Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar

Repülőgép futómű-rendszerének modellezése Event-B segítségével

Írásos összefoglaló Amel Mammar és Régine Laleau Modelling a landing gear system in Event-B című művéről

Kacsó Ágota Enikő
2015. december 10.

1. Bevezetés

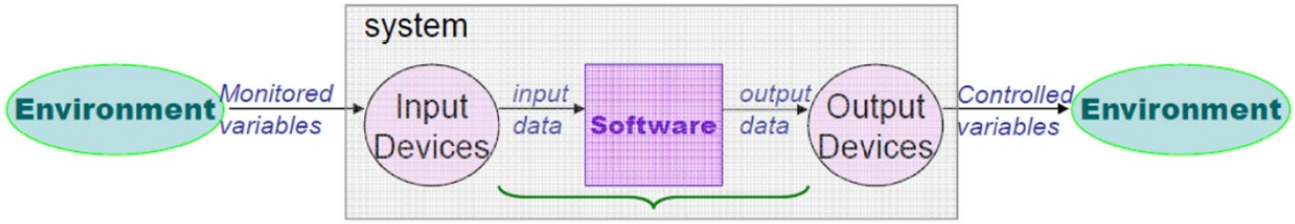
Mivel a biztonságkritikus rendszerek tervezésében bármilyen apró hiba súlyos következményekhez, akár többek halálához is vezethet, nagyon fontos ezeknek a részletes és alapos ellenőrzése. Napjainkban széles körben elfogadott a formális módszerek használata az ilyen rendszerek helyességbizonyítására.

Az általam választott esettanulmány egy repülőgép futómű-rendszerének Event-B segítségével történő modellezését és modellellenőrzését mutatja be.

A futómű-rendszer feladata biztosítani a futóművek biztonságos kiengedését és behúzását, amikor a repülőgép landol, illetve felszáll. Mindegyik futómű egy ajtóval ellátott futómű dobozban van, amelyik ki kell nyíljon, amikor a futómű behúzódása, vagy kiengedése éppen folyamatban van, és be kell záródjon, amikor ezek teljesen behúzódtak vagy kiengedtek. Ennek érdekében egy kontroller szenzorok segítségével periodikusan olvassa a különböző elemek állapotát (futómű, ajtók, karok, stb.) és utasításokat küld akülönböző elektro-szelepeknek, hogy például kinyissák/becsukják az ajtókat, vagy kiengedjék/behúzzák a futóműveket.

2. A rendszer modellezése

A rendszer modellezéséhez a szerzők a változókat a Parnas és Madey féle négyváltozós modell szerint osztályozták (1. ábra):



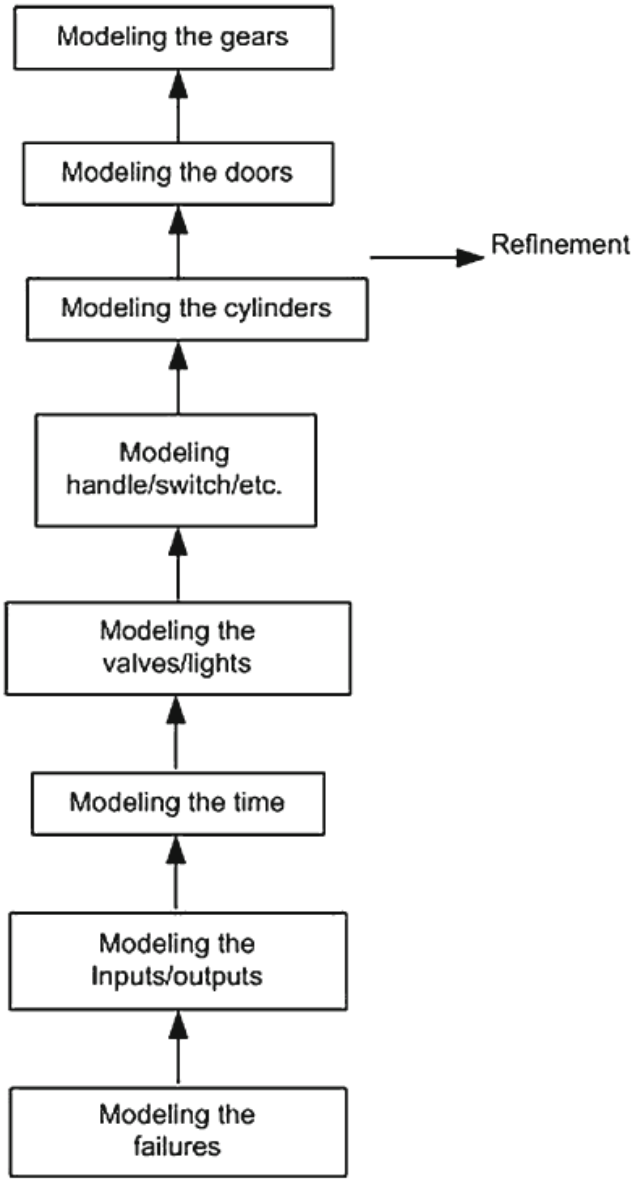
1. Ábra: A négyváltozós modell

1. Környezeti változók: a kontrolleren kívüli változó állapotát adja meg
    a. Megfigyelt változók: az értéküket nem a kontroller számolja, mérhetőek, megfigyelhetőek.
    b. Ellenőrzött változók: ezek értékét a kontroller határozza meg.
2. Kontroller változók: A kontrolleren belüli változók tartoznak ide.
    a. Bemeneti adat: a szenzorok által meghatározott értékek.
    b. Kimeneti adat: a kontroller által a különböző környezeti elemeknek kiadott parancsok tartoznak ebbe a kategóriába.

A rendszer modellezése az alábbi fő lépéseken keresztül történt (2. ábra):

2.1. A megfigyelt változók modellezése:

A megfigyelhető változók a fizikai komponensek állapotához köthetőek. Ezek: a futóművek, az ajtók, a hengerek, karok, a hidraulikus kör és a kapcsoló. A szerzők a futómű leírásával kezdték a rendszer modellezését. Kezdetben csak a kiengedett és nem kiengedett állapotok léteztek, majd az első finomításban a nem kiengedett állapotot felbontották két részre: visszavont és részlegesen kiengedett. A részlegesen kiengedett átmeneti állapotra azért van szükség, mert a futómű kiengedése/visszahúzása időt vesz igénybe, tehát léteznie kell olyan állapotnak is, amikor se nem kiengedett se nem visszahúzott a futómű. A második és harmadik finomítások az ajtókra vonatkoztak. A futóműhöz hasonlóan itt is először két állapottal (nyitott vagy nem nyitott), majd három állapottal (részben nyitott állapot bevezetése) modellezték a rendszert. A negyedik finomítás során vezették be a hengereket, amelyek a futóművek és az ajtók mozgását teszik lehetővé. Az ötödik finomítás során modellezték a karokat, kapcsolókat, a hidraulikus kör és a lengéscsillapítókat.



2. Ábra: A modellezés fő lépései

2.1.1 A futóművek modellezése: a kezdeti modell és az első finomítási lépés

A futóművek modellezésekor bevezettek egy kontextust ezek helyzetének (front, left, right) megkülönböztetésére, és egy logikai változót lehetséges két áttapotát (extended, retracted) leírására. A futómű kiengedését és visszahúzását, azaz az állapotátmeneteket a következő két esemény segítségével vezették be: *Make_GearExtended* és *Make_GearRetract*.

Az első finomítás során bevezetett részlegesen kiengedett állapot bezezetéséhez szükség volt egy újabb logikai változóra. Továbbá finomítani kellett, valamint ki kellett egészíteni a futóművek mozgatásához szükséges eseményeket.

2.1.2 Az ajtók modellezése: a második és harmadik finomítási lépések

A futóművek modellezéséhez hasonlóan, ez is két szinten történt. Először bevezettek egy logikai változót az ajtó nyitott/ zárt állapotának jellemzésére, majd egy finomítási lépésben bevezettek egy újabb logikai változót, hogy különbséget lehessen tenni a részben nyitott állapot és a zárt állapot között.

Itt szükség volt egy további invariáns bevezetésére

is, miszerint ha a futóművek részlegesen kiengedettek, akkor minden ajtónak nyitva kell lenni, valamint, hogy egy ajtó nem lehet egyszerre nyitva is és zárva is. Ennek értelmében kellett tehát finomítani és kiegészíteni az ajtók nyitásához/zárásához rendelt eseményeket.

2.1.3 A hengerek modellezése: a negyedik finomítási lépés

A futóművek és az ajtók mozgása hengerek segítségével valósul meg. Állapotuk lehet zárt vagy nem zárt. Az ajtókhoz/futóművekhez rendelt hengerek nem lehetnek zárt állapotban, ha ezek mozogni kezdenek. Ezeket figyelembe véve finomították tovább a meglévő eseményeket, illetve a hengerek állapotátmeneteihez szükséges új eseményeket is bevezették ebben a lépésben.

2.1.4 A kar/kapcsoló/lengéscsillapító/hidraulikus kör modellezése: az ötödik finomítási lépés

Első lépésként a kar (up és down) és a kapcsoló (opened és closed) állapotának leírásához szükséges volt a kontextus kibővítése az új szettekkel. A kar és a kapcsolók állapotának megadását szintén logikai változókkal valósították meg. Ezenkívül egy újabb logikai változó bevezetése is szükséges, amely kar állapotának a változását jelöli. Hasonlóan az eddigiekhez, a kar mozgatásához, azaz állapotátmenetéhez ebben a finomítási lépésben is új eseményeket vezettek be.

A hidraulikus kör modellezésére bevezetett logikai változó értéke adja meg, hogy biztosított-e a megfelelő nyomás vagy sem. Az eddigiekhez hasonlóan ezeknek az állapotátváltozását is két esemény segítségével biztosították. Következésképpen minden olyan eseményt finomítani kellet, amely az ajtók/futóművek mozgásával vagy a hengerek zárásával/nyitásával kapcsolatos: ezek csak akkor történhetnek meg, ha a megfelelő nyomás biztosított.

Végezetül a lengéscsillapítók modellezésére is sor került egy újabb logikai változó bevezetésével, amely a különböző pozíciókban levő futóművekhez rendelt lengéscsillapítók állapotát jellemzi (aktív/inkatív). Továbbá egy invariánst vezettek be annak érdekében, hogy a lengéscsillapítók aktívak legyenek, amikor a futómű ki van engedve. Mindezek értelmében tovább kellett finomítani a futóművek kiengedéséhez rendelt eseményeket, valamint bevezetni a lengéscsillapítók állapotátmenetét biztosító eseményeket.

2.2 Az ellenőrzött változók modellezése: a hatodik finomítási lépés

Ez a fejezet részletezi, hogy a szelepek hogyan lesznek aktívak/inaktívak, illetve hogy a fényjelzések milyen feltételeknek megfelelően kapcsolnak ki/be. Ahogy korábban is, a szelepek és fényjelzések állapotának jellemzésére is logikai változókat vezettek be. Továbbá minden szelephez két esemény is társul, ami aktívvá/inaktívvá teszi őket.

Értelemszerűen finomítani kellett az ajtók/futóművek mozgatásához kapcsolódó eseményeket, úgy hogy a megfelelő szelepek aktívak/inaktívak legyenek. Szintén finomítani kellett az ajtókhoz/futóművekhez rendelt hengerekhez tartozó műveleteket is a megfelelő feltételekkel.

A fényjelzőket a szelepekhez hasonló módon vezették be. Mindegyik modellezéséhez felvettek egy logikai változót és két eseményt, amely ki és bekapcsolja őket.

2.3. A bemeneti/kimeneti adatok és a kontroller modellezése: a hetedik finomítási lépés

Ebben a részben derül fény arra, hogy a szenzoroktól kapott értékeknek megfelelően a kontroller hogyan dönti el, hogy ki-/bekapcsoljon egy fényjelzést vagy aktiváljon/deaktiváljon egy szelepet. Ehhez a kontroller a fizikai komponensek állapotát olvassa le. A szerzők bevezettek minden fizikai komponens állapotát jelölő változónak egy ugyanilyen változót, amely értéke a kontroller által észlelt állapotát jelenti. Ezen bemeneti paraméterek alapján dönt a kontroller arról, hogy a szelepeknek és égőknek milyen utasításokat küld.

Ugyancsak ebben a lépésben a szelepek aktiválását/deklarativitást és a fényjelek bekapcsolását/kikapcsolását kiváltó eseményeket is finomították egy feltétellel, ami biztosítja, hogy a változást kiváltó utasítás a kontrollertől érkezett.

2.4. Az időzítési szempontok bevezetése: a nyolcadik finomítási lépés

Ebben a rendszerben az időzítés négy szempont miatt fontos: az analóg kapcsoló ki/bekapcsolása időt vesz fel, a szelepek start/stop stimulációi között el kell telnie valamennyi időnek, valamint a szelepek és hengerek állapotváltozásai, le/fel mozgása is időt igényelnek. A szerzők ebben a tanulmányban csak az első három szemponttal foglalkoztak. Ehhez három diszkrét időzítési tulajdonságot vezettek be:

– határidő (deadline): egy B eseménynek meg kell történnie egy adott időn belül.
– késleltetés (delay): egy B esemény nem történhet meg egy adott időn belül.
– lejárati idő (expire): egy B esemény nem jelenhet meg egy adott idő eltelte után.

Továbbá definiáltak egy *currentTime* változót az aktuális idő jelölésére, és egy *vB* időzítő változót minden olyan eseményhez, amelynek végrehajtási időpontja egy előző esemény végrehajtási időpontjától függ. Például, ha egy B esemény az A esemény után *t* időegység elteltével kell megtörténjen, akkor az A esemény a *vB* időzítő változót *currentTime*+*t*-re kell állítsa. Az idő telik tovább amíg el nem ér a *vB* határidőig. Ha a határidőhöz tartozó B esemény megjelent, akkor a *vB* törlődik és a *currentTime* megkapja a *vB* értékét. Ezeket alkalmazva finomítottak minden olyan eseményt, amire valamilyen időbeli megszorítás vonatkozott. Ebben a lépésnem nem térnek ki arra az esetre, amikot a várt B esemény nem következik be. Ezt a következő finomítási lépés fogja majd kiküszöbölni.

2.5. Meghibásoások modellezése: a kilencedik finomítási lépés

Mivel a fizikai komponensek bármelyike meghibásodhat bármely időpillanatban, a rendszer modellezésekor ezt is figyelembe kell venni. Ennek érdekében minden ilyen komponenshez hozzárendeltek egy-egy eseményt, ami előidézi ezt a meghibásodást és egy-egy logikai változót amely megmondja, hogy az adott komponens éppen meghibásodott állapotban van-e vagy sem.

Ennek következtében finomítani kellett minden olyan eseményt, ami ezen fizikai komponensek állapotátmenetéhez kapcsolódik úgy, hogy ez csak abban az esetben valósulhasson meg, ha az adott komponens épp nincs meghibásodva. A bekövetkezett meghibásodásokat a kontroller feladata észlelni. Ezért jól meghatározott időben a szenzorokon keresztül le kell olvassa a fizikai komponensek állapotát, és ha valami rendelleneset észlel, akkor azt közvetítse a pilótának.

Ezeket is figyelembe véve újabb finomítást kellett elvégezni a modellen, mégpedig úgy, hogy a kontroller csak abban az esetben tudjon utasításokat küldeni a fizikai komponenseknek, ha nem észlelt rendellenességet.

3. A tulajdonságok ellenőrzése: tizedik finomítási lépés

A szerzők a ProB modellellenőrzőt és AnimB animátort használták a finomítás első lépéseinek ellenőrzésére, főleg azért, hogy kiküszöböljék a kezdetleges hibákat még a hosszadalmas és bonyolult bizonyítások elvégzése előtt.

Ha a ProB talál egy ellenpéldát, akkor megad egy olyan egymást követő műveletsort is, amin keresztül a kezdeti állapotból el lehet jutni egy érvénytelen állapotba. Így ezekben az esetekben könnyen visszakövethető volt a probléma, és kiküszöbölhető plusz feltételek bevezetésével vagy az események módosításával. Ezzel a módszerrel tesztelték például az alábbi invariánsok teljesülését:

INVARIANTS

inv16: $\neg(open\_EV{=}\text{TRUE} \wedge close\_EV{=}\text{TRUE})$ //Req R_{41}

inv17: $\neg(extend\_EV{=}\text{TRUE} \wedge ret\_EV{=}\text{TRUE})$ //Req R_{42}

inv18: $(open\_EV = \text{TRUE} \vee close\_EV = \text{TRUE}) \Rightarrow general\_EV = \text{TRUE}$

inv19: $(extend\_EV = \text{TRUE} \vee ret\_EV = \text{TRUE}) \Rightarrow open\_EV = \text{TRUE}$ //$inv18 + inv19 = R_{51}$

Ezek az invariánsok biztosítják, hogy normál üzemmódban:

R41.: az ajtó nyitását/zárását biztosító szelepek ne legyenek egyszerre stimulálva

R42.: a futóművek kiengedését/visszahúzását biztosító szelepek ne legyenek egyszerre stimulálva

R51.: a mozgató (ajtót nyitó/záró, futóművet kiengedő/visszahúzó) szelepeket ne lehessen stimulálni az általános szelep stimulálásán nélkül; a futóművek mozgatásához hozzájáruló szelepet ne lehessen stimulálni az ajtó nyitását biztosító szelep stimulálása nélkül.

Az AnimB egy animátor bővítmény, ami lehetővé teszi bizonyos forgatókönyvek lejátszását, és így ellenőrizni lehet a modell viselkedését az elvárásokhoz képest. Ilyenkor minden lépésben megfigyelhetők a változók értékei, és az, hogy mely események végrehajthatóak és melyek nem az adott lépésben. Ezt használták az alábbi két követelmény ellenőrzésére:

R11: normál üzemmódban, ha a kar le van nyomva, és ebben az állapotban is marad, akkor a futómű kiengedett állapotba blokkolódik, és az ajtók a kar lenyomásától számított kevesebb mint 15s alatt becsukódnak.

R12: normál üzemmódban, ha a kar fel van húzva, és ebben az állapotban is marad, akkor a futómű visszahúzott állapotba záródik, és az ajtók a kar felhúzásától számított kevesebb mint 15 s alatt becsukódnak.

Ezek a dinamikus tulajdonságok két állapotot kötnek össze, melyek között rengeteg átmeneti állapot valósulhat meg. Az ilyen tulajdonságok modell ellenőrzőkkel vagy tétel bizonyítókkal való igazolása nagyon nehéz lenne, mivel a bizonyítás során rengeteg átmeneti tételt is bizonyítani kellene.

Ez az eszköz segített a szerzőknek kiválasztani a megfelelő megközelítést az időzítési szempontok bevezetésére, mivel rávilágított egy előzőleg alkalmazott megközelítésnek ama hiányosságaira, hogy nem lehet kezelni azokat az eseteket, amikor több eseménynek is ugyanaz a határideje.

Összességében elmondható, hogy az AnimB nagyon hasznos eszköznek bizonyult a modellellenőrzésre a modell kezdeti fázisában, azonban ahogy a finomítások során egyre több változót és eseményt tartalmazott a modell, az AnimB nagyon lelassult, egyre kevésbé lehetett hatékonyan használni.

A további ellenőrizendő követelmények nagy része időbeli tulajdonság, ezek nem invariánsok által írhatóak fel, hanem kikövetkeztetettek a rendszer viselkedéséből. Például kikövetkeztethető, hogy a kar mozgatása után kevesebb mint 15s múlva a futóművkenek ki kell engedődniük, mivel az ajtók nyitásához a hengerek nyitott állapotba kerüléséhez és mozgásához szükséges összes idő kevesebb mint 15s.

Vizsgáljuk példaként az alábbi követelményeket:

R74: Ha a futóművek egyike nincs leengedett állapotba blokkolva több mint 10s-mal a megfelelő szelep stimulálása után, akkor rendellenességet detektál a kontroller.

Ennek az ellenőrzéséhez egy új változó bevezetése volt szükséges, amelyben tárolták, hogy mikor volt stimulálva az adott szelep: *TimeStimulExtRetEv*, így a tulajdonság a következőképpen írható le:

$$\left(\begin{array}{c} currentTime > TimeStimulExtRetEv + 100 \\ \wedge \\ extend\_EV = \texttt{TRUE} \end{array}\right) \Rightarrow \left(\begin{array}{c} anomaly = \texttt{TRUE} \\ \vee \\ gear\_ext\_ind = PositionsDG \times \{\texttt{TRUE}\} \end{array}\right)$$

Ennek a teljesítéséhez az alábbi két tételt kell bizonyítani:

$$\left(\begin{array}{c} currentTime > TimeStimulExtRetEv + 100 \\ \wedge \\ extend\_EV = \texttt{TRUE} \end{array}\right) \Rightarrow nextInputGearEndExtRet = 0$$

és

$$\left(\begin{array}{c} nextInputGearEndExtRet = 0 \\ \wedge \\ extend\_EV = \texttt{TRUE} \end{array}\right) \Rightarrow \left(\begin{array}{c} anomaly = \texttt{TRUE} \\ \vee \\ gear\_ext\_ind = \texttt{PositionsDG} \times \{\texttt{TRUE}\} \end{array}\right)$$

Az első tétel biztosítja, hogy ne telhessen az idő a határidőn túl, anélkül hogy a futóművek állapotát leolvasná a kontroller. A második kimondja, hogy ha az idő meghaladta a határidőt, akkor vagy minden futómű kiengedett állapotban van, vagy pedig rendellenességet detektál a kontroller.

R21: Normál üzemmódban, ha a kar a Down pozícióban marad, akkor a futóművek nem húzódhatnak vissza. Ennek a tulajdonságnak a megadásához bevezették a *stayDown* logikai változót, aminek az értéke akkor igaz, ha a kar Down állapotban maradt több mint két kontroller cikluson keresztül, valamint egy *end_cycle* logikai változót, amit igaz értékkel inicializáltak, és akkor válik hamissá, ha a kontroller változást észlelt a kar állapotában. Ezek alapján az előbbi tulajdonság a következőképpen írható fel:

$$\left(\begin{array}{c} stayDown = \texttt{TRUE} \\ \wedge \\ end\_cycle = \texttt{TRUE} \end{array}\right) \Rightarrow retract\_EV = \texttt{FALSE}$$

Ehhez a következő tétel igazolása szükséges:

$$handle\_ind = up \Rightarrow stayDown = \texttt{FALSE}$$

Ez tulajdonképpen azt ellenőrzi, hogy a *stayDown* változó frissítése helyes-e: ha a rendszer Up állapotban látja a kart, akkor a *stayDown* változó értékét hamisra kell állítani.

4. Következtetések:

A rendszer modellezése során a szerzők 66 változót és 48 eseményt vezettek be 10 finomítási lépés során. Mindez 285 bizonyítandó tételhez vezetett. Ezeknek 72%-a automatikusan igazolódott. A fennmaradt rész bizonyítását a szerzők Atelier B és SMT bizonyítók segítségével végezték el interaktívan. Ezek az eszközök nem csak a tulajdonságok ellenőrzésében bizonyultak hasznosnak, hanem már a modell fejlesztése során is.

A modellezés során a fő nehézséget az okozta a szerzőknek, hogy egy olyan modellt határozzanak meg, ami a rendszer komplexitását kezelni tudja. Ez a rendszer úgy tekinthető mint két alrendszer, a kontroller és a környezet, amelyek egymással kölcsönhatnak. A kihívás az volt, hogy a két rendszer határait azonosítsák és meghatározzanak egy tervezési megközelítést a formális modell építéséhez. A négyváltozós modell és az EventB finomítási folyamatának kombinációja relevánsnak bizonyult az ilyen rendszerek esetén.