

## □ Øvelse 16 (Gruppeøvelse) – Domænemodellering

### □ Information

I den forrige øvelse arbejdede I med kode, hvor en stor del af domænets betydning var skjult. Det gjorde det vanskeligere både at forstå, hvad koden gjorde, og at vurdere, om de implementerede regler var korrekte og sikre.

I denne øvelse arbejder vi den modsatte vej.

I skal forsøge at gøre **domænet eksplicit** ved at udarbejde en simpel domænemodel for den applikation, I tidligere implementerede i opgave [36 - Grundlæggende OOP med C#](#).

En domænemodel er en forenklet model af det område af virkeligheden, som softwaren arbejder med. Modellen skal hjælpe os med at identificere centrale begreber, deres relationer og de regler, der gælder i domænet.

Typisk udvikles en domænemodel med udgangspunkt i virkelighedens domæne og i samarbejde med personer, der har viden om dette domæne.

I denne øvelse arbejder vi delvist med *reverse engineering*: I skal ud fra den eksisterende implementering forsøge at rekonstruere det domæne, som koden repræsenterer.

□ □ I praksis ville en domæneekspert bidrage med viden om domænet og dets regler. I denne øvelse kan I bruge underviseren som domæneekspert og sparringspartner.

---

### □ Domænemodellering – grundprincipper

En simpel domænemodel kan blandt andet beskrive:

- **Centrale begreber eller entiteter:** Ting eller begreber, som er vigtige i domænet, fx Kunde, Ordre og Produkt.
- **Egenskaber:** Information, der beskriver et begreb, fx navn, adresse, ordreID eller dato.
- **Relationer:** Sammenhænge mellem begreber, fx at en kunde kan have flere ordrer.
- **Forretningsregler og invarianter:** Regler for hvilke tilstande, værdier og handlinger der er gyldige i domænet.

Formålet er ikke at lave et klassediagram over hele implementeringen.

I skal forsøge at modellere **domænet**, ikke blot kopiere strukturen fra koden.

□ Eksempel på domænemodel

| Begreb  | Egenskaber                   | Relationer                          |
|---------|------------------------------|-------------------------------------|
| Kunde   | Navn, Adresse, Telefonnummer | En kunde kan have mange ordrer      |
| Ordre   | OrdreID, Dato, Beløb         | En ordre tilhører én kunde          |
| Produkt | Navn, Pris                   | Et produkt kan indgå i flere ordrer |

■ Husk, at en domænemodel er en model af et givet domæne.

Der findes mange forskellige tilgange til domænemodellering, og dette er blot et simpelt eksempel.

En model er en **forenkling af virkeligheden**. Derfor er det vigtigt at overveje, hvilke dele af domænet der er relevante i den kontekst, modellen skal bruges i.

En brugbar model har et passende **abstraktionsniveau**: Den indeholder de begreber og regler, der er nødvendige for at forstå problemet, uden at forsøge at beskrive alle detaljer i virkeligheden.

□ Udvidet eksempel: Egenskaber og forretningsregler

En domænemodel kan også bruges til at gøre regler omkring centrale begreber eksplicitte.

Her er et lille eksempel fra domænet omkring en webshop:

| Begreb | Egenskab      | Eksempelværdi    | Forretningsregel                                                                          |
|--------|---------------|------------------|-------------------------------------------------------------------------------------------|
| Kunde  | Navn          | "Martin Nielsen" | Kunden skal have et registreret navn. Navnet må ikke være tomt.                           |
| Kunde  | Telefonnummer | "22998811"       | Telefonnummeret skal være et nummer, som virksomheden kan anvende til at kontakte kunden. |

| Begreb  | Egenskab | Eksempelværdi | Forretningsregel                                                                                            |
|---------|----------|---------------|-------------------------------------------------------------------------------------------------------------|
| Ordre   | Beløb    | "599,95 DKK"  | Ordrebeløbet må ikke være negativt og beregnes ud fra ordrens indhold.                                      |
| Ordre   | Dato     | "2024-09-24"  | En registreret ordre skal have en dato for, hvornår ordren blev oprettet.                                   |
| Produkt | Navn     | "USB-C Kabel" | Et produkt skal have et navn, som gør det muligt at identificere produktet i domænet.                       |
| Produkt | Pris     | "199,00 DKK"  | En pris må normalt ikke være negativ. En pris på 0 kan eksempelvis kræve en særlig forretningsmæssig årsag. |

Reglerne behøver ikke kun gælde for enkelte værdier.

Der kan også være regler på tværs af flere begreber eller handlinger.

Eksempelvis:

■ En ordre kan kun afsendes, hvis den er betalt.

eller:

■ En udbetaling må ikke bringe en konto under den tilladte saldo.

Sådanne regler kaldes ofte **forretningsregler** eller **invarianter**.

## □ Domænemodellering og sikkerhed

En domænemodel kan gøre de begreber og regler, som systemet skal håndhæve, mere eksplicitte.

Det kan gøre det lettere at:

- forstå, hvad systemet faktisk skal gøre
- opdage fejl i forretningslogikken
- diskutere hvilke tilstande og handlinger der er gyldige

- definere meningsfuld inputvalidering
- identificere antagelser og manglende domæneviden
- teste om implementeringen overholder domænets regler
- sikkerhedsvurdere implementeringen

Domænemodellering erstatter ikke tekniske sikkerhedsforanstaltninger.

Eksempelvis kræver beskyttelse mod SQL injection stadig korrekt håndtering af databaseforespørgsler. Domænemodellen hjælper os derimod med at forstå **hvad systemet skal beskytte, og hvilke regler implementeringen skal opretholde.**

Hvis vi ikke forstår domænet og dets regler, bliver det også vanskeligere at afgøre, om en given implementering faktisk er korrekt og sikker.

---

## ▮ Instruktioner

1. ▮ Identificér de **centrale begreber i domænet.**

Forsøg i første omgang at tænke på domænet frem for på C#-klasser og implementation.

1. ▮ Beskriv **relationerne mellem begreberne.**

Eksempelvis:

▮ En kunde kan have flere konti.

1. ▮ Identificér relevante **egenskaber og værdier**, som hører til de forskellige begreber.
2. ▮ Identificér **forretningsregler og invarianter.**

Overvej blandt andet:

- Hvilke værdier er gyldige?
- Hvilke tilstande må eksistere?
- Hvilke handlinger må udføres?
- Hvad skal altid være sandt?
- Er der handlinger, som kun må ske under bestemte betingelser?
- ▮ Sammenlign jeres domænemodel med den eksisterende C#-kode.

Diskutér:

▮ Kan I genkende domænets centrale begreber og regler i implementeringen?

1. ☐ Undersøg, om der findes steder, hvor implementeringen **ikke håndhæver de regler, som I mener gælder i domænet**.

Det betyder ikke nødvendigvis, at I har fundet en konkret sårbarhed, men det kan pege på områder, som bør undersøges nærmere.

1. ☐ Diskutér til sidst:

Hvordan gør en tydelig domænemodel det lettere at forstå, teste og sikkerhedsvurdere implementeringen?

### ☐ Refleksion

Sammenlign denne øvelse med den foregående øvelse, hvor domænets betydning i høj grad var fjernet fra koden.

Overvej:

- Hvilken forskel gør det for jeres forståelse af koden, når domænets begreber og regler er tydelige?
- Kan man meningsfuldt vurdere, om en implementering er sikker, hvis man ikke forstår det domæne og de forretningsregler, som implementeringen skal håndhæve?
- Hvordan kan en domænemodel gøre det lettere for flere personer i et udviklingsteam at diskutere de samme begreber og regler?

| Element i øvelsen                               | Relateret standard / princip                                  | Kommentar                                                                                                        |
|-------------------------------------------------|---------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------|
| Entiteter og relationer                         | OWASP ASVS 5.0.0 – V1 (Encoding and Sanitization)             | Kræver, at en datamodel for systemet er defineret.                                                               |
| Attributter og forretningsregler                | OWASP ASVS 5.0.0 – V2.2 (Input Validation Requirements)       | Input skal valideres mod forventede formater, længder og semantiske regler – defineret gennem forretningsregler. |
| Diskussion af modellens betydning for sikkerhed | OWASP ASVS 5.0.0 – Bilag D (Security Requirements Definition) | Sikre afklaring af sikkerhedskrav, herunder hvilken data der må behandles                                        |

---

Last update: 2026-09-23 09:28:00