Title	Mallintarkastus kriittisten automaatiojärjestelmien suunnittelun tueksi
Author(s)	Pakonen, Antti; Lahtinen, Jussi; Björkman, Kim; Valkonen, Janne
Citation	Automaatio XIX Seminaari, Hotelli Crowne Plaza, Helsinki, 15.-16.3.2011
Date	2011
Rights	Copyright © (2011) Suomen Automaatioseura. Esitys on ilmestynyt Suomen Automaatioseuran julkaisusarjassa nro 41.

Mallintarkastus kriittisten automaatiojärjestelmien suunnittelun tueksi

Antti Pakonen, Jussi Lahtinen, Kim Björkman, Janne Valkonen
VTT, Systeemitutkimus, PL 1000, 02044 VTT
Puh. 020 722 6783, telefax 020 722 6027, etunimi.sukunimi@vtt.fi, http://www.vtt.fi

AVAINSANAT mallintarkastus, verifiointi & validointi, automaatiosuunnittelu

TIIVISTELMÄ

Mallintarkastus on formaali menetelmä kriittisten järjestelmien verifiointiin ja validointiin (V&V). Simulointi ja testaus perustuvat aina tiettyjen ennalta määriteltyjen sekvenssien ja kombinaatioiden läpikäyntiin, mutta mallintarkastuksen avulla voidaan kattavasti tarkastaa, voiko jostakin järjestelmästä tehty malli käyttäytyä ei-toivotulla tavalla, vasten järjestelmän vaatimuksia. Analyysin kattavuuden vuoksi ns. piileviä suunnitteluvirheitä on käytännön sovelluksissa löytynyt myös järjestelmistä, joita on jo tarkastettu testaamalla ja simuloimalla.

Suomessa mallintarkastuksen soveltuvuutta mm. automaatiosuunnittelun arviointiin on tutkittu kansallisessa ydinvoimalaitosten tutkimusohjelmassa SAFIR:ssa. Onnistuneista tutkimuspiloteista on nopeasti siirrytty käytännön sovelluksiin. VTT on soveltanut menetelmää niin voimalaitos-, tehdas-, kone- kuin sähköautomaationkin kohteisiin.

VTT:n toimesta mallintarkastusta on tähän mennessä useimmiten sovellettu suunnittelu- ja V&V-prosessin jo läpikäyneiden järjestelmien riippumattomaan arviointiin. Virheiden löytäminen näin myöhäisessä vaiheessa on kuitenkin erittäin kallista. Koska menetelmä on laskennallisesti tehokas, ja analyysi siten nopea suorittaa, herää kysymys, miksei mallintarkastusta ole otettu osaksi suunnittelun työkaluja ja prosesseja.

Vaikka mallintarkastukseen käytetyt työkalut ovat jo kypsiä, koko mallintarkastusprosessin läpikäynti sisältää edelleen monia vaiheita, jotka joko vaativat asiantuntemusta, tai koostuvat yksinkertaisesta ja toistuvasta käsityöstä. Järjestelmästä on ensin laadittava malli, ja lisäksi järjestelmään kohdistuvat vaatimukset on formalisoitava mallintarkastimen (tietokoneohjelma, jolla mallintarkastusta tehdään) käyttämään muotoon. Jos jokin mallin suoritus on vaatimusten vastainen, mallintarkastin palauttaa vastaesimerkin, joka kuvaa vaatimuksen rikkovan suorituspolun. Vastaesimerkkien tulkinta ilman tietoteknisiä apuvälineitä on työlästä.

Käytännön sovelluksissa on havaittu, että toimilohkokaaviopohjaisten järjestelmäkuvausten mallintarkastusta voidaan huomattavasti helpottaa rakentamalla malli modulaarisesti, valmiin toimilohkokirjaston perusteella. Käynnissä onkin työkalukehitys, jonka tavoitteena on automatisoida useita mallintarkastuksen työvaiheita. NuSMV-mallintarkastin on tarkoitus liittää Simanticsiin, joka on avoimen lähdekoodin ohjelmistoalusta eri mallinnus- ja simulointityökalujen integrointiin. Simanticsin avulla malli voitaisiin paitsi laatia graafisen käyttöliittymän avulla, jopa luoda automaattisesti jostain muusta, esim. prosessisimulaattorille (kuten Apros) laaditusta automaatiojärjestelmän mallista. Vaikkei kaikkia työvaiheita saataisikaan helposti automatisoitua, tavoitteena on kuitenkin tuoda mallintarkastus lähemmäs automaatiojärjestelmän suunnitteluvaihetta.

1 JOHDANTO

Ohjelmoitavan automaation käyttö niin kustannus- kuin turvallisuuskriittisissäkin kohteissa on jo arkipäivää. Perinteisesti ohjelmiston virheettömyys on pyritty osoittamaan mm. testauksella ja simuloinnilla. Virheitä jää kuitenkin löytämättä, johtuen siitä, ettei tarkastelu ole koskaan täysin kattavaa.

Etenkin turva-automaatiojärjestelmien ohjelmistosuunnittelussa on syytä suosia suoraviivaista binäärilogiikkaa. Tällaisissakin kohteissa järjestelmän mahdollisten tilojen määrä kuitenkin helposti kasvaa räjähdysmäisesti. Esimerkiksi järjestelmällä, jolla on viisikymmentä binääristä tuloa, on 10^{15} tulojen mahdollista kombinaatiota. Jos järjestelmässä on sisäistä muistia (jo yksinkertaiset kiikku- ja viivelohkot riittävät), tilojen määrä kasvaa entisestään. Kaikkien järjestelmän tilojen kattava tarkastus on vaadittavan laskenta-ajan vuoksi mahdotonta.

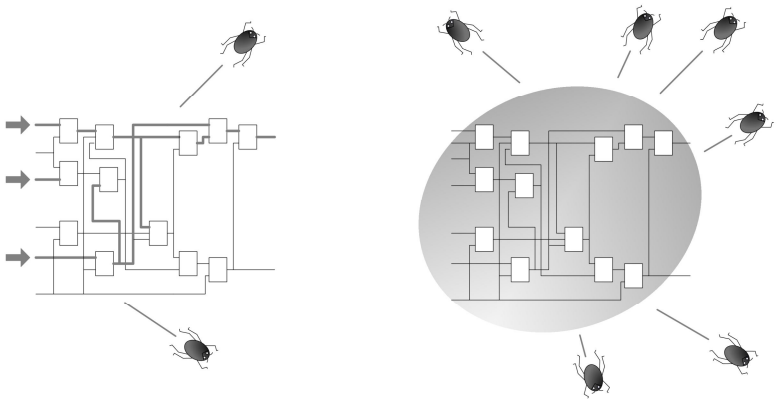
Testauksessa ja simuloinnissa on pohjimmiltaan kyse tiettyjen valittujen kombinaatioiden ja sekvenssien läpikäynnistä. Mallintarkastus /1/ on menetelmä, jossa asetelma käännetään hieman toisinpäin. Tavoitteena on näyttää toteen, ettei järjestelmän malli voi käyttäytyä ei-toivotulla tavalla. Tämän osoittamiseksi ei tarvitse laskea kaikkia mahdollisia tiloja, vain tietyn vaatimuksen kannalta olennaiset.

2 MALLINTARKASTUS JA OHJELMOITAVA AUTOMAATIO

Mallintarkastus /1/ on tietokone-avusteinen formaali menetelmä, jonka avulla voidaan varmistaa, että järjestelmän malli toimii järjestelmävaatimusten määrittelemällä tavalla mallin kaikissa tiloissa. Kyse ei ole uudesta menetelmästä, sillä mallintarkastuksen perusidea esiteltiin jo 80-luvulla, ja sitä on hyödynnetty laitteistosuunnittelussa 90-luvulta lähtien mm. mikroprosessorien suunnittelun tukena. Laskentakyvyn lisääntyminen ja mallintarkastukseen käytettyjen työkalujen kehitys ovat myöhemmin mahdollistaneet menetelmän soveltamisen uusiin ja monipuolisempiin kohteisiin /2/. Esimerkkejä uudemmista soveltajista ovat avaruus- ja ilmailuala /3/, toisaalta esiin tulevat yhä arkipäiväisemmät kohteet, kuten tiedonsiirtoprotokollat /4/ tai käyttöjärjestelmien laiteajurit.

Tarkasteltavasta järjestelmästä laaditaan malli, ja lisäksi formalisoidaan järjestelmää koskevat vaatimukset. Tietokoneohjelma nimeltä mallintarkastin analysoi tämän jälkeen, käyttäytyykö malli kaikissa tilanteissa vaatimusten kuvaamalla tavalla. Jos vaatimukset rikkova mallin suorituspolku löytyy, se palautetaan käyttäjälle vastaesimerkkinä. Vastaesimerkin avulla käyttäjä voi tämän jälkeen paikantaa mahdollisen suunnitteluvirheen.

Mallintarkastuksen soveltamista ohjelmoitavan automaation suunnittelun arviointiin on Suomessa tutkittu vuodesta 2007 alkaen kansallisessa ydinvoimalaitosten turvallisuustutkimusohjelmassa (SAFIR). Turvallisuuskriittinen automaatio on havaittu sopivaksi sovelluskohteeksi, ja SAFIR-ohjelman puitteissa on raportoitu onnistuneita pilottisovelluksia (/5, 6, 7/). Tutkimuksesta on siirrytty nopeasti käytännön soveltamiseen, VTT on mm. tarkastanut ydinvoimalaitosten automaatiojärjestelmiä Säteilyturvakeskuksen (STUK) toimeksiannosta.



Kuva 1 Testaamalla voidaan käydä läpi vain rajattu joukko testitapauksia. Mallintarkastus huomio kaikki oleelliset tarkasteltavan järjestelmän tilat, ja paljastaa siten virheitä joita testaamalla on vaikea havaita. /3, 8/

Täysin kattava tarkastelu kuulostaa kovalta lupaukselta. Syytä onkin tarkentaa, että tyypillisesti menetelmä soveltuu vain kohteisiin, jotka voidaan kuvata diskreettiaikaisina tilakoneina (jatkuva-aikaistenkin mallien tarkastelu onnistuu, mutta tällöin voidaan käsitellä vain hyvin rajattuja kohteita). Monimutkaisten säätöalgoritmien kuvaus ei siis onnistu, mutta turvallisuuskriittiselle automaatiolle ominainen suoraviivainen binäärilogiikka on mitä sopivin kohde.

Juuri analyysin kattavuuden vuoksi olemme mallintarkastuksen avulla kyenneet löytämään suunnitteluvirheitä myös sellaisista automaatiojärjestelmistä, jotka ovat jo läpikäyneet perinteisemmin keinoin tehdyn verifioinnin ja validoinnin (V&V). Koska ns. helpot tapaukset löytyvät perinteisilläkin menetelmillä, mallintarkastus tyypillisesti nostaa esiin epätodennäköisiä, tai oikeammin kummallisia tilanteita. Tyypillisiä syitä löydetyille virhetilanteille voivat olla 1) käyttäjän väärät tai väärin ajoitetut toimenpiteet, 2) satunnaiset, samanaikaiset signaalipulssit eri lähteistä, 3) viallinen, ristiriitainen tai puuttuva mittausdata, ja lisäksi 4) useampi edellä mainituista tilanteista samanaikaisesti. Tällaisten tilanteiden huomiointi esim. testaussuunnittelussa on vaativaa, ja mallintarkastuksen etu onkin, ettei tapahtumaketjuja tarvitse lainkaan määritellä tai keksiä.

3 KÄYTÄNNÖN HAASTEET MALLINTARKASTUSTYÖSSÄ

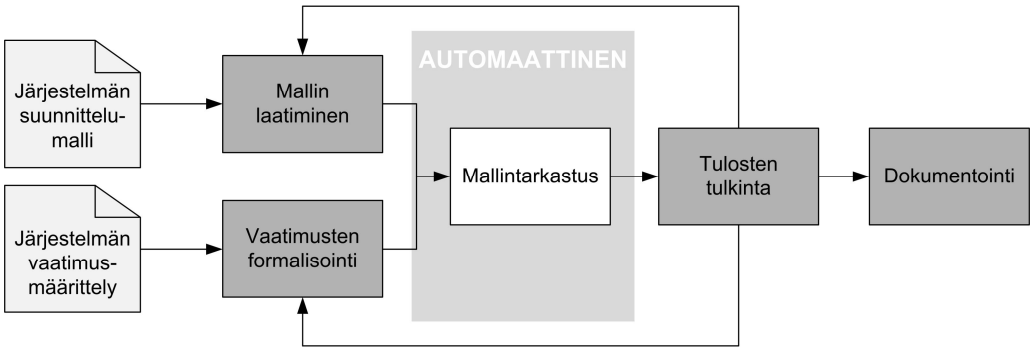
Mallintarkastuksen luotettavuus riippuu siitä, onko järjestelmän malli ja vaatimukset kuvattu riittävällä tarkkuudella ja oikein. Menetelmä on kuitenkin jossain määrin itseään korjaava. Suoranaiset virheet mallinnuksessa (tai vaatimusten formalisoinnissa) oireilevat usein mallintarkastimen tuottamina ”väärinä” vastaesimerkkeinä, joita analysoimalla virheen oikea lähde paljastuu. Kysymyksiksi jäävätkin lähinnä, onko malli riittävän tarkka ja oikein rajattu, ja ovatko kaikki vaatimukset otettu tarpeeksi kattavalla tavalla huomioon.

Koska menetelmä on laskennallisesti tehokas ja analyysi melko nopea suorittaa, herää kysymys, miksei mallintarkastusta voisi käyttää jo automaatio-ohjelmistojen suunnittelun yhteydessä. Mikroprosessorien valmistuksessa mallintarkastus on jo pitkään ollut integroituna suunnittelutyökaluihin. Mitä aiemmin suunnitteluvirheet huomataan, sitä vähemmän kustannuksia niiden korjaamisesta seuraa.

Merkittävimmät haasteet mallintarkastuksen käytännön soveltamiseen johtuvat tarvittavan käsityön määrästä. Mallintarkastuksen edellyttämä kokonaisprosessi sisältää monta virheille altista vaihetta, jotka vaativat joko paljon asiantuntemusta, tai paljon toistuvaa mekaanista työtä. (Kts. kuva 2.)

Mallin laatiminen edellyttää sekä lähdeaineiston että mallintarkastimen käyttämän mallinnuskielen tuntemusta. Esim. NuSMV-mallintarkastimen /9/ käyttämä tilakonepohjainen esitystapa edellyttää usein kohteen yksinkertaistamista, ja mallintajan on ymmärrettävä mitä erilaiset yksinkertaistukset merkitsevät mallin ilmaisuvoiman kannalta.

Koska toimilohkopohjaiset kielet ovat automaatio-ohjelmistoissa kohtuullisen yleisiä, olemme havainneet, että tällaisissa kohteissa mallin modularisointi nopeuttaa mallintarkastustehtävää huomattavasti. Mallintamalla käytetyt toimilohkot ensin valmiiksi kirjastoksi, toimilohkoista koostetun järjestelmän mallinnustehtävä muuttuu kopioinniksi. Järjestelmämallien koostaminen ei enää edellytä syvää asiantuntemusta, mutta voi käsin tehtynä olla edelleen virheille altista.



Kuva 2 Koko mallintarkastusprosessin läpikäynti sisältää vielä paljon käsin tehtävää työtä. /10/

Vaatimusten formalisointi edellyttää tyypillisesti luonnollisella kielellä ilmaistujen järjestelmän toiminnallisten vaatimusten esittämistä eksaktilla tavalla. Mallintarkastuksessa käytetään usein ns. temporaalilogiikkaa, jonka avulla voidaan kuvata järjestelmän toimintaa (sen tulojen, lähtöjen, ja sisäisten muuttujien funktiona) ajassa. Esimerkiksi: ”Järjestelmän on aina lopulta käynnistettävä toiminto X, jos tulon Y arvo on raja-arvoa suurempi”.

Haasteena on se, etteivät alkuperäiset vaatimukset aina ole yksiselitteisiä. Lisäksi, kutakin vaatimusta kohden mallintajan on tyypillisesti kirjoitettava useampi temporaalilogiikan lause. Tyypillisesti ns. negatiiviset vaatimukset (”järjestelmä ei saa käynnistää turvatoimintoa, jos sen oikeuttava kriteeri ei ole voimassa”) paljastavat suunnitteluvirheitä, joita testaamalla on vaikea havaita.

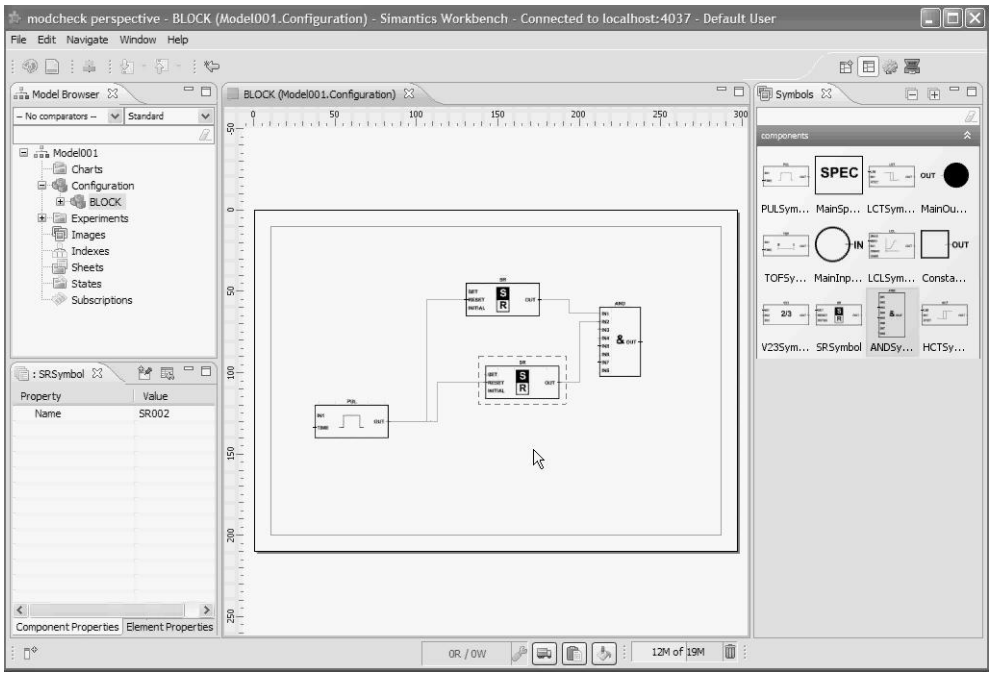
Temporaalilogiikan lauseista muodostuu helposti myös niin monimutkaisia, että mallintajan voi myöhemmin olla vaikea hahmottaa, mitä on alun perin tarkoittanut lausetta kirjoittaessaan.

Kuten edellä on mainittu, jos mallintarkastin löytää mallin suorituspolun, joka on vaatimusten vastainen, se palautetaan käyttäjälle vastaesimerkkinä. Tyypillisesti tulosten tulkinta koostuukin vastaesimerkkien analysoinnista. Esim. NuSMV palauttaa vastaesimerkit tekstimuotoisena listana, joka kuvaa miten mallin muuttujat käyttäytyvät ajassa, korostamatta mitenkään varsinaista ongelmakohtaa. Suurilla malleilla kyse voi olla tuhansien tekstirivien tulosteesta. Virheen lähteen identifiointi on turhauttavaa käsityötä, ja ongelma vain korostuu, jos mallinnusvirheet tuottavat vääriä vastaesimerkkejä, jotka on myös analysoitava.

4 SIMANTICS

Simantics /11, 12/ on avoimen lähdekoodin ohjelmistoalusta mallinnukseen ja simulointiin. Kehitys on lähtöisin VTT:ltä, mutta nykyään Simanticsin omistaa Teollisuuden hajautetun tiedonhallinnan yhdistys THTH ry, ja alustaa kehitetään kaikille avoimena yhteisöprojektina. Client/server-arkkitehtuurin perustana on semanttinen, ontologiapohjainen ydin, johon voidaan joustavasti kytkeä erilaisia mallinnus- ja simulointityökaluja plug-in -rajapinnan avulla. Simantics tarjoaa mallinnusympäristön näiden työkalujen tiedonsiirtoon ja yhteiskäyttöön. Reaaliaikaista mittausdataa voidaan myös liittää mukaan.

Useita kaupallisia ja ei-kaupallisia simulointi- ja suunnittelutyökaluja on jo kytketty Simanticsiin, esimerkkeinä OpenModelica, BALAS, Apros, OpenFoam, Comos ja SmartPlant. V&V-työkalujen liittäminen mukaan tarjoaa selkeitä etuja; esim. automaatiojärjestelmän Apros-mallista voitaisiin automaattisesti generoida mallintarkastimelle soveltuva malli. Työ NuSMV-mallintarkastimen integroimiseksi Simanticsiin onkin käynnissä.



Kuva 3 Simantics-alustalle rakennettu graafinen käyttöliittymä mallintarkastuksessa käytettävän mallin määrittelyyn valmiin toimilohkokirjaston pohjalta /10/

5 MALLINTARKASTUSPROSESSIN AUTOMATISOINTI

Käytännön mallintarkastustehtävissä olemme havainneet, että toimilohkokaavoin ilmaistujen järjestelmien arviointia voidaan helpottaa ongelman modularisoinnilla. Käytettyjen lohkojen toimintalogiikka kuvataan ensin mallintarkastimen käyttämällä kielellä, jonka jälkeen varsinainen asiantuntimusta vaatima mallinnustyö on enimmäkseen tehty. Tämän jälkeen minkä tahansa toimilohkoilla kuvatun järjestelmän mallinnus on käytännössä kopiointia, tai ”viivojen yhdistelyä”. Modulaarisen lähestymistavan avulla voidaan joitakin mallintarkastusprosessin vaiheita automatisoida kohtuullisen helposti.

5.1 Mallimuunnos

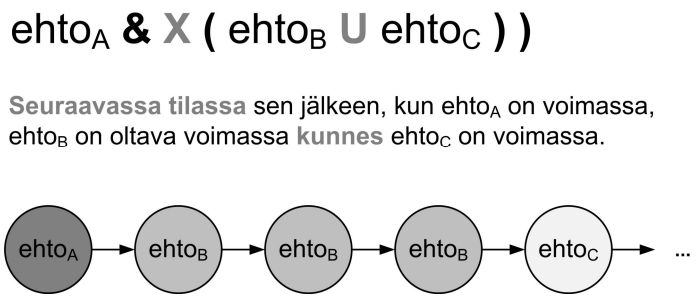
Kun toimilohkokirjasto on mallinnettu, voidaan varsinaisen järjestelmämallin laadintaa joko helpottaa tai automatisoida. Kuva 3 esittää prototyyppisovellusta, jossa Simantics-alustan käyttöliittymän avulla malli voidaan laatia valmiista lohkoista graafisessa näkymässä, ja generoituva koodi analysoidaan NuSMV-mallintarkastimella. Seuraava Simantics:n mahdollistama kehitysaskel on kaavioiden automaattinen muunnos jostain muusta automaatio-ohjelmiston mallista. Työvaihe nopeutuu, ja mallinnusvirheet vähenevät.

On syytä painottaa, että toimilohkopohjaistenkaan kohteiden mallinnus ei ole täysin suoraviivaista. Mallintarkastimen käyttämän kielen vaatimien yksinkertaistusten vuoksi mallinnuksessa törmätään joihinkin asioihin, joiden huomiointi vaatii mallintajalta ajatustyötä. Tällaisia ongelmakohtia ovat: 1) erimittaisten aikaviiveiden huomiointi, 2) analogisignaalien diskretointi tarvittavalla resoluutiolla, 3) asynkroniset mallit, joissa osia sovelluksista ajetaan eri suorittimilla, ja 4) liian laajat mallit /7/. Haasteena on löytää hyvä tasapaino mallin ilmaisuvoiman ja laskennallisen monimutkaisuuden välillä. Edellä mainittuihin ongelmakohtiin voidaan kuitenkin määritellä lähestymistapoja, toimenpidelistoja tai suunnittelumalleja (design pattern). Mallinnustyön automatisoimiseksi nämä tehtävät olisi toteutettava ohjelmallisesti.

5.2 Järjestelmävaatimusten formalisointi

Järjestelmävaatimusten kuvaaminen temporaalilogiikan rakenteina on työvaihe, jonka automatisointi on erittäin haasteellista. Vaatimukset on usein pääsääntöisesti kuvattu ei-formaalilla tavalla, yhden vaatimuksen kattava huomiointi vaatii useamman lauseen, ja temporaalilogiikan lauseista tulee helposti monimutkaisia. Mallintajaa avustavia työkaluja /A/ voidaan kuitenkin määritellä ja toteuttaa. Esimerkkejä työkaluista:

- Temporaalilogiikan lauseiden visualisointi tilakone-esityksinä
- Järjestelmän vaatimuslausetta noudattavan, esimerkillisen suorituspolun visualisointi (kuva 4)
- Temporaalilogiikan kaavojen esittäminen luonnollisella kielellä (kuva 4)
- Usein käytettyjen temporaalilogiikan rakenteiden suunnittelumallit (design patterns)
- Vaatimustenhallinnan työkalu, joka kytkee temporaalilogiikan lauseet alkuperäisiin järjestelmä-vaatimuksiin.



Kuva 4 Yksinkertainen temporaalilogiikan lause havainnollistettuna sekä kirjoittamalla se auki luonnolliselle kielelle, että esittämällä mallin esimerkkisuoritus, joka noudattaa lausetta.

5.3 Tulosten visualisointi

Mallin vaatimusten vastaista toimintaa esittävät vastaesimerkit tulisi esittää käyttäjän helposti ymmärtämässä muodossa. Koska kyse on mallin käyttäytymisestä jonkin ajanjakson aikana, selkeintä olisi näyttää vastaesimerkit ”2D-animaationa” alkuperäisten toimilohkokaavioiden avulla. Vaihtamalla signaalilangan väriä binääriarvon mukaan tai esittämällä analogisignaalin arvo numerona saavutetaan havainnollinen esitys, josta suunnittelijan on helpompi huomata virheen syy. Vastaesimerkkien 2D-visualisointi Simantics-ympäristössä on yksi ominaisuuksista, joita tulemme kehittämään jatkossa.

6 JOHTOPÄÄTÖKSET

Suomessa mallintarkastuksen soveltamista automaatiokohteisiin on vienyt eteenpäin ydinvoima-alan toimijoiden kiinnostus, ja tutkimusta jatketaankin edelleen mm. SAFIR-ohjelmassa.

VTT on useissa käytännön sovelluksissa osoittanut, että mallintarkastuksen avulla on mahdollista löytää suunnitteluvirheitä kohteista, joita on jo tarkasteltu perinteisin V&V-menetelmin. Koska menetelmä on laskennallisesti tehokas, olisi hyödyllistä saada mallintarkastus osaksi suunnittelijan käytössä olevia työkaluja. Mallintarkastuksen soveltaminen vaatii kuitenkin melko paljon käsityötä. Suurin osa työvaiheista on mahdollista automatisoida, keskeisistä toimenpiteistä lähinnä järjestelmävaatimusten formalisointi on haasteellista.

Simantics on avoimen lähdekoodin ohjelmistoalusta, joka mahdollistaa eri suunnittelu- ja mallinnustyökalujen tehokkaan integroinnin. Työn alla on NuSMV-mallintarkastimen kytkeminen Simanticsiin, mikä mahdollistaisi mm. mallin generoinnin mallintarkastimelle suoraan esim. automaatiojärjestelmän Apros-mallista.

Automaatio-ohjelmistojen mallintarkastusta voidaan etenkin toimilohkokaavioin esitetyissä kohteissa tehostaa, mutta muitakin ohjelmointikieliä on. VTT on tarkastanut myös tilakonepohjaisia (Prosa) järjestelmiä, mutta mallin automaattista generointia tällaisiin kohteisiin ei ole tutkittu.

7 KIRJALLISUUSLUETTELO

1. Clarke E.M. Jr., Grumberg O., Peled D.A.: Model Checking, The MIT Press, 1999.

2. Burch J., Clarke E., McMillan K, Dill D., Hwang L.: Symbolic Model Checking: 10^20 States and Beyond. Information and Computation, 98(1992), pp. 142-170.

3. Cofer D., Whalen M., Miller S.: Model-Checking of Safety-Critical Software for Avionics. ERCIM News 75, October 2008, pp. 15-16.

4. Holzmann G.: The model checker Spin, IEEE Transactions on Software Engineering, 23(1997)5, pp. 279-295.

5. Valkonen J., Björkman K., Frits J., Niemelä I.: Model checking methodology for verification of safety logics, Proceedings of the 6th International Conference on Safety of Industrial Automated Systems – SIAS 2010, Tampere, 14.-15.6., 2010.

6. Björkman K., Valkonen J., Ranta J.: Verification of Automated Changeover Switching Unit by Model Checking, 7th International Topical Meeting on Nuclear Plant Instrumentation, Control and Human-Machine Interface Technologies – NPIC&HMIT 2010. Las Vegas, Nevada, November 7.-11., 2010.

7. Lahtinen J., Björkman K., Valkonen J., Frits J., Niemelä I.: Analysis of an emergency diesel generator control system by compositional model checking, VTT Working Papers 126, VTT Technical Research Centre of Finland, 2010. ISBN 978-951-38-7497-1.

8. Pakonen A., Valkonen J.: Mallintarkastus löytää automaation piilevät suunnitteluvirheet. Automaatioväylä, 26(2010)7, 16-17.

9. Cavada R., Cimatti A., Jochim C.A., Keighren G., Olivetti E., Pistore M., Roveri M., Tchaltsev A.: NuSMV 2.5 User Manual, ITC-IRST, http://nusmv.irst.itc.it/.

10. Pakonen A., Lahtinen J., Kuutti V-P, Karhela T.: Integrating Model Checking with Safety-Critical I&C Software Design, 7th International Topical Meeting Nuclear Plant Instrumentation, Control and Human-Machine Interface Technologies – NPIC&HMIT 2010. Las Vegas, Nevada, November 7.-11., 2010.

11. Simantics – a software platform for modelling and simulation, https://www.simantics.org/.

12. Villberg A., Lehtonen T., Karhela T., Kondelin K.: Applying Semantic Modelling Techniques in Large Scale Process Simulation, Proceedings of the 1st IFAC Workshop on Applications of Large Scale Industrial Systems – ALSIS '06, Suomen Automaatioseura, 2006.