POLITECHNIKA POZNAŃSKA

Wydział Informatyki i Telekomunikacji

SYMULACJA CYFROWA

PROJEKT

Szymon Lewandowski

Metoda interakcji procesów

1. Treść zadania

W chińskiej restauracji pracuje k kelnerów obsługujących n_2 stolików dwuosobowych, n_3 stolików trzyosobowych oraz n_4 stolików czteroosobowych. Klienci pojawiają się w restauracji jako grupy 1-, 2, 3- lub 4-osobowe z prawdopodobieństwami odpowiednio p_1, p_2, p_3 oraz p_4. Odstęp czasu rozdzielający pojawienie się kolejnych grup klientów jest zmienną losową o rozkładzie normalnym ze średnią μ_a i wariancją σ_a^2 . Jeśli jest dostępny stolik odpowiadający wielkości grupy (lub większy), klienci są do niego prowadzeni przez kierownika sali (czynność ta zajmuje s jednostek czasu). W przeciwnym przypadku grupa oczekuje na stolik w kolejce. Średnio połowa klientów korzysta z samoobsługowego bufetu, przy którym może znajdować się jednocześnie b osób. Czas spędzany przy bufecie przez grupę klientów jest zmienną losową o rozkładzie normalnym ze średnią μ_b i wariancją σ_b^2 . Pozostali klienci są obsługiwani przez tego z kelnerów, który jako pierwszy będzie wolny. W pierwszej kolejności klienci otrzymują napoje, a następnie serwowane jest danie główne. Czas obsługi w obu przypadkach jest zmienną losową o rozkładzie wykładniczym ze średnimi odpowiednio λ_n oraz λ_j (te dwie wielkości uwzględniają zarówno czas oczekiwania na zrealizowanie zamówienia jak i sam czas podania napojów i posiłku głównego) Po zakończeniu konsumpcji, której długość jest zmienną losową o rozkładzie wykładniczym ze średnią λ_f, klient płaci jednemu z c zatrudnionych kasjerów. Czas obsługi przez kasjera jest zmienną losową o rozkładzie wykładniczym ze średnią λ_p.

W restauracji zamontowano nieprawidłowo system przeciwpożarowy. Co jakiś czas, bez przyczyny, rozlega się dźwięk alarmu. Część gości, świadoma nieprawidłowości, pozostaje na miejscu, natomiast reszta opuszcza restaurację. Odstęp czasu rozdzielający kolejne alarmy jest zmienną losową o rozkładzie normalnym ze średnią 4200 i wariancją 50^2 . Prawdopodobieństwo, że dana grupa nie opuści restauracji wynosi 70%.

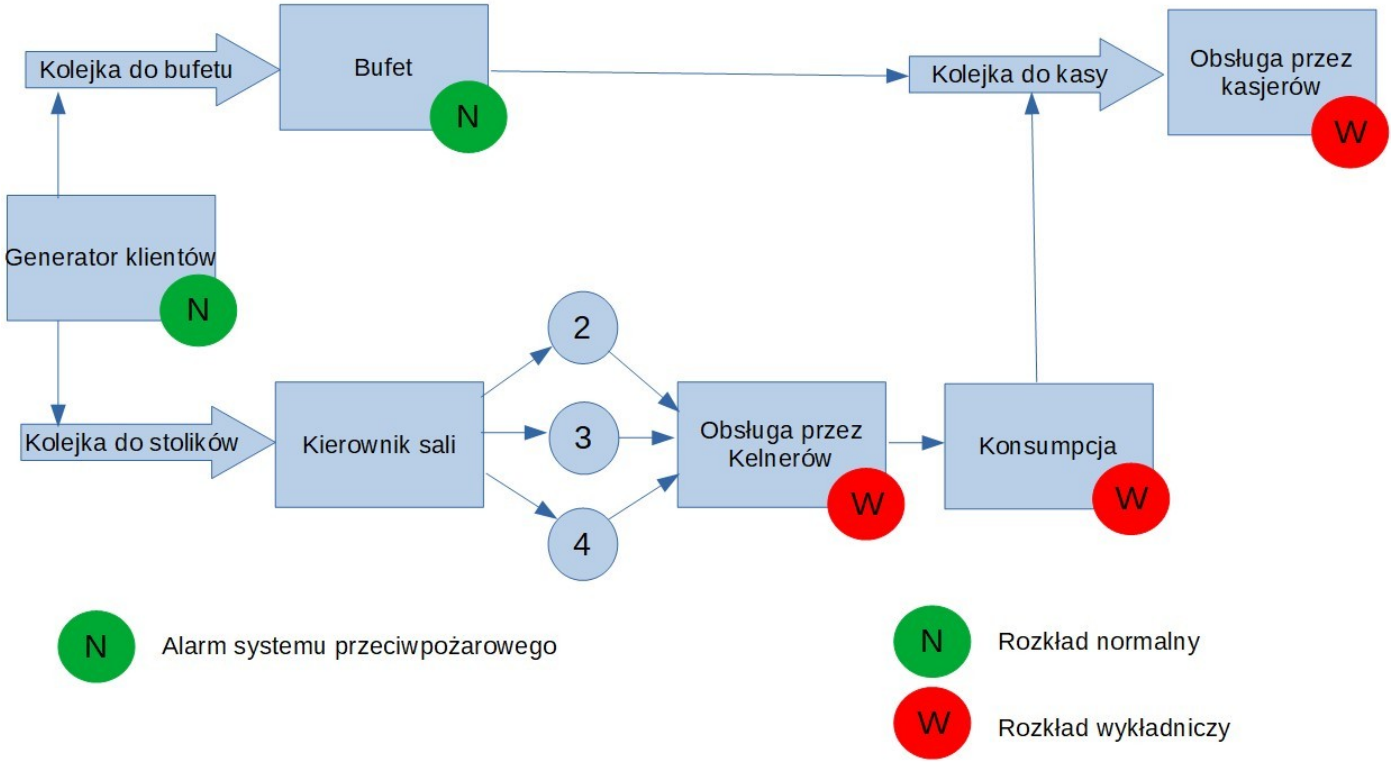
Zakładając, że kierownik sali zawsze wybiera stolik najlepiej pasujący do danej grupy oraz stosuje jedną z podanych niżej zasad obsługi kolejki, oszacuj za pomocą odpowiedniego eksperymentu symulacyjnego:

a) średni czas oczekiwania na stolik,
b) średnią długość kolejki oczekujących na stolik,
c) średni czas oczekiwania na obsługę przez kelnera od momentu zajęcia miejsca przy stoliku,
d) średnią długość kolejki przy kasach.

Strategia pracy kierownika sali:

S4 - Kierownik sali łączy wolne stoliki tak, aby zmieściła się pierwsza grupa w kolejce

2. Schemat modelu symulacyjnego i opis klas



Rys. 2.1. Schemat ideowy modelu symulacyjnego

Obiekt	Nazwa klasy	Opis	Atrybuty

restaurant	Restaurant	Klasa reprezentująca całą symulację. Tworzy i usuwa obiekty reprezentujące stałe zasoby systemu.	`Tables* tables_;`
			`Manager* manager_;`
			`Waiters* waiters_;`
			`Buffet* buffet_;`
			`Cash* cash_;`

tables_	Tables	Klasa implementuję kolejkę do stolików oraz same stoliki. Dostarcza odpowiednich funkcji pozwalających na sprawdzanie czy można dodać grupę do stolików. Implementuje mechanizm łączenia stolików oraz odpowiednie funkcje WakeUp.	`queue<Process*> queue_;`
			`queue<Process*>pending_processes_;`
			`const static int double_seats_ = 4;`
			`const static int  triple_seats_ = 10;`
			`const static int quadruple_seats_ = 4;`
			`double double_tables_[double_seats_][2]{};`
			`double triple_tables_[triple_seats_][3]{};`
			`double quadruple_tables_[quadruple_seats_][4]{};`
			`int queue_clients_ = 0;`

manager_	Manager	Klasa przechowuje jedynie informację o zajętości managera. Udostępnia metod pozwalające na rezerwację oraz zwolnienie managera.	`bool is_free_;`
waiters_	Waiters	Klasa przechowuje jedynie informacje o aktualnej zajętości kelnerów. Udostępnia metod pozwalające na rezerwację oraz zwolnienie kelnera.	`static const int waiters_ = 7;` `int actual_;`

buffet_	Buffet	Klasa implementuje kolejkę do bufetu oraz stanowiska prze bufecie.	`static const int number_of_seats_ = 14;`
			`int seats_[number_of_seats_] = {};`
			`queue<Process*> queue_;`

cash_	Cash	Klasa implementuje kolejkę do kasy oraz same stanowiska kasy.	`int cash_desks_[4]{};` `queue<Process*> queue_;`

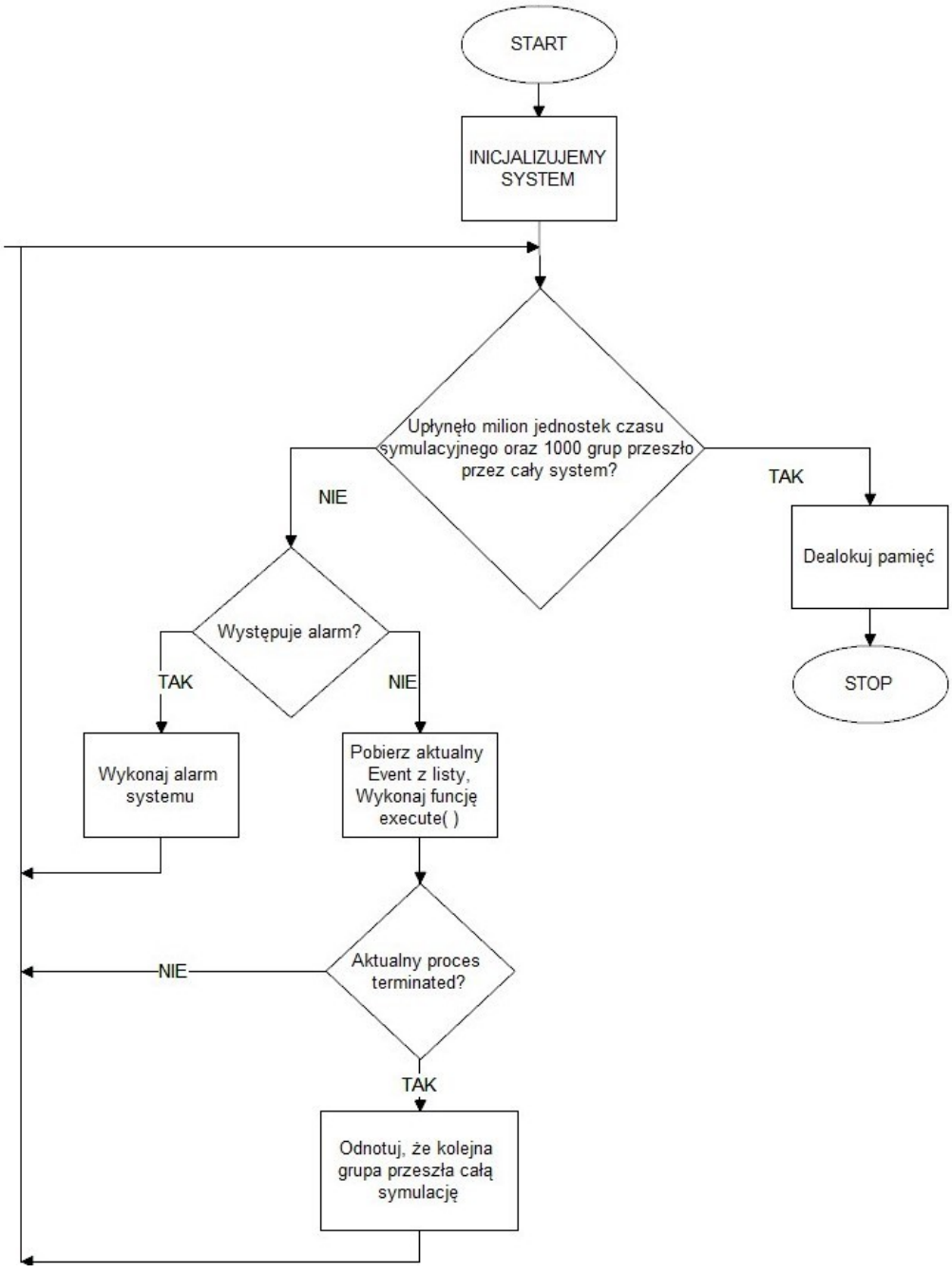
Process	process	Klasa bazowa procesów systemu. Przechowuje podstawowe dane konieczne do poprawnego sterowania metodą symulacyjną.	`int id_;`
			`int phase_;`
			`bool terminated_;`
			`int group_size_;`
			`Event_list* event_list_;`
			`Event* my_event_;`

customer	Customer	Klasa pochodna klasy Process. Implementuje virtualną metodę execute() klasy bazowej tym samym implementując sposób działania symulacji. Zawiera ponadto czasy przebywania danej grupy w kolejce do stolików oraz oczekiwania na obsługę przez kelnera.	`Restaurant* restaurant_;`
			`static const int seed_ = 8599765;`
			`double queue_for_tables_start_time_ = -1;`
			`double queue_for_tables_end_time_ = -1;`
			`double pending_processes_start_time_ = -1;`
			`double pending_processes_end_time_ = -1;`
			`bool drink_ = false;`
			`bool dish_ = false;`

Event	Event	Klasa implementuje zdarzenie powiązana na stałe z procesem którego dotyczy.	`double event_time_;`
			`Process* proc_;`
			`Event* next_;`
			`Event* prev_;`

event_list	Event_list	Klasa implementuje listę zawiadomień o zdarzeniach stosowaną do przechowywania obiektów klasy Event.	`Event* first_ = nullptr;` `Event* last_ = nullptr;`

3. Schemat blokowy pętli głównej i procesy



Rys. 2.2. Schemat blokowy pętli symulacyjnej

Proces klienta

Faza 0: Pojawienie się nowej grupy klientów

a) Zaplanuj pojawienie się nowej grupy klientów

b) Umieść grupę w kolejce do stolików lub w kolejce do kasy

c) Jeśli umieściłeś grupę w kolejce do stolików, a przy stolikach są wolne miejsca zdolne pomieścić grupę, manager jest wolny oraz kolejka do stolików jest pusta, przejdź do fazy 1

d) Jeśli umieściłeś grupę w kolejce do bufetu, grupa ta jest jedyną grupą w kolejce a w bufecie są wolne miejsca zdolne pomieścić grupę, przejdź do fazy 9

Faza 1: Początek obsługi przez managera

a) Zabierz pierwszą grupę z kolejki do stolików

b) Zajmij managera

c) Wpisz na listę zawiadomień o zdarzeniach komunikat o zakończeniu obsługi

Faza 2: Koniec obsługi przez managera

a) Przypisz grupę do wolnych miejsc przy stolikach

b) Zwolnij managera

c) Jeśli w kolejce do stolików jest oczekujący proces, przy stolikach jest wystarczająco wolnych miejsc aby go pomieścić a manager jest wolny, wybudź go

d) Jeśli jest wolny kelner przejdź do fazy 3

Faza 3: Kelner podaje napój

a) Zajmij kelnera

b) Wpisz na listę zawiadomień o zdarzeniach komunikat o zakończeniu obsługi

Faza 4: Kelner podaje danie główne

a) Odnotuj otrzymanie napoju

b) Wpisz na listę zawiadomień o zdarzeniach komunikat o zakończeniu obsługi

Faza 5: Rozpoczęcie konsumpcji

a) Odnotuj otrzymanie dania głównego

b) Zwolnij kelnera

c) Jeśli przy stolikach znajduje się oczekujący proces, wybudź ten który czeka najdłużej

d) Wpisz na listę zawiadomień o zdarzeniach komunikat o zakończeniu konsumpcji

Faza 6: Koniec konsumpcji

a) Zabierz grupę ze stolików i przenieś do kolejki do kasy

b) Jeśli w kolejce do stolików jest oczekujący proces, przy stolikach jest wystarczająco wolnych miejsc aby go pomieścić a manager jest wolny, wybudź go

c) Jeśli jest wolne stanowisko przy kasie, idź do fazy 7

Faza 7: Rozpoczęcie obsługi przy kasie

a) Zabierz grupę z kolejki do kasy i przypisz stanowisku

b) Wpisz na listę zawiadomień o zdarzeniach komunikat o zakończeniu obsługi

Faza 8: Koniec obsługi przy kasie

a) Zwolnij stanowisko przy kasie

b) Ustaw aktualną grupę jako ukończoną

c) Jeśli w kolejce do kasy jest oczekujący proces, wybudź go

Faza 9: Początek obsługi w bufecie

a) Zabierz grupę z kolejki do bufetu i umieść w bufecie

b) Wpisz na listę zawiadomień o zdarzeniach komunikat o zakończeniu obsługi

Faza 10: Koniec obsługi w bufecie

a) Zabierz grupę z bufetu i umieść w kolejce do kasy

b) Jeśli w kolejce do bufetu jest oczekujący proces, a liczba wolnych miejsc jest w stanie pomieścić grupę, wybudź go

c) Jeśli jest wolne stanowisko przy kasie, idź do fazy 7

5. Parametry wywołania programu

Nadejście nowej grupy klientów:
Rozkład normalny z średnią 500 i wariancją 100

Czas podania napoju przez kelnera:
Rozkład wykładniczy z średnią 900

Czas podania dania głównego przez kelnera:
Rozkład wykładniczy z średnią 2500

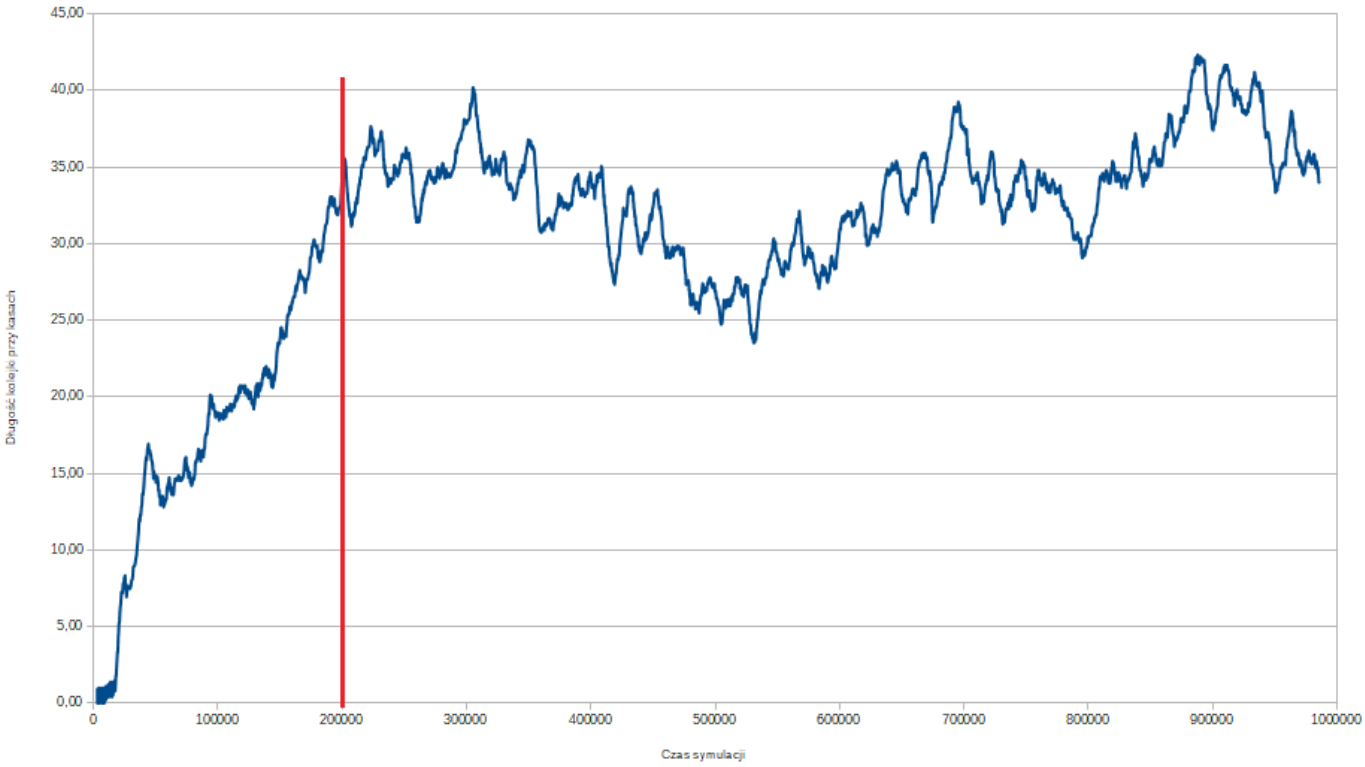
Czas konsumpcji przez grupę:
Rozkład wykładniczy z średnią 1900

Czas obsługi przy kasie:
Rozkład wykładniczy z średnią 900

Czas obsługi w bufecie:
Rozkład normalny z średnią 3200 i wariancją 100

Czas między kolejnymi alarmami:
Rozkład normalny z średnią 25000 i wariancją 500

symulacja:	ziarno:
1	154796
2	9438215
3	6559724
4	3297156
5	2189759
6	5781349
7	1674932
8	342756
9	784392
10	1628453
11	5974285
12	29135744
13	84953713
14	1584928
15	8599765



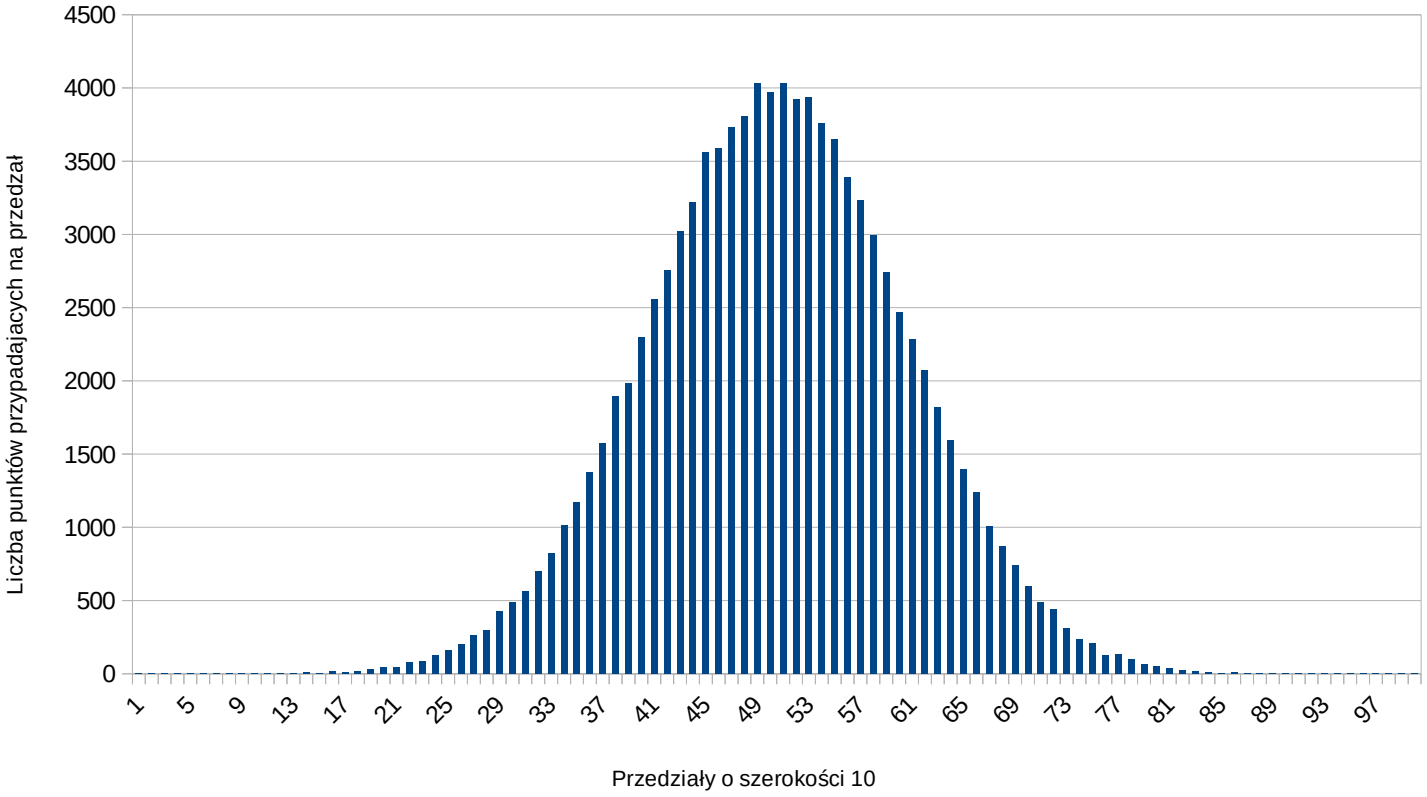
Rys.5.1. Średnia długość kolejki przy kasach dla 15 przebiegów symulacyjnych w funkcji czasu bezwzględnego symulacji

Na wykres czerwoną linią naniesiono granicę stanu końca fazy początkowej.

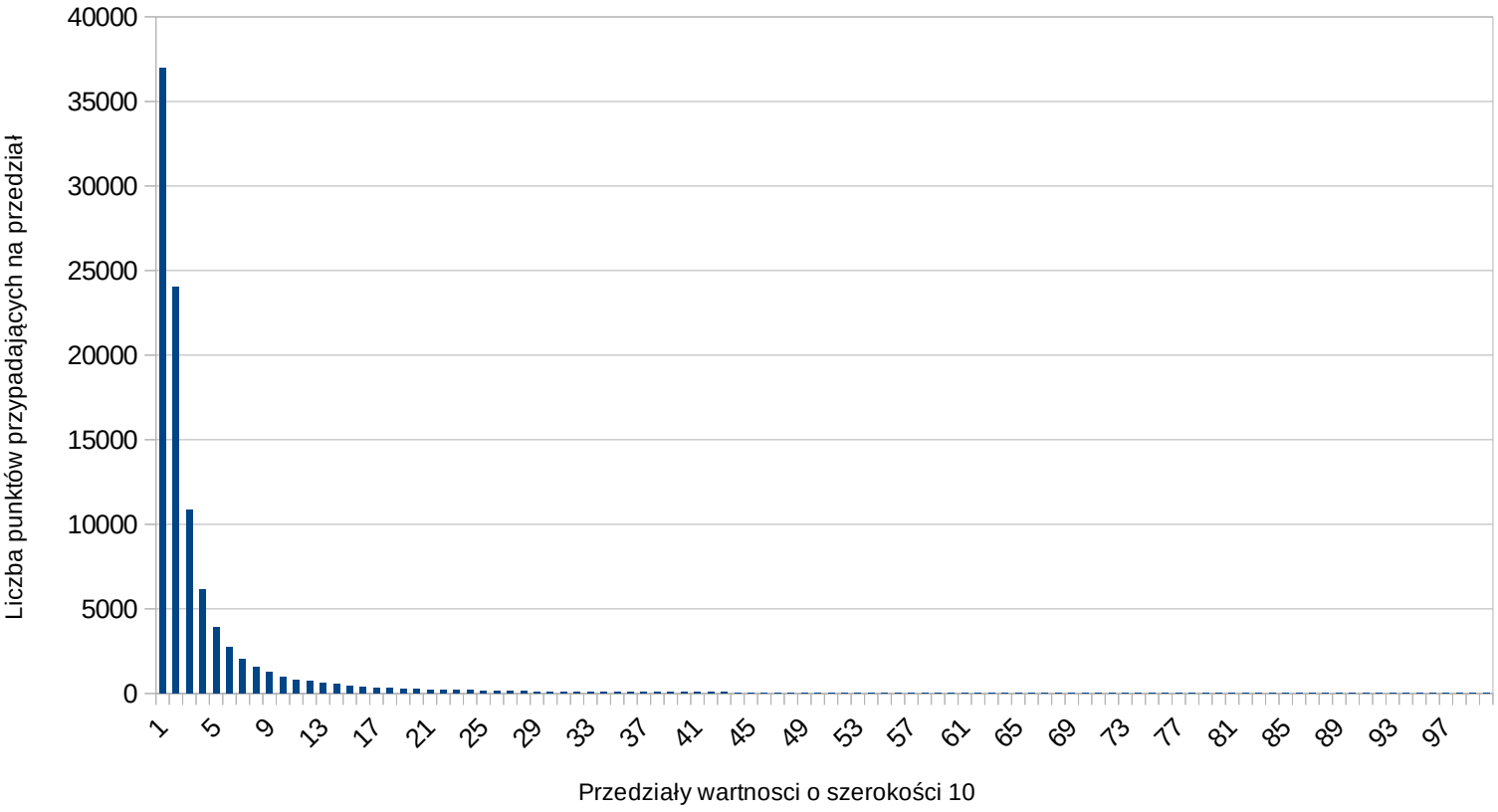
6. Generatory

W symulacji zastosowano generatory pseudolosowe będące częścią STL. Dostępne są one w bibliotece <random>.

Poniżej zaprezentowane histogramy zastosowanych generatorów.



Rys. 6.1. Histogram rozkładu normalnego o średniej 5000 i wariancji 1000 w przedziale od 0 do 10000



Rys. 6.2. Histogram rozkładu wykładniczego o średniej 100 w przedziale od 0 do 10000

W celu zapewnienia niezależności wyników symulacji zastosowano metodę wielokrotnych replikacji. Symulację przeprowadzono piętnaście razy z tymi samymi danymi wejściowymi ale różnymi wartościami ziaren (ceedów) generatorów pseudolosowych.

7. Metody testowania i weryfikacji poprawności programu

- Analizowanie w trybie krokowym poprawności działania symulacji
- Umieszczanie pułapek wyświetlających komunikat błędu i zatrzymujących program, w miejscach których program nie powinien osiągnąć przy prawidłowej pracy
- Stosowanie debugera Visual Studio do monitorowania stanu symulacji, np. ilości stosowanej przez program pamięci w celu walidacji czy nie dochodzi do wycieków pamięci
- Manipulowanie wartościami generatorów (średnimi i wariancjami) w celu wprowadzenie symulacji w stany skrajne i obserwacja poprawności jej działania
- Zaimplementowanie metod wyświetlających aktualny stan systemu po każdej iteracji pętli symulacyjnej
- Krytyczna interpretacja danych zbieranych do plików w ramach statystyk

8. Wyniki i wnioski

Nr symulacji	Średni czas oczekiwania na stolik	Długość kolejki oczekujących na stolik	Średni czas oczekiwania na obsługę przez kelnera	Długość kolejki przy kasach
1	31770,42	37,58	7969,75	35,43
2	24661,80	34,91	9090,60	17,96
3	30795,77	34,84	8401,12	30,69
4	27317,13	36,03	11260,66	28,11
5	29624,51	34,65	10178,53	30,61
6	27289,06	32,51	8927,69	48,88
7	27880,41	27,24	7691,87	29,98
8	27641,49	36,92	10082,96	30,74
9	26217,24	29,70	7501,42	39,30
10	30718,50	36,22	8822,03	24,48
11	26932,79	33,51	7579,90	31,87
12	26735,82	29,93	8012,69	27,12
13	26234,82	28,05	7921,97	32,91
14	26200,19	27,84	7633,90	33,96
15	26674,47	29,29	9336,79	46,50
Średnia:	27779,63	32,61	8694,13	32,57
Odchylenie standardowe:	2028,70	3,61	1128,35	7,85
Przedział ufności:	1026,65	1,83	571,01	3,97

Uzyskanie dobrych jakościowo wyników wymaga znacznej ilości niezależnych iteracji symulacji. Wynika to między innymi z wpływu alarmu mającego znaczny wpływ na stan systemu.

Dalsze wydłużenie czasu trwania pojedynczej symulacji oraz ich ilości pozytywni wpłynęła by na wiarygodności otrzymywanych wyników.

Stworzenie projektu programistycznego jakim jest symulacji cyfrowa zdarzeń dyskretnych znacznie poszerzyło moje umiejętności programistyczne. Pozwoliło zastosować bardziej zaawansowane narzędzia dostarczane przez język C++ oraz zapoznać się w praktyce z możliwościami IDE (Visual Studio). Poznałem podstawy Google C++ Style Guide oraz zacząłem stosować oprogramowanie Resharper Ultimate (JetBrains). Nie mniej jednak największą korzyścią osiągniętą w tym projekcie była możliwość testowanie i walidacji poprawności złożonego oprogramowania.