REPÜLÉSTUDOMÁNYI KÖZLEMÉNYEK
31. évfolyam (2019) 2. szám 57–66. • DOI: 10.32560/rk.2019.2.4

Ady László, Tokody Dániel

# Komplex rendszerek kommunikációjának hatásai és tervezési irányelvei

*A komplex rendszerek térnyerése jelentős méreteket öltött. A számosság növekedése és a felhasználási körülményekből következően a hibák, balesetek kockázata növekszik. Ez a kockázat a komplex rendszerek tervezése által csökkenthető. A komplex rendszerek elosztott együttműködő alrendszerek kommunikációjának együtteséből épülnek fel. A tervezés két fő részre bontható, a részegységek viselkedése és a részegységek együttműködése. A részegységek kommunikációját és együttműködését a protokoll írja le. A komplex rendszerek lehetnek nyitottak és zártak, de protokoll tekintetében mindig nyitottként kell kezelni, tehát nem lehet megbízni a részegységekben az elvárt viselkedés betartásában. A protokoll feladata a részegységek hibájának és a protokoll belső hibájának eszkalációját gátolni. A jelenlegi komplex rendszerek tervezési módszereit tekintjük át és foglaljuk össze. Ezek eredményes módszereit integráljuk egy komplex rendszer tervezési iránymutató-ajánlásban.*

***Kulcsszavak:*** *komplex rendszerek, elosztott rendszerek, protokoll, hibabiztos (fail-safe), üzemkritikus (mission critical)*

## Bevezetés

A komplex és elosztott rendszerek bonyolultságának és elterjedésének növekedésével a potenciális hibák kockázata is növekszik. A rendszerek egyre gyakrabban nyitottak. Ez alatt azt értjük, hogy a rendszer gyártójától független termék csatlakozhat a rendszerhez és részt vehet a rendszer működésében, teljes értékű elemként. Az EN 50129 rendelkezik a hálózatok besorolásáról, az 1. táblázatban található felosztás szerint. Ami a csatlakoztatható eszközök típusa, eszközök száma, a hálózat beállítása, illetéktelen behatolás kockázata alapján történik.

A kommunikációs hálózat osztályozása:

- I. osztályú (zárt);
- II. osztályú (nyílt);
- III. osztályú (nyílt).

1. táblázat
*EN 50129 szabvány szerinti hálózati besorolás* [11]

| Tulajdonság | I. osztályú (zárt) Belső hálózat | II. osztályú (nyílt) Zárt hálózat | III. osztályú (nyílt) Nyílt hálózat |
|---|---|---|---|
| Csatlakoztatható eszközök | Csak a gyártó által engedélyezett | Idegen (ellenőrizetlen) eszközök | Idegen (ellenőrizetlen) eszközök |
| Eszközök száma | A gyártó rögzíti | Nem ismert, változhat | Nem ismert, változhat |
| A hálózat beállítása | A gyártó által előírt | Dinamikusan változhat | Dinamikusan változhat |
| Illetéktelen behatolás | Elhanyagolható | Elhanyagolható | Nem elhanyagolható |

Így nem lehet építeni csupán a hálózatra kötött eszközök viselkedésére. A védelmi (funkcionális) és hibakezelő eljárásokat a hálózati protokollba kell integrálni. Ezek specifikációja és működése bonyolult. A fejlesztésekben jelentős figyelmet kell fordítani a protokollok komplexitásának kezelésére.

## Komplex rendszer

Komplex rendszer általános definíciója: A fizikai és matematikai modellezés egy ága, amely azt vizsgálja, hogy a részek kapcsolata hogyan vezet kollektív viselkedéshez, és a rendszer hogyan alakít ki kölcsönhatásokat a környezetével [10] (1. ábra).


1. ábra
*Komplex rendszerek* [1]

Komplex rendszer definíciója kommunikáció szempontjából: A komplex rendszert alkotó részek kapcsolata elosztottan és önszerveződőn alakítja ki a kollektív viselkedést a környezet változásaihoz alkalmazkodva.

## Elosztott rendszer

Elosztott rendszer definíciója az informatikában Andrew S. Tanenbaum és Maarten Van Steen szerint: Az elosztott rendszer olyan független számítógépek gyűjteménye ami a felhasználók számára egységes koherens rendszerként látszik [2].

Az elosztott rendszerek csoportosítása hardverarchitektúra (CPU-kommunikáció) szerint:

- szorosan csatolt rendszerek (memóriacímzési terület közös minden CPU-nak);
- lazán csatolt rendszerek (minden CPU-nak saját memóriája van, közvetlen buszon kommunikálnak);
- multiprocesszoros rendszerek (CPU-knak saját memóriája van, de van közös memória-területük is, több CPU alkot egy egységet);
- elosztott rendszerek (a CPU-knak nincs közös memóriája, közvetett buszon kommunikálnak).

A modern komplex rendszereket elosztott architektúrával valósítják meg.

Az elosztott rendszerek csoportosítása szoftverarchitektúra szerint:

- hálózati operációs rendszer;
- multiprocesszoros operációs rendszer;
- elosztott operációs rendszer;
- együttműködő autonóm rendszer.

A modern komplex rendszereket együttműködő autonóm rendszer architektúrával valósítják meg.

Az elosztott rendszerek lehetnek:

- homogének;
- inhomogének.

A modern komplex rendszereket inhomogén architektúrával valósítják meg.

## A kommunikáció hatásai

A komplex elosztott rendszerek a monolitikus rendszerekhez képest legfőképpen az adatkezelésben térnek el. A monolitikus rendszerben egységes globális adat, érték és idő van.

A komplex, elosztott rendszerekben nincs globális egyidejűség és globális adat. Léteznek technológiák központi adattárolásra, ezáltal globális adat kialakítására, de ez csökkenti a rendszer teljesítményét, mert idő mire a rendszeren körbe ér az információ (lock/unlock).

Vagyis kiemelt hibaforrás a rendszer elemei közötti egyetértés (adott pillanatban mindegyik szereplő ugyanazt az állapotot látja, és az alapján hoz döntést).

Elosztott rendszereknél ismert absztrakt hibák:

- csoporttagság (group membership): rendszeresen aktualizált lista a működő és nem működő rendszerelemekről;
- tudathasadás: a rendszer több különálló, de működő részre szakad;
- amnézia: kiesés után a kiesett rendszerelemek nem látnak rá a közben történt változásokra;

- konfiguráció-eltérés: a rendszer elemei eltérő konfigurációval rendelkeznek;
- verziószám-eltérés: a rendszer elemei eltérő szoftververzióval rendelkeznek.

A kritikus komplex rendszerek működése során több eltérő viselkedési elvárás lehet:

- fail-safe (FS);
- mission-critical (MC);
- high-availability (HA).

Az FS olyan tervezési és viselkedési mód, amikor meghatározott hibák esetén a rendszer úgy reagál, hogy nem okoz kárt vagy a lehető legminimálisabban tartja a baleset kockázatát. Akár a rendszer leállásával.

Az MC olyan tervezési és viselkedési mód, amikor a rendszer leállása komoly kárt vagy balesetet eredményez, akkor a meghatározott hibákra olyan módon reagál, hogy a rendszer képes legyen a biztonságosnak tekinthető állapotig eljutni.

A HA olyan tervezési és viselkedési mód, amikor a rendszer leállása kárt okoz, a hiba esetén keletkező károk kisebbek mint a rendszer leállításának a kára, ezért a rendszer, akár hibásan is, részfunkcionalitással vagy hibás funkcionalitással üzemben van tartva.

A rendszerfelhasználás jellege és a kockázatelemzés vagy törvényi, illetve szabvány-előírások alapján lehet meghatározni, hogy mikor melyiket kell választani.

Generikus komplex elosztott rendszerek esetén felmerülhet olyan igény, amikor a felhasználás jellege határozza meg hogy melyik viselkedés az elvárás. Vagyis a rendszernek képesnek kell lennie konfiguráció útján a viselkedését meghatározni.

A három viselkedés eltérő kommunikációs stratégiát és architektúrát igényel. Felépíthető átjárás ezek között.

A HA-működés a legmegengedőbb, majd az FS és végül a legkomolyabb elvárás az MC. Ilyen rendszer tervezésekor az MC-re kell tervezni és a konfiguráció során az MC-viselkedés FS-re vagy HA-ra vonatkozó megkötéseit lehet kikapcsolni. Persze ez egy komoly kockázat, amit ki kell értékelni és megfelelő eljárást kell alkalmazni a kockázat minimálisra csökkentése érdekében

## Protokoll

Protokoll definíció általánosan: A kommunikációban részt vevők által elfogadott és magukra nézve kötelezően betartott viselkedési forma.

Informatikában: Az informatikában a protokoll egy egyezmény vagy szabvány, amely leírja, hogy a hálózat résztvevői miképp tudnak egymással kommunikálni [7].

Általános értelemben a számítógépek közötti kommunikáció adatcserével történik – az adott rendszertől függően valamilyen módon kódolt információ. Ez a csere diszkrét lépésekből áll, amelyeket elemi kommunikációnak nevezünk, mindent ilyen üzenetben továbbítanak.

A rendszertől függően az üzenet lehet:

- egy elektronikus jel;
- nagyobb adatmennyiség.

Általánosságban az üzenet típusát használjuk az általános tartalom és a részletes kódolás meghatározására [3].

A modern elosztott rendszerekben egyre nagyobb jelentőség hárul a kommunikációra, a komplexitás és a rendszerek méretének növekedése miatt.

A „*Biztonságkritikus szoftverfejlesztés*" [12] cikkben publikált programhiba-kockázatokat az elosztott rendszerek esetén protokollhibákkal egészítjük ki.

A kommunikációs protokollal szembeni elvárások:

- kommunikációs hibadetektálás:
  - üzenetismétlődés;
  - üzenet-kimaradás;
  - üzenet beékelődés;
  - üzenet újraszámozás;
  - adatkorrupció;
  - üzenetkésés;
  - üzenet maszkolás;
  - résztvevők beállítás hiba;
  - FIFO-hiba.
- hibaeszkaláció megakadályozása;
- kommunikációs hatáskörök és jogosultságok kikényszerítése és ellenőrzése.

## Komplexitás, bonyolultság

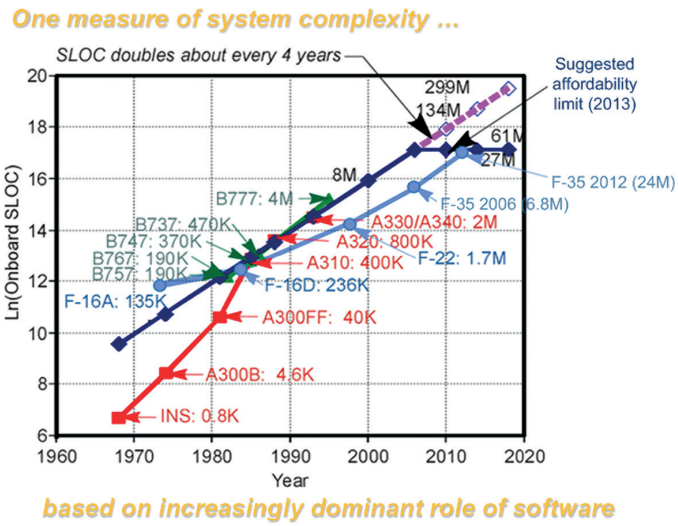
Elosztott rendszerek kommunikációja problémákat, lehetséges hibákat jelent. Ezeket a hibákat nehéz diagnosztizálni. A rendszerek funkcionalitása és mérete folyamatosan növekszik. Az avionika-rendszerek fejlődése követhető a 3. ábrán. A modern komplex rendszerekkel szemben jellemzően elvárt követelmény a folyamatos szoftverfrissítés és új funkcionalitások bevezetése. Ezek a funkciók lehetnek akár komoly funkcionális biztonságot érintők. Például a Tesla Autopilot 7.1 rendszerének Summon-funkciója, ami lehetővé teszi a vezető nélküli parkolást. A Tesla most bemutatott újdonsága nem egyedülálló, más autógyártó termékei esetében is elérhető ez a funkció. Ám olyan még soha nem történt, hogy egy, már eladott autót ruháznak fel ezzel a képességgel, az internetről letölthető frissítés segítségével [8].

A HURD-projekt egy elosztott operációsrendszer-kernel. A fejlesztése során kiemelt szempont volt a kommunikáció. A projekt fejlesztése lassan haladt és megelőzte a Linux kernel kiadása, így piacot vesztett. Richard Stallman – a GNU/HURD-projekt alapítója – nyilatkozata (Revolution OS 25m35s) [4]:

„*Nos, mi igazából nem sokkal azelőtt kezdtük a GNU Hurd-ot, mielőtt ő kezdte a Linuxot.*

*És hát az történt, hogy bár mi egy nagyon fejlett tervezési módot választottunk, mármint a teljesítőképesség szempontjából, de kiderült, hogy nagyon nehéz így a hibajavítás. Eldöntöttük, hogy felosztjuk a magot – ami hagyományosan egy program volt –, sok kisebb programra, amik szinkronizálatlan üzeneteket küldenek egymásnak, hogy kommunikáljanak.*

*Az a baj, hogy ez a programozásmód melegágya a hibáknak, amelyeket gyakran nagyon nehéz megtalálni, mert minden összefügg mindennel* […] *ez a program a másik üzenet előtt vagy után küldi el ezt az üzenetet* […] *szóval az eredmény az lett, hogy évekbe telt, hogy működjön a dolog.*"




3. ábra
*Avionikaszoftverek méretének változása* [9]

A Mars Climate Orbiter katasztrófájának az oka szintén az elosztott rendszer kommunikációjára vezethető vissza [5].

Ebből a rövid áttekintésből látható hogy a komplex, elosztott rendszerek komplexitáskezelése még a legkiemeltebb projektek esetén is nehézségeket okoz. Ezért az ajánlásunk a komplex elosztott rendszerek esetén:

- a protokolltervezés előtérbe helyezése;
- több védelmi (funkcionális) funkció integrálása a protokollba;
  - hibadetektáló;
  - hibajavító;
  - hibaelkerülő;
  - hibaeszkaláció-gátlás;
- protokoll formális specifikációja;
- protokoll formális verifikációja.

## Formális módszerek

A modern komplex rendszerek masszívan elosztott, eseményvezérelt aszinkron üzeneteket küldenek. Így a modern komplex rendszerek tervezése és vizsgálata során elengedhetetlen a kommunikáció hibamentességének bizonyítása.

A kiemelten magas biztonságintegritási szintet igénylő rendszerek tervezése, implementálása és üzemeltetése szabványokban meghatározott. Repülés területén DO-178C (DO-333) szabvány, vasút területén EN 50128, automotív területen ISO 26262 szabvány előírja a szoftverek vizsgálatára a formális módszereket. Ezek a szabványok az IEC EN 61508 szabványra épülnek.

Formális módszerek:
- CSP (Communicating Sequential Processes);
- CCS (Calculus of Communicating Systems);
- HOL (Higher Order Logic);
- LOTUS (Language for Temporal Ordering Specification);
- OBJ;
- Temporal logic;
- VDM (Vienna Development Method);
- Z módszer;
- B módszer;
- Model Checking.

A CSP (Communicating Sequential Processes) az egyidejűleg működő szoftverrendszerek specifikálására szolgáló technika. A rendszert független folyamatok hálózataként, a folyamatok egymás utáni vagy párhuzamos összeállításával modellezzük. A folyamatok csatornákon keresztül kommunikálhatnak (szinkronizálhatnak vagy cserélhetnek), a kommunikáció csak akkor történik meg, amikor mindkét folyamat készen áll.

A CCS (Calculus of Communicating Systems) a CSP-hez hasonlóan egy matematikai számítási módszer, amely a rendszerek viselkedésére vonatkozik. A rendszert független folyamatok hálózataként modellezik, amelyek egymást követően vagy párhuzamosan működnek. A folyamatok portokon keresztül kommunikálhatnak (hasonlóan a CSP csatornáihoz), a kommunikáció csak akkor történik, ha mindkét folyamat készen áll. A nem-determinizmus modellezhető.

A HOL (Higher Order Logic) egy logikai jelölés és annak gépi támogatási rendszerét jelenti, amelyeket a Cambridge Egyetem Számítógépes Laboratóriumában dolgoztak ki. A logikai jelölés többnyire a Church's Simple Theory of Types származik, és a gépi támogatási rendszer az LCF (Logic of Computable Functions) rendszerén alapul.

A LOTUS (Language for Temporal Ordering Specification) a CCS elgondolásán alapul, kiegészítve a CSP és a CIRCAL (Circuit Calculus) kapcsolódó algebrájával. Kiküszöböli a CCS adatstruktúra-kezelésben és értékkifejezésben jelentkező gyengeségeit.

Az OBJ egy algebrai specifikációs nyelv. A követelményeket algebrai egyenletek segítségével határozzák meg. A rendszer viselkedését az adatokon végzett műveletek szempontjából vizsgálja. Az OBJ egy ADT-technika. Mint egyéb ADT-technikák, az OBJ is csak szekvenciális rendszerekhez, vagy egyidejű rendszerek szekvenciális szempontjaihoz alkalmazható.

A Temporal logic az elsőrendű logikát modális operátorokkal egészíti ki.

A VDM (Vienna Development Method) egy matematikai specifikációs technika, amely lehetővé teszi a megvalósítás megfelelőségének bizonyítását a specifikáció alapján. A működése halmazelméleti alapokra épül, amiben a halmaz elemei a rendszerállapotok. A VDM-et főleg specifikáció során alkalmazzák. De lehetőség van a forráskódig való alkalmazásra is. A VDM csak szekvenciális programok vizsgálatához használható.

A Z módszer egy olyan specifikációs technika, ami lehetővé teszi a specifikációból végrehajtható algoritmus készítését oly módon, hogy az algoritmus bizonyítottan a specifikáció szerint működik. A Z módszer adatorientált szekvenciális rendszerek fejlesztésére alkalmas.

A B módszert a Z-hez fejlesztették ki, amely egy lépésről lépésre történő finomítási módszer.

A Model Checking egy folyamat, amely ellenőrzi, hogy egy adott szerkezet egy adott logikai képlet modellje-e. A koncepció általános, és mindenféle logikára és megfelelő struktúrára vonatkozik. Egy egyszerű modellellenőrzési probléma az, hogy teszteljük, hogy egy adott struktúra egy adott képlet a javasolt logikában teljes-e.

A Model Checking módszerek fontos osztályát képezi a formális rendszerek algoritmikus ellenőrzése. Ezt úgy érhetjük el, hogy meggyőződünk arról, hogy a szerkezet, amely gyakran egy hardver- vagy szoftvertervből származik, megfelel-e egy formális specifikációnak, tipikusan időbeli logikai képletekkel.

Ha a kommunikációt formálisan írják le, lehetőség van matematikai vizsgálatra. A folyamatok vizsgálatára kidolgozott formális módszerek közül a CSP és a TimedCSP alkalmas leginkább a protokollok vizsgálatára. 2018-ban a PKMv3-protokollt vizsgálták meg [6] CSP formális módszer alkalmazásával. Formális módszerekkel lehetővé válik:

- a protokoll formális modellezése és analízise;
- követelmények formalizálása;
- a prokoll formális verifikációja modellellenőrzéssel;
- állapotfüggő dinamikus viselkedés modellezése;
- konkurens rendszerek modellezése és analízise;
- adatfüggő viselkedés modellezése;
- adatfeldolgozás modellezése;
- szoftver forráskód alapú formális verifikációs technikák.

A CSP formális leírás a rendszer vagy a rendszer elemeinek a folyamatait írja le. Az elosztott komplex rendszer kommunikációjában részt vevők az egymással folytatott üzenetváltás során képesek befolyásolni egymás működését. A CSP-ben a folyamatok eseményekből épülnek fel.

Az események és folyamatok között operátorok teremtik meg a kapcsolatot.

Alap operátorok:

- prefix;
- választás:
  - determinisztikus;
  - nem determinisztikus;
  - környezeti választás.
- elágazás;
- üzenet:
  - küldés;
  - fogadás.
- párhuzamos folyamat;
- traces;
- STOP.

A valós rendszerekben az üzenet küldésének időigénye van. A Timed CSP az idővel kapcsolatos operátorokat vezeti be és teszi lehetővé a valós rendszerek leírását és elemzését.

A komplex elosztott rendszerekben a kommunikáció leírása viselkedést leíró programhalmazokkal oldható meg. Amennyiben a protokoll követelményspecifikációja formális leírással történt és a megtervezett protokoll leírása formális. Matematikai módszerekkel

ellenőrizhető hogy a tervezés során létrehozott protokoll az elvárásoknak megfelelő viselkedéssel rendelkezik-e.

A protokollra épített rendszer kommunikációjának a megfigyelése során lehetőség van üzem közben ellenőrizni, hogy a kommunikáció a protokoll követelményspecifikációja szerint zajlik-e.

## Felhasznált irodalom


[1] H. Sayama, *Introduction to the Modeling and Analysis of Complex Systems.* Geneseo: Open SUNY Textbooks, Milne Library, 2015.

[2] A. S. Tanenbaum, M. van Steen, *Distributed Systems: Principles and Paradigms.* US: Prentice Hall, 2002.

[3] R. Sharp, *Principles of Protocol Design.* Berlin Heidelberg: Springer-Verlag, 2008. DOI: https://doi.org/10.1007/978-3-540-77541-6

[4] J. T. S. Moore, "Revolution OS," 2001, youtube.com, [Online]. Elérhető: www.youtube.com/watch?v=jw8K460vx1c; imdb.com, www.imdb.com/title/tt0308808/

[5] Mars Climate Orbiter Mishap Investigation Board Phase I Report, November 10, 1999, NASA, [Online]. Elérhető: https: //llis.nasa.gov/llis_lib/pdf/1009464main1_0641-mr.pdf

[6] J. Jiang, H. Mao, R. Shao, and Y. Xu, Formal Verification and Improvement of the PKMv3 Protocol Using CSP, In Proc. 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), 2018. DOI: https://doi.org/10.1109/COMPSAC.2018.10318

[7] Wikipedia The Free Encyclopedia, „Protokoll (informatika)," [Online]. Elérhető: https://hu.wikipedia.org/wiki/Protokoll_(informatika)

[8] Zs. Simon, „Nem kell többé a parkolóban keresned a Teslád, odamegy hozzád!" villanyautosok.hu, 2019, [Online]. Elérhető: https: //villanyautosok.hu/2019/03/25/nem-kell-tobbe-a-parkoloban-keresned-a-teslad-odamegy-hozzad/

[9] System Architecture Visual Integration, "About the SAVI Program," *System Architecture Visual Integration,* [Online]. Elérhető: https: //savi.avsi.aero/about-savi/

[10] M. Kellermayer, „Komplex Rendszerek," [Online]. Elérhető: http: //biofiz.semmelweis.hu/run/dl_t.php?id=2449&tid=94

[11] MSZ EN 50129: 2003: Vasúti alkalmazások. Távközlési, biztosítóberendezési és adatfeldolgozó rendszerek. Biztonsági elektronikai rendszerek biztosítóberendezésekhez, MSZT, 2003.

[12] Gy. Schuster, L. Ady, „Biztonságkritikus szoftverfejlesztés," *Repüléstudományi Közlemények,* 30. évf. 1. sz. 2018. [Online]. Elérhető: www.repulestudomany.hu/folyoirat/2018_1/2018-1-11-0453_Schuster_Gyorgy-Ady_Laszlo.pdf


## További irodalom


J. Delange, B. Nichols, J. Hudak, J. McHale, and M.-Y. Nam, "Evaluating and Mitigating the Impact of Complexity in Software Models," Tech Report, CMU/SEI-2015-TR-013, 2015. DOI: www.doi.org/10.13140/RG.2.1.4028.1840

RTCA SC-205: DO-178C, Software Considerations in Airborne Systems and Equipment Certification, 2012.
MSZ EN 50128: 2011: Vasúti alkalmazások. Távközlési, biztosítóberendezési és adatfeldolgozó rendszerek. Szoftverek vasúti vezérlő- és védelmi rendszerekhez, MSZT, 2011.
International Organization for Standardization: ISO 26262 Road vehicles – Functional safety, 2011.
MSZ EN 61508: 2010: Villamos/elektronikus/programozható elektronikus biztonsági rendszerek működési biztonsága, MSZT, 2010.

## EFFECTS AND DESIGN GUIDELINES FOR COMMUNICATING IN COMPLEX SYSTEMS

*The expansion of complex systems has taken on considerable proportions. Due to the increase in numericity and the conditions of use, the risk of errors and accidents increases. Reducing this risk can be obtained by designing complex systems. Complex systems are built up of a combination of distributed collaborative subsystems. The design can be divided into two main parts, the behaviour of the components and the overall operation of the components. The protocol describes the communication and collaboration of the components. Complex systems can be open and closed, but they should always be treated as open in terms of protocol, so one cannot trust the units to follow the expected behaviour. The task of the protocol is to prevent escalation of the component failure and the internal error of the protocol. The design of the current complex systems is reviewed and summarised. Their effective methods are integrated into a complex system design guideline.*

***Keywords:*** *complex systems, distributed systems, protocol, fail-safe, mission critical*

*Ady László*
*Ügyvezető,*
*Kísérleti fejlesztésvezető*
*NextTechnologies Kft.; Subi-Ker 2000 Kft.*
*adylaszlo@nexttechnologies.hu*
*https://orcid.org/0000-0001-6702-6000*

*László Ady*
*Executive Director*
*Experimental Development Leader*
*NextTechnologies Ltd.; Subi-Ker 2000 Ltd.*
*adylaszlo@nexttechnologies.hu*
*https://orcid.org/0000-0001-6702-6000*

*Tokody Dániel*
*Vezető kutató*
*Subi-Ker 2000 Kft.; NextTechnologies Kft.*
*tokody.daniel@nexttechnologies.hu*
*https://orcid.org/0000-0002-9984-0434*

*Dániel Tokody*
*Principal Researcher*
*Subi-Ker 2000 Ltd.; NextTechnologies Ltd.*
*tokody.daniel@nexttechnologies.hu*
*https://orcid.org/0000-0002-9984-0434*

http://journals.uni-nke.hu/index.php/reptudkoz/article/view/300/166