2. Hvad, Hvorfor og Hvordan af Software Rot

- **Hvad:** Manglende evne til kontinuerlig understøttelse af produktet.
- **Hvorfor:** Designet understøtter ikke (uventede) ændringer i krav.
- **Hvordan:** Uventede ændringer bliver improviseret ind i eksisterende kode.

3. Rigidity og Fragility

- **Rigidity:**
  - **Symptom:** Simple ændringer forårsager kaskade af ændringer i relaterede komponenter.
  - **Problem:** Ledere og programmører tør ikke lave ikke-kritiske ændringer.
- **Fragility:**
  - **Symptom:** Løsning af et problem introducerer flere problemer.
  - **Problem:** Software bliver umuligt at vedligeholde.

4. Immobility og Viscosity

- **Immobility:**
  - **Symptom:** En komponent kan genbruges et andet sted, men afhænger af for meget bagage.
  - **Problem:** Softwarekomponenter genbruges ikke, men omskrives.
- **Viscosity:**
  - **Symptom:** Det er lettere at løse et problem ved at lave en hurtig løsning end ved at bevare designet.
  - **Problem:** Softwaren degenererer på grund af for mange hurtige løsninger.

5. Unødvendig Kompleksitet og Gentagelser

- **Unødvendig Kompleksitet (YAGNI):**
  - Problemer med overengineering til fremtidige ændringer.
- **Unødvendige Gentagelser (DRY):**
  - Ændringer implementeret ved copy-paste i stedet for korrekt abstraktion.

6. Opacitet

- **Problem:** Koden bliver sværere at forstå over tid uden ordentlig refaktorering.

7. Bottom Line

- Behovet for at identificere og rette design lugte for at forbedre software vedligeholdelse og pålidelighed.

Noter til "SOLID - SO"

1. Introduktion til SOLID Principper

- Forelæser: Henrik Bitsch Kirk, Aarhus Universitet
- Institution: Institut for Elektroteknologi og Computerteknik

2. SOLID Akronym

- **S:** Single Responsibility Principle (SRP)
- **O:** Open Closed Principle (OCP)
- **L:** Liskov's Substitution Principle (LSP)
- **I:** Interface Segregation Principle (ISP)
- **D:** Dependency Inversion Principle (DIP)

3. Single Responsibility Principle (SRP)

- **Definition:** En klasse skal kun have én grund til at ændre sig.
- **Problemer ved at bryde SRP:** Vanskelig testning, vedligeholdelse og ændring af design.
- **Eksempler:**
    - **IModem Problem:** At kombinere flere ansvarsområder i én grænseflade.
    - **Person Klasse:** Forkert håndtering af flere formateringsansvar i én klasse.

4. Open Closed Principle (OCP)

- **Definition:** Software entiteter skal være åbne for udvidelse, men lukkede for modifikation.
- **Eksempel:**
    - **Editor Klasse:** Problemer med at ændre lagringstype uden at ændre koden.

5. Implementering af OCP

- **Løsning:** Anvendelse af grænseflader til at abstrahere lagring, hvilket gør det muligt at udvide uden at ændre eksisterende kode.

6. Recap

- **SOLID Principper:**
    - **SRP:** En klasse skal have én grund til at ændre sig.
    - **OCP:** En klasse skal være åben for udvidelse, men lukket for modifikation.

Spørgsmål?

- Afsluttende diskussion og spørgsmål.