Projekt TKOM
Jakub Strawa 300266

Temat: Łączenie plików emx zawierających diagram klas wykonany w programie Rational Software Architect Designer w jeden spójny diagram.

1. Zakładane funkcjonalności
   We wszystkich poniższych przykładach xmi:id reprezentuje unikalne id konkretnego elementu modelu.
   1.1 Dopuszczalne konstrukcje językowe z przykładami
   - Klasa
     Przykładowa klasa Auto:
     <packagedElement xmi:type="uml:Class"
     xmi:id="_uk1mEJCaEeuG94qDkPAb9w" name="Auto">
     </packagedElement>

   - Stereotyp
     Przykładowe stereotypy pojazd i koła:
     <eAnnotations xmi:id="_bmHsIJrIEeuG7NKZ46hTYw"
     source="http://www.eclipse.org/uml2/2.0.0/UML">
     <details xmi:id="_d1H6UJrIEeuG7NKZ46hTYw" key="pojazd"/>
     <details xmi:id="_d1IhYJrIEeuG7NKZ46hTYw" key="koła"/>
     </eAnnotations>

   - Atrybut
     Przykładowy publiczny atrybut „nowy” z wielokrotnością 1 oraz domyślną wartością 10:
     <ownedAttribute xmi:id="_0_xpMJCaEeuG94qDkPAb9w" name="nowy"
     visibility="public">
     <type xmi:type="uml:PrimitiveType"
     href="pathmap://UML_LIBRARIES/JavaPrimitiveTypes.library.uml#int"/>
     <upperValue xmi:type="uml:LiteralUnlimitedNatural"
     xmi:id="_P6o3sJCcEeu5Cc4IUnM0CA" value="1"/>
     <lowerValue xmi:type="uml:LiteralInteger"
     xmi:id="_P6oQoJCcEeu5Cc4IUnM0CA" value="1"/>
     <defaultValue xmi:type="uml:LiteralInteger"
     xmi:id="_97w3UJCbEeu5Cc4IUnM0CA" value="10"/>
     </ownedAttribute>

   - Metoda
     Przykładowa metoda edytuj:
     <ownedOperation xmi:id="_BL8OoJCcEeu5Cc4IUnM0CA"
     name="edytuj"/>

   - Parametr metody
     Przykładowy parametr osoba o wielokrotności 1:

<ownedParameter xmi:id="_KYHqUJCcEeu5Cc4IUnM0CA" name="osoba" type="_QtWKMJCgEeuacZWeASWGOQ">
<upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="_NemTEJCcEeu5Cc4IUnM0CA" value="1"/>
<lowerValue xmi:type="uml:LiteralInteger" xmi:id="_NelE8JCcEeu5Cc4IUnM0CA" value="1"/>
</ownedParameter>

- Asocjacja
<ownedAttribute xmi:id="_IARdoJCmEeuacZWeASWGOQ" name="osoba" visibility="private" type="_QtWKMJCgEeuacZWeASWGOQ" association="_IAPBYJCmEeuacZWeASWGOQ"/>

- Generalizacja
<generalization xmi:id="_x0RyAJCkEeuacZWeASWGOQ" general="_jnVLIJCkEeuacZWeASWGOQ"/>

- Agregacja
<ownedAttribute xmi:id="_W3llQJCoEeuICJsfN6k0VQ" name="preferencja" visibility="private" type="_JW2kIJCoEeuICJsfN6k0VQ" aggregation="shared" association="_W3k-MJCoEeuICJsfN6k0VQ"/>

- Kompozycja
<ownedAttribute xmi:id="_lKfj0JCgEeuacZWeASWGOQ" name="właściciel" visibility="private" type="_QtWKMJCgEeuacZWeASWGOQ" aggregation="composite" association="_lKbSYJCgEeuacZWeASWGOQ">

2. Zakres projektu
Projekt zakłada scalanie 2 plików emx w 1 spójny plik. Użytkownik wybiera dwa pliki i następnie sposób scalania plików. Można utworzyć nowy plik emx lub wynikiem działania programu podmienić wybrany plik emx. Użytkownik może zdecydować czy chce rozwiązywać problemy w trakcie działania programu czy aby program utworzył plik tekstowy opisujący wszystkie kolizje oraz napotkane błędy. Program skupia się na scalaniu modelu. Warstwa graficzna diagramu klas, jeśli istnieje, jest kopiowana bez zmian w celu uniknięcia potencjalnych konfliktów. Z tego powodu opis gramatyki nie uwzględnia szczegółów dotyczących sposobu opisywania warstwy graficznej w plikach emx.

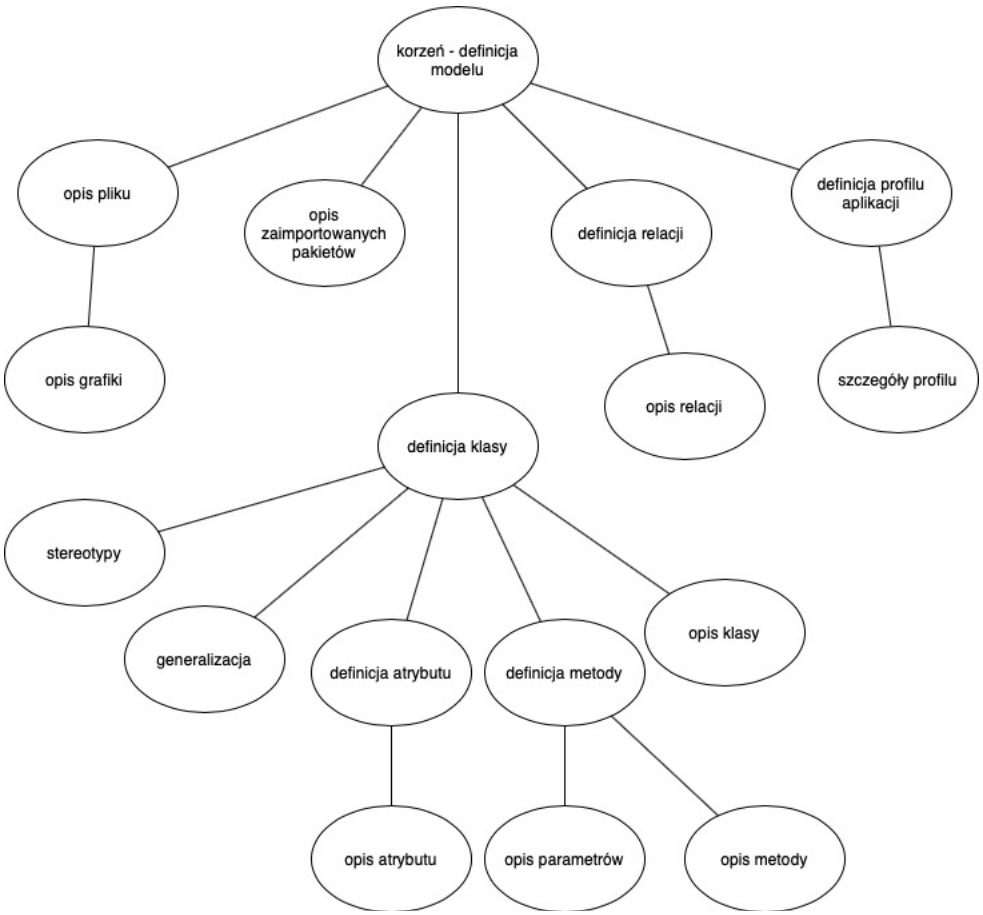
3. Składnia
3.1. Rozpoznawane tokeny:
   - <
   - </
   - >

- />
- uml:Model
- xmi:id
- name
- eAnnotations
- source
- contents
- xmi:type
- type
- element
- xsi:nil
- packageImport
- importedPackage
- href
- packagedElement
- generalization
- general
- ownedAttribute
- visibility
- type
- upperValue
- value
- lowerValue
- defaultValue
- aggregation
- association
- ownedOperation
- ownedParameter
- isStatic
- memberEnd
- ownedEnd
- profileApplication
- references
- appliedProfile
- xmlns:notation
- children
- element
- target
- details
- key
- isLeaf
- isOrdered
- isReadOnly
- isDerived
- isDerivedUnion

- isQuery
- direction
- isUnique
- isAbstract
- EOF

3.2 Budowa drzewa

Drzewo będzie zbudowane z głównych elementów pliku emx wyszczególnionych w podpunkcie 10. Drzewo skupiać się będzie głównie na opisie klas i wszystkich jej podległych elementów. Elementy wykrywane poprzez tokeny, na przykład: widoczności atrybutów, typy parametrów czy wartości domyślne będą przechowywane w osobnych klasach odpowiadających obiektom w pliku emx. Węzłami w drzewie będą: definicje modelu, klas, atrybutów i operacji, opisy pliku, zaimportowanych pakietów i relacji.



3.3 EBNF

number = "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" | "0";
letter = "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z" | "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z" ;
symbol = number | letter | ":" | "/" | "." | "_" | "#" | "-" | "?";
boolean = "true" | "false";

```
id = '"',("_" | letter | number), { number | letter },'"';
double id = '"',("_" | letter | number), { number | letter }, " ", ("_" | letter | number),
{ number | letter }, '"';
path = '"', symbol, {symbol}, '"';

emx = '<?xml version="1.0" encoding="UTF-8"?>
<!--xtools2_universal_type_manager-->
<!--Rational Software Architect Designer 9.7.0-->
<?com.ibm.xtools.emf.core.signature <signature
id="com.ibm.xtools.mmi.ui.signatures.diagram" version="7.0.0"><feature
description="" name="Rational Modeling Platform (com.ibm.xtools.rmp)" url=""
version="7.0.0"/></signature>?>
<?com.ibm.xtools.emf.core.signature <signature id="com.ibm.xtools.uml.msl.model"
version="7.0.0"><feature description="" name="com.ibm.xtools.ruml.feature" url=""
version="7.0.0"/></signature>?>', model;


model = "<uml:Model", model description ,">", model contents, "</uml:Model>";
model description = 'xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore"
xmlns:uml="http://www.eclipse.org/uml2/3.0.0/UML"
xmlns:umlnotation="http://www.ibm.com/xtools/1.5.3/Umlnotation" xmi:id=', id, '
name=', id;
model contents = file description,{ package import },{ packaged element },{ profile
application };

file description = "<eAnnotations xmi:id=", id, ' source="uml2.diagrams">', file name,
"</eAnnotations>";
file name = "<contents xmi:type="umlnotation:UMLDiagram" xmi:id=", id, '
type="Class" name=', id, ">", graphic description, "</contents>";
graphic description. = children description, '<element xsi:nil="true"/>', edges
description;
children description = ...; //nie interesuje nas
edges description = ...; //nie interesuje nas

package import = "<packageImport xmi:id=", id, ">",  package, "</packageImport>";
package = '<importedPackage xmi:type="uml:Model" href=', path, "/>";

packaged element = "<packagedElement xmi:type=", ( class | association ),
"</packagedElement>";
class = '"uml:Class" xmi:id=', id, " name=", id, visibility, ['isLeaf="true"'],
['isAbstract="true"'], ">", [ stereotype ], [ generalization ], {attribute}, {operation};

stereotype = "<eAnnotations xmi:id=", id, " source=", path, ">", stereotype
description, {stereotype description}, "</eAnnotations>";
```

```
stereotype description = "<details xmi:id=", id, " key=", id, "/>";

generalization = "<generalization xmi:id=", id, " general=", id, "/>";

attribute = "<ownedAttribute xmi:id=", id, " name=", id, attribute parameters, ( "/>" |
attribute description );
attribute description = ">", [type], [limit], [default value], "</ownedAttribute>";
attribute parameters = visibility, ['isLeaf="true"'], ['isStatic="true"'],
['isOrdered="true"'], ['isReadOnly="true"'], ['isDerived="true"'],
['isDerivedUnion="true"'], [short type], [association type];
visibility = " visibility=", visibility type;
visibility type = "public" | "private" | "protected" | "package";
short type = " type=", id;
association type = [ 'aggregation="composite"' | 'aggregation="shared"' ],
"association=", id;
type = "<type xmi:type=", id, "href=", path,"/>";
limit = upper limit, lower limit;
upper limit = "<upperValue xmi:type=", id, " xmi:id=", id, [" value=", ("1" | "*")] ,"/>";
lower limit = "<lowerValue xmi:type=", id, " xmi:id=", id, [" value=", ("1" | "*")] ,"/>";
default value = "<defaultValue xmi:type=", id, " xmi:id=", id, " value=", id, ( default
value type | "/>" );
default value type = type, "</defaultValue>";

operation = "<ownedOperation xmi:id=", id, " name=", id, [ operation parameters ],
("/>" | parameter );
operation parameters = visibility, ['isLeaf="true"'], ['isStatic="true"'],
['isQuery="true"'];
parameter = owned parameter, {owned parameter}, "</ownedOperation>";
owned parameter = "<ownedParameter xmi:id=", id, " name=", id, [owned parameter
parameters], ("/>" | owned parameter description);
owned parameter parameters = short type, ['isOrdered="true"'], ['isUnique="false"'],
[parameter direction];
parameter direction = " direction=", direction type;
direction type = "return" | "out" | "inout";
owned parameter description = ">", [type], [upper limit], [lower limit], [default
value], "</ownedParameter>";

association = '"uml:Association"', " xmi:id=", id, " memberEnd=", double id, ">",
owned end;
owned end = "<ownedEnd xmi:id=", id, " name=", id, visibility, short type,
"association=", id, ">", upper limit, lower limit, "</ownedEnd>";

profile application = "<profileApplication xmi:id=", id, ">", eannotation,
"<appliedProfile href=", path, "/>", "</profileApplication>";
eannotation = "<eAnnotations xmi:id=", id, " source=", path, ">", '<references
xmi:type="ecore:EPackage" href=', path, "/>", "</eAnnotations>";
```

4. Obsługa błędów
   Błąd składniowy w podanych plikach emx jest traktowany jako błąd krytyczny, ponieważ zakładam, że użytkownik nie ingerował w wygenerowane przez RSAD pliki, więc taki błąd przerywa działanie programu.
   Konflikt atrybutów/metod/parametrów/relacji pomiędzy klasami jest pozostawiony użytkownikowi do rozstrzygnięcia lub pozostaje w takiej formie w jakiej występuje w pliku głównym (1 z 2 plików), a informacja o takim zdarzeniu jest zapisywana do pliku tekstowego.

5. Sposób uruchomienia
   Aplikacja okienkowa wykonana w Javie pozwalająca na wybór 2 plików z dysku. Dodatkowo umożliwia ona wybór trybu działania programu i sposób obsługi sytuacji konfliktowych.

6. Sposób realizacji
   Aplikacje: lekser, parser oraz łącząca pliki emx zostaną napisane w Javie. Diagramy klas zapisane w pliku emx są tworzone przez użytkownika w programie RSAD.

7. Sposób testowania
   Testowanie odbywać się będzie na zasadzie wielu testów jednostkowych skupiających się na różnych poziomach skomplikowania. Zaczynając od 2 identycznych plików, następnie takich różniących się atrybutami i metodami poprzez pliki różniące się praktycznie wszystkimi klasami i relacjami. Każdy kolejny test skupiał się będzie na sprawdzeniu działania programu z większą ilością konfliktów.

8. Wymagania funkcjonalne
   - Program tworzy 1 spójny pliku emx z 2 plików.
   - Program rozróżnia klasy, atrybuty i metody.
   - Program zapamiętuje stereotypy klas
   - Program rozróżnia 5 typów relacji pomiędzy klasami
   - Program rozróżnia 4 typy parametrów metod: in, out, return oraz inout.
   - Program rozróżnia fragmenty odpowiedzialne za model i grafikę.

9. Wymaganie niefunkcjonalne
   - Program tworzy plik emx, który jest bez błędów wczytywany przez RSAD.
   - Program jest w stanie poradzić sobie z sytuacjami konfliktowymi/błędami niekrytycznymi.

10. Krótki opis budowy pliku emx
    - Opis środowiska, kodowanie, wersja programu
    - Rozpoczęcie opisu modelu
    - Opis pliku
      - Opis grafiki modelu
    - Opis zaimportowanych pakietów typów danych
    - Opis klas

    - Stereotypy
    - Generalizacja
    - Atrybuty
        - Typ oraz rodzaj relacji
        - Górny limit
        - Dolny limit
        - Wartość domyślna
    - Metody
        - Parametry
            - Górny limit
            - Dolny limit
- Opis relacji pomiędzy klasami
- Opis profili aplikacji
- Zakończenie opisu modelu