Generatory zdarzeń do symulacji systemów webowych z wykorzystaniem TCPN

Andrzej BOŻEK, Tomasz RAK, Dariusz RZOŃCA

Katedra Informatyki i Automatyki
Politechnika Rzeszowska

28 maja 2025

RZESZOW UNIVERSITY OF TECHNOLOGY

Spis treści

- Wprowadzenie
- Podstawowe modele generatorów [1]
  - QPN
  - HTCPN
- Klasyfikacja modeli generatorów [2]
  - Założenia
  - Implementacja
  - Przykład – wyniki symulacji
- Wnioski
- Cytowania, zastosowania, dalsze prace

---

[1] Rak, T.; Rzońca, D. Recommendations for Using QPN Formalism for Preparation of Incoming Request Stream Generator in Modeled System. Appl. Sci. 2021, 11, 11532. https://doi.org/10.3390/app112311532

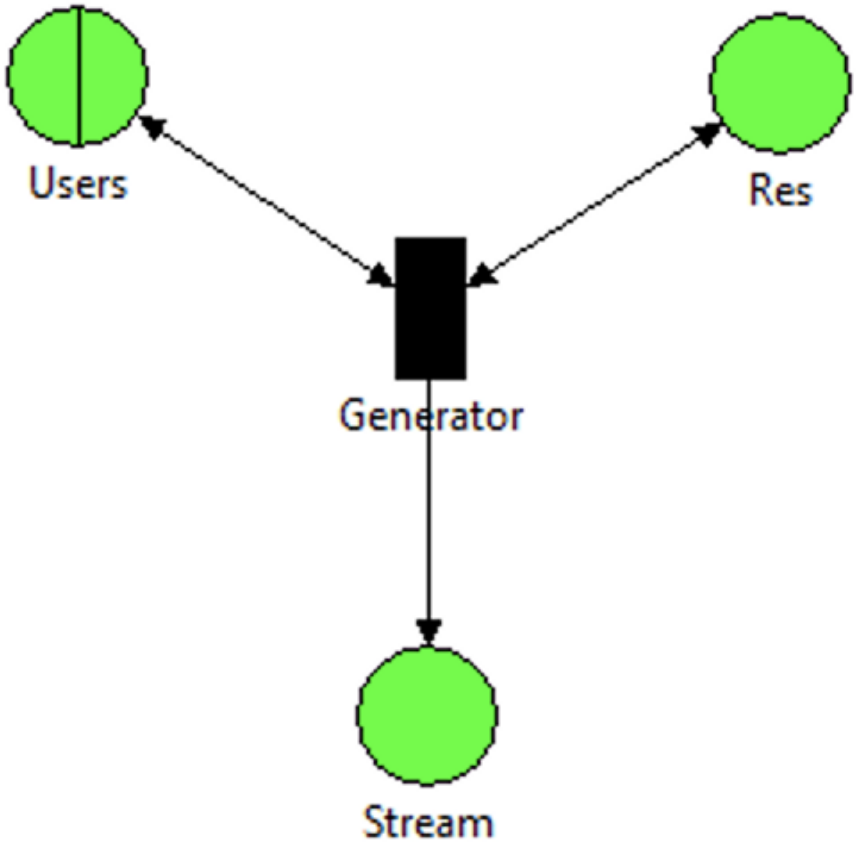
[2] Bożek, A.; Rak, T.; Rzońca, D. Timed Colored Petri Net-Based Event Generators for Web Systems Simulation. Appl. Sci. 2022, 12, 12385. https://doi.org/10.3390/app122312385

Generatory obciążenia

Różne podejścia

Curiel, M.; Pont, A. Workload Generators for Web-Based Systems: Characteristics, Current Status, and Challenges. IEEE Commun. Surv. Tutorials 2018, 20, 1526–1546.
Magalhães, D.; Calheiros, R.N.; Buyya, R.; Gomes, D.G. Workload Modeling for Resource Usage Analysis and Simulation in Cloud Computing. Comput. Electr. Eng. 2015, 47, 69–81.
Gozhyj, A.; Kalinina, I.; Gozhyj, V.; Vysotska, V. Web Service Interaction Modeling with Colored Petri Nets. In Proceedings of the 2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Metz, France, 18–21 September 2019; Volume 1, pp. 319–323.
Gaur, N.; Joshi, P.; Jain, V.; Srivastava, R. Coloured Petri Nets Model for Web Architectures of Web and Database Servers. Int. J. Comput. Inf. Eng. 2015, 9, 2066–2075.
Kounev, S.; Lange, K.D.; von Kistowski, J. Systems Benchmarking: For Scientists and Engineers; Springer: Berlin/Heidelberg, Germany, 2020.

Ograniczenia w modelach QPN

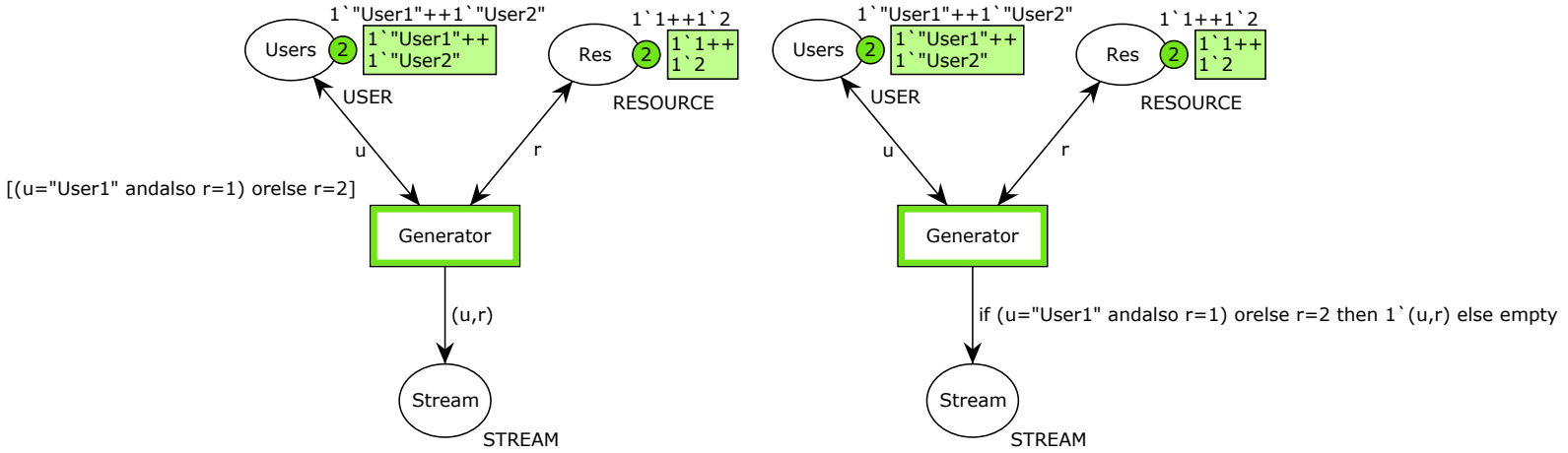


Wyniki symulacji – Generator strumieniowy (Kolory tokenów)

Tryby	Wagi w	Kolory tokenów	Średnia populacja tokenów dla *Stream*
$mode_1$	1.0	$u1r1$	9,978.882
$mode_2$	2.0	$u1r2$	19,827.981
$mode_3$	3.0	$u2r1$	29,978.301
$mode_4$	4.0	$u2r2$	39,679.93
$mode_1$	1.0	$u1r1$	14,195.611
$mode_2$	2.0	$u1r2$	28,625.108
$mode_3$	1.0	$u2r1$	14,308.058
$mode_4$	3.0	$u2r2$	42,688.806
$mode_1$	1.0	$u1r1$	24,931.845
$mode_2$	1.0	$u1r2$	25,170.6
$mode_3$	0.0	$u2r1$	0.0
$mode_4$	2.0	$u2r2$	50,286.744

(...)

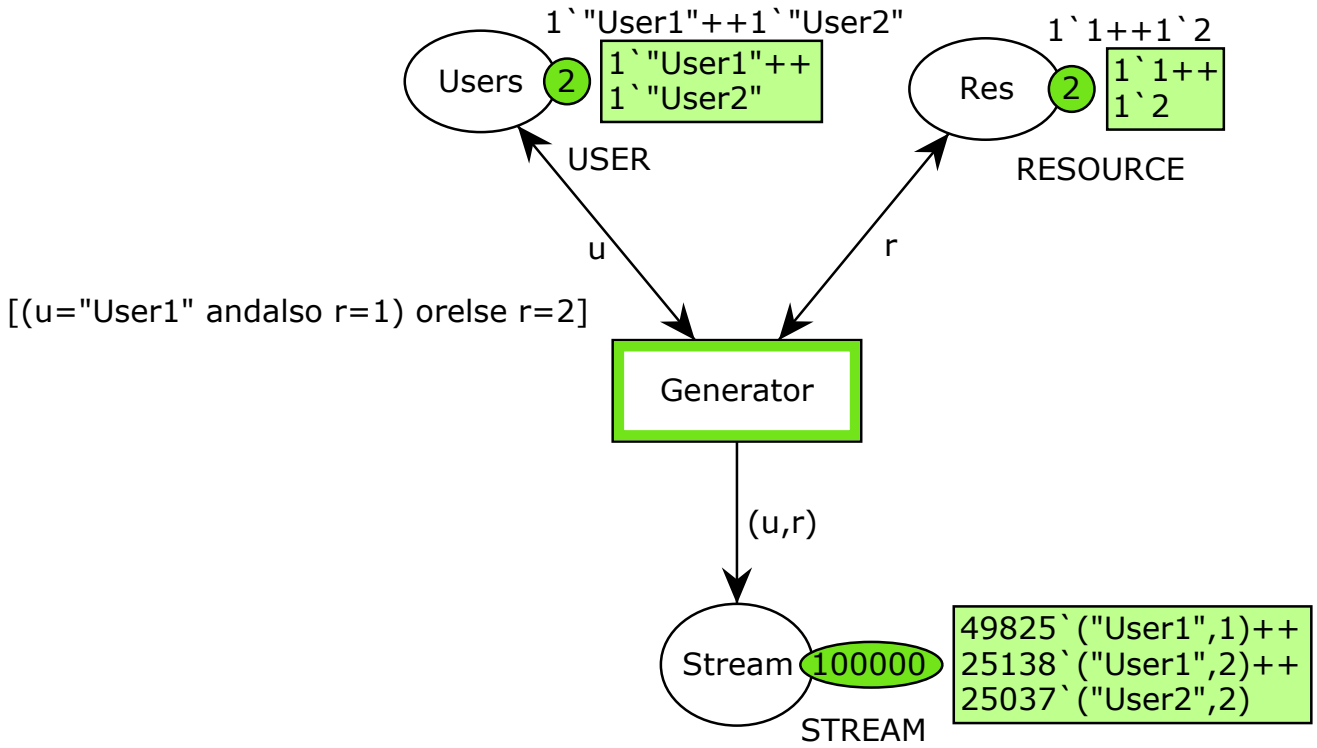
Ograniczenia w modelach CPN



- Powiązanie użytkowników z zasobami inskrypcją w dozorze
- User1 może używać Res1 lub Res2, User2 tylko Res2

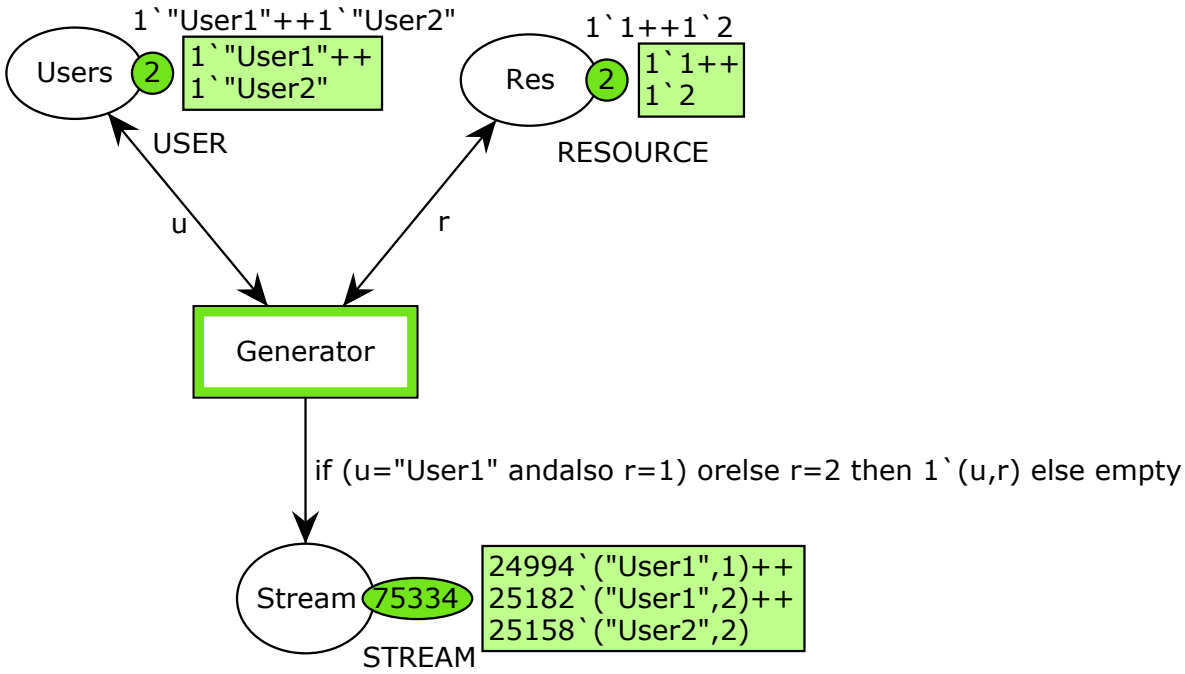
- Powiązanie użytkowników z zasobami inskrypcją na łuku
- User1 może używać Res1 lub Res2, User2 tylko Res2

Wyniki symulacji – Generator strumieniowy 1 (Dozór)



- Przejście zostało odpalone 100 tys. razy
- Jedna z wygenerowanych par występuje częściej niż inne

Wyniki symulacji – Generator strumieniowy 2 (Łuk)



- Przejście zostało odpalone 100 tys. razy, ale tylko ∼75 tys. tokenów zostało wygenerowanych
- Rozkład każdej pary jest równy

Spójna klasyfikacja generatorów

- Uwzględnienie upływu czasu
  - **Bezczasowe** – sieć jest bezczasowa (nie ma wymiaru czasu), albo wszystkie zdarzenia są generowane w czasie 0
  - **Czasowe** – zdarzenia są emitowane w różnych chwilach czasu
- Powtarzalność działania
  - **Deterministyczne** – każde powtórzenie symulacji generuje identyczne zdarzenia
  - **Stochastyczne** – niektóre parametry zdarzeń są warunkowane losowo i zmieniają się w kolejnych powtórzeniach symulacji

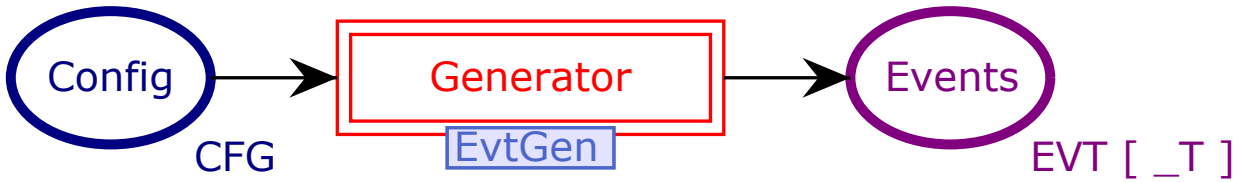
	deterministyczne	stochastyczne
bezczasowe	✓	✓
czasowe	✓	✓

Jednolity interfejs generatorów

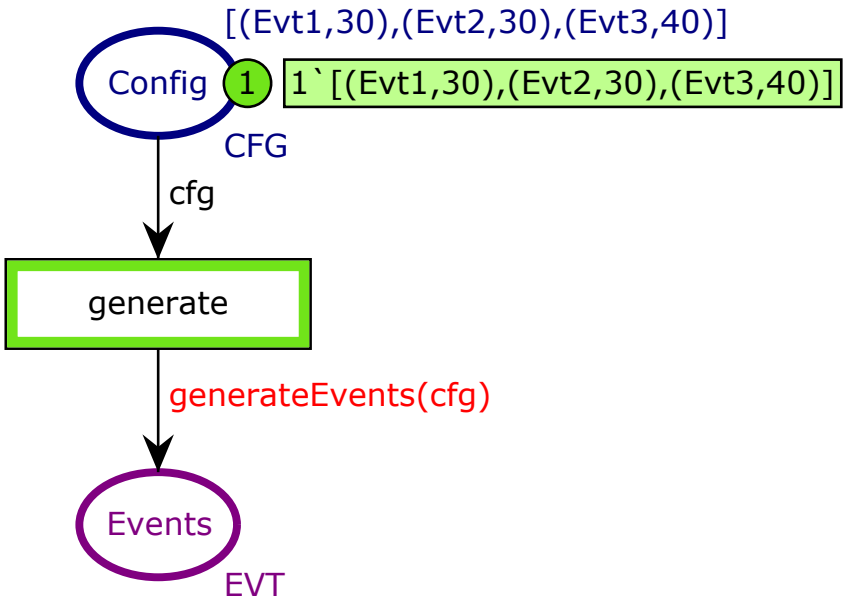
- Rodzaje zdarzeń są rozróżniane za pomocą dedykowanego zbioru kolorów `EVT` (bezczasowy) lub `EVT_T` (czasowy)
- Konfigurację generatora opisuje znakowanie początkowe miejsca `Config`

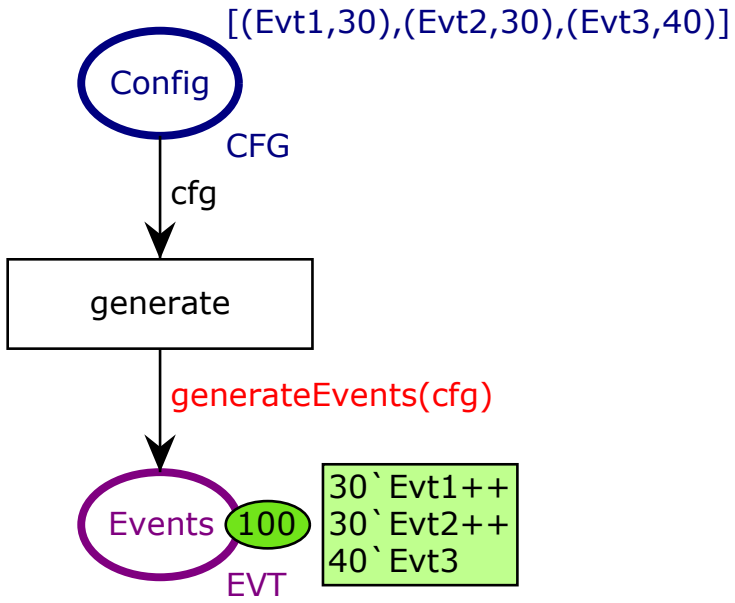
```
colset ECFG = product EVT * PARAMS;
colset CFG = list ECFG;
```

- Wygenerowane zdarzenia są reprezentowane przez tokeny (typ `EVT` lub `EVT_T`) wprowadzane do miejsca `Events`



Generator bezczasowy deterministyczny

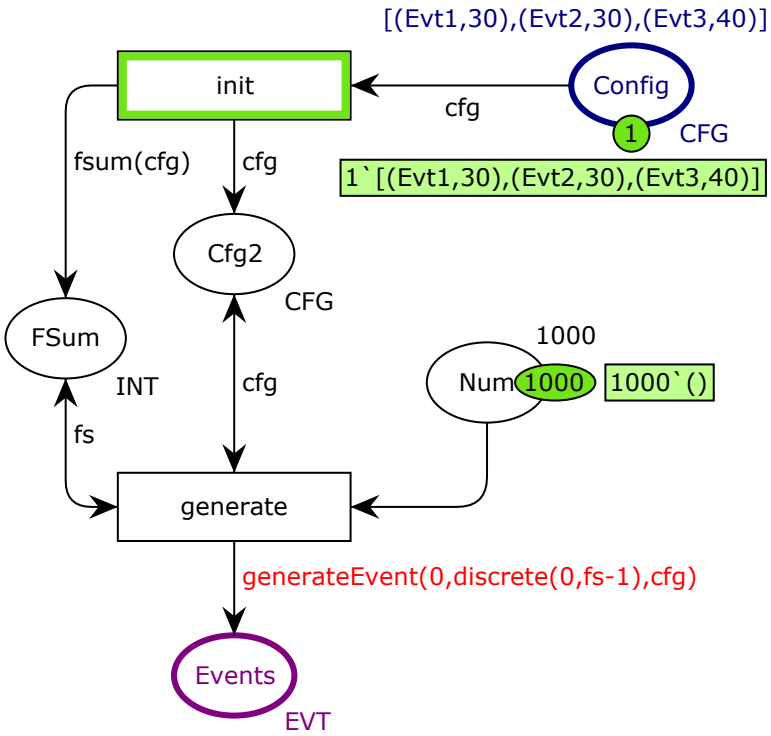


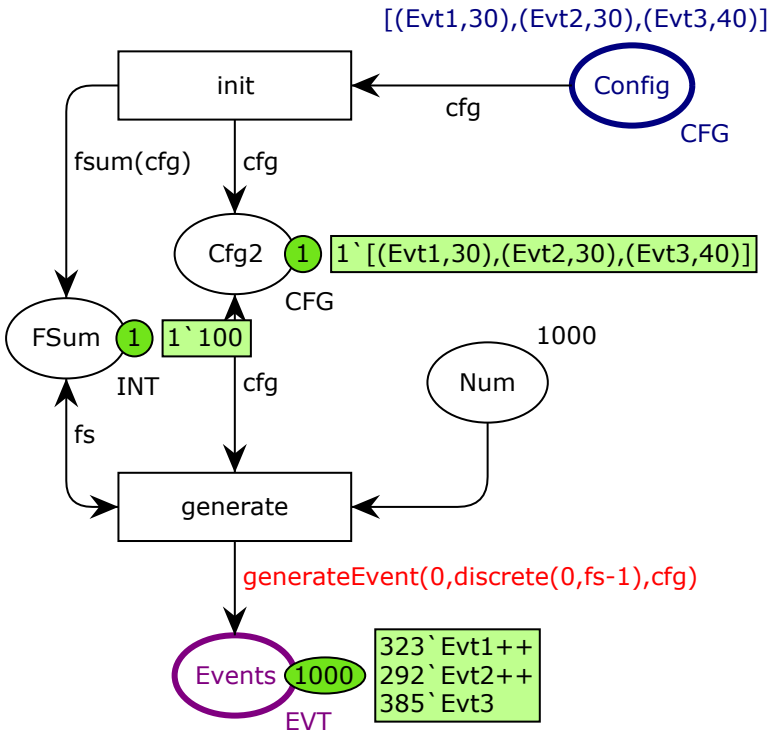



```
fun generateEvents([]) = empty
| generateEvents((e,n)::r)= (n'e) ++ generateEvents(r);
```

Funkcja `generateEvents` przekształca znakowanie konfiguracyjne na tokeny zdarzeń w odpowiedniej liczbie.

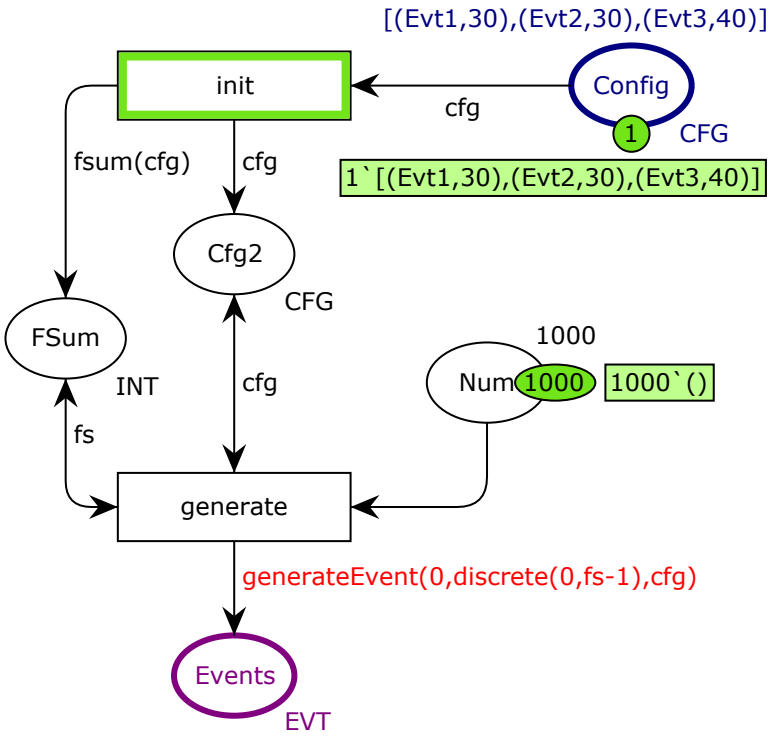
Generator bezczasowy stochastyczny

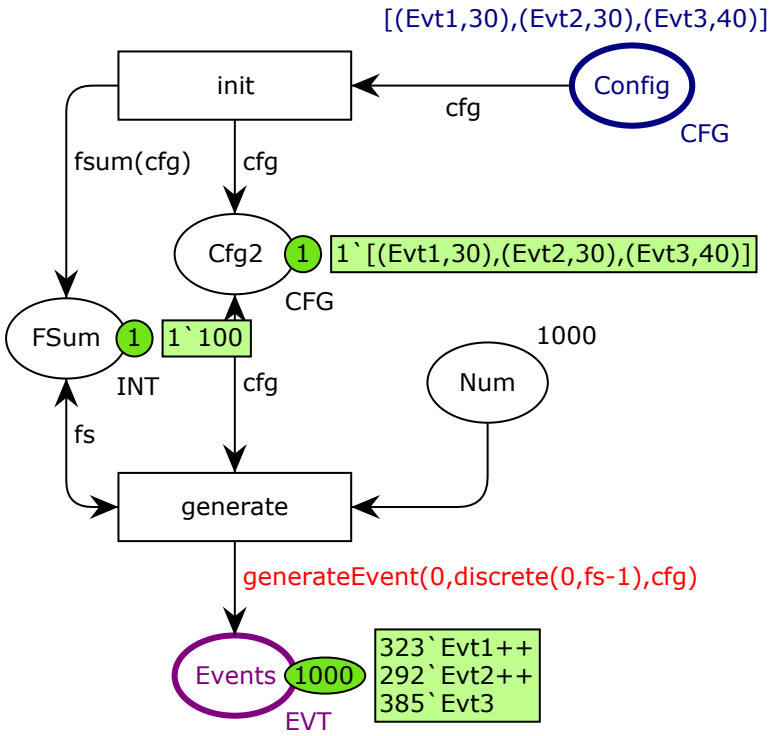




Liczby w znakowaniu miejsca `Config` reprezentują względną szansę wystąpienia danego zdarzenia. Wartość znakowania początkowego miejsca `Num` określa łączną liczbę zdarzeń.

Generator bezczasowy stochastyczny






```
fun fsum([]) = 0 | fsum((e,f)::r) = f+fsum(r);

fun generateEvent(c,s,[]) = empty | generateEvent(c,s,(e,f)::r)
   = if c<=s andalso c+f>s then 1'e else generateEvent(c+f,s,r);
```

Generator bezczasowy stochastyczny

Konfiguracja generatora

- znakowanie miejsca `Config`: $[(e_1, f_1), \ldots, (e_i, f_i), \ldots, (e_K, f_K)]$,
- znakowanie miejsca `Num`: N

Prawdopodobieństwo wygenerowania zdarzenia e_i wynosi

$$\pi_i = \frac{f_i}{\sum_{i=1}^{K} f_i}.$$

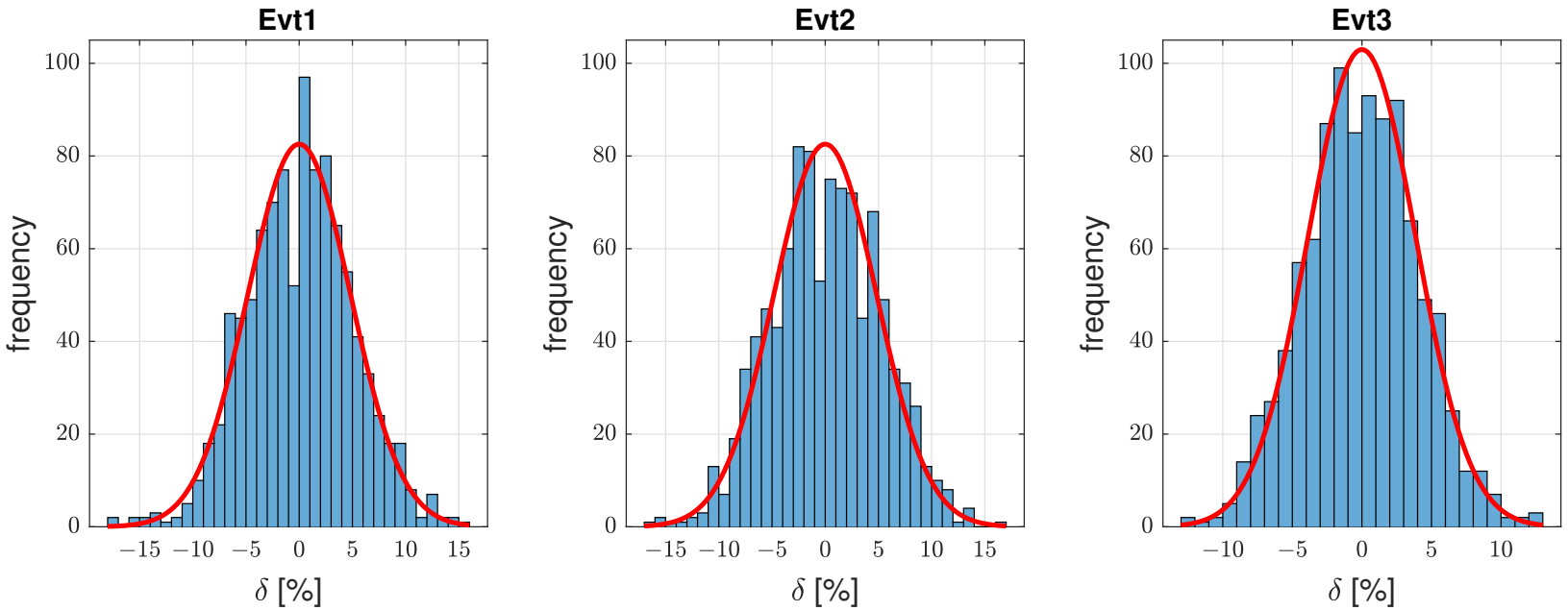
Liczba zdarzeń e_i wygenerowanych w N próbach jest zmienną losową o rozkładzie dwumianowym, który w granicy $N \to \infty$ dąży do rozkładu normalnego (w praktyce dla $N\pi_i > 5$) o parametrach

$$\mu_i = N\pi_i, \qquad \sigma_i = \sqrt{N\pi_i(1-\pi_i)}.$$

Generator bezczasowy stochastyczny

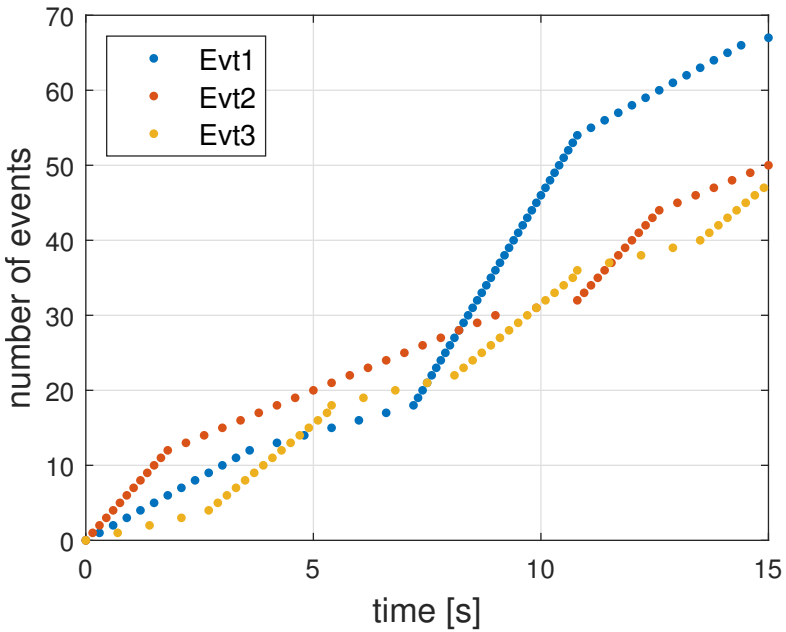
Przykład:

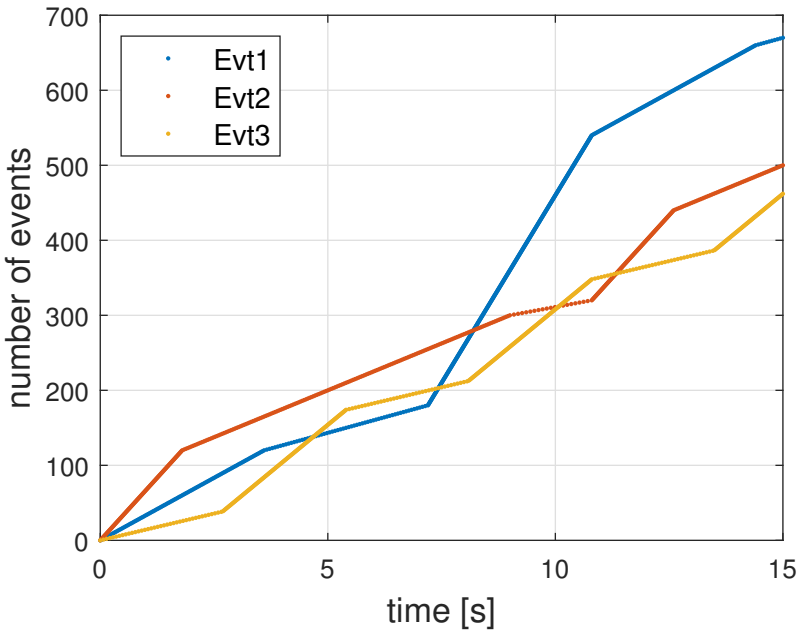
- 1000 powtórzeń dla konf. `(Evt1,30),(Evt2,30),(Evt3,40)`
- na wykresach odchyłki od wartości oczekiwanej liczby zdarzeń: symulacja vs rozkład normalny



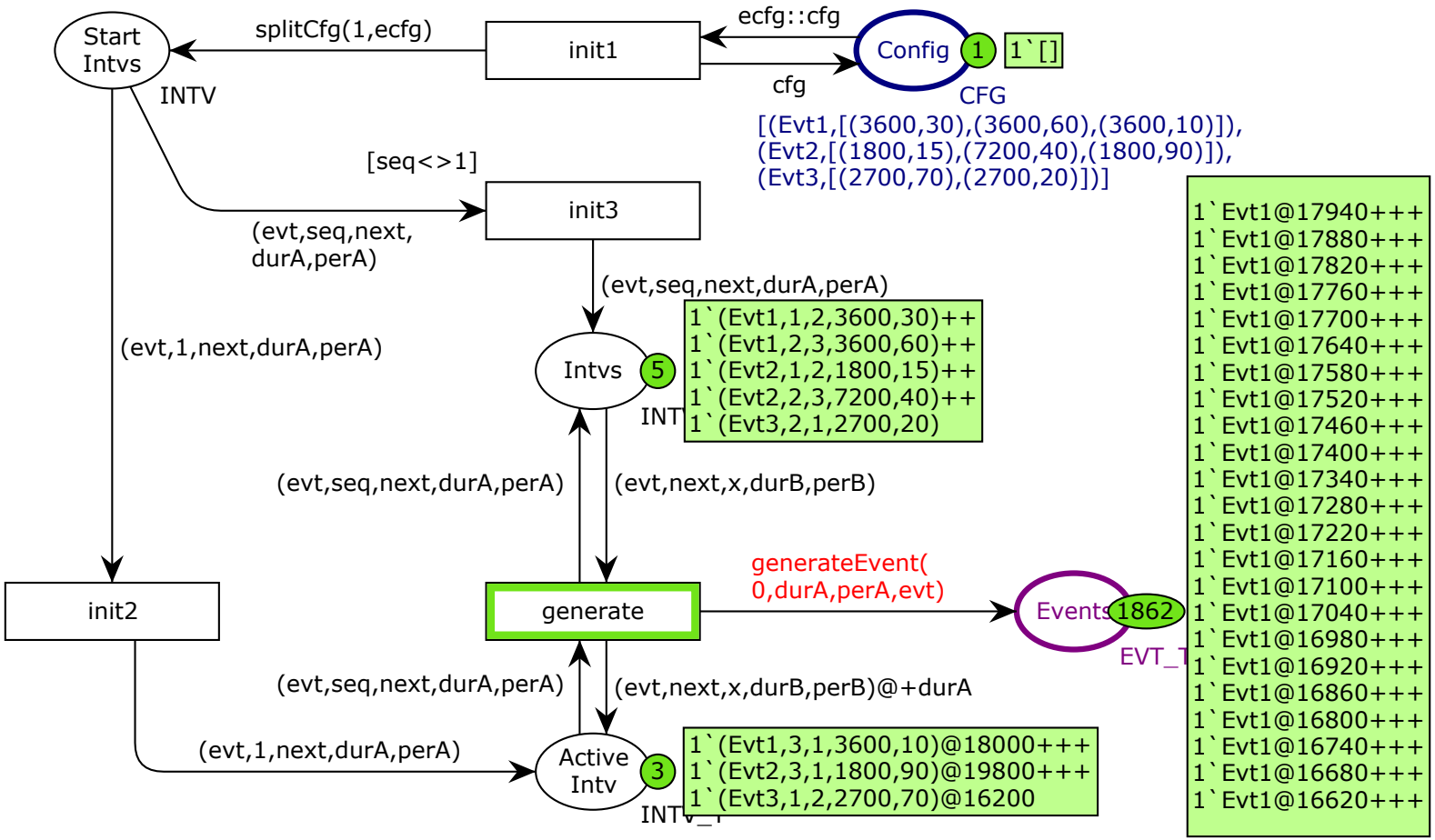
Generator czasowy deterministyczny

Dla każdego typu zdarzenia znakowanie miejsca konfiguracyjnego zawiera listę par (d, p) opisujących przedziały czasowe o stałej szybkości generowania. Wartość d reprezentuje długość przedziału, a p określa czas pomiędzy kolejnymi zdarzeniami. Po przetworzeniu wszystkich elementów listy generowanie powtarza się cyklicznie.





Generator czasowy deterministyczny




```
fun generateEvent(t,dur,per,evt) =
   if t<dur then (1`evt@+t)+++generateEvent(t+per,dur,per,evt) else empty
```

Generator czasowy deterministyczny

Dla znakowania konfiguracyjnego

$$[\ldots,(e_i,[(d_{i,1},p_{i,1}),(d_{i,2},p_{i,2}),\ldots,(d_{i,j},p_{i,j}),\ldots,(d_{i,m_i},p_{i,m_i})]),\ldots]$$

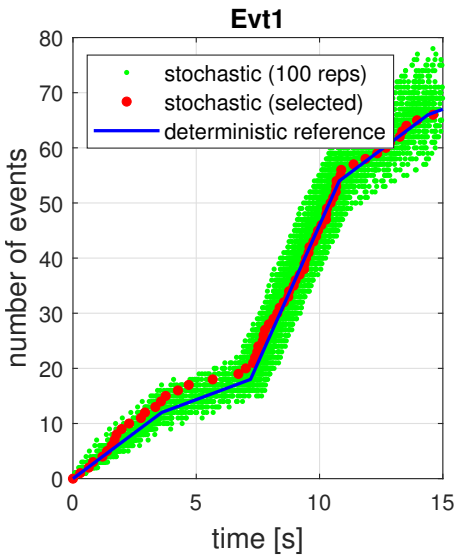
zdarzenie typu e_i jest generowane w chwilach należących do zbioru

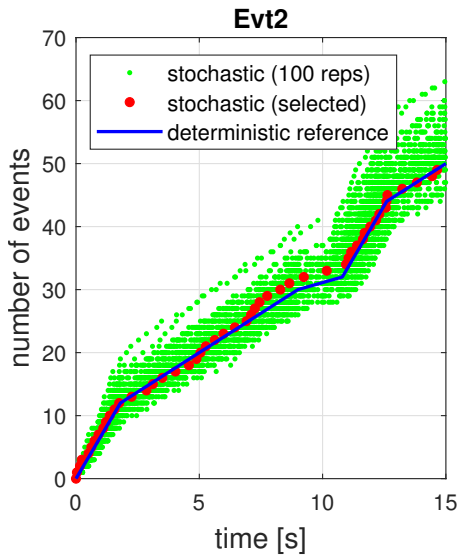
$$\mathbb{T}_i = \bigcup_{g=0}^{\infty} \bigcup_{h=0}^{\left\lfloor \frac{-1+d_{i,(g \bmod m_i)+1}}{p_{i,(g \bmod m_i)+1}} \right\rfloor} \left\{ hp_{i,(g \bmod m_i)+1} + \sum_{k=1}^{m_i} \left\lfloor \frac{m_i+g-k}{m_i} \right\rfloor d_{i,k} \right\}.$$

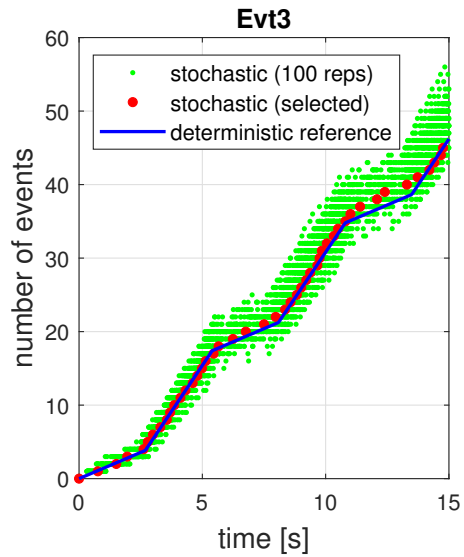
Generator czasowy stochastyczny

Model jest podobny do deterministycznego, główne różnice:

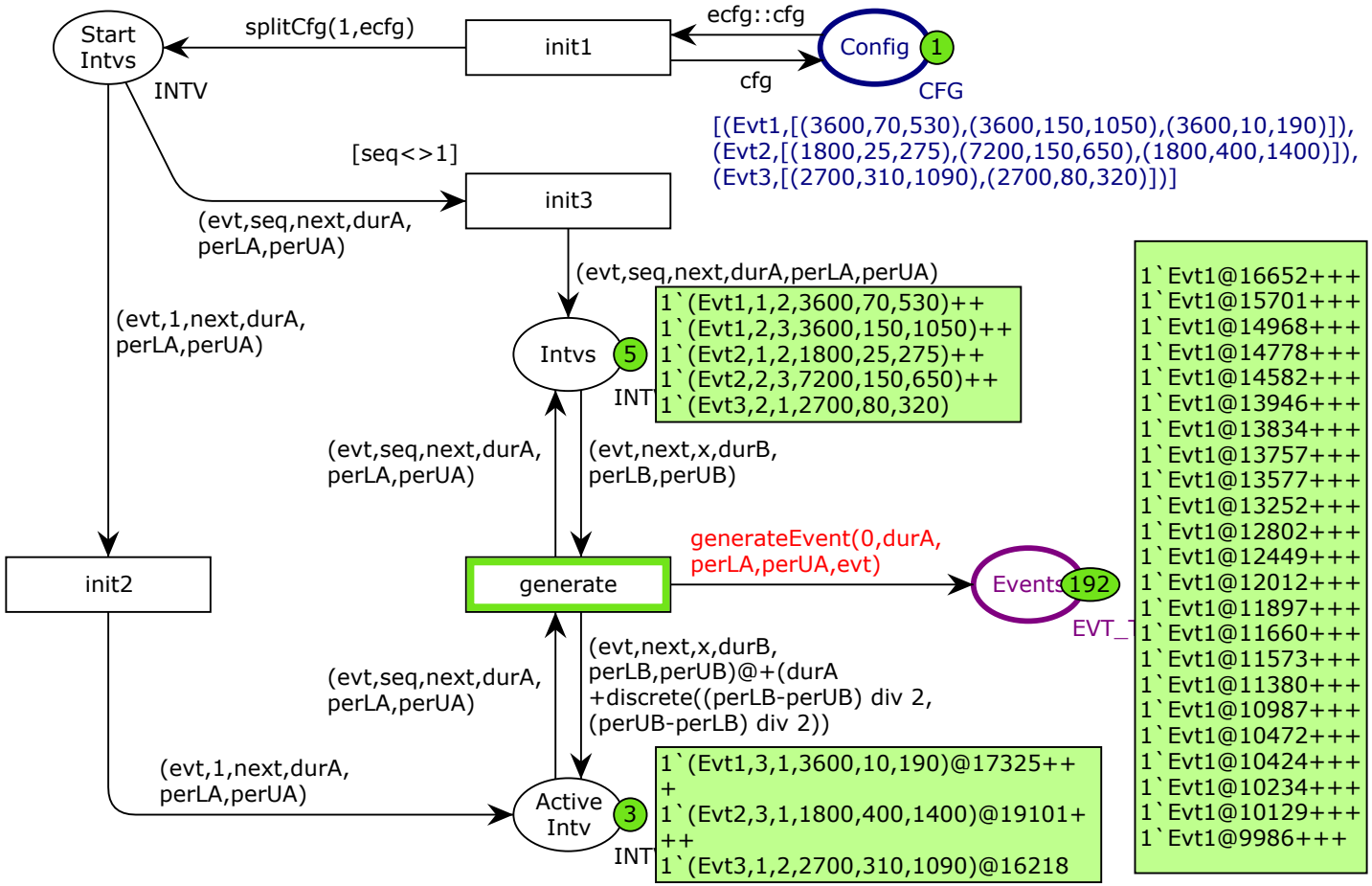
- Znakowanie konfiguracyjne nie określa stałego czasu pomiędzy zdarzeniami, ale ograniczenia dolne a i górne b tej wartości
- Funkcja `generateEvent` losuje odstępy pomiędzy kolejnymi zdarzeniami z rozkładem $\mathcal{U}_D(a, b)$







Generator czasowy stochastyczny




```
fun generateEvent(t,dur,perL,perU,evt) = if t<dur then (1`evt@+t)
  +++generateEvent(t+discrete(perL,perU),dur,perL,perU,evt) else empty
```

Obsługa zleceń w kolejce FIFO

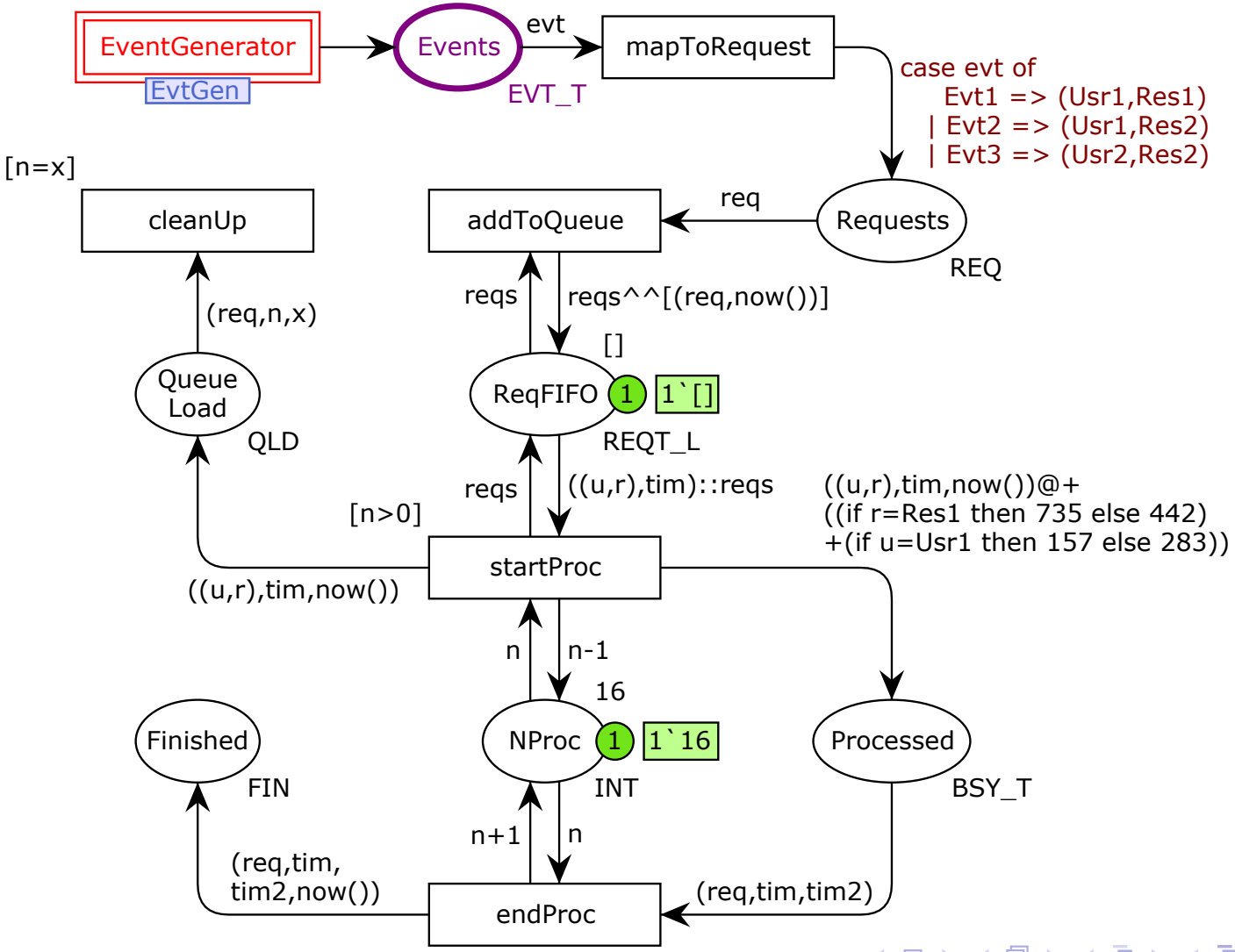
Budowa i parametry badanego systemu

- Generowane zdarzenia reprezentują żądania wysyłane do systemu (np. webowego) – gen. **czasowy stochastyczny**
- Każde żądanie jest reprezentowane przez parę (*Usr*,*Res*), która determinuje czas obsługi
- Żądania są obsługiwane przez *NProc* identycznych procesorów
- Przy braku wolnych procesorów nowe żądania są kolejkowane w trybie FIFO

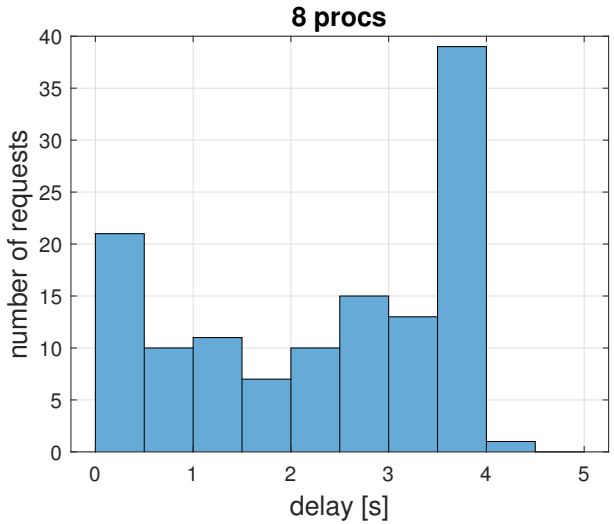
Analiza symulacyjna, np.

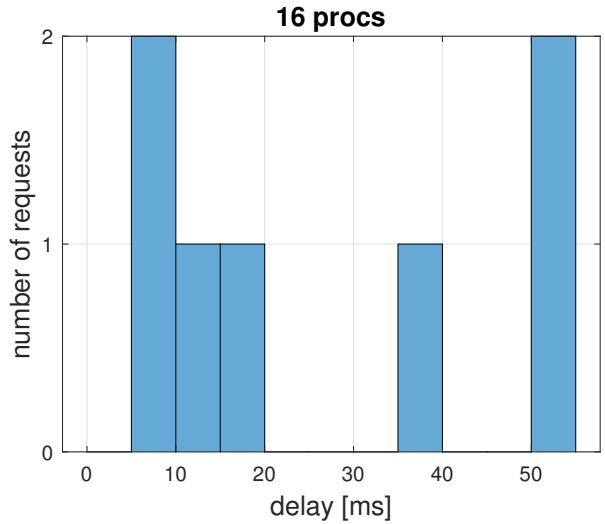
- Wpływ parametrów strumienia żądań na jakość obsługi
- Czas obsługi żądań zależnie od liczby procesorów

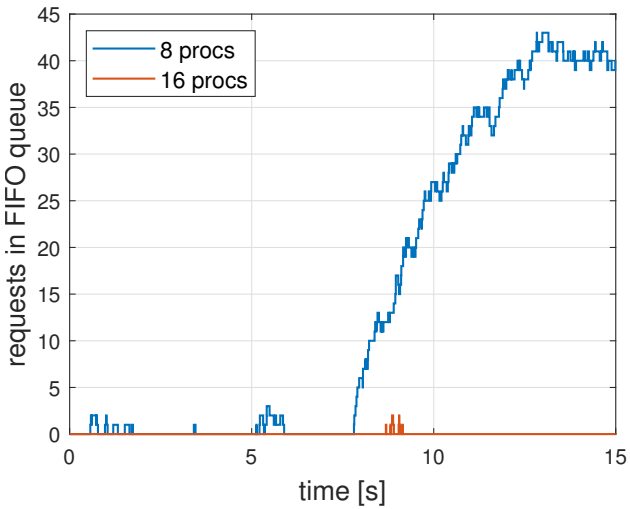
Obsługa zleceń w kolejce FIFO – model HTCPN

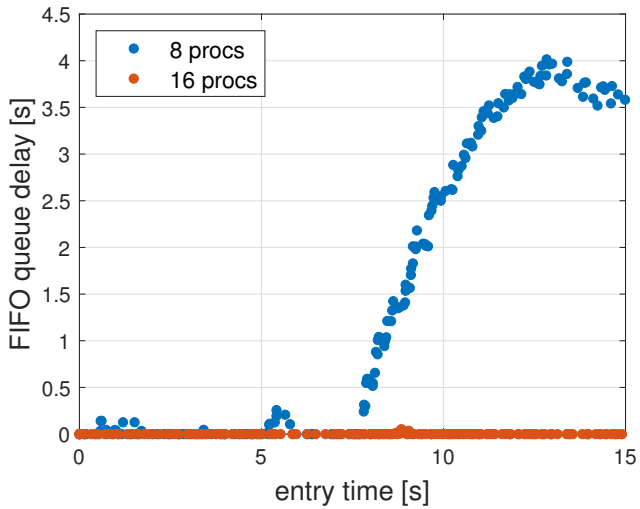


Obsługa zleceń w kolejce FIFO – wyniki symulacji









Wnioski

- Zaproponowano klasyfikację generatorów zdarzeń, która porządkuje ich warianty pod względem najważniejszych cech. Zaimplementowano przykłady wszystkich czterech wariantów.
- Zaprojektowano jednolity interfejs HTCPN dla generatorów, ułatwiający ich konfigurację i łączenie z innymi częściami modelowanych systemów.
- Znakowanie początkowe konfigurujące generatory może określać: typy generowanych zdarzeń, liczbę zdarzeń, parametry statystyczne emisji zdarzeń (gen. stochastyczne), parametry czasowe, np. odstęp pomiędzy emisjami (gen. czasowe).
- Jeden z opracowanych generatorów (czasowy stochastyczny) został zastosowany do symulacyjnego zbadania efektywności działania systemu obsługi zleceń dla różnej liczby procesorów.

Cytowania

Bożek, A.; Rak, T.; Rzońca, D. Timed Colored Petri Net-Based Event Generators for Web Systems Simulation. Appl. Sci. 2022, 12, 12385. https://doi.org/10.3390/app122312385

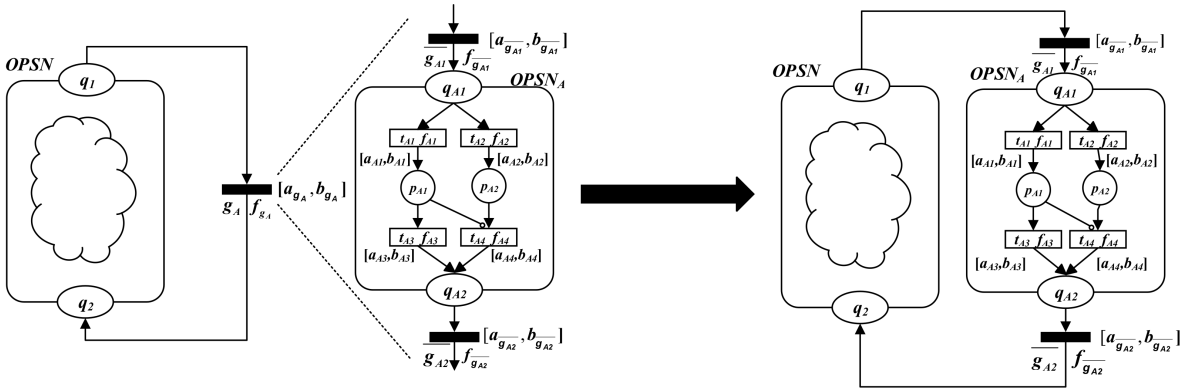
Artykuły

- Xia, C.; Qin, M.; Sun, Y.; Guo, M. Property Analysis of Gateway Refinement of Object-Oriented Petri Net with Inhibitor-Arcs-Based Representation for Embedded Systems. Electronics 2023, 12, 3977. https://doi.org/10.3390/electronics12183977
- Geng, R., Lin, Q., Liu, J., Huang, G., Guo, Q.: Modeling and analysis of transport behavior of custom furniture QC assembly line, Journal of Forestry Engineering, 2024, 9(4), pp. 185-192. https://doi.org/10.13360/j.issn.2096-1359.202311006
- Bortoloto, E.R.; Bizarria, F.C.P.; Bizarria, J.W.P. Petri Nets Applied in Purge Algorithm Analysis for a Rocket Engine Test with Liquid Propellant. Aerospace 2023, 10, 212. https://doi.org/10.3390/aerospace10030212

Monografia

- Romansky, R. P.; Hinov, N. L.: Deterministic and Stochastic Approaches in Computer Modeling and Simulation. IGI Global, 2023. https://doi.org/10.4018/978-1-6684-8947-5

Nowy rodzaj sieci – OOPIRES+



- Propozycja sieci OOPIRES+ (*object-oriented Petri net with inhibitor-arcs-based representation for embedded systems*)
- Rozszerzenie sieci PIRES+, bramka jest modelowana pewną podsiecią zachowując własności formalne (żywotność, ograniczoność, osiągalność, itd.)

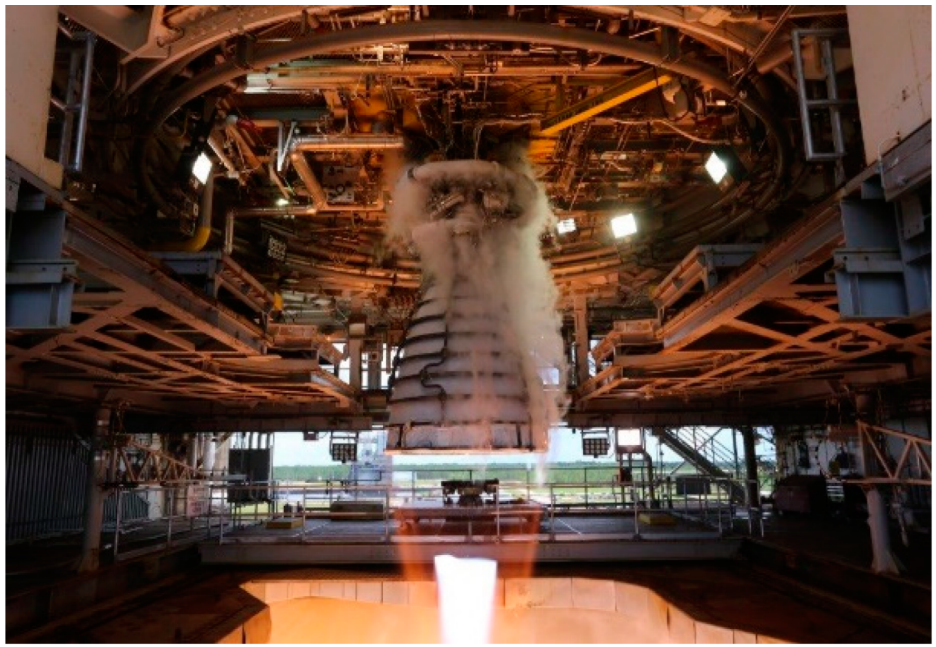
Xia, C.; Qin, M.; Sun, Y.; Guo, M. Property Analysis of Gateway Refinement of Object-Oriented Petri Net with Inhibitor-Arcs-Based Representation for Embedded Systems. Electronics 2023, 12, 3977. https://doi.org/10.3390/electronics12183977

Zastosowania – model linii montażowej

- Zastosowanie sieci Petriego do modelowania przemysłowej linii montażowej

Geng, R., Lin, Q., Liu, J., Huang, G., Guo, Q.: Modeling and analysis of transport behavior of custom furniture QC assembly line, Journal of Forestry Engineering, 2024, 9(4), pp. 185-192. https://doi.org/10.13360/j.issn.2096-1359.202311006

Zastosowania – *rocket science*



- Zastosowanie sieci Petriego do analizy algorytmu wykorzystywanego podczas testowania silników rakietowych

Bortoloto, E.R.; Bizarria, F.C.P.; Bizarria, J.W.P. Petri Nets Applied in Purge Algorithm Analysis for a Rocket Engine Test with Liquid Propellant. Aerospace 2023, 10, 212. https://doi.org/10.3390/aerospace10030212