CENTER FOR SCIENCEUDDANNELSE
SCIENCE AND TECHNOLOGY
AARHUS UNIVERSITET

15.04.2013

It-fagets metoder, version 0.3

Et notat der beskriver it-fagenes metoder med specielt fokus på det nye forsøgsfag "informationsteknologi" i de gymnasiale uddannelser: stx, hhx, htx og hf.

Michael E. Caspersen & Palle Nowack, Center for Scienceuddannelse, Aarhus Universitet.

Introduktion

Når vi taler om it-fagets metoder er det vigtigt at gøre sig klart, hvad vi mener med "faget". I dette notat skelner vi mellem:

- **Professionsfaget**: Hvordan er man kreativt og konstruktivt skabende med it? Dvs. hvordan bruges it til at forstå, ændre og skabe produkter, services, processer og organisationer?
- **Forskningsfaget**: Hvordan bedriver man forskning indenfor it-faget? Dvs. hvordan tilvejebringer og skaber man ny viden om it?
- **Undervisningsfaget**: Hvordan underviser man i it? Dvs. hvordan skaber man effektive læreprocesser i it i f.eks. de gymnasiale uddannelser?

For alle tre syn på faget, gælder det, at valget af metoder altid kan diskuteres, og at en given valgt metode altid skal tilpasses den konkrete situation, hvori den anvendes. Indenfor professionsfaget påvirkes valget af systemudviklingsmetode f.eks. af organisationens og projektlederens erfaring og kompetence, udviklernes ditto, time-to-market, tilgængeligheden og kompetencen af de fremtidige brugere, den valgte teknologi osv. Tilsvarende overvejelser gør sig gældende indenfor forskning og undervisning.

I det følgende fokuserer vi på fagets forskningsmetoder, men til sidst i noten vender vi tilbage til professionsfaget og undervisningsfaget, og foreslår, at der er betydelige sammenfald i metoderne, hvilket kan udnyttes meget konstruktivt i undervisningen.

Kategorier af forskningsmetoder

Vi skelner mellem 5 kategorier af forskningsmetoder indenfor it-faget:

- **Formelle** metoder anvendes typisk til at bevise egenskaber ved algoritmer og systemer, f.eks. tidskompleksitet, pladskompleksitet, og korrekthed. Dette anvendes f.eks. til forskning i kryptografi og sikkerhed. Matematik og logik er centrale redskaber.
- **Eksperimentelle** metoder anvendes typisk til at evaluere nye løsninger på problemer, og er ofte opdelt i to faser:
    - En udforskende fase, hvor man finder/udvikler de meningsfyldte spørgsmål at stille eller kriterier at måle på (hvilket svarer til hypoteseformulering indenfor den naturvidenskabelige metode).
    - En evalueringsfase, hvor man forsøger at besvare ovenstående spørgsmål eller måle på de nævnte kriterier.
- **Konstruktions**-metoder består i at konstruere en digital artefakt, for at demonstrere, at det er muligt. For at kunne betragte dette som forskning, må egenskaber ved enten konstruktionsprocessen eller artefaktet kunne påvises at være nye.
- **Proces**-metoder anvendes til at forstå de processer, man anvender til at løse opgaver indenfor it. Dette gælder f.eks. indenfor systemudvikling, interaktionsdesign og kunstig intelligens.

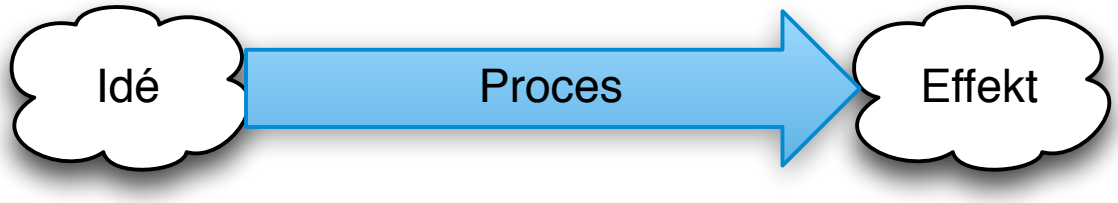
- **Modellerings**-metoder anvendes til at udforme abstrakte modeller af systemer. Modellen reducerer kompleksiteten af det virkelige/forestillede system og tillader derfor forskeren at fokusere på udvalgte egenskaber ved systemet, som vedkommende ønsker at undersøge.

Det er vigtigt at understrege, at man som forsker indenfor it-faget anvender elementer fra flere forskellige kategorier, og at ovenstående kategorier kan kombineres på mange forskellige måder. Nogle få eksempler:

- Man kan konstruere en model af en proces. Dvs. at en proces kan betragtes som et system. Dette kan f.eks. være en udviklingsproces eller en brugsproces.
- En model af en proces kan danne udgangspunkt for en række af eksperimenter, f.eks. ved at benytte modellen til at konstruere en simulation af en proces, som kan afprøves på en computer.
- De formelle metoder kan anvendes til at bevise egenskaber ved en proces-model man har lavet.

Forskning fra idé til effekt: kompleksitet og usikkerhed

Uanset hvilke metoder fra ovennævnte kategorier man anvender, så følger de overordnet den samme proces. Med udgangspunkt i en idé ønsker man at opnå en effekt.



Idéen kan være:

- en hypotese man ønsker afprøvet
- et problem man ønsker løst
- visionen om et produkt (eller en service, en proces eller en organisation) man ønsker udviklet

Den ønskede effekt er så henholdsvis:

- ny viden baseret på afprøvning af en hypotese
- et løst problem
- et nyt produkt

Hvis idéen (hypotesen, problemet, visionen) er relativ simpel og velforstået er der ofte ikke så langt fra idé til opnået effekt. Ud fra idéen er det ret åbenlyst, hvad der skal gøres for at opnå den ønskede effekt. Disse simple processer betragter vi sjældent som forskning.

Når man bedriver forskning er det omvendt typisk fordi, at den situation, vores idé (hypotese, problem, vision) opstår i, er præget af høj usikkerhed eller/og høj kompleksitet.

Den slags processer kræver omtanke og systematik. Dette har Lars Mathiassen & Jan Stage sammenfattet[1] i følgende princip:

- Hvis vi er i en situation præget af **kompleksitet** er det typisk fordi, at vi har for meget information (kvantitet), og/eller at vi har at gøre med indviklet information (kvalitet). I dette tilfælde har vi brug for en **analytisk** tilgang for at reducere kompleksiteten.
- Hvis vi er i en situation præget af **usikkerhed** er det typisk fordi, at vi har for lidt information (kvantitet), og/eller at vi har at gøre med upålidelig information (kvalitet). I dette tilfælde har vi brug for en **eksperimentel** tilgang for at skaffe mere information.

Bemærk at de to situationer og anbefalede tilgange er hinandens modsætninger: hvis vi er i en situation præget af kompleksitet og anvender analytiske tilgange, så reducerer vi mængden af information, som vi arbejder med, og dette skaber ny usikkerhed. Og vice versa. Dette ligger naturligt op til en arbejdsproces, hvor man benytter begge tilgange, så man hele tiden tilstræber at have en passende mængde information at arbejde med.

Forskning: En iterativ og inkrementiel proces

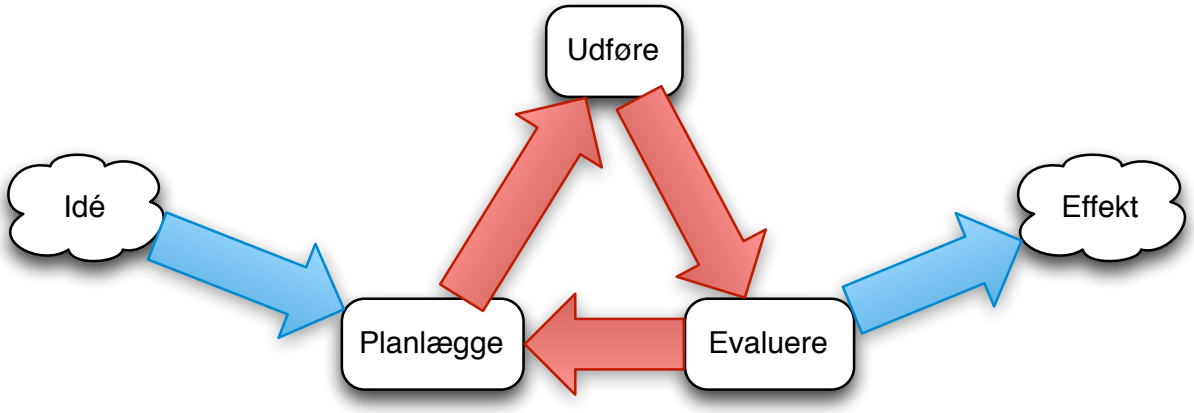
Uanset hvordan vi kombinerer vores analytiske og eksperimentelle aktiviteter, når vi arbejder, har vi brug for at planlægge og evaluere disse aktiviteter, og fordi vi hele tiden bliver klogere i vores arbejde skal planlægningen og evalueringen være en del af en iterativ og inkrementiel proces.

Iterativt betyder, at vi gentager de samme processer flere gange (f.eks. en sekvens af planlægning, kravspecifikation, analyse, design, konstruktion, test og evaluering). Hvis vi ikke er tilfredse med resultatet af en proces, så gentager vi den. Ikke bevidstløs repetition af præcist det samme, men med variationer, som vi tror, kan bringe os tættere på den ønskede effekt.

Inkrementielt betyder, at vi for hver gang, vi itererer, tilføjer noget af værdi til den effekt, som vi forsøger at opnå, hvad enten det er viden, problemløsninger eller produkter.

[1] "The Principle of Limited Reduction in Software Design", 1992.

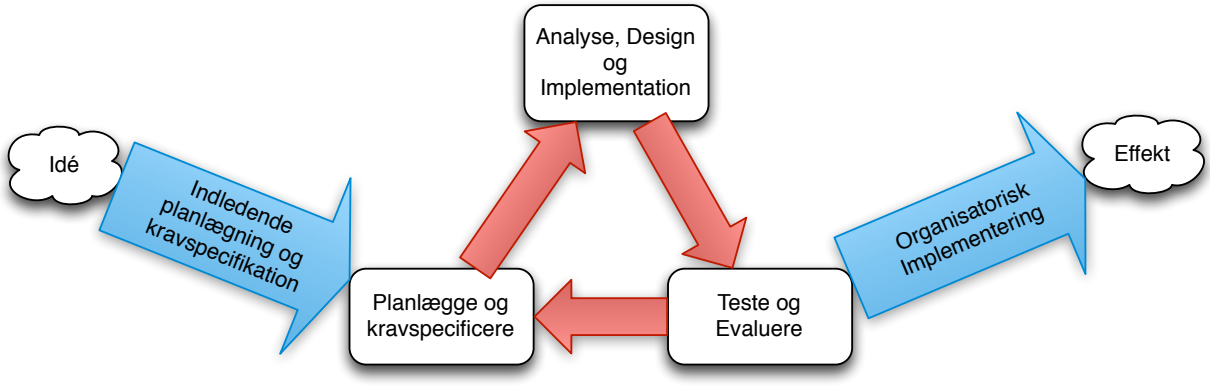
Dette har vi udtrykt i nedenstående figur, der illustrerer en iterativ og inkrementiel proces, der fører os fra idé til effekt.



Hver iteration består af en sekvens, hvor vi planlægger det skal udføres, så udfører vi det, og endelig evaluerer vi det. Hvis evalueringen resulterer i, at vi har opnået den ønskede effekt, så er vi færdige med processen, og alternativt, hvis vi endnu ikke har nået den ønskede effekt, så går vi tilbage til planlægningsprocessen og justerer vores plan, vores udførsel, og vores evaluering, osv. Dette er naturligvis meget simplificeret: hver iteration (illustreret ved den røde trekant) vil medføre nye idéer og nye effekter. Pointen er blot, at der er en idé, der udløser/igangsætter processen, og der er en effekt, som afslutter processen.

Professionsfagets metoder

Princippet fra forskningsmetoden kendes også fra professionsfagets metoder til systemudvikling, f.eks. iterativ og inkrementiel systemudvikling (a la Unified Process eller SCRUM) og teknikken trinvis forfinelse ("stepwise improvement") kendt fra programmering.



Bemærk at denne figur ikke fortæller **hvordan** opgaven skal nedbrydes eller **hvilke** specifikke metoder og teknikker, man anvender i hver enkelt kasse. Det vil helt afhænge af situationen. Men figuren udtrykker, at man gør det iterativt og inkrementielt.

Eksempel 1: Mht. opgavenedbrydningen (eller projektplanlægningen om man vil), så kan man f. eks. diskutere om man skal følge denne rækkefølge af iterationer for et tænkt projekt:

1. *lave et arkitekturskelet bestående af et databaselag, et servicelag, og et grænsefladelag, men hvor lagene kun er stubbe og drivere uden indhold.*
2. *lave kommunikationen mellem lagene*
3. *implementere en simpel førsteudgave af databasen (til testbrug)*
4. *implementere en simpel førsteudgave af grænsefladen (til testbrug)*
5. *implementere en simpel førsteudgave af servicelaget (til testbrug)*
6. *implementere et antal use-case tilhørende én aktør baseret på ovenstående skelet*
7. *implementere den rigtige (avancerede, ønskede) database*
8. *osv.*

Eksempel 2: Hvis vi er ved at lave et kalendersystem, kan vi f.eks. lave en første udgave, der antager, at alle måneder er lige lange. Det er ikke korrekt, men vi kan da lave noget, hvor man kan indtaste noget og få vist et output (ganske vist forkert, men så er der da "hul igennem"). 2. udgave kan så tilpasses med månedernes korrekte længde, så vi får langt flere korrekte output. 3. udgave kan tage højde for skudår. 4. udgave kan gemme indtastede aftaler i en database. 5. udgave kan dele databasen/kalenderen mellem flere brugere via internettet. 6. udgave kan tilføje en anden brugergrænseflade. 7. udgave inkluderer en App-klient til en smartphone. Osv.

Mht. de specifikke metoder og teknikker, man kan anvende, så findes der et hav af metoder til at planlægge, estimere tidsforbrug og fastlægge krav; lave analyse, design og implementere; samt teste og evaluere. Disse metoder og teknikker opstår i forskellige miljøer med deraf følgende meget forskellige fokus: f.eks. applikationsarkitektur, projektstyring, modellering, teknologier, brugergrænseflader, brugersamarbejde, osv.

Beslutningen vedr. både projektplanlægning samt valg af metoder afhænger generelt af **hvor** der er usikkerhed og kompleksitet i projektet. Ikke bare i det system, man ønsker at konstruere, men også i de ressourcer man har til rådighed: teknologi, udviklere, brugere, tid, penge osv. Heldigvis er der ikke mangel på konsulenter, kurser og lærebøger, der vil sælge netop deres geniale løsning på denne udfordring.

Trinvis forbedring er en ramme for iterativ og inkrementiel udvikling af artefakter. Teknikken beskriver, at enhver udvikling finder sted i henhold til tre ortogonale dimensioner:

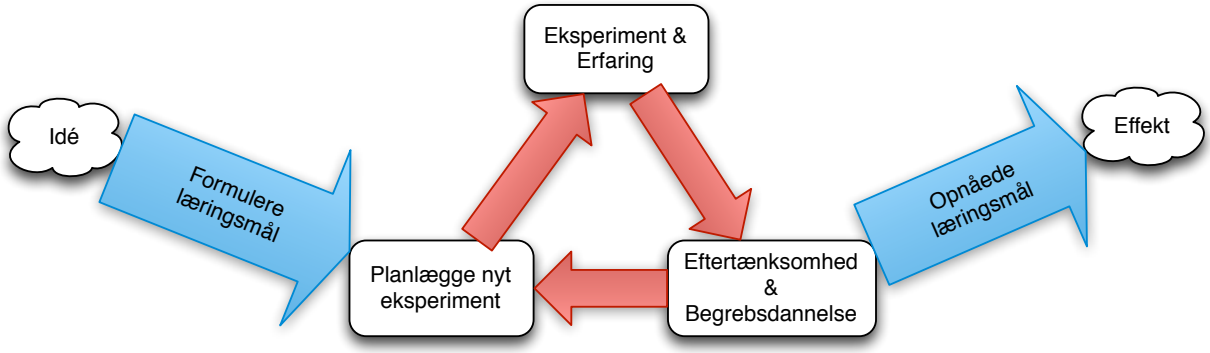
- Fra det abstrakte til det konkrete (svarende til top-down design)
- Fra dele til helheder (svarende til bottom-up design)
- Fra det ustrukturerede til strukturerede (svarende til refaktorerings-teknikker kendt fra programmering og agil udvikling)

Alternativt formuleret kan udviklingen af en artefakt beskrives som en blandet sekvens af konkretisering, udvidelse, og omstrukturering.

Det er oplagt, at trinvis forbedring kan bruges som princip til at planlægge sine iterationer i et udviklingsforløb.

Undervisningsfagets metoder

Princippet fra forskningsmetoden kendes også fra undervisningsfagets metoder. F. eks. kan man nemt reformulere Kolb's læringscyklus til at passe med vores metode[2]:



Det nye it-fag er opbygget omkring 5 didaktiske principper, som vi i det følgende relaterer til vores generelle metode for faget og ovenstående figur.

Et læringsforløb er ikke (nødvendigvis) det samme som et kernestofområde

Det nye fag er opbygget omkring 7 faglige kernestofområder, men konkrete læringsforløb går typisk på tværs af disse områder. Dette er naturligt, da opdelingen i kernestofområder er en måde at strukturere det faglige indhold på, hvorimod læringsforløb er en måde at strukturere indlæringsprocessen på. Dette kan knyttes til vores undervisningsmetode (og illustreres med vores figur) på den måde, at de indledende og afsluttende processer (de blå pile) er knyttet til kernestofområderne, hvorimod den iterative proces (de røde pile) er knyttet til læringsforløbene. Dette kan benyttes ved udformningen af læringsforløb, idet man up-front tager stilling til de ønskede læringsmål i henhold til idéen, man designer det antal iterative forløb (eksperiment-erfaring-begrebsdannelse) eleverne skal igennem, samt tager stilling til, hvordan man vil måle på effekten af læringsaktiviteten i henhold til læringsmålene.

Læringsaktiviteter bør være applikations-orienterede

Traditionelt underviser man "bottom-up" i datalogi og mange andre tekniske fag. Det klassiske eksempel er, at eleverne skal lære at programmere et simpelt "Hello World" program, inden de får lov til at anvende databaser, grafiske brugergrænseflader, osv. Dette kan give mening, hvis formålet med undervisningen er at give eleverne veldefinerede snævre faglige kompetencer. Men for ikke teknisk-interesserede elever kan det give en alt for stejl indlæringskurve, inden de kan foretage sig noget meningsfyldt med teknologien, hvilket typisk giver sig udslag i udfordringer vedr. motivation.

---

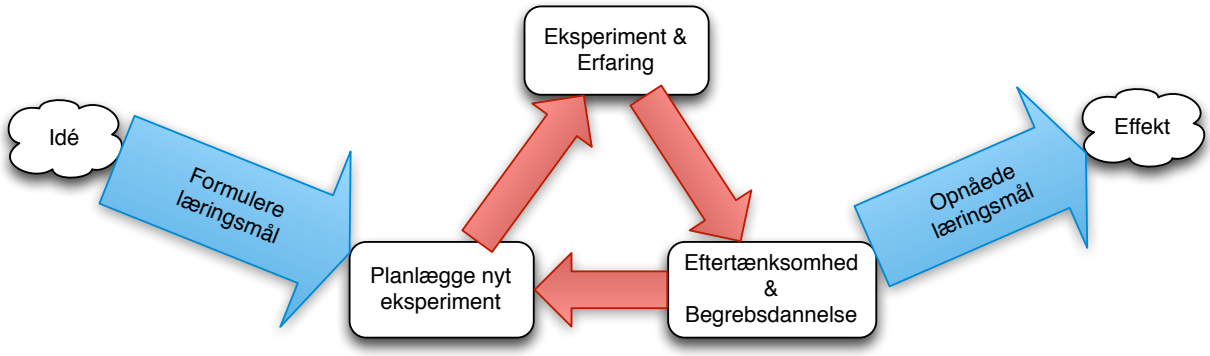
[2] Kolb skelner dog ikke som vi, mellem planlægning og udførsel af eksperiment. Til gengæld er han mere præcis, hvad angår forskellen på eksperiment, erfaring, eftertænksomhed og begrebsdannelse.

I det nye it-fag, hvor man sigter mod at skabe interesse for it-faget generelt, opøve kritisk tænkning, og give et bredere sæt af kompetencer ("computational thinking and practice"), giver det derfor mere mening at tage udgangspunkt i applikationer og teknologier, som eleverne på forhånd kender, er trygge ved og interesserede i: Facebook, YouTube, Photoshop, Skype, osv.

I henhold til vores illustration af fagets metode, betyder det, at når man designer og anvender læringsforløb (den røde trekant), så bør man tage udgangspunkt i applikationer, som eleverne kender og måske alligevel bringer ind i klasselokalet. Progressionen i det iterative forløb vedr. en sådan applikation er udtrykt ved næste princip, og koblingen til læringsmålene (de blå pile) er udtrykt ved det forrige princip.

Læringsaktiviteter bør understøtte en progression fra "at bruge" via "at ændre" til "at skabe"

Dette princip er velkendt fra undervisning i programmering (f.eks. at man bruger en eksisterende variabel, procedure eller klasse inden man selv skal lave en), men vi mener, at det kan generaliseres og anvendes bredt i faget, når man designer læringsforløb.



Eksempel: Hvis man f.eks. designer et læringsforløb, hvor ét af læringsmålene er knyttet til kernestofområdet interaktionsdesign, så kunne man vælge at strukturere forløbet i tre iterationer, der direkte afspejler princippet:

1. *Eleverne prøver i første iteration at **bruge** et eller flere systemer, der illustrerer velkendte mønstre og principper fra interaktionsdesign (f.eks gestalt-lovene).*
2. *Eleverne får i anden iteration udleveret et (dårligt?) interaktionsdesign, som de så skal foreslå **ændringer** til baseret på deres kendskab til mønstre og principper for godt design.*
3. *Eleverne prøver i tredje iteration at **skabe** et interaktionsdesign til deres egen applikations-idé, hvor de prøver at anvende (og teste op i mod) velkendte mønstre og principper.*

Hver af disse tre iterationer skal planlægges, udføres og evalueres (røde pile), og der skal sammenholdes med læringsmålene for de involverede kernestofområder (blå pile).

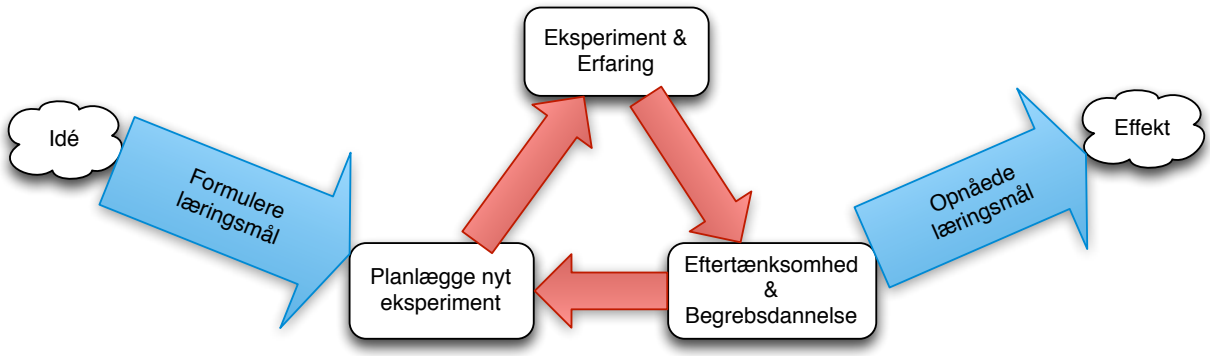
Bemærk at ovenstående kan udføres som simple tænke-højt øvelser, med et simpelt prototype værktøj, eller i en reel programmeringsomgivelse, helt afhængig af elevernes

interesser og kompetencer. Princippet er således ikke knyttet til teknologi og praktisk håndelag, men snarere til forskellige måder at tænke på.

Læringsaktiviteter bør inkludere et antal substantielle "worked examples"

Et "worked example", bestående af en problemformulering og en tilhørende beskrivelse af en systematisk proces for at løse problemet, er et effektivt redskab til at forbedre elevers mentale modeller af problemområder og løsninger. WE's kan således illustrere hvordan tilsvarende problemer kan løses. Princippet kendes også fra f.eks. Design Patterns indenfor objekt-orienteret programmering. WE benyttes ofte sammen med princippet om "Faded Guidance", hvor en række øvelser, der ligner hinanden strukturelt, udføres i sekvens, mens man gradvist reducerer mængden af hjælp fra materialet og eller underviseren.

Dette kan ligeledes knyttes sammen med vores generelle metode for undervisning i faget, da iterationerne kan udføres som en række gennemløb af WE's med stadig mindre hjælp fra underviserens side.



Læringsaktiviteter bør illustrere trinvis forbedring, som en generel teknik til iterativ og inkrementiel udvikling af artefakter

Idet trinvis forbedring er en helt central teknik i professionsfagets metode, så er det oplagt, at man kan strukturere læringsforløb i henhold til denne. Bemærk, at når man anvender trinvis forbedring til at strukturere og inddele sine iterationer i et læringsforløb, så er der to forbedringer (værdiforøgelser) man søger for hver iteration:

- Dels en forbedring af selve artefaktet under udvikling (systemet, applikationen, algoritmen, datastrukturen, grænsefladen, proceduren, osv).
- Dels en forbedring af elevens viden, kompetencer og færdigheder.

Se mere om trinvis forbedring under beskrivelsen af professionsfagets metoder.

En metode for innovation?

Innovation ("fra forskning til faktura") er kendetegnet ved at være markedsdrevet. Hvor forskning sigter mod viden, der er "nyt for verden", og læring sigter mod viden, der er "nyt for den enkelte", så sigter innovation mod produkter, services, organisationer, og processer, der er "nyt/nye for markedet."

Nogle gange er det problem, vi skal løse veldefineret (”lav et program, der løser andengradsligninger”), og andre gange arbejder vi på et problem som måske kun er diffust erkendt og vi måske ikke ved, om overhovedet kan løses, eller om det overhovedet opfylder et behov, som nogen har. F.eks. så er udviklingen af FaceBook og en iPod nok næppe startet med en klar idé om dens anvendelse, men mere med en fornemmelse af, at det måske kunne være sjovt at have et system, som gør sådan og sådan og muliggør dit og dat.

I det første tilfælde vil man ofte starte med at lave en kravspecifikation, der angiver præcist, hvad vores system skal kunne (f.eks. hvad skal indtastes og hvordan skal resultatet præsenteres). I det andet tilfælde (som man kunne kalde innovation), så må vi søge inspiration, være kreative i vores arbejde, lave eksperimenter for at afprøve idéer, få nye idéer og prøve os frem.

De to ovennævnte arketypiske scenarier udspænder et felt mellem veldefinerede problemer på den ene side og gryende erkendte muligheder på den anden side. De to arketyper finder vi sjældent i praksis. Det første måske som opgaver i lærebøger, og det andet måske som temaer for kreativitetsworkshops med vilde konsulenter. Langt de fleste reelle problemer og projekter ligger i spændingsfeltet imellem: noget ligger fast og noget er usikkert – forholdet mellem disse størrelser varierer fra projekt til projekt, og typisk også over tid i det enkelte projekt efterhånden som vi bliver klogere. IT-fagets metoder dækker derfor begge aspekter, og blander det typisk i praksis.

Vi tror på, at man kan skabe arbejdsbetingelser, der øger muligheden for (men bestemt ikke garanterer) at man kan være kreativ, få nye ideer, og skabe nye produkter, services, processer, og organisationer.

Et centralt element i en sådan innovationsproces udgøres af eksperimenter, dvs. at man kan prøve forskellige idéer af i en relevant kontekst. Dette relaterer i høj grad til iterativ og inkrementiel udvikling. Men i stedet for at sigte efter et bestemt foruddefineret (kravspecificeret) mål, så udforsker vi i stedet mulighederne, og generelt kan man sige, at det system, som hver iteration resulterer i, ikke er et mål i sig selv. Den værdiforøgelse som hver iteration resulterer i, er derfor ikke et større eller bedre system, men den øgede viden, som processen har medført (dette kendes også som ”throw-away prototyping”).

For yderligere diskussion af dette henviser vi til Ivan Aaen: Software Innovation (http://iftek.dk/innovation).