At komme med forslag til andre måder at løse den story man er i gang med lige nu."

Jeg mener kommunikation er et af de vigtigste aspekter af Extreme Programming

- 67% Enig
- 17% Uenig
- 17% Meget enig

Par-programmering har gjort mig til en dygtigere udvikler

- 50% Enig
- 33% Hverken enig eller uenig
- 17% Meget enig

Kommentar:

"Par-programmering giver mere ensartet kode og fjerner de "unoder" man ellers kan få. Gør det også lettere at læse den fælles kode da der nu er åbenhed om kodestil."

"Par-programmering har været en god måde for mig at lære udviklingskulturen på mit arbejde at kende."

"Jeg er helt klart blevet en bedre udvikler, men det skyldes måske bare at jeg får mere erfaring."

(Hvis du svarede bekræftende på det foregående spørgsmål)
Hvorfor er du blevet dygtigere?

I et par lærer man hurtigere end man gør alene

- 50% Ja
- 50% Nej

Arbejdet med mere erfarne udviklere

- 67% Nej
- 33% Ja

Jeg er blevet bedre til at formulere mine tanker og ideer

- 67% Nej
- 33% Ja

Andet (kommenter venligst)

- 67% Nej
- 33% Ja

Kommentar:

"Vi er alle erfarne udviklere (15-20 års erfaring), så den største fordel er nok, at det er sjovere at programmere sammen end det er at sidde alene med en opgave."

"Ideer bliver løbende evalueret og testet. Dårlige ideer bliver hurtigt forkastet gode bliver forfremmet"

"Det er godt at skulle argumentere for hvorfor man vil gøre som man er i gang med. Det gør det klarere for mig selv."

Forskelle i erfaring kan skabe problemer ved par-programmering

- 67% Uenig
- 33% Hverken enig eller uenig

Par-programmering hjælper med at udviske forskelle i erfaring

- 50% Enig
- 33% Hverken enig eller uenig

- 17% Uenig

Jeg lærer hurtigere om et emne når vi par-programmerer

- 50% Enig
- 17% Meget enig
- 17% Uenig
- 17% Hverken enig eller uenig

Min personlige tilfredshed ved par-programmering er større end ved solo-programmering

- 50% Enig
- 33% Meget enig
- 17% Uenig

Antallet af defekter i vores kode er mindre når vi par-programmerer

- 83% Meget enig
- 17% Enig

Jeg par-programmerer ikke lige godt med alle

- 83% Enig
- 17% Hverken enig eller uenig

Kommentar:

"Stilmæssig forskelle i hvordan man programmerer er nogen gange en forhindring for at to personer kan par-programmere sammmen."

"Det er som ved alt samarbejde, der skal en vilje til samarbejde for at det fungerer - er den vilje ikke til stede dur det selvfølgelig ikke med par-programmering. I det team jeg er i nu, er viljen til stede og det fungerer derfor godt."

"Men det går nu alligevel, selvom det ikke er alle diskusioner der er lige frugtbare."

"Jeg synes ikke jeg har oplevet de store problemer, jeg svinger meget godt med de andre i min udviklingsgruppe."

Hvis jeg selv kan bestemme min udviklingsmetode, vil jeg foretrække Extreme Programming

- 50% Enig
- 33% Meget enig
- 17% Uenig

Generelle kommentarer om Extreme Programming

Kommentar:

"Det "sjove" ved at tage XP i brug er at det er svært at bruge fordi det er en meget løs defineret metode. Så hvis man ikke har været igennem et metodemæssig forløb hvor man har prøvet andre mere stringente metoder såsom vandfald o.lign. så er det meget svært at komme i gang. Vi var vant til at bruge vandfaldsmetoden, så et skifte til XP var forholdsvis nemt. Men det betyder selvfølgelig også meget at hver enkelt udvikler er indstillet på at være med til en forandring i udviklingsmetoden. "

"XP kan kun indføres i et team/projekt, hvis der også er ledelsens opbakning til det. Vores erfaring er, at XP bedst indføres lidt af gangen, så overgangen bliver mere glidende - f. eks. kan man starte med kode review. Sker det hele på en gang vil man opleve en stor modstand fra nogen, som ikke har set "lyset" men kun set forandringen. Det er vigtigt at man oplever det positive i XP og derefter bliver mere åben for at tage flere XP tiltag ind i udviklings processen - omstilling til en ny arbejdsform kan være svær, specielt for erfarne udviklere, der "ved hvordan man gør". Et andet element er at gøre "min" kode til "vores" kode (som der ikke bliver spurgt om i dette skema). Det kan være en svær omstilling for mange udviklere - det er sjældent at man starter helt forfra på software projekter og der vil derfor være meget "min" kode når man starter på at indføre XP i et team/projekt."

"Extreme Programming er mere end blot par-programmering, det omfatter hele udviklingprocessen. Den absolutte styrke er den løbende evaluering (der er altid mindst 2 der har set på produktet). "

"Extreme Programming har været noget af en omvæltning for mig. Jeg havde kun prøvet at programmere efter den almindelige metode før, og det var en udfordring at skulle arbejde tæt sammen med en anden udvikler det meste af tiden. Det var en god oplevelse at slippe for at bruge meget tid på at dokumentere, når dokumentationen sjældent blev brugt alligevel."

"Det mest af min arbejdstid går med at vedligeholde vores eksisterende systemer, der ikke er udviklet ved brug af XP. Derfor er XP ikke den bedste metode til mine opgaver."

"XP er fedt til, blandt andet fordi man får en rigtig god kontakt til kunden, og det gør

det nemmere at løse de rigtige problemer."

Extrem Programming

Extrem Læring?

En specialeafhandling indenfor Humanistisk Datalogi

Mads Peter Bach
Institut for Kommunikation
Aalborg Universitet

2006