Use-Case modelleerimine

Use case modelleerimistehnikat kasutatakse selleks, et näidata, mida uus süsteem peaks tegema või mida olemasolev süsteem juba teeb. Use-case mudel ehitatakse iteratiivses protsessis, mille käigus toimub diskussioon süsteemi arendajate ja tellijate (ja/või lõppkasutajate) vahel, mille tulemusena saadakse vajaduste spetsifikatsioon, millega kõik osapooled nõus on.
Use-Case modelleerimine loodi Ivar Jacobsoni (I. Jacobson, 1994) poolt ning seda kasutatakse paljudes OO meetodites (OOSE , Booch'I meetod, Rational Unified Process).

Use-case mudeli põhikomponentideks on *use case'*id, *actor'*id, ning modelleeritav *süsteem*. Süsteemi piirid defineeritakse süsteemi poolt käsitletava funktsionaalsusega. Funktsionaalsus esitatakse hulga *use case'*idega, millest igaüks spetsifitseerib tervikliku funktsionaalsuse. Kui funktsionaalsus on terviklik (lõpetatud), peab *use case* käsitlema tervikfunktsiooni, alates tema initsialiseerimisest (käivitamisest) välissubjekti (*actor'*i) poolt kuni nõutud funktsionaalsuse täitmiseni. Use case peab alati andma mingi väärtuse tegutseja (actor) jaoks, kus väärtuseks on miski, mida tegutseja soovib süsteemilt. Tegutseja (actor) on igasugune välisüksus, kes/mis soovib suhelda antud süsteemiga. See võib olla inimene või süsteem või riistvaraseade, mis peab suhtlema antud süsteemiga.

Use-case modelleerimises vaadatakse süsteemi "musta kastina", mis pakub tegutsejatele (actor) teenuseid (use case). Kuidas süsteem seda teeb, kuidas use case'id on realiseeritud ja kuidas nad sisemiselt töötavad, pole esialgu oluline ning seda ei pruugi teada veel isegi projekteerijad.

Use Case'ide põhieesmärgid:

- Otsustada ja kirjeldada süsteemi funktsionaalsed vajadused koostöös Tellijate (ja/või lõppkasutajate) ja süsteemi projekteerijate/ehitajate vahel.
- Anda selge ja kooskõlaline kirjeldus sellest, mida süsteem tegema peab. Seda kirjeldust kasutavad läbi kogu arendusprotsessi süsteemi kõik osapooled, et üle anda soovitud tulemusi/funktsionaalsusi.
- Anda alus süsteemi testimisele, et saaks kindlaks teha, kas üleantud süsteem täidab esialgselt soovitud funktsionaalsust
- Võimaldada jälgida funktsionaalsete vajaduste teisenemist süsteemi klassideks ja operatsioonideks. Lihtsustada süsteemi muutmist ja

laiendamist use-case mudeli muutmise ja muudetud use-case-ide jälgimise kaudu süsteemi disaini ja realisatsiooni tasemel.

Use-case mudeli koostamine:

- Süsteemi defineerimine
- Tegutsejate ja use case-ide leidmine
- Use case-ide kirjeldamine
- Seoste defineerimine use case-ide vahel
- Mudeli õigsuse kontroll

On hea, kui süsteemi kasutaja (esindab tegutsejat use-case diagrammis) osaleb aktiivselt use-case modelleerimisel, sest niiviisi saab kohandada mudelit täpselt kasutaja soovidega. Use case-id kirjeldatakse tellija/kasutaja keeles ning mõistetes. Niiviisi kindlustatakse kommunikatsioon süsteemi tellijate/kasutajate ning arendajate, integreerijate, testijate ja muude osapoolte (müük, toetus, dokumenteerimine) vahel.

Use-case mudel esindab süsteemi use-case vaadet. See on keskne vaade, mis mõjutab süsteemi kõiki teisi vaateid. Use case-id mõjutavad nii süsteemi loogilist kui ka füüsilist arhitektuuri, kuna use case-idega spetsifitseeritud funktsionaalsused realiseeritakse nendes arhitektuurides. Disainitav lahendus peab rahuldama sisulisi vajadusi.

Use-case modelleerimine on kasulik mitte ainult uute süsteemide loomisel, vaid ka olemasolevate süsteemide uute põlvkondade (versioonide) arendamisel.

Use-Case diagramm

Use-case mudel kirjeldatakse UML-is *use-case diagrammidena*, kusjuures use-case mudel võib olla jagatud paljudeks use-case diagrammideks. Use-case diagramm sisaldab mudelielemente süsteemi, tegutsejate ning use case-ide jaoks ning näitab erinevaid seoseid (üldistus, assotsiatsioon, sõltuvus) nende elementide vahel.

Use-case’i sisu kirjeldus antakse tavaliselt (esialgu) vaba teksti vormis. (UML-is käsitletakse kirjeldust kui use-case elemendi dokumentatsiooniomadust). Alternatiiviks on use case-i formaalne kirjeldamine dünaamikadiagrammiga, kuid lõppkasutaja jaoks on tekstiline kirjeldus tavaliselt vajalik.

Süsteem

Use-case modelleerimine defineerib arendatava süsteemi piirid. Piiride täpne paikapanemine (mis võtta sisse, mis jätta välja) on eduka süsteemiarenduse esimene tingimus.

Tähtis otsustus on, kui suur peaks olema süsteemi esimene väljalase. See peaks olema väheste, kuid kõige tähtsamate funktsionaalsustega ning hea arhitektuuriga, kuhu saaks hiljem valutult soovitud funktsionaalsusi juurde lisada. Nii saadakse ka esimene versioon ruttu valmis.

Süsteemi jaoks tasuks kohe koostada ka põhimõistete sõnastiku, mida kasutatakse use case-ide kirjeldamisel ning edasise domeenianalüüsi käigus. See võib olla lihtne/üldine kontseptuaalne mudel või tavaline tekstikataloog mõistete ja nende selgitustega.

Use-case diagrammis esitatakse süsteem kastiga, mille sisse joonistatakse antud süsteemi use-case'ide sümbolid.

Tegutsejad

Tegutseja (actor) on keegi või miski, mis suhtleb süsteemiga / kasutab süsteemi. Kasutaja saadab ja/või saab sõnumeid süsteemile/süsteemilt. Tegutsejad viivad läbi use case-e. Tegutseja võib olla inimene või teine süsteem.

Tegutseja on tüüp (klass), mitte eksemplar. Ta esitab rolli, mitte üksikkasutajat süsteemis. Ühe isiku kohta võib olla mitu tegutsejat, sõltuvalt tema rollist süsteemis. Tegutseja nimi peab väljendama tema rolli.

Use case algatatakse alati tegutseja poolt, kes saadab talle sõnumi, mida mõnikord nimetatakse stiimuliks (stimulus). Kui use case on läbi viidud, peab ta saatma sõnumi ühele või mitmele tegutsejale. Need sõnumid võivad minna ka teistele tegutsejatele lisaks sellele, kes algatas use case-i.

Tegutsejaid saab liigitada. Primaarne tegutseja (primary actor), kes kasutab süsteemi põhifunktsioone (näiteks kindlustusosakond kindlustusfirmas). Sekundaarne tegutseja on see, kes kasutab süsteemi sekundaarseid funktsioone, nagu süsteemi hooldus, andmebaaside,

kommunikatsiooni, taastamise jt. administreerimisülesannete juhtimine. Ainult päringusüsteemi kasutaja (olgugi et võib-olla tippjuht) on süsteemi seisukohalt sekundaarne tegutseja.

Aktiivne tegutseja on see, kes käivitab use case-i. Passiivne tegutseja üksnes osaleb ühes või enamas use case-is.

Tegutsejate ülesleidmine

Tegutsejate ülesleidmisega pannakse paika üksused/osapooled, mis/kes on huvitatud süsteemiga suhtlemisest ja selle kasutamisest. Siis on võimalik asetuda tegutseja positsioonile ning üritada identifitseerida tegutseja nõuded süsteemile ning milliseid use case-e ta vajab. Tegutsejate ülesleidmiseks võib vastata järgmistele küsimustele:

- Kes hakkab kasutama süsteemi põhifunktsionaalsusi (primaarsed tegutsejad)?
- Kes vajab toetust süsteemilt oma igapäevaste ülesannete täitmisel?
- Keda on vaja süsteemi hooldamiseks, administreerimiseks ja töös hoidmiseks (sekundaarsed tegutsejad)?
- Milliseid riistvaraseadmeid peab süsteem käsitlema?
- Milliste teiste süsteemidega peab süsteem suhtlema? Nii süsteemid, mis algatavad kontakti mei süsteemiga, kui ka süsteemid, millega meie süsteem ise kontakteerub.
- Kes või mis omab huvi tulemuste (väärtuse) vastu, mida süsteem toodab?

Use-case modelleerimist kasutatakse ärimodelleerimiseks, seepärast on tegutsejad tavaliselt selle äri (välimised või sisemised) kliendid, mitte tingimata vahetud arvutikasutajad.

Tegutsejate leidmiseks võib küsitleda olemasoleva (arvuti- või käsi-) süsteemi kasutajaid, uurida milliseid erinevaid rolle nad täidavad igapäevatöös süsteemiga.

Tegutseja eksemplaride (näiteks inimeste) nimetamise kaudu saab kontrollida, kas selline tegutseja tegelikkuses eksisteerib. Tegutseja peab omama assotsiatsiooni ühe või enama use case-ga. Kui ta use-case-i ei käivita, peab ta sellega mingis punktis suhtlema.

Tegutsejad UML-is

Tegutsejad UML-is on klassid stereotüübiga “actor” ning klassinimi on ka tegutsejanimeks (väljendades tegutseja rolli). Tegutsejaklass võib omada atribuute, käitumist ning dokumentatsiooniomadust.

Use-case diagrammil on vastava stereotüübi ikooniks tavaliselt kriipsujuku.

Seosed tegutsejate vahel

Kuna tegutsejad on klassid, võivad nad omada ka samu seoseid. Use-case diagrammidel kasutatakse neist üksnes üldistusseoseid paljude tegutsejate ühise käitumise kirjeldamiseks.

Üldistusseos

Kui tegutsejad, oma rolli osana, täidavad ka üldisemat rolli, kirjeldatakse seda üldistusseosega. Üldisema rolli käitumine kirjeldatakse tegutseja ülemklassis. Spetsialiseeritud tegutseja pärib pärib ülemklassi käitumise ning laiendab seda kuidagi. Tegutsejate üldistamiseks kasutatakse use-case diagrammil sarnast esitust kui suvaliste klasside vahel UML-is.

Use Case-d

Use case esindab terviklikku funktsionaalsust tegutseja jaoks. UML-is defineeritakse use case kui *süsteemi poolt läbiviidavate tegevuste järjestuste hulk, mis annab jälgitava väärtustulemuse konkreetsele tegutsejale*. Tegevused võivad sisaldada kommunikatsiooni paljude tegutsejatega (kasutajad ning teised süsteemid), samuti teostada arvutusi ja töid süsteemi sees. Use case-i omadused:

- *Use case käivitatakse alati tegutseja poolt:* Use case viiakse alati läbi tegutseja poolelt. Tegutseja peab otseselt või kaudselt käskima süsteemil use case-i täita. Mõnikord ei pea tegutseja otseslt use case-i käivitama (trigerid).
- *Use case annab tegutsejale väärtuse*: use case peab kasutajale üle andma “käegakatsutava” väärtuse. Väärtus ei pea olema väljapaistev, võib olla lihtsalt tajutav.

- *Use case peab olema täielik (complete)*: Tavaline viga: jagatakse use case väiksemateks use case-ideks mis realiseerivad üksteist nagu üksteist väljakutsuvad funktsioonid programmeerimiskeeles. Use case pole täielik, kui ta pole tootnud lõppväärtust kasutaja jaoks, isegi kui ta sisaldab kasutajadialooge.

Use case-id ühendatakse tegutsejatega assotsiatsioonide kaudu, mida mõnikord nimetatakse *kommunikatsiooni assotsiatsioonideks*, mis näitavad, milliste tegutsejatega use case suhtleb. Assotsiatsioon on tavaliselt üks-üks seos ilma suunata. See tähendab, et üks tegutseja eksemplar suhtleb ühe use-case eksemplariga ning kommunikatsioon toimub mõlemas suunas. Use case nimetatakse tema eesmärgi järgi, nagu *Kindlustuslepingu sõlmimine*. Tavaliselt on nimeks fraas, mitte üksik sõna.

Use case on klass, mitte eksemplar. Ta kirjeldab tervikfunktsionaalsust, kaasa arvatud võimalikud alternatiivid, vead, erijuhtumid use case-i täitmisel. Use case-i eksemplari nimetatakse *stsenaariumiks*, mis esindab süsteemi konkreetset täitmist.

Use Case-ide ülesleidmine

Use case-ide ülesleidmise protsess algab tegutsejate ülesleidmisega. Iga ülesleitud tegutseja kohta võiks küsida:

- Milliseid funktsioone tegutseja vajab süsteemilt? Mida tegutseja peab tegema?
- Kas tegutseja peab lugema, looma, hävitama, või salvestama mingit liiki informatsiooni süsteemis?
- Kas tegutseja peab saama teateid sündmustest või ise registreerima sündmusi süsteemis? Mida need sündmused esindavad funktsionaalsuse mõttes?
- Kas tegutseja igapäevatööd saab lihtsustada või muuta effektiivsemaks süsteemi uute funktsioonide kaudu (mis siiani oli automatiseerimata)?

Küsimused, mis ei puuduta ühte konkreetset tegutsejat:

- Milliseid sisendeid/väljundeid süsteem vajab? Kust sisendid tulevad ja kuhu väljundid lähevad?
- Millised on süsteemi praeguse realisatsiooni põhiprobleemid

Viimaste küsimustega identifitseeritakse esmalt use case-id ning seejärel nendega seotud tegutsejad. Use case peab olema seotud vähemalt ühe tegutsejaga.

Use Case-id UML-is

Tähistatakse ellipsiga, mis sisaldab use case-i nime. Nimi võib olla ka use case-i sümboli all. Tavaliselt paigutatakse use case süsteemi piiride sisse ning seostatakse tegutseja(te)ga assotsiatsiooni või kommunikatsiooniassotsiatsioni abil.

Seosed Use Case-ide vahel

Use case-ide vahel on kolme liiki seoseid: extends, uses, ning grupeerimine. Grupeerimine on seotud use case-ide paigutamine pakettidesse.

- *Extends seos:* on üldistusseos, kus üks use case laiendab teist, lisades tegevusi üldisemale use case-ile. Laiendav use case võib sisaldada käitumist laiendatavast use case-ist, sõltuvalt laiendamise tingimustest.
- *Uses seos:* on üldistusseos, kus üks use case kasutab teist use case-i, näidates viimast spetsialiseeritud use case-I osana, üldise use case-i käitumine võetakse samuti sisse.
- *Grupeerimine:* kui palju use case-e käsitlevad sarnast funktsionaalsust või on kuidagi üksteisega seotud, pakitakse nad UML-i paketti. Pakett grupeerib seotud mudelielemente. Paketil pole semantilist tähendust.

Extends seos

Kui üks use case laiendab teist use case-i, siis esimene võib sisaldada laiendatava use case-i mingit käitumist. Ta ei pea sisaldama kogu käitumist, vaid võib valida, milliseid osi üldisema use case-i käitumisest ta soovib taaskasutada. Laiendatav use case peab olema täielik. Kuna use case-e tavaliselt kirjeldatakse tekstiga, siis võib olla keeruline defineerida, milliseid üldisema use case-I osi taaskasutatakse, millised defineeritakse üle ning millised lisatakse laiendatud use case-is.

Laiendatud use case võib käsitleda üldisema use case-i erijuhtumeid, mida on raske kirjeldada üldisemas use case-is eneses, või mis tehakse süsteemi arendusprotsessi kulgedes. Laiendusseos näidatakse üldistussümboliga koos stereotüübiga “extends”.

Uses seos

Kui palju use case-e omavad ühist käitumist, võib see käitumine olla modelleeritud eraldi use case-is, mida teised use case-id kasutavad. Kui üks use case kasutab teist,, peab ta kasutama teise kogu käitumist (kuigi kasutatava use case-i tegevused ei pea olema samas järjestuses; nad võivad olla läbisegi kasutava use case-i tegevustega). Kui use case, mida kasutatakse, iseennast kunagi ei kasuta, nimetatakse teda *abstraktseks use case-iks.*

Kasutusseost näidatakse üldistussümboliga (kolmnurk kasutatava use case-i poole) koos stereotüübiga “uses”.

Use Case-ide kirjeldamine

Use case kirjeldatakse tavaliselt vaba tekstiga. See on lihtne ja kooskõlaline kirjeldus sellest, kuidas tegursejad ja use case-id (süsteem) suhtlevad. See keskendub süsteemi välisele käitumisele ning ignoreerib aspekti, kuidas asi tegelikult toimub süsteemi sees. Kirjelduse keel ja terminoloogia peavad olema samad, mis süsteemi tellijal/kasutajal.

Tekstiline kirjeldus peaks sisaldama:

- *Use case-i eesmärk:* use case-id on tulemusorienteeritud ning iga use case-i eesmärk/tulemus peab olema selge.
- *Kuidas use case käivitatakse:* Milline tegutseja algatab use case-i täitmise ning millistes olukordades?
- *Sõnumivoog tegutsejate ning use case-i vahel:* Milliseid sõnumeid või sündmusi use case ja actor vahetavad üksteise teavitamiseks, info uuendamiseks ja otsimiseks, üksteise otsuste toetamiseks? Mida peab kirjeldama põhiline sõnumivoog süsteemi ja tegutsejate vahel, ning milliseid olemeid süsteemis kasutatakse ning uuendatakse?
- *Alternatiivsed vood use case-is:* Use case võib omada alternatiivseid täitmisi sõltuvalt tingimistest. Need tuleb mainida, kuid mitte kirjeldada liiga detailselt, nii et põhiline tegevusvoog üldjuhu jaoks “ära peidetaks”. Spetsiifiline veakäsitlus kirjeldatakse stsenaariumiga.
- *Kuidas use case lõpetatakse koos väärtusega tegutsejale:* Kirjelda, millal use case loetakse lõpetatuks ning mis liiki väärtus antakse üle tegutsejale.

Kirjeldus tehakse välise tegutseja vaates, kuidas tema jaoks asjad toimuvad, mitte süsteemi siseselt. Tekst peab olema selge ja kooskõlaline, nii et tellija saaks aru ning suudaks kontrollida selle õigsust/sobivust tema vajadustega. Hoidu keerukatest lausetest, mida saab mitutpidi mõista

Use case-e saab saab kirjeldada ka tegevusdiagrammiga (activity diagram)

Use-case (üldise, täieliku) kirjelduse lisandina kasutatakse paljusid konkreetseid illustreerivaid stsenaariume use case-i täitmise kohta konkreetsetel tingimustel (eksemplari tase).

Seoste kirjeldamiseks pole enne piisavalt teadmisi, kui kõik use case-id on kirjeldatud. Seoste kirjeldamisl vastata järgmistele küsimustele:

- Kas kõik use-case-iga seotud tegutsejad omavad sellega kommunikatsiooniseoseid?
- Kas esineb sarnasusi erinavate tegutsejate hulgas, kes esindavad ühist üldisemat rolli ja mida võiks kirjeldada baasklassi tegutsejana (*base class actor)*?
- Kas esineb sarnasusi erinevate use-case-ide hulgas, mis esindavad ühist käitumist, mida saaks kirjeldada “uses” seosega mingisse use-case-i?
- Kas eksisteerib use-case-i erijuhtumeid, mida võiks kirjeldada “extends” seosega?
- Kas on mingeid tegutsejaid või use-case-e ilma kommunikatsiooniseosteta? Kui jah, milleks sellised tegutsejad?
- Kas on teada funktsionaalseid vajadusi, mida ükski use-case ei käsitle? Kui jah, loo uus use case selle vajaduse jaoks.

Use Case-ide testimine

Use case-e kasutatakse testimisel. Tehakse kahte erinevat tüüpi teste: *verifitseerimine* ja *valideerimine.* Verifitseerimine kinnitab, et süsteemi arendatakse korrektselt või vastavalt tehtud spetsifikatsioonidele. Valideerimine kindlustab, et arendatav süsteem on selline, mida tellija või lõppkasutaja tõesti vajab.

Valideerimist tehakse arendusprotsessi alguses. Niipea, kui on olemas lõpetatud use-case mudel (võib-olla juba use-case mudeli väljatöötamise ajal), esitatakse see tellijatele ja lõppkasutajatele arutamiseks. Need peavad kontrollima, et see mudel (eriti nende jaoks oluliste

funktsionaalsuste täitmise viis) rahuldab korrektselt ning täielikult nende ootusi süsteemi suhtes. Selleks peab arendaja kindlustama, et tellijad saaksid tõesti mudelist ja tema tähendusest aru, et vältida millegi vastuvõetamatu pealesurumist tellijale. Selle protsessi käigus tekib uusi ideid, mis lisatakse use-case mudelisse enne lõplikku valideerimist. Valideerida saab muidugi ka süsteemi testimise ajal, kuid kui siis süsteem ei rahulda korrektselt kasutaja vajadusi, tuleb kogu projekt otsast peale ümber teha.

Süsteemi verifitseerimine testib, kas süsteem töötab vastavalt spetsifikatsioonidele. Seda ei saa teha enne, kui on olemas mingi töötav osa süsteemist. Kui selline valmib, saab testida, kas süsteem käitub nii, nagu kasutajad spetsifitseerisid, kas use case mudelis kirjeldatud case-id töötavad ning käituvad vastavalt kirjeldusele.

Use Case-ide “läbijalutamine”

Läbijalutamise tehnikat kasutatakse nii use case-ide defineerimisel kui ka testimisel. Toimub rollimäng, kus modelleerimisgrupi erinevad liikmed mängivad tegutsejaid ning süsteemi konkreetses use-case-is. Inimene, kes mängib tegutsejat, ütleb, mida tegutseja süsteemiga teeb. Selle tulemuseks on vastava tegevusega käivitatava use-case-i täitmine süsteemi poolt; isik, kes mängib süsteemi rolli, ütleb, mida ta teeb use-case-i täitmise ajal. Arendajad, kes parasjagu pole rollimänguga hõivatud, teevad märkmeid ning püüavad leida puudusi mängijate poolt esitatavates use-case-ides. Tüüpiliselt leitakse, et mingid alternatiivid on täiesti käsitlemata ning mingid tegevused kirjeldatud ebapiisava detailsusega.

Mida paremat ettekujutust süsteemi kasutusest rollimängijad omavad, seda paremaks testimine osutub. Erinevate rollide esitajate vahetamine annab erinevaid interpretatsioone ja vaateid, millest on abi use-case-ide täiustamisel. Kui kõikide tegutsejate rollid kõikides use case-ides on läbi mängitud, on use case mudeli terviktestimine lõppenud.

Use Case-ide realiseerimine

Use case-id on süsteemi funktsionaalsuse realisatsioonist sõltumatud kirjeldused. See tähendab, et use case-id realiseeritakse süsteemisiseselt, kus vastutus use case-ide kirjeldustes kirjeldatud tegevuste täitmise eest omistatakse koostoimivatele objektidele, mis antud funktsionaalsust realiseerivad.

UML-I printsiibid use case-ide realiseerimisel on järgmised:

- *Use case realiseeritakse kollaboratsioonis:* kollaboratsioon näitab use case-i sisemist, realisatsioonist sõltumatut lahendust klasside / objektide ja nendevaheliste seoste (kollaboratsiooni kontekst) ning soovitud funktsionaalsuse saavutamiseks vajaliku suhtlemise (kollaboratsiooni interaktsioon) kaudu. Kollaboratsiooni sümboliks on ellips, mis sisaldab kollaboratsiooni nime.
- *Kollaboratsioon esitatakse UML-is paljude diagrammiden, mis näitavad nii konteksti kui interaktsiooni kollaboratsioonis osalejate vahel.* Kollaboratsioonis osalevad klassid ning kollaboratsiooni eksemplaris osalevad objektid. Diagrammideks on kollaboratsiooni, jada ja tegevusdiagramm. Diagrammitüüp, mida kasutatakse kollaboratsioonist tervikpildi andmiseks, sõltub konkreetsest juhtumist. Mõnikord piisab ühest kollaboratsioonidiagrammist; teistel juhtudel on vajalik kombinatsioon erinevatest diagrammidest.
- *Stsenaarium on use case-i või kollaboratsiooni eksemplar:* Stsenaarium on konkreetne täitmistee (sündmusvoog), mis esitab use case-i konkreetset eksemplari (süsteemi ühte kasutust). Kui stsenaariumi vaadatakse use case-ina, kirjeldatakse üksnes tema välist käitumist tegutsejate suhtes. Kui stsenaariumi vaadatakse kollaboratsiooni eksemplarina, kirjeldatakse sisemist realisatsiooni: hõlmatud klasse, nende operatsioone ning kommunikatsiooni.

Use case-i realiseerimine tähendab teisendada tema erinevad sammud ja tegevused (mis on kirjeldatud tekstina või tegevusdiagrammina) klassideka, nende klasside operatsioonideks ning nendevahelisteks seosteks. Vastutus iga sammu eest use case-is omistatakse use case-i realiseerivas kollaboratsioonis osalevatele klassidele. Leitakse lahendus use case-is kirjeldatud välise käitumise saavutamiseks ning kirjeldatakse kollaboratsioonina süsteemi sees.

Use case-i iga samm teisendatakse reeglina mitmeks operatsiooniks; ebatõenäone on üks-üks seos use case-i tegevuse ja operatsiooni vahel osalevate klasside objektide interaktsioonis. Klass võib osaleda paljudes use case-ides. Klassi täielikuks vastutuseks on kõikide rollide integratsioon, mida ta mängib erinevates use case-ides.

Seost use case-i ja teda realiseeriva kollaboratsiooni vahel näidatakse kas peenendusseosega (punktiirnool) või vastavas vahendis näiteks nähtamatu hüperlingiga. Viimasel juhul saab lülituda use case diagrammilt viimast realiseerivasse kollaboratsioonidiagrammi. Samuti saab lülituda use case-ilt tema konkreetsele stsenaariumile (eksemplarile), mis on esitatud ühega

dünaamikadiagrammidest: tegevus-, jada- või kollaboratsioonidiagrammiga.

Klassidele vastutuse määramine on keeruline iteratiivne tegevus, mis nõuab suurt kogemust. Jacobson kasutab kollaboratsiooni kirjeldamisel kolme klassi stereotüüpi: piirobjektid (boundary objects), juhtobjektid (control objects), olemobjektid (entity objects).

- *Piirobjektid (liidesobjektid):* paiknevad süsteemi piiri läheduses, kuid ikkagi veel sees. Nad suhtlevad süsteemiväliste tegutsejatega ning vahendavad nende sõnumeid teist tüüpi objektidega süsteemi sees.
- *Juhtobjektid:* kontrollivad (juhivad) interaktsioone mingis objektigrupis. Selline objekt võib olla "kontroller" kogu use case-i jaoks või realiseerida ühist suhtlusjada erinevates use case-ides. Sageli eksisteerib selline objekt ainult use case-i täitmise ajal.
- *Olemobjektid:* esindavad süsteemi poolt käsitletava probleemvaldkonna põhiolemeid (domain entity). Need on tüüpiliselt passiivsed, ei algata iseseisvalt interaktsioone. Infosüsteemides on nad tavaliselt salvestatud püsivalt andmebaasi.. Tüüpiliselt osalevad paljudes use case-ides.

Need stereotüübid omavad spetsiaalseid ikoone klassi- ja kollaboratsioonidiagrammis. Pärast erinevat tüüpi objektide leidmist ning kollaboratsiooni spetsifitseerimist püütakse leida sarnasusi kollaboratsioonide vahel, et osasid klasse saaks kasutada paljudes use case-ides. Use case-ide selline rakendamine on süsteemi analüüsi ja disaini aluseks: use case-driven development process.

Millal omistada vastutused use case-idest klassidele. Erinevad meetodid käituvad erinevalt. Mõned meetodid soovitavad, et domeenianalüüs peab olema enne tehtud, kirjeldatud domeeni kõik klassid ja nende seosed. Siis võtab arendaja järjest iga üksiku use case-i ning omistab vastutuse analüüsimudeli klassidele, mõnikord muutes mudelit või lisades uusi klasse. Teised meetodid soovitavad, et use case-id oleksid klasside ülesleidmise aluseks, nii et vastutuste omistamise käigus domeenianalüüsi mudelit tegelikult koostatakse.

On oluline rõhutada, et arendustöö on iteratiivne. Kui vastutust omistatakse klassidele, leitakse vigu ja puudujääke klassidiagrammis ning viimast modifitseeritakse. Kindlasti avastatakse uusi klasse use case-ide toetamiseks. Mõnikord on vaja muuta ka use case diagramme, kui sügavam arusaamine süsteemist viib arendaja äratundmisele, et use case on valesti kirjeldatud. Use case-id aitavad keskenduda süsteemi

funktsionaalsusele, et see oleks korrektselt kirjeldatud ning realiseeritud süsteemis. Suur probleem nendes objektorienteeritud meetodites, mis ei kasuta use case-e, on keskendumine klasside ja objektide staatilistele struktuuridele (kontseptuaalne modelleerimine) ning arendatava süsteemi funktsionaalsete ning dünaamika aspektide ignoreerimine.

Kokkuvõte

Use-case modelleerimine on süsteemi funktsionaalsete vajaduste kirjeldamise tehnika. Kirjelduse elementideks on välised tegutsejad, use case-id, ning modelleeritav süsteem. Tegutsejad esitavad välisüksuse (kasutaja, riistvara, teine süsteem) rolli suhtlemises süsteemiga. Tegutsejad käivitavad use case-e ning suhtlevad nendega., kusjuures use case on süsteemis täidetavate tegevusjadade hulk. Use case peab üle andma tegutsejale “käegakatsutava” väärtuse. Use case kirjeldatakse tavaliselt tekstidokumentatsiooniga. Tegutsejad ja use case-id on klassid. Tegutseja on ühendatud ühe või enama use case-iga assotsiatsioonide kaudu ja nii tegutsejad kui use case-id võivad omada üldistusseoseid, mis kirjeldavad ühist käitumist ülemklassides, mis päritakse ühe või enama spetsialiseeritud alamklassi poolt. Use case mudel kirjeldatakse ühes või enamas UML use case diagrammis.

Use case-id realiseeritakse kollaboratsioonides, kus kollaboratsioon on konteksti kirjeldus, mis näitab klasse/objekte ja nende seoseid, ning interaktsioon, mis näitab, kuidas klassid/objektid suhtlevad konkreetse funktsionaalsuse teostamiseks. Kollaboratsioon kirjeldatakse tegevusdiagrammide, koostöödiagrammide ja jadadiagrammidega. Kui use case-i realiseeritakse, omistatakse vastutus use case-I iga tegevussammu eest kollaboratsioonis osalevatele klassidele, kirjeldades operatsioone nendel klassidel, mille kaudu need suhtlevad. Stsenaarium on use case-i või kollaboratsiooni eksemplar, mis näitab konkreetset täitmisteed. Seega on stsenaarium use case-i või kollaboratsiooni illustratsioon või näide. Vaadatuna use case-i eksemplarina, kirjeldatakse ainult interaktsiooni use case-I ning välistegutsejate vahel, kuid vaadatuna kollaboratsiooni eksemplarina kirjeldatakse stsenaariumis interaktsiooni süsteemisiseste klasside/objektide vahel, mis realiseerivad seda süsteemi.