Implementacja algorytmu EDF w aplikacji .NET Framework

Dokumentacja projektu zaliczeniowego dla przedmiotu Modelowanie Procesów Dyskretnych.

Przemysław Kleszcz

Informatyka II stopień (niestacjonarne)

Styczeń 2018

Spis treści

1. Wprowadzenie

W zakresie projektu zaliczeniowego zrealizowana została aplikacja typu *Desktop* w technologii .NET (język C#). Wykorzystana została rama projektowa WPF (*Windows Presentation Foundation*). Wykonany program posiada graficzny interfejs użytkownika, przy pomocy którego użytkownik wprowadza i przetwarza dane oraz jest w stanie obserwować wyniki zaistniałych w czasie działania programu operacji.

Zadaniem aplikacji jest szeregowanie zdefiniowanych przez użytkownika zadań cyklicznych, za pomocą algorytmu EDF oraz ukazanie reprezentacji wyników szeregowania za pomocą graficznego interfejsu aplikacji. Interfejs zapewnia możliwość dodawania oraz usuwania zadań z listy oraz wygenerowania wyników przeliczeń.

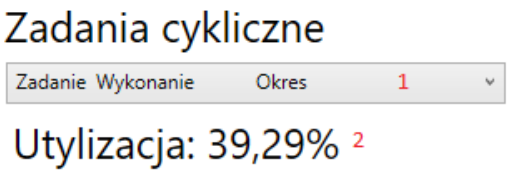
2. Graficzny interfejs użytkownika

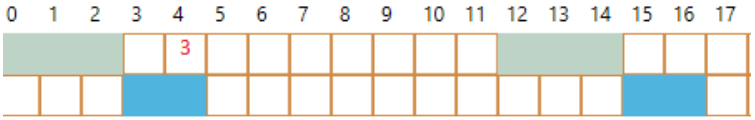
2.1 Przedstawienie rozwiązania

Do zaprojektowania interfejsu użytkownika aplikacji komputerowej wykorzystano platformę WPF. Powodem takiej decyzji była łatwość i szybkość implementacji. Dzięki zastosowaniu odpowiednich mechanizmów stanowiących warstwę abstrakcji dla natywnych bibliotek systemu Windows, aplikacja została zbudowana zgodnie z zasadami RAD (*Rapid application development*) – serią zasad oraz specyfikacji ułatwiających i przyspieszających w znacznym stopniu działania implementacyjne.

2.2 Szczegóły interfejsu

1. Pole tekstowe pozwalające na wprowadzenie czasu wykonania dla nowego zadania.
2. Pole tekstowe pozwalające na wprowadzenie okresu dla nowego zadania.
3. Przycisk, po którego kliknięciu nowe zadanie zapisuje się do listy zadań.
4. Przycisk, po którego kliknięciu następuje szeregowanie zapisanych w liście zadań.
5. Pole wyboru – po wybraniu określonego zadania, daje możliwość usunięcia zadania z listy.
6. Pole określające ile zadań na liście oczekuje na szeregowanie w danym czasie.
7. Współczynnik utylizacji dla wszystkich zadań na liście.





1. Pole wyboru pozwalające na identyfikację określonego zadania.
2. Współczynnik utylizacji dla zadań poddanych szeregowaniu.
3. Graficzna reprezentacja uszeregowania.

3. Charakterystyka środowiska implementacyjnego

Aplikacja została w pełni opracowana w środowisku *Microsoft Visual Studio.* Jest to zintegrowane środowisko produkcyjne (IDE) zaprojektowane przez firmę *Microsoft.* Jest używane do produkcji programów komputerowych na platformę Windows. Za jego pomocą można produkować zarówno kod natywny jak i zarządzany przez platformę .NET.

4. Opis stosowanych technologii

- *.NET Framework* – Wykorzystany w niniejszym projekcie .NET Framework to nic innego jak środowisko uruchomieniowe dla napisanych programów a także odpowiedni zbiór typów bazowych, z których te programy korzystają. Programy tworzone na platformę .NET mogą być tworzone w wielu językach programowania naraz. Odpowiednie dane wynikowe każdej kompilacji – pakiet a także jego metadane stanowią klucz do tej funkcjonalności. Albowiem kompilacja źródeł każdego języka platformy produkuje wynik w postaci kodu CIL (kodu pośredniego). Treść tego kodu jest taka sama niezależnie od zastosowanego rozwiązania językowego.

-*WPF* – Nazwa biblioteki i silnika graficznego wchodzącego w skład .NET Framework. Zestaw WPF integruje interfejs użytkownika, grafikę 2D i 3D oraz multimedia. Interfejs tej platformy opiera się głównie na języku XAML.

5. Implementacja

Aplikacja składa się z jednego komponentu stworzonego w technologii WPF przy użyciu modelu *code behind.*

5.1 Elementy składowe

Na kod źródłowy aplikacji składa się dziesięć plików. Są to pliki konfiguracyjne *xml*, pliki *cs* z kodem w języku C# oraz pliki *XAML* definiujące wygląd interfejsu użytkownika.

App.config – Plik konfiguracyjny aplikacji.

App.xaml – Punkt startowy każdej aplikacji WPF.

ProcessInput.xaml – Główne okno aplikacji. Plik zawiera strukturę interfejsu użytkownika oraz powiązaną z nim logikę biznesową.

MainWindow.xaml – Okno prezentujące wyniki szeregowania.

PeriodicTask.cs – Plik zawiera definicję struktury opisującej zadanie cykliczne.

PeriodicTaskManager.cs – Plik zawiera definicję struktury zawierającej główną logikę biznesową szeregowania zadań.

Task.cs – Plik zawiera definicję struktury opisującej parametry zadania.

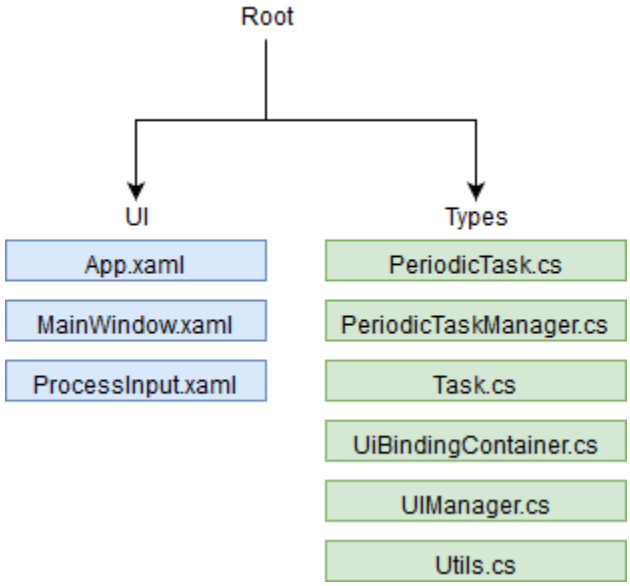
UIManager.cs – Plik zawiera definicję struktury zawierającej główną logikę biznesową

związaną z manipulacją interfejsem użytkownika.

UIBindingContainer.cs – Plik grupuje odniesienia do wybranych komponentów interfejsu użytkownika.

Utils.cs – Wszelkie niezbędne metody pomocnicze.

5.2 Struktura projektu



5.3 Algorytm

Algorytm EDF przypisuje priorytety do zadań w prosty sposób: priorytet zadania jest odwrotnie proporcjonalny do jego okresu. Innymi słowy największy priorytet posiada zadanie z najkrótszym okresem. W przypadku kiedy dwa lub więcej zadań posiadają taką samą wartość okresu, jako zadanie z najwyższym priorytetem wybierane jest pierwsze napotkane w kolejce.

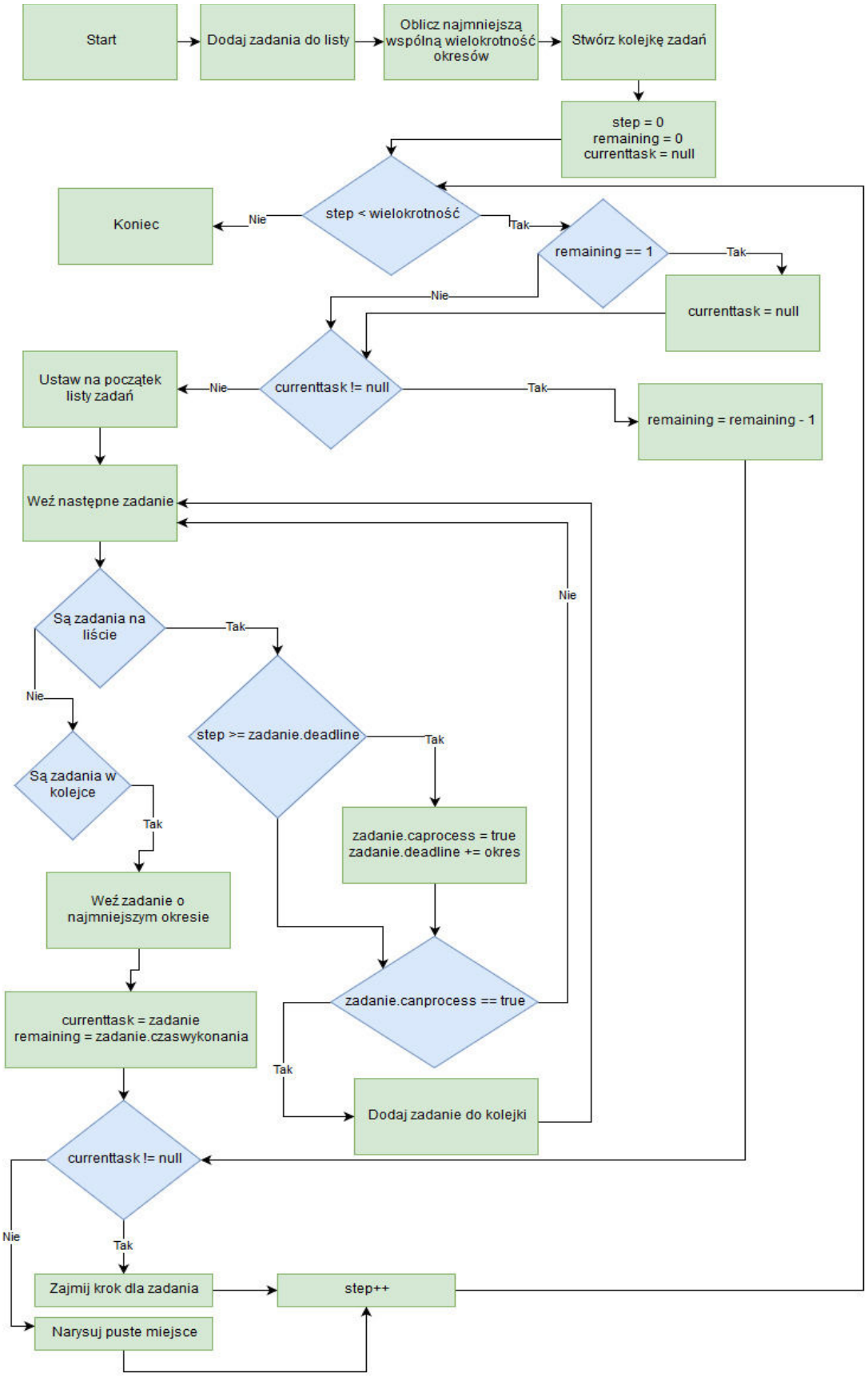
Założenia:

- Szeregowanie zadań cyklicznych

- Zadanie musi zostać zakończone przed rozpoczęciem przetwarzania następnego

- Zadania są względem siebie niezależne

- Czas wykonania dla każdego zadania jest stały i nie zmienia się.


Start
Dodaj zadania do listy
Oblicz najmniejszą wspólną wielokrotność okresów
Stwórz kolejkę zadań
step = 0
remaining = 0
currenttask = null
step < wielokrotność
Nie
Koniec
Tak
remaining == 1
Tak
currenttask = null
Nie
currenttask != null
Nie
Ustaw na początek listy zadań
Tak
remaining = remaining - 1
Weź następne zadanie
Są zadania na liście
Tak
Nie
Są zadania w kolejce
Tak
Weź zadanie o najmniejszym okresie
step >= zadanie.deadline
Tak
zadanie.caprocess = true
zadanie.deadline += okres
Nie
zadanie.canprocess == true
Tak
Dodaj zadanie do kolejki
currenttask = zadanie
remaining = zadanie.czaswykonania
currenttask != null
Nie
Tak
Zajmij krok dla zadania
step++
Narysuj puste miejsce

Zadanie szeregowane algorytmem EDF można scharakteryzować poprzez dwa parametry:

- Okres

- Czas wykonania

Aby oznaczyć *m* cyklicznych zadań, możemy zapisać je jako: *P1, P2,…, Pm,* okresy zadań jako *T1, T2,…,Tm* oraz czasy wykonania jako odpowiednio *C1,C2,…,Cm*. Zatem zadanie *Pi* jest możliwe do wykonania co *Ti* jednostek i musi trwać maksymalnie *Ci* jednostek. Musi zostać zachowana nierówność *Ci <= Ti.* Zbiór cyklicznych zadań jest możliwy do uszeregowania tylko wtedy jeśli spełniony zostanie warunek:

$$U = \sum_{i=1}^{N} \frac{Ci}{Ti} \leq \quad 1$$

6. Ograniczenia aplikacji i przyszłe możliwości rozwoju

W chwili obecnej limit możliwych do szeregowania zadań w aplikacji to dziesięć. Wynika to z pewnych ograniczeń nałożonych przez zaprojektowany interfejs użytkownika. Rozwijając projekt w przyszłości można by rozwinąć aplikację w taki sposób aby możliwe stało się dodawanie nieograniczonej liczby zadań (U <= 1).

7. Podsumowanie i wnioski

Główne zadanie postawione przed programem zostało zrealizowane. Definiowane przez użytkownika zadania są szeregowane w sposób poprawny za pomocą algorytmu EDF a interfejs użytkownika w czytelny sposób prezentuje wyniki szeregowania na ekranie.