

MODELE I METODY OCENY I PRZEWIDYWANIA NIEZAWODNOŚCI OPROGRAMOWANIA

Vitaliy Yakovyna



UNIwersYTET
WARMIŃSKO-MAZURSKI
W OLSZTYNIE



Jakość oprogramowania

- **Jakość systemu** to stopień, w jakim system spełnia deklarowane i dorozumiane potrzeby różnych interesariuszy, a tym samym zapewnia wartość
- **Model jakości** określa, które charakterystyki jakościowe będą brane pod uwagę podczas oceny właściwości oprogramowania

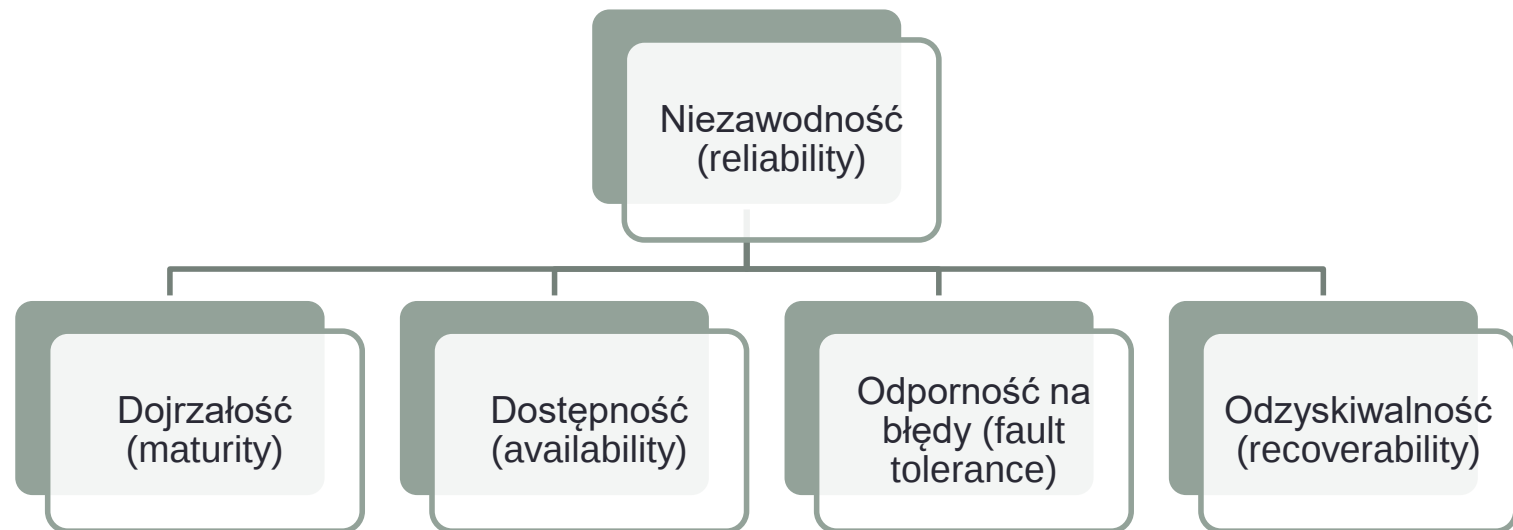
(ISO/IEC 25010:2011 – *System and software quality models*)

Charakterystyki jakości (ISO/IEC 25010:2011)



Niezawodność oprogramowania

- **Niezawodność** (ang. **reliability**) to stopień, w jakim system, produkt lub komponent wykonuje określone funkcje w określonych warunkach przez określony czas (ISO/IEC 25010:2011)



Definicje niezawodności oprogramowania

- **Niezawodność oprogramowania** (ang. **software reliability**) to prawdopodobieństwo bezawaryjnej pracy oprogramowania przez określony czas w określonym środowisku

(ANSI-729-1991 – *Standard Glossary of Software Engineering Terminology*)

- **Pewność** (ang. **dependability**) obejmuje dostępność (ang. *availability*), niezawodność (ang. **reliability**), odzyskiwalność (ang. *recoverability*), utrzymywalność (ang. *maintainability*) i wydajność obsługi technicznej (ang. *maintenance support performance*), a w niektórych przypadkach inne cechy, takie jak wytrzymałość (ang. *durability*), bezpieczeństwo (ang. *safety*) i ochrona (ang. *security*)

(ISO/IEC/IEEE 15026-1:2019 – *Systems and software engineering. Systems and software assurance. Part 1: Concepts and vocabulary*)

Problemy analizy niezawodności oprogramowania

Złożoność oprogramowania rośnie

- Modele oparte na **niejednorodnym procesie Poissona** $\Rightarrow$ zakłada się niezależność awarii oprogramowania
- Modele oceny niezawodności oprogramowania oparte na podejściu architektonicznym wykorzystują teorię **procesów Markowa** $\Rightarrow$ zakłada się niezależność wykonywania modułów systemu programowego

Agenda

- ✓ modele i metody oceny niezawodności oprogramowania oparte na niejednorodnym procesie Poissona
- ✓ Markowskie modele i metody oceny niezawodności systemów programowych
- ✓ modele przewidywania awarii oprogramowania na podstawie analizy szeregów czasowych

OCENA NIEZAWODNOŚCI OPROGRAMOWANIA NA PODSTAWIE WYNIKÓW TESTOWANIA

Modele niezawodności oparte na niejednorodnym procesie Poissona

- Funkcja niezawodności:

$$R(t) = P(T > t) = \int_t^{\infty} f(u) du$$

- Funkcja gęstości prawdopodobieństwa $f(u)$ zakłada czas awarii oprogramowania jako zmienną losową T
- Zakłada się, że funkcja niezawodności $R(t)$ ma rozkład wykładniczy z parametrem λ
- Jeśli λ jest funkcją wartości średniej $m(t)$, która zmienia się w czasie $\Rightarrow$ niejednorodny proces Poissona:

$$\Pr\{N(t) = n\} = \frac{\{m(t)\}^n}{n!} e^{-m(t)}, n = 0, 1, 2, \dots, t \geq 0$$

- $m(t)$ jest funkcją intensywności wskazującą liczbę awarii chwilowych w czasie t
- Wartość $m(t)$ jest obliczana na podstawie relacji między liczbą awarii $a(t)$ a współczynnikiem wykrywalności awarii $b(t)$

$$\frac{dm(t)}{dt} = b(t)[a(t) - m(t)]$$

- Z tego równania różniczkowego pochodzą poszczególne modele niezawodności

Uogólniony model niezawodności ze wskaźnikiem złożoności

$$\lambda(t) = \alpha \beta^{s+1} t^s \exp(-\beta t)$$

$$m(t) = \int_0^t \lambda(\tau) d\tau = \alpha \left[-\beta^s t^s e^{-\beta t} + s \Gamma_{\beta t}(s) \right]$$

$m(\infty) = \alpha s \Gamma(s)$ – prognoza całkowitej liczby awarii

$\lambda(t)|_{s=0} = \alpha \beta e^{-\beta t}$ – model Goel'a – Okumoto

$\lambda(t)|_{s=1} = \alpha \beta^2 t e^{-\beta t}$ – model w kształcie litery S

t – czas, $\lambda(t)$ – intensywność awarii, α – współczynnik charakteryzujący całkowitą liczbę awarii, β – współczynnik charakteryzujący wskaźnik wykrywalności awarii, s – wskaźnik złożoności oprogramowania

Badanie funkcji intensywności awarii dla modelu ze wskaźnikiem złożoności

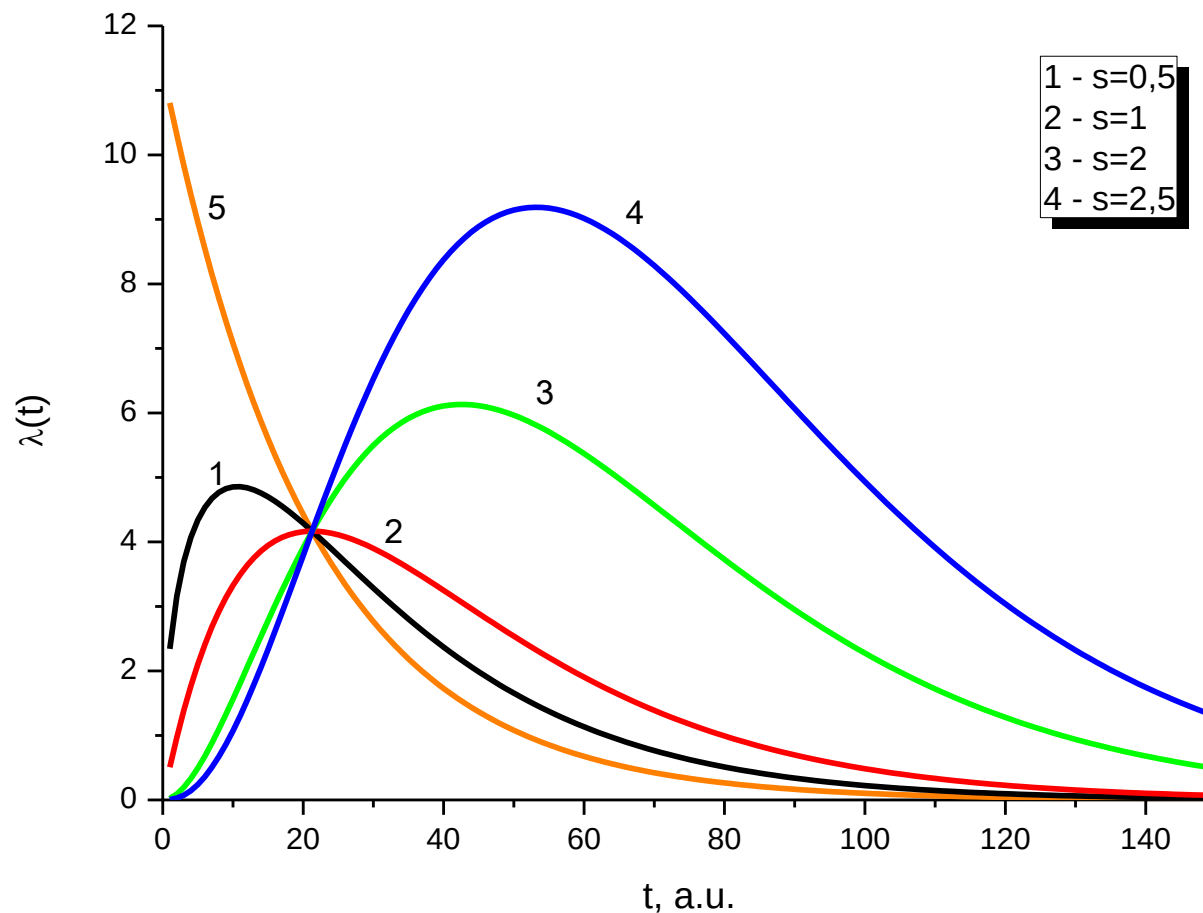
$$\frac{d\lambda(t)}{dt} = \alpha\beta^{s+1} \left(\frac{dt^s}{dt} e^{-\beta t} + \frac{de^{-\beta t}}{dt} t^s \right) = \lambda(t)(st^{-1} - \beta)$$

$$\alpha\beta^{s+1}t^s e^{-\beta t}(st^{-1} - \beta) = 0$$

$$t^s(st^{-1} - \beta) = 0$$

$$t_{max} = \frac{s}{\beta}$$

Funkcja intensywności awarii dla modelu ze wskaźnikiem złożoności



Wpływ parametrów modelu na funkcje intensywności awarii

$$\lambda_{max}(\alpha, \beta, s) = \lambda(t_{max}) = \alpha \beta^{s+1} \left(\frac{s}{\beta}\right)^s e^{-\beta \frac{s}{\beta}} = \alpha \beta s^s e^{-s}$$

$$\frac{\partial \lambda_{max}(\alpha, \beta, s)}{\partial \alpha} = \beta s^s e^{-s}$$

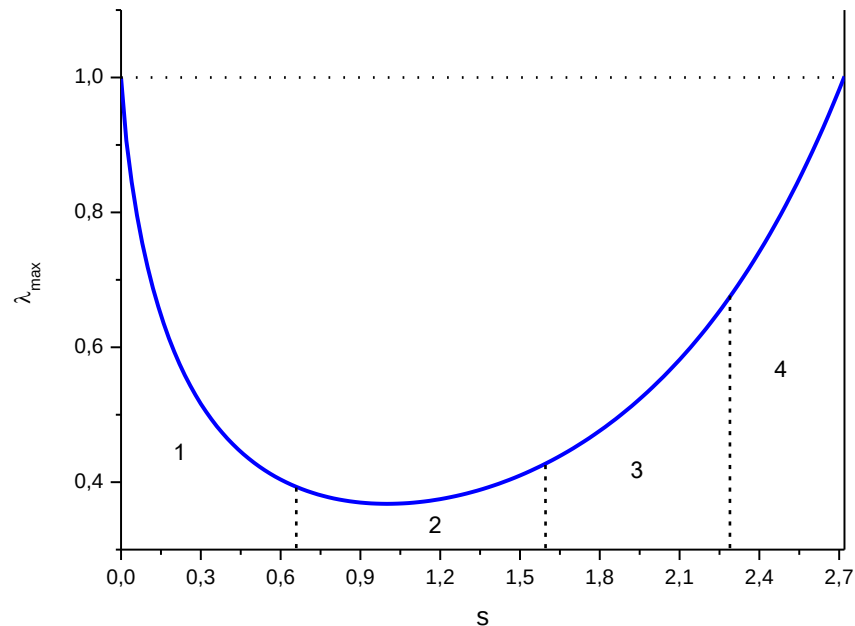
$$\frac{\partial \lambda_{max}(\alpha, \beta, s)}{\partial \beta} = \alpha s^s e^{-s}$$

$$\frac{\partial \lambda_{max}(\alpha, \beta, s)}{\partial s} = \alpha \beta s^s (\ln s + 1) e^{-s} - \alpha \beta s^s e^{-s} = \alpha \beta s^s e^{-s} \ln s$$

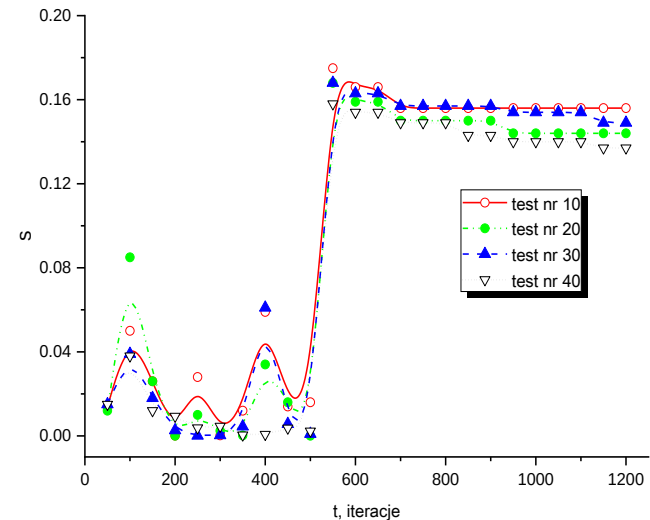
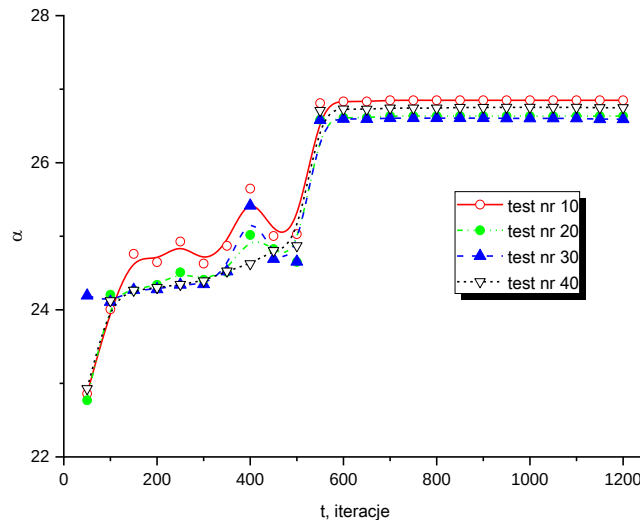
Wpływ wskaźnika złożoności na intensywność awarii oprogramowania

Przedziały wskaźnika złożoności:

1. $s \in [0; 0,66)$ oprogramowanie jest proste;
2. $s \in [0,66; 1,6)$ oprogramowanie o średniej złożoności;
3. $s \in [1,6; 2,28)$ oprogramowanie jest złożone;
4. $s \in [2,28; e]$ oprogramowanie jest bardzo złożone.

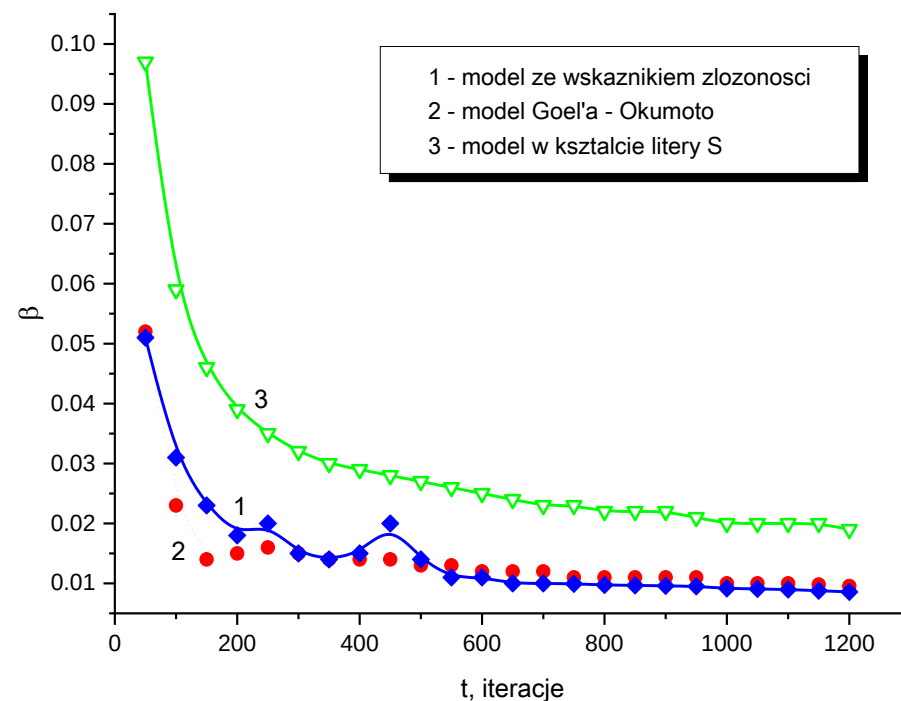
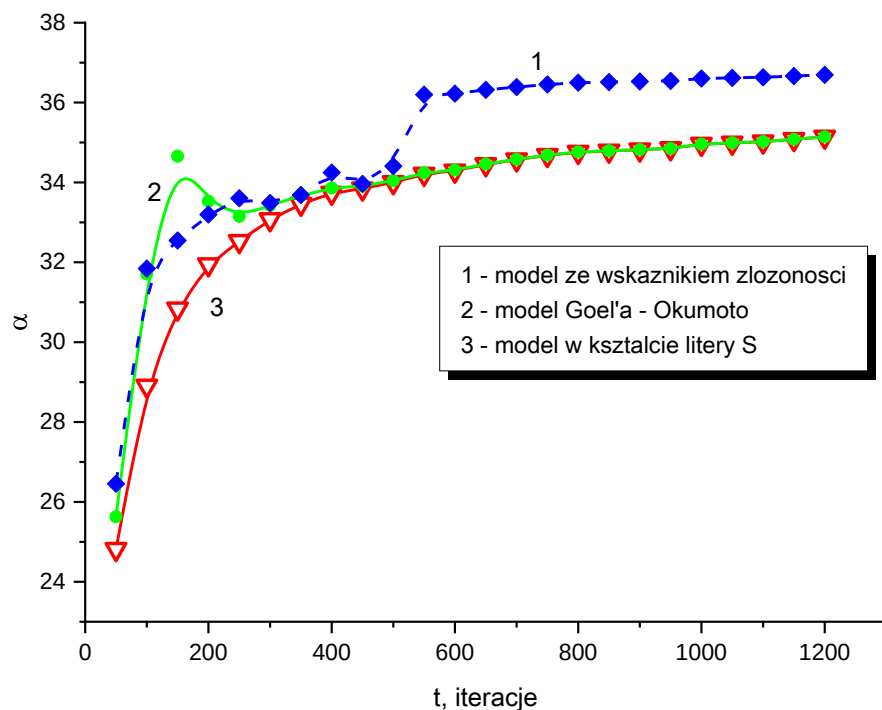


Jednorodność statystyczna estymacji parametrów modelu

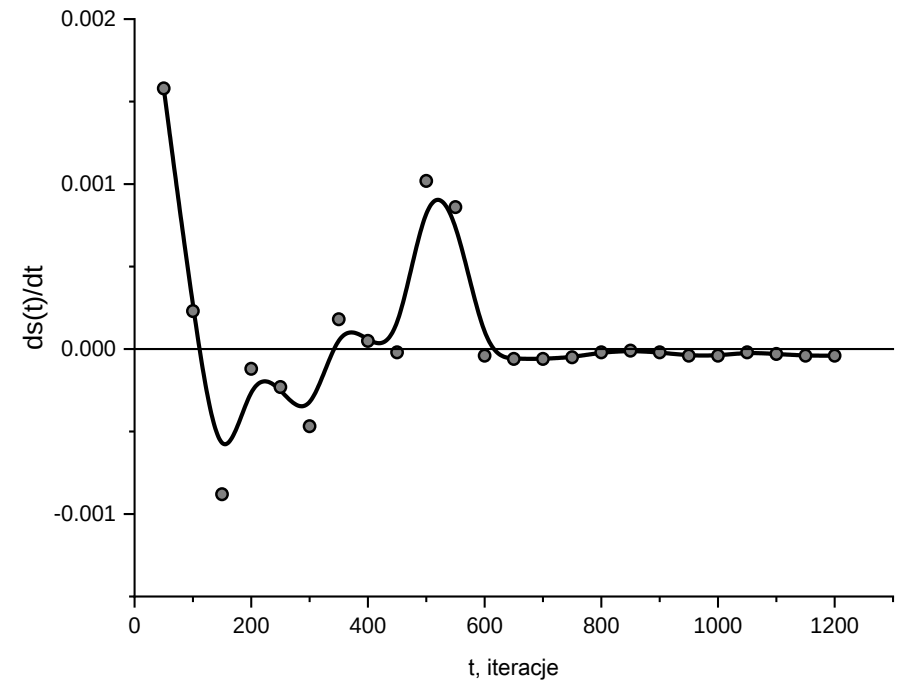
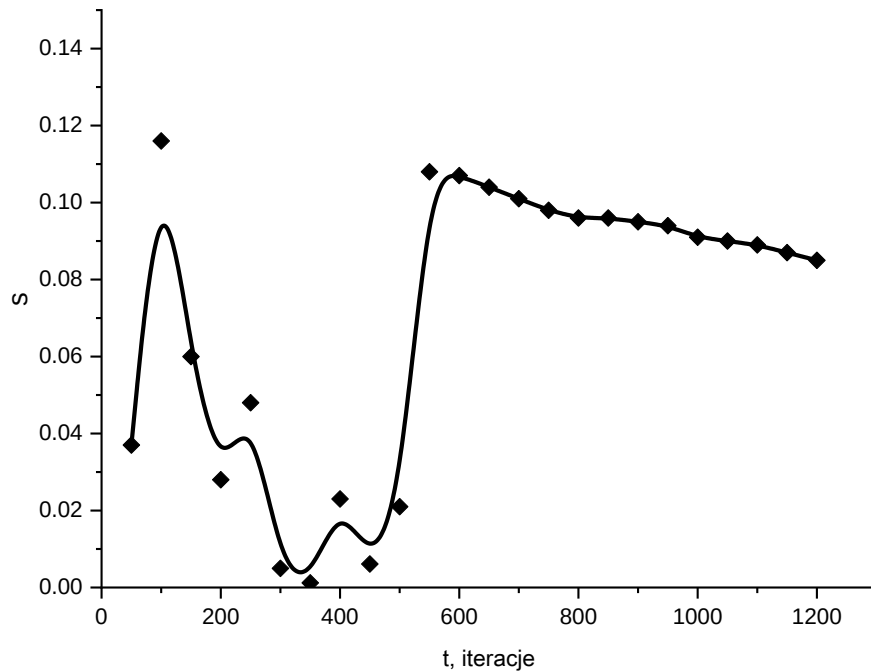


Zależność parametrów α i s modelu ze wskaźnikiem złożoności od czasu trwania procesu testowania dla różnych profili testowych

Kryterium wystarczalności testowania



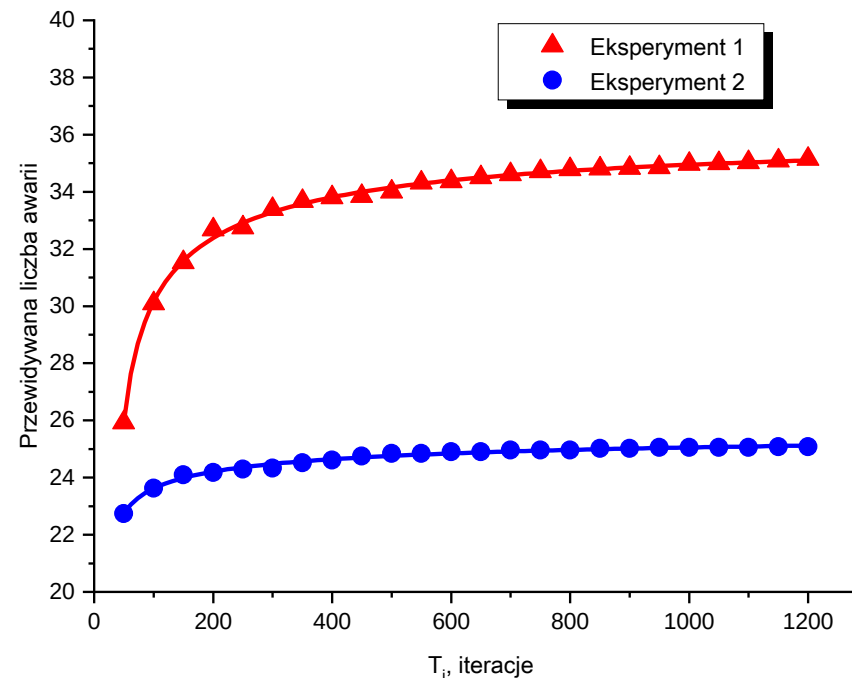
Kryterium wystarczalności testowania



$$\frac{ds(t)}{dt} \xrightarrow{t \rightarrow \infty} 0$$

Przewidywanie liczby awarii. Model regresyjny

$$m_{\infty} = A \left[1 - \exp \left(- (k(T_i - T_c))^d \right) \right]$$



Wyniki przewidywania liczby awarii oprogramowania za pomocą regresji

Liczba iteracji	Prognoza wg $m_{\infty} = \alpha s \Gamma(s)$	Prognoza wg modelu regresji	Współczynnik determinacji (R^2)	Poprawa prognozy, %
300	33.4	34.6	0.992	3.7
400	33.8	35.0	0.996	3.5
500	34.0	34.6	0.997	1.7
600	34.4	35.1	0.997	2.1
700	34.6	35.4	0.997	2.4
800	34.8	35.7	0.997	2.7
900	34.8	35.7	0.997	2.6
1000	35.0	35.7	0.997	2.2
1100	35.0	35.8	0.998	2.1
1200	35.1	35.8	0.998	2.0

Weryfikacja modelu ze wskaźnikiem złożoności

Model	α	β	s	Prognoza liczby awarii
Goel'a – Okumoto	25.08	0.016	–	25.08
Model w kształcie litery S	25.08	0.028	–	25.08
Model ze wskaźnikiem złożoności	26.75	0.013	0.137	25.08

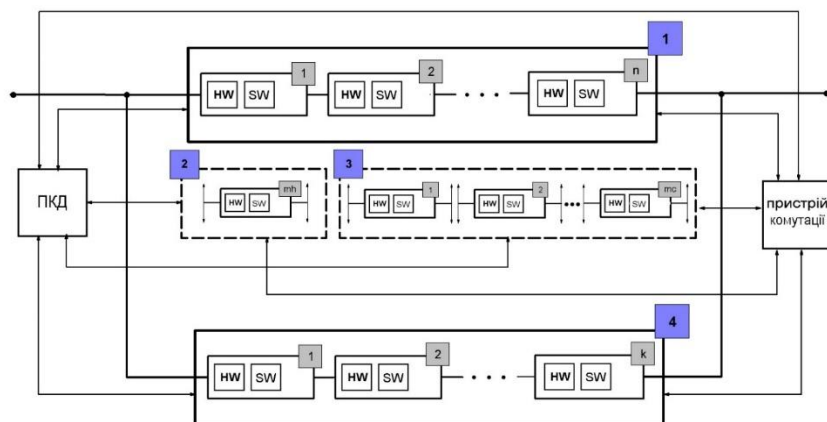
Model	Współczynnik determinacji	Błąd średniokwadratowy
Goel'a – Okumoto	0.974	0.137
Model w kształcie litery S	0.995	0.105
Model ze wskaźnikiem złożoności	0.989	0.080

Ocena wskaźników niezawodności systemu skomputeryzowanego

- Zbadano wpływ modelu niezawodności oprogramowania na estymację funkcji gotowości systemu z redundancją
- Intensywność awarii oprogramowania wyliczona na podstawie różnych modeli niezawodności:

λ, h^{-1}	Model Goel'a – Okumoto	Model w kształcie litery S	Model ze wskaźnikiem złożoności
λ_{sw11}	3.516×10^{-6}	1.744×10^{-9}	1.221×10^{-3}
λ_{sw12}	2.143×10^{-9}	3.640×10^{-14}	7.286×10^{-8}

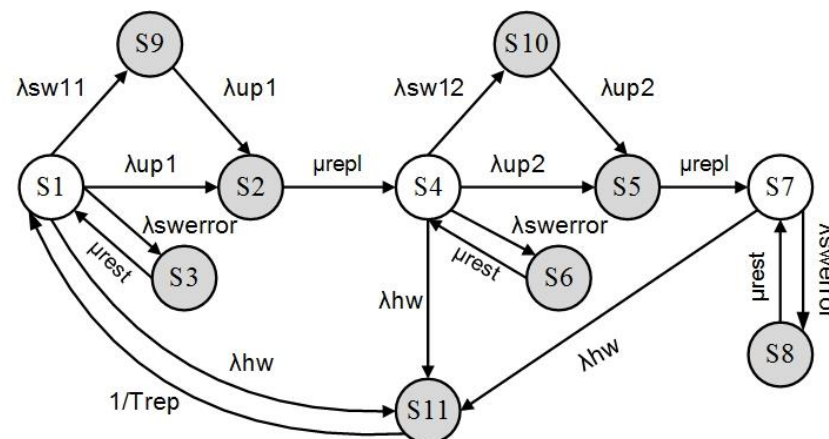
Blokowy schemat niezawodności badanego systemu



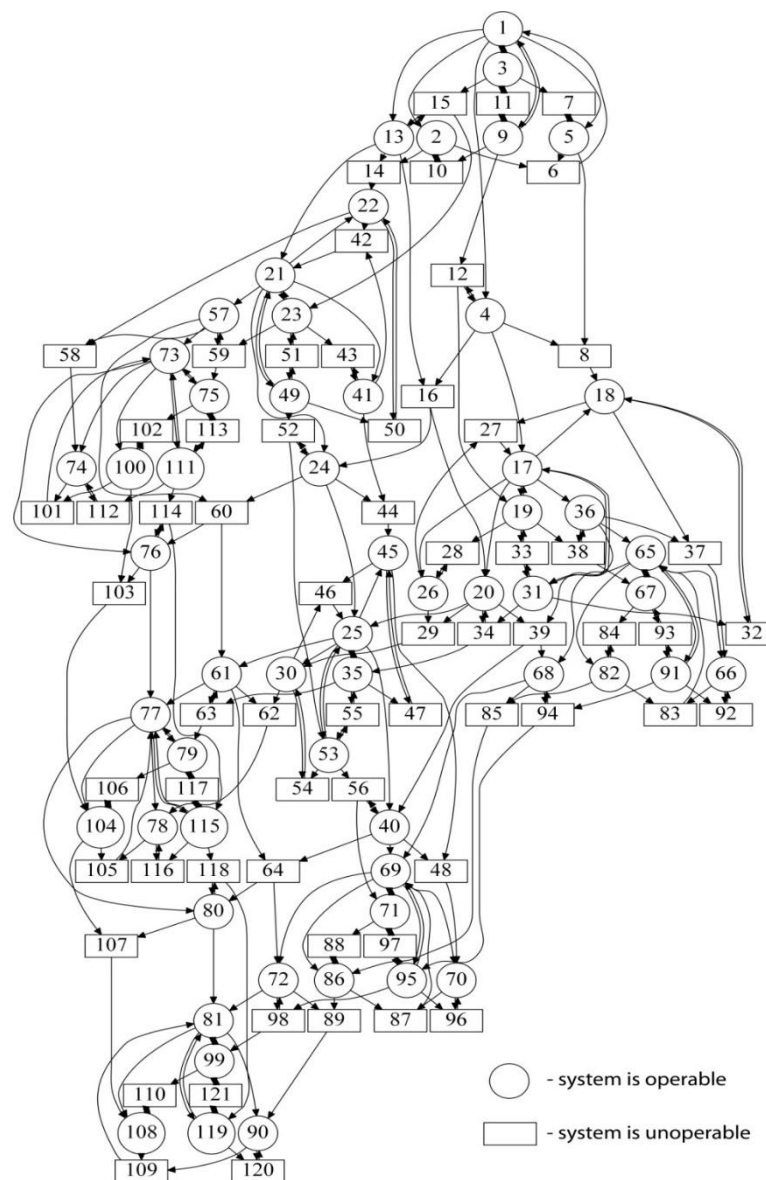
- Odporny na błędy system składa się z systemu głównego, składającego się z n procesorów, oraz systemu zapasowego składającego się z k procesorów
- Dla obu systemów występuje wspólna redundancja ślizgowa procesorów, przy czym jeden procesor zapasowy znajduje się w gorącej rezerwie, a reszta - w zimnej

Model zachowania badanego systemu

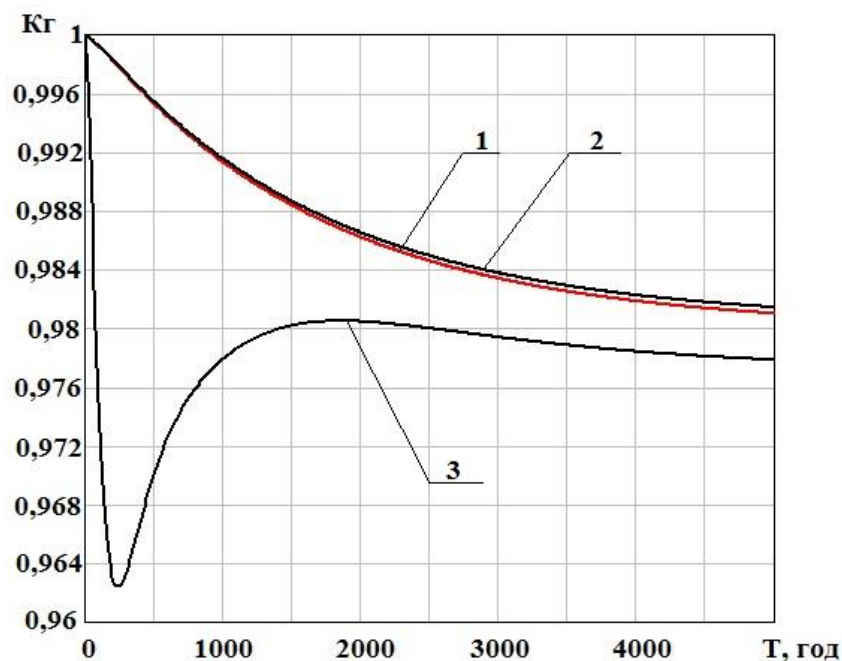
- Przy ocenie wskaźników niezawodności systemu uwzględniono:
- awarie oprogramowania;
- aktualizacje oprogramowania;
- automatyczne ponowne uruchamianie oprogramowania;
- awarie sprzętu



Markowska model niezawodności systemu



Wpływ modeli niezawodności oprogramowania na ocenę funkcji gotowości systemu



Model ze wskaźnikiem złożoności:

- współczynnik gotowości 0.977

Tradycyjne modele:

- współczynnik gotowości 0.981

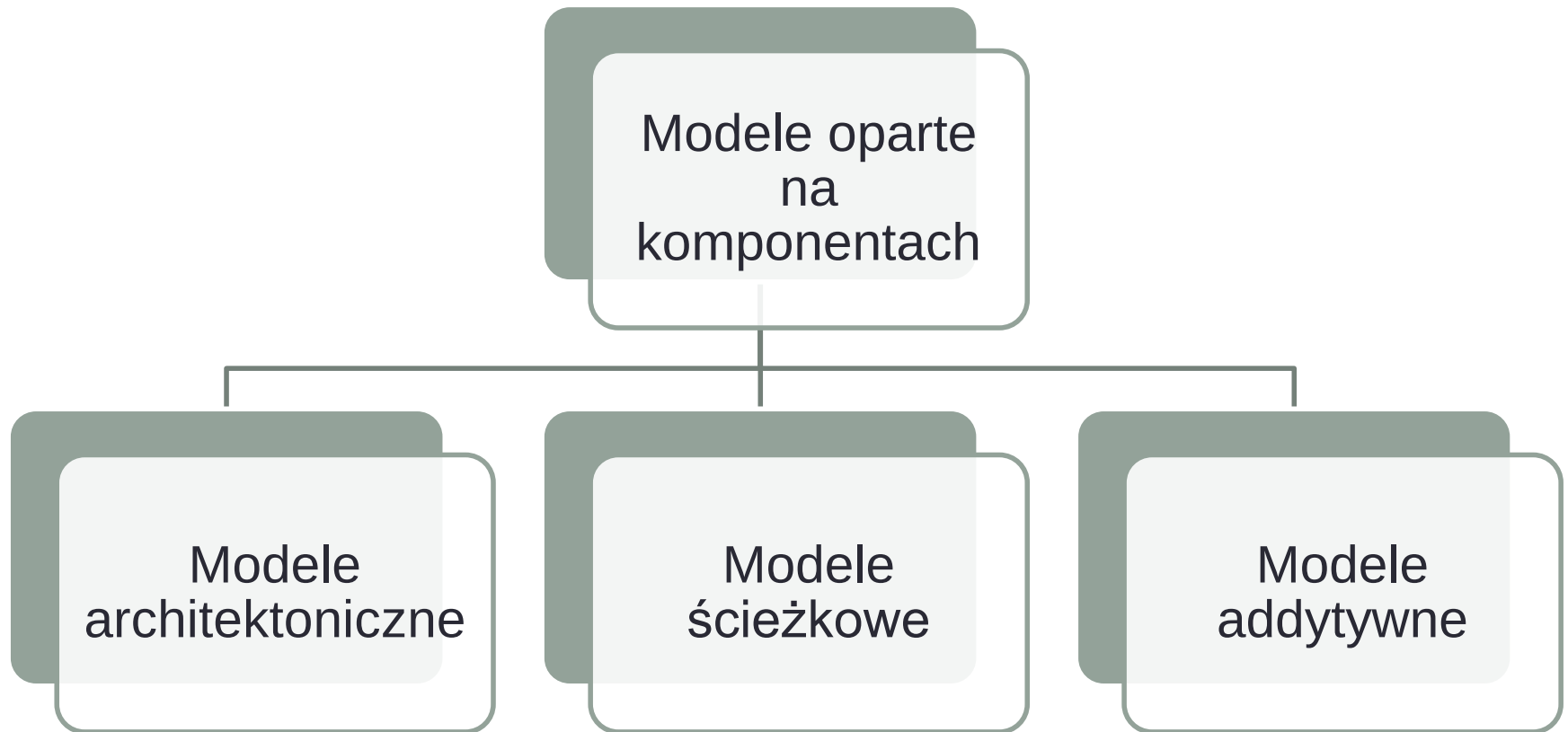
Mimo tego model za wskaźnikiem złożoności była stosowana dla oceny niezawodności systemu sterowania elektrownią jądrową* oraz podsystemu transmisji danych IoT**

* Volochiy, B., Yakovyna, V., Mulyak, O., Kharchenko, V.: Availability model of critical nuclear power plant instrumentation and control system with non-exponential software update distribution. In: N. Bassiliades, V. Ermolayev et al. (eds.), Communications in Computer and Information Science, CCIS 826 (2018), Springer, pp. 3–20.

** Volochiy, B., Yakovyna, V., Mulyak, O.: Analytical Model for Availability Assessment of IoT Service Data Transmission Subsystem. In: N. Shakhovska and V. Stepashko (eds.), Advances in Intelligent Systems and Computing II, AISC 689 (2018), Springer, pp. 588–600.

OCENA NIEZAWODNOŚCI SYSTEMÓW PROGRAMOWYCH Z UWZGLĘDNIENIEM ICH ARCHITEKTURY I ZŁOŻONOŚCI

Modele niezawodności oprogramowania oparte na komponentach



MODEL NIEZAWODNOŚCI OPARTA NA PROCESIE MARKOWA WYŻSZEGO RZĘDU Z CZASEM DYSKRETNYM

Założenia modelu

- system programowy składa się ze skończonej liczby modułów, które pod względem diagramu niezawodności są połączone szeregowo
- przekazywanie kontroli między modułami programu odbywa się w tych samych dyskretnych odstępach czasu
- każdy moduł systemu programowego może być testowany niezależnie i jego intensywność awarii jest $\lambda_i(t)$
- prawdopodobieństwa przekazywania kontroli p_{ij} między modułami i i j zależy od listy poprzednich modułów, która kończy się modułem bieżącym i
- niezawodność i -tego modułu zależy tylko od całkowitego czasu wykonania tego modułu i jego intensywności awarii, ale nie zależy od konkretnych momentów czasu wykonania tego modułu

Struktura modelu

- $\{C_i\}$ to graf skierowany, który pokazuje strukturę systemu programowego; wierzchołki grafu odpowiadają modułom programu; a krawędzie odzwierciedlają przekazywanie kontroli między modułami ($i = \overline{1, N}$; N to całkowita liczba modułów)
- $\mathbf{P} = \{p_{ij...kl}\}$ to macierz prawdopodobieństw przejścia w moduł l w zależności od wykonania poprzednich K modułów
- $\lambda_i(t)$ to intensywność awarii odpowiedniego modułu

Opis modelu

- Funkcja niezawodności systemu programowego:

$$R(t) = \prod_{i=1}^N R_i(t)$$

- Funkcja niezawodności każdego modułu:

$$R_l(t) = \exp \left(- \int_0^{\sum_{ij \dots k} (V_{ij \dots kl} \cdot t_{ij \dots kl})} \lambda_l(\tau) d\tau \right)$$

- Oczekiwana liczba wykonań modułu l w zależności od wykonania poprzednich K modułów $V_{ij \dots kl}$ otrzymuje się z układu liniowych równań algebraicznych:

$$V_{j \dots kl} = q_{j \dots kl} + \sum_{i=1}^{N-1} V_{ij \dots k} p_{j \dots kl}$$

MODEL NIEZAWODNOŚCI OPARTA NA PROCESIE MARKOWA WYŻSZEGO RZĘDU Z CZASEM CIĄGŁYM

Założenia modelu

- system programowy składa się ze skończonej liczby modułów, które pod względem diagramu niezawodności są połączone szeregowo
- przekazywanie kontroli między modułami programu następuje w losowych momentach
- każdy moduł systemu programowego może być testowany niezależnie i jego intensywność awarii jest $\lambda_i(t)$
- intensywność przejść a_{ij} między modułami i i j zależy od historii przejść pomiędzy poprzednimi modułami, która kończy się modulem bieżącym i
- niezawodność i -tego modułu może zależeć od konkretnych momentów czasu wykonania tego modułu (w razie $\lambda_i = f(t)$)
- w dowolnym momencie t awaria systemu programowego może być spowodowana jedynie awarią aktualnie uruchomionego modułu

Struktura modelu

- $\{C_i\}$ to graf skierowany, który pokazuje strukturę systemu programowego; wierzchołki grafu odpowiadają modułom programu; a krawędzie odzwierciedlają przekazywanie kontroli między modułami ($i = \overline{1, N}$; N to całkowita liczba modułów)
- $A = \{a_{ij}(t)\}$ to macierz intensywności przejść między stanami systemu, które odpowiadają wierzchołkom grafu $\{C_i\}$;
- $P = \{p_i(t)\}$ to wektor prawdopodobieństwa, że system znajduje się w stanie C_i w czasie t
- $\lambda_i(t)$ to intensywność awarii odpowiedniego modułu

Opis modelu

- Intensywność awarii systemu programowego:

$$\lambda(t) = \sum_{i=1}^N p_i(t) \lambda_i(t)$$

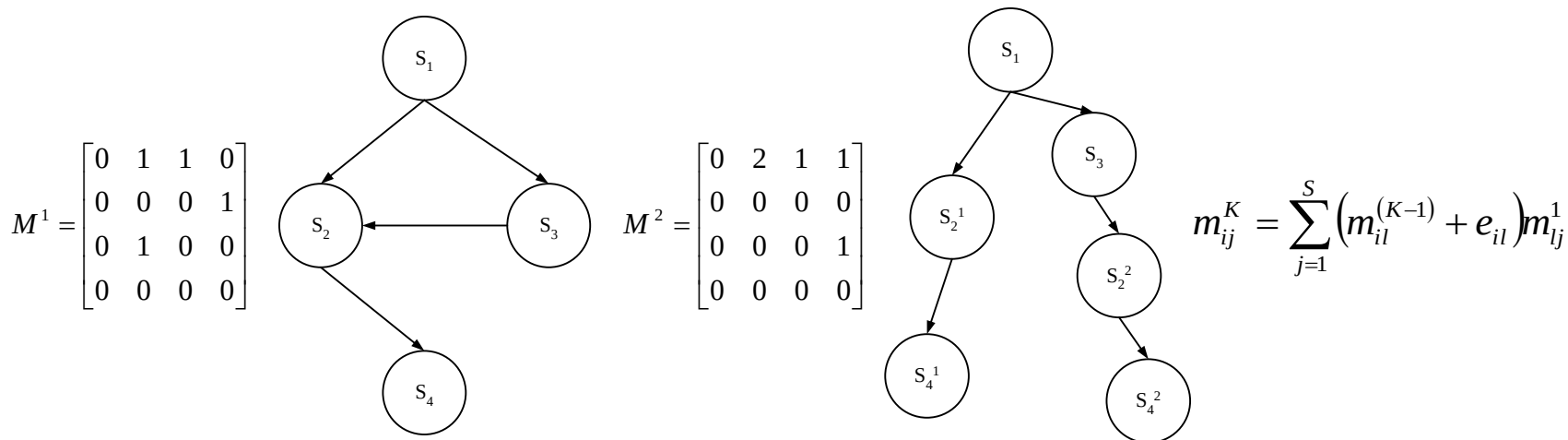
- Prawdopodobieństwa $p_i(t)$ są otrzymywane z układu równań Kołmogorowa – Chapmana:

$$\frac{dp_i(t)}{dt} = - \sum_{j \in S} a_{ij}(t) p_i(t) + \sum_{j \in S} a_{ji}(t) p_j(t), i \in S$$

- Proces wyższego rzędu polega na zależności intensywności przejścia między stanami $a_{ij}(t)$ od ścieżki wejścia w stan i
- Im wyższy rząd procesu, tym dłuższy łańcuch przejść wpływa na tę intensywność

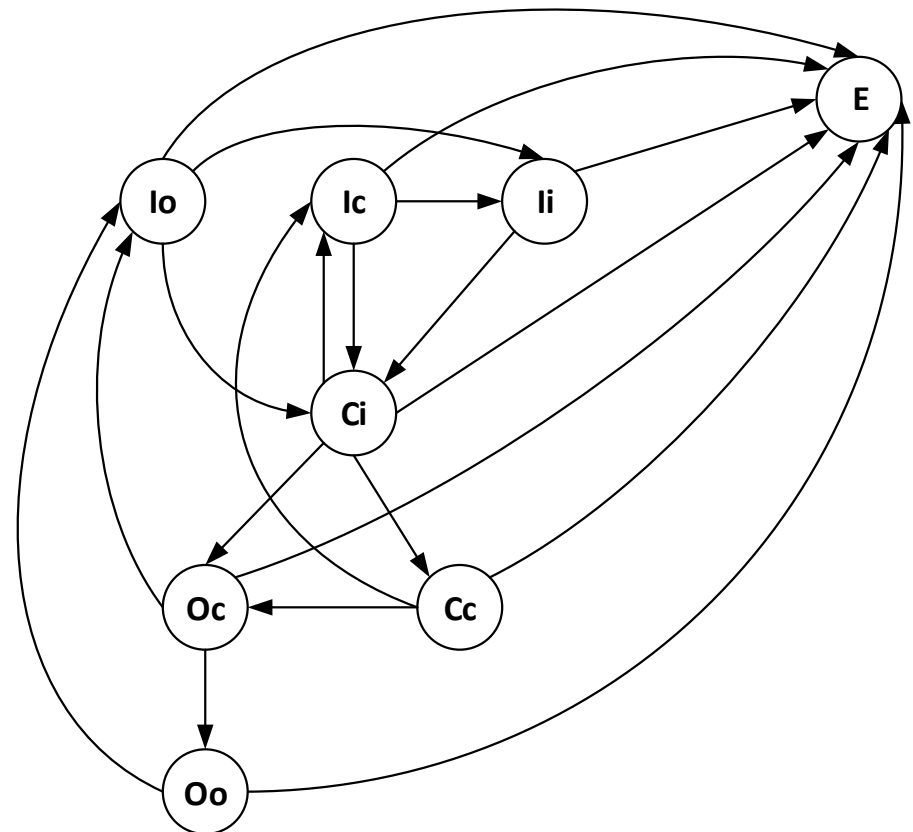
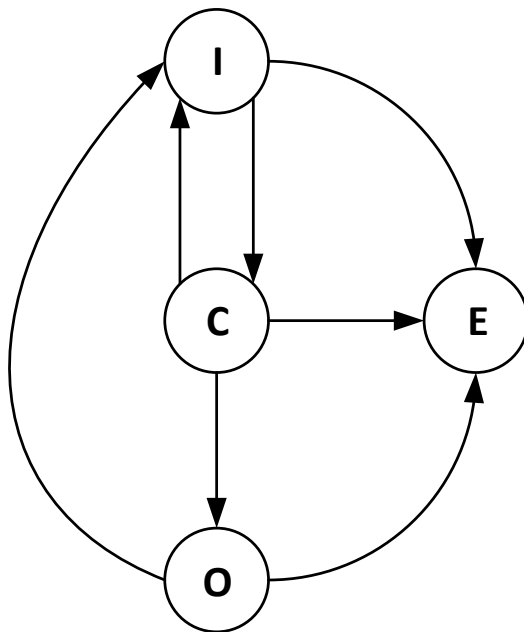
Równoważny proces pierwszego rzędu

- Elementy macierzy m_{ij}^K oznaczają liczbę możliwych ścieżek długości K od stanu C_i do C_j
- Suma elementów w j -tej kolumnie macierzy M^K jest równa liczbie ścieżek długości K wiodących do stanu C_j (liczba stanów wirtualnych)
- Suma elementów w i -tym wierszu macierzy jest równa liczbie ścieżek wychodzących ze stanu C_i



WERYFIKACJA MODELI

Struktura oprogramowania testowego

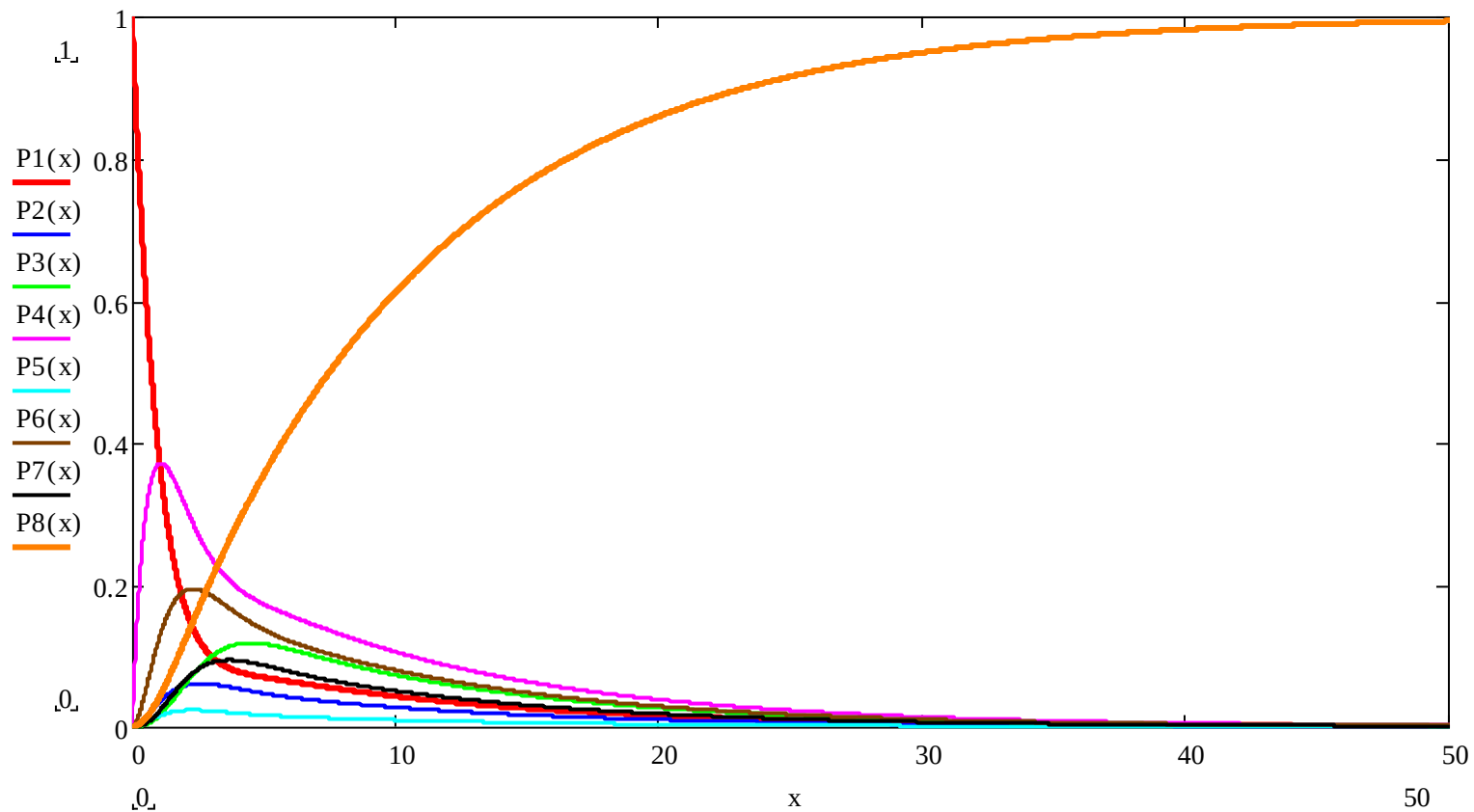


Charakterystyka oprogramowania testowego

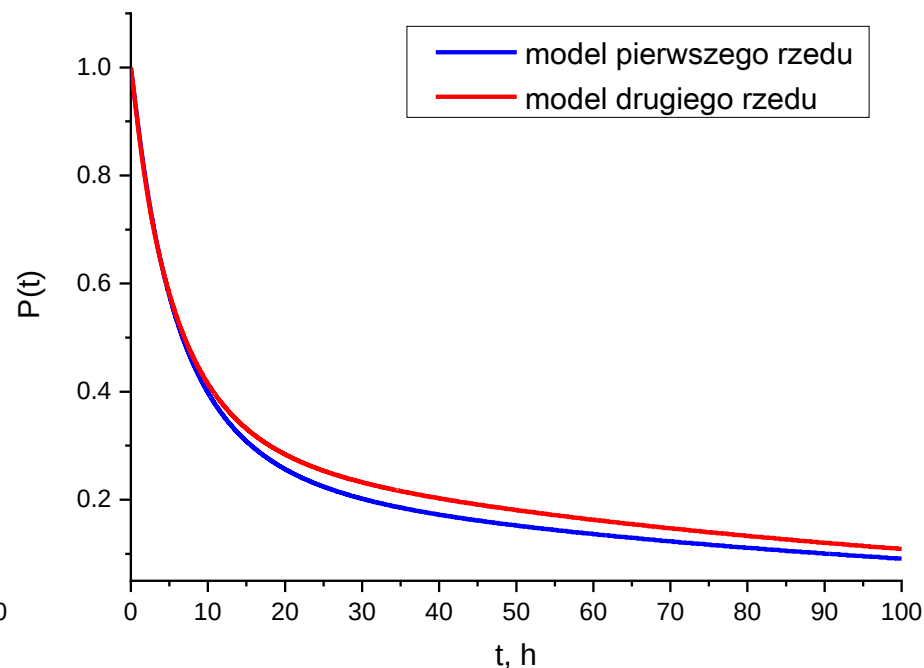
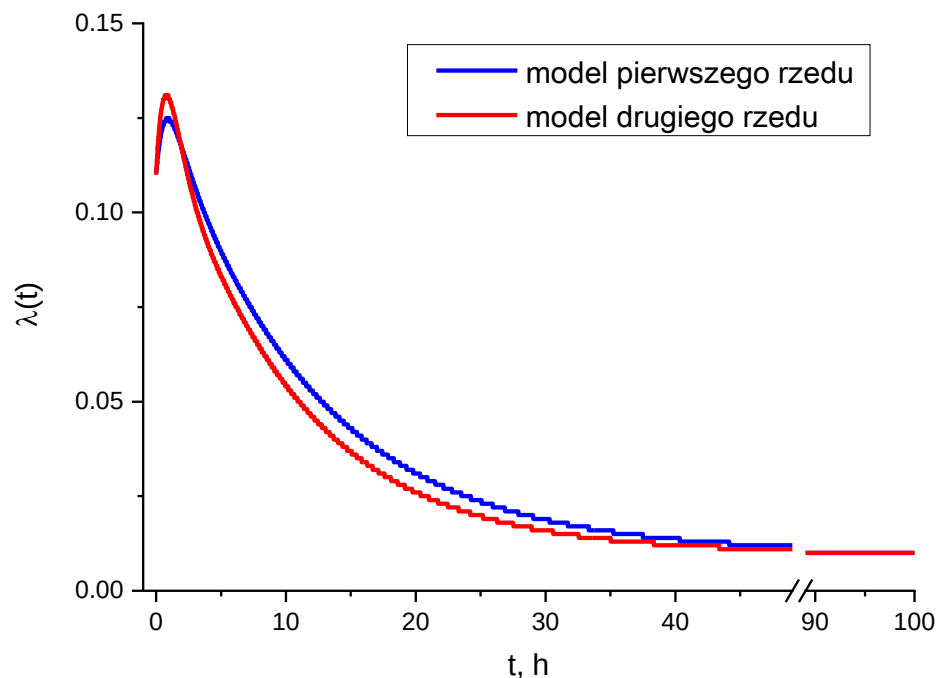
Moduł	Intensywność awarii, λ_i
Input	0.11
Calculation	0.18
Output	0.09
Exit	0.01

Moduł	Input	Calculation	Output	Exit
Input	0.2987	0.66233	0	0.03897
Calculation	0.2666	0.05	0.6333	0.0501
Output	0.5	0	0.3148	0.1852
Exit	0	0	0	1

Rząd procesu	Wartość kryterium AIC
1	223,7
2	203,5
3	352,7
4	1082,8

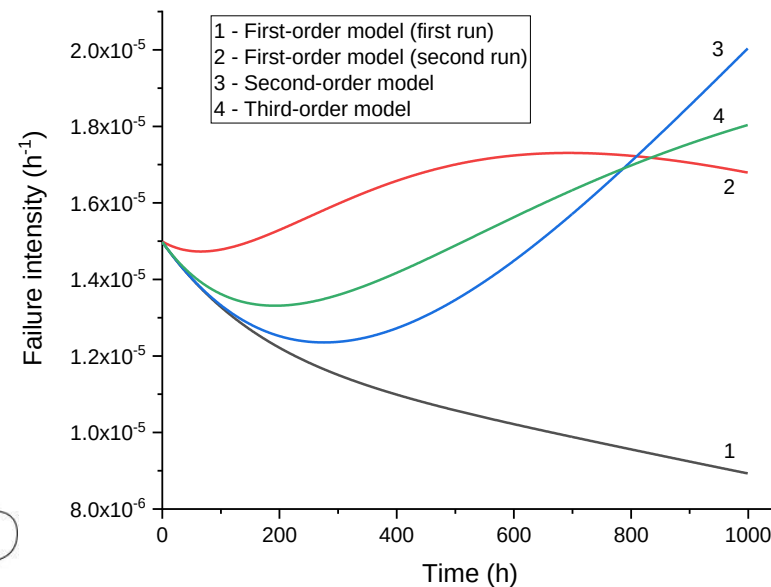
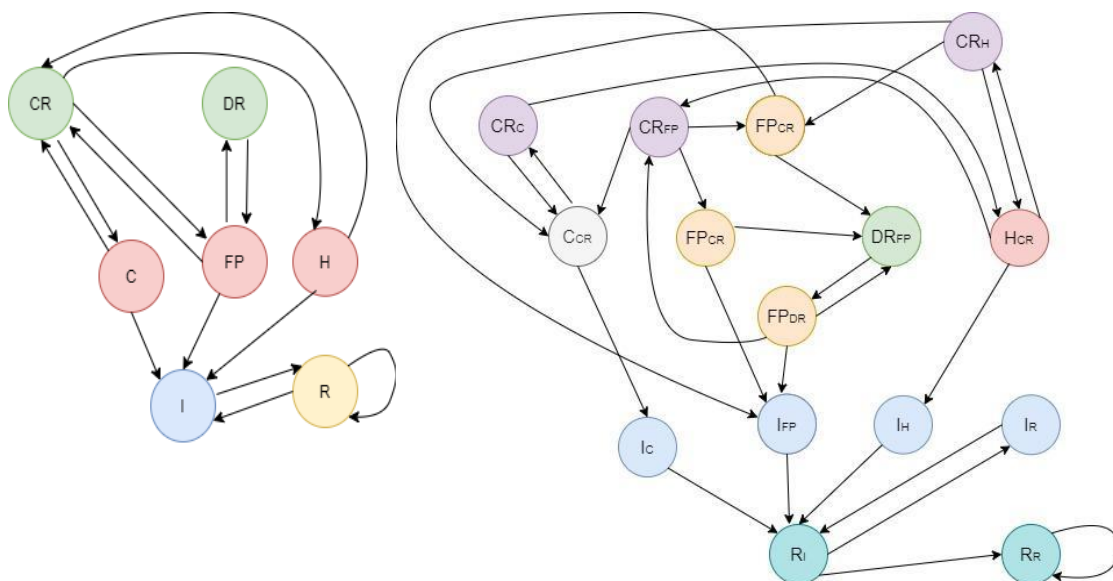


Wskaźniki niezawodności oprogramowania testowego



Średni czas między awariami: $T_1^{(1)} = 20.9 \text{ h}$; $T_1^{(2)} = 23.3 \text{ h}$

Zastosowanie równoważnego procesu pierwszego rzędu



Algorytm konstruowania równoważnego procesu pierwszego rzędu* został wykorzystany do opracowania modelu niezawodności oprogramowania lotniczego nanosatelity CubeSat**

*Yakovyna, V., Symets, I.: A method of high-order Markov chain representation through an equivalent first-order chain for software reliability assessment. Computer Systems and Information Technologies 3(5), 66–73 (2021)

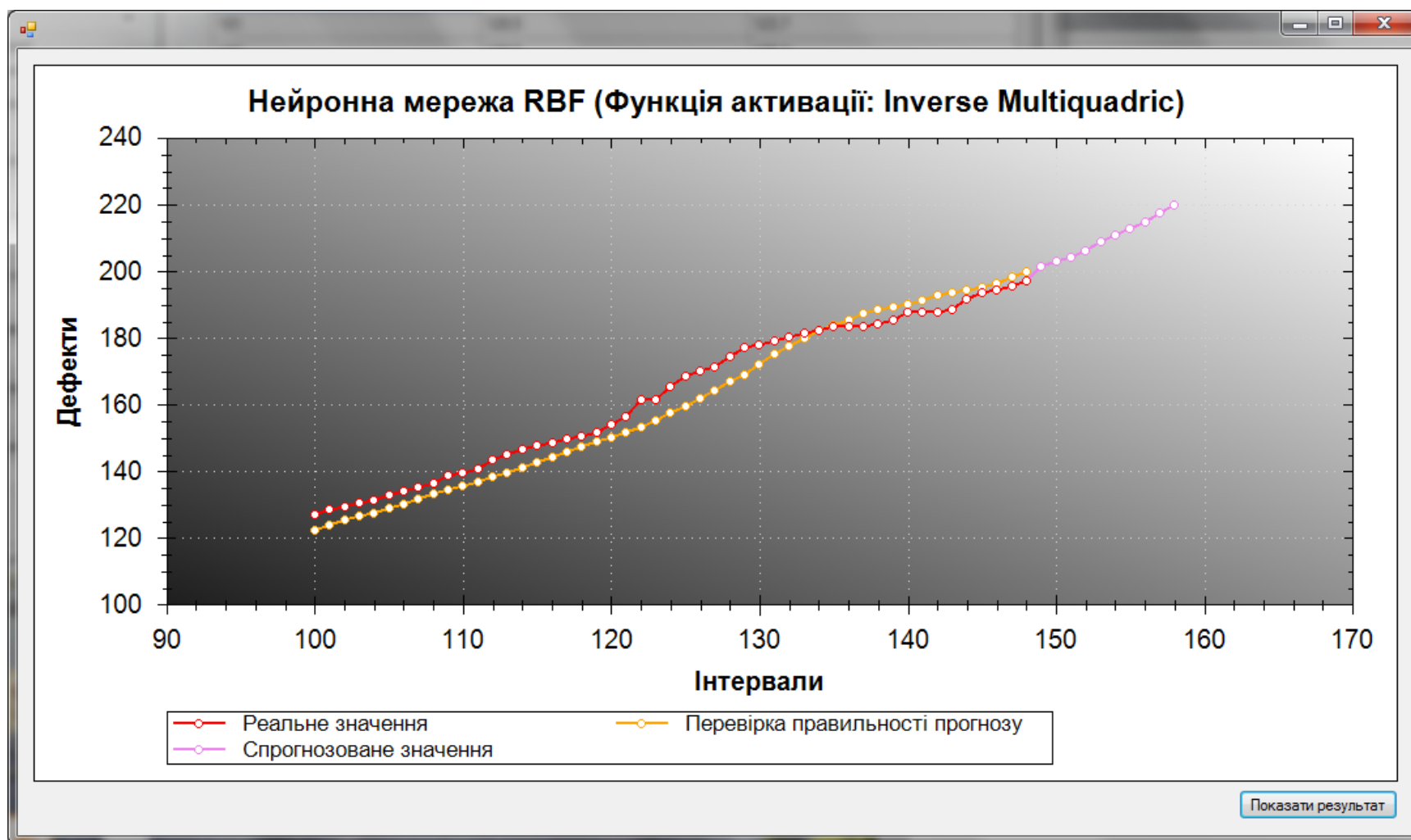
**Yakovyna, V., Symets, I.: Reliability assessment of CubeSat nanosatellites flight software by high-order Markov chains. Procedia Computer Science 192, 447–456 (2021)

PRZEWIDYWANIE AWARII OPROGRAMOWANIA METODAMI ANALIZY SZEREGÓW CZASOWYCH

Przewidywanie awarii oprogramowania sieciami neuronowymi

- **Dane wejściowe** – szeregi czasowe oparte na wynikach testowania dwóch programów typu open source: przeglądarki internetowej Chromium i systemu operacyjnego Chromium-OS
- **Sieci neuronowe** – RBF i rekurencyjna (Elman)
- **Funkcje aktywacji** – Gaussian, Multiquadric, Inverse Multiquadric, Mexican Hat (dla RBF); Liniowa, Tanh, Sin (dla Elman)
- Liczba neuronów **wejściowych** – 5, 10 ... 45
- Liczba **ukrytych** neuronów – 10, 30

Приклад przewidywania awarii systemu operacyjnego Chromium-OS



Zależność przewidywania awarii od liczby neuronów wejściowych sieci RBF

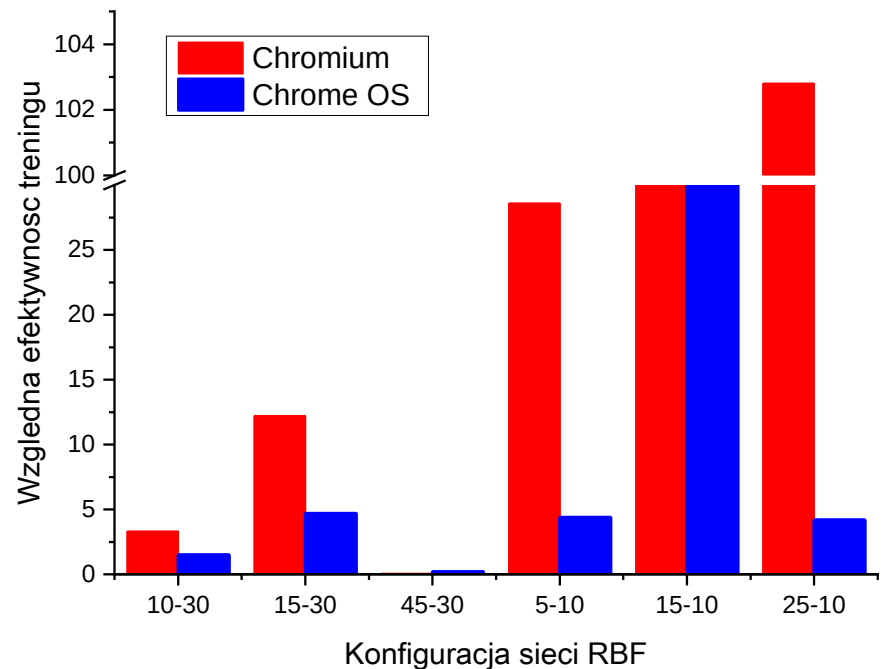
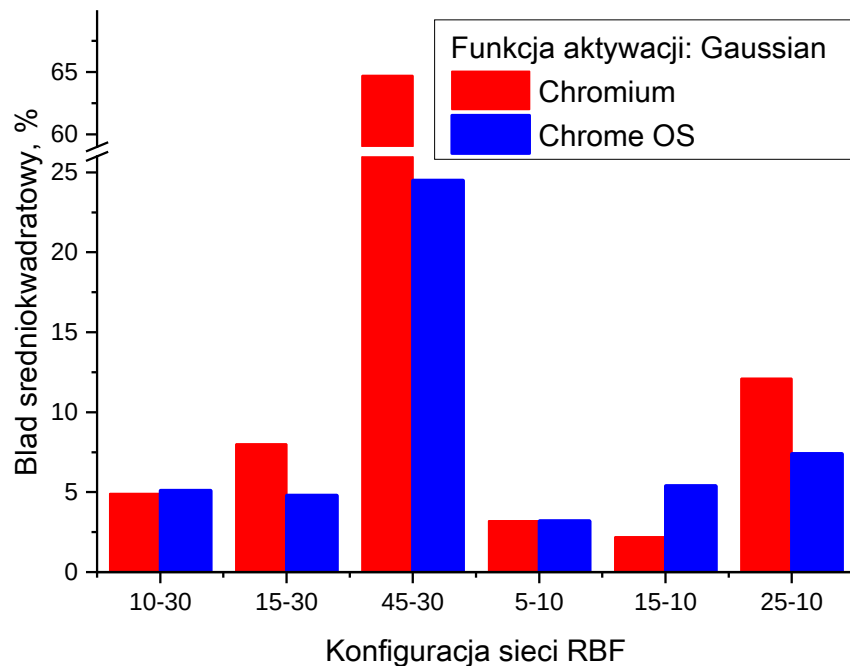
Przewidywanie awarii przeglądarki internetowej Chromium

Liczba neuronów wejściowych	Liczba ukrytych neuronów	Liczba epok treningowych		Współczynnik determinacji		Błąd średniokwadratowy, %	
		G	IM	G	IM	G	IM
10	30	146	478	0.996	0.998	4.9	1.3
15	30	14	171	0.997	0.998	8.0	2.4
45	30	443	27	0.997	0.994	64.7	29.0
5	10	69	1976	0.996	0.998	3.2	3.6
15	10	78	3916	0.997	0.998	2.2	12.4
25	10	68	6991	0.507	0.998	12.1	8.5

Przewidywanie awarii systemu operacyjnego Chromium-OS

Liczba neuronów wejściowych	Liczba ukrytych neuronów	Liczba epok treningowych		Współczynnik determinacji		Błąd średniokwadratowy, %	
		G	IM	G	IM	G	IM
10	30	212	324	0.882	0.983	5.1	3.2
15	30	59	279	0.969	0.970	4.8	3.3
45	30	113	26	0.988	0.971	24.5	22.4
5	10	364	1618	0.958	0.991	3.2	2.4
15	10	37	1536	0.987	0.979	5.4	3.0
25	10	62	260	0.982	0.967	7.4	3.4

Zależność przewidywania awarii od konfiguracji sieci RBF



Przewidywanie awarii metodami wygładzania, średniej ruchomej i sieci neuronowych*

Metoda	Pierwiastkowy błąd średniokwadratowy (RMSE)	Średni absolutny błąd skalowany (MASE)	Efektywność (sec)
Eksperyment 1			
Holt – Winters	12.93	1.37	2.81
ARIMA	9.83	1.15	0.49
Autoregresyjna SN	8.72	0.94	0.45
Eksperyment 2			
Holt – Winters	23.35	0.85	3.45
ARIMA	22.36	0.85	0.37
Autoregresyjna SN	8.72	0.32	0.49
Eksperyment 3			
Holt – Winters	126.44	6.12	3.48
ARIMA	14.26	0.66	0.44
Autoregresyjna SN	30.93	0.60	0.89
Eksperyment 4			
Holt – Winters	20.03	0.53	3.73
ARIMA	36.94	1.13	0.36
Autoregresyjna SN	16.57	0.44	0.30

*Yakovyna, V., Uhrynovskyi, B., Bachkay, O.: Software Failures Forecasting by Holt – Winters, ARIMA and NNAR Methods. In: Proc. of the IEEE 14th Intl. Conf. CSIT-2019, Lviv, Ukraine, pp. 151–155 (2019)

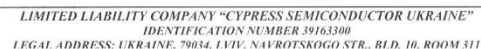
PODSUMOWANIE

Podsumowanie

- ✓ uogólniony model niezawodności oprogramowania ze wskaźnikiem złożoności pozwala zwiększyć dokładność oceny niezawodności
- ✓ kryterium wystarczalności testowania ułatwia podejmowanie decyzji na etapie testowania oraz zwiększa wiarygodność przewidywania liczby błędów programu
- ✓ regresyjny model przewidywania liczby awarii oprogramowania, który pozwala zwiększyć dokładność przewidywania liczby błędów programu lub skrócić czas testowania przy zachowaniu

Podsumowanie

- ✓ architektoniczne modele niezawodności oparte o proces Markowa wyższego rzędu z czasem ciągłym i dyskretnym pozwalają zwiększyć wiarygodność oceny niezawodności złożonych systemów programowych
- ✓ metoda reprezentacji procesu Markowa wyższego rzędu w postaci równoważnego procesu pierwszego rzędu ze stanami wirtualnymi ułatwia konstruowanie modeli niezawodności złożonych systemów programowych
- ✓ modele przewidywania awarii oprogramowania sieciami neuronowymi Elman i RBF dla stosowania na etapie eksploatacji do programów o różnym stopniu złożoności



DZIĘKUJĘ ZA UWAGĘ
