INF 1050
UKEOPPGAVER 1: SYSTEMUTVIKLINGSPROSESSER
INNSPILL TIL SVAR

Oppgave 1:
Her er tanken at dere skal diskutere dette i plenum på gruppetimen. Gruppelærere skal komme med innspill for å holde diskusjonen i gang hvis den stopper opp før noe fornuftig har kommet fram. Pass også på å ikke holde på for lenge.

Om oppgaven:
Denne oppgaven går på motivasjonen for faget (systemutvikling/software engineering) som dette kurset dreier seg om. Denne motivasjonen er et sentralt og underliggende spørsmål i diskusjonen om valg og bruk av utviklingsprosesser. Dagens utviklingsmodeller inkluderer temmelig rigide prosesser (vannfall) og ganske løse prosesser som setter programmererens behov mer i sentrum (XP). Ulike grader av rigiditet og formalisme er hensiktsmessig til forskjellig tid. Poenget med punkt d) er at slike likheter på tvers av prosjekter gjør at det er mulig å begynne å snakke om å gjøre ting systematisk. Relater til byggesteinene i enhver prosess (PULS-elementene i SAD/GS).

Tenk over at:

(1) Argumenter for å bruke liten grad av formalisme går ofte på at håndtverk (og kunst) ikke kan styres ved bruk av regler og krav. En av misforståelsene som gjerne oppstår i en slik diskusjon, er at den "kunstarten" som noen mener software-utviklere utøver, foregår på et langt lavere detaljnivå enn det utviklingsprosessene beskriver. Så selv om den overordnete prosjektformen er rigid, så kan likevel programmerere boltre seg i artig programmering innenfor rammene av prosjektet.

(2) Likevel: Ved økende grad av formalisme og krav til dokumentasjon, vil utviklingsprosessen påvirke detaljarbeidet sterkere. En viktig faktor her er at utviklere normalt liker å programmere, men ikke å skrive dokumentasjon. Enkelte er også lite interessert i å modellere for mye. En av grunnene til det siste, er at modellene ofte må endres når programkoden endres. Dette gjør utviklingsprosessen tyngre og mindre morsom. Dermed blir et et viktig argument for at en detaljert utviklingsprosess ikke garanterer et godt resultat, at det er utviklernes kompetanse som avgjør om sluttresultatet blir bra. (At det er utviklernes kompetanse som avgjør om sluttresultatet blir bra, er kjepphester til både Fred Brooks og Bob Glass, to kjente størrelser i software engineering). Dermed spiller det kanskje ikke så mye rolle om en følger en bestemt prosess? Et motargument her er da det faktum at to utviklere ikke nødvendigvis har samme syn på oppgaver som krever koordinering dem i mellom. Dette er det mange eksempler på. Og slik koordinering ivaretas av rigiditeten til en utviklingsprosess.

Oppgave 2:
Dette er et forholdsvis åpent spørsmål uten en ren fasit. Produksjon brukes gjerne om en repeterbar prosess der samme input leder til samme output hver gang. Programvare som aldri har blitt utviklet før, er rimeligvis ikke produsert iht. dette kriteriet. Siden våre omgivelser sett fra systemutviklingens perspektiv stadig er i endring, er behovet for repetisjon nærmest fraværende. Programvare kan jo kopieres så mange ganger vi vil, men det er lite interessant i denne sammenheng. Det er ny design, forbedrete tekniske egenskaper og flere funksjoner vi normalt vil bygge i et nytt prosjekt.

Et poeng er at, ja, for programvare så har utviklingsprosessen mye større tyngde en produktet. Dvs. det å lage støpeformen er det som tar (all) tid, ikke det å støpe produktet, for å bruke en analogi fra en annen type produktutvikling. Men, man prøver jo også å få programvare-produksjon mer lik annen produksjon, idet man prøver å gjenbruke komponenter, samt bruke standardsystemer/standardprogrammer.

Oppgave 3:
Det er for eksempel mulig å dele opp hovedstakeholderne kunde, leverandør og bruker i mer detalj. Leverandøren består gjerne av hel rekke forskjellige aktører (for eksempel produkteiere, prosjektledere, utviklere). Forskning har vist at disse subgruppene innenfor leverandøren ofte har vidt forskjellige prioriteringer og mål med et systemutviklingsprosjekt! Grunnen til at vi fremhever leverandør her, er at faget er "systemutvikling" og som potensielt fremtidige systemutviklere, vil dere kanskje havne i en av rollene produkteier, prosjektleder, utvikler. La oss se nærmere på de tre foreslåtte stakeholderne:

Produkteier (den som er ansvarlig overfor kunden---ofte er produkteier en ansatt i leverandørbedriften, som ikke har IT-faglig bakgrunn, men gjerne økonomisk eller markedsbakgrunn. Det kan også hende at produkteier er en representant for kunden, og som faktisk har IT-faglig bakgrunn, men det lar vi ligge foreløpig):
1) fornøyde sluttbrukere
2) robust system som er enkelt å drifte
3) enkelt å utvide og integrere med andre systemer

Prosjektleder (den som er ansvarlig for selve utviklingen av systemet):
1) levere i henhold til planen
2) levere med en kvalitet som er god nok
3) velfungerende prosjektteam

Utvikler (programmerer, modellerer):
1) kunne jobbe med interessant teknologi (ikke mange som ønsker å programmere i Cobol)
2) frihet til å gjøre egne designvalg
3) gode verktøy og mulighet til å fokusere på oppgaven

Mange glemmer at ledelsen (produkteier) er en viktig stakeholder. Innkjøpspris og kostnader til videreutvikling og drift er viktig. Ikke minst er det viktig at det er enkelt å finne personer som kan bidra i videreutvikling. F. eks. kan det være vanskelig å finne personer som kan programmere i Cobol (og disse har derfor god timelønn også i 2009). Videre er det

viktig at arkitekturen er åpen og enkel å endre slik at systemet kan leve lenge og at årlig kostnad er lav. Husk at det å bygge et nytt system kan koste mye (for mye?).

Andre stakeholder kan være:
- QA-folk, altså de innad i systemutviklingsbedriften (leverandøren) som skal kvalitetssikre systemet eller programvaren. (QA=quality assurance)
- Utviklere på andre systemer som skal kommunisere med systemet det er snakk om.
- Ledelse (behøver ikke være det samme som produkteiere)
- Support, dvs., de stakkarne som skal ta imot telefoner fra sure brukere av systemet når ting ikke virker, eller systemet er for vanskelig å bruke.

Oppgave 4:

a) Dette er en situasjon hvor en utvikler, som kanskje ikke har beslutningsmyndighet, men som kanskje har dyp teknisk innsikt, utfordrer en forretningsmessig beslutning tatt på ledelsesnivå. Dette kan føre til en hel haug med "interessante" mellommenneskelige konflikter. (Et eksempel på at de forskjellige sub-stakeholders innenfor hoved-stakeholderen "leverandør" ofte vil ha forskjellige syn på ting.)

Dersom det som utvikleren sier stemmer, vil produkteier trolig komme i en ubehagelig skvis. Produkteier, som her representerer leverandøren overfor kunden, har skrevet en kontrakt med kunden om a levere et system som skal løse kundens behov. Nå viser det seg at systemet kan komme til å koste mer enn det vil smake. Prosjektleder vil også komme i en ubehagelig situasjon, siden hun står mellom en produkteier som kanskje kontraktsmessig må gå med på kundens ønsker, mens utviklerne hennes er overbevist om at hele prosjektet bør skrinlegges.

Merk at mange mennesker ofte tenker mer på sin egen situasjon enn hva som er til virksomhetens eller samfunnets beste. Dette kan ende i et politisk spill som videre kan få ringvirkninger langt utover prosjektet. De brukerne som er tiltenkt rollen å styre systemet blir gjerne lite populære og betraktet å være en del av "fienden". Produkteier som har argumentert for å bygge systemet kommer i en svært vanskelig posisjon. Mye penger kan allerede ha blitt brukt og det forventes at systemet blir ferdigstilt som planlagt. Det er ofte mye prestisje i slike prosjekter og ingen leder ønsker å stå ansvarlig for en fiasko. Kan man stole på at vurderingen til den ene utvikleren (varsleren) er riktig, og at det er best å stoppe prosjektet? Utvikleren vil sannsynligvis komme i en svært presset situasjon fordi slike vurderinger som han har gjort her aldri er 100 % sikre. Noen vil kanskje stille spørsmålet om hvorfor ikke utvikleren oppdaget dette før? Kanskje noen til syvende og sist vil forsøke å legge alt ansvaret over på utviklerne?

b) Dette er en situasjon som demonstrerer at det er sannsynlig at feil prosess ble valgt i utgangspunktet. Produkteier vil måtte forsvare seg overfor kunden, men vil kanskje internt si at dette er prosjektleders ansvar. Prosjektleder vil nok være den som sitter i en skvis her, fordi han bør ha best forutsetninger for å vite hvilken utviklingsmodell som bør velges og hvilke risiki som følger med valget. (Dersom en velger fossefall bør en finne en eller annen måte å forbedre muligheten til å oppdage slike feil som har skjedd her.) Dette er også et hardt tilbakeslag for utviklerne, fordi de har lagt mange dager eller kanskje års arbeid i utviklingen av systemet. De har kanskje jobbet mye overtid, gjort vanskelige kompromisser, osv., alt til ingen nytte.

Oppgave 5 Innspill til svar:
a) Kvaliteten på kravene og hvordan disse er dokumentert vil rimeligvis påvirke programmererens evne til å overføre dem til et kjørende program. Dersom programmereren misforstår kravene, vil kvaliteten på sluttresultatet svekkes. Krav kan endres underveis, og dette setter programmereren i en svært vanskelig situasjon. Merk at programmereren ofte får ansvaret for slike problemer.

b) Feil i kravene er ofte de dyreste feilene vi har. Hvis vi følger en fossefallsmodell, kan det ta lang tid fra feilene introduseres til de oppdages i test. Desto lenger tid det tar fra feilen introduseres til den avdekkes, desto dyrere blir ofte feilsøkingen. De mest alvorlige feilene er manglende krav. Vi kan da ikke bruke kravspesifikasjonen til å sikre at løsningen tilfredsstiller behovet.

Oppgave 6 Innspill til svar:
Systemer som vi kjenner bruksmønster, krav og funksjonsbehov svært godt på forhånd. Dette inkluderer systemer der spesifikasjonen nærmest er matematisk definert. Eks: lommekalkulator, kompilator, etc. Endringer på standardprogrammer kan muligens også gjøres etter fossefallsmodellen.

Oppgave 7 Innspill til svar:
Arbeidslivet og arbeidsprosessene i en virksomhet er i stadig endring. Innovasjon bidrar til behov for nye løsninger og systemer som man ikke har tenkt på tidligere. Nye lover og regler vedtas og må følges. Når vi bruker programvaren, oppdager vi stadig bedre måter å designe den på. Man sier gjerne at den første versjonen av et nytt produkt kun bør benyttes av spesielt interesserte, og at det er mye risiko forbundet med å ta det i bruk. Etter hvert som brukerne blir kjent med produktet, får de behov for enklere løsninger som prioriterer hurtighet istedenfor at systemet er enkelt å forstå og lære. Eksempelvis taster erfarne bilagsførere så raskt at de ligger flere bilag (eller linjer) foran maskinen og hva denne klarer å vise på skjermen. Dette gjelder rimeligvis ikke for nybegynnere.

Denne oppgaven poengterer at produktene i systemutvikling, altså programmene og systemene, som kan være gigantiske---at de er nødt til å endre seg hele tiden for å opprettholde konkuransefortrinn, rette seg etter ny lovgivning osv... Og: Slik endring fører ofte til kaos både i utvikling og produkt. Tesen i dette kurset, er at en bevisst form for utvikling (altså en prosess!) vil bidra til kaoskontroll.

***