Luku 6

SIMULINK

Tässä luvussa pyritään luomaan jonkinlainen yleiskuva MATLABin SIMULINKin tarjoamista mahdollisuuksista. Ohjelmistotekniikan kannalta tärkeää on liittää kyseisen työkalun toiminta yleisiin asioihin, erityisesti ns. ohjelmointikielten luokitteluun ja komponenttipohjaiseen ohjelmankehitykseen, joiden yleisillä perusteilla siten aloitetaankin ...

6.1 Ohjelmointikielten luokittelu

Ohjelmointikieliä (tai -ympäristöjä) on tapana jaotella eri sukupolviin (nGL = n'th Generation Language) sen perusteella, mikä on kielen etäisyys ohjelmaa suorittavasta laitteesta ja toisaalta kielen läheisyys ohjelmoinnin kohteeseen eli sovellukseen. Käytännössä eri sukupolviin kuuluvat/laskettavat kielet ovat syntyneet historiallisestikin peräkkäin, joten tällainen jaottelu kuvaa samalla myös ohjelmointikielten kehityksen historiaa.

1GL: Ensimmäisiä varsinaisia tietokoneita ohjelmoitiin sellaisenaan suoraan konekielellä, joten ohjelmointi oli työlästä ja erityisesti muutosten teko hankalaa, sillä esimerkiksi kaikki muistiosoitteet olivat viittauksia absoluuttisiin muistipaikkoihin. Koska uuden käskyn lisäys keskelle ohjelmaa aiheutti ohjelman loppuosan siirtymisen koneen muistissa, piti myös kaikki loppuosaan kohdistuvat viittaukset päivittää.

2GL: Seuraava edistysaskel oli symbolinen konekieli eli assembler, jonka etuina olivat selkeämmät käskyjen nimet ja symboliset osoitteet. Assembler-ohjelmia voitiin myös kääntää erikseen moduleiksi, jotka voitiin myöhemmin linkittää suoritusmuotoiseksi ohjelmaksi. Ohjelmien luettavuus pysyi edelleen huonona, sillä yleensä korkeintaan ohjelman kirjoittaja pystyi tulkitsemaan syntynyttä ohjelmaa. Ohjelmat olivat lisäksi hyvin koneriippuvia: tiettyä prosessoria ajatellen suunniteltu assemblykoodi ei yleensä suorittunut sellaisenaan muissa koneissa.

3GL: Kolmannen sukupolven kieliä sanotaan myös korkean tason kieliksi. Niille on ominaista mm. abstraktit kontrollirakenteet (esim. *if-then-else*) tieto-objektien tyypitys ja kyky rakentaa uusia abstrakteja toimintokokonaisuuksia aliohjelmien muodossa. Ohjelmat ovat melko helposti siirrettävissä koneesta toiseen ja niiden ymmärrettävyys ja luotettavuus ovat kohtuullisella tasolla. Useimmat nykyisin käytössä olevat kielet ovat kolmannen sukupolven kieliä.

4GL: Termi neljännen sukupolven kieli pomppaa useimmiten esiin erilaisten sovelluskehittimien mainoksissa. Sovelluskehitin on abstraktilta kannalta työkalu (ohjelma), jolla

pystytään helposti generoimaan tietyn sovellusalueen uusia ohjelmia. Moniin nykyisiin sovelluskehittimiin liittyy kiinteästi ohjelman graafisen käyttöliittymän integrointi ohjelmointikieleen. Monet kielistä ovat myös rakentuneet jonkin tietokannanhallintajärjestelmän ympärille, jolloin tietokannan kyselykieli on keskeinen osa kokonaisuutta. Esimerkiksi MATLABin voidaan ajatella olevan jonkinlainen neljännen sukupolven ohjelmankehitysympäristö. Neljännen sukupolven kielten suurimpana ongelmana on niiden epäyhtenäisyys, sillä useimmat näistä "kielistä" ovat vain jonkun tietyn valmistajan kaupallisia tuotteita, joiden integraatio muihin kieliin ja sovelluksiin on puutteellinen.

5GL: Viidennen sukupolven kieliksi kutsutaan lähinnä sellaisia asiantuntijajärjestelmiä, jotka pyrkivät mallintamaan ihmisen ajattelua ja päätöksentekoa, usein jonkinlaisen logiikan pohjalta.

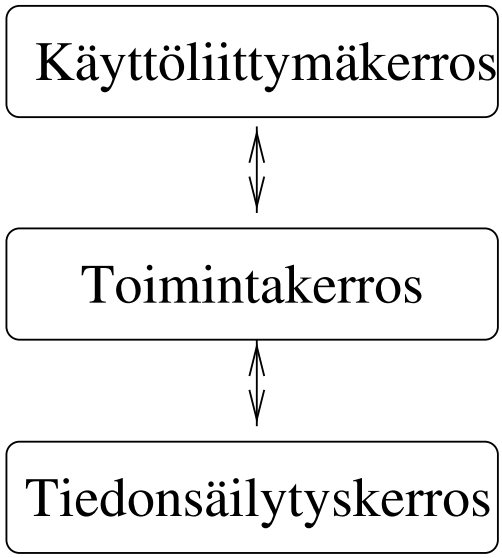
Koska suurin osa varsinaisista ohjelmointikielistä lukeutuu kolmanteen sukupolveen ja koska yo. muodossa määritellyt neljäs ja viides sukupolvi voidaan nähdä enemmän tai vähemmän jonkinlaisina "käyttöliittyminä" alhaisemman tason kielillä toteutettuihin järjestelmiin, ei yllä esitetty sukupolviajattelu ole välttämättä mielekäs tai ainakaan paras mahdollinen.

6.2 Komponenttipohjaisesta ohjelmankehityksestä

Neljännen sukupolven ohjelmankehityksen katsotaan usein perustuvan ns. uudelleenkäytettävien komponenttien luomiseen, hallintaan ja yhteiskäyttöön. Hajautetut komponenttiarkkitehtuurit tarjoavat mahdollisuuden kapseloida (piilottaa, *encapsulate*) sovellusten toiminnallisuutta monella tasolla. Yleisiä periaatteita komponenttipohjaisien järjestelmien suunnittelussa ovat mm.

- olemassaolevien komponenttimallien ja viestistandardien käyttö
- suunnittelun aikana keskittyminen vain komponenttien rajapintoihin
- rajapintojen palvelupohjaisuus (Asiakas/palvelin-malli, *Client/Server architecture*)
- arkkitehtuurin monitasoisuus, joka mahdollistaa komponenttien hajautuksen ja eri kerrosten selkeät vastuut

Komponenttien yhteydessä käytetty perusarkkitehtuuri on kolmikerroksinen:



1. **Käyttöliittymäkerros:** eri käyttäjille tarjotut käyttöliittymät
2. **Toimintakerros:** käyttöliittymien sisältämien toimintojen toteutus ja säilytettävän tiedon hallinta ja manipulointi
3. **Tiedonsäilytyskerros:** järjestelmään liittyvien pysyvien tietojen säilytyspaikka (yleensä tietokanta)

Tästä perusarkkitehtuurista muodostetaan luonnollisesti eri yhteyksissä sopivia modifikaatioita. Lisäksi valmiit hajautettujen komponenttien hallintajärjestelmät (esim. CORBA, (D)COM, EJB yms.) perustuvat yleensä jonkin tarkemmin spesifioituneen rakenteen pohjalle, joka yleensä pohjautuu tavalla tai toisella asiakas/palvelin-malliin.

Uudelleenkäytettävät komponentit ovat itseriittoisia (omatoimisia) ja selkeästi tunnistettavia. Ne kuvaavat tai suorittavat tiettyjä tehtäviä, joilla on selkeät rajapinnat (vrt. MEX), riittävä dokumentaatio sekä tarkoin määritelty tila uudelleenkäytön kannalta. Komponentin sisältämät rajapinnat voidaan jaotella esim. neljään eri tasoon:

käyttäjärajapinta: komponentin ja sen loppukäyttäjän välinen rajapinta

datarajapinta: komponentin ja sen käyttämän datan (muuttujien) välinen rajapinta

kompositiorajapinta: yksittäisen komponentin ja muiden komponenttien välinen rajapinta, joka sisältää

- komponenttiliittymän, jolla komponentti tarjoaa palveluitaan muille komponenteilla
- rajapinnan, jolla komponentti käyttää muiden komponenttien palveluja (proxy)

komponenttialusta: ohjelmisto, jota komponentti tarvitsee toimiakseen

Usein koetaan, että komponentin ja luokan välille on vaikea tehdä eroa (varsinkin, kun esim. CORBAsta puhutaan toisaalta hajautetun luokka-arkkitehtuurin toteuttamisvälineenä). Periaatteessa luokka edustaa loogista abstraktiota, kun taas komponentti fyysistä toimintakokonaisuutta, jossa esim. eri luokkien muodostama toiminnallinen yksikkö on pakattu yhteen. Luokilla on atribuutteja ja operaatioita, kun taas komponenteilla vain operaatioita, jot-

ka ovat rajapintojen kautta käytettävissä. Jotta ajatukset saataisiin varmasti sekaisin, todetaan lopuksi, että komponentti-idealogia sinänsä ei ota kantaa niiden varsinaiseen toteutukseen, joka yleensä perustuukin sopivan luokan generointiin!

6.3 SIMULINK

SIMULINK on MATLABin päälle rakennettu graafinen ympäristö, joka mahdollistaa valmiiden komponenttikirjastojen sisältämien toiminnallisten kokonaisuuksien yhdistämisen, hyvin pitkälle samaan tyyliin kuin GUIta rakennettaessa. Olemassaolevat komponenttikirjastot tukevat erityisesti ns. dynaamisten (ajasta riippuvien) systeemien mallintamista ja simulointia, mutta tällä kurssilla ei totisesti ole tarkoituksena enää sukeltaa tällaisten systeemien (matemaattisten perusteiden) syövereihin. Kuten jo GUIdenkin yhteydessä, päätarkoituksena on kokeilla itse SIMULINKin käyttöä ja muodostaa tämän avulla jonkinlainen kokonaiskuva siitä tavasta, jolla MATLAB komponenttipohjaista ohjelmankehitystä tukee.

SIMULINKiin liittyvät seuraavat peruskäsitteet:

komponentti (block): "rakennuspalikka" eli lohko, joka koostuu

i) toiminnasta

ii) sisään- ja ulostuloportteihin liitetyistä muuttujista

malli (model): komponentteja yhdistelemällä muodostettu toiminnallinen kokonaisuus

kanava (channel): kahden komponentin välillä tapahtuvan tiedonsiirron väylä, jossa muuttujat välitetään mallin eri komponenttien välillä

signaali (signal): kanavia pitkin vaeltava muuttuja (skalaari tai vektori), jonka arvoja mallin (systeemin) eri komponentit mahdollisesti muuttavat

SIMULINK-mallin rakenne koostuu yleensä seuraavista perusosista:

i) aikaväli $[t_0, T]$, jolla mallin toimintaa tarkastellaan (yleensä $t_0 = 0$)

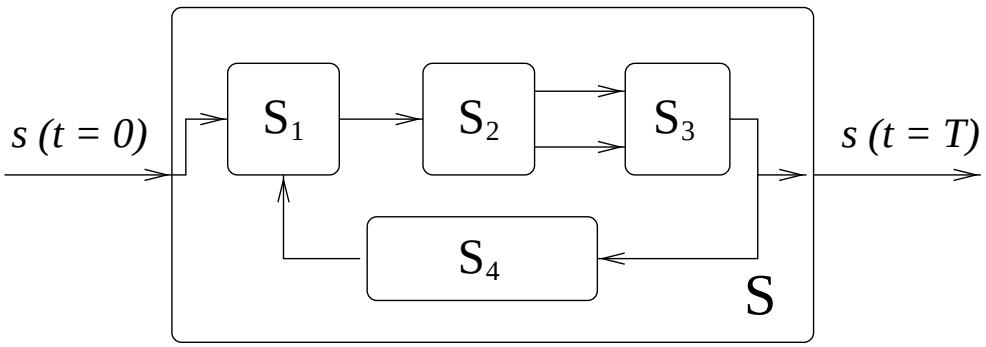
ii) input-muuttujat sekä koko mallille että sen osille

iii) muuttujille suoritettavat muunnokset eli mallin osien varsinaiset toiminnot

iv) output-muuttujat

v) kaikkien tai vain osamuuttujien (yleensä output) havainnollistus suoritettujen muunnosten yhteydessä: aikasarjat

SIMULINK-mallin rakennetta on havainnollistettu seuraavassa kuvassa:



SIMULINKin peruskäytön idealogia on täsmälleen sama kuin GUIdenkin rakentamisen yhteydessä - *What you see is what you get*! Siispä mallin rakentaminen perustuu ensin sen toiminnallisen rakenteen "piirtämiseen" graafisten työkalujen avulla (aivan samannäköiseksi kuin edellisen sivun alalaidan systeemin kuva) ja sen jälkeen mallin komponenttien sisältämien parametrien arvojen asettamiseen. Mallin rakentaminen pitää siten sisällään seuraavia toimenpiteitä:

- valitaan tarvittavat komponentit komponenttikirjastoista ja siirretään ne luotavaan malliin: hiiren avulla
- yhdistetään halutut komponentit toisiinsa mallin toiminnallisen kokonaisuuden kuvaamiseksi: hiiren avulla
- määritellään komponenttien sisältämien sekä simuloinnissa tarvittavien vakioiden arvot: hiiren ja *Dialog Box*in avulla
- talletetaan malli ja simuloidaan sen pohjalta muodostuvan järjestelmän toimintaa halutulla aikavälillä $[t_0, T]$: mallin sisältämän ikkunan *menu*n avulla

HUOM: Tämän kurssin yhteydessä tapahtuvassa SIMULINKin pintaraapaisussa käsitellään järjestelmän käyttöä vain sen päälle rakennetun GUIn avulla, joka mahdollistaa yo. toimintastrategian seuraamisen hiiren ja valikkotoimintojen avulla. Luonnollisesti niin mallien rakentaminen kuin niiden simulointi on mahdollista myös suoraan komentotilasta annettavien käskyjen avulla MATLABin "normaalin" peruskäytön mukaisesti.

Lisäpiirteitä, joiden tarkemmat kuvaukset löytyvät tarvittaessa SIMULINK-oppaasta:

- Malleja voidaan jakaa osakokonaisuuksiin eli alisysteemeihin (System/Subsystem), joiden avulla loogisia toimintakokonaisuuksia voidaan kapseloida. Tällöin on luonnollisesti eri alisysteemien kesken tapahtuvaan muuttujavaihtoon määriteltävä sopiva rajapinta, mutta SIMULINKin graafiset työkalut tarjoavat tässä suhteessa näppäriä apukeinoja etenkin, jossa alisysteemi irroitetaan muusta mallista sen luomisen yhteydessä.
- Alisysteemejä voidaan edelleen "maskata" "black-box"-tyyppisiksi kokonaisuuksiksi eli uusiksi komponenteiksi, joiden käyttö niiden luomisen jälkeen on siten samanlaista kuin SIMULINKin sisältämien valmiiden komponenttien. SIMULINKin yhteydessä uusien komponenttien luominen tapahtuu seuraavien osien määrittelemisen kautta:

  ICON: komponenttia graafisesti edustava ikoni,

  ALUSTUS: komponentin nimi sekä siihen liittyvät muuttujat ja niiden arvojen asettaminen *Dialog Box*in avulla,

  DOKUMENTOINTI: sanallinen kuvaus komponentin toiminnasta *Dialog Box*in avulla.

- Jonkin parametrin arvon mukaan *ehdollisesti suoritettavat alisysteemit*, jotka voivat olla tyyppiä: päälle/pois (enabled/disabled), suoritus muuttuvan signaalin perusteella: nouseva/laskeva (triggered), kahden em. yhdistelmä.
- Ulkopuolisten toiminnallisten kokonaisuuksien (M-funktiot, MEX-funktiot jne.) liittäminen SIMULINKiin: S-funktio standardi.