DSPNSim: Szimulációs Szoftver Determinisztikus és Sztochasztikus Petri Hálókra

DSPNSim: Simulation Software for Deterministic and Stochastic Petri Nets

Molnár András[a], **Horváth Ádám**[b]

[a]Nyugat-magyarországi Egyetem, gazdaságinformatikus M.Sc. hallgató
molnaran@gain.nyme.hu

[b]Nyugat-magyarországi Egyetem, Informatikai és Gazdasági Intézet
horvath@inf.nyme.hu

Absztrakt: A teljesítményvizsgálatoknak számos eszköze áll rendelkezésünkre, különösen a markovi modellek esetében. Sok folyamat azonban nem tekinthető markovinak, például egy gyártási folyamatban a különböző munkafázisok eloszlása általában determinisztikus. Determinisztikus és sztochasztikus késleltetésű folyamatok jól leírhatók egy determinisztikus és sztochasztikus Petri háló (DSPN) segítségével, melyet először Ajmone Marsan és Chiola javasolt [1]. A DSPN-ek analitikus megoldása néhány speciális esetet leszámítva [2, 3] arra az esetre korlátozódik, amikor a háló minden állapotában legfeljebb egy determinisztikus átmenet engedélyezett. Analitikus megoldás hiányában szimulációs szoftvereket használhatunk a háló egyensúlyi vagy tranziens eloszlásának meghatározására.
Bár vannak a területen ismert szoftverek, ezek némelyike elavult (DSPNExpress [4]), némelyike pedig tömeges eredményeket csak nehézkesen képes előállítani (GreatSPN [5], TimeNET [6]). Így született meg a DSPNSim, egy saját fejlesztésű szoftver, amely lehetővé teszi a DSPN paraméterfájllal való leírását, a kötegelt futtatást, s egy olyan statisztikai modullal egészül ki, amely biztosítja a szimuláció pontosságát (relatív hiba, konfidenciaszint). A statisztikai modulban egyes eljárásokat pontos közelítő algoritmusok segítségével valósítottunk meg, amely a szimulációnál kritikus futási időt rövidíti le.

Kulcsszavak*:* determinisztikus és sztochasztikus Petri hálók, szimulációs szoftver

Abstract: Performance analysis can be carried out in several ways, especially in case of Markovian models. However, many processes cannot be considered Markovian. For example, in a manufacturing process, the distribution of the work phases are generally deterministic. A process in which the delays are either deterministic or exponential, can be described by a deterministic and stochastic Petri net (DSPN), which was first proposed by Ajmone Marsan and Chiola [1]. Except some special cases [2, 3], the analytical solution of DSPNs is restricted to the case when only one deterministic transition is enabled in each marking. Therefore, simulation software can be used to obtain the steady state or transient solution of the DSPN.
Although there are some well-known simulation software for DSPNs, some of them are obsolete (DSPNExpress [4]), while some are not appropriate for generating results in batch mode (GreatSPN [5], TimeNET [6]). These facts were the main motivation factors of developing a new simulation software, which supports the parametric description of DSPN and the batch processing. Moreover, DSPNSim is complemented with a statistical module, which ensures the precision of the simulation (relative error, confidence level). In the statistical module, certain procedures were implemented using accurate approximate methods. So, the generally critical run time of the simulation process is shortened.

Keywords: deterministic and stochastic Petri nets, simulation software

1. Bevezetés

A determinisztikus és sztochasztikus Petri hálók olyan folyamat leírására alkalmasak, melyben a késleltetések eloszlása determinisztikus vagy exponenciális lehet, vagyis a DSPN a determinisztikus átmenetek megengedésével tekinthető a sztochasztikus Petri háló (SPN) kiterjesztésének is. A DSPN így nagyobb modellezési erőt jelent az SPN-nél, azonban a modell kiértékelésekor nagyobb korlátokba ütközünk, mint az SPN esetében. Általánosan csak abban esetben tudunk módszert a háló analitikus megoldásának előállítására, ha a háló minden állapotában legfeljebb egy determinisztikus átmenet engedélyezett [1]. Mivel ez egy erős megszorítás, a gyakorlatban sokszor merül fel az igény szimulációs technikák alkalmazására.

Bár Petri hálók szimulációjára számos alkalmazás született, s ezek közül több képes DSPN analitikus megoldására vagy szimulációjára, az egyes szoftverek mindegyike esetében hiányosságokkal szembesültünk. A DSPNExpress szoftver új, hatékony eljárásokat tartalmazott a DSPN-ek analitikus megoldására, ennek ellenére a szoftvert ma már nem fejlesztik tovább, és a régi változat sem elérhető.

A TimeNET szoftvert a mai napig fejlesztik az ilmenaui műszaki egyetemen (Technische Universität Ilmenau). Fejlesztői az általánosan alkalmazott eljárásokat építették bele a Petri hálót megoldó modulokba. Jól paraméterezhető a szimulációkhoz tartozó statisztikai modul. Grafikus felülettel rendelkezik, biztosítva a Petri hálók hatékony tervezését. A szoftver legfőbb hátránya, hogy a grafikus felületen nehézkes a szimulációs eredmények generálása, mivel a szimulációkat csak egyesével, a grafikus felületen történő paraméterezéssel futtathatjuk. További hátránya a szoftvernek, hogy a tüzelési politika nem állítható paraméter, hanem minden esetben az „enabling memory” szabályt [1] alkalmazza.

A GreatSPN szoftvert a torinói egyetemen (Università di Torino) kezdték fejleszteni 1985-ben. Az eszköz azóta számos változáson ment keresztül. A legnagyobb hátránya, hogy a jelenlegi verzió instabil, emellett a TimeNET-hez hasonlóan nehézkes a szimulációs eredmények tömeges generálása.

A DSPNSim a szimulációs eredmények hatékony generálására született meg. Grafikus felülettel nem rendelkezik, a szimulálni kívánt DSPN-t és a paramétereket egy XML fájl segítségével írhatjuk le. Az eredmények pontosságát a programhoz szorosan kapcsoló statisztikai modul biztosítja.

A cikk további felépítése a következő. A második fejezetben röviden összefoglaljuk a DSPN formalizmus alapjait. A harmadik fejezetben leírjuk a DSPNSim felépítését, míg a negyedik fejezetben összehasonlítások segítségével értékeljük annak eredményességét. Az ötödik fejezetben összefoglaljuk a cikket és megemlítjük a továbbfejlesztési lehetőségeket.

2. DSPN formalizmus

Ebben a fejezetben rövid leírást adunk a DSPN formalizmusról, míg részletes leírást például Ajmone Marsan és Chiola munkájában található [1].

A DSPN formálisan egy irányított gráf, amely két típusú csomópontot tartalmaz: helyeket és tranzíciókat. A helyek a rendszer állapotváltozóinak, míg a tranzíciók az állapotváltozásokat előidéző eseményeknek feleltethetők meg. A háló élei a helyeket kötik össze a tranzíciókkel, s a rendszerállapotok és a rendszerben értelmezett események kapcsolatát írják le. A helyek tokeneket tartalmazhatnak, s rendszerállapotok leírását az adja, hogy az adott állapotban hány token van a háló helyein. Általánosságban a rendszerállapotot

M-mel jelöljük, míg *M*(*p*) jelöli az *M* rendszerállapotban a *p* helyen lévő tokenek számát. A következőkben néhány DSPN-nel kapcsolatos alapfogalmat írunk le.

A DSPN rendszer a következő paraméterekkel írható le:

$$(P, T, I, O, H, M_0, \tau, \omega, e_{t_1,t_2}),$$

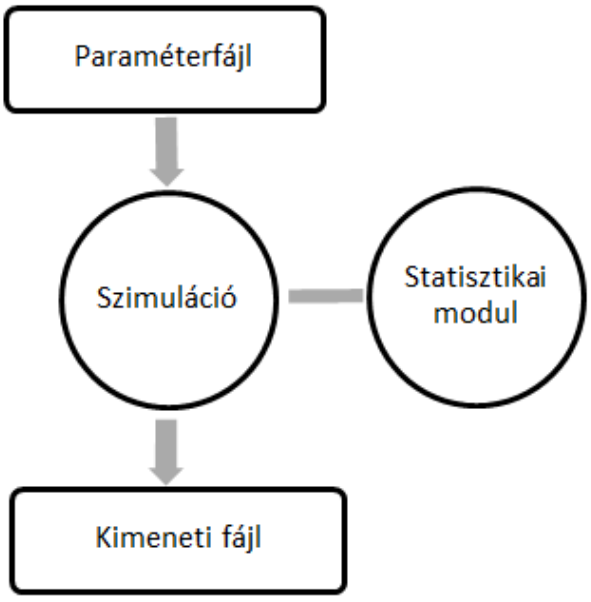
ahol:

- P *a helyek véges halmaza. Egy* $M \in \mathbb{N}^{|P|}$ *rendszerállapot minden* $p \in P$ *helyen lévő tokenek számát definiálja.*
- T *is a tranzíciók véges halmaza, amely a következő diszjunkt halmazokra osztható:* T^Z *a rögtöntranzíciók halmaza,* T^E *az exponenciális tranzíciók halmaza, míg* T^D *a determinisztikus tranzíciók halmaza. M* $P \cap T = \emptyset$.
- $I, O, H : \mathbb{N}^{|P|} \to \mathbb{N}$ *sorrendben a p-ből t-be menő bemenő élek, a t-ből p-be menő kimenő élek, és a p-ből t-be menő élek multiplicitása.*
- $M_0 \in \mathbb{N}^{|P|}$ *a kezdeti rendszerállapot.*
- $\tau : \mathbb{N}^{|P|} \to \mathbb{R}^+$ *az átlagos késleltetés* $\forall t \in T^E \cup T^D$*esetén (*$\tau$ *függhet az aktuális rendszerállapottól).*
- $\omega : \mathbb{N}^{|P|} \to \mathbb{R}^+$ *a tüzelési súly* $\forall t \in T^Z$ *esetén (*ω *függhet az aktuális rendszerállapottól).*
- $e_{t_1,t_2} : \mathbb{N}^{|P|} \to \{R, C\}$ *a* t_2*-re alkalmazott a tüzelési politika, amikor* t_1 *tüzel* $\forall t_1 \in T^Z \cup T^E, t_2 \in T^D$ *(*e_{t_1,t_2} *függhet az aktuális rendszerállapottól).* R *és* C *sorrendben az újraindítás (restart) és a folytatás (continue).*

3. A DSPNSim program leírása

A programot felépítését és működését tekintve a következő részekre bonthatjuk fel (1. ábra):

- szimulációt végző egység,
- statisztikai modul,
- beolvasást végző egység.



1. ábra. A program felépítése

A szimuláció megkezdése előtt a külső paraméterfájl beolvasásra kerül. Ebben a fájlban adhatjuk meg a szimulálni kívánt modell Petri-hálóját és további, szimuláció szempontjából fontos paramétereket. Az adatok leírása XML formátumban történik, ezért a háló megadása nem igényel külön eszközt, valamint átlátható hierarchiát eredményez.

A Petri-háló megadásánál meghatározhatunk helyeket és különböző tranzíciókat is. Helyek esetében kötelező megadni nevet, opcionális paraméterként definiálhatunk kezdeti tokenszámot, ennek segítségével megadható a kezdeti tokeneloszlás a modellben.

A tranzíciók megadásánál kötelezően meg kell határozni egy azonosítót, továbbá megadható a bemenő illetve kimenő helyek listája, valamint az egyes helyekhez tartozó élsúly is. Az élsúlyoknál a nullás érték különleges szereppel bír, ez jelenti ugyanis a tiltó élt. A megadható tranzíciók közül először a rögtön tüzelő tranzíciót tekintjük át. Itt késleltetést nem definiálunk, hiszen ez a tranzíció várakozás nélkül mozgatja a tokeneket. Opcionális paraméterként azonban megadhatunk prioritást, melynek segítségével a konkurens tranzíciók között definiálhatjuk a tüzelési sorrendet. Exponenciális késleltetésű tranzíció esetén az átlagos késleltetést tudjuk megadni, amely a tüzelési intenzitás reciproka. Megadhatjuk a kiszolgáló típusát is, amely exkluzív illetve végtelen kiszolgáló lehet. Végtelen kiszolgáló esetén a tüzelési intenzitás a bemeneti helyeken található tokenek számától függ, exkluzív esetében ez az intenzitás konstans. Rögtön tüzelő tranzíciók és exponenciális tranzíciók leírásakor tudjuk megadni a determinisztikus tranzíciókra vonatkozó gyújtási politikát is. A gyújtási politika azt határozza meg, hogy egy adott $t \in T^E \cup T^Z$ tranzíció tüzelése esetén egy $t \in T^D$ tranzíció tüzelési eseményét újrageneráljuk-e. Az alapértelmezett szabály a *folytatás*, ez lép életbe, ha nem adunk meg gyújtási politikát. A determinisztikus késleltetésű tranzíciókat az exponenciálishoz hasonlóan a késleltetéssel definiáljuk.

A Petri-hálón kívül a szimuláció beállításai is a bemenő paraméterfájlban írhatók le, meghatározhatjuk a szignifikáns helyeket, amelyekre a szimuláció a pontosságot mérni fogja, eldönthetjük, hogy a szimuláció rendelkezzen-e felfűtési időszakkal, valamint, hogy ez a felfűtési időszak milyen pontosság elérése után fejeződjön be. Definiálhatunk minimális tranziens illetve minimális futási időt, továbbá maximális futási időt is. A szignifikáns értékek vizsgálata szempontjából megadható a konfidenciaszint és a maximális relatív hiba. A program lehetőséget ad kötegelt futtatásra, ez az opció is ebben a fájlban határozható meg. Végül megadhatjuk a kimeneti fájl nevét és helyét.

A paraméterfájl beolvasása után a szimulációt végző egység a beolvasott hálón a megadott paraméterek mellett elkezdi a szimulációt. Ebben az egységben található a logika túlnyomó része, ez a program központi része, mely a többi osztályt és annak metódusait mint szolgáltatást hívja meg. A futás végét háromféleképpen érhetjük el: *i*) az események elfogynak az eseménysorból; *ii*) elérjük a maximális futási időt; *iii*) elérjük a megadott pontosságot a szignifikáns helyek vizsgálatakor.

A pontosság vizsgálata a szimuláció minden iterációjában megtörténik, erre a mérésére a statisztikai modul ad lehetőség. Ez a modul csupán a szimulációs modul által használt metódusokat tartalmazza, s a metódusok kifejezetten szimulációs célra optimalizált eljárásokat használnak.

Az itt található intervallumbecslő függvény a szignifikáns helyek átlagos tokenszámára határozza meg a konfidenciaintervallumot (a paraméterfájlban megadott paraméterek mellett). Az adott pontosság elérésének vizsgálata során az intervallum hosszát vetjük össze az általunk definiált megengedett hibával. A 3.1-es egyenlőtlenség a pontosság vizsgálatát írja le. A képlet bal oldala a maximális relatív hibát fejezi ki, a jobb oldal pedig az általunk definiált megengedett hiba mértékét jelenti az adott hely átlagos tokenszámára nézve.

$$z_{\alpha/2} \cdot \hat{S}_{\bar{\eta}} < \bar{x} \cdot e \tag{3.1}$$

Mivel ez a vizsgálat a szimuláció minden iterációjában kiszámításra kerül, ezért fontos, hogy a szükséges paraméterek számítása is költséghatékony legyen. Az aktuális iterációnál történő számításoknál a megelőző állapotban meghatározott eredmények kerülnek felhasználásra. Ez az átlagszámításnál súlyozott átlag használatával valósult meg (3.2-es egyenlet).

$$\bar{x}_n = \frac{(\sum_{i=1}^{n-1} t_i) \cdot \bar{x}_{n-1} + t_n \cdot x_n}{\sum_{i=1}^{n} t_i} \tag{3.2}$$

A szórásnégyzet számításához használt képletnél is ez a logikát használtuk (3.3-as egyenlet).

$$S_{n+1}{}^2 = \frac{\sum_{i=1}^{n} t_i}{\sum_{i=1}^{n+1} t_i} S_n{}^2 + \frac{t_{n+1}(x_{n+1} - \bar{x}_n)^2}{\sum_{i=1}^{n+1} t_i} + d^2 - 2d\bar{x}_{n+1} + 2d\bar{x}_n, \tag{3.3}$$

ahol

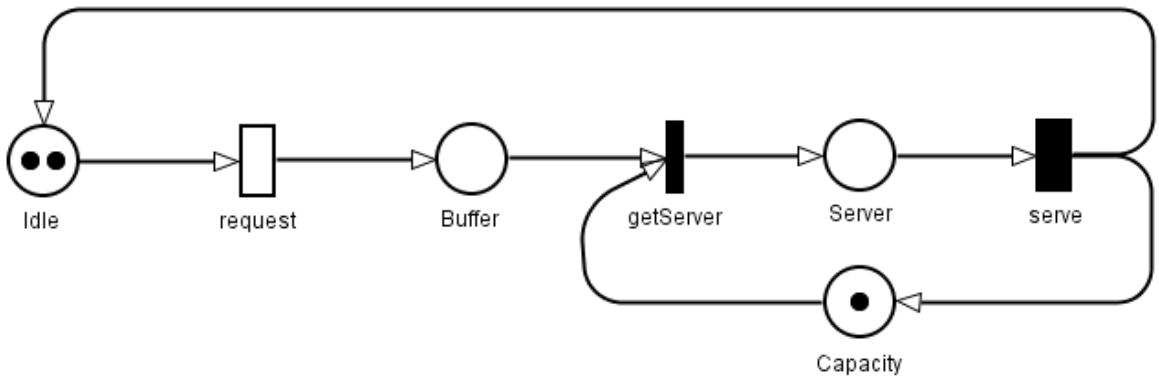
$$d = \bar{x}_{n+1} - \bar{x}_n. \tag{3.4}$$

Miután a szimuláció véget ért, a program a megadott fájlba kiírja a szimulációs eredményeket.

4. Az eredmények értékelése

Ebben a fejezetben a szimuláció által szolgáltatott eredményeket vizsgálunk meg két modell esetében: először egy egyszerűbb rendszert vizsgálunk, melynek könnyen megkapható az analitikus megoldása, majd egy összetettebb rendszert, melyben az eredményeket összevetjük a TimeNET [6] által meghatározott eredményekkel.

Az egyszerűbb rendszer az M/D/1/2/2-es sor. A vizsgálatot 95%-os konfidenciaszinten, 1%-os maximális hiba mellett végeztük. A következő ábrán a sornak megfelelő Petri-hálója látható (2. ábra).



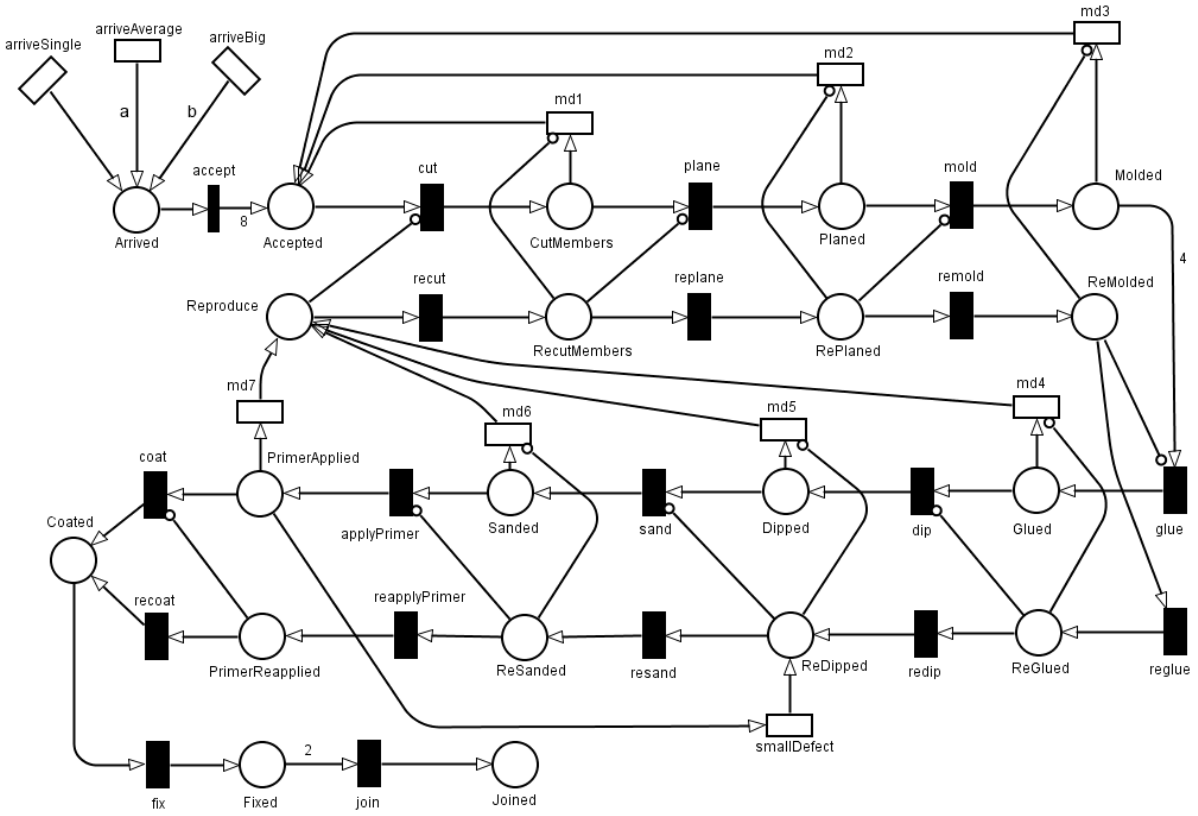
2. ábra - Az M/D/1/2/2-es rendszer Petri hálója

A hálóban található tüzelők a következők: a „*request*” exponenciális késleltetésű (a késleltetés értéke 0.1), végtelen kiszolgáló típusú, a „*getServer*” késleltetés nélküli, a „*serve*” pedig determinisztikus késleltetésű (a késleltetés értéke 0.6). A következő táblázat (1. táblázat) az analitikus eredményeket veti össze a DSPNSim által szolgáltatott eredményekkel.

1. táblázat – Az analitikus eredmények összevetése a DSPNSim programmal

Helyek	Analitikus eredmény	DSPNSim
Idle	1.14363374	1.142999951
Buffer	0.17018601	0.170601133
Server	0.68618025	0.686398917

Az összetettebb DSPN modell, melyet vizsgáltunk, a faablakok gyártási folyamatát modellezi (3. ábra, a részletekért ld. [7]). Ebben az esetben egy tranziens vizsgálatról van szó, melynek célja, hogy az egyes lépésekhez szükséges munkaerőt optimálisan osszuk el. A szimulációban a paramétereket valós adatok alapján állítottuk be, melyek egy naptári évre vonatkoznak[1]. Tranziens szimuláció esetén a pontosságot kötegelt futtatással tudjuk növelni, amikor az eredmények független egymást követő szimulációk átlagából adódnak.



3. ábra – A faablakok gyártási folyamatának DSPN modellje

Az eredményeket a DSPNSim 5000 független szimuláció átlagából számolta ki, ezeket vetettük össze a TimeNET szimulátorával, mely 99%-os konfidenciaszinten, 1%-os maximális relatív hiba mellett futott (2. táblázat).

[1] A 2014-es magyar munkanaptár alapján 2088 órával számoltunk.

2. táblázat – A TimeNET és a DSPNSim eredményeinek összevetése

Helyek	TimeNet	DSPNSim
Accepted	189.81933	190.8610859
CutMembers + RecutMembers	0.13467	0.133401
Planed + RePlaned	0.63745	0.63921723
Molded + ReMolded	3.25834	3.261711721
Glued + ReGlued	0.10666	0.107125
Dipped + ReDipped	0.70611	0.710063224
Sanded + ReSanded	0.35967	0.356171065
PrimerApplied+ PrimerReapplied	0.35056	0.354114951
Coated	584.55711	592.564000
Fixed	0.52422	0.510605551
Joined	3021.378	3017.342600

5. Összefoglalás

Bár sok esetben hasznos a determinisztikus és sztochasztikus Petri hálós modellek alkalmazása, numerikus megoldásuk gyakran problémákba ütközik, az elérhető szimulációs szoftverek pedig különböző korlátokkal rendelkeznek, főként a tömeges eredmények generálása nehézkes. A DSPNSim grafikus felület nélkül, egy paraméterfájl segítségével futtatható, így hatékonyan szállítja az eredményeket.

A tisztán szimulációs szoftverek természetesen nem korlátozódnak a DSPN-ekben megengedett eloszlásokra. A program egyik továbbfejlesztési iránya a további eloszlások beépítése, illetve ennek megfelelően a gyújtási politika finomítása. A másik továbbfejlesztési irány a szimuláció pontosságának meghatározása tranziens szimuláció esetén, ugyanis a jelenlegi verzióban a szimuláció pontosságának növelése a független futtatások számának növelésével lehetséges.

Irodalomjegyzék

[1] M. A. Marsan and G. Chiola, „On petri nets with deterministic and exponentially distributed firing times,” in Advances in Petri Nets 1987, pp. 132-145, Springer Berlin Heidelberg, 1987.

[2] C. Lindemann and G. S. Shedler, „Numerical analysis of deterministic and stochastic petri nets with concurrent deterministic transitions,” Perform. Eval., vol. 27-28, pp. 565-582, Oct. 1996.

[3] C. Lindemann and A. Thümmler, „Transient analysis of deterministic and stochastic petri nets with concurrent deterministic transitions,” Performance Evaluation, vol. 36, pp. 35-54, 1999.

[4] C. Lindemann, „DSPNexpress: A software package for the efficient solution of Deterministic and Stochastic Petri Nets.” In: Proc. 6th Int. Conf. on Modelling Techniques

and Tools for Computer Performance Evaluation, Edinburgh, Great Britain, pp. 15–29 (1992)

[5] E. A. Gilberto, S. Donatelli: „DSPN-Tool: A New DSPN and GSPN Solver for GreatSPN," In Proc. of the Seventh International Conference on the Quantitative Evaluation of Systems (QEST), pp.79-80, 15-18 Sept. 2010.

[6] A. Zimmermann, M. Knoke, „TimeNET 4.0: A software tool for the performability evaluation with stochastic and colored Petri nets; user manual," TU, Professoren der Fak. IV, 2007.

[7] Á. Horváth, „Usability of Deterministic and Stochastic Petri Nets in the Wood Industry: a Case Study," in Proc. of the 2nd International Conference on Computer Science, Applied Mathematics and Applications, pp. 119-127, 2014.