**REPÜLÉSTUDOMÁNYI KÖZLEMÉNYEK**
35. évfolyam (2023) 1. szám 71–80. • DOI: 10.32560/rk.2023.1.6

Schuster György

# Idősorok alkalmazása szoftvermegbízhatóság előrejelzésére

*A biztonságkritikus szoftverek fejlesztése kulcsfontosságú. Ez kifejezetten hangsúlyos olyan területeken, ahol a feladatok nagy része, vagy teljes egésze szoftverekre van bízva. Az ilyen szoftverek hibái komoly következményekkel járhatnak, ideértve az életveszélyes helyzeteket, a gazdasági károkat és egyéb komoly problémákat. Ebbe a kategóriába tartoznak a a hagyományos és autonóm járművek fedélzeti szoftverei, az űrkutatás autonóm eszközeinek, speciális ipari eszközöknek és fegyverrendszereknek a szoftverei is.*
*A megrendelő számára komoly kihívás a szoftver gyártójának kiválasztása. Mivel a szoftverek esetén a klasszikus gyártási folyamatok vizsgálati módszereit nem tudjuk alkalmazni, ezért a gyártót kell megvizsgálnunk. Az teljesen egyértelmű, hogy egy adott gyártónak megfelelő minősítésekkel kell rendelkeznie. De mi a helyzet akkor, ha több azonos minősítésű gyártó áll rendelkezésre, és ezek közül kell választani?*
*Ebben a cikkben javaslunk egy módszert, amellyel a gyártók megfelelő megszorítások melletti általános fejlesztési tevékenységét vizsgáljuk, az elkövetett működési és fejlesztési hibák alapján, ezeket idősorokba rendezve. A cikkben ismertetjük a módszert, annak előnyeit, korlátait, alkalmazásának akadályait és egy nem várt következményt az oktatásban.*

**Kulcsszavak:** *szoftverminőség-becslés, idősorok*

## 1. Bevezetés

A szoftver kritikus sikertényező. Egyre több feladatot bízunk rájuk. Ezek nagy része biztonságkritikus. Sokszor ez azt is jelentheti, hogy meghibásodás esetén emberi beavatkozásnak nincs lehetősége, vagy a beavatkozás lassú, esetleg rendkívül kockázatos.

Vegyünk például egy fly-by-wire repülésirányító rendszert. Egy szoftvermeghibásodás akár katasztrofális következményekkel is járhat, itt azonnali javításra nincs lehetőség, és az emberi beavatkozás is erősen korlátozott, esetleg az időkorlátok miatt. Ha például egy instabil repülőszerkezetben következik be irányítási hiba, a személyzet esetleg cselekvésképtelenné válhat a fellépő túlterhelés miatt.

Másik példánk legyen egy űrszonda. Ha programhiba lép fel kritikus helyzetben, ez az eszköz elvesztését is jelentheti. Ha ez nem kritikus helyzetben következik be, akkor esetleg a hiba javítható, de sok időbe kerül, és a beavatkozás igen kockázatos.

A klasszikus gépészeti és villamos gyártásban vannak megfelelő statisztikai módszerek, amelyekkel a gyártás folyamata minősíthető. Ezt sajnos a szoftverfejlesztésben elég nehezen tudjuk biztosítani. A szoftvercégek természetesen rendelkeznek megfelelő minősítésekkel, megfelelő módszertannal, eljárásrenddel. Ez a CMM[1] besorolási rendben legalább a meghatározott szintet jelenti [3].

Ez mind segíthet, de a folyamat végén ott van az ember, aki a fejlesztés kulcstényezője. Sajnos az ember mint fejlesztő okos, tapasztalt, intuitív és kreatív, és ugyanilyen mint hibaforrás. Ami előny, az egyszerre hátrány is.

## 2. Besorolási szintek

Keressük azokat a statisztikai jellemzőket, amelyek segítenek a fejlesztési folyamat vizsgálatában abból a célból, hogy a még el nem készült szoftvertermék minősége az elvárásoknak megfelelő.

A CMM- és a CMMI-besorolások[2] ugyan bizonyos mértékig információt szolgáltatnak az adott gyártó minősítéséről, ezért ezeket mindenképpen célszerű figyelembe venni.

### *2.1. CMM-besorolások*

A CMM egy olyan modell, amely a szoftvertervezési és -fejlesztési folyamatok érettségének és minőségének értékelésére szolgál. A modell az alábbi öt besorolásból áll [2]:

- kezdeti (kaotikus) szint: a folyamatok nem rendelkeznek standardizált módszerekkel, és nem kiszámíthatók. Az eredmények változók, és az egyén szerepe rendkívül fontos;
- menedzselt szint: a folyamatokat kezelik és ellenőrzik, és azokat a minőségi követelményeknek megfelelően végzik. A folyamatokat és a teljesítményt mérni lehet;
- meghatározott szint: a folyamatok szabványosítottak és dokumentáltak, ami lehetővé teszi a szabványos és következetes végrehajtást. A minőség ellenőrizhető és javítható;
- mennyiségi szempontból menedzselt szint: a folyamatokat mérhető, kvantitatív módon vezetik és ellenőrzik. A folyamatok szorosan kapcsolódnak a gyártói célokhoz és a minőségi követelményekhez;
- optimalizált szint: a folyamatokat mindig optimalizálják a minőség és hatékonyság növelése érdekében. A folyamatokat a legjobb gyakorlatok alapján végzik, és állandóan fejlesztik őket.

### *2.2. CMMI-besorolások*

A CMMI az előző CMM-modell továbbfejlesztett verziója, amely nemcsak a szoftvertervezési és -fejlesztési folyamatok értékelésére, hanem az egész szervezet működésének javítására is szolgál.

[1] Capability Maturity Model.
[2] Capability Maturity Model Integration.

A CMMI hat képességszintet határoz meg, amelyek a szervezetek érettségi szintjét tükrözik a folyamatok végrehajtása és a minőség ellenőrzése szempontjából. Az alábbiakban a hat képességszintet soroljuk fel [2]:

- kezdeti szint;
- menedzselt szint;
- meghatározott szint;
- mennyiségileg menedzselt szint;
- optimalizált szint;
- túlmutató szint: ezt a szintet a CMMI legújabb verziója határozza meg, és azt mutatja, hogy a szervezetek már nemcsak a folyamataikat optimalizálják, hanem azokat a megrendelői igényeknek megfelelően folyamatosan fejlesztik is [2].

Biztonságkritikus fejlesztésben csak az a gyártó jöhet szóba, aki legalább a harmadik szinten van.

## 3. A vizsgálat módszertana

### *3.1. Az idősor képzése*

Az idősor elemei a fejlesztés és/vagy a rendszer működése közben bekövetkezett hibák számából állnak. A probléma az időszelet meghatározása. A biztonságkritikus fejlesztés számos esetben egyedi fejlesztést jelent, ráadásul az egymást követő projektek eltérő platformon történnek.

A kérdés, hogyan határozzuk meg az időszeletet? Rögtön három probléma merül fel:

- a szoftverek nem futnak állandóan, így azonos platformon is nehéz a követés;
- az eltérő platformok működési sebessége eltérő lehet, így az időalapú összehasonlítás megint nem korrekt;
- az eltérő platformok eltérő fejlesztői környezeteket feltételeznek.

Ezért a következő javaslatot tesszük: az idő kezelése helyett a végrehajtott gépi utasítások számát vegyük alapul.

Az is nyilvánvaló, hogy a gépi utasításkészletek is különbözhetnek, illetve a fordított kódok sem felelnek meg egymásnak különböző platformokon utasításról utasításra. Ezért statisztikai alapon bevezethetünk egy súlyozó tényezőt. Így a végrehajtási számok összehasonlíthatók lesznek. Nem szabad arról sem elfeledkeznünk, hogy egy jó szoftvergyártónál a két programhiba között végrehajtott gépi utasítások száma óriási lehet. Az adatgyűjtés esetén ez nem igazán gond, mert az összehasonlítás után nem „nagyon" dolgozunk ezzel a számmal.

Az általunk javasolt vizsgálat olyan szoftvercsoportokat vizsgál, amelyek közvetlenül a fejlesztést hajtják végre, nem kizárva a tervezési folyamatot. Nem alkalmazható kísérletező, a projektben nem közvetlenül részt vevő kutatói csoportra, illetőleg olyan csoportokra, ahol gyakornokok is írnak olyan programokat, amelyek esetleg bekerülnek a végleges kódba.

Miután meghatároztuk a megfelelő csoportot, a következő statisztikai vizsgálatokat célszerű elvégezni:

- Hurst-analízis;
- autokorrelációs függvény;
- idősorra illesztett regressziós polinom.

A Hurst-analízis, más néven Hurst-exponens analízis, egy matematikai eszköz, amelyet idősorok hosszú távú memóriájának mérésére használnak [1].

*Megjegyzés:* számos Hurst-exponenst meghatározó módszer létezik [5]. Ezek nagy részét megvizsgáltuk, lényeges eltérést nem találtunk, ezért a legegyszerűbbet választottuk ki.[3]
A Hurst-exponens kiszámításának képlete [6], [7]:

$$\frac{R}{S} = N^H, \Rightarrow log(R/S) = Hlog(H), \Rightarrow H = \frac{log(R/S)}{log(N)} \quad (1)$$

és rövid magyarázata [1]:

A Hurst-exponens H kiszámításához az idősorokat különböző részekre, úgynevezett bokszokra osztják fel, majd minden egyes boxra meghatározzák a boxban található adatok szórását.

Ezután kiszámítják a logaritmusok átlagát és szórását az összes boxra. A Hurst-exponens értéke a logaritmusok szórásának és átlagának az aránya.

A paraméterek [3]:

R – a legnagyobb boxátlag;
S – a legnagyobb boxszórás;
N – a boxok száma;
H – a Hurst-exponens.

Az exponens értéke az idősor trendtartására utal, ha:

$0 \leq H < 05$ az idősor kiszámíthatatlan jellegű,
$H = 05$ az idősor teljesen véletlenszerű sorozat,
$05 < H \leq 1$ az idősor trendtartó jellegű. Minél magasabb ez az érték, annál erősebb a trendjelleg.

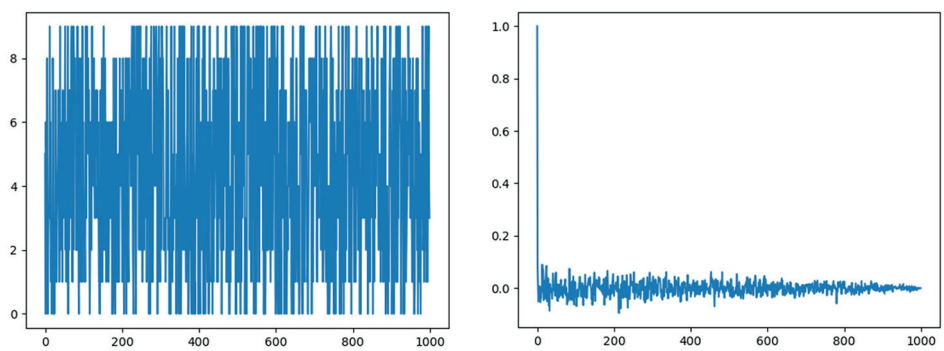
Emellett célszerű megvizsgálni az idősor autokorrelációs függvényét. Ez számos esetben segíthet a hibák okainak felderítésében. Ha az autokorrelációs függvény csak a kezdőértékben mutat 1 értéket és máshol 0, vagy közel nulla, akkor a hibák bekövetkezése teljesen véletlenszerű. Minél gyorsabban éri el a 0 értéket a sorozat, annál gyorsabban „felejti el" az előző értékeket [7].

Mivel tételszerűen kijelenthetjük, hogy hibátlan szoftver nincs, következésképpen hibátlan fejlesztési folyamat sincs. A véletlen jellegű szoftverhibák előfordulása nem szerencsés. Számunkra az a kedvező, ha az autokorrelációs függvény abszolút értéke gyorsan csökken, de nem túl gyorsan. Ha az autokorrelációs függvény „magas", az azt jelentheti, hogy a hibák rendszeresek és a gyártó ezt a problémát nem kezeli elég hatékonyan. Ha az autokorrelációs függvény „túl alacsony", ez azt jelenti, hogy a hibák közel teljesen véletlenszerűek, így ezek okainak felderítése nehezen, vagy egyáltalán nem megoldható, például hardverkörnyezet-eredetű hibák. Ha az autokorrelációs függvény ezek között van, az szintén egyfajta trendtartást jelent [9], [10].

[3] Amennyiben valakinek a Python-forráslistára szüksége van, szívesen rendelkezésére bocsátom.

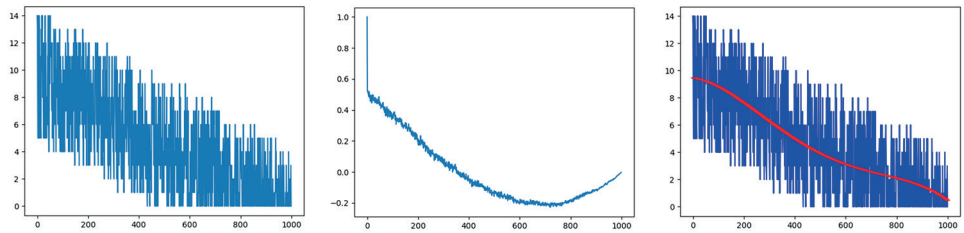
Az előző gondolatmenetben feltételeztük, hogy az elkövetett hibák száma csökken. Az autokorrelációs függvény nem mondja meg, hogy a sorozat elemei csökkenő vagy növekvő tendenciát mutatnak, csak a sorozat elemeinek összefüggéséről nyújt képet. Ezért meg kell néznünk a trend jellegét. Erre célszerű valamilyen regressziós függvény alkalmazása. A lehetőségek a következők:

- lineáris regresszió, vagyis regressziós egyenes illesztése az adatsorra. Ez arra alkalmas, hogy összességében lássuk, hogy a trend általánosságban növekszik vagy csökken. Azonban a finomabb részleteket elfedi [10];
- exponenciális regresszió, egyszeres exponenciális regresszió nem elegendő, mert egy trendváltozást képtelen kimutatni [11];
- szigmoid függvény és logisztikai görbe. A szigmoid függvény inflexiós pontja az értelmezési tartomány közepén van. Ez sajnos nem szerencsés. A logisztikai görbe csökkenő formája a szigmoid görbe inflexióproblémáját ugyan megoldja, de sajnos itt is azt tapasztaljuk, hogy a finomabb részleteket a görbe figyelmen kívül hagyja [12];
- regressziós polinom alkalmazása. Végül ezt a módszert alkalmaztuk. A vizsgálatok esetén maximum negyed-ötödfokú polinom alkalmazása volt célszerű [13]. A magasabb fokú polinomok túlságosan „figyelnek" a részletekre, és információt nem hordozó hullámzást mutatnak.




1. ábra
*Véletlenszerű folyamat idősora és autokorrelációs függvénye* [a szerző]

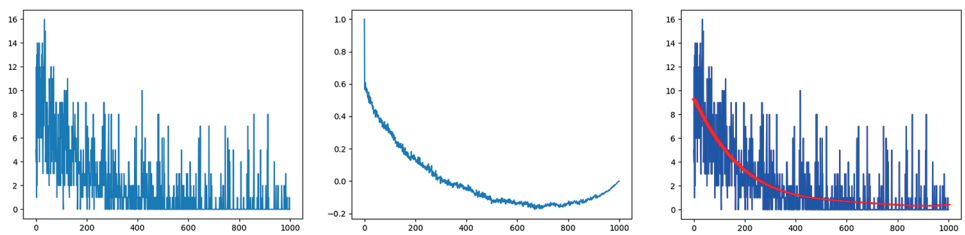
Az 1. ábrán egy véletlenszerű folyamat időDiagramját láthatjuk. Ez szimulált idősor. A Hurst-exponens értéke H = 0,47, amely nagyon közel van a teljesen független idősor 0,5 értékéhez. Ha megnézzük az 1. ábrán az autokorrelációs függvényt, akkor láthatjuk, hogy a 0 értéknél látható az 1 érték, a diagram további részei közel 0 értékűek. Ez alátámasztja a véletlenszerűség tulajdonságot.



2. ábra
*Trenddel rendelkező folyamat idősora, autokorrelációs függvénye és polinomiális regressziós függvénye* [a szerző]

A 2. ábrán szintén egy szimulált idősort láthatunk, de a generáló függvénybe építettünk egy lineáris csökkenő tendenciát. Ez jól látszik a diagramon. A Hurst-exponens H = 0,91. Az autokorrelációs függvényen is egyértelműen látszik az idősorban a „hosszú távú emlékezet".

A 2. ábra harmadik részábráján az idősorra egy ötödfokú regressziós polinomot illesztettünk. Ez a piros görbe.



3. ábra
*Egy valós fejlesztési projekt jellemzői* [a szerző]

A 3. ábrán egy valós vizsgálat adataiból származó idősor adatait láthatjuk. (Ezek az egyetlen céges adatsorból származó adatok, ezt később tárgyaljuk.) A Hurst-exponens értéke H = 0,93. Az autokorrelációs függvény szintén mutatja az „emlékezetet". A harmadik részábra az idősorra illesztett ötödfokú polinomot mutatja. Ez egyértelműen mutatja a trend jellegét.

A vizsgálatokat 1000 elemű idősorokra végeztük el.

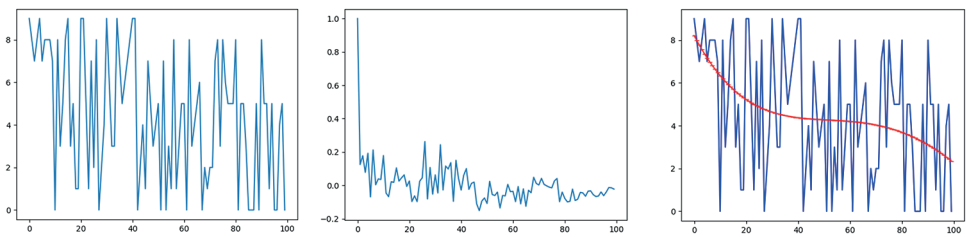
## 4. Másodlagos felhasználás

A módszert sajnos cégek esetén nem tudtam alkalmazni, ennek okairól nem kívánok a továbbiakban beszélni.

Ez alól csak egy cég volt kivétel, de ott is csak egyetlen programozót vizsgálhattam, mert a cégben lévő két másik programozó nem szoftverprojektekben dolgozik, hanem kutatást végez, és erre, mint már korábban említettük, a módszer nem alkalmazható. Viszont oktatási vizsgálatokat tudtam végezni különböző tantárgyak esetén.

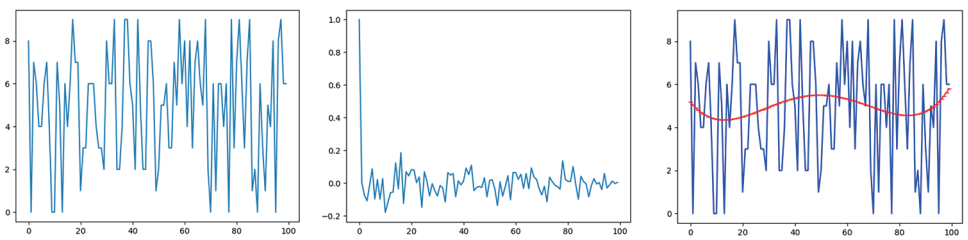
Az adatgyűjtés módszere az volt, hogy a hallgatók által írt programok szintaktikai és szemantikai hibáit vizsgáltuk. Az így kapott idősorok egyes hallgatók esetén egyértelműen megmutatták a kérdéses hallgató képességeit. A kapott Hurst-exponens, a regressziós polinom és hibaráta jól korrelált az adott szemeszter végén kapott érdemjeggyel.

Másodlagos eredmény volt az, ha az oktatókat vizsgáltuk. Szintén a Hurst-exponenst, a regressziós polinomot és hibaarányt vizsgáltuk. Mivel az adott tantárgyakban a szubjektív osztályzás kizárt, ezért az eredmények az oktatót minősíthetik. Az eredmények alapján az oktatók rangsorolhatók. Lényeges, hogy a vizsgált minta elég nagy legyen. Jó lenne az is, ha a két féléves laboratóriumi oktatás esetén, a kurzusokon azonos hallgatók lennének, de ez sajnos nem biztosítható. Kiválasztottunk két hallgatót, egy jól teljesítőt és egy nem annyira jól teljesítőt, aki viszont a vizsgálat két félévét elsőre sikeresen abszolválta. Lényeges, hogy a két hallgató azonos kurzuson vett részt.



4. ábra
*A jól teljesítő hallgató diagramjai* [a szerző]

A 4. ábrán sorra a jól teljesítő hallgató diagramjai láthatók. A Hurst-exponens H = 0,75 értékű.



5. ábra
*A gyengébb hallgató diagramjai* [a szerző]

Az 5. ábrán a gyengébb hallgató diagramjait tüntettük fel. A Hurst-exponens H = 0,27, amely nem teljes véletlenszerűséget mutat, de az eredmény eléggé következetlennek számít.

Első pillantásra a két hallgató idősora között szemmel alig látható különbség van. A jelleget a regressziós polinom és a Hurst-exponens mutatja. A szemeszterekét lezáró jegyek az első hallgatónál jó és jeles, a második hallgatónál elégséges és elégséges. Tehát az idősor-analízis helyesen mérte fel a hallgatók teljesítményét.

Az oktatóknál a korrekt eredményekhez sokkal nagyobb mintát kell vizsgálni. A vizsgálatot erősen torzítja az, hogy nem azonos populáción végezzük a vizsgálatot. Ezért gyors eredmény

nem várható el. Sajnos a pandémia ezt a vizsgálatot lehetetlenné tette, és az eredmények elvesztek. A vizsgálatokat a 2021/22-es tanév első félévében kezdtük újra. Ez az időtartam túl rövid, ezért nem tartjuk megfelelőnek, hogy az eredményeket jelen pillanatban publikáljuk.

## 5. Minőségi következtetés

Az előző vizsgálatok egyértelműen meghatározták a fejlesztők tevékenységének trendjét, azonban ez még kevés a várható minőség becslésére. Ez egyértelmű, ha nézzük a következő erősen sarkított példát: adott „A" fejlesztő, a vizsgálati trendje jó, a Hurst-exponens értéke H = 0,85, az autokorrelációs függvény a megfelelő szinten van, de a hiba előfordulásának gyakorisága $10^{-10}$, ez rendkívül magas hibaráta. „B" fejlesztő értékei hasonlók, míg Hurst-exponense kicsit gyengébb, mint az előző fejlesztőé, H = 0,79, autokorrelációs függvénye is fölötte halad az előzőnek, a trend iránya jó, viszont a hiba előfordulásának gyakorisága $10^{-25}$. Ez sokkal jobb arány, tehát nyilvánvalóan a „B" fejlesztőt fogjuk választani. Az egyértelmű, hogy egy viszonylag rossz eredményből sokkal könnyebb javulni, mint egy eleve jobb eredményből.

Az előzőekben ismertettük egy fejlesztő cég idősorjellemzőit és hallgatók idősorjellemzőit. A hallgatóknál egyértelmű a laborgyakorlaton nyújtott teljesítmény. A fejlesztői cégnél két lehetőségünk adódik, ezek:

- egy adott projektben nyújtott teljesítmény, ez a vizsgálat gyakorlatilag megegyezik a hallgatói teljesítmény vizsgálatával;
- az eddig elkészült szoftverekből származó idősorok vizsgálata.

A fejlesztők esetén célszerű mindkét vizsgálatot elvégezni.

## 6. Összefoglaló

Az idősorok segítségével nyomon követhetjük, hogy az adott rendszerben vagy folyamatban milyen gyakorisággal és időzítéssel jelennek meg a hibák, és ezeket az információkat felhasználhatjuk a rendszer javításához vagy optimalizálásához. Az általunk javasolt eljárással az idősorok eltérő platformokon fejlesztett és futó szoftverek esetén is használhatók vizsgálatokra.

Az idősorok elemzése során az említett Hurst-exponens, autokorrelációs függvény és regressziós polinomok hasznos eszközök lehetnek a hibák előrejelzéséhez. Lehetőséget nyújtanak a megrendelőnek, hogy az adott szoftver vagy szoftverelem fejlesztéséhez melyik szoftverfejlesztő céget válassza ki. Ez különösen fontos lehet működésbiztos (safety-critical), vagy veszélybiztos[4] (mission-critical) fejlesztés esetén.

Ebben a cikkben három jellemző alapján határozzuk meg a vizsgálat tárgyának időbeli viselkedését: a Hurst-exponens alapján, amely megmutatja a folyamat trendjellegét, autokorrelációs függvény alapján, amely grafikusan mutatja az idősor összefüggését és a regressziós polinom segítségével, amely megmutatja a trend irányát.

Ezeket a vizsgálatokat elvégeztük egy fejlesztő cég egy adott projektjére, lásd 3. ábra és két kiválasztott hallgatóra, lásd 4. és 5. ábra.

[4] Erre a kifejezésre nincs igazán jó magyar fordítás.

A cég esetén a projekt sikeres volt. A hallgatóknál a módszer elég jól előrejelezte az évközi jegyeket. Jelenleg az oktatói minősítésen is dolgozunk.

## Felhasznált irodalom


[1] Szilágyi, G. A. „ A légi balesetek fraktáldimenziója," *Repüléstudományi Közlemények,* 28. évf. 2. sz. pp. 41–48. 2016. Online: https://folyoirat.ludovika.hu/index.php/reptudkoz/article/view/4484

[2] B. Lutkevich, Capability Maturity Model (CMM). *Techtarget,* 2022. április. Online: www.techtarget.com/searchsoftwarequality/definition/Capability-Maturity-Model

[3] NEDL, Hurst exponent explained: Long-term memory in time series (Excel). *YouTube,* 2021. január 12. Online: www.youtube.com/watch?v = l08LICz8Ink

[4] S. Aref, Hurst Phenomenon and Fractal Dimensions in Long-Term Yield Dataterm Yield Data. in *Conference on Applied Statistics in Agriculture 1998 – 10th Annual Conference Proceedings.* 1998. pp. 32–42. Online: https://newprairiepress.org/cgi/viewcontent.cgi?article = 1275&context = agstatconference

[5] M. de las Nieves López García, J. P. Ramos Requena, „Different Methodologies and Uses of the Hurst Exponent in Econophysics," *Studies of Applied Economics,* 37. évf. 2. sz. pp. 96–108. 2019. Online: https://doi.org/10.25115/eea.v37i2.2603

[6] B. Davidson, Hurst exponent. *MathWorks,* 2023. szeptember 7. Online: www.mathworks.com/matlabcentral/fileexchange/9842-hurst-exponent

[7] https://github.com/ziyadparekh/MATLAB/blob/master/hurst_exponent.m

[8] P. Dix, *Time Series Forecasting Methods.* Influxdata, 2020. Online: https://www.influxdata.com/time-series-forecasting-methods/#download

[9] NIST, „6.4. Introduction to Time Series Analysis," in *NIST/SEMATECH e-Handbook of Statistical Methods.* 2012. Online: www.itl.nist.gov/div898/handbook/pmc/section4/pmc4.htm

[10] C. Zaiontz, Exponential Regression using a Linear Model. *Real Statistics Using Excel,* [é. n.]. Online: https://real-statistics.com/regression/exponential-regression-models/exponential-regression/

[11] Wikipedia, *Logistic Regression.* [é. n.]. Online: https://en.wikipedia.org/wiki/Logistic_regression

[12] S. Anayah, The Sigmoid in Regression, Neural Network Activation and LSTM Gates. *GitHub,* 2020. január 17. Online: https://suzyahyah.github.io/machine%20learning/2020/01/17/Sigmoid.html

[13] R. Agrawal, Master Polynomial Regression With Easy-to-Follow Tutorials. *Analytics Vidhya,* 2021. július 9. Online: www.analyticsvidhya.com/blog/2021/07/all-you-need-to-know-about-polynomial-regression/

[14] Polynomial Regression – Least Square Fittings. *Matrixlab,* [é. n.]. Online: www.matrix-lab-examples.com/polynomial-regression.html

## Application of Time Series to Predict Software Reliability

*Choosing a software manufacturer is a serious challenge for the customer. Since we cannot apply the test methods of classic production processes in the case of software, we must therefore examine the manufacturer. It is quite clear that a specific manufacturer must have appropriate qualifications. But what if there are several manufacturers with the same certification, and you have to choose between them.*

*In this article, we propose a method that examines the general development activities of manufacturers under appropriate restrictions based on the operational and development errors made, arranging them in time series.*

*The article describes the method, its advantages, limitations, obstacles to its application and an unexpected consequence in education.*

**Keywords:** *software quality estimation, time series*

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