REPÜLÉSTUDOMÁNYI KÖZLEMÉNYEK
35. évfolyam (2023) 1. szám 59–69. • DOI: 10.32560/rk.2023.1.5

Schuster György

A fejlesztésben használt szoftvermodellek alkalmazása és minősítése

A feladatok komplexitásának növekedésével a fejlesztőknek különböző eszközöket kell használni. Erről egy előző publikációban már részletesen beszéltünk. Ebben a cikkben egy speciális területet érintünk, amely napjainkban egyre inkább előtérbe kerül, ez a modellek alkalmazásának kérdése. A modellek alkalmazása biztonságkritikus rendszerekben azonban felveti a modellek megbízhatóságának kérdését, illetve ezeknek a szoftvereszközöknek a minősítési kérdéseit.

Kulcsszavak: *modellalapú fejlesztés, modellek minősítése*

1. Bevezetés

A modern eszközök összetettsége már olyan szintű, hogy klasszikus tervezési és konstrukciós szempontból áttekinthetetlenek. A tesztelés szempontjából is fontosak a modellek, mivel fizikai eszköz nélkül tudunk ellenőrizni különböző szoftverelemeket, illetve a teljes szoftverrendszer működését. Ez mind a fejlesztési időben, mind költség szempontjából kedvező.

A modellalapú fejlesztés olyan módszer, amely a szoftverfejlesztés során a számítógépes modellek használatát jelenti, vagy egy adott specifikus területen, vagy egy teljes rendszert tekintve.

A modell egy rendszer adott szempontkészletének, esetlegesen fizikai felépítésének és működésének absztrakt reprezentációja, amelyet elemzéshez, verifikációhoz, szimulációhoz, kódgeneráláshoz vagy bármely más, a rendszerrel kapcsolatos tevékenységhez használhatunk.

Ha a fejlesztés célja egy biztonságkritikus rendszer létrehozása, akkor a felhasznált modelleknek pontosaknak, megbízhatóknak kell lenniük, és megfelelő minősítéssel kell rendelkezniük.

A minősített modellező eszközök használata, amelyek automatikusan generálnak kódot, sőt bizonyos esetekben automatikusan generálnak tesztvektorokat, egy viszonylagosan új paradigmaváltás a járműfedélzeti szoftverek gyártásában.

A modellek alkalmazása hatalmas előnyökkel jár, illetőleg járhat, de nem mentes a kockázatoktól sem.

Ebben a cikkben a modellalapú fejlesztés előnyeit és hátrányait egyaránt megvizsgáljuk mind a biztonság, mind tanúsítás szempontjából.

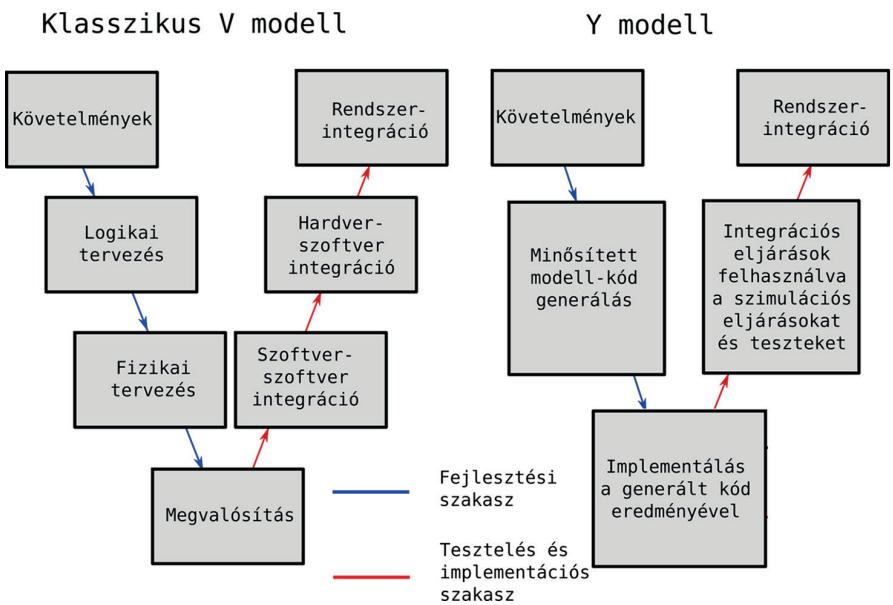
Ezenkívül rövid áttekintést nyújtunk a DO-331-ről, mint a DO-178C és DO-178A szabványok modellalapú fejlesztési és ellenőrzési kiegészítéséről.

2. A modellalapú fejlesztés szintjei, előnyei és problémái

A modellalapú fejlesztésnek és ellenőrzésnek számos lehetséges előnye van. Az előnyök kihasználásának képessége a megvalósítás részleteitől függ. Ezek az előnyök a következők.

2.1. A modellalapú fejlesztés előnyei

1. Áttérhetünk V életciklusmodellről Y életciklusmodellre.



1. ábra
Egyszerűsített V és Y életciklusmodellek [a szerző]

A modellalapú fejlesztés egyik fő motivátora a hagyományos V életciklusról Y életciklusra való átállás képessége, amely csökkenti a fejlesztési időt, költséget és esetleg még az emberi hibákat is.

A rendszerkövetelményekre való összpontosítással és az automatizálással az életciklus lerövidülhet. Egyes becslések 20%-os csökkenést mutatnak minősítetlen kódgenerátor használata esetén, és akár 50%-os csökkenést minősített kódgenerátor használatakor.

Számos módja van a modellek bevezetésének a termék életciklusába. Előállíthatók a rendszer szintjén, a szoftverkövetelmények szintjén és a szoftvertervezés szintjén. A legtöbb vállalat megpróbál egy platformfüggetlen szintű modellt létrehozni, így azt hordozhatóbbá teszi. A platformspecifikus részletek az alacsonyabb szintű modellben is megjeleníthetők.

2. Nagyobb összpontosítás a követelményekre.
Ismert, hogy a követelményekkel kapcsolatos hibák gyakran a funkcionális szoftverhibák okai. Ismert tény az is, hogy minél később találják meg a követelményhibát, annál költségesebb annak javítása.

Ha egy modell a követelményeket reprezentálja, akkor a követelmények rögzítése során tisztább képet ad a rendszerről és a szoftver működéséről, mint a szokásos szöveges követelményleírások.

Ebben nagy segítséget nyújtanak a grafikus tervezőeszközök, különösen azok, amelyek automatikus verifikációt is biztosítanak. Így könnyebb az aktuális részletekre összpontosítani, és nem veszünk el az adott szinten még lényegtelen részletekbe.

Ha mégis fellépne valami hiba a tervezés során, akkor a hiba gyökerének feltárása is hatékonyabb lehet.

3. A korai ellenőrzés lehetősége.
Ha minősített modell-kód eszközöket használnak, a hangsúly a kódról a modellre tolódik át. A modellszimuláció és az automatikus teszteset-generátorok használata elősegíti a korai ellenőrzést, lehetővé téve a követelmények magasabb érettségi szintjét és az esetleges hibák felismerését.

Az ellenőrzés gyakran a szoftverfejlesztési erőforrások 50–70%-át veszi igénybe, a hibák korábbi észlelése és az emberi ellenőrzési munka csökkentése a projekt költségvetése és ütemezése szempontjából előnyös lehet.

4. A szükségtelen redundancia kiküszöbölése.
A hagyományos szoftverfejlesztés három követelményszintet alkalmaz:
   - a szoftverhez rendelt rendszerkövetelmények;
   - magas szintű szoftverkövetelmények[1] és
   - alacsony szintű szoftverkövetelmények[2].

A követelmény rögzítési módjától függően jelentős redundancia léphet fel e szintek között. Ez a szinte elkerülhetetlen változtatások során következetlenségekhez vezethet.

A minősített modellezőeszközök használata csökkentheti ezeket a redundanciákat és következetlenségeket. Ha a modelleknek több szintje is van, akkor könnyen jelentős mennyiségű redundancia, illetve következetlenségek is előfordulhatnak.

Ezért a többi említett előnyhöz hasonlóan ez is az alkalmazott megközelítéstől függ.

5. A követelmények jobb érthetősége.
Az eszközök olyan modelleket állítanak elő, amelyek könnyebben érthetők, mint a szöveges követelmények. Ezért elősegíthetik a nagyobb pontosságot, következetességet és teljességet. A modellek intuitívabbak lehetnek, mint a szöveges követelmények, így könnyebben észlelhetők a hibák és a tervezett viselkedés hiányosságai.

Azonban tapasztalatunk szerint nem minden modellezőeszköz hoz létre könnyen érthető modelleket, és néhány generált modell meglehetősen zavaró lehet. A modellezőeszközöket ezért körültekintően kell kiválasztani.

[1] High-level Requirements, HLR.
[2] Low-level Requirements, LLR.

Egy jó modellezési szabvány, illetőleg előírás elengedhetetlen annak biztosításához, hogy minden fejlesztő megfelelően használja a modelljelöléseket, szimbólumokat és eszközöket.

6. Jobb interakció az ügyfelekkel.
A modellek vizuálisabb természete miatt hasznosak lehetnek az ügyfelek és a tervezők véleményének megszerzésében. Javíthatja a kommunikációt.

7. A növekvő eszköztámogatás.
Léteznek eszközök a kódgenerátorok kimenetének ellenőrzésére. Ha az egyik eszközbe vetett bizalom nem teljes, akkor számos esetben lehetséges eltérő eszköz használata.
Egyes kritikusabb rendszerek – például járműirányítás – esetében ez akkor is ajánlott és hasznos módszer lehet, ha megbízunk az eszközökben, ezzel csökkenthető a bizonytalanság és növelhető az eszközbe vetett bizalom.

8. A formális módszerek fejlesztése.
A modellezési technológia alkalmazása, az eszköztámogatás és a formális módszerek konvergenciája jelentősen növelheti a modellalapú fejlesztés és ellenőrzés előnyeit.
Ha egy modellező eszközkészlet – beleértve a kódgenerátort is – helyességét formálisan ellenőrzik, várhatóan az eszköz megbízhatóan teljesíti a kívánt funkciót.
Formális ellenőrzés nélkül mindig felmerül az a kellemetlen érzés, hogy a modellezőeszköz vagy a mögöttes kódgenerátor hibát vihet be a fejlesztési folyamatba.

2.2. A modellalapú fejlesztés problémái

Mint minden technológia esetében, a modellalapú fejlesztés és ellenőrzés használatakor számos lehetséges probléma és kihívás adódhat. Ezek közül néhányat itt megemlítünk.

1. A többszörös követelmény felülvizsgálatának figyelmen kívül hagyása.
Amikor a modelleket automatikus kódgenerálással valósítják meg, a követelmények hagyományos ellenőrzésének egy része elveszhet, különösen, ha a modellt rendszerszinten vezetik be.
A hagyományos szoftverfejlesztés során áttekintik a rendszerkövetelményeket, a szoftverkövetelményeket, a tervezést és a kódot. Ezen túlmenően, mivel minden egyes követelményszintet elemeznek az alacsonyabb szint létrehozásához, a problémákat azonosítják és megoldják. Például a kódot ellenőrző személy ellentmondást találhat a követelményekkel.
Tapasztalt szakemberek, akik kódot terveznek, fejlesztenek vagy teszteseteket határoznak meg, biztosítják a szoftverrendszer informális validálását. Miközben ezek a mérnökök megvizsgálják az adatokat, hibákat észlelnek, és biztosítani tudják a megfelelő korrekciós intézkedéseket. Ha azonban a modellt magasabb szintre emelik, és a megvalósítási folyamatot eszközök kezelik, akkor ez a „biztosíték" csökkenhet, esetleg teljesen elvész, mert a tervezők túlságosan távol kerülnek a kódtól.

2. A rendszermérnöki szerepkör bővítése.
Előfordulhat, hogy a rendszermérnökök nem ismerik a követelmények lefedettségének, a robusztussági tesztelésnek, a modell lefedettségelemzésének adott fogalomkörét. Ezt tapasztalataink alátámasztják. A probléma megoldása érdekében célszerű a rendszermérnöki csapatot tapasztalt szoftvermérnökökből álló szakértői csapattal kiegészíteni.

3. A nyomon követhetőség nehézkes.
Habár a követelmények grafikusan modellként jelennek meg, akkor is követniük kell a magasabb és alacsonyabb szintű követelményeket.
A magasabb szintre és az alsó szintre vezető útvonalaknak kétirányúnak kell lenniük. Egyes modellezőeszközök sajnos nem támogatják megfelelően a követhetőséget. A szöveges követelmények és a modellalapú követelmények megfeleltetése komoly kihívás lehet.

4. A teszt teljességével kapcsolatos nehézségek.
A modell jellemzőitől és a tesztcsoport tapasztalataitól függően nehéz biztosítani, hogy a modell által reprezentált követelmények teljes mértékben igazolhatók legyenek.
Az összetett modellek teljes körű ellenőrzése különösen nagy kihívást jelenthet. Részletes tesztelési irányelvekre van szükség annak biztosításához, hogy a követelményeken alapuló tesztek teljes mértékben igazolják a modellt.

5. A szimulációs eljárások hitelességének kérdése.
A modellalapú fejlesztés és ellenőrzés egyik mozgatórugója a szimulációs eszközök használatának lehetősége a hibák mielőbbi azonosítása érdekében.
A DO-331 megpróbálja tisztázni, hogy mi várható és mi megengedett a szimulációhoz. A DO-331 bevezetése előtt egyedi eljárások folytak a tanúsító hatóságokkal, és természetesen a DO-331 alkalmazása esetében is szoros együttműködésre van szükség velük.

6. A modellek gyakran keverik, a „mit" és a „hogyan"-t.
Mivel a modellek gyakran tartalmaznak részleteket a megvalósításról, néha nehéz lehet felismerni, mit csinál a modell.
Alapvetően a tervezés és a követelmények keveredhetnek. Ez gyakran probléma a hagyományos fejlesztésben is. Sajnos a modellek alkalmazásának esetén ez gyakran előfordul. Ezért a DO-331 erre a problémára előírásokat tartalmaz.

7. A specifikációs modellek elkülönítése a tervezési modellektől.
A DO-331 útmutatást ad a specifikációs modellekhez és a tervezési modellekhez. Az útmutató egyértelműen kimondja, hogy a modellek nem kombinálhatók.
A valóságban nehéznek bizonyulhat a specifikáció és a tervezési modellek megkülönböztetése, különösen akkor, ha a már meglévő modelleket használják. A modellek többsége bizonyos fokig ezeket összevonja, ami valamilyen mértékben hasznos lehet, de néha zavaró.
Ezenkívül a legtöbb modellezési technika ösztönzi a tervezési részletek, például egyenletek és algoritmusok beépítését. Nagyrészt ezért is használjuk a modelleket, de ez inkább a kódolási szakaszban hasznos.
A specifikációs modellek és a tervezési modellek elkülönítése a legtöbb szervezet számára jelentős paradigmaváltást igényel.

8. A rendszer- és a szoftverszerepek összemosása.
Attól függően, hogy a modellt hol vezetik be, előfordulhat, hogy egy rendszert tervezők a hagyományos eljárásnál jobban részt vesznek az alacsonyabb szintű tervezésben, a szoftvertervezők pedig a hagyományosnál jobban belefolynak a magasabb szintű tevékenységekbe.

Ha ezt a megfelelő módon teszik, akkor előnyös lehet a rendszermérnökök és a szoftvermérnökök közötti együttműködés szorosabb kialakításában.

9. A hagyományos és a modellalapú fejlesztést ötvöző kihívások.
A legtöbb fejlesztési projektben a modelleken kívül hagyományos tervezési módszereket is alkalmazni kell. Ez igaz a kódgenerátorok esetén is.

Elkerülendő a későbbi problémákat a hagyományos és a modellalapú fejlesztés alkalmazásának területein, gondosan meg kell határozni a határterületeteket azért, hogy a fejlesztés következetes legyen, ne maradjon ki semmi, és kerüljük az átfedéseket.

Hasonlóképpen, a DO-178C és a DO-331 szabványokat egyaránt használni kell a hagyományos (DO-178C) és a modellalapú (DO-331) fejlesztési tevékenységben.

A tervezés során egyértelműen meg kell határozniuk a módszerek alkalmazásának területeit és integrálásuk módját.

10. Inkonzisztens modellértelmezés.
Ha a modellalkalmazás fogalomrendszere nem jól definiált és helyesen implementált, az a modell hibás, nem következetes értelmezéséhez és alkalmazásához vezethet.

A jól definiált modellezési szabványok és azok szigorú betartása nagy valószínűséggel megoldja ezt a problémát.

11. A modell karbantartása.
Megfelelő dokumentáció nélkül előfordulhat, hogy a modellek nem karbantarthatók. Egy modell alkalmazása, karbantartása és tanúsítása nem lehet személyhez kötött.

A modell minden szempontból dokumentált kell hogy legyen.

12. Az automatikus tesztgenerálás problémája.
Egyes modellezőeszközök nemcsak kódot generálnak automatikusan, hanem teszteket is. Ez előnyös az informális ellenőrzés és a modellbe vetett bizalom kialakítása szempontjából. Ha azonban egy projekt automatikusan generál teszteket és a fejlesztő ezt a tanúsításra is használni akarja, az számos problémát okozhat.

Ha a modell hibás, előfordulhat, hogy a hiba nem észlelhető időben. A géppel generált tesztek nem biztos, hogy egyes területeken olyan hatékonyak, mint az emberi tesztelők. Viszont sokban segítik a tesztelés folyamatát.

Az automatikusan generált tesztek használatakor biztosítani kell a megfelelő függetlenséget. Ez azt jelenti, hogy adott szoftverszintekhez a tesztgenerátort és a kódgenerátort egymástól függetlenül kell fejleszteni.

13. Az eszköz instabilitása.
A modellalapú fejlesztés és az eszközök ellenőrzése miatt az eszközök instabilitása problémás lehet. Az „éretlen" eszközök gyakran változhatnak, ami a modellek változtatását

és újbóli ellenőrzésének szükségességét eredményezheti. Egy eszköz változásának hatása különböző mértékű hatással lehet a rendszerre, különösen biztonságkritikus területen.

14. A modellezés korlátai.
Nem minden rendszer alkalmas a modellezésre. A modellalapú fejlesztés és ellenőrzés nem minden projekt vagy szervezet számára megfelelő megközelítés. Számos állítás ellenére nem minden esetben ez a végső megoldás.

3. A DO-331 szabvány áttekintése

A DO-331 szabvány a DO-178C és DO-178A szabványok modellalapú fejlesztési és ellenőrzési kiegészítése volt. Ezt az RTCA 205. számú Különbizottság (SC-205) és az EUROCAE munkacsoportja fejlesztette ki [10].

A DO-331 útmutatás igyekszik rugalmasságot biztosítani a modell megvalósításához, ugyanakkor biztosítja, hogy a modell segítségével generált szoftver a rendeltetésszerű működését és csak azt teljesítse. A DO-331 is a DO-178C-ből indul ki, és szükség szerint módosítja, helyettesíti vagy hozzáadja a DO-178C-hez a szükséges célokat, tevékenységeket és útmutatást [12].

A DO-331 felismeri, hogy az adott modell rendszerkövetelményként, szoftverkövetelményként vagy szoftvereszközként kerülhet be a szoftver életciklusába. Azt is figyelembe veszi, hogy a modellabsztrakciónak több szintje is lehet.

A DO-331 kétféle modellt határoz meg [10]:

1. a specifikációs modell magas szintű követelményeket képvisel, ezek:
   - funkcionális követelmények;
   - teljesítménykövetelmények;
   - interfészkövetelmények;
   - biztonsági követelmények.

   A specifikációs modellnek ezeket egyértelműen ki kell fejeznie.

   A specifikációs modellek nem határozhatnak meg olyan szoftvertervezési részleteket, mint adatszerkezetek és algoritmusok. Tehát csak és kizárólag a magas szintű követelményekkel foglalkozhatnak.

2. a tervezési modell alacsony szintű követelményeket és architekturális kérdéseket tartalmaz [3].

Ha egy modell szoftvertervezési adatokat fejez ki, függetlenül az egyéb tartalomtól, akkor azt tervezési modellnek kell minősíteni.

A DO-331 megtartja a DO-178C útmutatásainak nagy részét, de hozzáad néhány modellspecifikus információt. Általánosságban elmondható, hogy a DO-178C-ben szereplő, a magas szintű követelményekre vonatkozó útmutatások többsége a specifikációs modellekre vonatkozik, a szoftvertervezési útmutató pedig a tervezési modellekre [4].

A DO-331 és a DO-178C közötti legjelentősebb különbségeket az alábbiakban foglaljuk össze. Ezek azok a területek, ahol a DO-331 módosítja vagy pontosítja (azaz kiegészíti) a DO-178C-t a modellalapú fejlesztés és ellenőrzés során [2]. Ezek az alábbiak.

1. Modelltervezés.
   A tervezési szakaszban a terveknek tisztázni kell a modellezés alkalmazási körét és azt, hogy az hogyan illeszkedik a szoftver életciklusába. Továbbá a terveknek meg kell határozni, hogy az egyes modellek milyen szoftveréletciklus-szinteket képviselnek, milyen modellszabványokat fognak használni, és a tervezett ellenőrzési megközelítést. Ha szimulációt használnak a minősítéshez, a terveknek egyértelműen részletezniük kell azt a pontos megközelítést, hogy milyen minősítési szempont szerint végzik ezt. A tervezési szakasznak meg kell határoznia a modell szimulációs környezetét is, beleértve a módszereket, eszközöket, eljárásokat és működési környezetet.

2. Modellszabványok.
   Minden használt modelltípusnak rendelkeznie kell megfelelő szabványokkal, amelyek meghatározzák és ismertetik a modellezési technikákat, megszorításokat és eljárásokat. A DO-331 szerint minden szabványnak a következő minimális információkat kell tartalmaznia [10]:
   - az alkalmazandó módszerek és eszközök ismertetése és használatuk indoklása;
   - a használandó modellezési eszköz jellemzőinek szintaktikai, szemantikai ismertetése;
   - a bonyolultsági korlátozások, amelyek lehetővé teszik, hogy a modellezési megközelítés és annak megvalósítása egyértelmű, determinisztikus legyen és megfeleljen a DO-331 célkitűzéseinek;
   - a bonyolultsági korlátozások különösen fontosak. Általában korlátozni kell a modell mélységét (beágyazási szintek), a strukturális rétegek számát [1];
   - a megszorítások felsorolása és ismertetése a modellezőeszközök és a támogató könyvtárak megfelelő használatának biztosítására [10];
   - a követelményrétegek közötti kétirányú nyomon követhetőség létrehozása, a származtatott követelmények azonosítása, az összes származtatott követelmény indoklásának dokumentálása.

3. Modellkönyvtárak.
   A legtöbb modellező eszközben széles körben használják a könyvtárakat. Például a modell grafikus ábrázolására használt szimbólumok egy szimbólumkönyvtárból származnak. Minden egyes modellben használható könyvtárelemnek biztosítani kell a megfelelő szoftverszintet a DO-178C szerint. Alapvetően a könyvtári elemekhez szükség van tervekre, fejlesztési szabványokra, követelményekre, tervezésre, kódra, ellenőrzési esetekre és eljárásokra, mint minden más biztonságkritikus szoftverre. Ha vannak olyan elemek a könyvtárban, amelyek nem rendelkeznek megfelelő szintű bizonyossággal, akkor azokat nem szabad használni.

4. Modell-lefedettség elemzése tervezési modellekhez.
   A DO-331 a következőképpen határozza meg a modell-lefedettség elemzését: olyan elemzés, amely meghatározza, hogy a tervezési modell által kifejezett követelmények mely követelményeket nem érvényesítették a tervezési modell kidolgozásának követelményei alapján való ellenőrzéssel. Ennek az elemzésnek az a célja, hogy támogassa a nem szándékolt funkció észlelését a tervezési modellben, ahol a modell kifejlesztésének követelményeit a verifikációs esetekkel sikerült lefedni [6].

5. Modellszimuláció.
A DO-331 szószedet a következőképpen határozza meg a modellszimulációt és a modellszimulátort [9].

Definíció: a modellszimuláció: a modell viselkedésének megvalósítása.

Definíció: a modellszimulátor: olyan eszköz, számítógépes program vagy rendszer, amely lehetővé teszi egy modell végrehajtását, hogy demonstrálja annak viselkedését az ellenőrzés, illetve a tanúsítás támogatása érdekében.

Megjegyzés: A modellszimulátor célja az, hogy végrehajthasson egy olyan kódot, amely a végleges tárgykódot reprezentálja.

A DO-331 konkrét útmutatást ad a modellszimulációhoz. Modellszimuláció használható a DO-331 ellenőrzési, felügyeleti és tesztelési célkitűzéseinek támogatására, illetőleg teljesítésére. Ha szimulációs eseteket és eljárásokat használunk a formális ellenőrzési jóváíráshoz, a szimulációs esetek és eljárások helyességét ellenőrizni kell, a szimuláció eredményeit szintén ellenőrizni kell, és értékelni kell az esetleges eltéréseket.

4. Alkalmazási példák

Számos példa van a járműfejlesztésben a modellalkalmazásra. Ezeknek a felsorolására nincs lehetőség, ezért, a teljesség igénye nélkül, az A380 fejlesztésében alkalmazott modelleket említjük. Ezek az alábbiak [10]:

- repülésvezérlési rendszerek;
- robotpilóta;
- felügyelő és figyelmeztető rendszerek;
- a pilótafülke-kijelző rendszerei;
- üzemanyag-gazdálkodás rendszerei;
- kormányvezérlés;
- terhelésirányítás.

A tesztelésben alkalmazott eljárások közül leginkább a „software in the loop" és a „hardware in the loop" eljárásokat érdemes megemlíteni, ahol az elsőben még a hardvert is modellezzük a rendszer mellett, a másodikban a már kész hardvert szoftverrel együtt, a kapcsolódó rendszer modelljével vizsgáljuk [5], [9].

5. Összefoglaló

Az avionikai szoftverfejlesztés egyik legfontosabb és legmeghatározóbb szabványa a DO-178C, amely előírja az érintett fejlesztőknek a szoftverfejlesztés során követendő szigorú szabályokat. A DO-178C kiegészítői további iránymutatásokat adnak a szabvány alkalmazásához bizonyos területeken [10].

Az egyik ilyen kiegészítő a DO-331, amely a Model-Based Development and Verification Supplement (MBDVS) címet viseli. A DO-331 a modellalapú fejlesztési és verifikációs módszerek alkalmazását szabályozza, és azt írja elő, hogy a fejlesztőknek a szoftvertervezés során

alkalmazniuk kell a modellezési módszereket, valamint a modellekből kiinduló verifikációs tevékenységeket.

A DO-331-et azonban nehéz alkalmazni, különösen akkor, ha a fejlesztők már meglévő modelleket szeretnének felhasználni a szabvány szerinti fejlesztéshez és verifikációhoz. Az ilyen modellek esetében nehéz lehet meghatározni, hogy milyen mértékben kell alkalmazni a modellezési és verifikációs előírásokat, és hogyan kell összehangolni a meglévő modelleket a DO-178C követelményeivel [7], [8].

Az ilyen kihívások miatt a DO-331 alkalmazása nagyobb szaktudást és erőforrásokat igényelhet, és a fejlesztőknek előzetesen meg kell vizsgálniuk a meglévő modellek alkalmazhatóságát és azok összhangját a DO-178C szabvánnyal. Emellett ajánlott, hogy a fejlesztők konzultáljanak szakértőkkel vagy tanácsadókkal a DO-331 alkalmazásával kapcsolatban, és a modellezési és verifikációs előírásokat a lehető legjobban illesszék a meglévő modellekhez.

Összességében, bár a DO-331 fontos előírásokat tartalmaz a modellezési és verifikációs tevékenységekhez, kihívást jelenthet a már meglévő modellek esetében, és a fejlesztőknek alaposan fel kell készülniük az alkalmazására.

A modellek alkalmazása azonban elvitathatatlan előnyökkel rendelkezik, ha a cikkben leírt szempontokat figyelembe vesszük, és ha a modellek alkalmazását a helyén kezeljük, a fejlesztési és a verifikációs folyamat gyorsabb és hatékonyabb lehet, mint a klasszikus módszerek alkalmazása [11].

Felhasznált irodalom

[1] Hardware in the Loop. *Blackberry/QNX,* [é. n.]. Online: https://blackberry.qnx.com/en/ultimate-guides/cloud-computing/hardware-in-loop

[2] Model Based Design Overview for System Development. *Collimator,* 2022. november 11. Online: www.collimator.ai/post/model-based-development

[3] Hardware-in-the-Loop for embedded system developers. Compact and fast for earlier testing. *Protos,* [é. n.]. Online: www.protos.de/en/products/minihil/

[4] Sz. n., *Model based software architectures.* [é. n.]. Online: www.pvpsiddhartha.ac.in/dep_it/lecture%20notes/SPM/unit3.pdf

[5] Basics of Hardware-in-the-Loop simulation. *Mathworks,* [é. n.]. Online: www.mathworks.com/help/simscape/ug/what-is-hardware-in-the-loop-simulation.html

[6] M. Woodside, et al., „Transformation Challenges: From Software Models to Performance Models," *Software & Systems Modeling,* 13. évf. 4. sz. pp. 1529–1552. 2014. Online: https://doi.org/10.1007/s10270-013-0385-x

[7] A. R. Puerta, „A Model-based Interface Development Environment," *IEEE Software,* 14. évf. 4. sz. pp. 40–47. 1997. Online: https://doi.org/10.1109/52.595902

[8] Wikipedia, *Model-based Design.* [é. n.]. Online: https://en.wikipedia.org/wiki/Model-based_design

[9] dSPACE Group, Linux-Based V-ECUs in Software-in-the-Loop. *YouTube,* 2022. Online: www.youtube.com/watch?v = PheVfABN2B8

[10] L. Rierson, *Developing Safety-critical Software. A Practical Guide for Aviation Software and DO-178C Compliance.* Boca Raton, CRC Press, 2013.

[11] K. Schmiechen et al., „A Requirements Management Template in Polarion for Model-Based Development of Airborne Systems," in *Software Engineering 2021 Satellite Events, Lecture Notes in Informatics (LNI),* S. Götz et al. szerk., Bonn, Gesellschaft für Informatik, 2021. Online: www.researchgate.net/publication/358646267
[12] V. Hilderman, DO-331 Introduction – Model Based Development. *AFuzion,* 2014. Online: https://afuzion.com/do-331-introduction-model-based-development/

Application and Qualification of Software Models Used in Development

As the complexity of tasks increases, developers have to use different tools. We also talked about this in detail in a previous publication. In this article, we will discuss a special area that is increasingly coming nowadays, which is the application of models. However, their application in safety-critical systems development raises the question of the reliability of the applied models and the questions of the certification of these special software tools.

Keywords: *model-based development, model certification*

Dr. Schuster György
docens
Óbudai Egyetem
Kandó Kálmán Villamosmérnöki Kar
Elektronikai és Kommunikációs Rendszerek Intézet
Műszertechnikai és Automatizálási Tanszék
schuster.györgy@kvk.uni-obuda.hu
orcid.org/0000-0002-8573-3670

György Schuster, PhD
Associate Professor
Óbuda University
Kandó Kálmán Faculty of Electrical Engineering Institute of Electronic and Communication Systems Department of Instrumentation and Automation
schuster.györgy@kvk.uni-obuda.hu
orcid.org/0000-0002-8573-3670