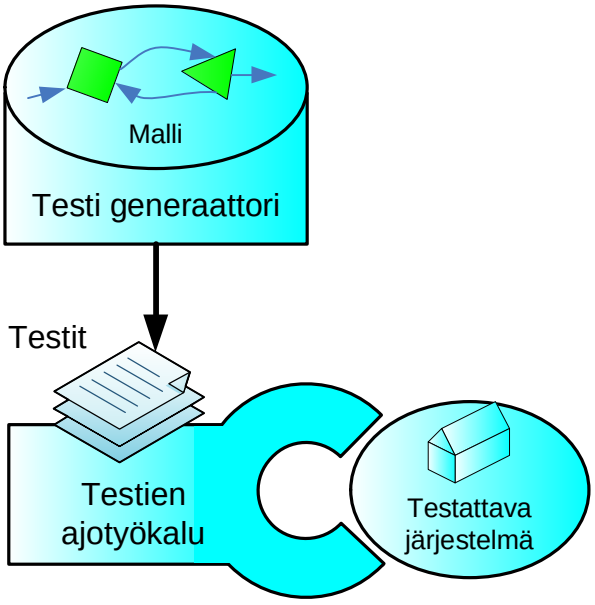
Mallipohjainen testaus ennen, nyt ja tulevaisuudessa

Työtuntien kalleus, tietokoneiden tehojen nousu ja järjestelmien monimutkaistuminen houkuttelee käyttämään tietokonetta myös testauksen apuna niin paljon kuin mahdollista. Mallipohjainen testaus tarjoaa tähän yhden apuvälineen. Mallipohjaista testausta on tutkittu jo pitkään ja monipuolisesti ja joitain mallipohjaisen testauksen tapoja käytetään jo menestyksekkäästi mutta paljon on vielä saavutettavissa.

Mallipohjaisessa testauksessa tuotetaan automaattisesti useita testejä ohjelman käyttäytymistä kuvaavan mallin perusteella. Tämä säästää aikaa ja kustannuksia, koska jokaista testitapausta ei tarvitse kirjoittaa käsin. Testit voidaan aina generoida uudestaan kun mallia on muutettu ja näin tarvitsee ylläpitää vain yhtä mallia useiden testitapausten sijaan. Näin se auttaa sekä testitapausten luonnissa että niiden ylläpidossa. Mallipohjainen testaus voidaan jakaa kolmeen päävaiheeseen, joita ovat mallinnus, testien generointi ja testien ajo. Mallipohjaiset testaustyökalut sisältävät vähintään testien generoinnin mutta joskus myös muita vaiheita.

Mallipohjaisessa testauksessa on kaksi pääsuuntaa: offline ja online tyyppinen testaus. Offline lähestymistavassa testit ensin generoidaan ja tallennetaan erillisiksi testiskripteiksi. Tämän jälkeen ne ajetaan erikseen erillisen testien suoritukseen tehdyn työkalun avulla. Online-lähestymistavassa taas nämä kaksi vaihetta on yhdistetty, eli testien suunnittelutyökalu etenee mallissa luoden samalla testiaskeleita, niin että edellinen askel suoritetaan järjestelmää vasten ennen kuin työkalu suunnittelee seuraavan askeleen. Eli online-lähestymistapa on kuin turisti joka katsoo karttaa joka risteyksessä. Online-lähestymistapa antaa myös mahdollisuuden testata epädeterministisiä malleja kun päätös suoritettavasta polusta voidaan tehdä dynaamisesti mallin tarkastelun ja testin suorituksen aikana. Online mahdollistaa myös pitkien testiajojen suorituksen, eli työkalu voidaan käytännössä laittaa ajamaan yhtä pitkää testijaksoa kunnes se erikseen keskeytetään. Koska mallipohjaisessa testauksessa mallinnus keskitetään halutulle abstraktiotasolle piilottamalla testien kannalta epäolennaiset asiat, mikä helpottaa mallintamista myös sovellusalueen asiantuntijan näkökulmasta.



Kuva 1. Offline mallipohjainen testaus

Sovellusalueita

Mallipohjaisen testauksen voidaan sanoa olevan parhaimmillaan testattaessa järjestelmiä korkeamman tason näkökulmasta, eli esimerkiksi järjestelmätestauksessa. Yleisesti mallinnuksessa haetaan korkeampaa abstraktiotasoa, jotta mallinnus olisi tehokasta ja voitaisiin hyödyntää sovellusalueasiantuntijoita. Siten yksityiskohtien testaamiseen perinteiset yksikkö- ja integrointitestauksen muodot täydentävät mallipohjaisen testauksen lähestymistapaa hyvin. Mallipohjaista testausta voidaan myös soveltaa yksityiskohtaisemmallekin tasolle, mutta sen kustannustehokkuus on parhaimmillaan korkeammalla tasolla.

On myös havaittu, että monesti jo pelkkä mallinnus auttaa havaitsemaan monia virheitä koska yleensä järjestelmän määritykset ovat epäformaalissa muodossa josta ei helposti nähdä moniselitteisyyksiä tai ristiriitaisuuksia. Kun määritykset kuvataan malliin tulevat myös moniselitteisyydet ja ristiriitaisuudet esiin ja ne joudutaan ratkaisemaan.

Mallinnuksessa on tarkoitus tuottaa testien generointityökalulle tarvittava tieto järjestelmän toiminnallisuudesta. Koska malli on testigeneraattorin ainoa tieto järjestelmästä testikehyksen lisäksi, se käytännössä määrittää miltä osin ja miten kattavasti testigeneraattori voi luoda testejä järjestelmälle. Järjestelmää ei tarvitse mallintaa kokonaan vaan voidaan testata myös osa kerrallaan, keskittymällä mallinnuksessa kiinnostaviin toiminnallisuuden osiin. Täten on myös mahdollista luoda useita eri testimalleja eri näkökulmista.

Testien automaattinen tuottaminen ja suoritus

Testien generoinnissa siihen käytetty työkalu pyrkii tuottamaan annetun mallin ja parametrien perusteella niin hyvät testitapaukset kuin mahdollista. Testikattavuuden parametreina käytetään yleensä yleisiä mallin osia, kuten tila, tilasiirtymä tai muita kattavuuksia tai erikseen käyttäjän merkitsemiä kohtia mallissa jotka testigeneraattorin pitäisi saada katettua. Testityökalu yleensä tarjoaa myös monia vaihtoehtoja testien tulostukseen helppokäyttöiseen muotoon.

Testien suoritus on offline-lähestymistavassa samanlaista kuin automatisoitu testien ajo yleensä ja siihen on monia hyviä ratkaisuja olemassa. Online mallipohjainen testaus vaatii testien suunnitteluohjelman yhdistämisen kiinni testattavaan järjestelmään testien ajoa ja tulosten tarkastamista varten. Testien ajo ja raportointi vaiheet on tärkeää automatisoida kokonaan jotta mallipohjaisesta testauksesta saataisiin mahdollisimman paljon hyötyä.

Vaikka mallipohjaista testausta on tutkittu pitkään, niin siltikään monimutkaisemmissa tapauksissa algoritmit eivät keksi kaikkia haluttuja testitapauksia ainakaan järkevässä ajassa. Mallinnus on myös yleensä aikaa vievää, vaikeaa ja virhealtista työtä. Testien suoritusautomatiikka ja työkaluketjujen rakentaminen on kypsempää tekniikkaa ja siihen on monia pitkälle kehittyneitä ratkaisuja olemassa.

Pilvipalveluista tehoa testien automaattiseen suunnitteluun

Testien suunnittelualgoritmien tutkimus perustuu pitkälti matemaattiseen analyysiin ja tällä puolella on voitu hyödyntää pitkää historiaa algoritmien tutkimuksessa, kuten tilakoneiden analysointia ja symbolista suoritusta. Ongelmia aiheuttavat edelleen mallien monimutkaiset toistorakenteet, monisäikeistykset, monimutkaisemmat kaaviota malleissa ja testitapausten määrän ja laadun optimointi. Tietokoneiden tehojen kasvu ja pilvipalvelujen käyttö laskennassa antavat mahdollisuuksia käyttää yhä enemmän laskentaa vaativia algoritmeja ja vievät mallipohjaista testausta eteenpäin.

Sovellusaluekohtainen mallinnus tehostaa mallinnusta

Sovellusaluekohtaista mallinnusta (domain-specific modelling) käytetään helpottamaan mallintajan työtä. Tällöin voidaan tehdä sovellusalueelle oma kieli joka on optimoitavissa mahdollisimman tehokkaaksi sovellusalueelle ja mallinnusnäkökulmalle. Tällaisella mallinnuskielellä tehty malli voidaan muuntaa sopivaksi valitulle testien generointityökalulle käyttämällä mallitransformaatiota ja siihen sopivia työkaluja. Kun testimalli luodaan erillään mallipohjaisesta testaustyökalusta ja sovellusalueen omilla kielillä on myös mahdollista vaihtaa testigeneraattoria tai käyttää useita testigeneraattoreita yhtäaikaisesti. Tällöin mallipohjaisen testauksen käyttöönotto voidaan esimerkiksi aloittaa avoimen lähdekoodin työkaluilla ja myöhemmin siirtyä kaupallisiin työkaluihin tarpeiden kasvaessa, tai lisätä itse ohjelmoiden spesifisiä testien generointialgoritmeja. Yleisesti ottaen mallipohjaisen testauksen työkalut ovat hyvin erilaisia joten testien tuottaminen usealla työkalulla antaa monipuolisempia testejä.

Havainnoinnista automatisoitua apua mallinnukseen

Yksi suurimpia ja haasteellisimpia vaiheita mallipohjaisen testauksen käyttöönotossa on sopivan ja tehokkaan testimallin tekeminen. Tähän liittyen havainnointiin perustuva mallintaminen (observation based modelling) pyrkii luomaan automaattisella havainnoinnilla edistyneen alkumallin. Havainnot perustuvat esimerkiksi olemassa oleviin testeihin (yksikkö-, integraatio-, tai muita testejä), tai käyttäjäsessioiden monitorointeihin. Käyttäessään tällaista havainnoista automaattisesti luotua mallia testaaja vertaa sitä järjestelmän vaatimuksiin käyttäen apunaan mallipohjaista testaustyökalua. Tätä mallia joutuu käytännössä aina muokkaamaan, koska se perustuu rajalliseen joukkoon havaintoja (esim. olemassa olevat yksikkötestit). Näin mallia tulee käytännössä ”yleistää” kuvaamaan järjestelmän kokonaistoiminnallisuutta havaitun lisäksi. Kun mallipohjainen testaustyökalu sitten käy mallia läpi ja suorittaa siitä testejä vasten testattavaa järjestelmää, erot toteutuksen ja määritelmän välillä tulevat esille. Näin mallipohjainen testaustyökalu edesauttaa myös uusien virheiden löytämistä, koska se pakottaa tekemään tästä edistyneestä alkumallista vertailun määrityksiin sekä käy mallia läpi uusien polkujen ja syötteiden avulla luodessaan uusia testikombinaatioita testien generointialgoritmiensa avulla. Tällä tavalla havaintoihin perustuva testaus tuottaa tietoa järjestelmän toiminnasta, päivittämään ja löytämään virheet dokumentoinnista, sekä löytämään virheitä itse toteutuksessa.

Mallien osittaminen parantaa niiden hallintaa ja tehokkuutta

Perinteisesti testimalli mallipohjaisessa testauksessa sisältää yhdessä mallissa sekä syötteiden tuottamiseen tarvittavan tiedon, sekä saatujen vasteiden tarkastamiseen tarvittavan tiedon. Näistä mallipohjainen testaustyökalu generoi sekä syötteet että spesifiset tarkistukset kullekin saadulle vastineelle. Tämän perinteisemmän lähestymistavan lisäksi testimalli voidaan kuvata erillisinä osina ja myös jaotella testityökalu vastaaviin osiin. Jaottelu voidaan tehdä ajattelemalla eri osia ohjelmiston toiminnallisuuden pysyvien ominaisuuksien (invariant properties) summana.

Esimerkiksi voidaan määritellä yksi malli joka määrittää mahdolliset syötteet eri käyttöliittymän näkymille, sekä tilasiirtymät näiden välillä. Yksi työkalu voi silloin luoda syötteitä tästä mallista pyrkien ajamaan mallia läpi mahdollisimman kattavasti. Toinen malli voi kuvata eri näkymän komponenttien sisältöjen ja muiden ominaisuuksien (esim. näkyvyyden) vaatimuksia. Toinen työkalu voi sitten tarkistaa jatkuvasti että kaikki saadut vastineet vastaavat odotuksia ja järjestelmä on kokonaisuudessaan jatkuvasti eheä, eli se vastaa sille määriteltyä ”invariantteja”. Etuna voidaan nähdä esim. mahdollisuus fokusoitua spesifisiin mallinnuksen komponentteihin sekä laajennettu kattavuus mahdollisten tarkistusten ja luotujen syötteiden osalta mukaan lukien järjestelmän eri osien välillä ja suhteissa toisiinsa. Toisaalta tämän lähestymistavan voidaan nähdä tarjoavan oman lisänsä perinteisiin mallipohjaisiin testausmenetelmiin ja siksi myös näiden yhdistäminen on potentiaalinen lähestymistapa.

Mallipohjainen testauksen käyttöönotto

Mallipohjainen testaus soveltuu hyvin yleiseen järjestelmän toiminnan testaamiseen ja automaattisesti tuotettuja testejä voidaan käyttää useanlaisessa testauksessa. Jotta mallipohjaisen testauksen voisi ottaa käyttöön, täytyy testien automaattinen suoritusympäristö olla olemassa. Mallipohjainen testaus vaatii normaalin testiautomaation tavoin alkupanostuksia ja ylläpitoa toimiakseen. Parhaillaan se kuitenkin auttaa löytämään virheitä joita ei välttämättä käsin testattaessa löydetä ja lisää testauksen kattavuutta tehtyyn työhön nähden.

Käyttökelpoisia mallipohjaisen testauksen työkaluja alkaa olla jo markkinoilla niin kaupallisina tuotteina kuin vapaan lähdekoodin projekteinakin. Työkalut ovat yleensä kehittyneet selvästi jonkin tietyn sovellusalueen vaatimuksista ja siksi käyttöönotossa työkalua kannattaa kokeilla huolellisesti. Käytännössä monissa tapauksissa jo yksinkertaisemmat algoritmit antavat huomattavan hyödyn. On myös tapauksia joissa sovellusalueella on erityispiirteitä joita ei yleisillä algoritmeilla saa tehokkaasti testattua. Näissä tapauksissa voi olla hyödyllistä rakentaa oma työkalu sen sijaan että käyttäisi olemassa olevia.

Mallipohjainen testaus tulevaisuudessa

Mallipohjaista testausta on olemassa hyvin monenlaista ja sitä on tutkittu parikymmentä vuotta. Näiden vuosien aikana monenlainen teknologinen kehitys ja osaamistason nousu on tuonut mallipohjaisen testauksen hyödyt yleisemmin saataville.
Tällä hetkellä mallipohjainen testaus on oikeassa käytössä monessa yrityksessä, mutta se ei ole vielä pääsuuntaus testausautomaation käytännössä. Työkaluja on jo olemassa joitakin niin avoimen lähdekoodin kuin kaupallisiakin työkaluja.
Näyttää siltä että seuraavan kymmenen vuoden aikana tullaan löytämään yhä uusia paikkoja mallipohjaiselle testaukselle ja sen työkalutuki kasvaa madaltaen käyttöönottokynnystä. Kymmenen vuoden kuluttua mallipohjainen testaus on yksi lisätyökalu testaajan pakissa, joka voidaan ottaa käyttöön testitapausten suorituksen automatisoinnin jälkeen jos tarvitaan vielä lisää tehoa testaukseen.

Yhteenveto

Mallipohjainen testaus on saanut tukevan jalansijan testausautomaatiossa. Kokoajan kehittyvät mallinnus menetelmät ja algoritmit yhdessä testien ajoympäristöjen kanssa auttavat soveltamaan mallipohjaisen testauksen periaatteita yhä uusiin sovellusalueisiin. Mallipohjainen ei kuitenkaan ole hopealuoti ja vaatii sekä työtä, osaamista ja ylläpitoa ollakseen hyödyllinen.

Kirjoittajat:

Olli-Pekka Puolitaival
Tutkija, olli-pekka.puolitaival@vtt.fi
Olli-Pekka toimii tutkijana VTT:llä testausautomaation parissa erityisalueenaan sovellusaluekohtaisen mallipohjainen testaus.

Teemu Kanstrén
Tutkija , teemu.kanstren@vtt.fi
Teemu tutkii VTT:llä ohjelmistojen toiminnallisuuden analysointia. Hän väitteli havaintoihin perustuvasta mallinnuksesta 2010.

Liite 1 Lisämateriaalia

Linkkivinkit:

- VTT:n testausautomaatiosivut: http://www.testautomation.fi/
- Kirja "Practical Model-Based Testing: A Tools Approach": http://www.cs.waikato.ac.nz/~marku/mbt/
- Wikipedia-artikkeli: http://en.wikipedia.org/wiki/Model-based_testing

Miten ja mitä kaikkea voi mallintaa?

- Näkökulmia
  - Ulkoinen - tilat, protokollat, toiminnot, palvelut...
  - Sisäinen - tilat, säikeet, tietorakenteet...
  - Systeemien välinen – tilat, viestintä, tietovirrat...
- Kieliä
  - Kaaviot, graafit, visuaaliset mallinnuskielet
  - Tekstipohjaiset kielet, testaustapausten kuvauskielet, rakenteiset dokumentit
  - Hybridiratkaisut
  - Yleinen kielet: UML, SDL...
  - Sovellusaluekohtaiset kielet
  - Todennäköisyyspainotteiset tekniikat
  - Tietovirtapainotteiset tekniikat
- Tasoja
  - Binäärilogiikka
  - Functiot ja toiminnot
  - Järjestelmän sisäiset käyttötapaukset
  - Käyttöliittymä
  - Kommunikointirajapinta
  - Järjestelmien kokonaisuus
  - Järjestelmien toimintaympäristö
- Rakenne
  - Yksi komponentti kerralla
  - Koko ohjelma
  - Osa ohjelmaa
  - Yhdistelmämalli
  - Alimallit
  - Rinnakkaiset mallit