Közúti fedezőjelzőrendszer formális modellezése UPPAAL keretrendszer felhasználásával

Formal modelling of level crossing system for trams using UPPAAL framework

LUKÁCS Gábor[1], dr. BARTHA Tamás[2]

Budapesti Műszaki és Gazdaságtudományi Egyetem, Közlekedés- és Járműirányítási Tanszék
Magyarország, 1111. Budapest, Stoczek u. 2. St. ép. 114.
http://kjit.bme.hu/index.php/hu/
[1] +36303046472, lukacs.gabor@edu.bme.hu
[2] +3612796227, bartha.tamas@kjk.bme.hu

ABSTRACT

Development based on systems modelling is a widely used technique in the engineering practice. Formal modelling and verification is an effective set of tools that combines modelling with mathematical precision. In this paper, we present the formal modelling of an urban railway safety-critical system using the UPPAAL framework.

Keywords: urban railway, modelling, functionality, formal methods, timed automata

KIVONAT

A rendszerek modellezésén alapuló fejlesztés széles körben használt technika a mérnöki gyakorlatban. A formális modellezés és verifikáció egy hatékony eszközkészlet, amely a modellezést kombinálja a matematikai precizitással. E cikkben egy városi vasúti biztonságkritikus rendszer formális modellezését mutatjuk be az UPPAAL keretrendszer felhasználásával.

Kulcsszavak: városi vasút, modellezés, funkcionalitás, formális módszerek, időzített automaták

1. BEVEZETÉS

A városi vasutak biztonságával kapcsolatos társadalmi elvárások a népsűrűség növekedésével együtt nőnek. A városi közlekedésben számos különböző lehetséges műszaki megoldás alkalmazható a közlekedés biztonságának biztosítására, ezért a biztonságos közlekedést lehetővé tevő rendszerek országonként eltérhetnek. Az eltérések többek között a városi közösségi közlekedési kultúra sok éven át kialakult különbségeiből adódnak. Például sok város rendelkezik olyan közúti vasúti (villamos) hálózattal, amelyet a jelenlegi városszerkezet miatt már lehetetlen elkülöníteni a közúti személygépjármű forgalomtól. Ezért szükségszerűen léteznek közút-villamos szintbeni kereszteződések, ahol gyakran történnek balesetek. A balesetek és következményeik (személyi sérülések és anyagi károk) megelőzése céljából számos műszaki megoldást és közlekedési szabályt alkalmaznak.

Ebben a tanulmányban a biztonság igazolására egy formális modellezésen alapuló módszertan alkalmazhatóságát mutatjuk be egy esettanulmányon keresztül, ami egy Magyarországon alkalmazott villamos-közút szintbeni kereszteződést biztosító rendszer. A javasolt eljárás egy specifikációs-verifikációs környezetet biztosít a szakterületi mérnökök számára. Ez a keretrendszer egy hatékony megoldást kínál a rendszerfejlesztés során a rendszer funkcionalitásának leírására a helyesség, teljesség, konzisztencia és ellenőrizhetőség szempontjait figyelembe véve. Az általunk javasolt formális modellezés alapú módszertan használatával olyan ellenőrzött funkcionális specifikáció készíthető, amely kevesebb hibát fog tartalmazni a hagyományos fejlesztésben alkalmazott technikákhoz képest. A módszertan kiemelendő előnye, hogy a formális módszerek alkalmazásához minimális többlet erőforrás ráfordítást igényel a szakterületi fejlesztőmérnököktől, amit a szükséges matematikai háttérismeretek elrejtésével érünk el.

2. KAPCSOLÓDÓ KUTATÁSI EREDMÉNYEK

Ebben a fejezetben áttekintést adunk azokról a kutatási eredményekről és esettanulmányokról, amelyek a jelen írásunkban megfogalmazott célokhoz szorosan kapcsolódnak. Ezek az eredmények általánosságban abban különböznek az általunk javasolt módszertantól, hogy szerzőik más alkalmazási területet céloztak meg vagy eltérő célokra összpontosítottak, esetleg más eszközöket alkalmaztak a biztonságkritikus rendszerek specifikációjának elkészítéséhez és ellenőrzéséhez.

Az [1] tanulmány egy módszertant javasol a rendszerek modellezésére és ellenőrzésére, amelynek alkalmazásához az Unified Modelling Language (UML [2]) és az UPPAAL [3] (amely egy formális modellezési keretrendszer) eszközöket ajánlják. A módszertan igazolása érdekében a szerzők különböző esettanulmányokat mutatnak be: egy kritikus, 5G-vel támogatott robotsebészeti egészségügyi alkalmazást és egy nem kritikus video-streaming alkalmazást. A tanulmány konklúziója, hogy a mérnöki gyakorlatban széles körben használt UML és UPPAAL előnyeinek ötvözésével hatékony megoldást kapunk a modellezett probléma kezelésére azáltal, hogy a formális ellenőrzést már a fejlesztési életciklus korai szakaszában is elvégezhetjük. A tanulmányban használt két eszköz megegyezik az általunk javasolt módszertan eszközeivel. A tanulmány és az általunk javasolt módszertan (lásd 3. fejezet) közti különbséget az jelenti, hogy kutatásunk során figyelembe vettük a vasúti szakterületre jellemző szempontokat, és a módszertanunkat ezek alapján dolgoztuk ki.

A [4] értekezés kutatási célja az volt, hogy elemezze a formális módszerek alkalmazhatóságát az ipari irányítórendszerek (PLC, programozható logika vezérlők) területén és javaslatot tegyen egy specifikációs-verifikációs környezetre. A szerző a modellellenőrzést javasolta az ipari irányítórendszerek szoftverfejlesztésének támogatására, aminek a fő kihívásait a teljesítmény és a használhatóság jelentette. Az értekezésben megadott keretrendszer a formális részletek, a matematikai háttér elrejtésével támogatja a szakterületi mérnököket, hasonlóan az általunk javasolt módszertanhoz (lásd 3. fejezet). Az értekezésben kidolgozott esettanulmányok egy speciális szakterületről származtak (akár az általunk megcélzott speciális vasúti környezet), amit a CERN (Organization for Nuclear Research) támogatott.

A [5] tanulmány a modellellenőrzés alkalmazását mutatja be vasúti biztosítóberendezési alkalmazásokon. A szerzők a biztosítóberendezés modellezést és modellellenőrzését időzített automaták segítségével végezték el UPPAAL keretrendszer felhasználásával. A modellellenőrzéshez szükséges biztonsági funkciók specifikációját CTL (Computational Tree Logic) formulákkal adták meg. Az [5] kutatás és az általunk javasolt módszertan (lásd 3. fejezet) közti különbség, hogy az [5] tanulmány a rendszer szintű biztonsági funkciókra fókuszált, míg az általunk javasolt módszertan a komponens szintű modellezést és modellellenőrzést helyezi előtérbe.

A [6] kutatási munka során a szerzők egy általános biztosítóberendezés funkcionalitásának modellezését adják meg többlépcsős finomítást alkalmazva, az Event-B tételbizonyítás felhasználásával. A kutatás célja az volt, hogy a fejlesztőket támogassák a lehetséges specifikációs hibák azonosításában az életciklus korai szakaszában. A szerzők javaslatot tesznek a modellellenőrzés és a tételbizonyítás egymást kiegészítő gyakorlati alkalmazására. A [6] tanulmány és az általunk javasolt módszertan (lásd 3. fejezet) közti hasonlóság az azonos vasúti szakterület, míg a különbséget az alkalmazott technikák (modellellenőrzés, illetve tételbizonyítás) jelentik.

3. MÓDSZERTAN

Ebben a fejezetben áttekintjük az általunk javasolt módszertan hátterét, alapelveit, az alkalmazott technikákat és eszközöket.

3.1. Háttér

Az életciklusnak nagyon sokféle megközelítése létezik a különböző szakterületeken. Például a vasúti biztonságkritikus rendszerek esetében a [7] szabvány által ajánlott életciklus modell, közismert nevén a „V-modell” alkalmazható. Az általunk javasolt módszertan a V-modell tervezési szakaszát fedi le a követelményektől a részletes tervekig, beleértve az ezekhez tartozó verifikációs tevékenységeket is.

A [7] szerinti életciklus alapján a követelményekkel kapcsolatos feladatok két életciklus szakaszra („Rendszerkövetelmény specifikáció” és „Architektúra és rendszer követelmények allokálása”) korlátozódnak, melyeket kiegészítenek a verifikációs és validációs feladatok. A projektmenedzsment és a rendszertervezés legújabb eredményei azonban felhívják a figyelmet arra, hogy a követelményekkel kapcsolatos tevékenységek nem korlátozódhatnak kizárólag ezekre a fázisokra. A követelmények tervezése (RE, Requirement Engineering) olyan tevékenységek összességét jelenti, amelyek célja a kifejlesztendő rendszer céljainak, képességeinek, a korlátoknak és előfeltételeknek a feltárása, értékelése és dokumentálása [8]. A követelmények kezelése (RM, Requirement Management) [9] – mint egy a teljes életciklust átfogó technika – magában foglal olyan

folyamatokat, mint a követelmények követése, rangsorolása, ellenőrzése, fenntartása stb. Napjainkban számos széles körben használt módszertan és módszer áll rendelkezésre az RE és az RM támogatására különböző eszközökkel vagy keretrendszerekkel (pl. [10]).

A modell-alapú rendszertervezés (MBSE, Model-Based Systems Engineering [11]) és a modell-vezérelt fejlesztés (MDD, Model-Driven Development [12]) népszerű megközelítések a jelenlegi mérnöki gyakorlatban. Az MBSE és MDD támogatására számos eszköz (pl. MATLAB [13.], Simulink [14], stb.) létezik. Egy modell egy adott probléma célszerű absztrakciója, amely világos, jól kezelhető képet ad a problémához tartozó kérdések megválaszolásához. A modell a probléma egyszerűsített leírása, amelyben elhanyagoljuk azokat a részleteket, amelyek nincsenek hatással a kérdések megválaszolására vagy a kitűzött modellezési cél elérésére. Egy modellnek számos megjelenési formája lehet, pl. szöveges, grafikus vagy vegyes. A vasúti mérnöki gyakorlatban a grafikus megjelenítés a legnépszerűbb, néha azonban szöveges leírásokra is szükség lehet. A vasúti területen számos MBSE-hez kapcsolódó technikát és eszközt használnak a modellezés támogatására (pl. UML [2] vagy SysML [15]). A modellezés az EN 50128 [16] szabvány ajánlásai között is szerepel (lásd A.17 táblázat), beleértve a formális módszereket is.

A formális módszerek [17] olyan modellezési és elemzési technikák, amelyeket korábban elsősorban az informatika területén alkalmaztak. Ezek a technikák a matematikán, különösen a diszkrét matematikán és a matematikai logikán alapulnak. A formális modellek szemantikája és szintaxisa jól meghatározott és teljes, ezáltal arra motiválják a mérnököket, hogy mélyen és szisztematikusan gondolkodjanak el egy problémáról, így jelentősen csökkenthető a specifikációs hibák, hiányosságok és kétértelműségek száma. Napjainkban a vasúti biztosítóberendezések tervezési gyakorlatát a nem formális vagy félformális (szöveges és esetleges jelölési rendszereket tartalmazó) specifikációk jellemzik. Ezek gyakran a fejlesztési folyamat résztvevői közötti félreértésekhez, bizonytalanságokhoz vagy mulasztásokhoz vezetnek. A formális módszerek használata a fejlesztési folyamat során csökkentheti ezek valószínűségét. A modellellenőrzés [5] olyan formális módszer, melynek alkalmazása során a szakterületi mérnöki tudást egy átlátható szemantikai modellel írják le. A modellellenőrzés során azt ellenőrzik, hogy a megadott formulák teljesülnek-e a létrehozott modellen. A modellellenőrzés közelebb áll a rendszermérnöki gyakorlathoz, mint a tételbizonyítás [6], mivel támogatja az úgynevezett ellenpélda létrejöttét egy-egy követelmény nem teljesülése esetén, aminek a segítségével a szakterületi mérnökök kijavíthatják a specifikáció és tervezés során vétett hibákat. Az általunk javasolt módszertan (lásd 3.2. fejezet) kidolgozása során figyelembe vettük a modellellenőrzés előbbiekben tárgyalt előnyeit.

3.2. Az FMBRSE módszertan

A 3.1 fejezetben leírt háttérismeretekre alapozva létrehoztunk egy módszertant, amelyre a következőkben FMBRSE (Formal Model-Based Railway Safety Engineering) néven hivatkozunk. Az FMBRSE célja, hogy támogassa a vasúti szakterületi mérnököket a formális specifikáció és verifikáció alkalmazásában a biztonságkritikus vasúti rendszerek fejlesztése során. A módszertan elvárt eredményeként a rendszer/komponensek formálisan ellenőrzött funkcionális modelljét kaphatjuk meg. A FMBRSE alkalmazásakor végrehajtandó folyamat lépései a következők:

1. A folyamat bemenete: a megrendelői követelményspecifikáció a fejlesztendő rendszerhez (előfeltételezés, hogy létezik ilyen specifikáció; ha nem létezik, akkor ezt a lépést is végre kell hajtani)
2. Rendszerkövetelmény specifikáció
    a. Funkcionális követelmény specifikáció
        i. Funkcionális követelmények transzformációja CTL formulákra
3. Architektúra tervezés és követelmények allokálása
4. Rendszer/Komponensek tervezése (interfészek, paraméterek, viselkedés)
    a. Rendszer/Komponensek formális modellje
5. Modellellenőrzés (melynek bemeneteként 2/a/i. és 4/a. pontok kimenete szolgál)
6. A folyamat kimenete: verifikációs jelentés a modellellenőrzés eredményéről.

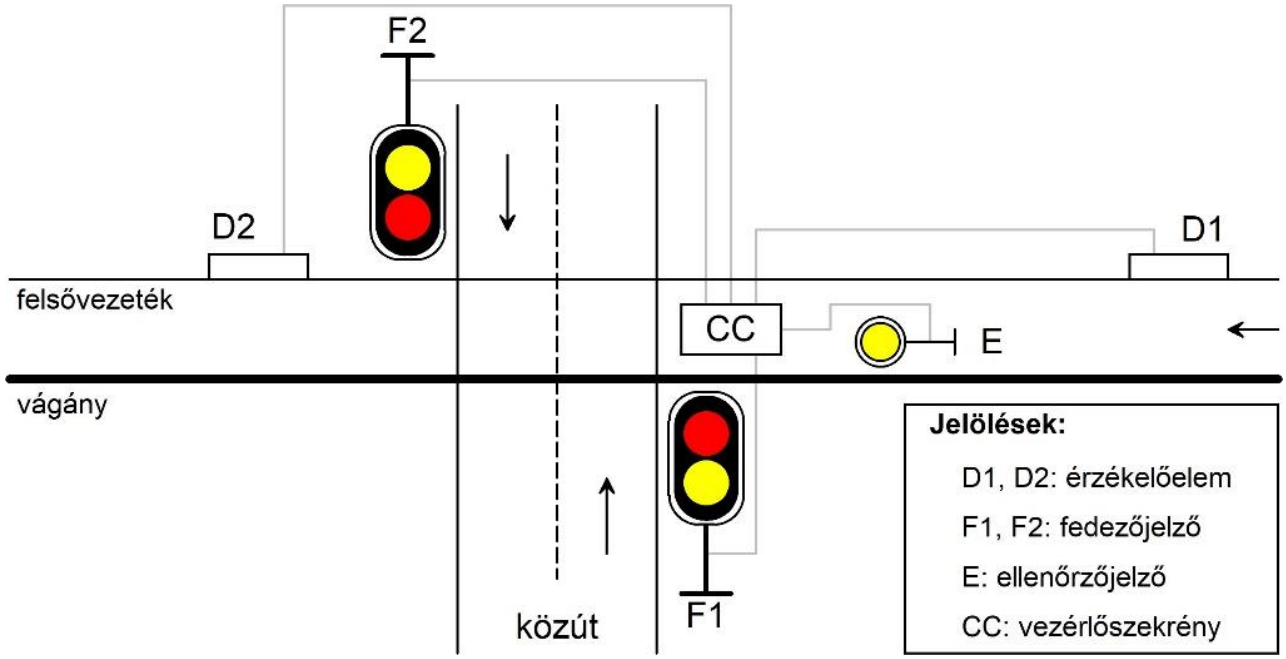
Az FMBRSE módszertan a tulajdonságait tekintve egy terv-vezérelt/prediktív [18] módszertan, ami azt jelenti, hogy a jövőbeli tevékenységek megtervezése a lehető legrészletesebben előre megtörténik. Az FMBRSE egy hibrid módszertan, amely több már létező megközelítést ötvöz, melyek közül a legfontosabbak a strukturált és objektumorientált tervezési filozófiák [19]. Az FMBRSE alkalmazza az MDD és a FBD (Function-Based Development, [20]) már elért eredményeit. A módszertan tervezési alapelve a „top-down”, hierarchikus dekompozícióval [21] és iterációkkal. Az FMBRSE módszertan életciklus [7] fedettsége részleges, csak a tervezési szakaszra terjed ki, pontosabban a követelményektől a részletes tervezés szakaszáig. A FMBRSE már meglévő technikákat és eszközöket integrál (pl. UPPAAL).

4. ESETTANULMÁNY

A közúti-vasúti átjárók biztonsága kiemelt fontosságú a magyarországi villamosüzemeltetők számára. Ezekben a közút-vasút kereszteződésekben két kockázatcsökkentő műszaki megoldás jellemző Magyarországon: forgalomirányító berendezések (összetett csomópontok esetén) és villamos-közúti vasúti átjáró (TRLC, TRam Level Crossing) rendszer (egyszerű csomópontok esetében). Ez utóbbi megoldás előnye a költséghatékonyságban rejlik.

A TRLC rendszerek biztonsága érdekében a BKV (Budapesti Közlekedési Vállalat) kidolgozott egy követelményfüzetet [22] (lásd 3.2. fejezet, FMBRSE módszertan 1. lépése). Ez a követelményfüzet képezi a bemutatott esettanulmány alapját.

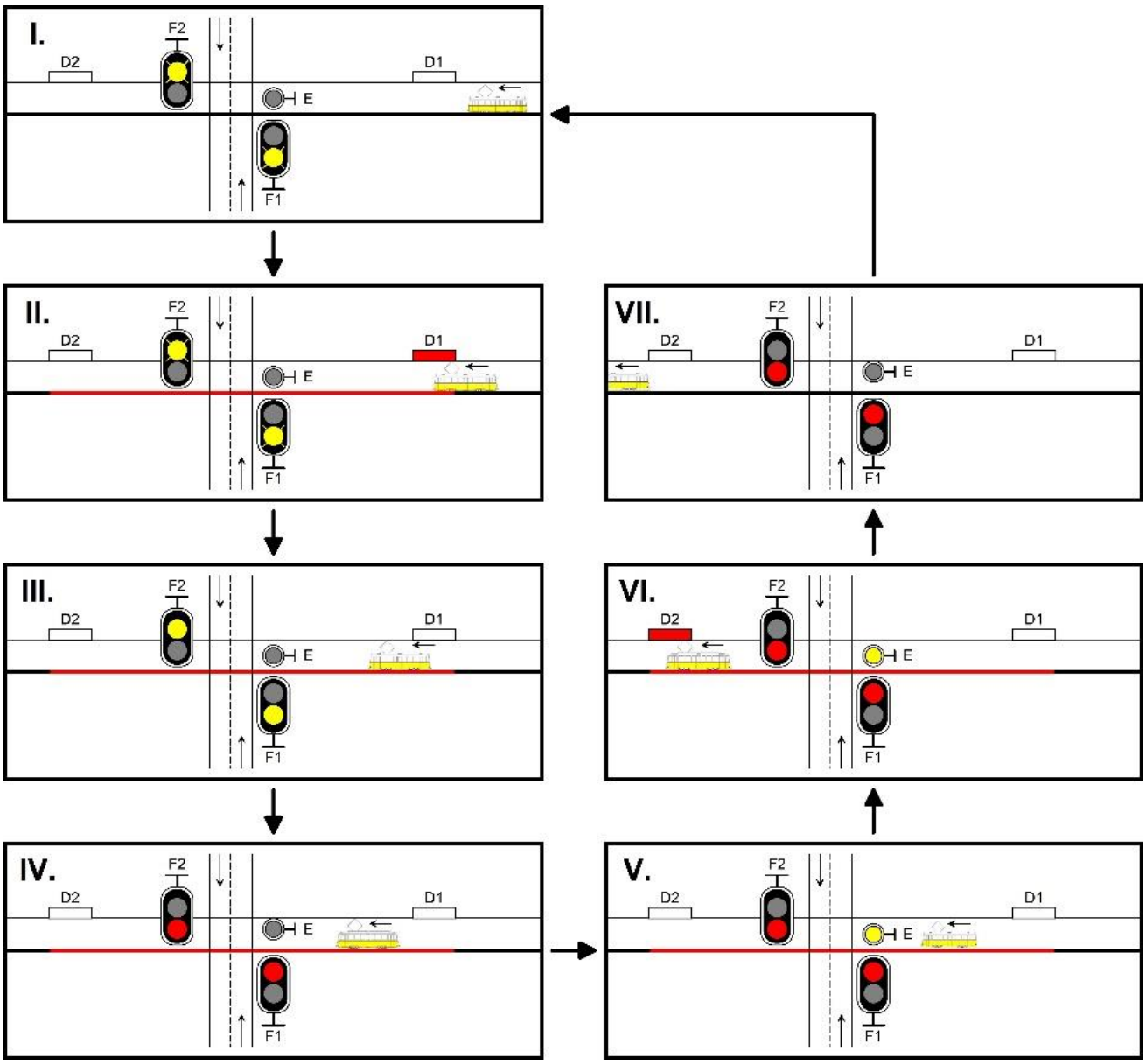
Az 1. ábra egy lehetséges konfigurációt mutat be a TRLC rendszer alkalmazására egy egyszerű, egyvágányos helyszínen. Ez a rendszer két érzékelőelemből (D1 és D2) áll. Ezeknek a feladata, hogy felismerjék a villamos jelenlétét a rendszer hatókörében. A közúti forgalmat fedezőjelzők (F1 és F2), míg a villamosközlekedést egy ellenőrzőjelző (E) szabályozza. A rendszer elemei a központi vezérlőszekrénnyel (CC) állnak kapcsolatban.



1. ábra.

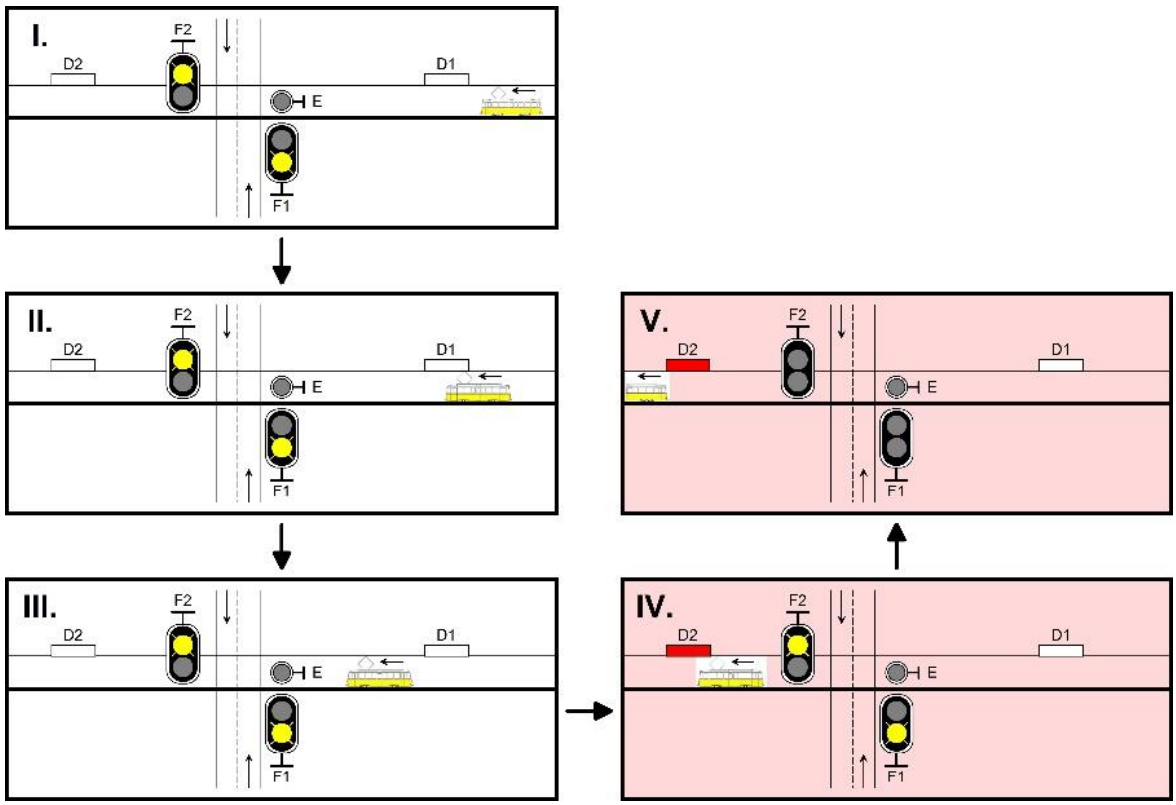
Közúti fedezőjelző rendszer (egy lehetséges konfiguráció)

A rendszer alapállapotában (nincs villamos a rendszer hatókörében) a fedezőjelzők sárgán villognak, az ellenőrzőjelző pedig sötét. Amikor egy villamos érkezik a D1 érzékelőelem hatókörébe, a fedezőjelzők folyamatos sárga jelzési képre váltanak. Ez az állapot jól meghatározott, rövid ideig (4-10 másodpercig) tart. Ezt követően a fedezőjelzők piros jelzési képre váltanak, aminek a hatására a közúti gépjárműforgalomnak meg kell állnia a hatályos közlekedési szabályok [23] értelmében. A fedezőjelzők piros jelzési képét követően az ellenőrzőjelző sárga jelzési képre vált. Ez a villamos jármű vezetője számára azt jelenti, hogy a közúti forgalom számára a „tilos az áthaladás” jelzési képek sikeresen megjelentek a fedezőjelzőkön. Így a villamos a nemzeti szabályok által megengedett legnagyobb sebességgel lépheti át a kereszteződést (nem szükséges fékeznie/megállnia). Amikor a villamos elhagyta a rendszert a D2 érzékelőelemen át, a berendezés alapállapotba kerül. A leírt folyamatot lépésről lépésre nyomon követhetjük a 2. ábrán.



2. ábra.
Közúti fedezőjelző rendszer működése

Az 1. ábra a közúti fedezőjelző rendszer hibamentes állapot melletti működését mutatja be. A rendszerhez azonban számos meghibásodás kapcsolódhat. A 3. ábrán a rendszer egyik hibamódját ismertetjük: a D1 érzékelőelemen a villamos jelenlétének érzékelése valamilyen oknál fogva nem valósul meg. Ebben az esetben a villamos úgy fog áthaladni a rendszeren, hogy az ellenőrzőjelzőn (E) nem jelenik meg a villamosvezető számára az áthaladást engedélyező jelzési kép (a jelző sötét marad, mivel F1 és F2 jelzőn nem jelent meg a piros jelzési kép). A villamos jármű rendszerből való távozását követően (D2 érzékelőelem érintésére) a TRLC zavarállapotba kerül és üzenetet küld a hibáról az Üzemeltetőnek, hogy ellenőrizze a rendszert. Az Üzemeltető ezt követően hibaelhárítást kezdeményez az adott helyszínen (pl. D1 jelű érzékelőelem cseréje).



3. ábra.
Példa közúti fedezőjelző rendszer viselkedésére egy lehetséges hibamód esetén

Jelen írásunkban 3.2. fejezetben ismertetett FMBRSE módszertanhoz definiált folyamatból csak az 4/a. lépést ismertetjük a bemutatott esettanulmány felhasználásával. A formális modellek előállítása/előállíthatósága a módszertan egyik kulcskérdése, (mivel a FMBRSE 2/a/i. és 4/a. lépései a hagyományos fejlesztési folyamatban nem találhatók meg.) A modellezés bemeneteként [22] füzetben megadott követelményrendszer szolgált. Megjegyezzük továbbá, hogy TRLC rendszer meghibásodási módjai közül csak a 3. ábrán bemutatott meghibásodási módot és három további meghibásodási esetet (az egyik vagy mindkét piros optika egyidejű kiégése a fedezőjelzőkön) modelleztünk (lásd 5. fejezet). E néhány meghibásodás modellezése jól illusztrálja a hibák figyelembevételét a viselkedés formalizálásakor. Ugyanakkor a lehetséges hibamódok összeségének feltárása és modellezése nem volt célja jelen írásunkban bemutatott esettanulmánynak.

5. RENDSZERMODELLEZÉS

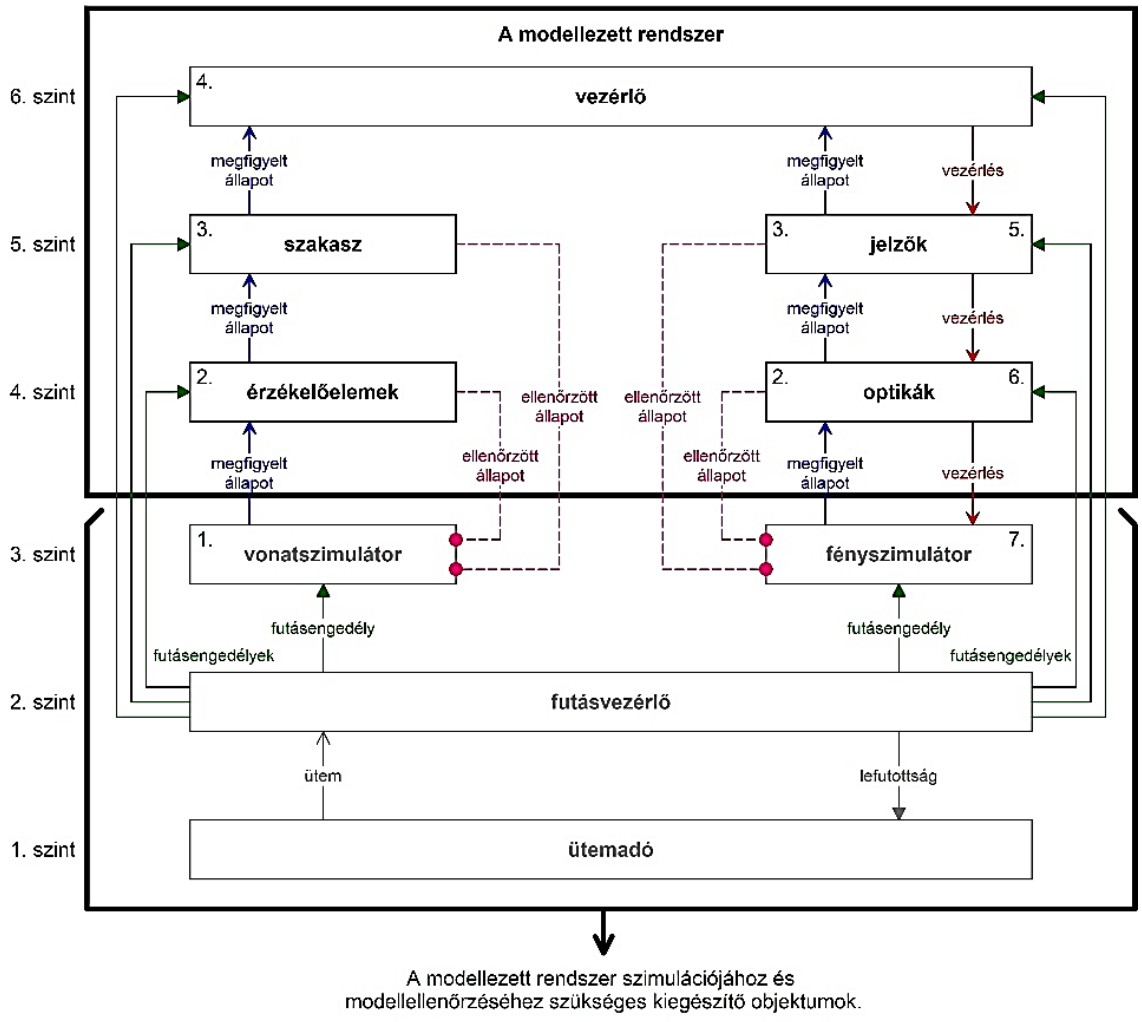
A 4. fejezetben leírt esettanulmányt UPPAAL keretrendszerben modelleztük. A modellezéshez az UPPAAL 4.2.24-es verzióját használtuk. A modellezés eredményét ebben a fejezetben ismertetjük.

5.1. A közúti fedezőjelző rendszer formális modelljének felépítése, mechanizmusai

A modell felépítését és a felhasznált mechanizmusokat a 4. ábra mutatja be. Az alkalmazott jelölések a következők:

- **téglalap:** automata sablon, ami egy komponenst modellez (a diagram a sablon egyes példányait külön nem jelöli, amelyik komponens neve többesszámban van, azon komponensek a modellezett rendszerben több példányban is léteznek)
  - **fekete téglalap:** a modellezett rendszerhez tartozó automaták (a követelményrendszerből leszármaztatható komponensek),
  - **szürke téglalap:** a modellezett rendszer szimulációjához (végrehajtásához) és modellellenőrzéséhez szükséges kiegészítő komponensek,
- **kék nyilak („megfigyelt állapotok”):** a nyíl minden esetben a megfigyelt automata felől a megfigyelő automata felé mutat; ezek az interfészek jelképezik a megfigyelő automaták számára átadott bemenet jellegű állapot információkat (ellenőrzési bemeneteket), amely bemenetek állapota az adott automata viselkedését befolyásolhatja,

- **piros nyilak („vezérlés”):** a nyíl minden esetben a vezérlő automata felől a vezérelt automata felé mutat; ezek az interfészek jelképezik a vezérelt automaták számára átadott vezérlési bemeneti információkat (vezérlési bemeneteket), amely bemenetek állapota az adott automata viselkedését befolyásolhatja,
- **zöld nyilak („futásengedélyek”):** a nyíl minden esetben az engedélyt adó (engedélyező) automata felől az engedélyt megkapó automata felé mutat; ezeken az interfészeken keresztül kapják meg az egyes automaták az engedélyt a „futásvezérlőtől” (5.4.2. fejezet) a futásra (azaz az állapotuk esetleges megváltoztatására; az állapotváltás a megfigyelt bemenetektől függhet),
- **rózsaszín nyilak („ellenőrzött állapot”):** a modellezett rendszer szimulációját és modellellenőrzését támogató automaták számára a modellezett rendszer automatáinak állapotát nyomon követő interfészek. A nyilak hasonló értelműek, mint a kék nyilak, csak ebben az esetben a megfigyelt automaták a modellezett rendszer automatái, míg a megfigyelő automaták a szimulációt és modellellenőrzést támogató automaták,
- **szürke nyilak („ütem” és „lefutottság”):** az „ütemadó” által adott és fogadott interfészek (lásd 4.4.1. fejezet).



4. ábra.
Közúti fedezőjelző rendszer modell-architektúra

A modell hierarchikus felépítésű, összesen 6 hierarchia szinttel jellemezhető. Az egyes szinteken szereplő automaták leolvashatók a 4. ábráról:

1. szint: ütemadó,
2. szint: futásvezérlő,
3. szint: vonatszimulátor és fényszimulátor,

4. szint: érzékelőelemek és optikák,
5. szint: szakasz és jelzők,
6. szint: vezérlő.

A modell időzített ütem (ciklikus) alapon működik. Az ütemet az ütemadó (5.4.1. fejezet) adja. (Egy ütem hossza lehet pl. 100 ms, de bármilyen tetszőleges hosszúságú, nemnegatív ütemhossz is elképzelhető, azonban a modell paramétereit a kiválasztott ütem függvényében szükséges beállítani.) Az ütemadó által kiadott ütem alapján a futásvezérlő (lásd 5.4.2. fejezet) vezérli/engedélyezi az egyes automaták „futását". Egy automata egy ciklusa alatt azt értjük, hogy fogadja, majd kiértékeli a bemeneteit, és ez alapján eldönti, hogy szükséges-e állapotot váltania és megváltoztatnia a kimeneteit. A modell működésének alapelve, hogy ha valamelyik automata futásengedélyt kap, az az automata mindenképpen végre fog hajtani egy élátmenetet még abban az esetben is, hogy ha ez nem jár az állapotának a megváltoztatásával. A futásvezérlő a következő sorrendben vezérli az egyes automaták futását (a vezérlési sorrend a 4. ábra téglalapjaiban szereplő arab számok segítségével is nyomon követhető):

1. vonatszimulátor (3. szint): célja, hogy modellezze az érzékelőelemek fizikai bemenetét, a vonatszimulátor mindig az adott ütem elején frissíti a saját állapotát,

   Megjegyzés: a vonatszimulátor az adott ütem elején állítja be a valós fizikai érzékelőelemek állapotát (ez lehet hibaállapot is), amit a modellezett rendszer még az adott ütemben érzékelni fog.

2. 4.-es szinten lévő automaták:
   a. érzékelőelemek: a vonatszimulátor által adott információkat értékelik, szükség esetén állapotot váltanak és megváltoztatják a kimenetüket az új állapotnak megfelelően,
   b. optikák: a fényszimulátor által adott információkat értékelik, szükség esetén állapotot váltanak és megváltoztatják a kimenetüket az új állapotnak megfelelően,
3. 5.-ös szinten lévő automaták:
   a. szakasz: az érzékelőelemek által adott információkat értékeli, szükség esetén állapotot vált és megváltoztatja a kimenetét az új állapotnak megfelelően,
   b. jelzők (fedezőjelzők és ellenőrzőjelző): az optikák által adott információkat értékelik, szükség esetén állapotot váltanak és megváltoztatják a kimenetüket az új állapotnak megfelelően,
4. vezérlő (6. szint): az 5. hierarchia szinten lévő automaták által adott információkat fogadja és értékeli és szükség esetén állapotot vált és megváltoztatja a kimeneteit, vagyis új vezérlési parancsot ad a forgalmat szabályozó kimenetek (jelzők) felé,
5. jelzők (5. szint): a 6. hierarchia szinten lévő vezérlő által adott vezérlési parancsot fogadják és értékelik, majd szükség esetén megváltoztatják a vezérlési kimenetüket a hozzájuk kapcsolódó optikák irányában,
6. optikák (4. szint): az 5. hierarchia szinten lévő jelzők által adott vezérlési parancsot fogadják és feldolgozzák, majd szükség esetén megváltoztatják a vezérlési kimenetüket a hozzájuk kapcsolódó fényszimulátor irányában,
7. fényszimulátor (3. szint): a 4. hierarchia szinten lévő optikák által adott vezérlési parancsot fogadja és feldolgozza, majd szükség esetén megváltoztatja a valós fizikai kimenetek állapotát (a fényszimulátor célja a rendszer valós fizikai fénykimeneteinek a szimulálása).

   Megjegyzés: A fényszimulátor az adott ütem végén állítja be a valós fizikai kimenetek új állapotát (ez lehet hibaállapot is), amit legkorábban a következő ütemben fog érzékelni a modellezett rendszer.

Megjegyzendő továbbá, hogy egy ütem alatt a jelzésadáshoz kapcsolódó automaták (optikák és jelzők) kétszer futnak le (bemenetek kiértékelése, majd a kapott vezérlési parancs megvalósítása). Az azonos hierarchia szinten belüli automata példányok futási sorrendjének modellezése nem volt cél. A futásvezérlő által vezérelt, azonos hierarchia szinten lévő automata példányok egyidejűleg futnak le. (Például a 4. szinten lévő D1, D2 érzékelőelemek és az F1P, F1S, F2P, F2S, EOS optikák egyidejűleg futnak le, vagy másképpen: a 4. szinten lévő automata példányok egyidejűleg lefutnak, ha a 4. szint megkapja a futásengedélyt.)

A vonatszimulátor – miután a kimenetét beállította („megfigyelt állapot”, kék nyíl), addig nem vált állapotot, amíg az érzékelőelemek, illetve a szakasz el nem érte az általa elvárt állapotot, erről az „ellenőrzött állapot” (rózsaszín) interfészen keresztül kap tájékoztatást a vonatszimulátor. A fényszimulátor működése ilyen szempontból a vonatszimulátor működésével analóg. A modellezés során ezt a folyamatot monitor rendszernek (monitorozásnak) neveztük el. A monitorozást nem modelleztük minden ütemben minden automata vonatkozásában, mert a monitorozás célja, hogy az automaták szempontjából kritikus állapotváltáskról (azok sikerességéről) adjon információt a szimulátorok számára (a folyamatok nyomon követése céljából). A szimulátorok a vezérlőt nem monitorozzák, ugyanakkor a vezérlő is egyfajta monitorozó automatának tekinthető, mivel közvetlenül monitorozza az 5. szinten lévő automatákat és közvetetten a 4. szinten lévő automatákat is („megfigyelt állapot”-ok, kék nyilak).

5.2. UPPAAL rendszerdeklarációk

Az UPPAAL keretrendszer rendszerdeklarációs részében az 1. táblázatban szereplő deklarációkat adtuk meg. A deklarációs rész a vonatszimulátor automata (lásd 5.4.3. fejezet) paramétereivel kezdődik. Ezek különböző időzítések „ütem” egységben megadva. Azaz pl. a „Td2 = 2” azt jelenti, hogy a D2 jelű érzékelőelem (2 ütem × 100 ms) = 200 ms hosszan lesz foglalt. A vonatszimulátor működéséhez szükséges paramétereket követően a különböző objektumok egyes példányainak egyedi azonosítóit (eid0…jid0) definiáltuk. Ezeket követik a vezérlő objektum (lásd 5.10. fejezet) által kezelt különböző rendszerszintű időzítéseknek a paraméterei. A soron következő bekezdésben részletezzük ezen paraméterek szerepét a rendszerben.

A behatás késleltetés azt jelenti, hogy ha alapállapotban villamos érkezik a D1 érzékelőelemre, akkor a rendszer a paraméternek megfelelő idő leteltét követően kezdi csak meg a lezárási folyamatot. Azaz a fedezőjelzők sárga villogó jelzési képet fognak adni egészen addig, amíg ez az idő le nem telt. Ennek jelentősége pl. olyan esetekben van, amikor egy megálló van a D1 pont után (azaz a villamos belép a rendszer hatókörébe, de a kereszteződésbe csak a megállóhelyen történő utascserét követően fog behaladni). Az átmeneti idő a fedezőjelzők folyamatos sárga jelzési képen való tartózkodásának az ideje. Az ellenőrzőjelző állítási késleltetés – amennyiben értéke nem 0 – azt jelenti, hogy a fedezőjelzőkön megjelenő piros jelzési képet követően az ellenőrzőjelző nem azonnal fog felkapcsolni (világít állapot), hanem csak a paraméterezett idő leteltével. Ha a D2 ponton a villamos elhagyja a rendszer hatókörét (és egy másik villamos nem tartózkodik a rendszer hatókörében), és az oldás késleltetés paramétere nem 0, akkor az ellenőrzőjelző azonnal sötétre fog kapcsolni, a fedezőjelzők viszont továbbra is piros jelzési képet fognak adni. Ennek az időzítésnek pl. olyan esetben van szerepe, ha a D2 pont olyan közel helyezhető csak el a kereszteződéshez, hogy miután a villamos kilépett rajta a rendszerből, a járműtest még mindig keresztezi a közúti forgalmat.

Végül az UPPAAL rendszerdeklarációs része tartalmazza a modellezett rendszerhez kapcsolódó objektum példányokat a paramétereikkel együtt.

1. táblázat. *UPPAAL rendszerdeklarációk*

```
// A vonatszimulátor paraméterei
// Beállítási lehetőségek: 0..n ütem, összefüggésben a vezérlő időzítéseivel
// (A modell helyes viselkedése függ a helyes beállítástól.)
const int Tsz1 = 20;    // minden érzékelőelem szabad (szakasz szabad)
const int Td1 = 2;      // a D1 érzékelőelem foglalt
const int Tsz2 = 50;    // minden érzékelőelem szabad (szakasz foglalt)
const int Td2 = 2;      // a D2 érzékelőelem foglalt
// Érzékelőelem azonosítók
const int eid0 = 0;     // D1 érzékelőelem azonosítója
const int eid1 = 1;     // D2 érzékelőelem azonosítója
// Szakasz azonosítók
const int sid0 = 0;     // D1D2 szakasz azonosítója
// Optika azonosítók
const int oid0 = 0;     // F1P optika azonosítója
const int oid1 = 1;     // F1S optika azonosítója
const int oid2 = 2;     // F2P optika azonosítója
const int oid3 = 3;     // F2S optika azonosítója
const int oid4 = 4;     // ES optika azonosítója
// Fedezőjelző azonosítók
const int fid0 = 0;     // F1 fedezőjelző azonosítója
const int fid1 = 1;     // F2 fedezőjelző azonosítója
// Ellenőrzőjelző azonosítók
const int jid0 = 0;     // E ellenőrzőjelző azonosítója
// A vezérlő időzítés paraméterei
// Behatás késleltetés
// Beállítási lehetőségek: 0..60 sec, 1 sec felbontásban, azaz:
// 0..600 ütem (vagy ciklus) 10 ütem felbontásban
const int Tbe = 10;      // Behatás késleltetés beállítása: 1  sec = 10 ütem
// Átmeneti idő
```

```
// Beállítási lehetőségek: 2..10 sec, 1 sec felbontásban, azaz:
// 20..100 ütem, 10 ütem felbontásban
const int Tat = 20;     // Átmeneti idő beállítása: 2 sec = 20 ütem
// Ellenőrzőjelző bekapcsolás késleltetés
// Beállítási lehetőségek: 0..60 sec, 1 sec felbontásban, azaz:
// 0..600 ütem (vagy ciklus) 10 ütem felbontásban
const int Tej = 10;     // Ellenőrzőjelző álbekapcsolási idő beállítása: 1 sec = 10 ütem
// Oldás késleltetés
// Beállítási lehetőségek: 0..60 sec, 1 sec felbontásban, azaz:
// 0..600 ütem (vagy ciklus) 10 ütem felbontásban
const int Tol = 10;     // Oldás késleltetés beállítása: 1  sec = 10 ütem
// Template-k
UA = UTEMADO();                      // Ütemadó
LEP = FVEZER();                      // Futásvezérlő
VSZ = VONATSZIM(Tsz1,Td1,Tsz2,Td2);  // Vonatszimulátor
FSZ = FENYSZIM();                    // Fényszimulátor
D1 = ERZEKELO(eid0);                 // D1 érzékelőelem
D2 = ERZEKELO(eid1);                 // D2 érzékelőelem
D1D2 = SZAKASZ(sid0);                // D1D2 szakasz
F1P = OPTIKA(oid0);                  // F1P optika
F1S = OPTIKA(oid1);                  // F1S optika
F2P = OPTIKA(oid2);                  // F2P optika
F2S = OPTIKA(oid3);                  // F2S optika
EOS = OPTIKA(oid4);                  // E optika
F1 = FJELZO(fid0);                   // F1 fedezőjelző
F2 = FJELZO(fid1);                   // F2 fedezőjelző
EJ = EJELZO(jid0);                   // E ellenőrzőjelző
VEZ = VEZERLO(Tbe,Tat,Tej,Tol);      // Vezérlő
// Folyamatok
system UA, LEP, VSZ, D1, D2, D1D2, F1P, F1S, F1, F2P, F2S, F2, EOS, EJ, VEZ, FSZ;
```

5.3. UPPAAL globális deklarációk

Az UPPAAL globális deklarációk részében (2. táblázat) az ütemadó automata (lásd. 5.4.1. fejezet) által adott ütem csatornaként lett definiálva („UTEM”), míg annak ellenőrzésére, hogy az adott ütemben minden objektum (minden példánya) lefutott, egy boolean típusú változót („LEFUTOTT”) hoztunk létre. Az objektumok futási sorrendjét (lásd 5.4.2. fejezet, futásvezérlő automata) „broadcast” (üzenetszórásos) csatornák segítségével biztosítottuk („FUTHAT1”...”FUTHAT7”). A globális deklarációk között definiáltuk az egyes objektumpéldányok számát, valamint egy kódolást vezettünk be az objektumok különböző állapotaihoz, hogy az automaták természetes nyelven is jól értelmezhetők legyenek. Végül a modell globális deklarációi között szerepelnek a monitorozó rendszerhez felhasznált „broadcast” csatornák.

2. táblázat. *UPPAAL globális deklarációk*

```
chan UTEM;                // az ütemadó által adott ütem (1 ütem 100 ms)
bool LEFUTOTT = false;    // ütem megvalósulása (false=nem valósult meg, true=megvalósult)
// Objektumfutások sorrendjének vezérlése
broadcast chan FUTHAT1;    // 1. lépés: a vonatszimulátor fut
broadcast chan FUTHAT2;    // 2. lépés: az érzékelőelemek és az optikák futnak
broadcast chan FUTHAT3;    // 3. lépés: a szakasz és a jelzők futnak
broadcast chan FUTHAT4;    // 4. lépés: a vezérlő fut
broadcast chan FUTHAT5;    // 5. lépés: a jelzők futnak
broadcast chan FUTHAT6;    // 6. lépés: az optikák futnak
broadcast chan FUTHAT7;    // 7. lépés: a fényszimulátor fut
// Konstansok
const int ESZ  = 2;    // Érzékelőelemek száma
const int SZK  = 1;    // Szakaszok száma
const int OPT  = 5;    // Optikák száma
const int FJSZ = 2;    // Fedezőjelző száma
const int EJSZ = 1;    // Ellenőrzőjelzők száma
const int hibas   =  0;           // objektumok hibaállapota
// 1-től 9-ig fenntartott
const int szabad  = 10;           // érzékelőelem és szakasz szabad állapota
const int foglalt = 11;           // érzékelőelem és szakasz foglalt állapota
// 12-től 19-ig fenntartott
const int sotet   = 20;           // jelzésadó elemek(optikák, jelzők) sötét állapota
// 21-től 29-ig fenntartott
const int vilagit = 30;           // optika világít állapota
const int villog  = 31;           // optika villog állapota
// 32-től 39-ig fenntartott
const int folyamatossarga = 40;   // jelzők folyamatossárga jelzési képe
const int villogosarga    = 41;   // jelzők villog sárga jelzési képe
```

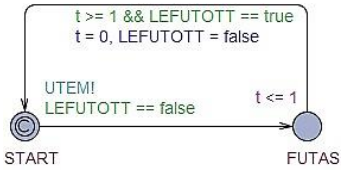
```
const int piros           = 42;   // jelzők piros jelzési képe
// 43-tól 49-ig fenntartott

// Az objektum interfészeken átadott információk
// int típus használt a bővíthetőség miatt
// IO kezelő foglaltsági kimenetei (érzékelőelemek bemenete)
// Sorrend: D1, D2
// Értelmezés: 10=szabad, 11=foglalt
int ErzekeloBemenet[ESZ] = {szabad,szabad};   // érzékelőelemek bemenete
// Érzékelőelem kimenete - szakasz bemenete
// Sorrend: D1, D2
// Értelmezés: 10=szabad, 11=foglalt
int Erzekeloelem[ESZ] = {szabad,szabad};      // érzékelőelemek állapota
// Szakasz
// Értelmezés: 10=szabad, 11=foglalt, 0=hibás
int Szakasz[SZK] = {szabad};           // D1D2 szakasz állapota (szakasz kimenete-vezérlő bemenete)
// IO kezelő fény kimenetei (optikák bemenete, a fizikai fénykimenetek ellenőrzése)
// Sorrend: F1P, F1S, F2P, F2S, ES
// Értelmezés: 20=sötét, 30=világít, 31=villog
int FenyBemenet[OPT] = {sotet,villog,sotet,villog,sotet};     // fények ellenőrzött állapota
// IO kezelő fény bemenetei (optikák kimenete, a fizikai fénykimenetek vezérlése)
// Sorrend: F1P, F1S, F2P, F2S, ES
// Értelmezés: 20=sötét, 30=világít, 31=villog
int FenyKimenet[OPT] = {sotet,villog,sotet,villog,sotet};     // fények vezérelt állapota
// Optikák
// Sorrend: F1P, F1S, F2P, F2S, ES
// Értelmezés: 20=sötét, 30=világít, 31=villog
int OptikaAllapot[OPT] = {sotet,villog,sotet,villog,sotet};   // optikák ellenőrzése
int OptikaVezerles[OPT] = {sotet,villog,sotet,villog,sotet};  // optikák vezérlése
// Fedezőjelzők
// Sorrend: F1, F2
// Értelmezés: 20=sötét, 40=folyamatos sárga, 41=villogó sárga, 42=piros
int FedezojelzoAllapot[FJSZ] = {villogosarga,villogosarga};   // fedezőjelzők ellenőrzése
int FedezojelzoVezerles[FJSZ] = {villogosarga,villogosarga};  // fedezőjelzők vezérlése
// Ellenőrzőjelző
// Értelmezés: 20=sötét, 40=folyamatos sárga
int EllenorzojelzoAllapot[EJSZ] = {sotet};     // ellenőrzőjelző ellenőrzése
int EllenorzojelzoVezerles[EJSZ] = {sotet};    // ellenőrzőjelző vezérlése
// Monitoring rendszer tartozékai
broadcast chan MERZEKELO;   // érzékelőelem állapotváltás monitor
broadcast chan MSZAKASZ;    // szakasz állapotváltás monitor
broadcast chan MOPTIKA;     // optika állapotváltás monitor
broadcast chan MFJ;         // fedezőjelző állapotváltás monitor
broadcast chan MEJ;         // ellenőrzőjelző állapotváltás monitor
```

5.4. A modellezett rendszer szimulációjához, modellellenőrzéséhez definiált automaták

A jelen fejezetben bemutatott automaták létrehozását az indokolta, hogy a szóban forgó közúti fedezőjelző rendszer komponenseinek modellezésével olyan modellt kaptunk, amely önmagában alkalmatlan szimulációra és modellellenőrzésre, így a jelen tanulmányban leírt kutatási eredmények további felhasználásra alkalmatlanok lennének, pontosabban a 3.2. fejezetben ismertetett FMBRSE módszertan folyamatának 5. és 6. lépése nem lenne végrehajtható. Az itt megadott szempontok figyelembevétele elengedhetetlen az FMBRSE módszertan alkalmazásához, azonban megjegyezzük, hogy a jelen fejezetben prezentált megoldások nem kizárólagosak. Természetesen a javasolt módszerhez képest létezhetnek más, akár hatékonyabb megoldások is. Összefoglalva három olyan szempontot találtunk, melyek az UPPAAL-ban létrehozott formális modell szimulációjához és modellellenőrzéséhez szükségesek: időkezelés (5.4.1. fejezet), végrehajtás vezérlése (5.4.2. fejezet) és bemeneti függvény/függvények (5.4.3. fejezet).

5.4.1. Ütemadó

A modellezett rendszer ciklus alapú végrehajtási sémával megvalósított, ami azt jelenti, hogy a végrehajtása (az automaták végrehajtása/lefutása) rendszeres időközönként történik meg. Az ütemadó automata (5. ábra) célja, hogy alapot adjon a rendszeres, meghatározott időközönként (ciklusonként) történő viselkedés modellezéséhez.



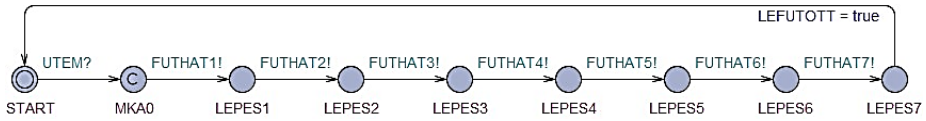
5. ábra.
Ütemadó automata

A „START” állapotban az idő nem telik a teljes modellezett rendszerre nézve, amire az állapotba írt “C” utal (committed state). A „FUTÁS” állapotban az ütem ideje telik. Az állapothoz tartozik egy invariáns: $t \leq 1$. Az 1-es érték egységnyi időértéket jelent, amelyhez tetszőleges érték társítható. Az esettanulmányban az egységnyi időértékhez 100 ms értéket társítottunk. Megjegyezzük, hogy a modellben használt időzítések és paraméterek aktuális értéke az eltelt ütemek számával adható meg. (Pl. ha az átmeneti idő követelmény szerinti értéke 4 s, akkor ez 4000 ms-et, azaz 40 ütemet jelent.) Az invariáns $t \leq 1$, azaz „FUTAS” állapotban mindig igaz, vagyis ebben az állapotban a t idő értéke mindig a meghatározott egységnyi időnél kisebb vagy azzal egyenlő.

Az „UTEM!” szinkronizáció egy új ütem megkezdését jelzi a „FUTASVEZERLO” automata felé. A „LEFUTOTT” változót „true” értékét a futásvezérlő automata (lásd 5.4.2. fejezet) állítja be: csak akkor indulhat egy újabb ütem, ha az előző már befejeződött/lefutott.

5.4.2. Futásvezérlő

A futásvezérlő automata (6. ábra) célja a modellben szereplő automaták futási sorrendjének meghatározása egy ütemen/cikluson (5.4.1. fejezet) belül. Ezeket a futásengedélyeket a „FUTHATi” csatorna segítségével valósítja meg az automata. Az automata kezdőállapota a „START” állapot, amelyből az ütemadó (5.4.1 fejezet) automatától kapott „UTEM” csatornán érkező jelre indíthatja el működését. Ha futásvezérlő minden automatának adott engedélyt a működésre, akkor tevékenységeinek elvégzését a „LEFUTOTT” változón keresztül jelenti vissza az ütemadó automatának.



6. ábra.
Futásvezérlő automata

5.4.3. Bemeneti függvények

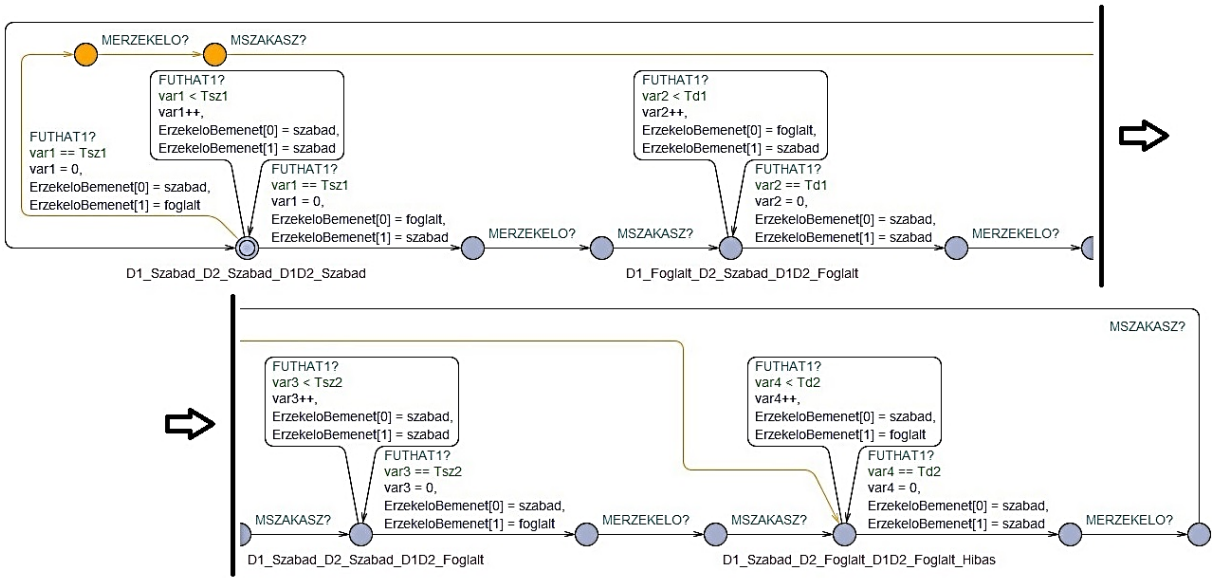
A vonatszimulátor automata (7. ábra) célja, hogy az esettanulmány által leírt vonatmozgások (menetek) modellezését megvalósítsa. Az esettanulmány két menetet tárgyal:

a. helyes menet: D1 majd D2 érzékelőelem érintése,

b. helytelen menet: D1 érzékelőelem érintése kimarad, D2 érzékelőelem érintése.

A helytelen menethez kötődő állapotokat és éleket a vonatszimulátor automatában narancssárga szín különbözteti meg a helyes menethez tartozó állapotoktól és élektől.

A vonatszimulátor automata elnevezett állapotai alapján (7. ábra) jól követhető az érzékelőelemek és a belőlük képzett logikai szakasz tényleges állapota és az automata működése. Az automata működésének értelmezéséhez szükséges az 5.2. és 5.3. fejezetben leírt deklaráción túl az 5.4.2. fejezetben megadott futásvezérlő ismerete („FUTHAT1” csatorna), valamint az, hogy az automatához tartozó deklarációs részben a „var1”…”var4” változók rendre az 1. táblázatban megadott Tsz1, Td1, Tsz2, Td2 paraméterekhez tartozó időzítéseket jelölik.

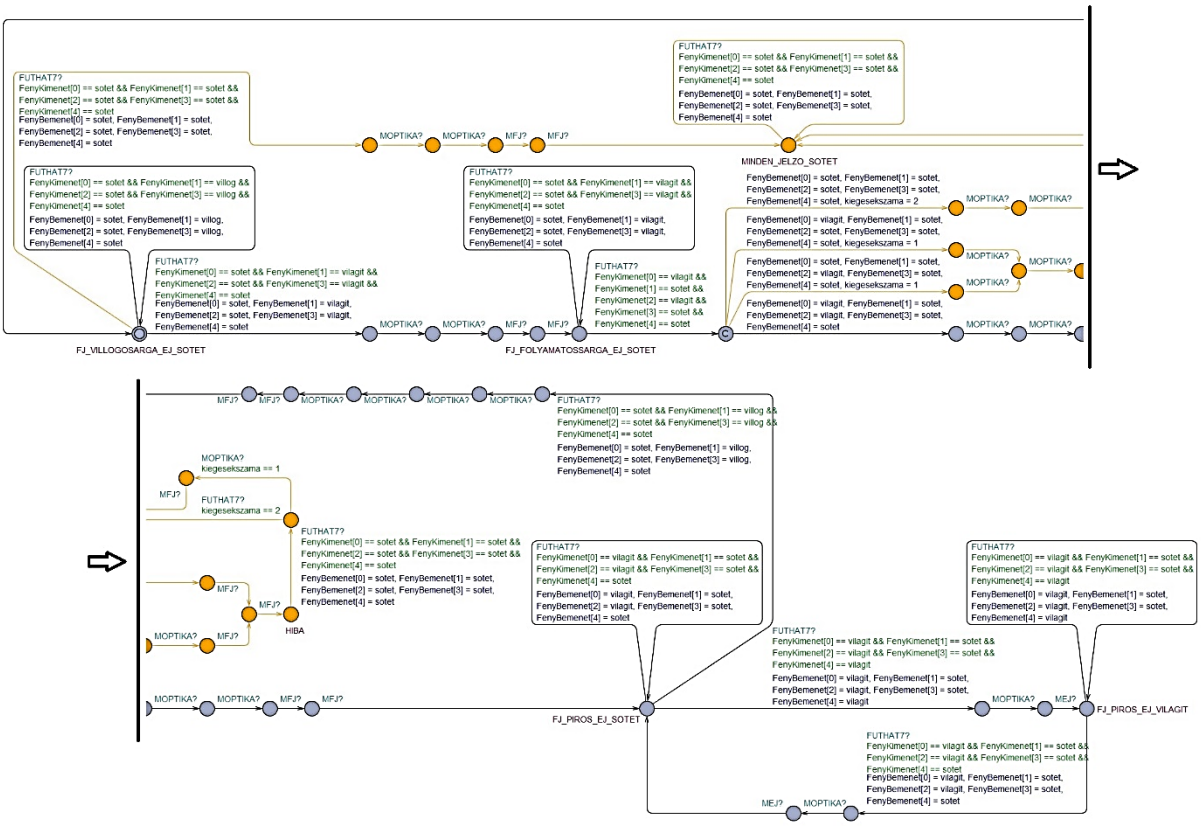


7. ábra.
Vonatszimulátor automata

A fényszimulátor automata (8. ábra) célja kettős:

a. egyrészt helyettesíti az optikák állapotának fizikai megfigyelését (áram és feszültség mérők állapotai), amely alapján bemenetet szolgáltat az optika objektumok számára azok tényleges állapotáról (sötét, világít, villog),

b. másrészt fogadja az optikák által a fizikai kimenetre kiadott jelzési parancsot és szimulálja azok beállítását (piros optikák esetén képes a világít parancs ellenére sötét szimulálására, ami a „kiégett” optika hibának felel meg).



8. ábra.
Fényszimulátor automata

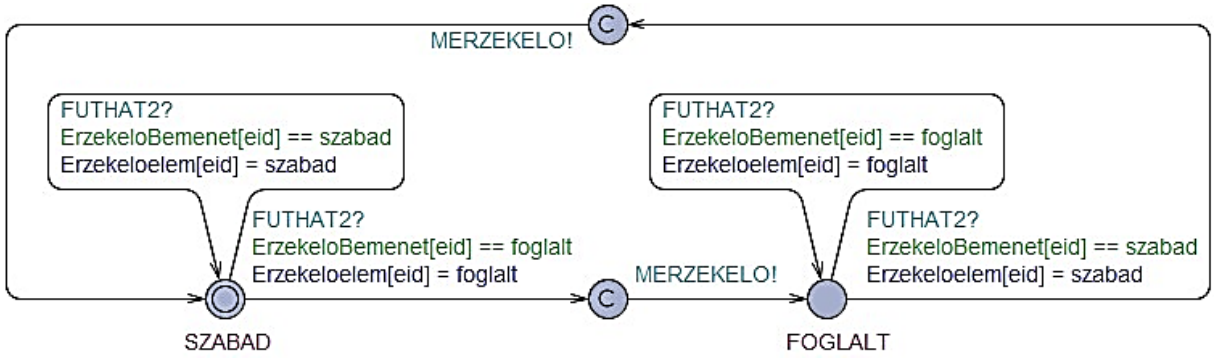
A piros optika hibákhoz (kiégésekhez) tartozó állapotokat és éleket a fényszimulátor automatában narancssárga szín különbözteti meg a helyes optika állapotokhoz és vezérlésekhez tartozó állapotoktól és élektől.

A 8. ábra elnevezett állapotai alapján jól követhető az esettanulmányban bemutatott fedezőjelzők és ellenőrzőjelző valós állapotának alakulása, amely segíti az automata viselkedésénekmegértését. Az automata működésének értelmezéséhez szükséges a 4.2. és 4.3. fejezetben leírt deklaráción túl a 4.4.2- ben megadott futásvezérlő ismerete („FUTHAT7" csatorna), valamint az, hogy az automatához tartozó deklarációs részben a „kiegesekszama" változó a kiégett optikák számának tárolására szolgál, értékkészlete: [0..2].

Összefoglalva a vonatszimulátor és fényszimulátor automaták „kvázi" bemeneti függvényeket és függvényértékeket képeznek a modell számára a különböző állapotváltások biztosításához. A bemeneti függvény(ek) megvalósítása – mely nem feltétlenül kell, hogy automata formájában jelenjen meg – nagyban függ a modellezés céljától. A bemutatott esettanulmányban pl. több olyan objektum is van, melyek bizonyos állapotai nem érhetők el az ebben a fejezetben megadott bemeneti függvények használatával.

5.4.4. Érzékelőelem

Az esettanulmányban szereplő D1 és D2 jelű érzékelőelem modellje látható a 9. ábrán. Az automata a szimuláció és modellellenőrzés során két példányban áll elő, ilyen módon a 9. ábrán szereplő automata egy absztrakció. Az automata működése egyszerű: az érzékelőelem szabad vagy foglalt állapotban lehet, attól függően, hogy tartózkodik-e felette villamos. A villamos jelenlétet és annak hosszát az érzékelőelem felett a vonatszimulátor automata (lásd. 5.4.3. fejezet) állítja elő a konfigurált időparaméterek alapján (1. táblázat, Td1 és Td2). Az érzékelőelem a futásvezérlővel (5.4.2. fejezet) a „FUTHAT2" csatornán keresztül tart kapcsolatot.

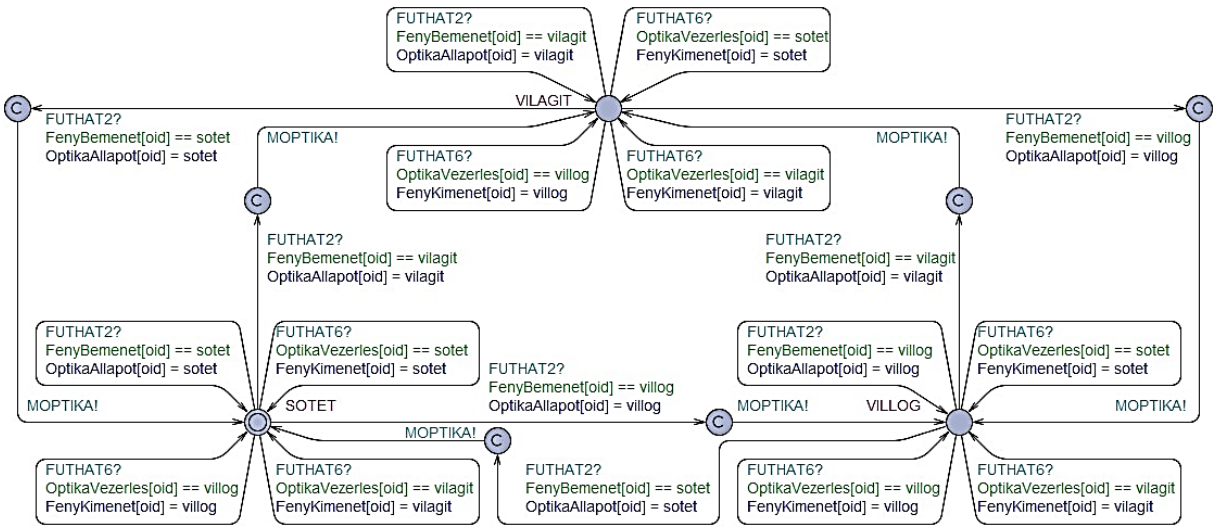


9. ábra.
Érzékelőelem automata

5.4.5. Optika

Az esettanulmányban összesen öt optika példány szerepel: két-két optika a két fedezőjelzőkhöz (F1 és F2) kapcsolódik, míg egy optika az ellenőrzőjelzőhöz (E). A 10. ábrán látható optika automata a szimuláció és modellellenőrzés során összesen öt példányban jön létre, ilyen módon a 10. ábrán szereplő automata egy absztrakció.

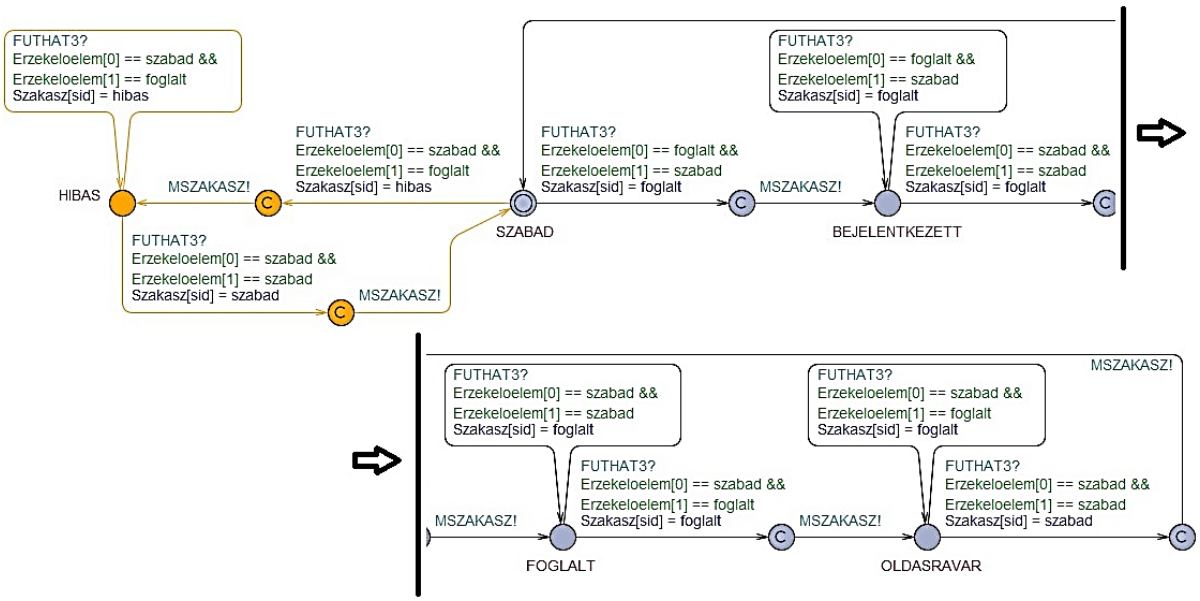
Az optika automata működése: az optika sötét, világít vagy villog állapotban lehet. Az optikák aktuális állapotát a fényszimulátor automata (5.4.3. fejezet) generálja, ahogy az optikákra kiadott új vezérlési parancsot is ez az automata juttattja érvényre a modellben. Az optika példányok a futásvezérlővel (5.4.2. fejezet) a „FUTHAT2" és a „FUTHAT6" csatornán keresztül tartanak kapcsolatot. Előbbi csatorna engedélyezi az aktuális bemenetek beolvasását és ezek alapján az optika állapotának megváltoztatását (ha erre szükség van), míg utóbbi engedélyezi az új vezérlés átadását a fényszimulátor automata felé.



10. ábra.
Optika automata

5.4.6. Szakasz

Az esettanulmányban egyetlen logikai szakasz szerepel, melynek formális modellje a 11. ábrán látható. A szakasz automata működése során az érzékelőelemek állapota alapján határozza meg saját állapotát, melyet továbbít a vezérlő automata (5.10. fejezet) számára.



11. ábra.
Szakasz automata

Az esettanulmány két menetet tárgyal, a szakasz modellje pedig ezeknek megfelelően kétféleképpen futhat le:

a. helyes menet: D1 majd D2 érzékelőelem érintése,

b. helytelen menet: D1 érzékelőelem érintése kimarad, D2 érzékelőelem érintése.

A helytelen menethez kötődő állapotokat és éleket a szakasz automatában narancssárga szín különbözteti meg a helyes menethez tartozó állapotoktól és élektől. Helytelen menet esetén a szakasz „HIBAS” állapotba kerül, ahonnan a hozzá kapcsolódó érzékelőelemek szabaddá válásával automatikusan alapállapotba képes kerülni (modellezési egyszerűsítés).

Helyes menet esetén a szakasz négyféle állapotba kerülhet a következő sorrendben:

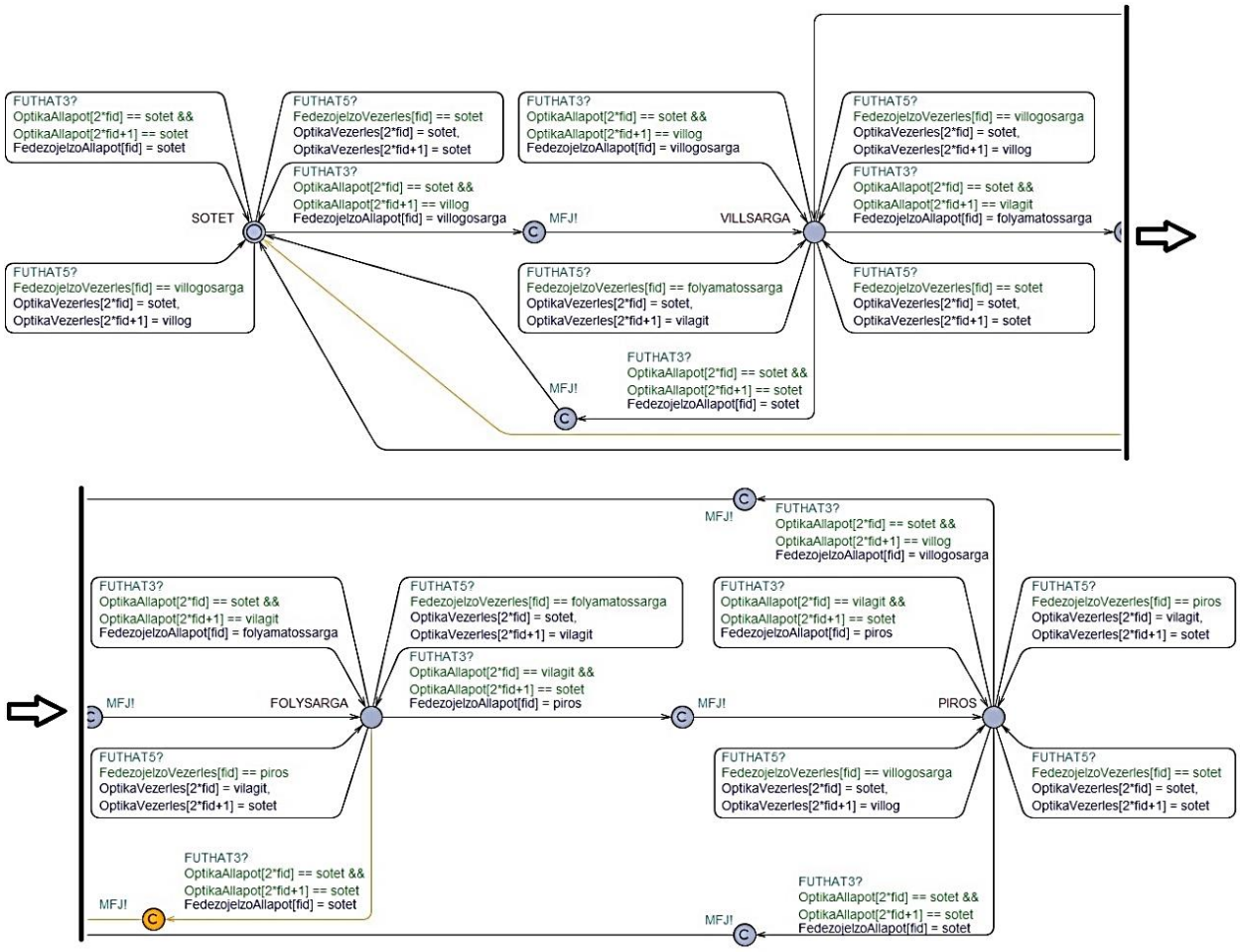
1. „SZABAD”: alapállapot (nincs érzékelt villamos jelenlét),
2. „BEJELENTKEZETT”: D1 érzékelőelem foglaltsága alatt,
3. „FOGLALT”: D1 foglaltságának megszűnését követően, (D1 és D2 érzékelőelem ebben az állapotban szabadok)
4. „OLDASRAVAR”: D2 érzékelőelem foglaltság alatt.

Fenti folyamatból jól látható pl. az a modellezési egyszerűsítés, hogy a szakasz nincs felkészítve követő villamos fogadására, illetve a villamos áramszedők számlálására (egy villamoshoz tartozó több áramszedő): azaz a modellezett szakasz egyidejűleg egy egyáramszedővel rendelkező villamos kezelésére képes. (További egyszerűsítés, hogy egyvágányú pálya esetében a szakasz csak egy irányban használható: D1 > D2).

A villamos jelenlét hosszát a vonatszimulátor automata (lásd. 5.4.3. fejezet) határozza meg a konfigurált időparaméterek alapján (1. táblázat, Tsz1 és Tsz2). A szakasz a futásvezérlővel (5.4.2. fejezet) a „FUTHAT3” csatornán keresztül van kapcsolatban.

5.4.7. Fedezőjelző

A modellezett esettanulmányban összesen két fedezőjelző példány szerepel (F1 és F2). A 12. ábrán látható fedezőjelző automata a szimuláció és modellellenőrzés során összesen két példányban jön létre, ilyen módon a 12. ábrán szereplő automata egy absztrakció.



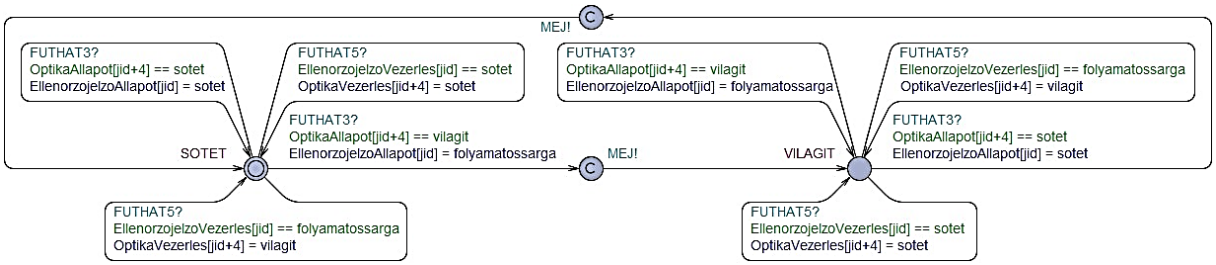
12. ábra.
Fedezőjelző automata

A fedezőjelző automata működése során négy stabil állapotot vehet fel: sötét, villogó sárga, folyamatos sárga, piros. Az automata az állapotát a hozzá tartozó két optika (5.6. fejezet) állapota és a vezérelt jelzési kép (vezérlő, 5.10. fejezet) alapján határozza meg. A fedezőjelző példányok a futásvezérlővel (5.4.2. fejezet) a „FUTHAT3” és a „FUTHAT5” csatornán keresztül tartanak kapcsolatot. Előbbi csatorna engedélyezi az aktuális optika bemenetek beolvasását, míg utóbbi engedélyezi az új vezérlés átadását az optikák felé.

A fényszimulátor automatánál (5.4.3. fejezet) bemutatott piros optika hibákhoz (kiégésekhez) tartozó állapotokat és éleket a fedezőjelző automatában narancssárga szín különbözteti meg a helyes fedezőjelző állapotokhoz és vezérlésekhez tartozó állapotoktól és élektől (lásd 12. ábra).

5.4.8. Ellenőrzőjelző

A modellezett esettanulmányban egy ellenőrzőjelző példány szerepel (E). A 13. ábrán látható az ellenőrzőjelző automata sablonja, (amely fel van készítve a példányosításra is).



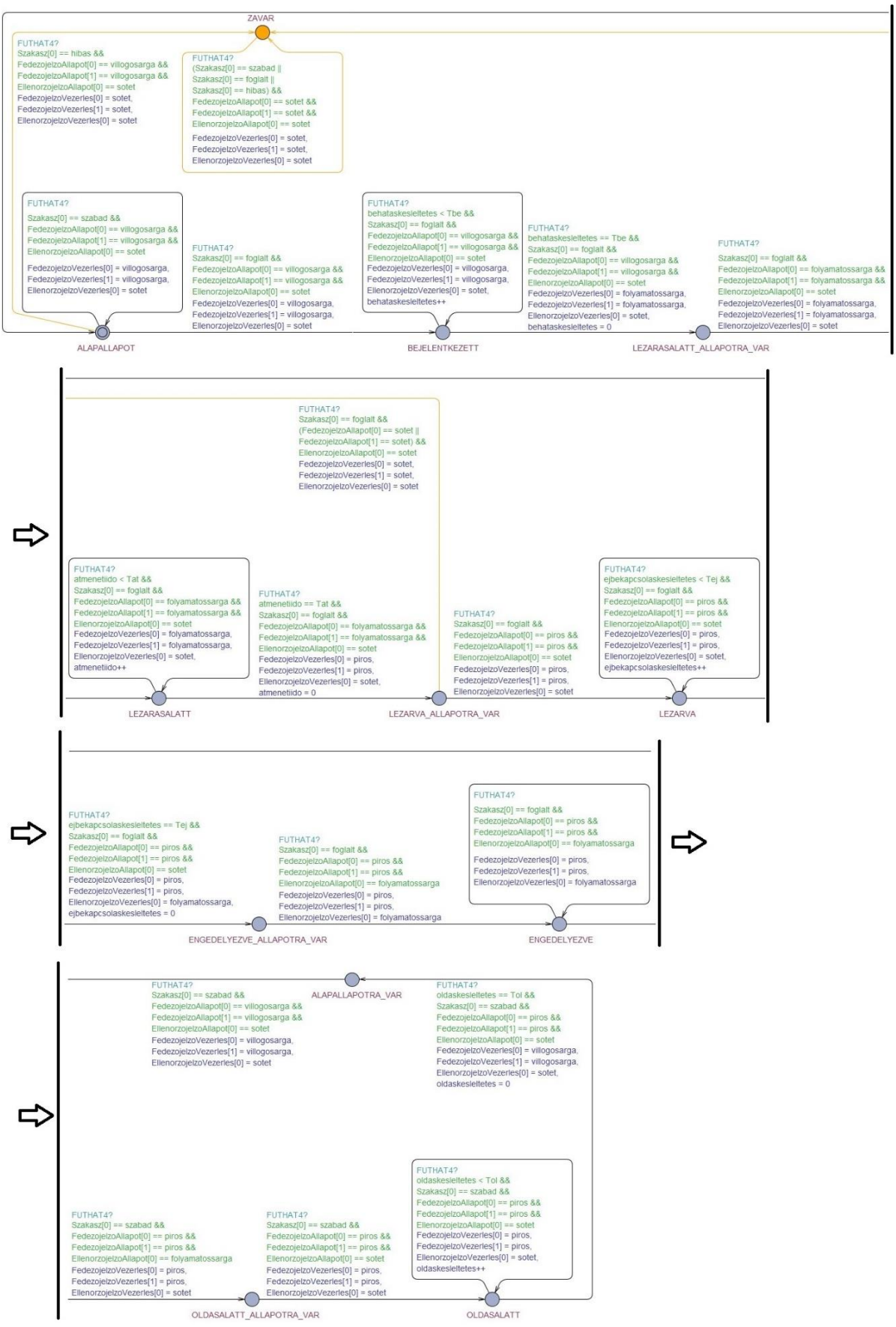
13. ábra.
Ellenőrzőjelző automata

Az ellenőrzőjelző automata működése során két stabil állapotot vehet fel: sötét és világít. Az automata az állapotát a hozzá tartozó egyetlen optika (5.6. fejezet) állapota és a vezérelt jelzési kép (vezérlő, 5.10. fejezet) alapján határozza meg. Az ellenőrzőjelző a futásvezérlővel (5.4.2. fejezet) a „FUTHAT3” és a „FUTHAT5” csatornán keresztül van kapcsolatban (hasonlóan a vele egy hierarchia szinten lévő fedezőjelzőkhöz, lásd 5.8. ill. 5.1. fejezetek). Előbbi csatorna engedélyezi a hozzá tartozó optika bemenetének beolvasását, míg utóbbi engedélyezi az új vezérlés átadását az optika felé.

5.4.9. Vezérlő

A vezérlő objektum automatája a 14. ábrán látható. Az alapfolyamat (hibamentes viselkedés) a következő lépésekből áll:

1. „ALAPÁLLAPOT”: nincs érzékelt villamos jelenlét (szabad szakasz, fedezőjelzők sárgán villognak, ellenőrzőjelző sötét),
2. „BEJELENTKEZETT”: jármű van a rendszer hatókörében (foglalt szakasz), behatás késleltetés fut (fedezőjelzők sárgán villognak, ellenőrzőjelző sötét),
3. „LEZARASALATT_ALLAPOTRA_VAR”: jármű van a rendszer hatókörében (foglalt szakasz), behatás késleltetés lejárt (fedezőjelzők sárgán villognak, ellenőrzőjelző sötét), a rendszer arra várakozik, hogy a fedezőjelzőkön a folyamatos sárga jelzési kép stabilan megjelenjen,
4. „LEZARASALATT”: jármű van a rendszer hatókörében (foglalt szakasz), átmeneti idő fut, (fedezőjelzők folyamatos sárgák, ellenőrzőjelző sötét),
5. „LEZARVA_ALLAPOTRA_VAR”: jármű van a rendszer hatókörében (foglalt szakasz), átmeneti idő lejárt (fedezőjelzők folyamatos sárgák, ellenőrzőjelző sötét), a rendszer arra várakozik, hogy a fedezőjelzőkön a piros jelzési kép stabilan megjelenjen,
6. „LEZÁRVA”: jármű van a rendszer hatókörében (foglalt szakasz), ellenőrzőjelző bekapcsolás késleltetés fut (fedezőjelzők pirosak, ellenőrzőjelző sötét),
7. „ENGEDELYEZVE_ALLAPOTRA_VAR”: jármű van a rendszer hatókörében (foglalt szakasz), ellenőrzőjelző bekapcsolás késleltetés lejárt (fedezőjelzők pirosak, ellenőrzőjelző sötét), a rendszer arra várakozik, hogy az ellenőrzőjelzőn a sárga jelzési kép stabilan megjelenjen,
8. „LEZARVA”: jármű van a rendszer hatókörében (foglalt szakasz), ellenőrzőjelző bekapcsolás késleltetés lejárt (fedezőjelzők pirosak, ellenőrzőjelző világít),
9. „OLDASALATT_ALLAPOTRA_VAR”: a jármű elhagyta a rendszer hatókörét (szabad szakasz, fedezőjelzők pirosak, ellenőrzőjelző világít), a rendszer arra várakozik, hogy az ellenőrzőjelzőn a sárga jelzési kép stabilan megszűnjön,
10. „OLDASALATT”: nincs jármű a rendszer hatókörében (szabad szakasz), oldás késleltetés fut (fedezőjelzők pirosak, ellenőrzőjelző sötét),



14. ábra.
Vezérlő automata

11. „ALAPALLAPOTRA_VAR”: nincs jármű a rendszer hatókörében (szabad szakasz), oldás késleltetés lejárt (fedezőjelzők pirosak, ellenőrzőjelző világít), a rendszer arra várakozik, hogy a fedezőjelzőkön a piros jelzési kép stabilan megszűnjön,
12. „ALAPÁLLAPOT”: nincs érzékelt villamos jelenlét (szabad szakasz, fedezőjelzők sárgán villognak, ellenőrzőjelző sötét),

A modellezett hibákhoz (piros optika kiégések, D1 érzékelőelem nem érzékel járművet) tartozó állapotokat és éleket a vezérlő automatában narancssárga szín különbözteti meg a helyes viselkedéshez tartozó állapotoktól és élektől.

A vezérlő automata a futásvezérlővel (5.4.2. fejezet) a „FUTHAT4” csatornán keresztül van kapcsolatban. Előbbi csatorna engedélyezi a vezérlő automata futását, ami a hozzá kapcsolódó szakasz (5.7 fejezet) és jelzők (5.8. és 5.9. fejezetek) állapota alapján meghatározza a rendszer állapotát és szükség esetén új vezérlési parancsot ad a jelzők felé.

KONKLÚZIÓ

Tanulmányunkban bemutattuk a FMBRSE módszertant, amelynek célja, hogy támogatást adjon a vasúti fejlesztésben dolgozó szakterületi mérnököknek a formális specifikáció és verifikáció hatékony alkalmazásához. A módszertan elvárt eredményeként a rendszer/komponensek formálisan ellenőrzött funkcionális modelljét kaphatjuk meg.

A FMBRSE módszertan által javasolt folyamat egy kritikus lépését, a formális modell létrehozását egy összetett, több komponensből álló esettanulmány segítségével mutattuk be. A modellezés bemeneteként egy már meglévő követelményspecifikációt használtunk fel. A modellezéshez az UPPAAL keretrendszert alkalmaztuk.

A modellezés során azt tapasztalatuk, hogy a vasúti szakterületen a funkcionalitást érintő problémakörök modellezése jól automatizálható, amennyiben elvégezzük a követelményspecifikációban szereplő, a modellezés szempontjából bemeneti információk/adatok rendszerezését, strukturálását, tisztázását. Ezen előfeltétel mellett egyúttal a létrehozott formális modell(ek) modellellenőrzése is lehetővé válhat, mert az elemzett, „formálisabbá tett” követelményspecifikáció jó alapot ad a modellellenőrzés bemeneteként szolgáló temporális logikai formulák levezetésére.

Az esettanulmány modellezése során azt tapasztaltuk, hogy nem elegendő a vizsgált rendszer szigorú értelemben vett funkcionalitását modellezni. Az elkészült funkcionális modellt a szimulálhatóság és modellellenőrizhetőség érdekében ezért kiegészítettük további modellelemekkel, amelyek eddigi megfigyeléseink alapján részben a tervezett rendszer/szoftver architektúrától, részben pedig a bennfoglaló rendszertől (környezettől) függhetnek. Előbbiekre példa az időkezelés modellezése vagy a vezérlés végrehajtását modellező automata, míg utóbbira a bemeneti függvény(ek) modellje, ha értelmezhető(ek).

A jelen tanulmányban bemutatott esettanulmánnyal kapcsolatos munka folytatásaként az elkészült modell modellellenőrzését tervezzük végrehajtani. Ezzel szeretnénk igazolni az általunk javasolt FMBRSE módszertan alkalmazhatóságát. A modellezéssel kapcsolatos eredményeinkre építve célszerűnek tartjuk az FMBRSE automatizálhatósági kérdéskörének alaposabb vizsgálatát. Célunk a módszertanhoz tartozó folyamat egyéb részeinek bemutatása is, melyet különböző esettanulmányok segítségével, további publikációk keretében tervezünk.

HIVATKOZÁSOK

[1] A. Kunnappilly, P. Backeman, C. Seceleanu, From UML Modeling to UPPAAL Model checking of 5G Dynamic Service Orchestration, ECBS 2021, Conference on the Engineering of Computer Based Systems, doi: 10.1145/3459960.3459965.

[2] Unified Modeling Language (OMG UML), 2017, Version 2.5.1.

[3] F. Vaandrager, A First Introduction to Uppaal, 2011

[4] D. Darvas Practice-Oriented Formal Methods to Support the Software Development of Industrial Control Systems, 2016, doi: 10.5281/zenodo.162950.

[5] Yul Y. Nazaruddin, Tua A. Tamba, K. Pradityo, B. Aristyo, A. Widyotriatmo, Safety Verification of a Train Interlocking Timed Automaton Model, IFAC-PapersOnLine, Volume 52, Issue 15, 2019, Pages 331-335, ISSN 2405-8963, https://doi.org/10.1016/j.ifacol.2019.11.696.

[6] Wang Keming, Wang Zheng, Zhang Chuandong, Formal Modeling and Data Validation of General Railway Interlocking System, 16th Internal Conference on Railway Engineering Design & Operation, Lisbon, Portugal, 2018, vol. 181, pp. 527-538, ISSN 1743-3509, doi: https://doi.org/10.2495/CR180471.

[7] CENELEC EN 50126:2017 Railway applications – The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS) – Part 1: Generic RAMS Process (English version).

[8] A. van Lamsweerde Requirements Engineering: From System Goals to UML Models to Software Specifications, 2009, ISBN: 978-0-470-01270-3.

[9] H. Colin Requirements Management, The Interface Between Requirements Development and All Other Systems Engineering Processes, 2007, Springer-Verlag Berlin and Heidelberg GmBH & Co. KG, ISBN: 9783540476894.

[10] IBM Rational DOORS, Requirements Management Framework Add-On, User Manual, Release 6.1, 2013

[11] D. Long, Z. Scott A primer for Model-Based Systems Engineering (2nd edition), 2011, ISBN 978-1-105-58810-5.

[12] H. Heinrich Model-Driven Development of Advanced User Interfaces, 2011, ISBN13: 9783642145612.

[13] B. R. Hunt, R. L. Lipsman, J. m. Rosenberg, et al. A guide to MATLAB for Beginners and Experienced Users, Cambridge University, 2001, ISBN-13: 978-0-511-07792-0.

[14] Devendra K. Chaturvedi, Modeling and Simulation of Systems Using MATLAB and Simulink, 2010, ISBN 9781439806722.

[15] J. Holt, S. Perry SysML for Model-Based Systems Engineering, Institution of Engineering and Technology, 2013, ISBN: 1849196516.

[16] CENELEC EN 50128:2011 Railway applications – Communication, signaling and processing systems – Software for railway control and protection systems (English version).

[17] Jiacun Wang, William Tepfenhart, Formal Methods in Computer Science, 2019, ISBN 9781498775328

[18] Barry Boehm, Dan Port & A. Winsor Brown (2002) Balancing Plan-Driven and Agile Methods in Software Engineering Project Courses, Computer Science Education, 12:3, 187-195, doi: 10.1076/csed.12.3.187.8617.

[19] PO Asagba, EE Ogheneovo, A Comparative Analysis of Structured and Obejct-Oriented Programming methods, African Journals (AJOL), vol. 11. No. 4 (2007), doi: 10.4314/jasem.v11i4.55190.

[20] Campean F., Henshall E., Yildirim U., Uddin A., Williams H. (2013) A Structured Approach for Function Based Decomposition of Complex Multi-disciplinary Systems. In: Abramovici M., Stark R. (eds) Smart Product Engineering. Lecture Notes in Production Engineering. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-30817-8_12.

[21] Herrigel S, Laumanns M, Nash A, Weidmann U. Hierarchical Decomposition Methods for Periodic Railway Timetabling Problems. Transportation Research Record. 2013;2374(1):73-82. doi:10.3141/2374-09.

[22] Budapest Transport Privately Held Corporation (BKV), Requirements booklet for safety elements and equipment for traffic control of trams (BKV-VILL-1.04), 2011.

[23] 1/1975. (II.5.) KPM-BM együttes rendelet a közúti közlekedés szabályiról.