Systemutviklingsprosessen

En systemutviklingsprosess er de aktivitetene som utføres for å utvikle et datasystem
Aktivitetene varierer, men vil alltid ha elementer av:
-spesifisering av kravene, dvs. hva systemet skal gjøre
-design av systemet (for eksempel lage en datamodell)
-implementering av koden (programmering)
-validering av at systemet gjør det kunden ønsker
-endringer av systemet i forhold til nye og endrede krav hos kunden

Roller
-Utvikler
-Vedlikeholder
-Arkitekt/system designer
-Grafisk designer
-Tester
-Prosjektleder
-Bruker-/kunderepresentant

Systemutviklingsprosess (= faktisk, reell prosess):
-de aktivitetene som utføres i et utviklingsprosjekt

Prosessmodell (=formell prosess)
-En abstrakt, forenklet representasjon av en prosess
- Deskriptiv: beskriver en prosess slik vi mener vi utfører den
- Normativ (preskriptiv): beskriver en prosess slik noen mener den bør være (vanligste betydning)

Nivåer av prosessmodeller:

(tilpasses prosjektets behov)

Random eks:

-Overordnet prosess: kanban-scrum hybrid

-Underordnet prosess scrum

-Under-underordnet prosess kanban

Hvordan tilpasse prosesser?

• Prosesser må tilpasses

-Ingen prosjekter er like

-Mange faktorer påvirker prosessen

• Hva kan tilpasses?

-Antall faser/aktiviteter, roller, ansvarsforhold, dokumentformater, formalitet/frekvens på rapporter og gjennomganger

• Hvordan tilpasse?

1. Identifiser prosjektomgivelser

-utviklingsstrategi, risiko, krav, applikasjonsområde, type kunde etc.

2. Innhent synspunkter fra utviklere, brukere, kunder
3. Definer prosesser, aktiviteter og roller
4. Dokumenter og begrunn tilpasningene

Fossefallsmodellen

• Plandrevet. Separate faser

• Vanskelig å tilpasse endringer i brukerkrav underveis

• Best ved godt forståtte krav og når det er lite sannsynlig med mye endringer underveis

-Men få systemer har stabile krav …

• Brukes mest i store prosjekter som gjerne utvikles på ulike steder. Plandreven utvikling gjør det enklere å koordinere arbeidet

• Men brukes også i små, godt forståtte prosjekter

Plandrevne (tunge) prosesser

- Prosessaktivitetene planlagt på forhånd. Progresjon måles i henhold til planen
- En tung prosess inkluderer mange aktiviteter og ofte roller. Krever formelle, detaljerte og konsistente prosjektdokumenter
- Ofte "for-tunge", dvs. vektlegger aktiviteter som gjøres tidlig i prosessen (planlegging, analyse & design)

Smidige (lette) prosesser

- Planleggingen gjøres litt etter litt (inkrementelt)
- Enklere å endre prosessen for å tilpasse endrede krav fra kunden
- Fokuserer mer på fundamentale prinsipper (f.eks. "kontinuerlig testing"). Har færre formelle dokumenter og er ofte mer iterative

Et **inkrement** er et tillegg i funksjonaliteten – et aspekt ved systemet
En **iterasjon** er en syklus i utviklingen – et aspekt ved prosessen

Rational Unified Process (RUP)

- Rammeverk for å bygge arkitektur/UML-modeller

Komponenter og gjenbruk

Eksisterende programvare gjenbrukes i større eller mindre grad utviklingen av nye systemer

Service-orientert arkitektur (SOA)

- Brukes for å utvikle distribuerte systemer der komponentene er selvstendige tjenester
- Tjenestene vil kunne utføres på ulike maskiner fra ulike tjenesteleverandører
- Standard protokoller har blitt utviklet for å støtte kommunikasjon og utveksling av informasjon

Smidig

plandrevet vs smidig

- Programmeringsfokuserte metoder
  - Ekstrem programmering (XP)
  - Eksperiment på parprogrammering
- Prosessfokuserte metoder

Scrum (tidsboksbasert)

Velg noen prioriterte oppgaver og jobb med dem i faste tidsintervaller med definerte oppstarts- og avslutningsaktiviteter (Scrum)

- Planleggingsfasen: overordnede mål for prosjektet etableres og programvarearkitekturen designes
- Gjennomføringsfasen: en serie med sprintiterasjoner, der hver iterasjon leverer et inkrement av systemet
- Avslutningsfasen: nødvendig dokumentasjon som hjelp-funksjoner og brukermanualer fullføres, og

Kanban (flytbasert) (toyota - lean)

Fokuserer på at oppgaver skal ”flyte” uten avbrudd gjennom de nødvendige aktivitetene til de er ferdige

- Fokus på gjennomstrømningshastighet på arbeidspakkene = antall features/brukerhistorier implementert per tidsenhet
- Begrense antall arbeidspakker som det jobbes med i parallell (Work In Progress) for å hindre flaskehalser WIP totalt eller per tilstand
- Antakelse: J-høyere WIP, j-saktere flyter arbeidspakken gjennom arbeidsprosessene
- Når en pakke er ferdig, kan man etterspørre en ny som man begynner å jobbe med (pull)
- Slakk i tidsplanen er OK, dvs. en utvikler vil kunne vente hvis det optimaliserer overordnet flyt
- Mindre fokus på estimering
- Kommer fra Lean systemutvikling (toyota)

Prosjektledelse

- Forretningsplan (Hvorfor?)
- Organisasjon (Hvem?)
- Kvalitet (Hva?)
- Plan (Hvordan? Hvor mye? Når?)
- Risik-(Hva hvis?)
- En risiko
    - -er en sannsynlighet for at uønskede omstendigheter skjer
    - – Prosjekt-risikoer vil ha effekt på tidsplanen og/eller ressurser
    - – Produkt-risikoer vil ha effekt på kvaliteten eller av programvaren som utvikles
    - – Forretnings (Business)-risikoer vil ha effekt på organisasjonen som utvikler eller eier programvaren
- Endring (Hvilke endringer gir systemet?)

Krav

Hva er et krav?

-alt fra en abstrakt beskrivelse av en tjeneste til en detaljert, matematisk beskrevet funksjon

En kravspesifikasjon kan ha flere formål:

- -Basis for anbud (rom for fortolkninger, ulike tilbydere vil kunne tilby ulike måter å løse kundens behov på)
- -Basis for kontrakt (detaljer)
- -Basis for design og implementasjon av systemet

Typer krav

• Brukerkrav

- -Krav uttrykt i naturlig språk og diagrammer som viser tjenestene (funksjonene) til systemet og føringer som gjelder
- -Skal forstås greit av kunden

• Systemkrav

- -Strukturert, detaljert beskrivelse av systemets funksjoner og føringer som gjelder
- -Definerer hva som skal implementeres
- -Utgangspunkt for kontrakt mellom oppdragsgiver (kunde) og utviklerorganisasjon

Funksjonelle krav

Hva systemet skal gjøre?

- -Hvilke tjenester (funksjoner) systemet skal tilby?
- -Hvordan det skal reagere på ulike typer input?
- -Vil kunne beskrive hva systemet ikke skal gjøre også

Ikke-funksjonelle krav

Hvordan systemet skal implementere de funksjonelle kravene

Typer ikke-funksjonelle krav

Produktkrav: sikkerhets-(kryptering), brukbarhets-, effektivitets-(ytelse), pålitelighets-(feilrater).
Organiastoriske krav: miljø-, utviklings-(kostnader, ressurser, gjenbruk),
Eksterne krav: Legislative-, sikkerhets-, etiske-, regler-

Domenekrav

Krav som settes av fagområdet systemet designes for.

Utfordringer

- Forståelighet
  - Kravene er ofte uttrykt i spesifikke domenespråk
  - Disse er ofte uforståelige for systemutviklere
- Implisitt
  - Domenespesialister kjenner ofte fagområdet så godt at de ikke tenker på å gjøre domenekravene eksplisitte
- En god systemutvikler har ofte god domenekunnskap. Industri og næringsliv etterspør ofte begge deler

Retningslinjer for skriving av kravspec

- Bruk et standard format på alle krav
- Bruk "må" for absolutte krav og "bør" for ønsker
- Uthev teksten på spesielt viktige deler
- Unngå IT-sjargong
- Inkluder forklaring på hvorfor et krav er nødvendig

Modellering

-Det sentrale med systemmodellering er å utvikle abstrakte modeller av et system, der hver modell representerer ulike perspektiver av systemet.
-Systemmodellering er viktig for å forstå funksjonaliteten i et system og modellene brukes til å kommunisere med kundene og til dokumentasjon
-En modell er en abstrakt oversikt av et system

Grafiske Modeller

-Et godt hjelpemiddel i diskusjonen om systemet
-Ikke-komplette eller ukorrekte modeller kan være OK så lenge formålet er å bidra til diskusjon
-Brukes ofte som en sentral del i dokumentasjon av et eksisterende system
-Modeller bør representere systemet korrekt, men trenger ikke være komplett
-En detaljert systembeskrivelse kan brukes til å implementere systemet
    -Modellen må både være korrekt og komplett

Kontekstmodeller

Kontekstmodeller viser hvordan et system relaterer seg til andre systemer og prosesser

Bruksmønsere – use case diagram

-brukes til å beskrive interaksjonen mellom brukere og systemer.
-beskriver interaksjonen mellom et system og eksterne aktører

Aktivitetsdiagram

se oblig 3

Sekvensdiagram

-brukes til å beskrive interaksjonen mellom brukere og systemer.
-brukes til å modellere interaksjonen mellom aktørene og objektene i et system
-viser sekvensen av interaksjoner som skjer under et gitt bruksmønster (spesielt for hovedløp)
-angir hvordan metodene i objektene anvendes

Klassediagram

-blir brukt i utviklingen av systemmodeller for å vise klasser i systemet og assosiasjoner mellom disse klassene

-En klasse kan bli sett på som en generell definisjon (mønster) av objekter som er instanser av klassen

-En assosiasjon mellom to klasser angir at det er en forbindelse mellom disse klassene

Tilstandsdiagram

-Mange systemer er drevet av ”hendelser”, med minimal data prosessering

-Modellene viser hvordan et system reagerer på eksterne og interne hendelser

-Baseres på antagelsen av at systemet har et endelig antall tilstander og at hendelser (stimuli) fører til at systemet går fra en tilstand til en annen