**REPÜLÉSTUDOMÁNYI KÖZLEMÉNYEK**
34. évfolyam (2022) 2. szám 19–28. • DOI: 10.32560/rk.2022.2.2

Schuster György

# Fejlesztésben alkalmazott szoftvereszközök minősítése

*A szoftverek fejlesztéséhez sok esetben segédeszközöket célszerű vagy kell használnunk. Ezek a segédeszközök az esetek 98%-ában speciális szoftvereszközök. A szoftvereszköz tehát egy speciális program, amelyet egy másik program fejlesztéséhez, teszteléséhez, adatainak előállításához, esetleg dokumentáció készítéséhez használunk. Ezeket az eszközöket az egész életciklus folyamán alkalmazzuk, így jelentős szerepük van az elkészült rendszer minőségében és minősítésében. A repülésben már az 1980-as évek óta jelen vannak, és alkalmazásuk az elmúlt években robbanásszerűen megnőtt. Ez nemcsak a fedélzeti rendszereket érinti, hanem minden olyan területet is, amelynek biztonsági követelményei vannak. Amennyiben valamely szoftvereszközt biztonságkritikus fejlesztésben kívánunk alkalmazni, akkor ezeknek az eszközöknek is megfelelő minősítéssel kell rendelkezniük. Ez a tanulmány ezeknek az eszközöknek a minősítésével foglalkozik.*

**Kulcsszavak:** *szoftverfejlesztés, szoftvereszközök, szoftvereszközök minősítése*

## 1. Bevezetés

A szoftvereszköz (eszköz) egy számítógépen futó program, programrendszer, esetleg ezeknek adott funkcionális része, amelyet egy másik program, annak adatai vagy dokumentációjának fejlesztésére, átalakítására, tesztelésére, elemzésére, előállítására vagy módosítására használnak. Az elmúlt években robbanásszerűen megnőtt az eszközök használata a szoftver életciklus-folyamatában, valamint más területeken, mint például a rendszerek, a programozható hardverek és a légi forgalmi adatbázisok.

Nemrég dolgoztunk egy projekten, amely tíznél több eszközt használt főleg tervezési és tesztelési területen, és viszonylag kevés volt a valós kódgenerálás. A projekt igen széles területet foglalt magában, és elég sok speciális igény merült fel. A kérdéses rendszer tervezése során egyértelműen kiderült, hogy elkerülhetetlen a szoftvereszközök használata. Ez a tapasztalat szintén megerősíti, hogy az eszközök használata célszerű, szükséges és alkalmazásuk minden területen növekedni fog. Az eszközök nem helyettesíthetik az emberi intelligenciát, azonban segíthetnek elkerülni a hibákat, azonosíthatják azokat, amelyeket az emberek elkövethetnek, illetve azokat, amelyeket a fejlesztők nem képesek azonosítani. Ezek az eszközök segíthetik a mérnököket munkájuk jobb elvégzésében, és lehetővé teszik számukra, hogy a nagyobb kihívást jelentő problémákra koncentráljanak, amelyek mérnöki készségeket, ítélőképességet

és intelligenciát igényelnek. Néhányan ellenállnak ezen eszközök használatának, míg egyesek sokszor felesleges túlzásokba esnek.

## 2. Szoftvereszközök felhasználási területei

Az 1. táblázatban láthatók a szoftver életciklusa alatt szóba jöhető eszközök.

1. táblázat
*A szoftvereszközök alkalmazási területei* [a szerző]

| Fejlesztőeszközök | Ellenőrző eszközök | Egyéb eszközök |
|---|---|---|
| • követelmény kezelése<br>• tervezés<br>• modellezés<br>• szövegszerkesztés<br>• fordítás<br>• linkelés<br>• automatikus kódgenerálás<br>• konfigurációs fájl generálása | • hibakeresés<br>• statikus elemzés<br>• legrosszabb eset végrehajtási időzítésének elemzése<br>• modellellenőrzés<br>• kódolási szabványoknak való megfelelőség<br>• nyomkövetés ellenőrzése<br>• strukturális lefedettség elemzése<br>• automatikus tesztelés<br>• emuláció<br>• szimuláció<br>• automatikus tesztgenerálás<br>• konfigurációs fájl ellenőrzése<br>• formális módszerek | • projektmenedzsment<br>• konfigurációmenedzsment<br>• probléma jelentése<br>• szakértőiértékelés-menedzsment<br>• dokumentáció generálása |

### *2.1. Kritériumok*

A kérdés az, hogyan tudjuk az alkalmazott eszközöket minősíteni, és melyek azok a szempontok, amelyeket szem előtt kell tartanunk.

Tapasztalataink és az irodalom a következő tevékenységeket javasolja [1], [2]:

- a fejlesztőeszközök minősítési kritériumainak meghatározása;
- a modellalapú fejlesztési eszközök kritériumainak meghatározása;
- kritériumok kidolgozása, amelyek egy adott szoftvereszköz különböző projektek közötti átvitelének lehetőségét határozza meg;
- az eszköz működési követelményeinek, használatának és minősítése módszereinek kidolgozása;
- az eltérő módon előállított szoftverelemek integrációs kritériumainak kidolgozása.

Több szabványt is elkészítettek, azonban vizsgálataink azt mutatták, hogy számunkra a repülésben alkalmazott DO-178B, DO-178C és a DO-330 felelt meg. A DO-178B és a frissített útmutató közötti különbségek is szóba kerülnek, mivel számos olyan eszköz van használatban, amelyet a DO-178B kritériumainak alapján minősítettek. Ezen túlmenően megvitatunk néhány, az eszközminősítéssel kapcsolatos speciális témát, valamint néhány lehetséges buktatót, amelyet el kell kerülni a minősítés vagy a minősített eszközök használata során.

Első körben tisztázni kell a következő kérdéseket [2]:

- Mit jelent az eszköz minősítése?

- Mikor szükséges?
- Milyen szintre kell minősíteni egy eszközt?
- Miben különbözik a DO-178B és a DO-178C eszközminősítési útmutató?

## *2.2. Szabványok*

A szoftvereszköz minősítése az a folyamat, amelyet egy szoftvereszköz adott minőségi tanúsítványának megszerzésére használnak. A szoftvereszköz minősítését az eszközt használó szoftver jóváhagyásával együtt adják meg, tehát ez nem egy önálló, örök időkre szóló jóváhagyás. Ez a minősítési folyamat tanúsítja a szoftvereszköz megbízhatóságát, amely megbízhatóságnak legalább egyenértékűnek kell lennie azzal a folyamattal, amelyet helyettesít. Az eszközminősítéshez szükséges tanúsítási és vizsgálati szint attól függően változik, hogy egy eszköz által generált hiba milyen potenciális hatással lehet a rendszer biztonságára és az eszköz általános használatára a szoftver teljes életciklusában. Minél nagyobb a kockázata annak, hogy egy szoftvereszközben lévő hiba hátrányosan befolyásolja a rendszer biztonságát, annál szigorúbb minősítés szükséges. A DO-178B szabvány, kétféle eszközt definiál [1]:

1. kategória: szoftverfejlesztő eszközök és
2. kategória: szoftverellenőrző eszközök.

Ezek a következő problémákat okozhatják:

- A fejlesztőeszköz: olyan eszköz, amelynek kimenete egy működő szoftver része, és így hibás működést okozhat.
- Az ellenőrző eszköz: olyan eszköz, amely nincs benne a kód előállítási folyamatában, de előfordulhat, hogy az általa ellenőrzött kódban előforduló hibát nem észleli.

A manapság használt szabványok, mint a DO-178C és a DO-330, háromféle eszközt definiálnak. Mivel az előzőekben említett két kategória – a DO-178B esetén – az életciklus adott fázisához volt kötve, nem pedig az eszköz hatásához, a DO-178C nem használja a fejlesztőeszközök és az ellenőrző eszközök kifejezéseket. Ehelyett a DO-178C három kategóriát határoz meg, amelyek az eszköz lehetséges hatására összpontosítanak. Ezek a kategóriák [2]:

1. kategória: eszköz, amelynek kimenete az eredményül kapott szoftver része, és így hibát okozhat;
2. kategória: eszköz, amely automatizálja az ellenőrzési folyamatokat, előfordulhat, hogy nem képes észlelni a hibát, holott ezek számának csökkentésére szolgál;
3. kategória: eszköz, amely a tervezett felhasználási körön belül nem képes észlelni a hibát.

Természetesen a DO-178B és a DO-178C kategóriái között nagy hasonlóság van. A DO-178C szerinti 1. kategóriájának megfelelő eszközök minősítési folyamata nagyon hasonló a DO-178B fejlesztőeszközeihez. Az 1. kategória eszközei közé tartoznak, a teljesség igénye nélkül, az automatikus kódgenerátorok, az eszköz működési követelményeit meghatározó eszközök, a konfigurációsfájl-generáló eszközök, a fordítók, a linkerek, a tervezőeszközök és a modellező eszközök.

Hasonlóképpen, a DO-178C szerinti 3. kategóriájának megfelelő eszközök minősítési folyamata nagyjából megegyezik a DO-178B ellenőrző eszközeivel. Ilyen eszközök például a teszteset-generáló eszköz működési követelményeit előíró eszközök, az automatizált teszteszközök, a szerkezeti és kódlefedettség eszközei és a statikus kódelemzők.

A lényeges különbség a DO-178B és a DO-178C között az, hogy a DO-178C egy harmadik kategóriát vezet be, a 2. kategóriát [2]. Ennek a speciális kategóriának a bevezetését elsősorban az indokolta, hogy fontoljuk meg olyan eszközök használatát – részben a jövőre való tekintettel –, amelyek kritikusabb döntéseket hozhatnak, mint a most használatos ellenőrző eszközök, hatásuk viszont kisebb az eredményül kapott szoftverre, mint a jelenlegi fejlesztőeszközöknek. Néhány formális módszer eszköz ebbe a kategóriába tartozik. Ilyen eszköz lehet egy olyan ellenőrző eszköz, amely egyetlen ellenőrző eszközként használható a forráskód-ellenőrzés egyes lépéseinek automatizálására, a szükséges tesztelés mennyiségének csökkentése érdekében.

Az eszközök és a kategóriák alapján a két szabvány minősítési szinteket határoz meg. A DO-178B szintjeinek elnevezése A, B, C, D. A DO-178C az eszközkategóriák és a kérdéses elkészítendő szoftver alapján úgynevezett TQL- *(Tool Qualification Level)* szinteket ad meg [3].

Öt TQL-szintet definiálnak. A TQL meghatározza a minősítési folyamat során szükséges „szigor" mértékét. A TQL-1 a legszigorúbb, a TQL-5 szint a legkevésbé szigorú (lásd 2. táblázat) [2].

2. táblázat
*A DO-178B és DO-178C szintek megfeleltetése* [5]

| **DO-178 eszköz kvalifikációs típusa** | **DO-178B** | **DO-178C** |
|---|---|---|
| Fejlesztés | A | TQL-1 |
| Fejlesztés | B | TQL-2 |
| Fejlesztés | C | TQL-3 |
| Fejlesztés | D | TQL-4 |
| Ellenőrzés | A…D | TQL-4 vagy TQL-5 |

Az eszköz minősítésére a két említett szabvány mellett bevezettek egy új szabványt, ez a DO-330. A DO-178C és a DO-330 célja, hogy az eszközminősítési kritériumokat világosabbá és áttekinthetőbbé tegye a DO-178B-vel szemben. Habár a DO-178B által minősített eszközök a legtöbb esetben elfogadhatók [3].

A DO-330 egy 128 oldalas dokumentum, így ebben a cikkben csak egy vázlatos áttekintést adhatunk [3]. A DO-330 olyan szempontok összessége, amelyek a fedélzeti és földi repüléselektronikai rendszereken kívül is használhatók. Tapasztalataink azt mutatják, hogy ezeket a szoftvereszközöket gyakran olyan csapatok fejlesztik, amelyek általános szoftverfejlesztéssel foglalkoznak. Ezek az eszközspecifikus dokumentumok segítenek abban, hogy az eszközfejlesztő csapat elkerülje a félreértéseket, az értelmezési problémákat, és sémát biztosítanak a minősítési eljáráshoz. A DO-330 eszközspecifikus útmutatást ad a szoftvereszköz elkészítéséhez. A repülés területén kívül más területek is használhatják. Az általunk fejlesztett terület, ahol ezt a szabványt figyelembe vettük, a biztonságkritikus szoftverek fejlesztése volt. Gyakori probléma, hogy az eszközöket gyakran olyan külső gyártók fejlesztették ki, akik esetleg alig értenek az adott szabványok által előírt szabályokhoz, vagy bármilyen más biztonságkritikus szabvány alatti fejlesztéshez. A DO-330-at úgy készítették, hogy a biztonságkritikus

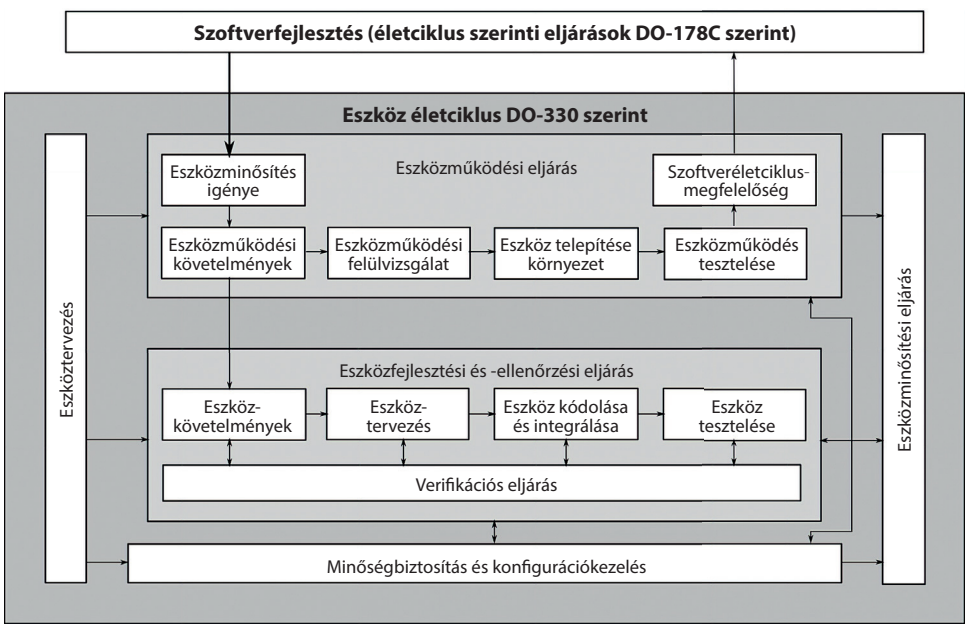
szoftverek fejlesztéséhez eszközspecifikus útmutatást nyújtson mind az eszközfejlesztők, mind a felhasználók számára. Ezért a dokumentum alapvetően a szoftver életciklusában használt eszközökre összpontosít. A dokumentum azonban más területekre is alkalmazható, ilyenek a programozható hardverek, az adatbázisok és akár általános célú rendszerek is [2]. A dokumentum adott tartományra való alkalmazásának módját a tartományra vonatkozó útmutatása határozza meg, például DO-178C stb.

A DO-330 kidolgozásánál a következő célokat fogalmazták meg [2]:

- lehetőség szerint tartsa meg a DO-178B megközelítését a hagyományos ellenőrző eszközök esetében;
- támogassa az újonnan megjelenő technológiákat;
- törekedjen arra, hogy az eszközminősítés szempontjait több projektbe is fel lehessen használni;
- azonosítsa az általános eszközfelhasználó és eszközfejlesztő szerepköreit azért, hogy támogassa az eszköz fejlesztői környezetbe integrálását;
- legyen egyértelmű, de rugalmas az eszközfejlesztők és a felhasználók számára;
- legyen alkalmazható többféle célra is.

Az 1. ábra a szoftverfejlesztés és az eszközfejlesztés kapcsolatát mutatja be. Az eszköz kétféleképpen „jöhet létre":

- előzőleg létrehozott vagy dobozos terméket használunk, például kockázatelemzésre, kezdeti dokumentumok és gyártási dokumentációk előállítására;
- az adott feladathoz kell előállítanunk, vagy előállíttatnunk speciális eszközt.




*1. ábra*

*A szoftveréletciklus és a DO-330 kapcsolata* [8]

### 2.2.1. Az eszköz működési eljárása

A második esetben az első lépés az eszköz tervezése, ehhez célszerű ismerni a generálandó szoftver alapvető jellemzőit működési szempontból. Szerencsés, ha a teljes követelményrendszer ismert, de számos esetben ez túl sok információt jelenthet, így a szűkített igény is elegendő lehet [1]. A szoftver adott életciklusfázisának megfelelően elemeznünk kell, hogy az eszköz minősítendő-e. A válasz általában igen. Ekkor meg kell határozni a követelményeket, amelyeket felül kell vizsgálni, lehetőleg egy független csapattal, illetve ha erre nincs lehetőség, legalább egy független szakértő bevonásával.

Amennyiben a terveket jóváhagyták, az eszközt létrehozzuk és telepítjük. A telepítés után megfelelő módon teszteljük. Ha sikeres ez a teszt, akkor a fejlesztett szoftver életciklusa szerint is megvizsgáljuk.

### 2.1.2. Az eszközfejlesztési és -ellenőrzési eljárás

A fejlesztés az előző pontban ismertetett követelmények alapján kezdődik. Minden lépést validációs eljárás követ, illetve minden egyes fejlesztési fázis a hozzá tartozó tesztelési eljárással együtt készül el.

A követelmények rögzítése után az eszköz tervezése a következő lépés. A tervezési fázisokat a kódolási és az integrálási fázis követi. Ezután következik a tesztelés. Mivel az adott eszközt az elkészítendő szoftver életciklusának általában csak egy szintjén használjuk, így ennek bonyolultsága valószínűleg nem éri el az elkészítendő szoftver vagy rendszer bonyolultságát, így tesztelése is egyszerűbb. Vegyük példának egy tesztesetgenerátor-eszköz működési követelményeit, ahol az eszköz feladata nem kódelőállítás, hanem a nagyszámú teszteset futtatása. Ez nem igazán bonyolult feladat, itt a számítógép és a program sebességét használjuk ki. De tudunk mondani jóval bonyolultabb eszközt is, ilyen például az időkritikus kódok generálására szolgáló modellbázisú kódgenerátor-eszköz. Amennyiben mind a működési eljárás, mind a fejlesztési eljárás minden lépését végrehajtottuk, az így elkészült eszközt minősítési eljárásnak is alá kell vetnünk. Ezt célszerűen szintén független csapatoknak kellene végrehajtani, amennyiben ez nem teljesül, az a bevett szokás, hogy ezt a tevékenységet az a csapat végzi, amelyik a követelményeket meghatározta.

A TQL szintek összefoglalása a DO-330 szemszögéből a következő [5]:

- TQL-5:
  - a minősítendő szoftvereszközök (terv a tanúsítás szoftveres vonatkozásaira);
  - az eszköz működési követelményeinek meghatározása;
  - minősítési stratégia (eszközminősítési terv);
  - eszközspecifikus információ;
  - győződjön meg arról, hogy az eszköz működési követelményei lefedik a tervtanúsítás szoftveres vonatkozásait;
  - konfigurációkezelés: a konfigurációs elemek azonosítása és verziókezelése;
  - alapvető minőségbiztosítási folyamatok meghatározása;
  - az eszköz működési követelményeinek ellenőrzése és érvényesítése;

- TQL-4:
  - eszközigény-kezelés: magában foglalja a definiálást, a nyomkövetést, az ellenőrzést és az eszközkövetelmények érvényesítését;
  - az 5 szükséges terv összeállítása:
    - eszközminősítési terv;
    - eszközfejlesztési terv;
    - eszközellenőrzési terv;
    - eszközkonfiguráció-kezelési terv;
    - eszközminőség-biztosítási terv;
  - az eszköz integrációjának fejlesztése;
  - szigorúbb eszközszintű konfigurációkezelés;
  - az eszköz végrehajtható kódjának kompatibilitási és robusztussági vizsgálata az eszköz követelményeivel összhangban;
- TQL-3:
  - a követelmények, a tervezés és a kód szabványok szerinti ellenőrzése;
  - átmeneti kritériumok, az eszköz folyamatai közötti összefüggések meghatározása;
  - az eszközfejlesztési környezet kiválasztása és meghatározása;
  - az alacsony szintű (kódközeli) követelmények kidolgozása és meghatározása;
  - a követelményeken alapuló tesztelés a megfelelőségiszint-elemzés végzése;
  - az eszközadatok és a program együttes elemzése;
- TQL-2:
  - az eszközellenőrzési folyamat tevékenységeit függetlenül kell végrehajtani;
  - külső komponensek elemzése és meghatározása;
  - az eszköz forráskódjának ellenőrzése;
  - alacsony szintű (kódközeli vagy kód-) követelmények ellenőrzése;
  - a követelményeken alapuló tesztelés, a döntési lefedettség elemzése;
- TQL-1:
  - fokozott függetlenség vizsgálata;
  - külső elemek ellenőrzése;
  - az esetleges módosított állapot/döntés hatásának vizsgálata.

Egyes eszközöket nehéz minősíteni a DO-178C/DO-330 szabvány alapján, ezért a minősítést végző személyek, illetve szervezetek megkerülő megoldást alkalmazhatnak. A DO-330 megemlíti az alternatív módszerek alkalmazásának lehetőségét [2].

A szabvány szerinti lehetséges alternatív módszerek közé tartozhatnak:

- kimerítő bemeneti tesztelés;
- formális módszerek és eltérő eszközök alkalmazása.

Az eszköz által használt külső komponenseket az eszköztervezési folyamat céljainak megfelelően interfészeikkel együtt kell leírni. Ezután ellenőrizni kell a helyességüket és tesztelni kell őket. A külső eszközök például az operációs rendszer, vagy a fordító futásidejű könyvtára által biztosított primitív függvények, vagy dobozos *(on the shelf),* esetleg nyílt forráskódú könyvtár által biztosított elemek. A korábban minősített eszközök használatát a DO-330 szintén tárgyalja. Ez felhasználási szempontból három kategóriára válik szét [2]:

- a korábban minősített, változatlan eszközök újrafelhasználása lehetséges, ha azok TQL-szintje, az életciklusadatok, az eszköz működési környezete, az eszköz működési követelményei és az eszköz verziója továbbra is ugyanaz, akkor az adott eszközt nem kell újra minősíteni;
- ha változások történtek az eszköz működési környezetében, akkor nincs szükség újraminősítésre, ha a felhasználó igazolni tudja, hogy az eszközellenőrzési környezet megfelelő az új eszköz működési környezetére;
- ha változtatások történtek magában az eszközben, akkor a hatáselemzésnek meg kell határoznia a szükséges újraellenőrzési tevékenységeket, például a nyomon követhetőség elemzése, regressziós tesztelés, a követelmények felülvizsgálata stb.

### *2.3. Felmerülő problémák*

A tapasztalatok és az irodalom számos buktatót tár fel. Ezekre a minősítési eljárás folyamán fokozott figyelmet kell fordítani. A leggyakrabban előforduló problémák [2], [8]:

- hiányzó dokumentáció;
- nem megfelelő verzió használata és vizsgálata;
- hiányos vagy hiányzó konfigurációs adatok;
- rosszul felmért minősítési szint;
- a működési követelmények és a környezet vizsgálhatósága;
- a „dobozos" eszközök minősítési adatai hiányosak, vagy hiányoznak;
- az eszköz szerepe nincs egyértelműen tisztázva a készítendő szoftver életciklusában;
- a minőségi mutatók nincsenek egyértelműen tisztázva;
- az eszköz tesztelését a fejlesztő végzi, nem független személy.

## 3. Összefoglaló

Ebben a cikkben összefoglaló áttekintést adtunk a biztonságkritikus szoftverfejlesztésben alkalmazott szoftvereszközök minősítésének legfőbb jellemzőiről, a minősítési eljárás szükségességéről, ennek lépéseiről és problémáiról. A legutóbbi biztonságkritikus projektünkben nem a repülés volt a fejlesztett terület, de a szabványok és eljárások vizsgálatakor a fejlesztő- és tesztelőcsapat egyértelműen alkalmazta a DO-178B és DO-178C szabványokat, illetve az alkalmazott és kifejlesztett fejlesztést támogató szoftvereszközöknél a DO-330 szabványt. Természetesen egyértelműen figyelembe vettük a biztonságkritikus fejlesztés „alapszabványát", az IEC61508-at és a rokon területek szabványait, mint az EN50128, EN50129 és az ISO 26262 szabványokat [6], [7].

A projekt során számos szoftvereszközt használtunk. Ezek egy része úgynevezett „dobozos", megvásárolható termék volt, másik része általunk előállított eszköz, amelyet minősíteni, illetve minősíttetni kellett. Ezek az eszközök jelentősen megkönnyítették a tervezési, fejlesztési, tesztelési, dokumentációs és minősítési tevékenységeket. A projekt szoftverfejlesztési szakasza sikeresen zárult.

### Felhasznált irodalom


[1] L. A. (Schad) Johnson, DO-178B, „Software Considerations in Airborne Systems and Equipment". University of Glasgow School of Computing Science, (é. n.). Online: www.dcs.gla.ac.uk/~johnson/teaching/safety/reports/schad.html

[2] L. Rierson, *Developing Safety-Critical Software. A Practical Guide for Aviation Software and DO-178C Compliance.* Boca Raton, CRC Press, 2013. Chapt. 12, 13. Online: https://doi.org/10.1201/9781315218168

[3] M. Ibrahim, U. Durak, „State of the Art in Software Tool Qualification with DO-330," in *Proceedings of the Software Engineering 2021 Satellite Events,* S. Götz, L. Linsbauer, I. Schaefer, A. Wortmann (szerk.), Bonn, Gesellschaft für Informatik, 2021. Online: http://ceur-ws.org/Vol-2814/paper-A4-4.pdf

[4] A. Methni, E. Ohayon, and F. Thurieau, „ASTERIOS Checker: A Verification Tool for Certifying Airborne Software," in *10th European Congress on Embedded Real Time Systems (ERTS 2020), Jan 2020, Toulouse, France.* Online: https://hal.archives-ouvertes.fr/hal-02508852/document

[5] J. Liu, X. Zhang, Y. Zhao, „Tool Qualification Requirements Comparison and Analyses Between RTCA/DO-178B and RTCA/DO-178C+DO-330," *Journal of Physics: Conference Series,* Vol. 1827. No. 1. pp. 012191. Online: https://doi.org/10.1088/1742-6596/1827/1/012191

[6] ISO 26262, Road vehicles – functional safety, Part 3 – 8, ISO Std., 2018. Online: https://bit.ly/3HgDYoz

[7] European Standards, *BS EN 50126 Railway Applications – The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS) – Part 1.* Online: https://bit.ly/3D1LSzM

[8] J. Marques, A. Marques da Cunha: „COTS Tool Qualification using RTCA DO-330: Common Pitfalls," in *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC).* 2017. pp. 1–6. Online: https://doi.org/10.1109/DASC.2017.8102031


## Certification of Software Tools Used in Development


*In many cases, it is useful or necessary to use tools to develop software. These tools are specialised software tools in 98% of cases. A software tool is a special program that is used to develop, test, or generate data for or produce documentation, for another program. These are used throughout the whole life cycle, consequently they play a significant role in the quality and qualification of the completed system.*

*These have been present in aviation since the 1980s, and their use has exploded in recent years. This applies not only to on-board systems, but also all areas that have safety requirements. If you want to use a software tool in safety-critical development, you must also have the appropriate certification of these tools.*


**Keywords:** *software development, software tools, software tool certification*

| | |
|---|---|
| Dr. Schuster György<br>docens<br>Óbudai Egyetem<br>Kandó Kálmán Villamosmérnöki Kar<br>Elektronikai és Kommunikációs Rendszerek Intézet<br>Műszertechnikai és Automatizálási Tanszék<br>schuster.györgy@kvk.uni-obuda.hu<br>orcid.org/0000-0002-8573-3670 | György Schuster (PhD)<br>Associate Professor<br>Óbuda University<br>Kandó Kálmán Faculty of Electrical Engineering Institute of Electronic and Communication Systems Department of Instrumentation and Automation<br>schuster.györgy@kvk.uni-obuda.hu<br>orcid.org/0000-0002-8573-3670 |