Jarosław Kuchta

Projekt ModelGen

data pierwszego opracowania: 27.10.2023

data ostatniej modyfikacji: 01.01.2024

# Wymagania systemowe

System ModelGen ma wspierać działania konstruktora modelu obiektowego dokumentu. Ma być aplikacją desktopową systemu Windows przeznaczoną do użytku indywidualnego. Ponieważ proces tworzenia modelu obiektowego dokumentu jest procesem złożonym, wymagającym częstej interakcji z użytkownikiem, będzie to aplikacja typu okienkowego z możliwością tworzenia diagramów i z interakcjami na tych diagramach. Ponieważ biblioteki źródłowe, z których typy będą wczytywanie i przetwarzane, zostały napisane w języku C#, więc aplikacja też będzie napisana w tym języku. Do wczytywania typów zostanie użyta biblioteka System.Reflection, a do konstrukcji interfejsu użytkownika framework WPF.

## Wymagania funkcjonalne

Z koncepcji procesu przetwarzania wynikają następujące wymagania funkcjonalne:

* [FREQ\_01] Wybór bibliotek źródłowych – do wyboru są dwa pakiety bibliotek DLL:
  + Microsoft.Office.Interop.Word,
  + DocumentFormat.OpenXml.

Być może do tego zostaną dołączone inne pakiety, np.

* + DocumentFormat.OpenXmlSDK.

Oprócz tego generowane przez aplikację i modyfikowane przez konstruktora moduły z typami docelowego modelu dokumentu będą kompilowane i mogą być traktowane jako biblioteki źródłowe.

* [FREQ\_02] Wczytywanie typów – typy publiczne z biblioteki źródłowej są wczytywane do programu ModelGen z uwzględnieniem przestrzeni nazw, właściwości publicznych i wartości zdefiniowanych w typach wyliczeniowych.
* [FREQ\_03] Filtrowanie typów –przestrzenie nazw, typy, właściwości i wartości są oznaczane jako zaakceptowane do dalszego przetwarzania lub odrzucone. Nie wszystkie muszą być zaakceptowane od razu.
* [FREQ\_04] Wczytanie wewnętrznej struktury typów – począwszy od wybranego przez konstruktora typu głównego zaakceptowane właściwości publiczne typów zaakceptowanych są rozwijane. Wczytywane są typy wartości tych właściwości.
* [FREQ\_05] Określanie relacji między typami – pomiędzy wczytanymi typami elementów określanie i zapamiętywane są relacje dziedziczenia, implementowania interfejsów oraz zawierania. W tym ostatnim przypadku może być potrzebna analiza schematów XML formatu OpenXml.
* [FREQ\_06] Uzupełnianie dokumentacji – typy akceptowane i właściwości akceptowane mają wczytywane opisy bezpośrednio z biblioteki lub z uprzednio przygotowanego pliku XML.
* [FREQ\_07] Określenie docelowej struktury nazw – na tę strukturę składają się przestrzenie nazw, nazwy typów, właściwości i wartości wyliczeniowych. Przewiduje się stosowanie analizy relacji między typami oraz reguł transformacji nazw z bibliotek źródłowych.
* [FREQ\_08] Określenie reguł transformacji typów, właściwości i wartości wyliczeniowych – reguły transformacji będą opisywać sposób wykorzystywania konwerterów zaimplementowanych przez konstruktora oraz zmiany nazw elementów źródłowych na docelowe.
* [FREQ\_09] Konwertowanie elementów – przetwarzanie elementów źródłowych na docelowe zgodnie z określonymi regułami.
* [FREQ\_10] Generowanie elementów wynikowych –kod elementów docelowych jest generowany i umieszczany w plikach o odpowiednich nazwach w katalogach tworzonych w ścieżce plików wynikowych na podstawie przestrzeni nazw.

W tym etapie mogą być wykorzystywane różne typy generatorów, które będą dawały różne wyniki. Przewidywane są następujące typy generatorów:

* [GEN\_01] Generator interfejsów– produkuje interfejsy docelowego modelu dokumentu.
* [GEN\_02] Generator klas adapterowych– produkuje klasy implementujące interfejsy docelowego modelu dokumentu, które zawierają obiekty klas OpenXml.
* [GEN\_03] Generator klas autonomicznych – produkuje klasy implementujące interfejsy docelowego modelu dokumentu, które działają niezależnie od klas OpenXml.
* [GEN\_04] Generator konwerterów – produkuje klasy typu Reader i Writer tłumaczące dokumenty OpenXml na dokumenty w docelowym modelu obiektowym.

Ostatnim krokiem przetwarzania będzie:

* [FREQ\_11] Kompilacja kodu źródłowego – wygenerowany kod źródłowy ma być kompilowany do postaci biblioteki DLL. Umożliwi to z jednej strony zweryfikowanie wyników przetwarzania (poprawność syntaktyczną generowanego kodu), a z drugiej przygotuje biblioteki wynikowe do ponownego wykorzystania w kolejnej iteracji procesu przetwarzania.

## Wymagania na dane

System ModelGen ma przetwarzać typy programowe zdefiniowane w bibliotekach wejściowych (Microsoft.Office.Interop.Word, DocumentFormat.OpenXml) na typy docelowego modelu danych, zapisywać je w postaci kodu źródłowego w języku C# i po skompilowaniu tworzyć biblioteki wynikowe. Dane źródłowe i wynikowe procesu będą więc przechowywane w bibliotekach DLL w postaci kodu CIL (Common Intermediate Language), czyli kodu pośredniego systemu Windows, który umożliwia łatwe pozyskanie informacji o definiowanych typach przez refleksję typów.

Jednak wiele informacji o wczytywanych i generowanych typach będzie musiało być tworzonych w procesie przetwarzania przez konstruktora modelu. Dane te będą zapisywane w metamodelu docelowego modelu dokumentu. Celowe jest znalezienie odpowiedniej formy przechowywania tych danych, aby uniknąć wielokrotnego wprowadzania tej samej informacji przez użytkownika.

Odrębną kategorię stanowić będą dane konfiguracyjne, które opisywać będą reguły odwzorowania typów źródłowych na typy wynikowe, w tym reguły przetwarzania nazw, właściwości i wartości wyliczeniowych oraz reguły użycia konwerterów dokumentów. Ze względu na dużą liczbę danych przetwarzanych reguły powinny uwzględniać symbole wieloznaczne (typu ‘\*’).

## Wymagania jakościowe

Wymagania jakościowe z reguły dzielą się na cztery grupy:

* wymagania dot. wiarygodności,
* wymagania dot. efektywności,
* wymagania dot. elastyczności,
* wymagania dot. użyteczności.

### Wymagania dot. wiarygodności

Wymagania dot. wiarygodności dotyczą trzech głównych kroków procesu przetwarzania:

* [RREQ\_01] Zapewnienie wiarygodności wczytywanych danych wejściowych – ponieważ danymi wejściowymi są specyfikacje typów skompilowanych w bibliotekach DLL, które już przeszły proces weryfikacji w czasie kompilacji, więc można uznać, że są one poprawne.
  + Wątpliwość można mieć do kompletności wczytywanych danych. Można to zweryfikować porównując liczbę zdefiniowanych typów, właściwości i wartości z liczbą wczytanych elementów tych rodzajów. Ponadto można sprawdzić, czy każda zaakceptowana do przetwarzania właściwość ma wczytany typ danych.
  + Można mieć wątpliwość co do spójności danych wczytanych i zaakceptowanych do przetwarzania. Np. można sprawdzić, czy każdy typ zaakceptowany do przetwarzania może być dostępny z innego typu (oczywiście z wyjątkiem typów głównych).
  + Można sprawdzić spójność między danymi wczytanymi z biblioteki Microsoft.­Office.­Interop.­Word i DocumentFormat.­OpenXml – brak zgodności z którejś ze stron oznacza, albo brak odpowiedniej definicji, albo brak odpowiedniej reguły odwzorowania i powinno to zostać uzupełnione przez użytkownika.
  + Można sprawdzić, czy każdy typ, właściwość i wartość wyliczeniowa ma opis pobrany z dokumentacji i czy jest to opis wystarczający (niewystarczający będzie opis trywialny, np. opis klasy „Style” w postaci „To jest klasa Style”). Jeśli nie ma takiego opisu, to wskazane jest zapewnić możliwość wczytywania opisów z zewnętrznego pliku z dokumentacją.
* [RREQ\_02] Zapewnienie wiarygodności odwzorowania i konwersji danych – można to zweryfikować poprzez testy konwersji dokumentów DOCX na dokumenty zapisywane w docelowym modelu i odwrotnie. Nie musi być zapewniona identyczność zapisu XML, ale poprawność dokumentów po konwersji zwrotnej i ich zgodność ze źródłowymi oceniana przez użytkownika.
* [RREQ\_03] Zapewnienie wiarygodności danych wynikowych – ponieważ dane wynikowe są najpierw generowane w kodzie C#, więc ich poprawna kompilacja stanowić będzie odpowiednią rękojmię ich poprawności.

### Wymagania dot. efektywności

Proces przetwarzania zapewniany przez aplikację ModelGen musi być efektywny na tyle, aby był użyteczny dla pojedynczego konstruktora. Stąd wymaganie:

* [PREQ\_01] Krótki czas przetwarzania – całkowity czas przetwarzania nie powinien być dłuższy niż 3 minuty.

### Wymagania dot. elastyczności

Program ModelGen jest przeznaczony na system operacyjny Windows i dla pojedynczego użytkownika. Dodatkowym wymaganiem dot. elastyczności jest:

* [FREQ\_01] Możliwości dodawania innych bibliotek źródłowych – przewidywana jest np. możliwość wykorzystania biblioteki DocumentFormat.­OpenXmlSDK.

### Wymagania dot. użyteczności

Wymagania dot. użyteczności określają:

* [UREQ\_01] Możliwość przeglądania wszystkich danych – typów, właściwości i wartości wyliczeniowych niezależnie od ich położenia w strukturze.
* [UREQ\_02] Łatwość filtrowania i sortowania danych – wg dowolnego pola lub wielu pól.
* [UREQ\_03] Możliwość przeglądania wyników pośrednich – dotyczy efektów kolejnych etapów procesu przetwarzania.
* [UREQ\_04] Możliwość przeglądania elementów w postaci drzewa – z rozwijaniem i zwijaniem poszczególnych elementów złożonych.
* [UREQ\_05] Możliwość przeglądania relacji między typami – w postaci graficznej ze zwiększaniem i zmniejszaniem liczby widocznych szczegółów.
* [UREQ\_06] Generowanie reguł – odwzorowania i przekształcania z użyciem symboli wieloznacznych na podstawie aktualnych ustawień szczegółówych.