Loeng 4.
Sissejuhatus objektorienteeritud analüüsi ja disaini

- Programmeerimiskeeled ennetavad nende kasutamise teooriaid:

1. "mittestruktuurne" programmeerimine;
2. struktuurprogrammeerimine;
3. struktuurne disain;
4. struktuurne analüüs.

1. objektorienteeritud programmeerimine: Simula 67, Smalltalk 70.-ndatel;
2. objektorienteeritud disain 80.-ndate keskel;
3. objektorienteeritud analüüs 80.-ndate lõpul.

- **Objektorienteeritud analüüs**: lahendatava probleemi mõistmine ja kirjeldamine.

- **Objektorienteeritud disain**: lahenduse kirjeldamine.

- **Objektorienteeritud programmeerimine**: lahenduse realiseerimine.

- **Radikaalne kaskaadmeetod** (*ingl. k.* radical waterfall approach): analüüsi, disaini, realiseerimist ja testimist teostatakse paralleelselt *prototüüpimise* abil.

- **Prototüüpimine** (*ingl. k.* prototyping): ülesande lahendamine järjest adekvaatsemate poolikute lahenduste loomise abil.

- Suhe objektorienteeritud analüüsi ja disaini vahel: semantilised objektid ⇒ kasutajaliidese objektid, realisatsiooniobjektid, utiliitobjektid.

Joonis 4.1. Suhe objektorienteeritud analüüsi ja disaini vahel.

- **Objektorienteeritud analüüsi sammud**:

1. Kogu kokku süsteemile esitatavad nõudmised.
2. Kirjelda tüüpilised stsenaariumid.
3. Tee kindlaks, millised on objektide kandidaadid ja mille eest igaüks neist “vastutaks” (s. t. millised on objekti *teenused*).
4. Määra kindlaks objektidevahelised relatsioonid.
5. Kirjelda iga objekti nõutavat käitumist ja *teateid*, mida ta teiste objektidega vahetab.
6. Korda samme 1-5.

- **Objektorienteeritud disaini sammud**:

1. Täpsusta objektidevahelisi relatsioone nii, et iga relatsioon kuuluks ühte kolmest liigist: pärimine, sisalduvus ja assotsiatsioon.
2. Määra kindlaks objektide meetodid ja sisemised muutujad (kaasa arvatud atribuudid).
3. Määra kindlaks objektide seosed utiliitklassidega.
4. Üldista objekte $\Rightarrow$ objektide arv muutub.
5. Korda samme 1-4.

Joonis 4.2. Objektorienteeritud analüüs, disain ja realisatsioon.

- Objektorienteeritud analüüs ja disain Ivar Jacobsoni järgi:

1. Objektid vastavad antud probleemvaldkonna nimisõnalistele terminitele. NB!
   - kaks nimisõna võivad olla sünonüümid;
   - termin võib antud valdkonnaga mitte seotud olla;
   - termin võib tähistada objekti atribuuti;
   - termin võib tähistada objekti teenust;
2. Objektid klassifitseeritakse (s. t. moodustatakse pärimishierarhiad) nendevahelise sarnasuse alusel.
3. Erinevate süsteemi võimalike kasutajate (*ingl. k.* actors) poolt algatatud tüüpilised stsenaariumid (*ingl. k.* use cases) määravad kindlaks, mida üks või teine objekt ootab teistelt objektidelt $\Rightarrow$ see defineerib objekti liidese ja määrab kindlaks sisalduvuse relatsioonid objektide vahel.
4. Objektide liidesed määravad kindlaks objektidele rakendatavad operatsioonid.
5. Iga objekti lõplik struktuur defineeritakse informatsiooni põhjal, mida see objekt peab sisaldama.

Joonis 4.3. Stsenaariumite algne kindlaksmääramine.

Joonis 4.4. Stsenaariumite täpsustamine.

- **Objektorienteeritud modelleerimise liigid**:

1. Staatiline ehk andmepõhine modelleerimine: objektide defineerimisel lähtutakse andmestruktuurist, mida üks või teine objekt esindab ja operatsioonidest, mida saab selle andmestruktuuriga seostada.
2. Dünaamiline ehk käitumispõhine modelleerimine: objekte defineeritakse kui üksusi, millest igaühel on teatud spetsiifiline käitumisviis ja rida võimalikke olekuid.

- Dünaamilist modelleerimist teostatakse **sündmus- ja olekudiagrammide** abil.

- **Sündmus** on mingi teate (s. t. informatsiooni) edastamine objektile:

1. Kasutaja nõuab objektilt mingit teenust.
2. Objekt nõuab teiselt objektilt mingit teenust.
3. Objekt tagastab teate nõutud teenusega või selle kohta.

- **Objekti olek** on määratud tema sisemiste muutujate väärtusega. Objektide olekuid muudavad sündmused.

Joonis 4.5. Sündmusdiagramm.

Joonis 4.6. Olekudiagramm.