Objektorienteret Modellering

Introduktion

UML (Unified Modeling Language) bruges til at visualisere, specificere, konstruere og dokumentere software systemer. Det giver et fælles sprog for udviklere, arkitekter og interessenter til at forstå systemets struktur og adfærd. UML muliggør en effektiv kommunikation om designvalg og systemarkitektur. Ved at bruge UML kan komplekse systemer opdeles i overskuelige diagrammer, hvilket letter både udvikling og vedligeholdelse.

UML bliver brugt lige fra omfattende stringent dokumentation og design af store software systemer til hurtige kladder på en whiteboardtavle for at facilitere kommunikation mellem udviklere.

Brugen af UML-diagrammer i dokumentation af kodeprojekter giver flere fordele:

1. **Klarhed og Overblik:**
   UML-diagrammer giver et klart visuelt overblik over systemets struktur og adfærd, hvilket gør det lettere at forstå komplekse systemer, især for nye udviklere.

2. **Kommunikation:**
   UML-diagrammer fungerer som et fælles sprog, der kan bruges af alle involverede i et projekt, fra udviklere til projektledere og andre ikke-tekniske roller, hvilket letter kommunikationen og samarbejdet.

3. **Fejlfinding og Optimering:**
   Ved at visualisere systemets arkitektur kan UML-diagrammer hjælpe med at identificere designproblemer eller optimeringsmuligheder tidligt i udviklingsprocessen.

4. **Planlægning og Design:**
   UML-diagrammer kan anvendes i de tidlige faser af softwareudvikling til at planlægge arkitekturen og designet af systemet, hvilket reducerer risikoen for dyre ændringer senere i processen.

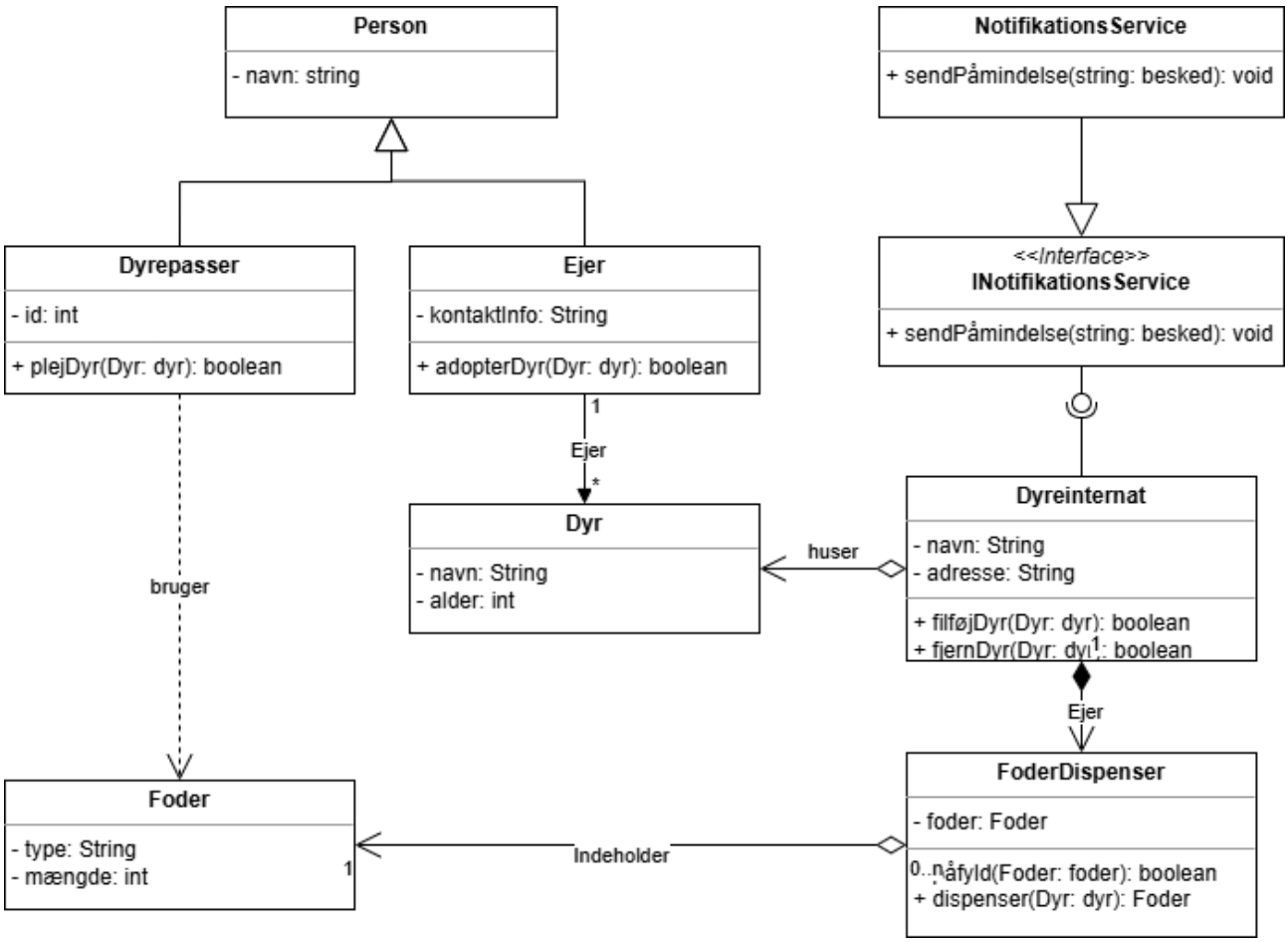
Der er mange forskellige typer af UML-diagrammer der modellerer alle dele af software systemer og arbejdsprocesser (UML kan modelleres via UML), dette dokument indeholder kun nogle af diagrammerne.

Klasse Diagram

Et klassediagram er et af de mest anvendte UML-diagrammer til at beskrive den statiske struktur af et system. Det viser klasser, deres attributter, metoder og relationer som arv, associationer og afhængigheder. Klasse diagrammer anvendes i softwareudvikling til at planlægge og dokumentere systemdesign, før implementeringen påbegyndes.

Elementer i et klasse diagram

- **Klasser**: Beskriver objekttypen og dens egenskaber.
- **Attributter**: Variabler tilknyttet en klasse.
- **Metoder**: Funktioner der definerer en klasses adfærd.
- **Relationer**:
    - **Association**: En relation mellem to klasser, f.eks. en kunde har en adresse. Repræsenteres som en simpel linje mellem klasserne.
    - **Aggregation**: En hel-del relation, hvor delene kan eksistere uden helheden. Modelleres med en tom rombe ved helhedsklassen.
    - **Komposition**: En stærkere hel-del relation, hvor delene ikke kan eksistere uden helheden. Modelleres med en udfyldt rombe ved helhedsklassen.
    - **Afhængighed**: En svag forbindelse mellem klasser, hvor en klasse bruger en anden kortvarigt. Repræsenteres med en stiplet pil.
    - **Arv**: En mekanisme, hvor en klasse nedarver egenskaber og metoder fra en anden klasse. Visuelt modelleres det som en pil med en tom trekant, der peger mod superklassen.
    - **Interfaces**: Definerer en kontrakt, som klasser kan implementere. Modelleres med en cirkel (lollipop notation) eller en klasse med stereotype <>.
        - **Expose interface**: En klasse kan gøre et interface tilgængeligt for andre klasser. Det kan modelleres med en linje med en cirkel for enden.
        - **Use interface**: En klasse kan implementere et interface for at tilføje bestemt funktionalitet. Det kan modelleres med en linje med en halvcirkel for enden der omslutter cirklen fra **Expose interface**.



Object Diagram

Et object diagram er en instans af et klassediagram og viser en øjeblikkelig tilstand af systemet ved runtime. Det anvendes til at verificere klasse diagrammer ved at vise, hvordan objekter interagerer i praksis.

Eftersom et objekt diagram er en instans af et klassediagram så kan det modelleres med den samme notation og strukturer.

Brug af objekt diagrammer

- **Visualisering af et system i en given tilstand**: Object diagrammer viser en øjeblikkelig snapshot af systemets tilstand på et specifikt tidspunkt, hvilket gør dem nyttige til at forstå systemets nuværende struktur.
- **Forståelse af objektrelationer**: Diagrammerne viser hvordan objekter hænger sammen og kommunikerer, hvilket hjælper med at identificere afhængigheder og potentielle designproblemer.

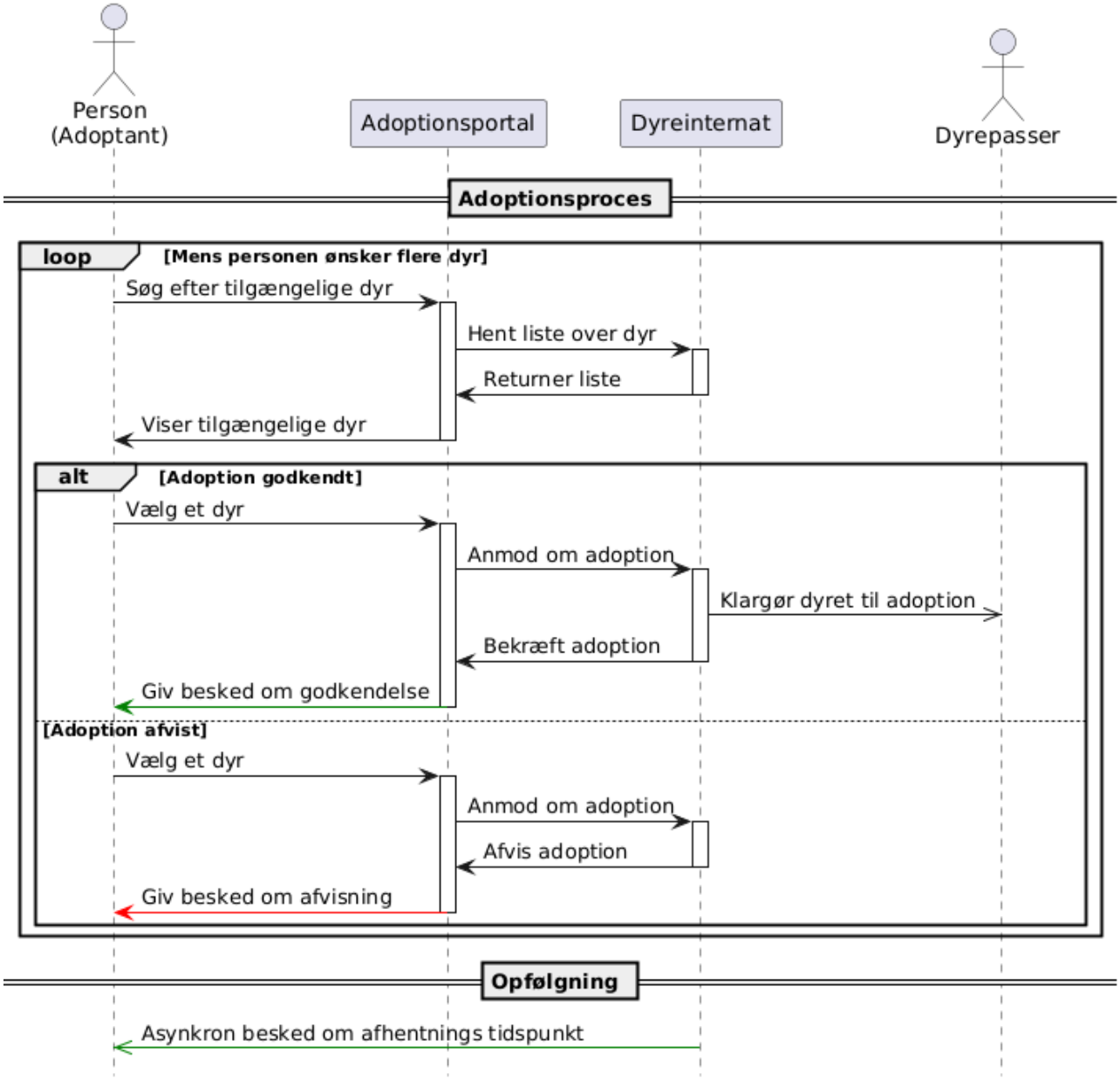
- **Test af systemets design og interaktioner**: Object diagrammer bruges til at validere systemets design ved at simulere objekters interaktioner i en given tilstand, hvilket kan afsløre fejl eller ineffektive strukturer.

Sekvens Diagram

Et sekvensdiagram bruges til at illustrere interaktionen mellem objekter i et system over tid. Det viser, hvordan beskeder udveksles mellem objekter for at udføre en funktion, hvilket gør det nyttigt til at beskrive use cases, systemadfærd og eksekverings flow.

Elementer i et sekvensdiagram

- **Objekter**: Repræsenteres som aktører eller systemkomponenter og vises øverst i diagrammet med en rektangulær boks.
- **Livslinje**: En stiplet lodret linje, der viser objektets eksistensperiode i interaktionen. Tiden starter i toppen of flyder nedad.
- **Beskeder**: Metodekald mellem objekter, der repræsenteres med pile:
    - **Synkrone beskeder**: Vises med en pil med en fyldt spids, der angiver et direkte metodekald, hvor afsenderen venter på et svar.
    - **Asynkrone beskeder**: Vises med en pil med en åben spids, hvilket indikerer, at afsenderen ikke venter på en umiddelbar respons.
- **Aktivering**: Markeres med en rektangulær boks på livslinjen og angiver, hvornår et objekt er aktivt i en interaktion.
- **Loops**: Anvendes til at angive gentagelse af en sekvens ved brug af en ramme med etiketten "loop".
- **Betingelser**: Angives med en ramme og etiketten "alt" eller "opt" for at vise alternative eller valgfrie forløb i interaktionen.



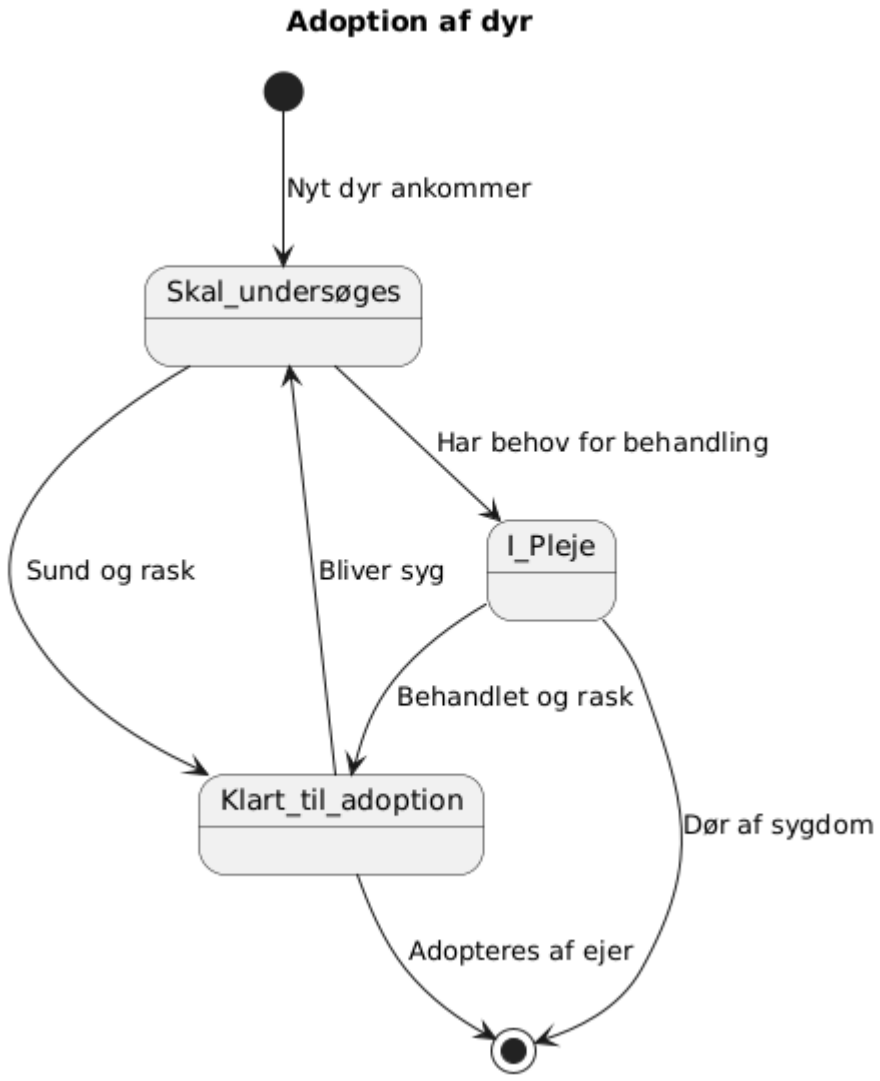
State Machine Diagram

Et state machine diagram anvendes til at modellere et objekts, livscyklus ved at beskrive de mulige tilstande, det kan være i, samt hvordan det bevæger sig mellem disse tilstande som reaktion på hændelser

Et state machine diagram kan ud over objekter også modellere, moduler, pakker, processer, sub systemer og hele systemer (og mere).

Elementer i state machine diagrammer

- **Tilstande**: Repræsenterer de forskellige stadier et objekt kan befinde sig i og vises som rektangler med afrundede hjørner.
- **Overgange**: Viser, hvordan et objekt skifter fra én tilstand til en anden, normalt repræsenteret med pile mellem tilstandene.
- **Hændelser**: Input eller handlinger, der udløser en overgang mellem tilstande.
- **Initial og final state**: Start- og sluttilstande, som repræsenteres af henholdsvis en sort cirkel og en sort cirkel med en ring omkring.



Reverse engineering

For Python-udviklere, er der ressourcer, der kan hjælpe med at generere UML-diagrammer fra eksisterende kodebaser, hvilket sparer tid og sikrer nøjagtighed. For eksempel, pyreverse er et værktøj, der kommer installeret med pylint, og som kan generere UML klasse diagrammer fra Python kode. Det kræver blot, at du peger på den kode, du ønsker at generere et diagram for, og bruger graphviz til at skabe diagrammet.

Kilder og Yderligere Information

- UML Standard:
  https://www.omg.org/spec/UML/
- UML guide:
  https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-practical-guide
- UML Class Diagram Tutorial:
  https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/
- Klasse diagrammer:
  https://www.uml-diagrams.org/class-diagrams-overview.html
  https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/
- Sekvens diagrammer:
  https://www.uml-diagrams.org/sequence-diagrams.html
  https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/
- State machine diagrammer:
  https://www.uml-diagrams.org/state-machine-diagrams.html
  https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-state-machine-diagram/
- Generer UML klasse diagrammer med pyreverse:
  https://www.redshiftzero.com/pyreverse-uml/
- Online værktøjer til at lave UML diagrammer:
  https://app.diagrams.net/
  https://plantuml.com/
  https://www.planttext.com/