Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

Verifikációs módszerek biztonságkritikus beágyazott rendszerekhez

Habilitációs tézisek

Majzik István
PhD
egyetemi docens

Budapest, 2013. október

Tartalomjegyzék

1. Bevezetés

Számítógépes rendszerek verifikációjának (helyességigazolásának) célja, hogy megállapítsuk, a fejlesztési és üzemeltetési életciklus minden fázisának kimenete a specifikáció és az előzetes fázisok által meghatározott követelményeket teljesíti. Biztonságkritikus beágyazott rendszerek esetén a verifikációval szemben támasztott elvárások különösen nagyok, hiszen egy bennmaradó fejlesztési vagy konfigurációs hiba balesethez, környezeti károkhoz is vezethet. Az ilyen rendszerek fejlesztése során alkalmazott verifikációs módszereket elsősorban a következő tényezők és kihívások határozzák meg:

- A specifikált követelmények nemcsak a rendszer funkcionális (logikai) helyességére, hanem olyan, tipikusan valószínűségi jellemzőkkel (időfüggvényekkel vagy várható értékekkel) megadott extra-funkcionális jellemzőire is vonatkoznak, mint a megbízhatóság, rendelkezésre állás, karbantarthatóság és biztonságosság [45]. Ennek megfelelően a verifikációnak ezeket a követelményeket is igazolnia kell.
- Biztonságkritikus rendszerekben a verifikációt részletesen előírják a fejlesztési szabványok (pl. az IEC 61508, ami az elektromos, elektronikus és programozható elektronikus rendszerek általános funkcionális biztonsági szabványa, vagy az ebből származó szakterület-specifikus szabványok, mint a vasúti szoftverekre vonatkozó EN 50128 vagy a gépjárművek szoftvereire vonatkozó ISO 26262). A szabványok az alkalmazás biztonságintegritási szintjétől függően részletesen előírják az egyes fejlesztési fázisokban kötelezően, nyomatékosan ajánlott vagy ajánlott módon elvégzendő verifikációs tevékenységek körét. A verifikáció eredményei a tanúsításhoz szükséges biztonságigazolás alapjául is szolgálnak.
- A szabványokban is előírt széles körű és részletes verifikáció adott idő- és költségkeretek között történő elvégzése precíz, hatékony, automatikus eszközökkel jól támogatott technikákat igényel.

Ezeket a kihívásokat a korszerű fejlesztési folyamatokban a következő alapvető trendeket követve igyekeznek hatékonyan teljesíteni:

- *Tervezői döntések korai verifikációja*: A fejlesztés során minél később kerül sor egy hiba detektálására, a javítása annál költségesebb. Ennek megfelelően a fejlesztés minél korábbi fázisában, lehetőleg már a tervezés során érdemes megvizsgálni a döntések helyességét, illetve következményeit a fejlesztett rendszer specifikált jellemzőire.
- *Modellvezérelt fejlesztés*: A korai ellenőrzésre lehetőséget ad, ha a fejlesztés modellvezérelt, azaz a tervezői döntéseket jól definiált modellek rögzítik (pl. architektúra és viselkedés modellek formájában). A későbbiekben a verifikált modellek képezik az implementáció (forráskód- és konfigurációgenerálás) alapját.
- *Formális módszerek alkalmazása*: Amennyiben az alkalmazott modellek formális szintaxissal és szemantikával rendelkeznek, akkor matematikailag precíz verifikációs algoritmusok használhatók, amelyek a specifikált tulajdonságok bizonyítására is lehetőséget adnak. (Az átvizsgálás, a szimuláció és a tesztelés, mint a leggyakrabban használt nem-formális verifikációs technikák, tipikusan nem alkalmasak teljes körű ellenőrzésre.)
- *Automatikus eszközök használata*: A formális modellek kezeléséhez és a verifikációs algoritmusok alkalmazásához szükséges speciális szaktudás eszközökbe integrálható. A gyakorlatban is bevált megközelítés a mérnöki modellek automatikus leképzése

(rendszerint alacsonyabb szintű) formális modellekké, amelyeken a verifikációs algoritmusok végrehajthatók, majd az eredmények visszacsatolhatók a mérnöki modellbe. Így mintegy „gombnyomásra” (automatikus modell-leképezési és modell-verifikációs lépéseken keresztül) elérhetőek a formális verifikáció eredményei, kézi újramodellezés és analízis nélkül.

A kutatásaim során ezekhez az igényekhez és trendekhez igazodva választottam kutatási célokat, a fejlesztési folyamat meghatározott verifikációs feladatainak támogatására.

2. Kutatási célok és módszerek

A kutatás során a modellvezérelt fejlesztés korai fázisaiban alkalmazható, hatékonyan automatizálható (eszközökkel támogatható), formális alapokkal rendelkező módszerek kidolgozását tűztem ki célul, a fejlesztési folyamat következő fázisaira fókuszálva:

- *Az architektúra tervezése*: Az architektúra tervezése alapvetően meghatározza egy fejlesztett rendszer szolgáltatásbiztonságát. Az első célkitűzésem az architektúra mérnöki modellje alapján a megbízhatóság, rendelkezésre állás és biztonságosság kvantitatív analízise. Ezáltal a szolgáltatásbiztonság szempontjából összevethetők lehetséges architektúra változatok, igazolhatók a kapcsolódó tervezői döntések, illetve azonosíthatók az elvárások teljesítése szempontjából kritikus komponensek.

  Az ad-hoc hálózati kapcsolatokra épülő elosztott rendszerekben az architektúra által nyújtott szolgáltatások és funkciók köre dinamikusan változhat, mégpedig nemcsak az egyes komponensek meghibásodása és helyreállítása, hanem a komponensek elérhetőségének változása miatt is. Ez meghatározza a szolgáltatásokra épülő felhasználói tevékenysorozat, vagy magasabb szintű alkalmazás sikeres végrehajtását. A második célkitűzésem a dinamikus környezetben futó alkalmazások megbízhatóságának analízise, ezáltal az architektúra tervezésének (szolgáltatások telepítésének) és a hálózati kapcsolatok felhasználásának validációja.

- *A komponensek részletes tervezése*: A komponensek részletes tervezése során történik a belső viselkedés meghatározása, ami az implementáció alapja. A célkitűzésem diszkrét állapotú, eseményvezérelt viselkedést leíró modellek alapján a viselkedés biztonsági és élőségi tulajdonságainak verifikációja, formalizált állapot elérhetőségi követelmények ellenőrzésével.

A célkitűzések részleteit, az alkalmazási körülményeket (feltételrendszert) és a kutatási módszereket a következő alfejezetek mutatják be.

2.1. Architektúra tervek verifikációja a megbízhatóság és biztonság szempontjából

Az architektúra megbízhatósági és biztonsági analízisének [46] alapfeladata, hogy meghatározza a rendszerszintű megbízhatósági és biztonsági jellemzőket az egyes komponensek lokális megbízhatósági jellemzői (pl. a komponensválasztásból és terhelésből adódó meghibásodási gyakoriság), a köztük lévő kapcsolatok lokális megbízhatósági jellemzői (pl. a hibaterjedési valószínűség), valamint a redundancia struktúrákban betöltött szerepük (pl. variáns, szavazó) alapján. A rendszerszintű megbízhatósági jellemzőket ebben az esetben a megbízhatóság és rendelkezésre állás valószínűségi időfüggvényei, valamint a hibák bekövetkezésének idejére vonatkozó várható értékek jelentik. Biztonsági jellemzőként a

kritikus hibák gyakoriságát tekintik, és ezt (ha pontosabb meghatározás nincs) a detektálatlanul maradó hibák gyakoriságaként veszik számításba.

A modell alapú megbízhatósági analízis [47] a rendszerszintű jellemzők számítását egy matematikailag precíz *megbízhatósági analízis modell* konstruálásával és annak analitikus vagy szimulációval történő megoldásával (azaz a modell jellemzőinek, pl. a hibamentes szolgáltatáshoz tartozó állapotokban való tartózkodás valószínűségének kiszámításával) éri el. Ha egyes komponensek lokális jellemzői nem pontosan ismertek, akkor is lehetőség van paraméterezhető modellek alapján architektúra változatok tulajdonságainak összevetésére, vagy a rendszerszintű jellemzők érzékenységének vizsgálatára. Független meghibásodási és javítási folyamatok esetén a megbízhatósági modell egyszerű kombinatorikus modell lehet (pl. hibafa, eseményfa, megbízhatósági blokkdiagram). Az általánosabb, állapotfüggő meghibásodási és hibakezelési folyamatok modellezésére a diszkrét állapotú, folytonos idejű sztochasztikus modellek terjedtek el. Alacsony szintű formalizmus a folytonos idejű Markov-lánc (CTMC [48]), konkurens meghibásodási és hibakezelési folyamatok kompakt modellezését is támogató magasabb szintű formalizmusok pedig az általánosított sztochasztikus Petri-hálók (GSPN [49]) vagy a sztochasztikus aktivitás hálózatok (SAN [50]). Ezek a formalizmusok különböznek a meghibásodási és javítási folyamatok sztochasztikus jellemzőinek megadásában is [21].

A megbízhatóság modell alapú analízise speciális szaktudást igényel. Fel kell mérni az architektúra komponenseihez köthető meghibásodási és hibaterjedési folyamatokat, valamint ezek rendszerszintű hatásait, figyelembe véve az architektúra tervezés során meghatározott hibakezelési megoldásokat (reaktív hibakezelés, redundancia használata, javítási folyamatok) [51]. Amennyiben ezek leírhatók analízis modell részletek formájában, akkor az architektúra terv alapján a rendszerszintű megbízhatósági modell összeállítása már egy szisztematikus munka [52], amit az architektúra terv változásakor újra el kell végezni. Ez eszközökkel segíthető, amennyiben az egyes rendszerkomponensekhez megbízhatósági szakértők előzetesen meghatározzák az adott alkalmazásra jellemző meghibásodási és hibaterjedési folyamatokat, ezekhez analízis modell részleteket rendelnek, ezekből pedig az architektúra terv alapján automatikus eszköz állítja elő a rendszerszintű megbízhatósági modellt. Így az egyes tervváltozatokhoz az analízis gyorsan és automatikusan, a megbízhatósági szakértők on-line közreműködése nélkül is elvégezhető.

A megbízhatósági modell automatikus összeállítási módszerének kidolgozása során UML alapú architektúra modelleket (osztály, objektum, csomag és telepítési diagramokat) vettem alapul. A megbízhatósági analízishez természetesen szükséges ezeknek a diagramoknak a kiterjesztése (UML profil segítségével) annak érdekében, hogy a lokális megbízhatósági jellemzők megadhatók legyenek [11]. A kutatás megkezdésekor ehhez szabványos UML profil még nem állt rendelkezésre, így saját megoldás készült; a későbbiekben OMG[1] szabvány is megjelent ilyen célra [53].

Ezeket az alapkoncepciókat figyelembe véve a kutatási célkitűzésem a következőkben összegezhető:

1. célkitűzés: Automatizálható módszer kidolgozása UML alapú architektúra tervek alapján megbízhatósági analízis modellek szisztematikus összeállítására, így lehetővé téve rendszerszintű megbízhatósági és biztonsági jellemzők számítását, ezzel az architektúrára vonatkozó tervezői döntések verifikálását.

[1] Object Management Group, http://www.omg.org/

2.2. Dinamikus környezetben végrehajtott tevékenységek és alkalmazások megbízhatóságának verifikációja

A kritikus beágyazott alkalmazások köre, elsősorban a mobil számítástechnika térhódításával, nemcsak a statikus fizikai és informatikai környezetben telepített, hanem a dinamikusan változó környezetben futó alkalmazásokra is kiterjed [54]. Ilyenek elsősorban a közlekedésben megjelenő alkalmazások (pl. veszély előrejelzés, kríziskezelés, elosztott fekete doboz [6]). Ezekben az alkalmazásokban jellemzően egy-egy (felhasználói) *tevékenységsorozat* végrehajtása történik, amely meghatározott funkciók elérését igényli a mobil ad-hoc valamint fix hálózati infrastruktúrán alapuló kapcsolatok, valamint a lokális és távoli erőforrások (komponensek) rendelkezésre álló szolgáltatásai által meghatározott dinamikus környezetben.

Ilyen alkalmazások esetén a megbízhatóság legfontosabb aspektusa, hogy adott környezetben milyen valószínűséggel futhat le sikeresen a tevékenységsorozat, figyelembe véve az igényelt funkciók *elérhetőségét* a mobilitás miatt dinamikusan változó hálózati topológia következtében, valamint a funkciók *rendelkezésre állását* az általuk használt szolgáltatásokat biztosító erőforrások meghibásodásának és helyreállításának függvényében. A modell alapú megbízhatósági vizsgálatok lehetőséget adnak már a tervezés korai fázisában elvégezni ezt a verifikációt, így elemezve kiválasztott forgatókönyveket (pl. szélsőséges környezeti feltételek hatásait), redundáns szolgáltatások használatának hatékonyságát, valamint az igénybe vett erőforrások jellemzőire való érzékenységet.

Kritikus mobil alkalmazások kvantitatív jellemzőinek analízise során a legfontosabb kihívás a komponensek és hálózati kapcsolatok számából, heterogenitásából és dinamikusságából adódó komplexitás kezelése. Ehhez a HIDENETS kutatási projektben[2] egy holisztikus és több absztrakciós szintet bevezető általános keretrendszert dolgoztunk ki:

- A holisztikus megközelítés lényege többféle analízis módszer (analitikus, szimulációs és kísérleti megoldások) integrálása, mivel egy módszer tipikusan nem képes a felhasználói szempontú jellemzők teljes analízisének végrehajtására [6]. Az egyszerűbb analízis részfeladatok megoldásához használt módszerek eredményeinek integrálására a következő általános interakciók használhatók: A *keresztvalidáció* lényege, hogy egy egyszerűbb részfeladat megoldása validál (érvényesít) egy összetett feladat megoldásához használt feltételezést (pl. szimulációval validálható egy analitikus modellben használt valószínűségi változó eloszlásfüggvénye). A *megoldás visszacsatolás* lényege, hogy egy részfeladat megoldása bemenetét képezi egy másik feladat megoldásának (pl. mérésekből származik egy paraméter értéke). A *problémafinomítás* lényege, hogy egy részfeladat megoldásából származó tudás alapján finomítható a teljes megoldás (pl. egy megbízhatóság szempontjából kritikusnak bizonyuló komponens esetén módosítható az analízis modell).
- A hierarchikus modellezés koncepciója több absztrakciós szintet definiál [55]. A legalsó hierarchiaszint az erőforrások és a kommunikációs kapcsolatok megbízhatósági modelljeit tartalmazza. A közbenső szint a szolgáltatások majd a funkciók megbízhatóságát modellezi, figyelembe véve ezek egymásra és az erőforrásokra épülését az architektúra terveknek megfelelően. A legfelső hierarchiaszint a felhasználói tevékenységek megbízhatóságának modellje. A hierarchikus felépítés lehetőséget ad a holisztikus interakciókra: modell paraméterek validálására, részmodellek elkülönített megoldására és az eredmények becsatolására a magasabb hierarchiaszintű modellbe, valamint adott részmodellek finomítására.

[2] Highly Dependable IP-based Networks and Services (HIDENETS), FP6 IST STREP 26979. http://www.hidenets.aau.dk/

Ezeket az alapkoncepciókat a fent említett megbízhatósági analízis feladat megoldása során is célszerű figyelembe venni, azaz olyan módszert kell kidolgozni, amely támogatja a hierarchikus modellezést és a kapcsolódó holisztikus interakciókat.

A megbízhatósági modell összeállításához a vizsgálandó forgatókönyveket az alkalmazás tervezése során előálló következő mérnöki modellek határozták meg:

- Felhasználói tevékenységek modellje (felhasználói munkafolyamat): Ez rögzíti a tevékenységsorozatot (megengedve elágazásokat és párhuzamos tevékenységeket is), a tevékenységek idő adataival kiegészítve. Egy tevékenység egy funkció (ezen keresztül szolgáltatások) igénybevételét jelenti. Elosztott alkalmazások esetén felhasználónként külön tevékenységmodellek készülnek. A felhasznált formalizmus idő adatokkal kiegészített UML aktivitásdiagram.
- Architektúra modell: Ez rögzíti a funkciók, szolgáltatások és erőforrások egymásra épülését (függőségeit). Az alkalmazáshoz szükséges minden hálózati csomóponthoz készül. A felhasznált formalizmus az UML architektúra modell diagramjai.
- Hálózati topológia modellje: Ez rögzíti az időben változó lehetséges hálózati kapcsolatokat. Nem kézi tervezéssel készül, hanem mobilitás modellek [56], útvonal és forgalom szimulátor felhasználásával automatikusan származtatható egy-egy vizsgálandó forgatókönyvhöz, több alkalmazható hálózati technológia esetén külön-külön származtatva a topológia modelleket. Ezek a hálózati topológia modellek (metamodellel definiált) specifikus modellek formájában készülnek.

Ezeket a feltételeket és alapkoncepciókat figyelembe véve a kutatási célkitűzés a következőkben összegezhető:

2. célkitűzés: Egy módszer kidolgozása, amely lehetővé teszi felhasználói tevékenységsorozat sikeres végrehajtási valószínűségének meghatározását, az alkalmazástervezés során előálló tevékenység és architektúra modellek, valamint a dinamikus hálózati környezetet leíró modellek alapján, lehetőséget adva holisztikus interakciókra részmodellek finomítása és részfeladatok megoldása során.

2.3. Állapot alapú, eseményvezérelt viselkedés verifikációja

Beágyazott rendszerekben az alkalmazások jelentős köre diszkrét állapot alapú, eseményvezérelt jellegű viselkedést valósít meg, amely (aszinkron) elosztott illetve időfüggő is lehet. Kritikus alkalmazásokban a viselkedés verifikációját már a tervezési fázisban célszerű megkezdeni a tervezési hibák (pl. nem biztonságos állapotba kerülés, hiányzó válasz miatti leragadás) detektálása és javítása érdekében. A verifikáció automatizálása régóta érvényes célkitűzés, az évek során nagyszámú megoldási javaslat született a viselkedés és az elvárt tulajdonságok formalizálására, valamint a formális leírásokra épülő verifikációs technikákra [57] [58]. Ezek gyakorlati elterjedését az korlátozta, hogy a javasolt formalizmusok sokszor távol álltak a mérnöki modellezéstől (absztrakt és alacsony szintű matematikai koncepciókat alkalmazva), valamint a verifikációs technikák használata is speciális szaktudást igényelt (pl. bizonyítási stratégiák kidolgozásához). Az első hátrány kezelhető, amennyiben az elterjedt mérnöki modelleket le tudjuk képezni a formális szemantikával rendelkező alacsonyabb szintű modellekre. A másik hátrány pedig kiküszöbölhető a speciális szaktudást nem igénylő, „gombnyomásra működő” verifikációs technikák, elsősorban a legsikeresebb eszközöket kínáló *modellellenőrzés* alkalmazásával [59].

A kutatás megkezdésekor beágyazott rendszerek fejlesztéséhez a speciális fejlesztői környezetek (pl. Esterel SCADE Suite[3], MathWorks Simulink StateFlow[4]) mellett terjedőben volt az UML modellek, így a viselkedés megadására az *UML állapottérképek* használata. A formális verifikációt ez esetben az nehezítette, hogy az UML szabvány rögzítette az állapottérkép diagramok szintaxisát és statikus szemantikáját (az alkalmazható modell elemeket és ezek megengedett kapcsolatait), de nem adott formális dinamikus szemantikát az állapottérkép által leírt viselkedés precíz származtatásához.

Az UML állapotgép alapú fejlesztés támogatásához így szükségessé vált a dinamikus szemantika rögzítése. A korábbi eredményekre, tehát létező formalizmusokra és a hozzájuk kidolgozott verifikációs technikákra és eszközökre leginkább úgy lehet építeni, hogy az UML állapottérképek dinamikus szemantikáját egy már létező formalizmusra való leképzéssel adjuk meg. Az állapot alapú viselkedést leíró állapottérképek esetén a Kripke struktúrákra illetve Kripke tranzíciós rendszerekre való leképzés volt a nyilvánvaló választás. (Egy későbbi továbbfejlesztésünk az időfüggő viselkedést is formalizáló időzített automatákra való leképzést valósította meg.) A kihívást a leképzés definiálása során az jelentette, hogy az UML állapottérkép formalizmus a konkurens működés hierarchikus modellezéséhez kötődően sok, nem-triviális konstrukciót megenged: állapotfinomítást hierarchikus és konkurens alállapotokkal, állapothierarchián átívelő átmeneteket, szétváló és egyesülő átmeneteket; valamint az állapothierarchiának megfelelő prioritásokat értelmez a konfliktusban lévő átmenetek között.

A leképzéssel definiált formális dinamikus szemantika alapján a mérnöki modell formális modellel történő újramodellezése nélkül, tervezési időben elvégezhető az UML állapottérkép modellek formális verifikációja, ami temporális biztonsági és élőségi tulajdonságok ellenőrzésével koncepcionális és tervezési hibákat tud kimutatni.

A formális szemantikát érdemes kihasználni a verifikáció kiterjesztésére, a fejlesztési életciklus későbbi fázisait is megcélozva:

- A tesztelés támogatható modell alapú teszt fedettségi kritériumok szerinti *automatikus tesztgenerálással*. A tesztek végrehajtásával implementációs és a futtató platformmal való együttműködési hibák is detektálhatók. A modell alapú tesztgenerálás igénye már a legújabb fejlesztési szabványokban (pl. ISO 26262) is megjelenik.
- A *futásidőbeli hibadetektálás* támogatható a szükséges monitor komponenseknek a viselkedés modell valamint a formalizált követelmények alapján történő szintézisével. A monitorozással működés közbeni véletlen hibák és konfigurációs hibák is detektálhatók. A futásidőbeli hibadetektálás kötelező előírásként jelenik meg biztonságkritikus rendszerek fejlesztési szabványaiban (pl. IEC 61508).

Ezeket az igényeket és alapkoncepciókat figyelembe véve a kutatási célkitűzés a következőkben összegezhető:

3. célkitűzés: Módszer kidolgozása UML állapottérképekkel modellezett állapot alapú, eseményvezérelt viselkedés formális modellekre történő leképzésére, lehetővé téve így a formális verifikációt, az automatikus tesztgenerálást, valamint a futásidőbeli ellenőrzést.

[3] http://www.esterel-technologies.com/products/scade-suite/

[4] http://www.mathworks.com/products/stateflow/

3. Új tudományos eredmények

Az új tudományos eredményeket a kutatási célkitűzések szerinti csoportosításban mutatom be.

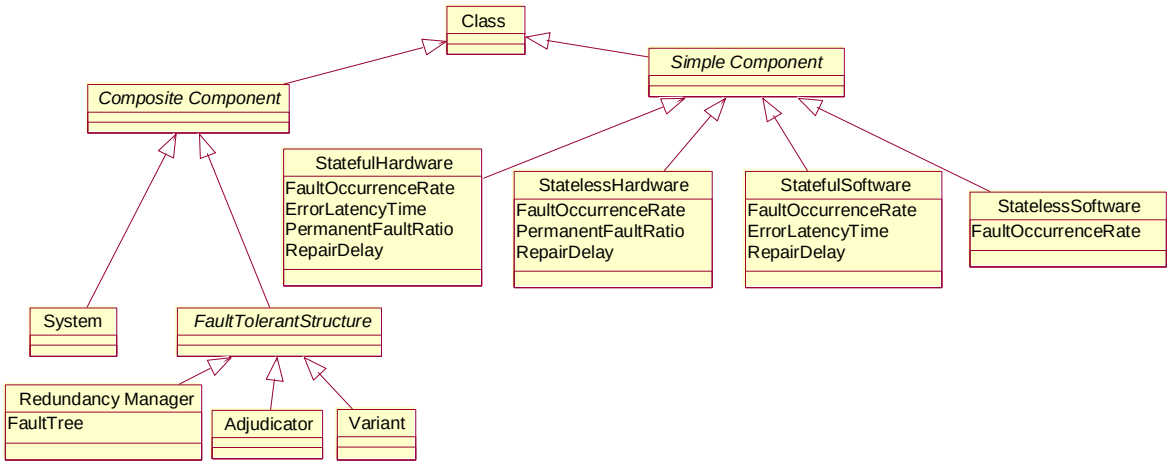
3.1. UML architektúra tervek verifikációja megbízhatósági modellezéssel

A modell alapú tervezés lehetőséget adott arra, hogy a mérnöki architektúra modellből (a konkrét kutatás keretében olyan UML architektúra modellből, amiben UML profilban rögzített kiterjesztések írják le az egyes komponensek és kapcsolatok lokális megbízhatósági jellemzőit) szisztematikusan történjen a megbízhatósági analízis modell előállítása. A kapcsolódó új tudományos eredményeket a következő pontok adják meg.

3.1.1. Absztrakt megbízhatósági modell definiálása

Mivel a megbízhatósági analízis modell többféle formalizmus segítségével is összeállítható, valamint az architektúra modellek is többféle módon készülhetnek, célszerű volt bevezetni köztes modellként egy *absztrakt megbízhatósági modellt*, ami származtatható az architektúra modellből, majd külön fázisban leképezhető a választott analízis formalizmus szerinti *konkrét megbízhatósági modellre*. Így a verifikáció bemeneti és kimeneti formalizmusainak függése szétcsatolható, kiemelve a verifikáció szempontjából lényeges információkat.

A fenti gyakorlati szempontok mellett az absztrakt megbízhatósági modell jelentőségét az adja, hogy ez egy matematikailag precíz modell (hipergráf), ami minden, a megbízhatósági analízis szempontjából releváns információt tartalmaz, de még nem kötődik konkrét analízis formalizmushoz. Metamodellje definiálja az architektúrában szereplő komponensek típusait és jellemzőit (pl. állapottal rendelkező vagy nem rendelkező hardver illetve szoftver komponens, szerepkör a redundáns struktúrákban, ld. 1. ábra), valamint hasonlóan a köztük lévő kapcsolatok típusait és jellemzőit (pl. használat, tartalmazás redundáns struktúrában).

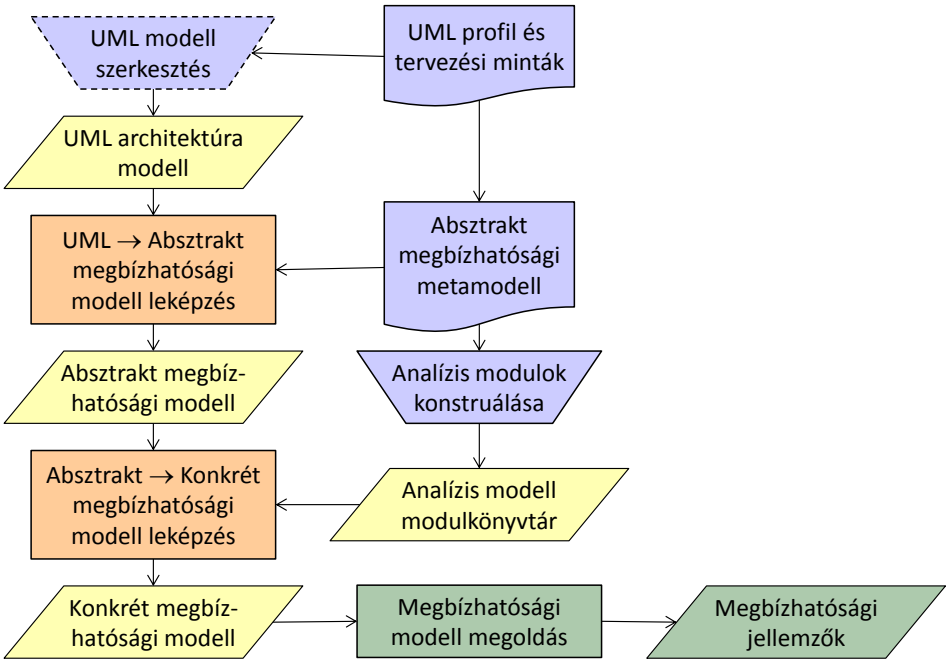


1. ábra A komponensek típusai és jellemzői

A kutatás során meghatároztam az UML architektúra modellek megbízhatósági analízishez szükséges kiterjesztéseit, majd definiáltam az absztrakt megbízhatósági modell származtatását. A módszer egy kiterjesztéseként hallgatói munka keretében megvalósult az AADL architektúra leíró nyelvben szereplő komponens típusok leképzése is.

3.1.2. A megbízhatósági modell összeállítása analízis modulok alapján

A konkrét megbízhatósági modell magasabb szintű formalizmusai (GSPN, SAN) támogatják a moduláris modellépítést [60]. Így az absztrakt megbízhatósági modell elemtípusaihoz *analízis modell modulok* (azaz a GSPN vagy SAN alhálói) rendelhetők, ezekből pedig paraméterezés után szisztematikusan, az absztrakt megbízhatósági modellben található kapcsolatok alapján összeállítható a konkrét megbízhatósági modell. Ez a megközelítés lehetőséget ad analízis modulkönyvtárak létrehozására, nemcsak komponens típusokhoz, hanem redundáns struktúrák tervezési mintáihoz is kötve. Az analízis modulok alkalmazásával történő megbízhatósági modell konstrukció és analízis így kidolgozott folyamatát a 2. ábra mutatja be.



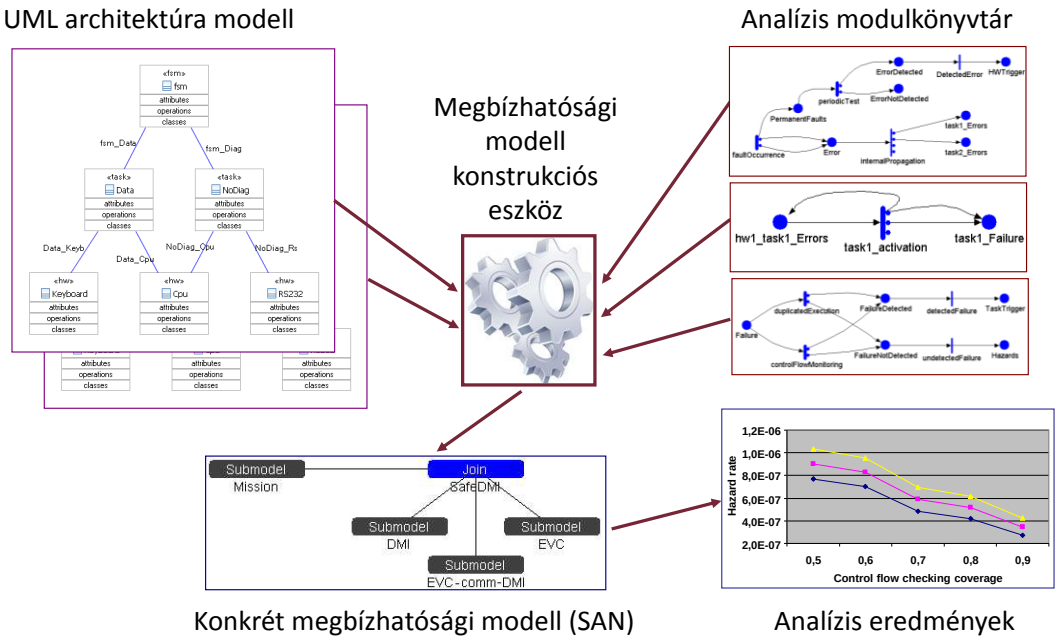
2. ábra Az UML alapú architektúra tervek megbízhatósági analízise

Az analízis modulok testreszabhatók az alkalmazás speciális igényei szerint. A kutatás során analízis modell modulokat, paraméterezési és összeállítási módszereket javasoltam a következő architektúra megoldásokhoz:

- kompozit (redundáns funkcionalitású komponenseket alkalmazó) hibakezelés hibatűrő rendszerekhez,
- reaktív (független hibadetektálást és beavatkozást alkalmazó) hibakezelés fail-stop jellegű megoldásokhoz,
- FT-CORBA köztesréteg hibakezelése [61] elosztott objektum-orientált rendszerekhez.

A moduláris modellezés ezek használata mellett lehetőséget ad egyes modulok független finomítására is (pl. amikor a tervezés során többlet információ áll rendelkezésre), így szelektíven finomítható azoknak az architektúra részeknek az analízis modellje, amelyekre érzékenynek mutatkoznak a rendszerszintű megbízhatósági jellemzők.

Az elméleti eredmények alapján automatikus eszköz készült UML architektúra modellekből az absztrakt megbízhatósági modell leképzésére, valamint ez alapján GSPN majd SAN formájában a konkrét megbízhatósági analízis modell összeállítására (3. ábra). A modellek megoldására a Möbius eszköz szolgált.



3. ábra Az UML alapú megbízhatósági modell konstrukciós eszköz

3.1.3. Aspektus-orientált architektúra modellezés és analízis

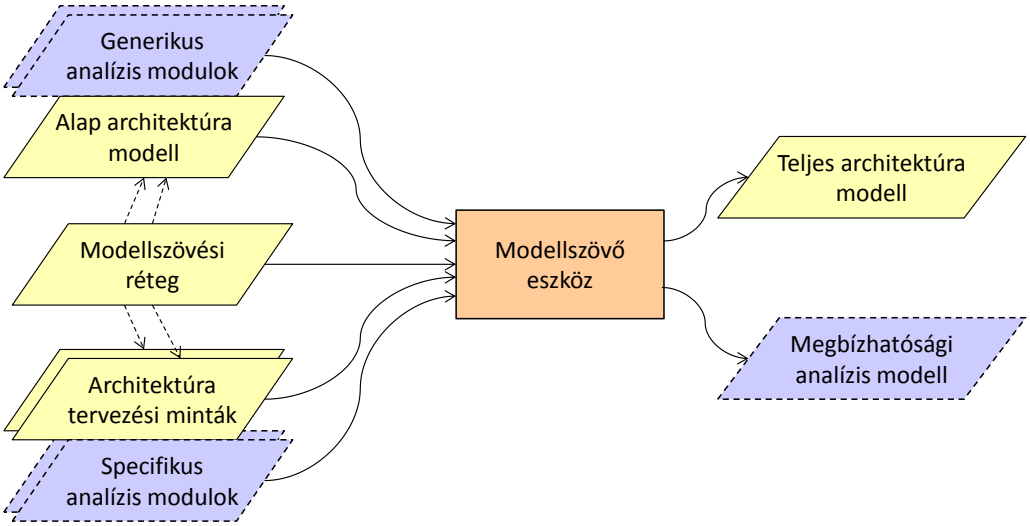
Az aspektus-orientált modellezés megközelítése szerint külön aspektusmodellekbe foghatók össze a követelmények különböző szempontjaihoz (aspektusaihoz) tartozó tervezői döntések [62], és ezek alapján a teljes rendszermodell aspektusmodell szövéssel [63] állítható össze.

Ezt az elvet javasoltam alkalmazni az architektúra tervezés során a megbízhatóság és rendelkezésre állás szempontjaihoz tartozó tervezői döntések összefogására, valamint egyúttal ezen döntések modell alapú verifikációjára. A javasolt modellezési és modellszövési módszer fő elemei a következők:

- A redundáns architektúrák tervezése során külön aspektusmodellekbe kerülnek a hibakezeléshez szükséges többlet komponensek (pl. hibadetektorok, variánsok kezeléséhez szükséges komponensek). Ezek az aspektusmodellek tipikusan jól bevált tervezési mintákat kötnek az alap architektúra modell kijelölt részeihez, ahogy azt forráskód szinten egyik kapcsolódó kutatásunk bemutatta [64]. A teljes architektúra modell ezek után aspektusmodell szövéssel állítható össze.
- A modellszövés kiterjed a megbízhatósági modell összeállítására is: az aspektusmodellekben hivatkozott tervezési mintákhoz rendelt specifikus analízis modulokat és az alap architektúra komponenseihez rendelt generikus analízis modulokat modellszövő kapcsolja össze, így előállítva a teljes architektúra *konkrét megbízhatósági analízis modelljét* is.

A kétféle modellszövés végrehajtását a 4. ábra mutatja be, itt a modellszövési rétegnek elnevezett aspektusmodell köti a tervezési mintákat az alap architektúra modellhez.

A módszer alkalmazásával a bevált architektúra megoldások és a hozzájuk rendelhető specifikus analízis modulok felhasználása is szisztematikussá, jól áttekinthetővé és könnyen módosíthatóvá válik.



4. ábra Architektúra modell és megbízhatósági modell előállítása modellszövéssel

3.1.4. Az új tudományos eredmények tézisszerű összefoglalása

1. tézis

Módszert dolgoztam ki UML alapú architektúra tervek megbízhatósági és biztonsági analízisére sztochasztikus megbízhatósági analízis modellek szisztematikus és moduláris összeállításával.

1.1. Az architektúra tervben szereplő, a megbízhatóság szempontjából releváns komponensek és kapcsolatok reprezentálására kidolgoztam és metamodelljének megadásával definiáltam egy absztrakt megbízhatósági modellt. Az UML architektúra modellek megbízhatósági analízise érdekében meghatároztam a szükséges kiterjesztéseket, és ez alapján megadtam az UML architektúra tervek elemeinek leképzését az absztrakt megbízhatósági modellre. [1][5]

1.2. Az absztrakt megbízhatósági modell alapján meghatároztam azokat az analízis modell modulokat, amelyekből modulárisan összeállítható egy konkrét megbízhatósági analízis modell, sztochasztikus aktivitás hálózat formájában. Analízis modell modulokat definiáltam kompozit (redundáns komponenseket alkalmazó) hibakezeléshez, reaktív (független hibadetektálást alkalmazó) hibakezeléshez, valamint az FT-CORBA köztesréteg hibakezelésére épülő architektúrákhoz. [2][6] [3][7]

1.3. Kidolgoztam a redundáns architektúrák aspektus-orientált modellezésének módszerét, ami külön aspektusmodell formájában teszi lehetővé a megbízhatóság növelése érdekében szükséges járulékos komponensek tervezését. Megadtam, hogy az aspektus-orientált architektúra modell alapján hogyan állítható össze modellszövéssel a rendszerszintű megbízhatósági modell. [4][8]

---

[5] A cikk többszerzős, a saját eredményem volt az absztrakt megbízhatósági modell definiálása, az UML profil kidolgozása és az UML modell elemek leképzése.

[6] A cikk többszerzős, a saját eredményem volt a megbízhatósági analízis modulok meghatározása. A cikk egy gyakorlati alkalmazásban mutatja be a megbízhatósági modellezést.

[7] A cikk kétszerzős, társszerzőm munkája az állapottérképek alapján történő analízis (6.1. fejezet).

[8] A cikk kétszerzős, saját eredményem volt a redundáns architektúrák aspektus-orientált modellezésének és a megbízhatósági modell modellszövéssel való előállításának módszere.

A megbízhatósági modell konstrukciót további publikációk is ismertetik [11] [12] [13] [14], kitérve alkalmazás-specifikus megbízhatósági analízis modulokra is [15]. Az aspektus-orientált modellezés szintén több publikáció témája volt [16] [17] [18]. Az elméleti eredmények alapján megbízhatósági modell konstrukciós eszközt is kidolgoztunk [19] [20], amit fejlesztési projektekben eredményesen használtunk.

A megbízhatósági modellezés alapfeladatainak megismerésében Andrea Bondavalli segített. A komponens- és hibaterjedés modellezésben felhasználhatók voltak Pataricza András kapcsolódó eredményei. Az első, sztochasztikus Petri-háló alapú analízis modell komponenseket Ivan Murával állítottuk össze. Az UML modellek feldolgozását és a megbízhatósági modellek automatikus összeállítását végző eszközöket Magyar Melinda fejlesztette. Az aspektus-orientált modellezéshez szükséges eszközöket és modelleket Domokos Péter készítette.

3.2. Felhasználói tevékenységek dinamikus környezetben történő végrehajtásának verifikációja megbízhatósági modellezéssel

A dinamikus környezetben végzett felhasználói tevékenységek sikeres végrehajtási valószínűségének számítása adott forgatókönyvek alapján a megbízhatósági analízis modell összeállításával és megoldásával történt. A forgatókönyvek az analízis modell összeállításának bemeneteként tartalmazták a felhasználói tevékenységek modelljeit, a mobilitás modellből származó hálózati topológia modelleket, valamint a rendszer architektúrájának (a funkciók, szolgáltatások és erőforrások egymásra épülésének) modelljét. Az analízis modell összeállításához részben felhasználható volt az 1. tézisben kidolgozott módszer, mivel ez a hierarchikus modellezéshez lehetővé teszi az architektúra modellnek megfelelő szintű megbízhatósági modell részletek származtatását. Ugyanakkor új megoldásokat igényelt a végrehajtott tevékenységek és a hálózati topológia időbeli változásainak kezelése. Az új tudományos eredményeket a következő pontok mutatják be.

3.2.1. A megbízhatósági analízis modell összeállítása idő alapú dekompozícióval

A felhasználói tevékenységek és a hálózati környezet (a bemeneti modellekben explicit módon megadott) változásainak kezelésére *idő alapú dekompozíciót* javasoltam, külön fázisokként kezelve a változások közötti „stabil” szakaszokat, amikor csak az erőforrások véletlenszerű meghibásodását kell figyelembe venni. Az idő alapú dekompozíció szerinti analízishez a többfázisú rendszerekhez (Multiple Phased System, MPS [65]) kidolgozott módszert adaptáltam. Az MPS modellek két részből állnak, ezek a szokásos megnevezés szerint a *fázismodell* (phase net, ami determinisztikus időzítésű tranzíciókkal bővített GSPN), és a *rendszermodell* (system net, ami GSPN). A rendszermodellben hivatkozhatók a fázismodell állapotai.

A forgatókönyv szerinti bemeneti modellek alapján meghatároztam az MPS analízis modell összeállításának módszerét:

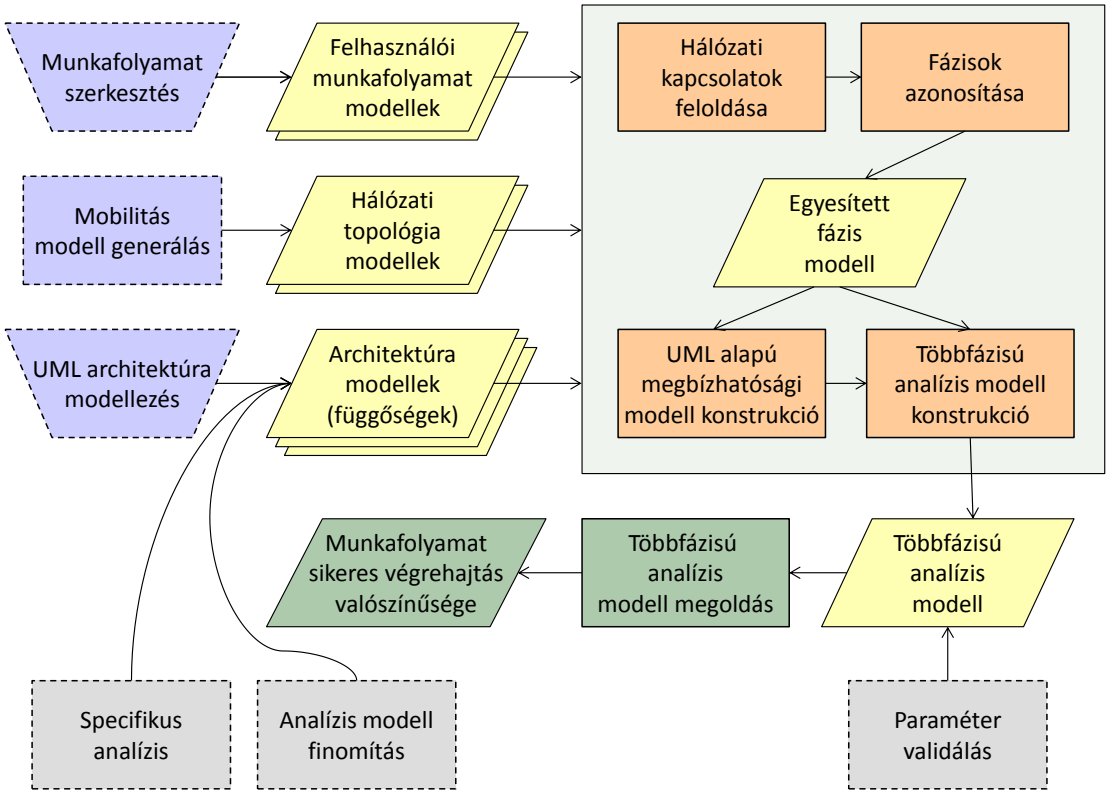
- A fázismodell egyes fázisait (ezek modellezését) a felhasználói tevékenységek és a hálózati topológia modell által leírt változások alapján határoztam meg. Egy-egy fázis addig tart, amíg változatlan a tevékenység és az általa használt hálózati kapcsolatok.
- A rendszermodellt a tevékenységek által igényelt, a hálózati kapcsolatokon elért funkciók mögött megjelenő rendszer architektúra alapján határoztam meg, az 1. tézis szerinti módszerrel modulárisan összeállítva az architektúrához tartozó megbízhatósági analízis modellt.

Az így előálló többfázisú analízis modell megoldására a Möbius [66] és a DEEM [67] eszközök voltak használhatók.

3.2.2. Analízis munkafolyamat definiálása a holisztikus interakciók támogatásával

A többfázisú megbízhatósági analízis modell összeállítása majd az analízis végrehajtása több modell-leképzési és modell-analízis lépést igényelt. Követelményként jelent meg a holisztikus analízis során szükséges keresztvalidáció, megoldás visszacsatolás, valamint problémafinomítás lépések támogatása.

A kutatás során az adott verifikációs feladatra egy olyan, nagy részben automatizált analízis munkafolyamatot javasoltam, ami támogatja ezeket a holisztikus interakciókat is. Az analízis folyamat meghatározza a használandó modell-összeállítási, -paraméterezési és -analízis lépéseket (5. ábra), valamint az interakciók helyét.



5. ábra Dinamikus környezetben végrehajtott tevékenységek megbízhatósági analízise

A többfázisú megbízhatósági analízis modell hierarchikus, az összeállítás folyamata a felhasználói, architektúra (funkció, szolgáltatás és erőforrás komponensek) és kommunikációs szinteknek megfelelően építi be az analízis modell modulokat. Így a megoldás visszacsatolás közvetlenül megjelenhet: amennyiben egy funkció vagy szolgáltatás jellemzője specifikus analízissel (pl. külön modellezéssel és szimulációval) számítható ki, akkor ez becsatolható és kiváltja az alsóbb szintű „generikus” analízis modell modulokat. A problémafinomításra az ad lehetőséget, hogy az érzékenységvizsgálattal kritikusnak bizonyuló komponensekhez tartozó analízis modulok tovább finomíthatók, és a finomított modulok építhetők be. A keresztvalidáció akkor releváns, ha egy analízis modul paraméterét külső vizsgálattal (pl. mérésekkel) kell érvényesíteni.

3.2.3. Az új tudományos eredmények tézisszerű összefoglalása

2. tézis

Módszert dolgoztam ki felhasználói tevékenységek sikeres végrehajtásának modell alapú verifikálására olyan dinamikus forgatókönyvek esetén, amelyek tartalmazzák a felhasználói tevékenységsorozatok modelljeit, a mobilitás modellből származó hálózati topológia modelleket, valamint a rendszer architektúrájának modelljét. Így az analízis a használt erőforrások meghibásodása mellett a felhasználói tevékenységek és a távoli szolgáltatások eléréséhez szükséges hálózati kapcsolatok időbeli változásait is figyelembe veszi.

2.1. Koncepciót dolgoztam ki a megbízhatósági analízis modell idő alapú dekompozícióval történő összeállítására, többfázisú analízis modell (Multiple Phased System, MPS) formájában. Származtattam az MPS fázismodelljét a felhasználói tevékenységek és a hálózati topológia változásainak modelljeiből, valamint az MPS rendszermodelljét az architektúra modell elemeiből. [5][9]

2.2. Definiáltam egy olyan analízis munkafolyamatot, ami integrálja az automatikusan végrehajtható modell-leképezési és modellanalízis lépéseket, valamint lehetőséget ad a holisztikus analízis során alkalmazható keresztvalidáció, megoldás visszacsatolás, valamint problémafinomítás jellegű interakciókra. [6][10]

A kidolgozott, többféle bemeneti modell feldolgozásával megvalósuló megbízhatósági analízis módszert ismertettem a modell alapú megbízhatósági analízist áttekintő könyvfejezetben is [21].

A holisztikus analízis megközelítés Andrea Bondavalli, Paolo Lollini és Mohamed Kaaniche kezdeményezése volt, ehhez illeszkedett a tézisben szereplő megbízhatósági analízis munkafolyamat. Az analízishez szükséges konkrét modell-feldolgozási lépéseket és eszközöket Kovács Máté dolgozta ki. A munkafolyamat eszközeinek integrálásában Magyar Melinda is közreműködött.

A dinamikus környezetben végzett tevékenységek verifikációjához kötődő további kutatásaink meghatározott missziót teljesítő autonóm rendszerek (pl. robotok) kontextusfüggő viselkedésének tesztelésére fókuszáltak. A misszió biztonságos végrehajtására vonatkozó követelményeket egy kiterjesztett UML szekvencia diagram alapú scenario nyelvvel írtuk le, ebben hivatkozva a releváns környezetet megadó kontextus modell mintákra. Módszereket dolgoztunk ki teszt kontextusok automatikus generálására a viselkedés biztonságosságának és robusztusságának teszteléséhez, többféle teszt stratégia szerint [22]. Ezen felül módszereket adtunk monitor komponensek szintézisére szimulátorban vagy valós környezetben rögzített teszt lefutások automatikus kiértékeléséhez [23].

3.3. UML állapottérképek verifikációja formális modellekre történő leképzéssel

Az UML állapottérképek verifikációjához választott megközelítés a szemantika formalizálása alacsonyabb szintű, formális szemantikával rendelkező matematikai modellekre történő leképzéssel, majd ezekhez a modellekhez illeszkedő verifikációs technikák alkalmazásával. Az új tudományos eredményeket a következő pontok ismertetik.

---

[9] A cikk többszerzős. Saját eredményem az idő alapú dekompozícióval történő analízis módszer.

[10] A cikk többszerzős. Saját munkám az analízis munkafolyamat meghatározása (6. fejezet), valamint közreműködtem a 2. és 3. fejezet kidolgozásában.

3.3.1. UML állapottérképek leképzése Kripke struktúrákra kiterjesztett hierarchikus automatákon keresztül

Az UML állapottérképeknek az eredeti, Harel-féle állapottérkép formalizmusból átvett alapvető koncepciója a konkurens és hierarchikus állapotfinomítás. A hierarchikus állapotfinomítás reprezentálására formális szintaxisként a *hierarchikus automaták* szintaxisa alkalmazható, ezt az UML-től különböző szemantikájú Harel-féle állapottérképekhez már javasolták formális modellként [68]. A hierarchikus automatákhoz azt a formális szemantikát kellett tehát hozzárendelni, ami megfelel az UML szabványban leírt viselkedésnek. Ez a formális szemantika egy Kripke struktúra, ami közvetlenül lehetővé teszi a temporális logikák használatával formalizált követelmények modellellenőrzéssel történő verifikációját.

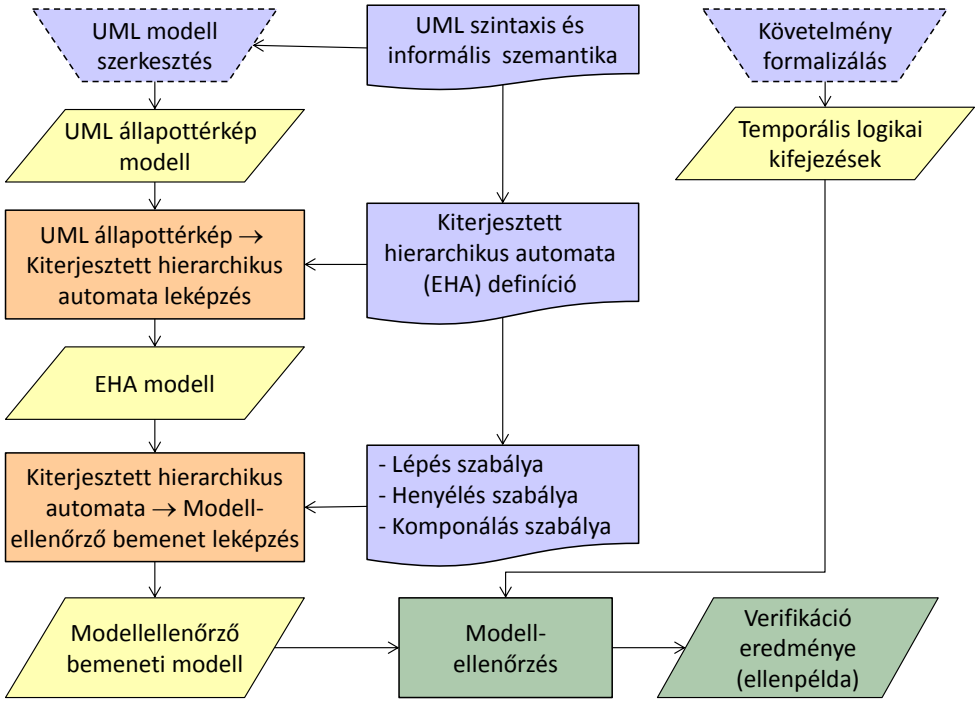
A szemantika formalizálásakor alapvető kihívást jelentettek az UML szemantika specialitásai és eltérései a Harel-féle állapottérképekhez képest. Többek között ilyen a konfliktusban lévő átmenetekre az UML-ben alkalmazott prioritási séma, ami az alacsonyabb hierarchiaszinten lévő forrásállapottal rendelkező átmenethez rendel nagyobb prioritást. Ennek reprezentálására a hierarchikus automaták olyan kiterjesztésére volt szükség, amiben a hierarchiaszinteken átívelő átmenetek az úgynevezett „legkisebb közös ős" állapot hierarchiaszintjén reprezentálhatók (ez tartalmazza az átmenet által elhagyott illetve az átmenet által elérendő alállapotokat), ugyanakkor a konfliktusok feloldásához megőrzik a forrásállapotra vonatkozó információt is.

A szükséges kiterjesztések bevezetésével definiáltam a *kiterjesztett hierarchikus automaták* (Extended Hierarchical Automata, EHA) formalizmusát, és megadtam az UML állapottérkép legfontosabb konstrukcióinak leképzését EHA modellre. Az EHA (ezáltal közvetve az UML állapottérkép EHA alapú) formális szemantikájaként megjelenő Kripke struktúra állapotainak és átmeneteinek meghatározása 3 *szemantikai szabállyal* történt:

- *Lépés szabálya*: A kiterjesztett hierarchikus automata egy automatája egy lépést tesz (azaz állapotátmenetet hajt végre), amennyiben található benne engedélyezett és a prioritása szerint végrehajtható átmenet.
- *Komponálás szabálya*: Egy automata komponálja az alsóbb hierarchiaszinteken lévő automatáinak lépéseit, ha azokban hajthatók végre lépések.
- *Henyélés szabálya*: Egy automata henyél, ha nincs végrehajtható átmenete, és nincs lépést végrehajtó alsóbb szintű automatája sem.

Ez a megoldás különösen hatékonnyá tette a modellellenőrzők alkalmazását, mivel a modellellenőrző bemeneti modelljében csak a kiterjesztett hierarchikus automatát (mint szintaktikus struktúrát) kellett megadni és a szemantikai szabályokat kellett megvalósítani (6. ábra). Ezzel elkerülhető volt az állapottérképhez tartozó Kripke struktúra közvetlen generálása, mivel ezt maga a modellellenőrző végzi, a szabályok alapján bejárva az állapotteret. Ezáltal a hatékony állapottér kezelésről már a modellellenőrző kifinomult belső megoldásai (szimbolikus technikák, részleges rendezés redukció) tudnak gondoskodni. Ez a megközelítés valósult meg az EHA modellnek a SPIN modellellenőrző [69] Promela bemeneti nyelvére történő leképzése esetén.

Az így definiált formalizálás lefedi a legfontosabb UML állapottérkép konstrukciókat: a hierarchikus és konkurens alállapotokkal történő állapotfinomítást, az állapothierarchián átívelő átmeneteket, a szétváló és egyesülő átmeneteket, a konfliktusban lévő átmenetek közötti prioritások kezelését. A formalizálás nem támogatja a ritkán használt konstrukciókat: az emlékező állapotokat, az aktivitás állapotokat, a paraméteres, késleltetett és változás eseményeket, valamint a dinamikus objektumlétrehozást és -törlést megvalósító eseményeket. Ezek nagy részének formalizálása kapcsolódó kutatásaink során valósult meg [38].



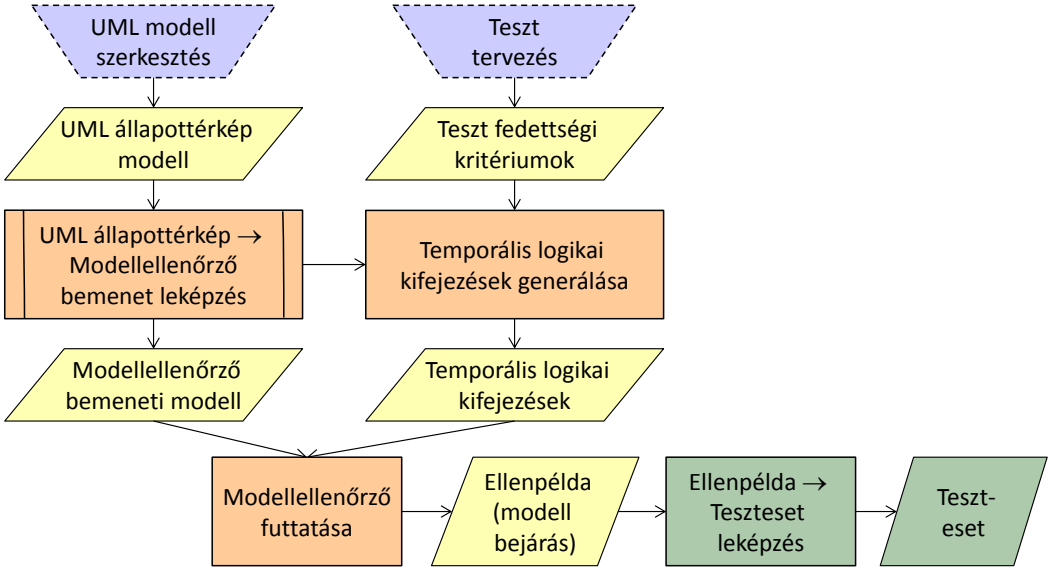
6. ábra UML állapottérképek formális verifikációja

A formális szemantika közvetlenül lehetővé teszi modellellenőrzők használatát a viselkedés biztonsági és élőségi tulajdonságainak ellenőrzésére. Ezek mellett specifikus verifikációs módszereket javasoltam a kivételkezelés [27] és hibatűrő rendszerekben a dinamikus replikakezelés ellenőrzésére is [28].

3.3.2. Tesztgenerálás az UML állapottérkép EHA formális szemantikája alapján

A formális szemantikával rendelkező UML állapottérkép modell lehetővé teszi, hogy ez alapján automatikusan történjen tesztesetek generálása a tényleges implementáció és a modell (mint referencia) viselkedésének összevetésére. A tesztelés nem csak kézi kódolással készült implementáció verifikálására alkalmas, hanem automatikusan generált implementáció viselkedésének validálására is. A szisztematikus tesztelést az biztosíthatja, ha a modell alapján generált tesztkészlet meghatározott fedettségi kritériumokat teljesít (pl. a modell minden állapotának, vagy minden állapotátmenetének fedését). Ezeket a fedettségi kritériumokat leképezve temporális logikai kifejezésekre, a modellellenőrző ellenpélda generálási módszere használható fel teszteset generálásra [70]: a modellellenőrző által generált ellenpélda éppen egy olyan modell állapottér bejárás lesz, ami az adott fedési kritérium szerinti tesztelési célt teljesíti (pl. egy állapot fedését megvalósító teszteset generálható annak specifikálásával, hogy az adott állapot nem érhető el – erre ellenpéldaként egy olyan bejárást generál a modellellenőrző, ami éppen ezt az állapotot fedi le).

A modellellenőrzővel történő tesztgenerálást korábban csak alacsony szintű matematikai modellekre és a Harel-féle állapottérkép szemantikára alkalmazták [71]. Ezt kiterjesztettem az UML állapottérképek EHA szemantikája alapján történő tesztgenerálásra: meghatároztam a tesztgenerálás folyamatát, ebben felhasználva a modellellenőrző bemenetére történő leképzést (itt az EHA modellnek a SPIN modellellenőrző Promela nyelvére történő leképzését), valamint a fedettségi kritériumok temporális logikai formalizálását (7. ábra).



7. ábra UML állapottérkép modell alapján történő tesztgenerálás

3.3.3. A vezérlési folyam futásidőbeli ellenőrzése az EHA formális szemantika alapján

A formális szemantikával rendelkező viselkedési modell arra is alkalmas, hogy a tervezési időben már verifikált modell, mint referencia alapján a *futásidőbeli viselkedés* is ellenőrizhető legyen. Mivel a modell állapot alapú, eseményvezérelt viselkedést rögzít, ez alapján a lokális vezérlési folyamot befolyásoló hibák detektálhatók: ilyen hatásai lehetnek a működés közben fellépő véletlen (hardver) hibáknak, a tesztelésből esetleg kimaradó implementációs vagy konfigurációs hibáknak, valamint a platformmal való együttműködés hibáinak. Ez a hibadetektálási módszer kiterjeszti a watchdog processzorok koncepcióját [72], mivel itt az ellenőrzés nem az implementált forráskódból kinyert vezérlési gráf, hanem a tervezés során készített és ellenőrzött UML állapottérkép modell alapján történik, kibővítve így az ellenőrzés körét az említett implementációs és konfigurációs hibákra is.

- A vezérlési folyam futásidőbeli ellenőrzéséhez szükséges monitor komponensek megvalósítására egy *monitor szintézis módszert* javasoltam. A monitor referenciaként az UML állapottérképet annak EHA alapú formális szemantikája szerint használja fel, tehát azt vizsgálja, hogy a futásidőbeli viselkedés megfelel-e ennek a modellnek. A monitor az aktuális futásidőbeli viselkedésről az állapotokhoz rendelt és futásidőben a monitorhoz küldött jelzőszámok alapján értesül, ezeket az állapottérkép alapján generált kódvázba aspektus-orientált programozással lehet illeszteni. Az így generált monitor egy-egy komponens szintjén képes a vezérlési folyamot befolyásoló hibákat detektálni, a részletes tervezés során elkészített állapottérkép modell szerint követve a bejövő események feldolgozását.
- Elosztott rendszerekben a lokális monitorozást kiegészítettem a *komponensek közötti interakciók* hibáinak detektálásával, a rendszerszintű követelmények alapján monitorozva az események generálását és továbbítását is. A rendszerszintű követelmények tipikusan temporális logikákkal és az elvárt illetve lehetséges lefutásokat megadó scenario modellekkel formalizálhatók. Az interakciókat ellenőrző monitorokhoz az UPPAAL modellező környezetben [73] használt TCTL (Timed Computational Tree Logic) temporális logika és LSC (Live Sequence Charts [74])

üzenet szekvencia diagram variánsok alapján, ezek formális szemantikájára építve javasoltam szintézis módszereket.

A futásidőbeli verifikáció végezhető lineáris idejű [75], vagy invariáns jellegű (minden végrehajtásra felírt) elágazó idejű temporális logikai kifejezések alapján generált monitorok alkalmazásával. Ennek kiterjesztéseként javasoltam az egzisztenciális jellegű TCTL kifejezések (melyekben az „exists" útvonal kvantor szerepel) alapján generált monitorok használatát, mégpedig a tesztelés során egy-egy tesztkészlet által kiváltott lefutás halmaz együttes ellenőrzésére.

3.3.4. Az új tudományos eredmények tézisszerű összefoglalása

3. tézis

Módszereket dolgoztam ki UML állapottérkép modellek kiterjesztett hierarchikus automatákra történő leképzésére formális verifikáció, modell alapú tesztgenerálás, és futásidőbeli ellenőrzés céljaira.

3.1. Kidolgoztam UML állapottérképek kiterjesztett hierarchikus automatákra, mint formális modellre történő leképzését, elemezve az UML szabványban rögzített szintaxist, statikus szemantikát és az informálisan megadott dinamikus szemantikát. A leképzés támogatja a konkurens dinamikus viselkedést leíró legfontosabb UML állapottérkép konstrukciókat: a hierarchikus és konkurens állapotfinomítást, az állapothierarchián átívelő átmeneteket, a szétváló és egyesülő átmeneteket, és a konfliktusban lévő átmenetek közötti prioritások kezelését. [7][11]

3.2. Modell alapú tesztgenerálási módszert javasoltam az UML állapottérkép modellből leképzett kiterjesztett hierarchikus automata modell alapján, alkalmazva a modell alapú teszt fedettségi kritériumok szerint végrehajtott modellellenőrzést az ellenpélda alapú tesztgenerálásra.[8][12]

3.3. Monitor szintézis módszert javasoltam a komponens szintű vezérlési folyam futásidőbeli ellenőrzéséhez, az UML állapottérkép kiterjesztett hierarchikus automata alapú formális szemantikáját felhasználva referenciaként [9][13]. A monitor szintézis módszert kiterjesztettem elosztott rendszerek komponensei közötti interakciók ellenőrzésére, az UPPAAL modellező környezetben használt TCTL temporális logika és Live Sequence Charts (LSC) variánsok használatával formalizált rendszerszintű követelményeket felhasználva referenciaként. [10][14]

Az UML állapottérképek szemantikájának formalizálását Diego Latellával és Mieke Massinkkal publikáltuk, akik a formalizálás alapján az EHA szemantikai szabályainak Promela nyelven történő megvalósítását végezték. Külön publikációinkban részletesebben is szerepel az UML állapottérképek EHA formalizmusra történő leképzése [24] valamint a formális szemantika további felhasználása [25]. Ezt a szemantikát adtam meg a BME VIK Formális módszerek tárgyának oktatásához jegyzetként használt „Formális módszerek az

[11] A cikk többszerzős. Saját munkám az UML állapottérképek kiterjesztett hierarchikus automatákra történő leképzése.

[12] A cikk többszerzős. A tesztgenerálási módszert egy komplex eszközkészlet részeként az 5. fejezet ismerteti. További részletek az ebben hivatkozott külön cikkben [29] találhatók.

[13] A cikk kétszerzős. Pintér Gergely végezte a monitor komponens részletes tervezését és megvalósítását.

[14] A cikk kétszerzős. Horányi Gergő végezte a monitor szintézis implementálását.

informatikában” könyv [26] „Állapottérképek” fejezetében is. Az UML állapottérképeket EHA formalizmusra leképző modelltranszformáció Varró Dániel, míg az EHA modell Promela nyelvre történő leképzését megvalósító eszköz Darvas Ádám munkája volt.

A modell alapú tesztgenerálási módszer részleteit külön publikáció tartalmazza [29]. A tesztgenerálás megvalósítását és a kapcsolódó méréseket Micskei Zoltán végezte.

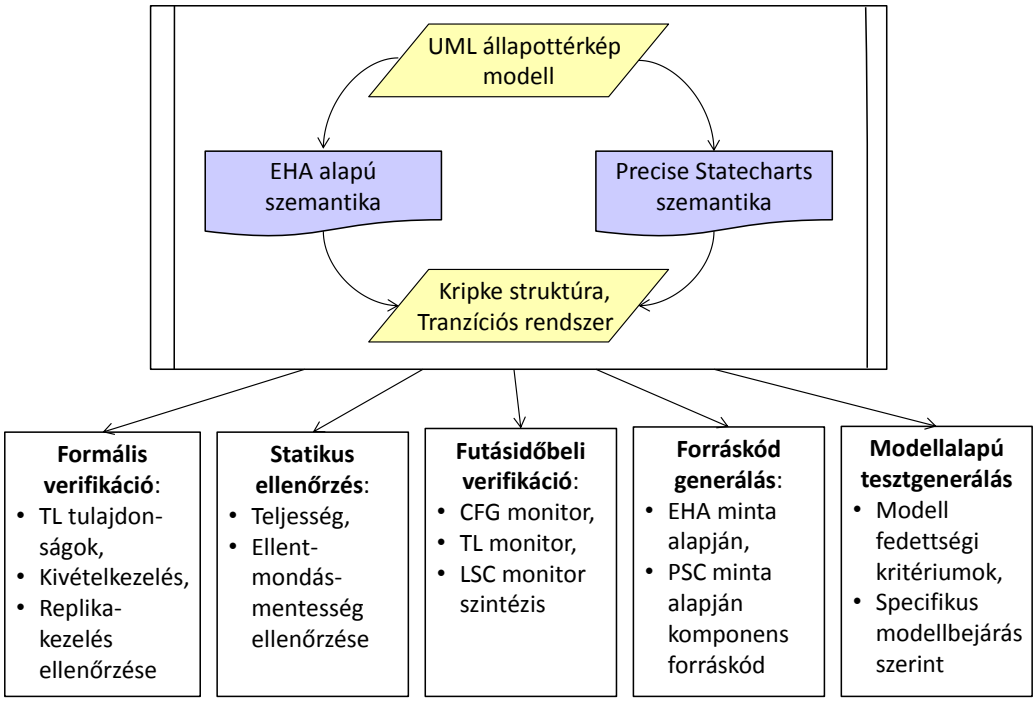
A monitorszintézis eszközök részletes tervezése és megvalósítása Pintér Gergely és Horányi Gergő munkája volt.

3.3.5. További kutatási eredmények

Témavezetésemmel és közreműködésemmel a 3. tézishez kapcsolódó további új kutatási eredmények is születettek, amelyek doktorandusz hallgatóimmal vagy tudományos diákköri hallgatóimmal együtt elért eredményekként PhD disszertációkba és TDK dolgozatokba kerültek.

1. **Statikus verifikáció az UML állapottérképek EHA szemantikája alapján:** A formális szemantikán alapuló modellellenőrzést alkalmaztuk UML állapottérkép specifikációk teljességi és ellentmondás-mentességi vizsgálatai során [30] [31] [32] [33].
2. **Modell alapú forráskód generálás:** Az UML állapottérképek EHA szemantikája alapján forráskód generálási módszereket javasoltunk [34] [35] [36].
3. **A formális szemantika bővítése:** Az UML állapottérkép szemantika formalizálását kibővítettük, a Precise Statecharts (PSC) formalizmussal lefedve az eddig nem kezelt konstrukciókat is [76]. A PSC formalizmust a fentiekhez hasonlóan alkalmaztuk formális verifikációra [37] és forráskód generálásra [38].
4. **Futásidőbeli verifikáció**: A temporális logika alapú monitorszintézis kutatás folytatásaként bevezettük az UML állapottérképek elemkészletét (eseményeket, akciókat, állapotátmeneteket) részletesen lefedő PSC-LTL lineáris temporális logika variánst, és ehhez hatékony futásidőbeli verifikációs algoritmust dolgoztunk ki [39] [40] [41].
5. **Modell alapú tesztgenerálás**: A modell alapú tesztgenerálás továbbfejlesztéseként megoldást adtunk és eszközláncot valósítottunk meg az időfüggő viselkedés ellenőrzéséhez szükséges tesztesetek automatikus generálására [42]. Ehhez időzített automaták alkalmazásával formalizáltuk az időfüggő viselkedés modellezésével kibővített UML állapottérképek szemantikáját.
6. **Formális módszerekre épülő automatikus eszközök**: A kidolgozott technológiák alapján modell alapú eszközöket fejlesztettünk, ezeket alkalmaztuk beágyazott vezérlők [43] és biztonságkritikus vasúti irányítástechnikai alkalmazások tervezésének támogatására [44].

Összességében a 3. tézisben szereplő, alapozó kutatások eredményeit, valamint azok itt felsorolt kiterjesztéseit alkalmazva többféle tervezési és ellenőrzési módszer és eszköz használata vált lehetővé (8. ábra).



8. ábra Az UML állapottérképekhez kapcsolódó módszerek és eszközök áttekintése

4. A tudományos eredmények hasznosítása

A tézisekben ismertetett tudományos eredményeket akadémiai és ipari partnerek közreműködésével futó kutatási és fejlesztési projektekben használtuk fel. Ezek közül kiemelhetők a következők:

- A „Safe Driver Machine Interface for ERTMS Automatic Train Control” (SAFEDMI) EU FP6 projekt[15] keretében az Ansaldo STS vezetésével egy SIL2 biztonságintegritási szintű mozdonyvezetői kezelőfelület fejlesztése történt. A tervezett architektúra megbízhatóságának és biztonságosságának modell alapú analízise az 1. tézisben ismertetett megbízhatóság modellezési módszerrel és eszközzel valósult meg [15] [2].
- A „Highly Dependable IP-based Networks and Services” (HIDENETS) EU FP6 projekt[16] keretében történt a holisztikus megközelítés kifejlesztése gépjárművek utasai által használható elosztott alkalmazások több szempontú analíziséhez [6]. Ehhez illeszkedett a 2. tézisben megadott megbízhatósági analízis módszer és eszköz, megvalósítva a felhasználói tevékenységek sikeres végrehajtásának vizsgálatát dinamikus környezetben. Ezt egy „ambulance warning” alkalmazás esetében mutattuk be [5].
- A „Keretrendszer nagymegbízhatóságú, biztonságkritikus rendszerek fejlesztéséhez és teszteléséhez” IKTA projekt[17] keretében többek között vezérlési folyamatok statikus ellenőrzése történt az UML állapottérképek EHA alapú formális szemantikája (3. tézis) alapján [33], vasúti irányítástechnikai és rádiós adatgyűjtő protokollok modelljein.

[15] FP6 IST STREP 31413, http://www.safedmi.org/

[16] FP6 IST STREP 26979. http://www.hidenets.aau.dk/

[17] IKTA 065/2000

- A „Kötöttpályás közlekedési rendszerek konstruktív, EU-konform biztonsági minősítése” GVOP projekt[18] keretében módszereket adtunk a modellalapú fejlesztés támogatására [44]. Ebben szerepet kapott az architektúra tervek modell alapú megbízhatósági analízise (1. tézis) valamint az UML állapottérképek alapján végzett formális verifikációs, monitor szintézis és tesztgenerálási módszerek (3. tézis). Mintakísérleteket végeztünk a Prolan Zrt által adott alkalmazások modellezésével.
- A „Model-based Generation of Tests for Dependable Embedded Systems” (MOGENTES) EU FP7 projekt[19] keretében UML állapottérképek formalizált szemantikája alapján az időfüggő viselkedés teszteléséhez szükséges tesztesetek automatikus generálását valósítottuk meg, a szükséges modell leképzési, modell-manipulációs és modellellenőrző eszközöket egy integrált eszközláncba illesztve [42]. A tesztgenerálást a Prolan Zrt által biztosított, vasúti objektumok állapotának dekódolását végző alkalmazáson végeztük, felmérve a módszer korlátait.

A kutatást ezeken kívül a következő kutatási projektek is támogatták és alkalmazták: „Assessing, Measuring and Benchmarking Resilience” (AMBER) EU FP7 projekt[20], „Robusztus objektum-orientált rendszerek optimális kialakítása” FKFP projekt[21], „Biztonsági követelmények formális ellenőrzése hibatűrő rendszerekben” OTKA projekt[22], „Objektum-orientált rendszerek megbízhatóságának vizsgálata” magyar-olasz kormányközi TéT együttműködési projekt[23], valamint „Hibatűrő rendszerek formális verifikációja és validációja” témában az MTA Bolyai János Kutatói Ösztöndíja.

A BME VIK mérnök-informatikus szak oktatásába kerültek a következő technológiák:

- Modell alapú megbízhatósági analízis a megbízhatósági modell moduláris összeállításával: Szolgáltatásbiztonságra tervezés tantárgy (mérnök informatikus MSc képzés, Szolgáltatásbiztos rendszertervezés szakirány).
- UML állapottérképek formális szemantikája alapján formális verifikáció, forráskód és monitorkód generálás: Formális módszerek tantárgy (mérnök informatikus MSc képzés, közös tárgy).
- Modell alapú teszteset generálás: Szoftverellenőrzési technikák tantárgy (mérnök informatikus MSc képzés, Szolgáltatásbiztos rendszertervezés szakirány), Szoftver verifikáció és validáció tantárgy (mérnök informatikus PhD képzés).

Az itt leírt eredményekre is épült Pap Zsigmond és Pintér Gergely sikeresen megvédett doktori disszertációja:

- Pap Zsigmond: Biztonságossági kritériumok ellenőrzése UML környezetben (2006)
- Pintér Gergely: Model Based Program Synthesis and Runtime Error Detection for Dependable Embedded Systems (2007)

[18] GVOP - 3.1.1 - 2004 - 05 - 0523/3.0

[19] EU FP7 ICT 216679, http://www.mogentes.eu/

[20] EU FP7 CA 216295, http://www.amber-project.eu/

[21] FKFP 0103/2001

[22] OTKA F-030553

[23] I-37/1999

5. A tézisekhez szorosan kötődő tíz publikáció

[1] Majzik I, Pataricza A, Bondavalli A, „Stochastic Dependability Analysis of System Architecture Based on UML Models”, In R. de Lemos, C. Gacek and A. Romanovsky (eds.): Architecting Dependable Systems, LECTURE NOTES IN COMPUTER SCIENCE 2677: pp. 219-244. (2003)

[2] I Majzik, A Bondavalli, S Klapka, T K Madsen, D Iovino, „Formal Methods in the Evaluation of a Safe Driver Machine Interface”, In: Tarnai G, Schnieder E (eds.) Formal Methods for Automation and Safety in Railway and Automotive Systems: Proceedings of Symposium FORMS/FORMAT 2008. Budapest, Hungary, 2008.10.09 - 2008.10.10. L'Harmattan, pp. 313-320. ISBN: 978-963-236-138-3 (2008)

[3] Majzik I, Huszerl G, „Towards Dependability Modeling of FT-CORBA Architectures”, LECTURE NOTES IN COMPUTER SCIENCE 2485: pp. 121-139. (2002), Proc. 4th European Dependable Computing Conference (EDCC-4), Toulouse, France, 2002.10.23-2002.10.25. (2002)

[4] Domokos P, Majzik I, „Design and Analysis of Fault Tolerant Architectures by Model Weaving”, In: C in M Dal, Bondavalli A, Suri N (eds.) Proc. Ninth IEEE Int. Symposium on High-Assurance Systems Engineering (HASE-05). Heidelberg, Germany, 2005.10.12-2005.10.14. IEEE Computer Society Press, pp. 15-24. ISBN: 0-7695-2377-3 (2005)

[5] M Kovács, P Lollini, I Majzik, A Bondavalli, „An Integrated Framework for the Dependability Evaluation of Distributed Mobile Applications”, In: Proceedings of the 2008 RISE/EFTS Joint International Workshop on Software Engineering for Resilient Systems (SERENE). Newcastle upon Tyne, United Kingdom, 2008.11.17-2008.11.19. ACM, pp. 29-38. ISBN: 978-1-60558-275-7 (2008)

[6] A Bondavalli, O. Hamouda, M. Kaâniche, P. Lollini, I. Majzik, H-P. Schwefel, „The HIDENETS Holistic Approach for the Analysis of Large Critical Mobile Systems”, IEEE TRANSACTIONS ON MOBILE COMPUTING 10:(6) pp. 783-796. (2011)

[7] Latella D, Majzik I, Massink M, „Automatic Verification of a Behavioural Subset of UML Statechart Diagrams Using the SPIN Model-checker”, FORMAL ASPECTS OF COMPUTING 11:(6) pp. 637-664. (1999)

[8] G Pintér, Z Micskei, I Majzik, „Supporting Design and Development of Safety Critical Applications by Model Based Tools”, ANNALES UNIVERSITATIS SCIENTIARUM BUDAPESTIENSIS DE ROLANDO EOTVOS NOMINATAE – SECTIO COMPUTATORICA 30: pp. 61-78. Paper 4. (2009)

[9] Pintér G, Majzik I, „High-level Supervision of Program Execution Based on Formal Specification”. In Proc. of the International Conference on Dependable Systems and Networks (DSN-2004), Architecting Dependable Systems, Firenze, Italy, 2004.06.28-2004.07.01. IEEE Computer Society Press, pp. 292-296. (2004)

[10] I Majzik, G Horányi, „Automated Code Synthesis for Run-Time Verification of Distributed Embedded Systems”, In: Jaan Penjam (eds.), Proc of The 12th Symposium on Programming Languages and Software Tools. Tallinn, Estonia, 2011.10.05-2011.10.07. Tallinn: pp. 161-172. ISBN: 978-9949-23-178-2 (2011)

6. A tézisekhez kapcsolódó további publikációk

6.1. Az 1. tézishez kapcsolódó publikációk

Megbízhatósági modellek konstrukciója UML architektúra modellek alapján:

[11] Bondavalli A, Dal Cin M, Latella D, Majzik I, Pataricza A, Savoia G, „Dependability Analysis in the Early Phases of UML-based System Design", COMPUTER SYSTEMS SCIENCE AND ENGINEERING 16:(5) pp. 265-275. (2001)

[12] Bondavalli A, Majzik I, Mura I, „Automated Dependability Analysis of UML Designs". In: Proc. 2nd IEEE International Symposium on Object-oriented Real-time Distributed Computing (ISORC'99). Saint-Malo, Franciaország, 1999.05.02-1999.05.05. pp. 139-144. ISBN: 0-7695-0207-5 (1999)

[13] Bondavalli A, Majzik I, Mura I, „Automatic Dependability Analysis for Supporting Design Decisions in UML". In: Proc. Fourth IEEE International Symposium on High-Assurance Systems Engineering (HASE'99). Washington, Amerikai Egyesült Államok, 1999.11.17-1999.11.19. Washington: pp. 64-71. ISBN: 0-7695-0418-3 (1999)

[14] Majzik I, Bondavalli A, „Automatic Dependability Modeling of Systems Described in UML". In: Chillarege R, Illgen Th (szerk.) Proc. 9th International Symposium on Software Reliability Engineering (ISSRE'98). Paderborn, Németország, 1998.11.04-1998.11.07. pp. 29-30. IEEE Computer Society ISBN: 3-00-003410-2 (1998)

A megbízhatósági modellezés alkalmazása a SAFEDMI projektben:

[15] A Bondavalli, A Ceccarelli, J Gronbaek, D Iovino, L Karna, S Klapka, T K Madsen, M Magyar, I Majzik, A Salzo, „Design and Evaluation of a Safe Driver Machine Interface", INTERNATIONAL JOURNAL OF PERFORMABILITY ENGINEERING 5:(2) pp. 153-166. (2009)

Aspektus-orientált modellezés és analízis:

[16] I Majzik, P Domokos, „Aspect-Oriented Modelling and Analysis of Information Systems". PERIODICA POLYTECHNICA-ELECTRICAL ENGINEERING 51:(1-2) pp. 21-31. (2007)

[17] Domokos P, Majzik I, „Automated Construction of Dependability Models by Aspect-Oriented Modeling and Model Transformation". In: Karl W, Becker J, Grosspietsch K E, Hochberger C, Maehle E (szerk.) Proc. 19th Int. Conf. on Architecture of Computing Systems: ARCS-06. Frankfurt am Main, Németország, 2006.03.13-2006.03.16. Gesellschaft für Informatik, pp. 66-75. (Lecture Notes in Informatics; P-81.) (2006)

[18] Majzik I, Domokos P, „Dependability Modeling Using Aspect Weaving Techniques". In: Cin M Dal, M Kaâniche, Pataricza A (szerk.), Dependable Computing - EDCC 2005: 5th European Dependable Computing Conference. Budapest, Magyarország, 2005.04.20-2005.04.22. Berlin ; Heidelberg: Springer, pp. 77-78. (Lecture Notes in Computer; 3463.) ISBN: 978-3-540-25723-3 (2005)

Eszközök fejlesztése megbízhatósági modellezéshez:

[19] Majzik I, Domokos P, Magyar M, „Tool-supported Dependability Evaluation of Redundant Architectures in Computer Based Control Systems", In: Schnieder E, Tarnai G (szerk.) FORMS/FORMAT 2007: Formal Methods for Automation and Safety in Railway and Automotive Systems. Braunschweig, Németország, 2007.01.25-2007.01.26. Braunschweig: GZVB eV, pp. 342-352. ISBN: 978-3-937655-09-03 (2007)

[20] M Magyar, I Majzik, „Modular Construction of Dependability Models from System Architecture Models: A Tool-Supported Approach”. In: Proc. 6th Int. Conf. on the Quantitative Evaluation of Systems (QEST 2009). Budapest, Magyarország, 2009.09.13-2009.09.16. pp. 95-96. Paper 14. ISBN: 978-0-7695-3808-2 (2009)

6.2. A 2. tézishez kapcsolódó publikációk

Modell alapú megbízhatósági analízis módszerek áttekintése:

[21] Andrea Bondavalli, Paolo Lollini, István Majzik, Leonardo Montecchi, „Modelling and Model-Based Assessment”. In: Wolter K, Avritzer A, Vieira M, van Moorsel A (szerk.), Resilience Assessment and Evaluation of Computing Systems. Berlin: Springer, 2012. pp. 153-165. ISBN: 978-3-642-29031-2 (2012)

Dinamikus környezetben végzett tevékenységek verifikációja teszteléssel:

[22] Z. Micskei, Z. Szatmári, J. Oláh, I. Majzik, „A Concept for Testing Robustness and Safety of the Context-Aware Behaviour of Autonomous Systems”. In Proc. 6th KES International Conference on Agent and Multi-Agent Systems, Technologies and Applications (KES-AMSTA 2012), Dubrovnik, Croatia, June 25-27, LNCS 7327, pp. 504-513, Springer Verlag, ISBN: 978-3-642-30946-5 (2012)

[23] G. Horányi, Z. Micskei, I. Majzik, „Scenario-based Automated Evaluation of Test Traces of Autonomous Systems”. In: Matthieu Roy (szerk.), ERCIM/EWICS Workshop on Dependable Embedded and Cyber-physical Systems at the 32nd Int. Conf. on Computer Safety, Reliability and Security (SAFECOMP 2013), Toulouse, France, September 24, pp. 181-192. ISBN: 2-907801-09-0

6.3. A 3. tézishez kapcsolódó publikációk

UML állapottérképek formális szemantikája:

[24] Latella D, Majzik I, Massink M, „Towards a Formal Operational Semantics of UML Statechart Diagrams”, In: Ciancarini P, Fantechi A, Gorrieri R (szerk.), Formal Methods for Open Object-Based Distributed Systems (Proc. Third IFIP International Conference on Formal Methods for Open Object-based Distributed Systems (FMOODS'99), February 15-18, 1999, Florence, Italy). Kluwer Academic Publishers, 1999. pp. 331-347. ISBN: 0-7923-8429-6 (1999)

[25] Gnesi S, Latella D, Majzik I, Massink M, „Formal Validation of UML Statechart Diagrams Models”, In: Reggio G (szerk.), UML2000 Workshop on Dynamic Behaviour in UML Models: Semantic Questions (The Third International Conference on The Unified Modeling Language). York, Egyesült Királyság, 2000.10.02-2000.10.03. Paper 9. (2000)

[26] Bartha T, Csertán Gy, Gyapay Sz, Majzik I, Pataricza A, Varró D, Pataricza A (szerk.), „Formális módszerek az informatikában”. Budapest: Typotex Kiadó, 2004. 312 p. (ISBN:963-9548-08-1) (2004)

Specifikus verifikációs módszerek

[27] Pinter G, Majzik I, „Modeling and analysis of exception handling by using UML statecharts”. LECTURE NOTES IN COMPUTER SCIENCE 3409: pp. 58-67. (2005) Proc. Scientific Engineering of Distributed Java Applications: 4th International Workshop, FIDJI 2004 (2004)

[28] Majzik I, Darvas Á, „Model Checking of Replication Management”. In: Giandomenico F Di (szerk.) Supplement of the EDCC-4 Conference - Fast Abstracts. Fourth European Dependable Computing Conference. Toulouse, Franciaország, 2002.10.23-2002.10.25. LAAS-CNRS, pp. 7-8. (2002)

Tesztgenerálás UML állapottérképek EHA szemantikája alapján:

[29] Micskei Z, Majzik I, „Model-based Automatic Test Generation for Event-driven Embedded Systems Using Model Checkers”. In: Ceballos S (szerk.), Proc. of Dependability of Computer Systems: DepCoS-RELCOMEX'06. Szklarska Poreba, Lengyelország, 2006.05.25-2006.05.27. Los Alamitos: IEEE Computer Society Press, pp. 191-198. ISBN: 0-7695-2565-2 (2006)

UML állapottérképek teljesség és ellentmondás-mentesség analízise az EHA szemantika alapján:

[30] Pap Z, Majzik I, Pataricza A, Szegi A, „Methods of Checking General Safety Criteria in UML Statechart Specifications”. RELIABILITY ENGINEERING & SYSTEM SAFETY 87:(1) pp. 89-107. (2005)

[31] Pap Zs, Majzik I, Pataricza A, „Checking General Safety Criteria on UML Statecharts”. In: Voges U (szerk.), Computer Safety, Reliability and Security: 20th International Conference, SAFECOMP 2001. Budapest, Magyarország, 2001.09.25-2001.09.27. Berlin ; Heidelberg: Springer, pp. 46-55. (Lecture Notes in Computer Science; 2187.) ISBN: 978-3-540-42607-3 (2001)

[32] Pap Zs, Majzik I, Pataricza A, Szegi A, „Completeness and Consistency Analysis of UML Statechart Specifications”. In: Hlavicka J, Renovell M, Pataricza A, Sziray J, Benyó B (szerk.), Proc. IEEE Design and Diagnostics of Electronic Circuits and Systems Workshop (DDECS 2001). Győr, Magyarország, 2001.04.18-2001.04.20. Győr: SZIF-Universitas, pp. 83-90. ISBN: 963-7175-16-4 (2001)

[33] Pataricza A, Majzik I, Huszerl G, Várnai Gy, „UML-based Design and Formal Analysis of a Safety-Critical Railway Control Software Module”. In: Tarnai G, Schnieder E (szerk.) Formal Methods for Railway Operation and Control Systems: FORMS 2003. Budapest, Magyarország, 2003.05.15-2003.05.16. Budapest: L'Harmattan Kiadó, pp. 125-132. ISBN: 963 9457 45 0 (2003)

Kódgenerálás UML állapottérképek EHA szemantikája alapján:

[34] Pintér G, Majzik I, „Impact of Statechart Implementation Techniques on The Effectiveness of Fault Detection Mechanisms”. In: Proceedings of The EUROMICRO'04 Workshop on Component Based Software Engineering. Rennes, Franciaország, 2004.08.31-2004.09.02. IEEE Computer Society Press, pp. 136-143. (2004)

[35] Pintér G, Majzik I, „Automatic Code Generation Based on Formally Analyzed UML Statechart Models”. In: Tarnai G, Schnieder E (szerk.), Formal Methods for Railway Operation and Control Systems: FORMS 2003. Budapest, Magyarország, 2003.05.15-2003.05.16. Budapest: L'Harmattan Kiadó, pp. 45-52. ISBN: 963 9457 45 0 (2003)

[36] Pintér G, Majzik I, „Program Code Generation Based on UML Statechart Models”. PERIODICA POLYTECHNICA-ELECTRICAL ENGINEERING 47:(3-4) pp. 187-204. (2003)

Precise Statecharts alapú verifikáció, kódgenerálás és monitor szintézis:

[37] Á Sisak, G Pintér, I Majzik, „Automated Verification of Complex Behavioral Models Using the SAL Model Checker”. In: Tarnai G, Schnieder E (szerk.) Formal Methods for Automation and Safety in Railway and Automotive Systems: Proceedings of Symposium FORMS/FORMAT 2008. Budapest, Magyarország, 2008.10.09-2008.10.10. Budapest: L'Harmattan Kiadó, pp. 35-42. ISBN: 978-963-236-138-3

[38] G Pinter, I Majzik, „Model Based Automatic Code Generation for Embedded Systems”. In: Varga K Attila, Vásárhelyi József, Samuelis Ladislav (szerk.), Proceedings of Regional Conference on Embedded and Ambient Systems: RCEAS 2007: selected papers. Budapest, Magyarország, 2007.11.22-2007.11.24. Budapest: Neumann János Számítógép-tudományi Társaság, pp. 97-106. ISBN: 978-963-8431-98-1 (2007)

[39] Pintér G, Majzik I, „Error Detection in Control Flow of Event-Driven State Based Applications”, In: Pelliccione P, Muccini H, Guelfi N, Romanovsky A (szerk.) Software Engineering of Fault Tolerant Systems. New Jersey: World Scientific Publishing Co., 2007. pp. 150-174. (SERIES ON SOFTWARE ENGINEERING AND KNOWLEDGE ENGINEERING; 19.) ISBN: 978-981-270-503-7 (2007)

[40] Pintér G, Majzik I, „Automatic Generation of Executable Assertions For Runtime Checking Temporal Requirements”, In: Proceedings of The 9th IEEE International Symposium on High Assurance Systems Engineering (HASE 2005). Heidelberg, Németország, 2005.10.12-2005.10.14. IEEE Computer Society Press, pp. 111-120. ISBN: 0-7695-2377-3 (2005)

[41] Pinter G, Majzik I, „Runtime verification of statechart implementations”. LECTURE NOTES IN COMPUTER SCIENCE 3549: pp. 148-172. (2005) Architecting Dependable Systems III (2005)

Tesztgenerálási módszerek továbbfejlesztése:

[42] B Polgár, I Ráth, I Majzik, „Model-based Integration Framework for Development and Testing Tool-Chains”. In: E Schnieder, G Tarnai (szerk.) FORMS/FORMAT 2010 - Formal Methods for Automation and Safety in Railway and Automotive Systems. Braunschweig, Németország, 2010.12.03-2010.12.04. Heidelberg: Springer, pp. 227-235.

Alkalmazás beágyazott és biztonságkritikus rendszerek tervezésében:

[43] Darvas A, Majzik I, Benyó B, „Verification of UML Statechart Models of Embedded Systems”, In: Straube B, Marinissen E J, Kotasek Z, Novak O, Hlavicka J, Ruzicka R (szerk.), Proc. IEEE Design and Diagnostics of Electronic Circuits and Systems Workshop (DDECS 2002). Brno, Csehország, 2002.04.17-2002.04.19. Brno University of Technology, IEEE Computer Society TTTC pp. 70-77. ISBN: 80-214-2094-4 (2002)

[44] I Majzik, Z Micskei, G Pintér, „Development of Model Based Tools to Support the Design of Railway Control Applications”, In: Saglietti F, Oster N (szerk.) Computer Safety, Reliability, and Security: Proceedings of the 26th International Conference on Computer Safety, Reliability and Security, SAFECOMP-2007. Nuremberg, Németország, 2007.09.18-2007.09.21. Berlin: Springer, pp. 430-435. (Lecture Notes in Computer Science; 4680.) ISBN: 978-3-540-75100-7 (2007)

7. Hivatkozások

[45] A. Avizienis, J-C. Laprie, B. Randell, C. Landwehr: Basic Concepts and Taxonomy of Dependable and Secure Computing. IEEE Transactions on Dependable and Secure Computing, Vol. 1. No. 1, pp 11-33, (2004)

[46] Goseva-Popstojanova, K., and K. S. Trivedi, „Architecture Based Software Reliability". In Proc. of the Int. Conf on Appplied Stochastic System Modeling (ASSM 2000), Kyoto, Japan (2000)

[47] Malhotra, M., and K. S. Trivedi, „Power-hierarchy among Dependability Model Types". IEEE Transactions on Reliability, Vol. 43, pp. 493-502 (1994).

[48] K. Trivedi, „Probability and Statistics with Reliability, Queuing, and Computer Science Applications", John Wiley and Sons, New York ISBN number 0-471-33341-7 (2001)

[49] Ajmone Marsan, M., G. Balbo and G. Conte: A Class of Generalized Stochastic Petri Nets for the Performance Analysis of Multiprocessor Systems. ACM TOCS, pp. 93-122 (1984)

[50] W.H. Sanders and J.F. Meyer, „Stochastic Activity Networks: Formal Definitions and Concepts", Lectures on Formal Methods and Performance Analysis, LNCS-2090, pp 315-343, Springer Verlag (2001)

[51] Laprie, J.-C. and K. Kanoun, „Software Reliability and System Reliability". In M. R. Lyu (editor), Handbook of Software Reliability Engineering, pp 27-69, McGraw Hill, New York (1995)

[52] Betous-Almeida, C., and K. Kanoun, „Dependability Evaluation - From Functional to Structural Modeling". In Proc. SAFECOMP 2001, pp 239-249, Springer Verlag (2001)

[53] Object Management Group: UML Profile For Modeling Quality Of Service And Fault Tolerance Characteristics And Mechanisms (QFTP). Version 1.0, http://www.omg.org/spec/QFTP/1.0/ (2006)

[54] C. Jones and B. Randell. "Dependable pervasive systems". Research Report CS-TR-839, University of Newcastle (2004)

[55] M. Kaâniche, P. Lollini, A. Bondavalli, K. Kanoun. "Modeling the Resilience of Large and Evolving Systems". Int. J. of Performability Engineering, 4(2), pp 153-168, (2008)

[56] J. Härri, F. Filali, and C. Bonnet, „Mobility Models for Vehicular Ad-Hoc Networks: A Survey and Taxonomy". In IEEE Communications Surveys & Tutorials, 11(4), (2009)

[57] E. M. Clarke, J. M. Wing, „Formal Methods: State of the Art and Future Directions", ACM Computing Surveys, Vol. 28, pp 626-643 (1996)

[58] P. Boca, J. P. Bowen, J. Siddiqi (eds.), „Formal Methods: State of the Art and New Directions", Springer Verlag, Berlin (2010)

[59] E. M. Clarke, O. Grumberg, D. Peled, „Model Checking". MIT Press, 1999.

[60] C. Betous-Almeida and K. Kanoun, "Construction and Stepwise Refinement of Dependability Models". Performance Evaluation, vol. 56, pp 277-306 (2004)

[61] Object Management Group: Fault tolerant CORBA. CORBA 2.6, Chapter 25 formal/01-12-63, OMG Technical Committee, http://www.omg.org (2001)

[62] T. Elrad, O. Aldawud, A. Bader, „Aspect-Oriented Modeling: Bridging the Gap between Implementation and Design”, In Proc. Generative Programming and Component Engineering (GPCE), Pittsburgh, Pennsylvania, pp. 189-201 (2002)

[63] J. Gray, T. Bapty, S. Neema, D. C. Schmidt, A. Gokhale, B. Natarajan, „An Approach for Supporting Aspect-Oriented Domain Modeling”. In Proc. Generative Programming and Component Engineering (GPCE 2003), Springer LNCS 2830, pp 151-168 (2003)

[64] Fajta R, Domokos P, Majzik I, „Adding High Availability Features to Server Applications Using Aspect-Oriented Programming”. HÍRADÁSTECHNIKA LXII:(1) pp. 56-62. (2007)

[65] I. Mura, A. Bondavalli, X. Zang and K. Trivedi, “Dependability Modelling and Evaluation of Phased Mission Systems: a DSPN Approach”. In Proc. 7th IFIP Int. Conference on Dependable Computing for Critical Applications (DCCA-7), San Jose, CA, USA, IEEE Computer Society (1999)

[66] D. Daly, D. D. Deavours, J. M. Doyle, P. G. Webster, and W. H. Sanders, „Möbius: An Extensible Tool for Performance and Dependability Modeling”. In Proc. 11th Int. Conf. TOOLS 2000, LNCS, pp 332-336, Springer, Berlin (2000)

[67] A. Bondavalli, I. Mura, S. Chiaradonna, R. Filippi, S. Poli, F. Sandrini, „DEEM: a Tool for the Dependability Modeling and Evaluation of Multiple Phased Systems”. In Proc. Int. Conf. on Dependable Systems and Networks (DNS), pp 231-236, IEEE CS (2000)

[68] E. Mikk, Y. Lakhnech, M. Siegel, „Hierarchical Automata as Model for Statecharts”. In Proc. 3rd Asian Computing Science Conference, LNCS-1345, pp 181-196, Springer, Berlin (1997)

[69] G. Holzmann, „The model checker SPIN”. IEEE Trans. on Software Engineering, 23(5), pp 279-295 (1997)

[70] A. Gargantini, C. Heitmeyer, „Using Model Checking to Generate Tests from Requirements Specifications”. In Proc. 7th European Software Engineering Conference, pp 146-162, Springer (1999)

[71] H. Seok Hong, I. Lee, O. Sokolsky, S. Deok Cha, “Automatic Test Generation from Statecharts Using Model Checking”. MS-CIS-01-07, University of Pennsylvania (2001)

[72] Majzik István, „Concurrent Error Detection in Multiprocessor Systems using Watchdog Processors”, PhD disszertáció, Budapesti Műszaki Egyetem, Villamosmérnöki és Informatikai Kar (1997).

[73] G. Behrmann, A. David, K. G. Larsen, J. Hakansson, P. Petterson, W. Yi, and M. Hendriks, „UPPAAL 4.0”. In Proc. of the 3rd Int. Conf. on the Quantitative Evaluation of Systems, pp 125-126,Washington, DC, USA, IEEE Computer Society (2006)

[74] W. Damm and D. Harel. LSCs: Breathing Life into Message Sequence Charts. In Formal Methods in System Design Vol. 19 pp 45-80, (2001)

[75] K. Havelund and G. Rosu, „Testing Linear Temporal Logic Formulae on Finite Execution Traces”. Tech. Rep. 20010091017, NASA Ames Research Center (2001)

[76] Pintér Gergely, „Model Based Program Synthesis and Runtime Error Detection for Dependable Embedded Systems”. PhD disszertáció, Budapesti Műszaki és Gazdaságtudományi Egyetem, Villamosmérnöki és Informatikai Kar (2007).