Softwareudvikling

Mads Thede

Last updated: April 15, 2024

Contents

1 introduction

Faser i traditionel softwareudvikling

1. Foranalyse
2. Analyse
3. Design
4. Implementering/programmering
5. Test
6. Idriftsættelse/deployment
7. Drift/vedligeholdelse
8. Udfasning

Typer as systemer

- Informationssystemer (IS) er forholdsvis store applikationer som håndterer store datamængder og interagerer med andre systemer.
- Indlejrede systemer som regel forholdsvis små og har en vandtæt specifikation.
- Kunstig intelligens og/eller maskinlæring, går kort fortalt ud på at løse vanskelige problemer

Database Management Systems (DBMS)

Et databasesystem skal muliggøre oprettelse af databaser ved at definere deres struktur med et data definition language. Det skal også tillade brugere at udføre forespørgsler og ændre data ved hjælp af et data manipulationsprog. Systemet skal understøtte langvarig lagring af store datamængder, sikre datakonsistens selv ved nedbrud, og håndtere samtidige brugeradgange på en måde, så deres handlinger ikke fører til inkonsistente data.

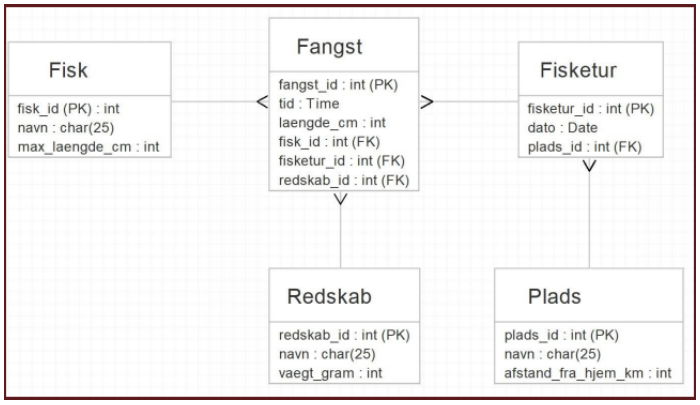
Relationelle database systemer

Tom Codd fremsatte sin relationelle model i 1970. Her skulle data organiseres i tabeller, kaldet relations. Brugeren skulle ikke bekymre sig om den faktiske lagring af data - det håndteres behind the scenes. Tilgang til databasen fandt sted ved hjælp af et højniveau sprog kaldet *structured query language*(SQL).

Entity relationship diagram - ERD

Beskriver relationen mellem de forskellige tabeller. I et ER-diagram modelleres en 1:M relation (fx en fiskeart kan indgå i mange fangster), således at foregreningen af den ene ende af forbindelsen mellem tabellerne vender ind imod den tabel, hvor fremmednøglen er.

[H]



2 User stories

User stories er en simpel måde at skabe krav der skaber værdi for kunden.

Eksempel: "As a banking Customer, I want to transfer funds within my own accounts so that i can move some balance across my accounts."

Hvordan vil systemet kunne bruges

Kode skal kun skrives hvis det skaber værdi.

Alt produktion tager udgangspunkt i en user story.

"In order to receive benefit as a role, I can goal/desire."

3 Udviklingsmetoder

3.1 Agil udvikling

Agil betyder forandringsparathed. Centrale værdier for agil softwareudvikling.

Individer og interaktioner er vigigere end processer og værktøjer. Software, der virker, er vigtigere end omfattende dokumentation. Samarbejde med kunden er vigtigere end kontraktforhandling. At kunne reagere på forandringer er vigtigere end at følge en plan.

The agile enterprise

- Der skal være en stærk virksomhedsideologi
- Der skal være en stærk virksomhedskultur med afsæt i ideologien.
- Man skal kunne agere proaktivt.
- Man skal kunne reagere reaktivt.The agile enterprise
- Forandringsparathed skal kunne måles
- Virksomheden skal råde over en række genbrugelige komponenter, som nemt kan kombineres.
- Ovennævnte plug-in kompatibilitet understøttes aktivt af udviklende standarder.
- Medarbejderne skal kunne eksperimentere frit i selvorganiserende grupper
- Virksomheden skal facilitere videndeling.
- Koordinering finder sted på individniveau

3.2 Kanban

Kanban er en metode til at styre flowet af arbejdet. Man bruger "noter" på "tavler" til at visualisere og holde styr på projektet.

Ved Kanban har man et commitment point, hvor vi beslutter os for at det skal udføres. Dertil er der lead time, som er hvor lang tid det tager at gennemføre. En vigtig begrænsing er WIP limits. Work in progress limits sikrer at man færdiggører opgaver og ikke bare tilføjer flere.

3.3 Rubber duck debugging

Ved at forklare sit problem kan man bedre forstå det.

3.4 Extreme programming

Vi prøver ikke at at forudse alt hvad der skal laves. I stedet fokuseres der på det mest værdifulde og så starter vi der. Da der ikke er lavet overordnede planer er der stor risiko for kaos. Der er derfor oprettet forskellige regler for at holde styr på kaosset. Det gøres ved forskellige niveauer af feedback.

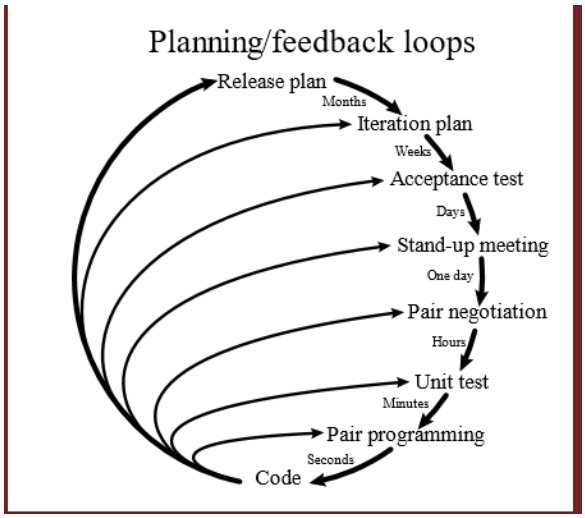
Pair programming

I extreme programming programmerer man ikke alene. Man gør det altid i par. Den ene skriver kode og arbejder med at finde den umiddelbare løsning. Den anden er "navigatør/observatør" og laver review af koden og er opmærksom på den længeresigtede planlægning.

Stand-up meetings

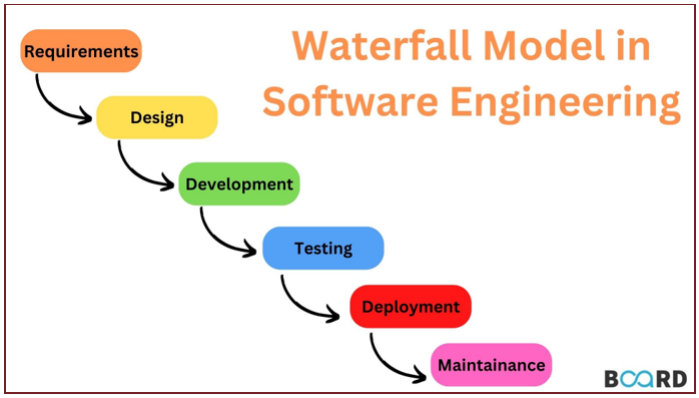
Man mødes alle på holdet først om dagen. Mødet foregår stående så det ikke tager for lang tid. Man fortæller hvad man er i gang med at arbejde på og om der er problemer. Derved ved hele holdet hvor alle er og muligheder for samarbejde forbedres.

Acceptance tests Er det vi har lavet acceptabelt for kunden? Der laves tests for kunden der viser det nuværende system. Eventuelle uoverensstemmelser kan rettes hurtigt i projektet.



3.5 Vandfaldsmodellen

Man fortsætter først når forrige skridt er gennemført. Passer godt med uvikling af systemer, hvor forandringer er dyre. f.eks. udvikling af tog. Modsat er det næsten umuligt at udvikle et system top-down.



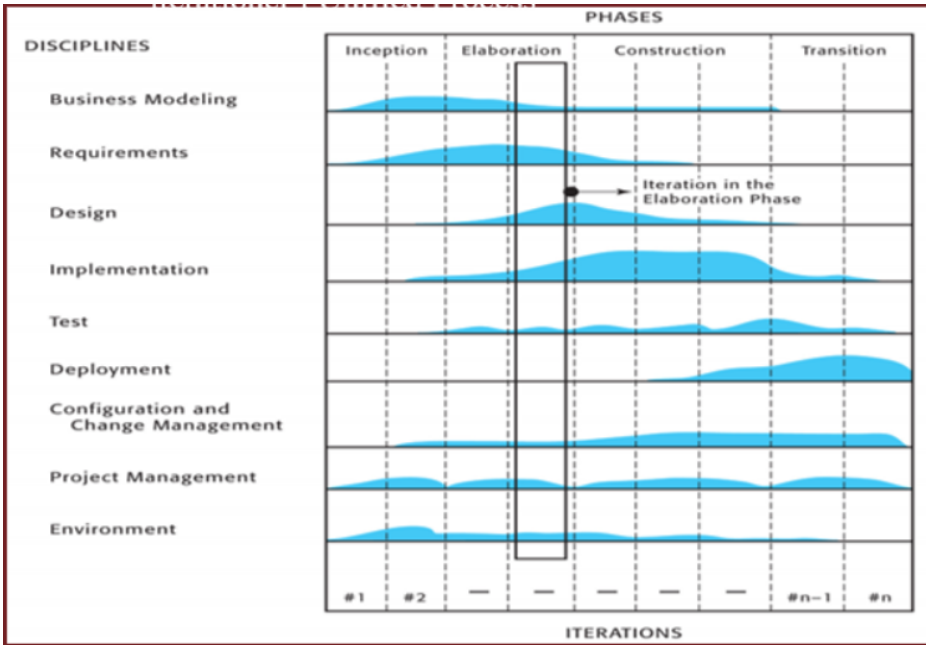
3.6 Iterativ softwareudvikling

Iterativ softwareudvikling er en metode hvor man gentager cyklusser af planlægning, implementering, test og evaluering. Modsat vandfalsmodellen er det muligt at lave ændringer undervejs. Dette gør det muligt at lave store projekter som kan være svære at overskue til start.

Iterativ softwareudvikling handler om at omfavne forandring i form af tilbagemeldinger og tilpasning. Det kan godt være at kunden får nye ønsker til produktet under testing og derfor skal man være klar til at lave noget af det om.

Fordele ved iterativ udvikling

Det hjælper med tidlig lindring af risici (tekniske, krav, anvendelighed etc.) Tidlige synlige fremskridt. Tidligt feedback, brugerinvolvering og engagement. Udviklingsprocessen kan forbedres on the fly.



Iterativ softwareudvikling består af **4 faser**:

1. Inception - approximate vision, business case, scope, vague estimates
2. Elaboration - Refined vision, iterative implementation of the core architecture, resolution of high risks, identification of most requirements and scope, more realistic estimates.
3. Construction - Iterative implementation of the remaining lower risk and easier elements and preparation for deployment.
4. Transition - Beta tests, deployment

Inception kan ses som begyndelse, der ikke behøver at tage lang tid. Det er et indledende, hurtigt forløb, som besvarer følgende spørgsmål:

- Hvad er visionen og business casen for projektet?
- Kan det lade sig gøre?
- Køb og/eller byg?
- Første estimat. 100kr? eller millioner?
- Skal vi fortsætte eller stoppe

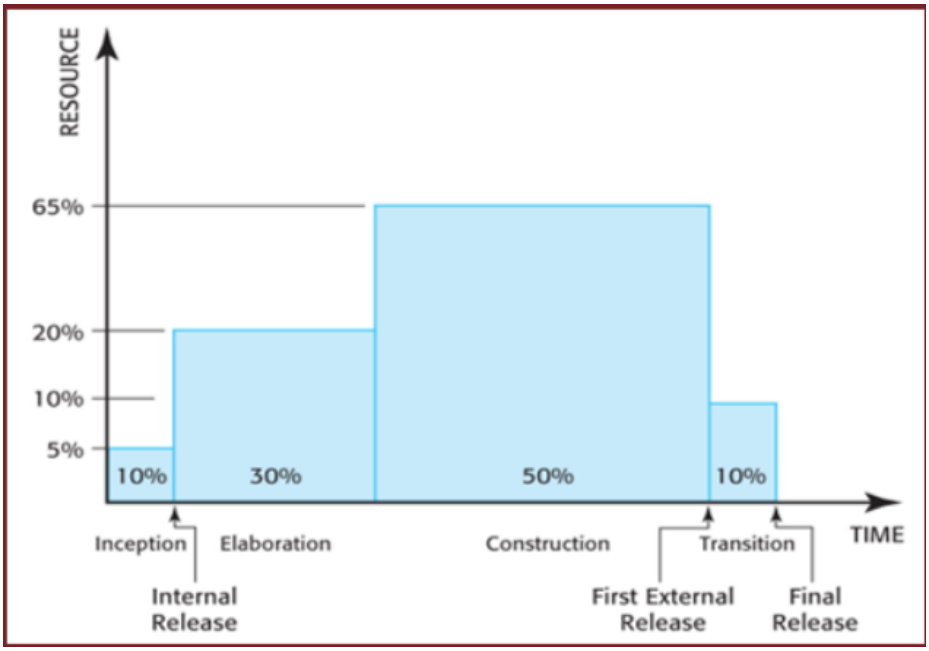
Keywords til inception

- **Vision and business case**: Describes the high-level goals and constraints, the business case, and provides an executive summary.
- **Use-case Model**: Describes the functional requirements, and related non-functional requirements.
- **Supplementary Specification**: Describes other requirements.
- **Risk list and Risk Management**: Describes the business, techhnical resource, schedule risks, and ideas for their mitigation or response.
- **Prototypes and proof-of-concepts**: To clarify the vision, and validate technical ideas.
- **Iteration plan**: Describes what to do in the first elaboration iteration.
- **Phase Plan and software Dev. plan**: Low-precision guess for elaboration phase duration and effort. Tools, people, education, and other resources.
- **Development case**: A description of the customized UP steps and artifacts for this project. In the UP, one always customizes it for the project.

Discipliner

1. Modellering af forretning
2. Fastlæggelse af krav
3. Design
4. Implementering - skriv programmerne
5. Test softwaren
6. Deployment - integrer softwaren i virksomhedens organisation
7. Konfigurations- og ændringsstyring
8. Projektledelse
9. Metode- og værktøjsunderstøttelse

Resursefordeling



Best practices til softwareudvikling

Arbejd først på de mest risikable og værdiskabende features. Involver brugerne kontinuerligt til evaluering, feedback og krav. Byg en sammenhængende basisarkitektur i de tidlige iterationer. Test ofte koden for at finde fejl. Modelér software visuelt (med UML).

Krav FURPS+

- Functional - features, capabilities, security
- Usability - human factors, help, documentation
- Reliability - frequency of failure, recoverability, predictability
- Performance - response times, throughput, accuracy, availability, resource usage
- Supportability - adaptability, maintainbility, internationalization, configurability

The "+" in FURPS+ indicates supplementary and sub-factors, such as: Implementation, interface, operations, packaging, legal.

4 SQL

Column Constraints

- Primary key: Constraint can be used to uniquely identify the row.
- Foreign key: Constraint can be used to enforce referential integrity.

Joins

Group by Gruppering efter en eller flere kolonner.

Having

5 Git

6 Normalisering af databaser

Normalisering af en database er processen hvorved en relation/tabels funktionelle afhængigheder og primære nøgler analyseres med henblik på visse designkrav, som kan formuleres meget teoretisk. I runde tal skal vi undgå tomme felter/celler og undgå redundante data med risiko for inkonsistens.

Definition En tabels normalform refererer til den højest normalformsbetingelse en tabel opfylder.

Første normalform

Hvis en tabel skal opfylde første normalform, må den kun indeholde atomare attributter, og der må ikke være gentagelser af attributter.

Anden normalform

Hvis første normalform er opfyld, og, hvis der er tale om en sammensat primærnøgle, alle attributter afhænger af hele nøglen, og ikke kun en del af den.

Eksempel på en tabel der ikke opfylder 2. normalform:

Essn	Pno	Hours	Ename

Dette opfylder ikke 2. normalform, da Ename kun afhænger af Essn og dermed ikke hele primærnøglen. Det skyldes at hele primærnøglen består af Essn og Pno.

Tredje normalform

En tabel opfylder tredje normalform, hvis den opfylder 2. normalform, og hvis der ikke forekommer transitive afhængigheder, dvs. at der ikke er attributter, som alene afhænger af en anden ikke nøgle-attribut.

Et eksempel på dette kan være en tabel over en person med attributterne

CPR, Navn, Adresse, Postnummer, By

Hvor by afhænger af postnummer, som ikke er en del af primærnøglen.

Fjerde normalform

Selv hvis attributterne begge afhænger af primærnøglen men ellers er uafhængige skal de deles op.

Femte normalform

Også kendt som "Projection-join normal form" er en normalform, der er designet til at minimere redundans i en database.

7 Software Arkitektur