Luku 7

Mitä tästä kaikesta voisi oppia?

Ohjelmistotekniikan näkökulmasta "Sovellusohjelmointi MATLAB-ympäristössä" -kurssi on ollut yhden tietyn laskennallisen menetelmän, MLP-verkon, prototyypitystä. Jos kootaan yhteen niitä ajatuksia ja lähtökohtia, jotka ovat olleet vaikuttamassa kurssin sisältöön, nousee esiin seuraavia tekijöitä.

Siis, jos tehtävänäsi on suunnitella ja toteuttaa ohjelma X :

- Pyri ymmärtämään tehtävä kokonaisuudessaan eli mitä pitäisi tuottaa (output) ja mistä lähtien (input). Matemaattisesti ajateltuna lopullinen ohjelma on vain halutunlainen muunnos $y = X\,x$ input-output –parien välillä (luonnollisesti tämä pätee myös osatehtäviin).

- Jaa ongelma osaongelmiin eli erillisiin komponentteihin. Tähän voit käyttää esim. objektiorientoituneita suunnittelumenetelmiä tai (kuten laskennallisessa ohjelmoinnissa usein) omaa tervettä järkeä. Tärkeintä on, että saat muodostettua hyvin määriteltyjä komponentteja, joita voidaan kehittää ja testata riippumattomasti (itse tai jonkun alihankkijan toimesta).

- Käytä mahdollisimman paljon valmiita, olemassaolevia "rakennuspalasia": aliohjelmia, objekteja yms. MUTTA (tämä täytyy ottaa huomioon projektin aikataulua suunniteltaessa), ymmärrä valmiissa palasissa käytettyjen ratkaisujen perusteet ja testaa niitä erikseen varmistuaksesi, että ne sopivat (toimivat) juuri sinun tapauksessasi niinkuin haluat.

  Esimerkkinä mainittakoon kurssilla käsitelty MLP-verkon opetus optimointitehtävän ratkaisemisen kautta. Jos näin saatua menetelmää verrataan ns. backpropagation algoritmiin, joka on toteutettu esimerkiksi MATLABin *Neural Networks Toolbox*issa, on opetusalgoritmien nopeudessa ja tarkkuudessa eroa kuin yöllä ja päivällä - meidän tapamme eduksi!

- Jos teet osatehtävissä jotain laskentaa, pyri visualisoimaan algoritmeja mahdollisimman hyvin. Näin varmistut toiminnan oikeellisuudesta ja saat paremman kuvan käytettyjen menetelmien tavasta toimia. Visualisoinnin avulla voit myös keksiä joitain konkreettisia parannuksia käytettyihin menetelmiin. MUTTA, jos korvaat jonkun valmiiksi annetun algoritmin omalla viritykselläsi, sinun tulee olla ehdottoman varma siitä, että kehittämäsi menetelmä toimii kaikissa mahdollisissa tapauksissa!

- Yleensä kaikkein vaikein asia on osaongelmissa käytettyjen ratkaisujen testaaminen eli sellaisten testiesimerkkien keksiminen, jotka käyvät tehdyn ohjelmanpätkän läpi mahdollisimman kattavasti.

  Tällä kurssilla sovellettu tekniikka testauksen järjestämiseen esim. MEX-ohjelmien yhteydessä oli hyvin yksinkertainen: tehdään sama homma kahdella eri tavalla (MATLAB ja MEX) ja varmistetaan, että molemmilla saadaan sama tulos.

- Kun prototyyppi, joka yhdistää käytetyt osaratkaisut, on valmis, täytyy sekin testata mahdollisimman kattavasti. Aluksi luonnollisesti varmistutaan, että ohjelma ratkaisee halutunlaisen tehtävän. Tässä vaiheessa asiakas usein toteaa, että "Emmehän me tätä halunneet!", jolloin prosessi käynnistyy uudestaa alkuruudusta - tosin sillä erotuksessa, että nyt on olemassa paljon valmiita palikoita, joita voi hyödyntää myös uudessa prototyypissä. Lisäksi varsinaisen prototyypin testauksessa varmistetaan, että käytetyt ratkaisut eri komponenttien rajapinnoille toimivat kuten haluttiin. Tämä on yleensä tilanne, jossa projektissa aletaan voivotella sitä, ettei rajapintojen läpi liikkuvia objekteja ja/tai muuttujia määritelty aluksi yksikäsitteisesti vaan eri osissa on saatettu käyttää erilaisia määrityksiä samalle rakenteelle. Kun näitä ruvetaan yhdenmukaistamaan jälkikäteen, kuluu aikaa ja syntyy taatusti uusia virheitä!