Dünaamika modelleerimine

Kõik süsteemid omavad staatilist struktuuri ja dünaamilist käitumist. Klassidiagramme kasutatakse süsteemi staatilise struktuuri (klassid, objektid ja nende seosed) väljendamiseks. Oleku-, jada-, koostöö ning tegevusdiagrammid sobivad süsteemi käitumise (dünaamika) väljendamiseks: kuidas objektid suhtlevad dünaamiliselt, erinevatel aegadel süsteemi toimimise käigus.

Klassidiagramm näitab, mida süsteem sisaldab ja kuidas need asjad on omavahel seotud, kuid ei selgita, kuidas need asjad toimivad koos oma ülesannete / süsteemi funktsionaalsuse täitmiseks.

Süsteemi objektid suhtlevad üksteisega, saates üksteisele sõnumeid. Näiteks tellija objekt Joe saadab müügimehe objektile Bill sõnumi sooviga, et viimane teeks midagi. Sõnum on tüüpiliselt operatsiooni väljakutse, mida üks objekt sooritab teisel objektil. Objektide suhtlemist ning sellise suhtlemise tagajärgi nimetataksegi süsteemi dünaamikaks; see tähendab, kuidas objektid suhtlemise kaudu koos töötavad ning muudavad oma seisundit süsteemi eluea jooksul. Suhtlemist (kommunikatsiooni) hulga objektide vahel mingi funktsiooni loomiseks/täitmiseks nimetatakse *interaktsiooniks*. Interaktsiooni saab kirjeldada kolme liiki diagrammidega: jada-, koostöö- või tegevusdiagrammiga.

Dünaamikadiagrammid on järgmised:

- *Olekudiagrammid (state diagrams):* kirjeldavad, milliseid seisundeid võib objekt omada oma elutsükli vältel, tema käitumist neis seisundites, sündmusi mis põhjustavad seisundi muutumist.
- *Jadadiagrammid (sequence diagrams):* kirjeldavad, kuidas objektid üksteisega suhtlevad. Põhiaspektiks, millele keskendutakse, on aeg. Jadadiagramm näitab, kuidas sõnumijada saadetakse ning võetakse vastu hulga objekti vahel mingi funktsiooni täitmiseks.
- *Koostöödiagrammid (collaboration diagrams):* kirjeldavad samuti objektide suhtlemist, kuid põhiaspektiks, millele koostöödiagramm keskendub, on ruum. See tähendab, et objektidevahelised (ruumilised) seosed (lingid) on tähelepanu keskmes ning seepärast otseselt näidatud diagrammil.
- *Tegevusdiagrammid (activity diagrams):* on samuti suhtlemise (interaktsiooni) väljendamise viis, kuid põhiaspektiks, mille tegevusdiagramm keskendub, on töö. Kui objektid üksteisega

suhtlevad, siis nad teevad ka tööd, mis väljendub tegevustes. Tegevusdiagramm kirjeldab tegevusi ja nende järjestust.

Kuna jada-, koostöö- ning tegevusdiagrammid väljendavad kõik suhtlemist, tuleb sageli teha valik, millist diagrammi kasutada suhtlemise (interaktsiooni) kirjeldamiseks. Otsus sõltub sellest, mis aspekti peame kõige tähtsamaks.

Lisaks dünaamilisele struktuurile ja dünaamilisele käitumisele on süsteemil ka funktsionaalne vaade, mis iseloomustab süsteemi poolt pakutavaid funktsionaalsusi (teenuseid). Use case-id annavad süsteemi funktsionaalse kirjelduse: kuidas tegutsejad saavad kasutada süsteemi. Use-case-id modelleeritakse juba varases (vajaduste analüüsi) staadiumis ning nad näitavad üksnes seda, kuidas tegutsejad võivad süsteemi kasutada, mitte kuidas süsteem on ehitatud. Klassid ja interaktsioonid realiseerivad use case-e süsteemis, seega eksisteerib seos ka süsteemi funktsionaalse ning dünaamika vaate vahel. Interaktsioonid modelleeritakse jada-, koostöö-, ja/või tegevusdiagrammides. Use case-ide realiseerimisel kasutatavaid klasse modelleeritakse klassi- ja olekudiagrammides (olekudiagramm on seotud klassi, allsüsteemi või süsteemiga).

Interaktsioonid objektide vahel (Sõnumid)

Objektorienteeritud programmeerimises teostatakse interaktsioon kahe objekti vahel sõnumina, mida üks objekt teisele saadab. Sõnum realiseeritakse kõige sagedamini lihtsalt operatsiooni väljakutsega: üks objekt kutsub välja teise objekti operatsiooni. Kui operatsioon saab täidetud, tagastatakse juhtimine väljakutsujale koos tagastatava väärtusega (return value). Sõnumiks võib olla ka tegelik sõnum, mille saatmiseks kasutatakse kommunikatsioonimehhanismi (üle võrgu või arvuti siseselt), kuid valdavaks on selline variant vaid reaalajasüsteemides. Sõnumeid kasutatakse kõikides dünaamikadiagrammides (jada-, koostöö-, oleku-, tegevusdiagrammid) objektidevahelise kommunikatsiooni vahenditena. Sõnum joonistatakse noolega joonena sõnumi saatja ja vastuvõtja vahel, kusjuures noole tüüp näitab sõnumi tüüpi:

- *Lihtne:* esitab lihtsalt juhtimisvoogu. Näitab, kuidas juhtimine edastatakse ühelt objektilt teisele ilma kommunikatsiooni detaile kirjeldamata. Kasutatakse siis, kui kommunikatsiooni detailid pole teada või neid peetakse ebaolulisteks antud diagrammis. Samuti

kasutatakse sünkroonse sõnumi tagastamise näitamiseks, mis joonistatakse sõnumit käsitlevast objektist tagasi väljakutsujale näitamaks, et juhtimine antakse tagasi.

- *Sünkroonne:* hierarhiline juhtimisvoog, tüüpiliselt realiseeritud operatsiooni väljakutsena. Sõnumit käsitlev operatsioon viiakse lõpule (kaasa arvatud kõik teised sõnumid, mis saadetakse käsitlemise osana), enne kui väljakutsuja saab jätkata täitmist. Tagastamist saab näidata lihtsa sõnumina või vaikimisi, kui sõnum on käsitletud.
- *Asünkroonne:* asünkroonne juhtimisvoog, kus pole selget tagastamist väljakutsujale ja kus saatja jätkab täitmist sõnumi saatmise järel ilma selle käsitlemist ära ootamata. Tüüpiliselt kasutatakse reaalajasüsteemides, kus objektid tegutsevad konkurentselt.

Lihtsat ja sünkroonset sõnumit saab kombineerida üheks sõnumijooneks koos sünkroonse sõnumi noolega ühes otsas ning lihtsa tagastamisnoolega teises otsas. See tähendab, et tagastamine toimub peaaegu koheselt pärast operatsiooni väljakutset.

Olekudiagramm

Olekudiagrammid hõlmavad objektide, allsüsteemide ja süsteemide elutsükleid. Näitavad, milliseid seisundeid objekt võib omada ning kuidas sündmused (vastuvõetud sõnumid, kindel aeg, vead, tõeseks saanud tingimus) mõjutavad neid seisundeid ajas. Olekudiagramm tuleks siduda kõikide klassidega, mis omavad selgelt identifitseeritavaid seisundeid ja keerukat käitumist; diagramm kirjeldab käitumist ning kuidas see erineb sõltuvalt jooksvast seisundist. Kirjeldab, millised sündmused muudavad klassi objektide seisundit.

Seisundid ja Üleminekud

Kõik objektid omavad seisundit; seisund on objekti poolt teostatud eelnevate tegevuste tulemus, mis on määratud tema atribuutide ja seoste (lingid teiste objektidega) väärtustega. Klassil võib olla kindel atribuut, mis näitab seisundit, või seisund on määratud objekti “tavaliste” atribuutide väärtustega (arvutatav nende alusel). Näited:

- Arve (objekt) on makstud (seisund).
- Auto (objekt) seisab (seisund).
- Masin (objekt) töötab (seisund).

- Jim (objekt) esineb müügimehe rollis (seisund).
- Kati (objekt) on abielus (seisund).

Objekt muudab seisundit, kui midagi juhtub (sündmus): keegi maksab arve, alustab auto juhtimist, abiellub. Dünaamikal on kaks mõõdet: suhtlemine (interaktsioon) ning sisemise seisundi muudatused. Interaktsioonid kirjeldavad objekti välist käitumist, kuidas ta suhtleb teiste objektidega (kas sõnumiedastuse või üksteisega ühendumise/lahtiühendumise teel). Sisemise seisundi muudatused kirjeldavad, kuidas objektid muudavad oma sisemiste atribuutide (ka seoste) väärtusi. Olekudiagrammid näitavad objekti reaktsiooni sündmustele ning kuidas ta muudab sisemist seisundit: arve loomisel saab tema seisundiks “maksmata”; kui keegi ta ära maksab, muudab arve seisundit (“maksmata” -> “makstud”).

Olekudiagramm võib omada lähtepunkti (algolek) ning mitut lõpp-punkti (lõppolekud). Seisundit näidatakse ümara kastiga, algolekut väikese täidetud ringiga, lõppolekut “härjasilma” sümboliga. Seisundi üleminekut joonega, mille ühes otsas nool ülemineku suuna näitamiseks. Seisundi üleminekud võivad olla märgistatud üleminekut põhjustava sündmusega. Sündmuse toimumisel viiakse seisundi üleminek tegelikult läbi (üleminek “põleb” ehk “käivitatakse”).

Seisund võib sisaldada kolme liiki sektsioone (osasid). Esimene sektsioon näitab seisundi nime, näiteks makstud, maksmata. Teine sektsioon on mittekohustuslik seisundimuutujate osa, kus võivad olla loetletud ning väärtustatud atribuudid (muutujad). Kolmas sektsioon on mittekohustuslik tegevuse sektsioon, kus võivad olla näidatud sündmused ja toimingud. Kasutatakse kolme standardsündmust: *entry, exit, do.* Entry sündmust kasutatakse seisundisse sisenemisel täidetavate toimingute kirjeldamiseks, näiteks atribuudi väärtustamine või sõnumi saatmine. Exit sündmust kasutatakse seisundist väljumisel täidetavate toimingute kirjeldamisel. Do sündmust saab kasutada seisundi kestel (kuni objekt on antud seisundis) täidetavate tegevuste kirjeldamiseks; näiteks sõnumi saatmine, ootamine, arvutamine. Tegevuse sektsiooni formaalne süntaks:

event-name argument-list ‘ / ‘ action-expression

Sündmuse nimi võib olla suvaline sündmus, kaasa arvatud standardsündmused. Tegevusavaldis näitab, milliseid tegevusi tuleb täita (operatsiooni väljakutsed, atribuutide väärtustamine,..). Sündmusel võivad olla argumendid. Standardsündmustel ei saa argumente olla.

Seisundi üleminekuga on tavaliselt seotud sündmus, kuid mitte tingimata. Do-tegevus seisundi sees võib olla pidev protsess (näiteks ootamine, operatsiooni juhtimine,..), mida teostatakse senikaua, kui objekt on antud seisundis. Do-tegevuse võib katkestada väline sündmus, mis põhjustab antud seisundi ülemineku järgmisesse seisundisse.

Kui seisundi üleminek ei oma sündmust, siis lähteseisund muutub automaatselt hetkel, kui temas kirjeldatud sisemised tegevused on täidetud (juhul kui seal on kirjeldatud entry, exit, do või kasutaja-defineeritud tegevusi).

Formaalne süntaks seisundi ülemineku spetsifitseerimiseks:

event-signature ‘’[‘ guard-condition ‘]’ ‘/’ action-expression ‘^’ send-clause

kus sündmuse signatuuri süntaks on järgmine:

event-name ‘(‘ parameter ‘,’,…’)’

ning send-klausli süntaks:

destination-expression ‘.’destination-event-name ‘(‘ argument ‘,’ …’)’

kus sihi avaldis väärtustab objekti või objektihulga.

Sündmuse signatuur

Sündmuse signatuur koosneb sündmusnimest ja parameetritest, mis annavad seisundi üleminekut käivitavale sündmusele lisaandmeid. Parameetrid on komaga eraldatud loetelu parameetritest, mille süntaks on:

> *Parameter-name ‘:’ type-expression, Parameter-name ‘:’ type-expression …*

Tüübiavaldise võib ära jätta

Seisundi ülemineku sündmuse signatuuri näited:

draw (f : Figure, c : Color)

redraw ()
redraw
print (invoice)

Valvurtingimus (Guard-Condition)

Valvurtingimus on Bool'I avaldisseisundi üleminekul. Kui see tingimus on kombineeritud sündmuse signatuuriga, siis sündmus peab toimuma *ja* tingimus peab olema tõene, et üleminek toimuks. Kui üleminekuga on seotud ainult valvurtingimus, siis üleminek toimub, kui tingimus saab tõeseks. Näited:

[t = 15sec]
[number of invoices > n]
[withdrawal (amount) [balance >= amount]

Tegevusavaldis (Action-Expression)

Tegevusavaldis on protseduurne avaldis, mida täidetakse ülemineku toimumusel. Ta võib olla kirjutadud seisundidiagrammi omava objekti operatsioone ning atribuute või sündmussignatuuri parameetreid kasutades. On võimalik kirjutada rohkem kui ühe tegevusavaldise seisundi üleminekule, kuid need peavad olema eraldatud kaldkriipsuga. Tegevusavaldised täidetakse üksteise järel spetsifitseerimise järjekorras vasakult paremale. Hierarhilised või rekursiivsed tegevusavaldised pole lubatud. On võimalik seisundi üleminek, mis sisaldab ainult tegevusavaldist. Näited seisundi ülemineku tegevusavaldistest:

increase () / n := n + 1 / m := m + 1
add (n) / sum := sum + n
/flash

Send-Klausel

Send-klausel on tegevuse (action) erijuhtum, süntaks mis näitab sõnumi saatmist seisundi ülemineku vältel. Süntaks koosneb sihiavaldisest ja sündmusnimest. Sihiavaldis näitab objekti või objektihulka, kellele sõnum saadetakse. Sündmusnimi on sellise

sündmuse nimi, mis omab tähendust sihtobjektile (või objektihulgale). Sihtobjektiks võib olla ka antud objekt ise.

[timer = Time-out] / go down (first floor)

võib teisendada send-klausliks:

[timer = Time-out] ˇ self.go down (first floor)

Teised näited seisundi üleminekutest send-klausliga:

out_of_paper()ˇindicator.light()
left_mouse_btn_down(location) /
color:=pick_color(location) ˇpen.set(color)

Olekudiagrammid peavad olema kergesti mõistetavad (nagu igasugused mudelid), kuid mõnikord on raske väljendada keerulist sisemist dünaamikat (objekti sisemisi seisundeid koos kõikide üleminekutega) ning samal ajal luua kergesti mõistetav mudel. Igas konkreetses olukorras peab modelleerija otsustama, kas modelleerida kogu sisemine dünaamika nii, nagu ta teatavaks saab, või lihtsustada arusaadavuse huvides (lihtsustus võib olla ajutine).