POLSKO-JAPOŃSKA WYŻSZA SZKOŁA TECHNIK KOMPUTEROWYCH

Wydział Informatyki

Katedra Systemów Informacyjnych

Inżynieria Oprogramowania i Baz Danych

Łukasz Ziemiańczyk

Nr albumu 3265

EduCASE – Narzędzie CASE wspomagające edukację

Praca magisterska

dr Mariusz Trzaska

Warszawa, wrzesień, 2009

Streszczenie

Praca dotyczy komputerowego wspomagania projektowania oprogramowania, jak również wspierania edukacji poprzez zastosowanie odpowiedniego oprogramowania. Istnieje szereg rozwiązań dotyczących jednego lub drugiego zagadnienia, lecz rzadko zdarza się poruszać je jednocześnie. Komputerowe wspomaganie projektowania oprogramowania realizowane jest między innymi przez bogatą rodzinę narzędzi CASE. Podczas przeglądania aplikacji tego typu dostępnych obecnie na rynku nie znalazłem takiej, która przybliżyłaby użytkownikowi język UML. Jednakże sam temat nauczania wspomaganego rozwiązaniami informatycznymi jest dość powszechny. Przykładu nie trzeba daleko szukać. Wystarczy wspomnieć o możliwości studiowania w trybie internetowym na uczelni PJWSTK. Oba zagadnienia są ciekawe i współczesne. Efektem przemyśleń na ich temat jest prototyp narzędzia EduCASE – narzędzie CASE wspomagające edukację. Ponadto warto w aplikacji zwrócić uwagę na wykorzystanie rozszerzalnego języka XML jako formatu zapisywania projektu oraz na generowanie kodu, które odnosi się do problemu RAD (Rapid Application Development) - błyskawicznego tworzenia oprogramowania – terminu bardzo ważnego w dzisiejszym świecie programistycznym.

Spis treści

1. Wstęp

1.1. System CASE wspomagający edukację

Narzędzia wspomagające edukację to bardzo ciekawe zagadnienie. W przypadku tej pracy zostanie ono przeanalizowane pod kątem przydatności aplikacji nauczającej jak poprawnie budować diagram klas. Jest to dopiero wstęp do stworzenia programu bardziej rozbudowanego obsługującego wszystkie typy modeli UML. Osobiste doświadczenia polegały na zaczerpnięciu wiedzy teoretycznej z podręcznika, a następnie wykorzystania jej na zajęciach w trakcie budowania diagramów na podstawie treści zadania. W ten sposób zrodził się pomysł stworzenia systemu, który będzie wirtualnie zastępował cały proces. Tak więc w EduCASE znajdziemy zarówno opis elementów diagramu klas, jak i możliwość stworzenia, podjęcia i sprawdzenia lekcji. Dla jaśniejszego zaprezentowania tej kwestii, algorytm sprawdzania lekcji został opisany w rozdziale Przyjęte rozwiązania implementacyjne. Jednocześnie nie można zapominać o aspekcie CASE. Istnieje szereg funkcjonalności tego typu narzędzi, które pomagają analitykom i programistom przy tworzeniu projektu. Zostaną one poddane analizie. Zbadamy, które są najbardziej przydatne oraz czy można wyprodukować nową jakość, mimo wielu bardzo dobrze dopracowanych rozwiązań dostępnych na rynku.

1.2. Cel pracy

Celem pracy jest zaprezentowanie rozwiązania posiadającego pewną funkcjonalność, której wykorzystanie wymaga od użytkownika wiedzy z zakresu projektowania oprogramowania w oparciu o diagram klas UML. Jednocześnie EduCASE samo w sobie ma za zadanie udostępniać tą wiedzę i wspomagać samodzielne uczenie się osoby korzystającej z programu.

1.3. Rezultaty pracy

Rezultatem pracy jest między innymi zbadanie, które z funkcjonalności narzędzi CASE są najbardziej pożyteczne i przyczyniają się do szybkiego tworzenia oprogramowania. Powstanie program i strona internetowa, dzięki którym możliwe będzie zdobycie wiedzy niezbędnej do tworzenia modeli UML i wykorzystanie tych umiejętności w praktyce. Aplikacja może być przydatna np. na zajęciach studenckich nawiązujących do projektowania rozwiązań informatycznych lub w firmach jako narzędzie używane przez analityków i programistów w początkowej fazie projektu.

1.4. Organizacja pracy

Praca zorganizowana jest w sposób, który powinien odzwierciedlać proces rozpatrywania pewnego problemu. Na początku zostały opisane rozwiązania dostępne obecnie na rynku. Ujawniono ich wady i zalety, aby przejść następnie do idei perfekcyjnego oprogramowania danego typu. W tym momencie następuje bilans zysków i strat, czego efektem jest stworzenie prototypu opisanego w pracy z wyszczególnieniem ciekawych funkcjonalności oraz planów rozwijania aplikacji.

2. Stan sztuki

2.1. Narzędzia CASE

2.1.1 Microsoft Visio

Zostanie tutaj opisana wersja Microsoft Office Visio 2007. Obecnie aplikacja wchodzi w skład pakietu Office, jednak przed przejęciem przez Microsoft była rozwijana samodzielnie przez Visio Corporation.

System ten jest bardzo uniwersalny. Pozwala stworzyć diagramy używane w wielu dziedzinach. Poza modelami UML można tam także narysować diagramy biznesowe, jak wykres organizacji, wykresy czasowe, np. Gantta, mapę strony internetowej i wiele innych. Uniwersalność ta niesie ze sobą pewne wady oprogramowania. Microsoft Visio nie jest typowym narzędziem CASE i nie zawiera specyficznych dla niego funkcjonalności.

Warto wymienić zaletę Visio, jaką jest dobra współpraca z innymi programami firmy Microsoft. Źródłem danych może być np. skoroszyt Excel-a, czy też baza danych Access-a lub SQL Server. Format wyjściowy dla plików to poza typami stworzonymi na potrzeby Visio także różne pliki graficzne (gif, bmp, jpg, png, tif), pliki dla aplikacji AutoCAD, pliki do zapisu grafiki wektorowej (svg, svgz), strona internetowa (html, htm), itd. Jak pozostałe składniki pakietu Office, jest lokalizowany na wiele języków oraz posiada rozbudowaną pomoc. Można pobrać darmową 60-dniową wersję programu.

Dawniej istniała możliwość przetestowania tego produktu przez samą przeglądarkę internetową. Niestety dzisiaj na stronie Microsoft Office Online znajdujemy komunikat, iż wersja próbna online pakietu Office 2007 nie jest już dostępna.

Visio to przede wszystkim aplikacja do szybkiego i łatwego rysowania diagramów. Pozwala stworzyć:

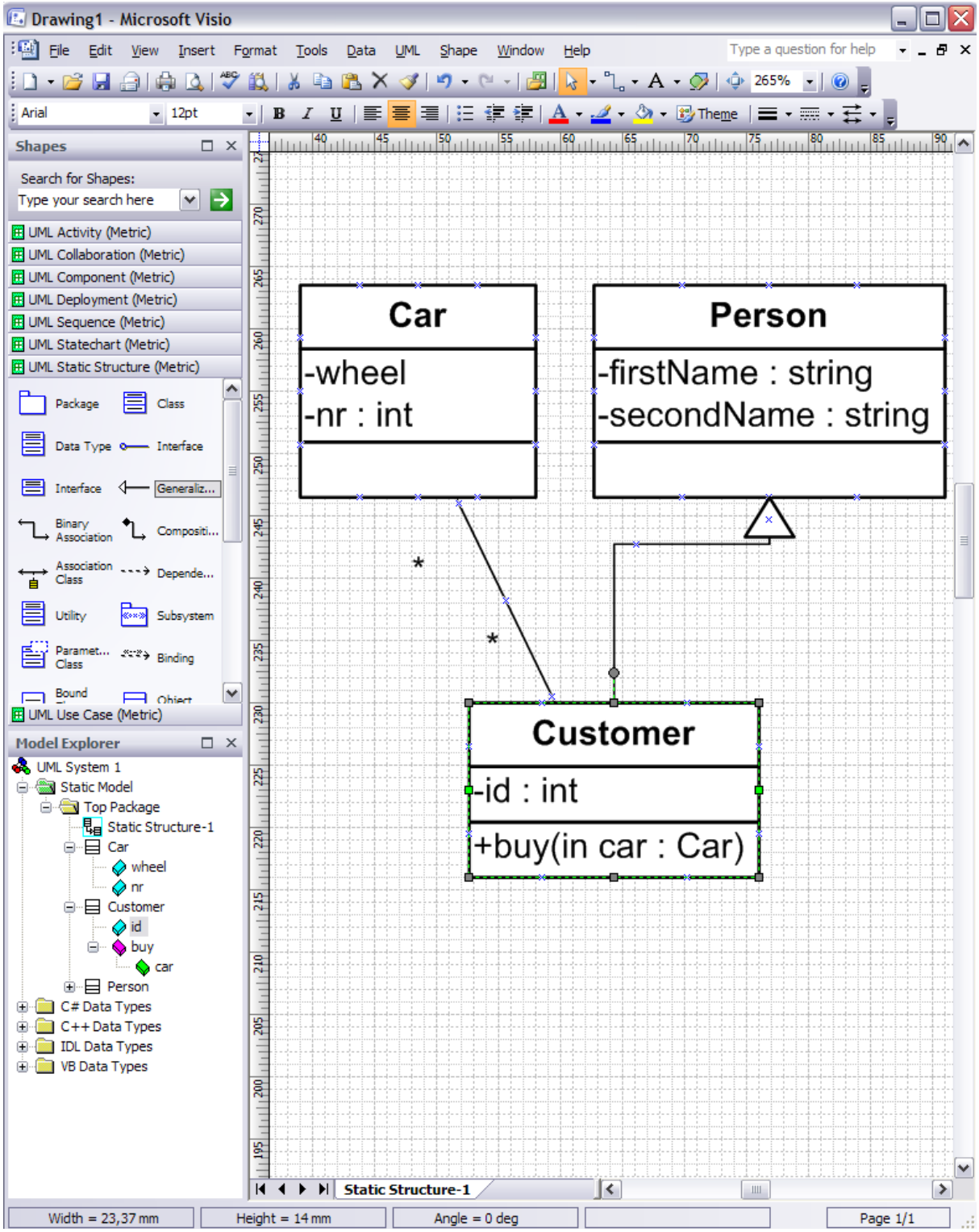
- diagram aktywności

- diagram komponentów
- diagram wdrożenia
- diagram sekwencji
- diagram klas
- diagram przypadków użycia
- diagram kolaboracji
- diagram maszyny stanów

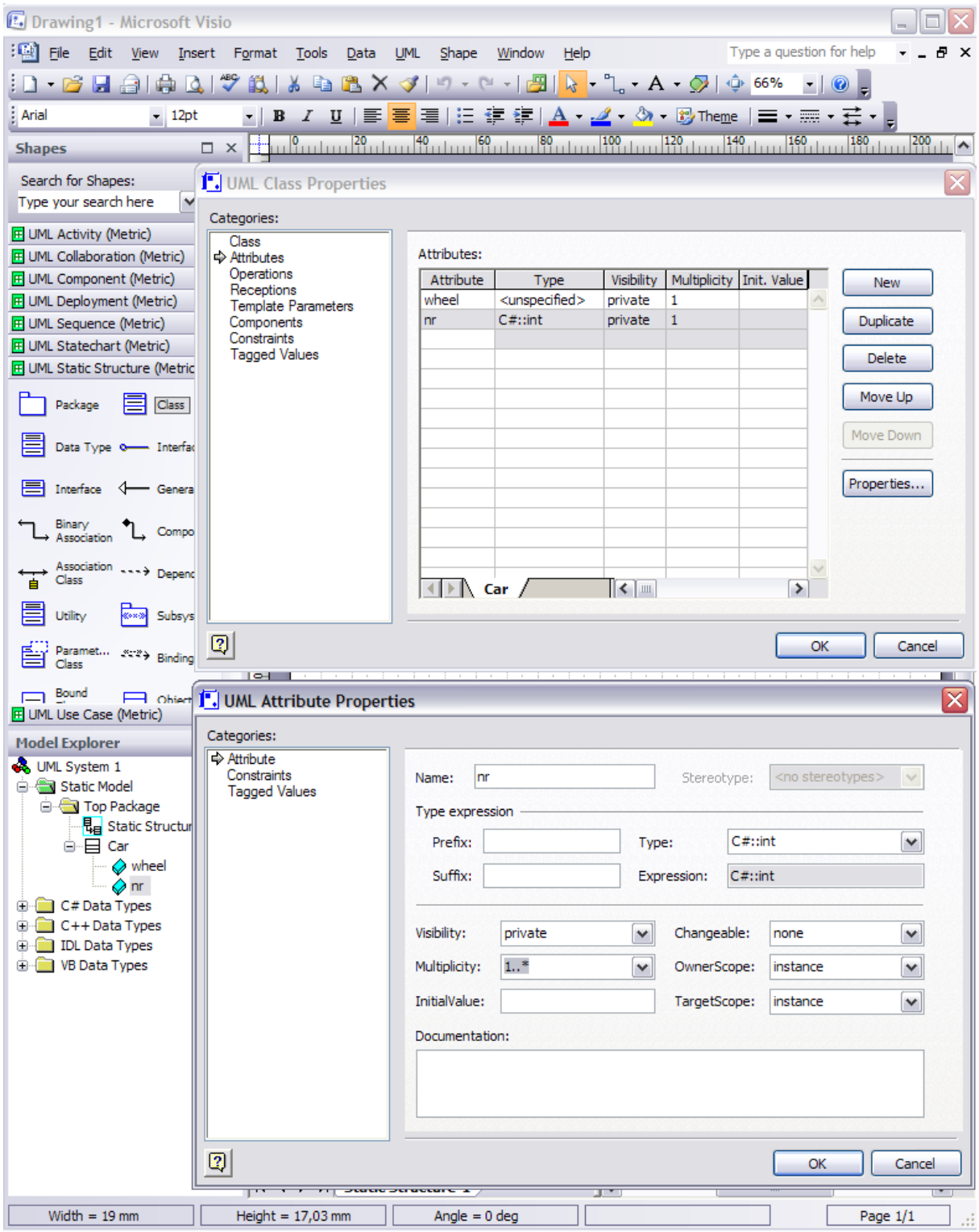
Wspiera typy danych znane z języków C#, C++, IDL, VB.

Produktowi temu brakuje kilku ważnych funkcjonalności typowych dla narzędzi CASE. Wśród nich w szczególności można wyróżnić brak opcji wygenerowania jakiegokolwiek kodu źródłowego. Nie ma także inżynierii odwrotnej, która pozwoliłaby na uzyskanie modelu z gotowego kodu. Nie ma opcji pozwalających na tworzenie składnej dokumentacji, czy też eksportowania projektu jako plik XMI. W programie i na stronie internetowej znajdziemy rozbudowaną pomoc oraz kursy na http://office.microsoft.com/pl-pl/training/CR101109221045.aspx, lecz odnosi się to jedynie do Visio. Brak objaśnień dotyczących notacji UML.

Przykładowe zrzuty ekranów:


Drawing1 - Microsoft Visio
File Edit View Insert Format Tools Data UML Shape Window Help
Shapes
Search for Shapes:
Type your search here
UML Activity (Metric)
UML Collaboration (Metric)
UML Component (Metric)
UML Deployment (Metric)
UML Sequence (Metric)
UML Statechart (Metric)
UML Static Structure (Metric)
UML Use Case (Metric)
Model Explorer
UML System 1
Static Model
Top Package
Static Structure-1
Car
wheel
nr
Customer
id
buy
car
Person
C# Data Types
C++ Data Types
IDL Data Types
VB Data Types
Car
-wheel
-nr : int
Person
-firstName : string
-secondName : string
Customer
-id : int
+buy(in car : Car)
*
*
Static Structure-1
Width = 23,37 mm
Height = 14 mm
Angle = 0 deg
Page 1/1

Rysunek 1:


Drawing1 - Microsoft Visio
File Edit View Insert Format Tools Data UML Shape Window Help
Shapes
Search for Shapes:
Type your search here
UML Activity (Metric)
UML Collaboration (Metric)
UML Component (Metric)
UML Deployment (Metric)
UML Sequence (Metric)
UML Statechart (Metric)
UML Static Structure (Metric
UML Use Case (Metric)
Model Explorer
UML System 1
Static Model
Top Package
Car
wheel
nr
C# Data Types
C++ Data Types
IDL Data Types
VB Data Types
UML Class Properties
Categories:
Class
Attributes
Operations
Receptions
Template Parameters
Components
Constraints
Tagged Values
Attributes:
Attribute Type Visibility Multiplicity Init. Value
wheel <unspecified> private 1
nr C#::int private 1
New
Duplicate
Delete
Move Up
Move Down
Properties...
OK
Cancel
UML Attribute Properties
Categories:
Attribute
Constraints
Tagged Values
Name: nr
Stereotype: <no stereotypes>
Type expression
Prefix:
Type: C#::int
Suffix:
Expression: C#::int
Visibility: private
Changeable: none
Multiplicity: 1..*
OwnerScope: instance
InitialValue:
TargetScope: instance
Documentation:
OK
Cancel
Width = 19 mm
Height = 17,03 mm
Angle = 0 deg
Page 1/1

Rysunek 2:

Strona domowa: http://office.microsoft.com/pl-pl/visio/default.aspx

2.1.2 Poseidon for UML

Wywodzi się z ArgoUML. Jest to jego rozwinięty, komercyjny następca. Istnieją różne wersje Poseidon for UML. Zostanie on omówiony ogólnie bez wyszczególniania różnic między edycjami Community, Standard, Professional i Embedded. Za jego pomocą można stworzyć następujące diagramy:

- diagram aktywności
- diagram komponentów
- diagram wdrożenia
- diagram sekwencji
- diagram klas
- diagram przypadków użycia
- diagram kolaboracji
- diagram maszyny stanów

Poseidon for UML wspomaga analizę, projektowanie oprogramowania, a także tworzenie dokumentacji. Inżynieria odwrotna pozwala na uzyskanie wizualnego modelu z istniejącego kodu źródłowego. Istnieje również możliwość edycji kodu w trakcie kreowania diagramu. Aplikacja wspiera wiele różnych formatów, w tym np. UMLdoc i XMI. Możliwa jest generacja kodu w wielu językach programowania, eksport do różnego typu plików graficznych oraz integracja ze środowiskiem Eclipse.

Jest zlokalizowany dla języków: angielski, niemiecki, rosyjski, francuski, hiszpański oraz chiński.

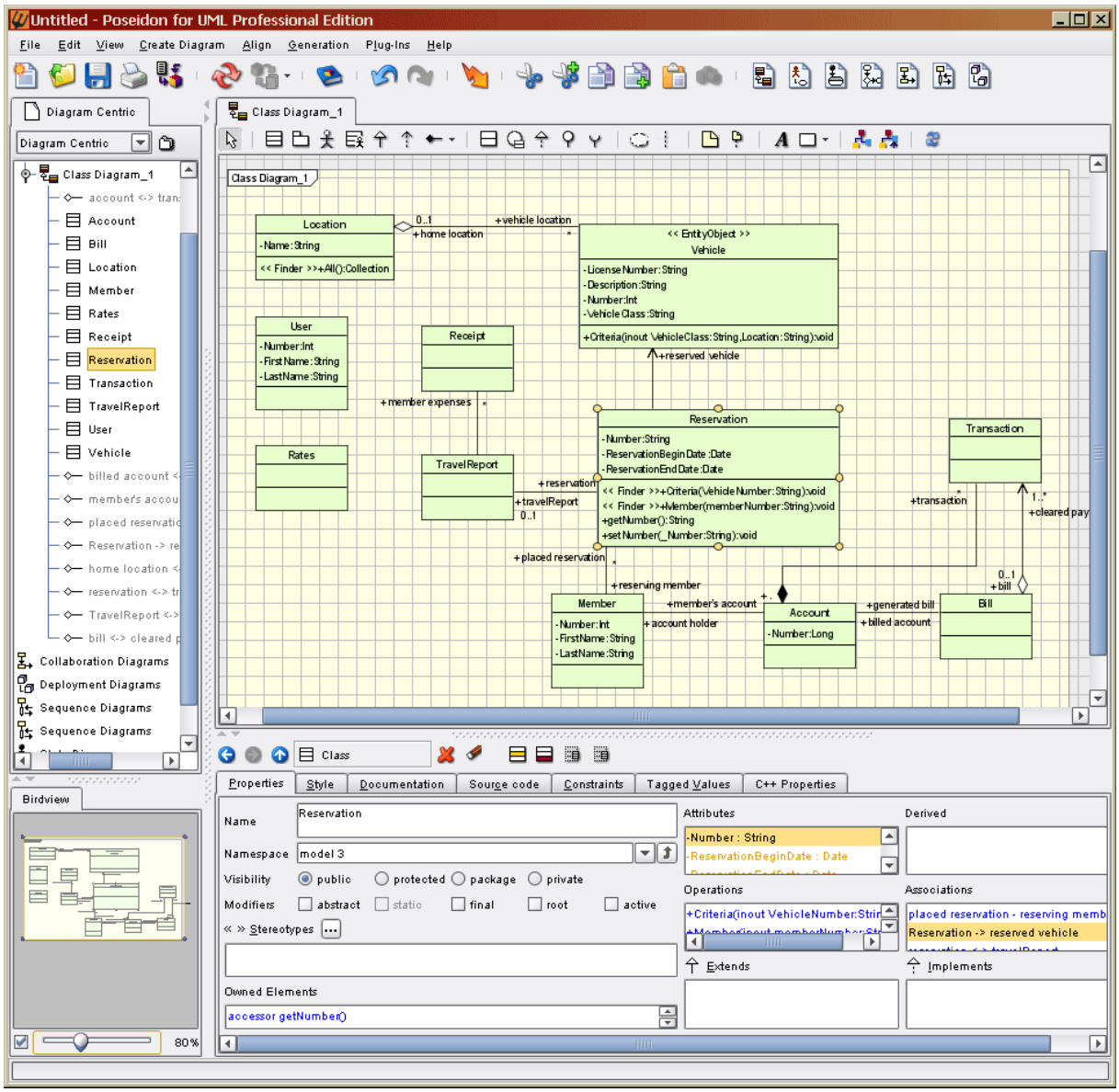
Używałem tego narzędzia wraz z pozostałymi studentami z grupy Inżynierii Oprogramowania w trakcie pracy inżynierskiej i zostało ono uznane jako bardzo dobra aplikacja, której podstawową zaletą jest łatwe i intuicyjne tworzenie diagramów w technologii UML.

Największą wadą aplikacji Poseidon jest jej cena. Oscyluje ona w okolicach 250 dolarów amerykańskich za wersję Standard Edition za licencję dla jednego użytkownika do ponad 2500 za Embedded Edition. Alternatywnym rozwiązaniem jest wykupienie okresowej subskrypcji Community Edition, która miesięcznie kosztuje 6 dolarów. Im dłuższy czas subskrypcji tym cena jest bardziej opłacalna. Na stronie http://www.gentleware.com/eval.html po wypełnieniu formularza można pobrać klucze, które umożliwiają darmowe korzystanie z programu przez 30 dni.

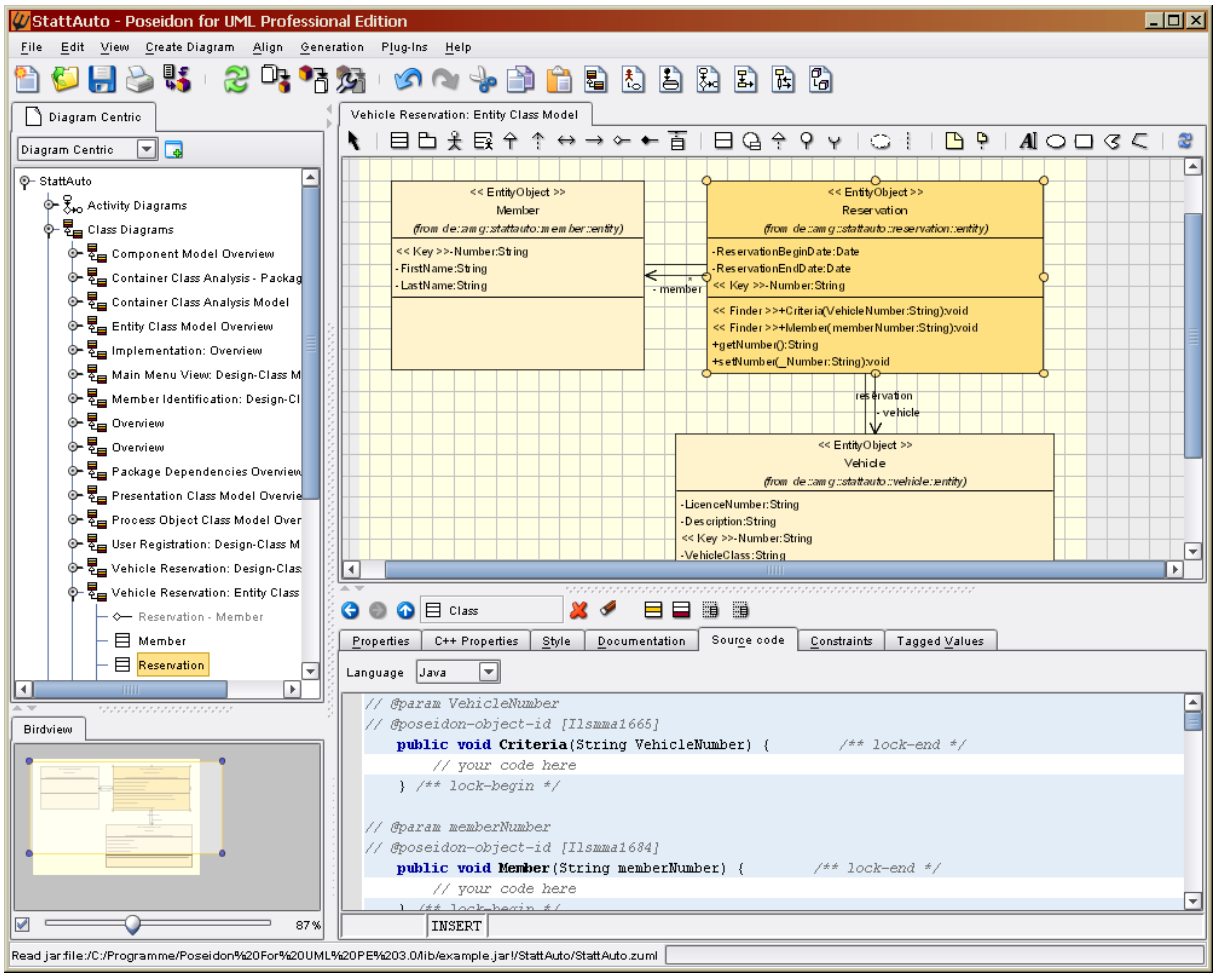
Obsługa programu jest intuicyjna. Poseidon został stworzony przede wszystkim dla profesjonalistów, którzy znają notację UML. Brak jakiejkolwiek funkcji nauczania tego języka. Studentów może jednakże zainteresować informacja o zniżce. Uprzywilejowanie polega na możliwości zakupu Professional Edition za 89 dolarów amerykańskich.

Strona internetowa http://www.gentleware.com/products.html skupia się ściśle na opisie Poseidon for UML dla ewentualnych klientów.

Przykładowe zrzuty ekranów:


Untitled - Poseidon for UML Professional Edition
File Edit View Create Diagram Align Generation Plug-Ins Help
Diagram Centric
Class Diagram_1
Location
-Name:String
<< Finder >>+All():Collection
0..1
+home location
+vehicle location
<< EntityObject >>
Vehicle
-License Number:String
-Description:String
-Number:Int
-Vehicle Class:String
+Criteria(inout VehicleClass:String,Location:String):void
+reserved vehicle
User
-Number:Int
-First Name:String
-Last Name:String
Receipt
+member expenses
Rates
TravelReport
Reservation
-Number:String
-ReservationBegin Date:Date
-ReservationEnd Date:Date
<< Finder >>+Criteria(VehicleNumber:String):void
<< Finder >>+Member(memberNumber:String):void
+getNumber():String
+setNumber(_Number:String):void
+reservation
+travelReport
+placed reservation
Transaction
+transaction
+cleared pay
Member
-Number:Int
-FirstName:String
-LastName:String
+reserving member
+member's account
+account holder
Account
-Number:Long
+generated bill
+billed account
Bill
+bill
Collaboration Diagrams
Deployment Diagrams
Sequence Diagrams
Sequence Diagrams
Birdview
80%
Properties Style Documentation Source code Constraints Tagged Values C++ Properties
Name
Reservation
Namespace
model 3
Visibility
public protected package private
Modifiers
abstract static final root active
« » Stereotypes
Owned Elements
accessor getNumber()
Attributes
-Number : String
-ReservationBeginDate : Date
Operations
+Criteria(inout VehicleNumber:Strin
Derived
Associations
placed reservation - reserving memb
Reservation -> reserved vehicle
Extends
Implements

Rysunek 3:



Rysunek 4:

2.2. Narzędzia wspomagające edukację

Wspomaganie nauczania w aplikacjach może odnosić się do dwóch aspektów. Z jednej strony chodzi o pomoc programu. Większość produktów posiada taką wbudowaną funkcjonalność lub odnośnik do strony internetowej. Wyjaśnione tam jest jak działają poszczególne opcje. W tej pracy skupiamy się jednak na czymś więcej. System powinien umożliwiać edukację z zakresu wiedzy, którego sam dotyczy.

2.2.1 Tutoriale W3C

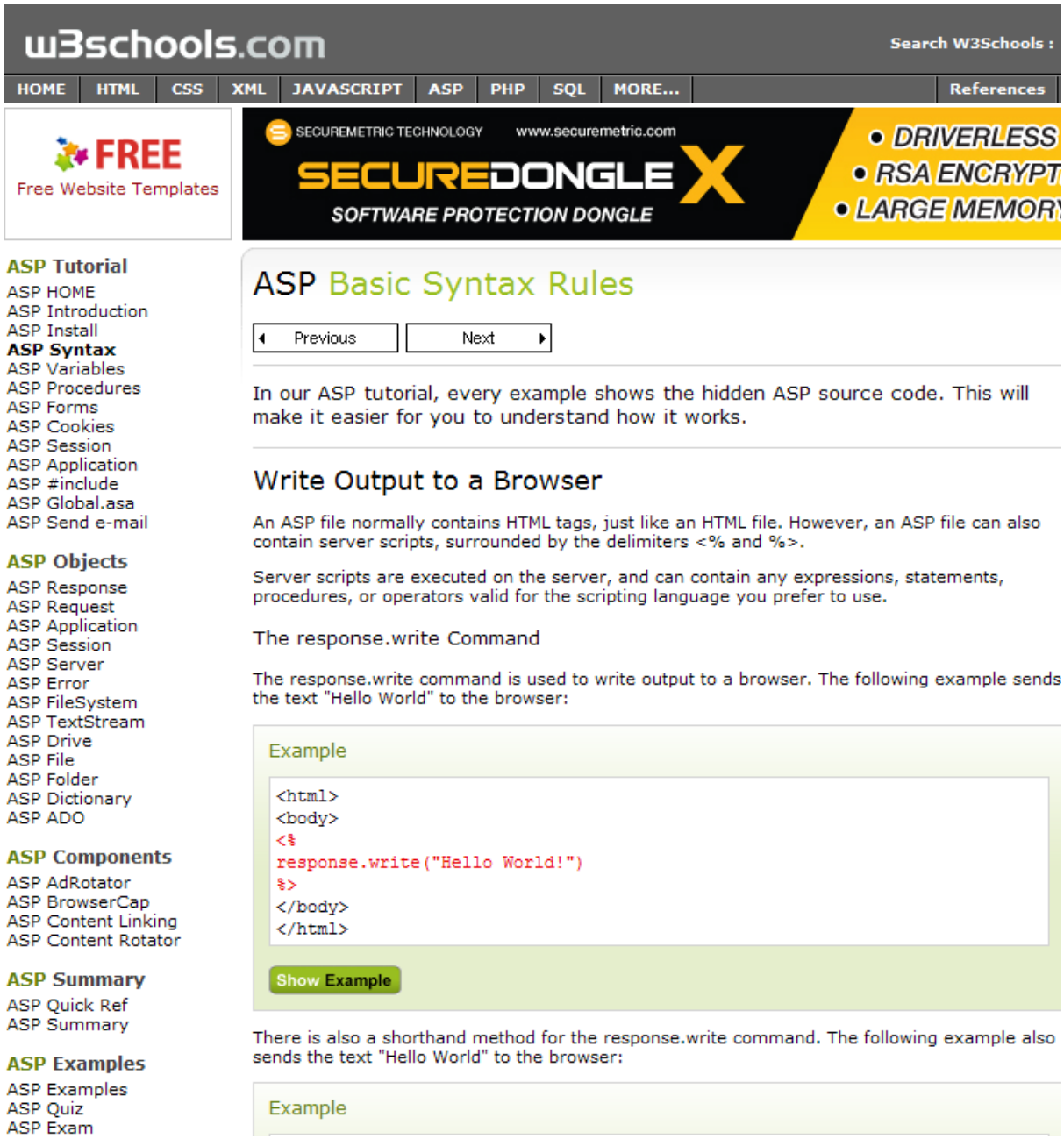
Jedną z bardzo popularnych witryn internetowych jest strona organizacji W3C. Jest to następstwem ogromnej liczby zamieszczonych tam tutoriali. Można wyróżnić działy, których tematyka skupia się wokół:

- HTML
- XML
- skrypty działające po stronie przeglądarki internetowej
- skrypty działające po stronie serwera
- multimedia
- ogólnie web

Tutoriale mają postać zwięzłego opisu kolejno dla każdej ważnej własności danej technologii. W razie potrzeby umieszczany jest przykładowy kod źródłowy, grafika prezentująca diagram, czy tabela wypisująca pewne właściwości. Wszystko po to, by zaprezentować użytkownikowi wiedzę w najbardziej przystępny sposób. Jest to przede wszystkim pomocne dla osób początkujących, jednakże także doświadczony developer może korzystać z w3schools jako kompendium wiedzy. Każda technologia zawiera działy, w których odwiedzający może znaleźć przykłady użycia oraz podjąć test sprawdzający znajomość informacji zamieszczonych na portalu.

Strona domowa: http://www.w3schools.com/

Przykładowy zrzut ekranu:



Rysunek 5:

3. Języki, metodyki, narzędzia

3.1. Java SE

Java SE – Java Standard Edition jest obiektowym językiem programowania, który został stworzony przez firmę Sun Microsystems (strona domowa: http://www.sun.com/). Do uruchamiania aplikacji napisanych w tym języku niezbędna jest JRE – Java Runtime Environment – maszyna wirtualna Javy. JRE interpretuje kod pośredni niezależnie od procesora czy systemu operacyjnego. Dzięki temu raz napisany program może być dowolnie uruchamiany na dowolnym systemie Windows, Linux lub inny z zainstalowaną wirtualną maszyną Javy. Wadą aplikacji stworzonych w tym języku jest ich mniejsza szybkość od aplikacji kompilowanych do kodu maszynowego. Aby programować w języku Java należy zainstalować JDK – Java Development Kit – pakiet zawierający między innymi źródła klas standardowych języka Java, kompilator oraz debuger. Wersja Javy była uaktualnia w trakcie pisania pracy. Ostateczną wersja to 1.6.0_06. Aby mieć pewność poprawnego uruchomienia aplikacji i skorzystania z wszystkich jej funkcji użytkownik powinien posiadać minimum wersję jre1.6.0_06 wirtualnej maszyny Javy.

Język ten został wybrany do celu implementacji aplikacji. Także kod generowany przez sam program jest kodem javowym, jednakże po drobnych modyfikacjach można z niego stworzyć kod w innym języku o czym zostanie wspomniane w rozdziale dotyczącym możliwych udoskonaleń aplikacji. Wybór Javy jest konsekwencją wielu zalet tego języka. Jest jednym z najbardziej popularnych języków na świecie. Największą wartość stanowi niezależność od architektury. Wirtualna maszyna interpretuje kod pośredni, który może być identyczny dla różnych procesorów i systemów operacyjnych. Twórcy dołożyli także starań, by język był niezawodny i bezpieczny. Kolejnym walorem Javy jest wspieranie jej przez solidne, darmowe oprogramowanie jak Eclipse czy NetBeans IDE.

Główne biblioteki standardowe wykorzystane przy realizacji pracy:

- javax.swing – graficzny interfejs użytkownika, obsługa zdarzeń
- javax.xml – tworzenie, parsowanie plików xml
- java.util – korzystanie z właściwości kolekcji
- java.io – działanie na plikach
- java.awt – ustawianie kolorów, działanie na kursorze

3.2. UML

UML – Unified Modeling Language – Ujednolicony Język Modelowania, początkowo tworzony przez firmę Rational, obecnie produkt firmy OMG – Object Management Group (http://www.omg.org/). Notyfikacja UML jest praktycznie niezastąpiona podczas fazy analizy większego projektu informatycznego i jest w dzisiejszych czasach bardzo rozpowszechniona. Brak jest znaczącej konkurencji na tym polu. Za jej pomocą możliwe jest porozumiewanie się ludzi zajmujących różne stanowiska i pracujących w różnych firmach. Posiadając stosowną wiedzę menedżer jest w stanie wytłumaczyć analitykowi specyfikację aplikacji, a ten z kolei posiada ułatwienie w komunikacji z grupą programistów przedstawiając im diagram przypadków użycia, diagram klas lub inny.

Istnieją różne wersje UML. Najnowsza to 2.2. W pracy wykorzystana będzie wiedza na temat rodziny wersji 2.x. Aktualną specyfikację można znaleźć pod tym adresem: http://www.omg.org/spec/UML/2.2/. Model UML może być zapisywany w formacie XMI – XML Metadata Interchange. Jest to format firmy OMG do zapisywania metadanych w plikach typu XML.

Notyfikacja UML zawiera specyfikacje różnych rodzajów diagramów, które stosuje się stosownie do własnych potrzeb. Wyróżniamy między innymi diagramy:

- diagram przypadków użycia – prezentuje jakie czynności mogą wykonać aktorzy. Czynności to funkcjonalności omawianego systemu, natomiast aktorem może być dowolny użytkownik, system zewnętrzny czy też urządzenie.

- diagram klas – omawiany wraz z opisem konstrukcji diagramu w dalszej części tego rozdziału
- diagram stanów i aktywności – pierwszy opisuje stany w jakich kolejno może znaleźć się wybrany obiekt, natomiast drugiego używa się, gdy zachodzi potrzeba zaprezentowania działania procesu, którego uczestnikami jest wiele obiektów
- diagram sekwencji – za jego pomocą modeluje się dynamiczne aspekty systemu. Dokładnie prezentuje jak w perspektywie czasu kilka obiektów współdziała, aby zrealizować dane zadanie. Na tym diagramie widać chronologię komunikatów występujących w danej funkcji.
- i wiele innych...

W tej pracy skupimy się na diagramie klas. Najważniejsze elementy diagramu klas:

Klasa to opis zbioru obiektów, które mają takie same atrybuty, operacje, związki i znaczenie. Klasa posiada także nazwę.

Klasa konkretna - można tworzyć jej obiekty.

Klasa abstrakcyjna - nie można stworzyć żadnego obiektu takiej klasy. Służy jedynie do tworzenia nadklas.

Atrybut to nazwana właściwość klasy. Atrybut posiada specyfikator dostępu, nazwę, typ, ew. wartość domyślną.

Atrybut opcjonalny - może występować lub nie.

Atrybut powtarzalny - może mieć wiele wartości.

Atrybut pochodny - wyliczany z innych wartości.

Atrybut klasowy - taki sam dla wszystkich obiektów danej klasy.

Operacja - proces(metoda), które klasa potrafi wykonać. Operacje posiadają specyfikator dostępu, nazwę, parametry, typ wartości zwracanej.

Metoda obiektu - działa na atrybutach, asocjacjach itd. jednego obiektu.

Metoda klasowa - działa na ekstensji klasy.

Metoda abstrakcyjna - nie posiada implementacji. Można ją definiować jedynie w klasach abstrakcyjnych.

Przesłanianie metod - występuje jeżeli w nadklasie i podklasie mamy metody o jednakowej nazwie, typie zwracanej wartości, typach, liczbie i kolejności przyjmowanych parametrów i jeśli zmieniamy jej implementację

Przeciążanie metod - podobne do przesłaniania metody, ale w tym wypadku różni się zwracaną wartością i/lub parametrami.

Specyfikatory dostępu dla atrybutów i operacji

+ - publiczny(public) - osiągalny spoza klasy

- chroniony(protected) - osiągalny wewnątrz klasy i w podklasach

- - prywatny(private) - osiągalny tylko wewnątrz klasy

Asocjacje reprezentują związki między instancjami klas. Każda asocjacja ma co najmniej dwa punkty końcowe; każdy z nich jest przytwierdzony do jednej z klas w asocjacji. Asocjacje łączące dwie klasy nazywamy binarnymi, a łączące więcej klas n-arnymi. Liczność danej klasy powinna pokazywać ilość możliwych związków obiektu tej klasy z pozostałymi obiektami asocjacji n-arnej.

Asocjacje skierowane - nawigalność, czyli możliwość przejścia z klasy do klasy. Klasa posiada informacje na temat klasy, do której prowadzi strzałka. Tej informacji nie ma w przeciwnym kierunku.

*Agregacja j*est szczególny przypadek asocjacji wskazujący na silny związek dwóch klas typu część-całość.

Kompozycja to silniejsza wersja agregacji. Występuje w przypadkach, gdy część możemy rozpatrywać tylko w relacji z całością. Część nie może istnieć bez całości. Usunięcie całości powoduje usunięcie części.

Dziedziczenie to związek łączący nadklasy z podklasami. Podklasy posiadają(dziedziczą) i mogą inaczej implementować(przesłaniać), jak również mogą dodawać nowe atrybuty, operacje, asocjacje z nadklasy.

Strona główna UML-a organizacji OMG: http://www.uml.org/

3.3. XML

XML (Extensible Markup Language) - Rozszerzalny Język Znaczników. Został utworzony przez World Wide Web Consortium (W3C, http://www.w3c.org), organ ustawodawczy Sieci. Język ten udoskonala funkcjonalność Webu, a także innych dziedzin informatycznych pozwalając składować informacje w bardziej wierny, elastyczny i łatwiejszy do dostosowania sposób. Służy do przechowywania, ułatwia przekazywanie danych, nie zajmuje się ich wyglądem.

XML jest rozszerzalny ponieważ nie jest zamkniętym zbiorem znaczników jak HTML. Jest on raczej metajęzykiem – językiem do opisywania innych języków – umożliwiającym tworzenie własnych, nowych języków znaczników dla nieograniczonych, różnorodnych typów dokumentów. Przykładem może być XMI – format służący do zapisywania modelu UML. Z kolei na potrzeby zapisywania lekcji i projektów w EduCASE został utworzony nowy format, którego opis można znaleźć w rozdziale „Przyjęte rozwiązania implementacyjne”. Także plik jnlp, który pozwala na pobranie i uruchomienie aplikacji z internetu jest tworzony za pomocą rozszerzalnego języka znaczników.

Sam XML jest jedynie podzbiorem (oferuje mniej więcej 80% możliwości, przy dużo łatwiejszym ich wykorzystaniu) języka SGML. SMGL z kolei jest bardzo ogólny i posiada ogromne możliwości, jednak nie zyskał popularności, gdyż jego uniwersalność utrudnia także jego naukę.

Na podstawie poniższego kodu zostanie opisana budowa XML:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<post>
     <mail priority=”1”>
```

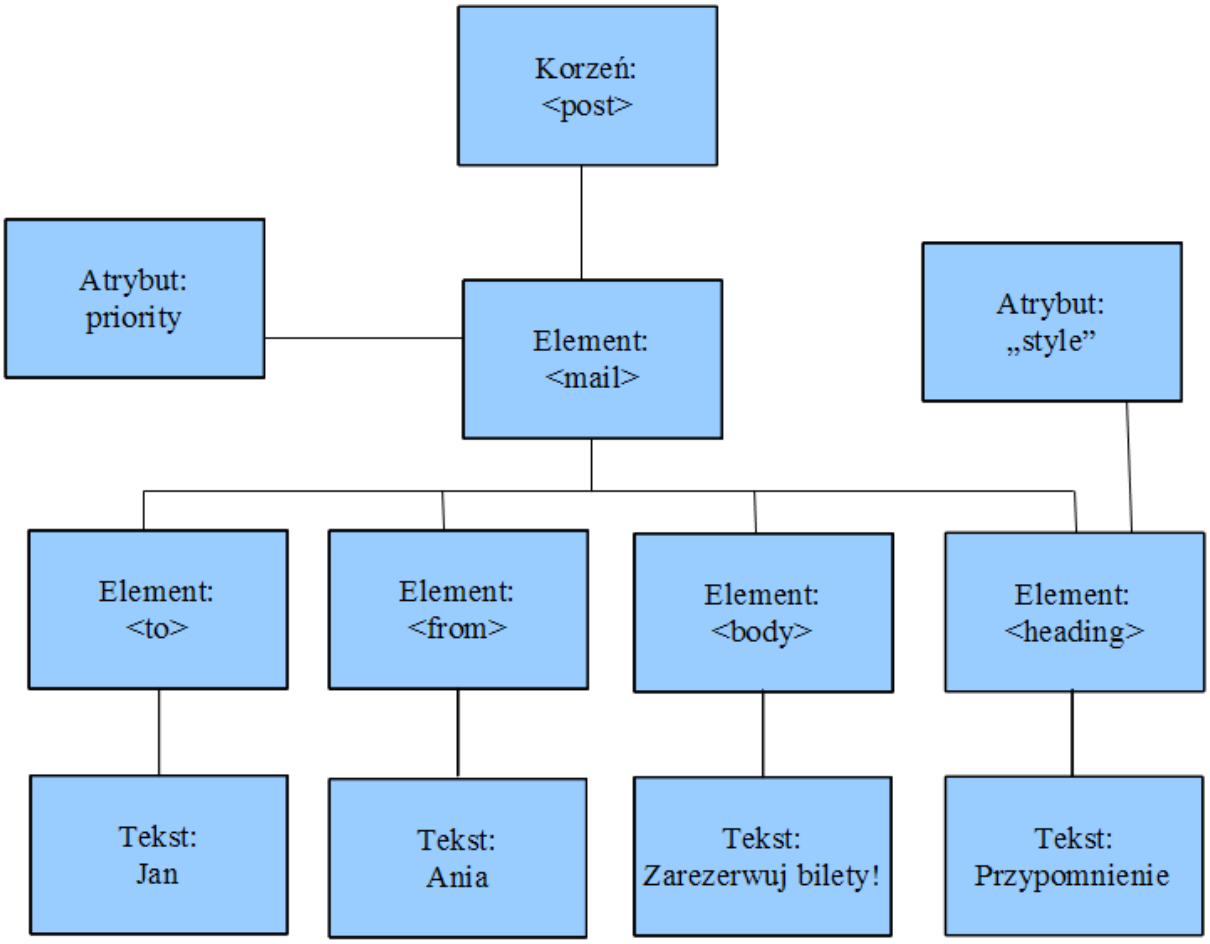
```
        <to>Jan</to>
        <from>Ania</from>
        <heading style="NORMAL">Przypomnienie</heading>
        <body>Zarezerwuj bilety!</body>
    </mail>
    <mail priority="2">
        <to>Ania</to>
        <from>Jan</from>
        <heading style="ITALIC">Odpowiedź</heading>
        <body>Ok</body>
    </mail>
</post>
```

Listing 1:

Pierwsza linia to deklaracja XML-a. Definiuje jego wersję (1.0) oraz używane kodowanie znaków.

W zwykłym XML-u tagi nie są predefiniowane, więc trzeba stworzyć własne tagi, takie jak <to> czy <from>. Każdy z tagów musi posiadać odpowiadający mu tag zamykający, takie jak </to> czy </from>. Nie dotyczy to deklaracji.

Plik posiada tak zwany korzeń – element główny. W nim zawiera się pozostała część kodu. W powyższym przypadku jest to <post>. Jest on elementem nadrzędnym - rodzicem dla <mail>. One z kolei zawierają swoje elementy podrzędne - dzieci <from>, <to>, <heading>, <body>. Najniższe elementy zawierają czysty tekst. Budowę tą można zaprezentować wizualnie. Jest to struktura drzewa:



Rysunek 6:

Rozróżniane są duże i małe znaki, czyli element <post> jest postrzegany jako co innego niż <Post>. Elementy muszą się poprawnie zagnieżdżać. Najlepiej zaprezentuje to przykład:

```
<to>Jan</to><from>Ania</from> - jest poprawne
<to>Jan<from>Ania</to></from> - jest niepoprawne
```

Listing 2:

Elementy mogą zawierać atrybuty w postaci par nazwa-wartość. Wartości muszą znajdować się w cudzysłowie. Powinny jedynie dostarczać informacji, które nie są częścią danych. Atrybuty:

- nie mogą zawierać wielu wartości

- nie mogą przechowywać struktury drzewa
- są trudno rozszerzalne

Dlatego bezpieczniej jest używać elementów, pozostawiając atrybuty do zapisu metadanych.

3.4. RAD

Termin RAD (błyskawiczne tworzenie oprogramowania) jest bardzo obszerny. To ważne zagadnienie w dzisiejszym świecie programistycznym. Często spotykamy sytuacje, kiedy ważne jest szybkie ukończenie projektu. RAD nawiązuje do wielu metodyk, które także zostały stworzone w celu wspomagania tworzenia oprogramowania w najszybszy i najbardziej poprawny sposób. RUP (Rational Unified Process) wspomina o wizualnym modelowaniu oprogramowania, co w praktyce odnosi się do UML i tym podobnych technik. Przy „szybkim tworzeniu oprogramowania" nie można zapomnieć o temacie generowaniu kodu, co znacząco przyspiesza budowę aplikacji. Dwa ostatnie zagadnienia są poruszane w pracy EduCASE.

3.5. Technologie Webowe

3.5.1 PHP

PHP to bardzo popularny język skryptowy wykonywany po stronie serwera. Służy głównie do tworzenia aplikacji internetowych, kod może zagnieżdżać się w HTML-u. Ma jedną ważną zaletę, jest całkowicie darmowy, dzięki czemu świetnie nadaje się do produkcji portali takich jak, np. serwisy społecznościowe czy galerie. PHP nie może jednak pochwalić się największym bezpieczeństwem czy niezawodnością. Jego kod nie jest kompilowany. Dlatego w sektorze bankowym przeważają systemy wykorzystujące konkurencyjne technologie takie jak Microsoft ASP.NET czy J2EE. Najczęściej współdziała z serwerem aplikacji Apache oraz bazą danych MySQL.

Strona domowa: http://www.php.net/

3.5.2 HTML

HTML – HyperText Markup Language (hipertekstowy język znaczników) - służy do tworzenia stron internetowych. Jest to bardzo stara technologia, której pierwsza specyfikacja sięga 1991 roku. Z czasem powstawały nowe wersje i odmiany takie jak DHTML – wykorzystujący dynamiczną zmianę strony, czy XHTML – przestrzegający zasad XML.

HTML zawiera ograniczoną liczbę znaczników, dzięki którym przeglądarki mogą zaprezentować treść strony internetowej w żądany przez autora sposób. Można to porównać do funkcjonalności edytora tekstu takiego jak MS Word. Do dyspozycji mamy tagi, które oznaczają, np. link, nagłówek, listę, podkreślenie, obrazek.

Tutorial: http://www.w3schools.com/html/DEFAULT.asp

3.5.3 CSS

CSS - Kaskadowe arkusze stylów (Cascading Style Sheets) to kolejna technika przydatna przy tworzeniu stron WWW. Dawniej wszelkie informacje na temat sposobu wyświetlania treści w przeglądarce zawierały się we wcześniej wspomnianym HTML-u. Z czasem odstąpiono od tego na rzecz CSS. W kaskadowych arkuszach stylów możemy zapisać w oddzielnym pliku jak będą wyświetlane poszczególne znaczniki HTML. Ponadto elementom można nadać klasy lub identyfikatory, które następnie mogą być zbiorczo stylizowane. Od czasów CSS zmian wyglądu można dokonywać w jednym miejscu zamiast przykładowo kilkudziesięciu, jak to miało miejsce dawniej. Możemy dzięki temu zaoszczędzić wiele czasu.

Tutorial: http://www.w3schools.com/Css/default.asp

3.5.4 JNLP

JNLP (Java Network Launch Protocol) to protokół oparty na xml, który może służyć do rozprowadzania aplikacji Java oraz JavaFX w internecie. Bezpośrednio korzysta on z aplikacji Java Web Start.

3.5.5 Java Web Start

Dzięki Java Web Start możemy pobrać i uruchomić aplikację napisaną w Javie bezpośrednio z internetu. Dzieje się to poprzez otworzenie pliku jnlp, który może znajdować się na stronie WWW lub na dysku lokalnym komputera. Plik ten z kolei zawiera ścieżkę do jar-a, który zawiera zarchiwizowane źródła i zasoby niezbędne do odpalenia programu. Java Web Start daje nam pewność, iż zainstalowaliśmy najnowszą wersję aplikacji równocześnie upraszczając cały proces.

Program Java Web Start stanowi element środowiska Java Runtime Environment (JRE) i jest instalowany wraz z nim.

Technologie webowe były używane przy tworzeniu strony internetowej dotyczącej tej pracy magisterskiej znajdującej się pod adresem: http://endrizzt.net/educase

3.6. Eclipse

W trakcie pisania pracy magisterskiej używałem Eclipse jako zintegrowanego środowiska programistycznego. Został on napisany w języku Java do tworzenia programów także w tym języku. Z czasem Eclipse rozwinął się jako cała kolekcja wolnego oprogramowania, wspierająca także inne języki programowania, np. C/C++ czy PHP. Jest dodatkowo rozszerzalny poprzez mechanizm wtyczek.

Strona domowa: http://www.eclipse.org/

Godnym konkurentem dla Eclipse jest NetBeans IDE firmy Sun Microsystems (http://www.netbeans.org/index.html)

3.7. Visual Editor

Visual Editor to wolna, otwarta platforma programistyczna pozwalająca na wizualne tworzenie interfejsu użytkownika. Dzięki VE budowanie aplikacji staje się, dużo łatwiejsze i zajmuje mniej czasu. VE dostępny jest jako plugin do środowiska Eclipse.

Strona domowa: http://www.eclipse.org/vep/WebContent/main.php

Wizualne modelowanie GUI dla języka Java jest także możliwe w NetBeans.

3.8. Subversion

Subversion jest udostępniany na zasadzie wolnego oprogramowania na licencji Apache. Jest to system kontroli wersji. Powstał na bazie CVS (Concurrent Versions System). Wyeliminował niektóre jego wady, dostarczając nowych możliwości. Warto tutaj wspomnieć np. o zapamiętywaniu historii zmian nazw plików i katalogów i atomowości transakcji. Teoretycznie taka aplikacja najbardziej przydaje się w sytuacji, gdy duża grupa ludzi działa na większej ilości plików, lecz SVN może być pomocny, gdy operuje się na wielu stanowiskach.

Najnowsza wersja pracy znajduje się także w repozytorium SVN pod adresem: http://endrizzt.net/svn/mgr

Strona domowa CVS: http://cvs.nongnu.org/

Strona domowa: http://subversion.tigris.org/

3.9. Open Office

Open Office to pakiet oprogramowania biurowego. Jest produkt typu open source, który godnie stara się konkurować z Microsoft Office. W skład Open Office wchodzą następujące moduły:

- Writer – edytor tekstu
- Calc – arkusz kalkulacyjny
- Impress – program do tworzenie prezentacji
- Draw – program, dzięki któremu można tworzyć i obrabiać grafikę wektorową
- Base – bazy danych
- Math – program, dzięki któremu można tworzyć formuły matematyczne

W pracy korzystano z modułu Writer, który służył głównie do stworzenia części opisowej i eksportowania jako plik pdf.

Strona domowa: http://www.openoffice.org/

3.10.Notepad++

Notepad++ jest bardzo dobrym edytorem kodów źródłowych. Jest darmowy, co moim zdaniem daje mu przewagę nad innym produktem tego typu, bardzo dobrze dopracowanym EditPlus. Program posiada szereg pluginów, w tym bardzo przydatny moduł do synchronizowania z ftp. Obsługuje wiele języków takich jak: ASP, C, C#, CSS, HTML, JAVA, PHP, XML i wiele innych.

Notepad++ był głównie używany do tworzenia strony internetowej odnoszącej się do tej pracy magisterskiej znajdującej się pod adresem: http://endrizzt.net/educase

Strona domowa: http://notepad-plus.sourceforge.net/uk/site.htm

4. Propozycja nowego systemu CASE wspomagającego edukację

Praca ma na celu zaproponowanie nowego systemu CASE, który będzie wspomagał edukację. Stworzenie nowej jakości jest możliwe dzięki przetestowaniu dotychczasowych rozwiązań. Część rezultatów można zaobserwować w rozdziale drugim opisującym stan sztuki. Można tam przeczytać zarówno na temat narzędzi służących do analizy i projektowania oprogramowania, jak i wspomagających edukację. Poniżej zostaną zaprezentowane funkcjonalności najbardziej pożądane z zastrzeżeniem jednak, iż nie wszystkie znajdą się w prototypie EduCASE. Natomiast elementem wyróżniającym produkt jest połączenie rozwiązań z dwóch różnych obszarów informatyki.

4.1. Pomoc programu

Zdecydowana większość oprogramowania posiada pomoc. Może to być wbudowana funkcjonalność lub odnośnik do strony internetowej. W EduCASE dostępna jest druga opcja pod adresem http://endrizzt.net/educase/help.php . Pomoc taka opisuje wszystkie opcje, z których użytkownik może skorzystać w aplikacji. Można zauważyć tutaj drobny paradoks. Z jednej strony wydaje się to niezbędne, z drugiej jednak jeżeli narzędzie jest dobrze zaprojektowane, interfejs jest intuicyjny, pomoc okazuje się najrzadziej używaną funkcją.

4.2. Opis teoretyczny

W rozdziale opisującym stan sztuki, a konkretnie tutoriale ze strony w3schools zauważamy zaletę jaką jest umieszczenie krótkich opisów teoretycznych opowiadających o poszczególnych zagadnieniach. Uznałem to za równie przydatną funkcjonalność. Jest ona dostępna w programie dla użytkownika, na stronie internetowej dla każdej osoby pragnącej poduczyć się teorii dotyczącej konstrukcji diagramów klas, jak również znajduje się ona w tej pracy, jednak w okrojonej wersji.

4.3. Podpowiedzi

W EduCASE wprowadzony został cały szereg podpowiedzi, które informują o konstrukcjach nieprawidłowych dla języka UML lub też dla języka programowania. Tego typu komunikatów brakuje w niektórych narzędziach, a między innymi zabronione nam będzie:

- stworzenie klasy o tej samej nazwie, co jedna ze stworzonych klas
- stworzenie atrybutu o tej samej nazwie, co jeden ze stworzonych stworzonych atrybutów w danej klasie
- stworzenie operacji o tej samej nazwie, parametrach, typach zwracanych, co jedna z już stworzonych operacji w danej klasie
- stworzenie klasy, atrybutu itd. bez podania nazwy
- stworzenie diagramu, na którym wystąpi pętla dziedziczenia
- stworzenie asocjacji lub dziedziczenia bez utworzenia wcześniej jednej lub dwóch klas

4.4. Lekcje

Wspomnieliśmy już o opisie teoretycznym i podpowiedziach dostępnych w programie. Teraz zajmiemy się częścią bardziej praktyczną. Można to porównać do testów sprawdzających wiedzę odwiedzającego stronę w3schools. W EduCASE wprowadzimy nową funkcjonalność niespotykaną wcześniej w narzędziach CASE. Więc czym są lekcje? Lekcja to tak naprawdę zapisany, zwykły projekt zawierający diagram klas UML, który odpowiada zamieszczonej razem z nim treści zadania. Po uruchomieniu takiego ćwiczenia użytkownik widzi opis i zaczyna rysować diagram. W każdym momencie może porównać swój model do wzorcowego. Otrzymuje komunikat zwrotny o braku lub nadmiarowości pewnych elementów. Program sprawdza:

- klasy – porównywane są nazwy klas, czy istnieje nazwa w diagramie wzorcowym, której brakuje w rysowanym lub czy jest nadmiarowa w rysowanym modelu, a nie ma jej we wzorcowym. Pomijana jest wielkość liter.
- atrybuty – jak wyżej, lecz porównywane są atrybuty dla danej klasy pod względem specyfikatora dostępu, nazwy, typu, powtarzalności, wartości domyślnej, czy atrybut jest pochodny, klasowy
- operacje – jak wyżej. Sprawdzane są specyfikatory, nazwy, parametry, typ zwracany, czy metoda jest abstrakcyjna, klasowa
- dziedziczenia – testowany jest brak lub nadmiar dziedziczeń, korzystając z nazw klas i z zachowaniem kierunku generalizacji
- asocjacje – porównywane są po nazwach łączonych klas razem z typem i kierunkiem asocjacji

Do tego dochodzą pewne wskazówki, np. mówiące o możliwej pomyłce atrybutu z klasą. Możemy otrzymać taki komunikat, jeżeli znaleziony zostanie atrybut w diagramie rysowanym o takiej samej nazwie co klasa we wzorcowym. Podobnie sprawdzane są nazwy klas w rysowanym modelu i atrybuty we wzorcowym.

4.5. Aplikacja internetowa

Warto zwrócić uwagę na coraz większy udział w rynku aplikacji internetowych, które do uruchomienia potrzebują jedynie przeglądarki internetowej plus ewentualnie pluginów obsługujących dodatkowo potrzebną technologię. Pionierem takiego oprogramowania była firma Google, która stworzyła między innymi produkty o okrojonej funkcjonalności Microsoft Office, czy kalendarz. W obecnej chwili systemy te, stworzone przecież bardzo niedawno, wydają się niezastąpione. Z kolei firma Adobe może pochwalić się webową wersją swojego najsłynniejszego wyrobu jakim jest Photoshop. Dostępna jest na stronie https://www.photoshop.com . Zwraca uwagę bardzo ładny interfejs aplikacji i niestety bardzo okrojone możliwości w stosunku do pierwowzoru. Moim zdaniem narzędzie CASE

uruchamiane z okna przeglądarki miałoby szansę uzyskanie przewagi nad konkurencją. Istnieje pomysł, aby w ten sposób przekształcić obecny prototyp, a na dzień dzisiejszy oferowana jest strona internetowa dotycząca produktu i pobieranie z niej nowych wersji.

Największym wyzwaniem jest przeniesienie interfejsu rysowania do aplikacji webowej. Cała reszta funkcjonalności jak pomoce, sprawdzanie lekcji czy generowanie kodu są proste do zaimplementowania. Dobrym rozwiązaniem byłoby wtedy stworzenie kont użytkowników, tak jak w innych produktach dostępnych w sieci. Projekty, lekcje zapisywane byłyby już w bazie danych, a nie plikach XML.

4.6. Diagramy UML

W typowym narzędziu CASE ważne jest, aby możliwe było tworzenie wszystkich diagramów UML. W prototypie skupiliśmy się na najpopularniejszym z nich, a mianowicie na diagramie klas. Pozostałe modele mogą zostać z czasem dodane przy rozwijaniu aplikacji, jednak w tym stadium zostały wyeliminowane ze specyfikacji. Sam diagram klas daje jednak duże pole do popisu, gdyż zawiera największą liczbę dostępnych konstrukcji. Różne systemy umożliwiają korzystanie z różnych elementów. W EduCASE skupiłem się na tym, aby tworzone modele miały odwzorowanie w kodzie języka Java, np. brak jest asocjacji n-arnych. W takich wypadkach implementacja odbywa się za pomocą asocjacji binarnych, więc stwierdziłem, że lepiej, jeżeli będzie to od razu widoczne na diagramie.

Opisywanie wszystkich konstrukcji diagramu klas w tym miejscu nie ma większego sensu, dlatego skupię się na pomysłach wykorzystanych w prototypie, a będzie to dotyczyło interfejsu aplikacji. Ważne, aby był intuicyjny i moim zdaniem taki jest. Wybieramy pożądany element, a po kliknięciu na ekran otwiera się nam okno jego edycji. Brak możliwości zmiany wielkości klasy na ekranie przez użytkownika zastępujemy robiąc to w sposób automatyczny. W oknie edycji klasy mamy łatwy dostęp do wszystkich atrybutów, operacji i ich parametrów. Z poziomu menu otwieramy pomoc programu, opis dotyczący UML oraz interfejs dotyczący generowania kodu. Warto tutaj zwrócić uwagę na często zbyt dużą liczbę opcji w innych narzędziach CASE, z których mało kto korzysta.

4.7. Generowanie kodu

Jest to moim zdaniem najważniejsza funkcja w perspektywie błyskawicznego tworzenia oprogramowania. Wiele firm korzysta z narzędzi CASE w fazie analizy i projektowania aplikacji. Dla tych osób posiadających diagramy UML powstałe w trakcie opracowywania projektu bardzo przydatna jest możliwość wygenerowania kodu.

Kreowanie źródeł może odnosić się do wielu płaszczyzn. Najbardziej popularnym efektem jaki otrzymujemy z diagramu klas są kody tych klas napisane w danym języku obiektowym. Można jednak uzyskać dużo więcej. Szczegółowo opisuję to w rozdziale 6.1 dotyczącym rozwoju oprogramowania. Wspominam tam także o sposobności uzyskania źródeł przydatnych przy dostępie do bazy danych, czy też odpowiedzialnych za wyświetlany interfejs, np. często spotykane, powtarzające się formularze w aplikacjach internetowych w sektorze bankowym.

Istnieje wiele języków programowania. Dla projektantów najlepszym rozwiązaniem byłoby, jeżeli narzędzie CASE obsługiwałoby je wszystkie. Jednakże co pewien czas stajemy się świadkami narodzin nowej technologii. Z kolei starsze sposoby tworzenia oprogramowania powoli zamierają. Prowadziłoby to do starzenia się takiej aplikacji, jaką jest EduCASE. Aby wspomóc obsługę przyszłych języków programowania należałoby wprowadzić możliwość aktualizacji poprzez system pluginów, które użytkownicy byliby w stanie tworzyć na własną rękę. W pluginach można byłoby przykładowo zapisać jak wygląda translacja klasy z diagramu na kod.

Kolejny aspekt to pozostawienie opcji wyboru żądanej treści do wygenerowania. Skupmy się na źródłach klas. Klient powinien mieć możliwość decydowania czy potrzebuje następujących elementów:

- kostruktory (puste, z parametrami)
- gettery (pobierające wartości właściwości)
- settery (ustawiające wartości właściwości)
- ekstensje klas (w każdej klasie osobno, czy jedna oddzielna klasa dla wszystkich)

- kod dla pojedynczych, czy wszystkich klas

Dzięki generowaniu kodu jesteśmy w stanie w dużej mierze przyspieszyć tworzenie oprogramowania. Nie tylko dostajemy wyprodukowaną dużą część za zaledwie kilka kliknięć myszką, lecz także odciążamy developera od pisania powtarzalnych konstrukcji, co wpłynie na jego płynniejszą pracę w dalszej fazie projektu.

4.8. Zapis/odczyt projektu

Zapis oraz odczyt projektu jest niezbędny dla aplikacji desktopowych, kiedy chcemy zapisać wyniki swojej aktualnej pracy i powrócić do niej w przyszłości. Bez tej funkcjonalności wiele programów nie nadawałaby się do użytku. Można do nich zaliczyć EduCASE. Powinniśmy mieć możliwość zapisania tworzonego przez siebie diagramu klas. Specyfika prototypu wymaga także, by można było także zachować lekcje, które następnie mogą być zmieniane jak zwykłe projekty, a także rozpoczynane przez użytkownika pragnącego sprawdzić swoją wiedzę.

W celu zapisywania pracy w EduCASE został stworzony XML o określonej składni, która powstała na potrzeby tej pracy. Dzięki temu mogliśmy pokazać, jak wyglądają pliki tego typu. Ogólna charakterystyka technologii znajduje się w punkcie 3.3, natomiast opis użytej składni w punkcie 5.4. Format zdobył dużą popularność w dzisiejszym świecie programistycznym i zapewne będzie jeszcze zyskiwał w przyszłości. Warto jednakże nadmienić o dużej liczbie typów, do których mogą eksportować inne narzędzia do projektowania oprogramowania. Wśród nich wyróżniamy między innymi:

- XMI – standard XML do zapisu diagramów UML
- formaty graficzne (bmp, jpg, gif, itd.)
- formaty do zapisu grafiki wektorowej
- formaty do współpracy z innymi aplikacjami (np. AutoCAD)
- strona internetowa i wiele innych

4.9. Aktualizacja oprogramowania

Możemy zauważyć, że w obszarze aplikacji desktopowych aktualizacja polega na pobraniu i zainstalowaniu nowej wersji ze strony internetowej lub pobieraniu przez sam program uaktualnienia, przy czym często niezbędne jest ponowne uruchomienie oprogramowania albo całego systemu operacyjnego. Taki cykl, jeśli powtarza się często może stać się irytujący dla użytkownika. Sytuację taką można zauważyć, np. w przypadku odtwarzacza multimedialnego Winamp.

Technologia Java i jej Java Web Start pozwalają uniknąć tej czasochłonnej solucji. Wykorzystanie pliku jnlp sprawia, iż za każdym razem, gdy pojawi się nowa wersja, aplikacja ta zostanie pobrana i zainstalowana. Eliminujemy dzięki temu czas, poświęcony na restartowanie systemu lub szukanie aktualizacji na portalu internetowym.

4.10. Podsumowanie

Powyżej opisaliśmy najważniejsze aspekty nowego narzędzia. Aplikacja ta skupia się na dwóch obszarach nowatorsko je łącząc, a mianowicie chodzi o komputerowe wspieranie projektowania oprogramowania oraz wspomaganie edukacji. EduCASE oferuje funkcjonalności, które są niezbędne takie jak pomoc programu, zapisywanie projektów czy generowanie kodu źródłowego. Ponadto dość nietypowo posiada kilka opcji pomagających w nauce tworzenia diagramów klas. Jest to zamieszczony opis teoretyczny poszczególnych elementów, system podpowiedzi oraz możliwość sprawdzenia wiedzy za pomocą lekcji. Jako prototyp narzędzie nie jest doskonałe. Brakuje między innymi funkcjonalności z zakresu inżynierii odwrotnej, aby z już istniejącego kodu można było stworzyć diagram. Z kolei inne aplikacje tego typu pozwalają na rysowanie także pozostałych modeli UML. Najważniejsze moim zdaniem byłoby wprowadzenie tego systemu do świata webu. Dzięki temu międzynarodowe zespoły programistów i analityków miałyby ułatwioną pracę, dzięki projektowaniu oprogramowania za pomocą narzędzia działającego w środowisku przeglądarki internetowej.

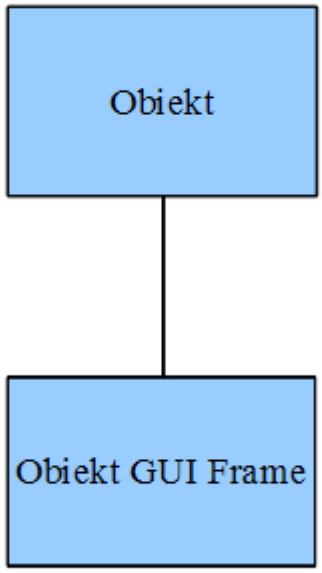
5. Przyjęte rozwiązania implementacyjne

5.1. Podział kodu

W aplikacji zastosowano podział kodu. Pomysł wziął się z wzorca projektowego MVC (Model – View – Controller) - Model – Widok – Kontroler. Wzorzec ten pozwala oddzielić model danych, logikę i interfejs użytkownika. Jest szeroko stosowany w aplikacjach webowych – bazodanowych. Powstało nawet wiele frameworków do szybkiego tworzenia internetowego oprogramowania korzystających z tego wzorca, co odnosi się także do RAD, o którym praca wspomina w innym miejscu. W EduCASE MVC zostało ograniczone do oddzielenia dwóch partii kodu:

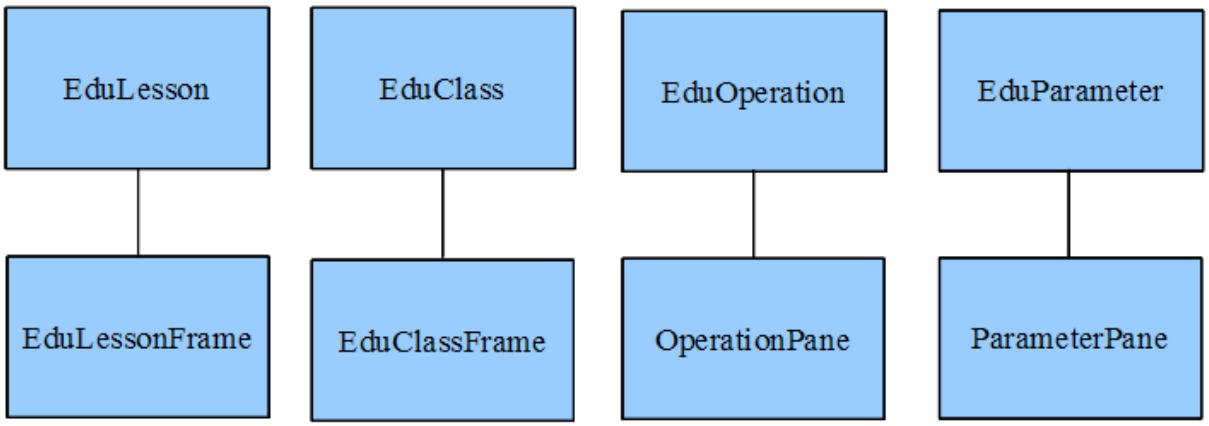
- interfejsu użytkownika (GUI)
- modelu i kontrolera

Wizualnie można przedstawić to w następujący sposób:



Rysunek 7:

Kilka przykładów:



Rysunek 8:

Można zaobserwować pewną niekonsekwencję w nazwach klas odpowiedzialnych za wyświetlanie GUI. Różnice zachodzą w zależności od nadklasy, po której dana GUI klasa dziedziczy. W podanym przykładzie EduLessonFrame i EduClassFrame dziedziczą po JinternalFrame, natomiast OperationPane i ParameterPane po klasie JLayeredPane.

5.2. Algorytm sprawdzania lekcji

Ciekawym i wartym udoskonalenia rozwiązaniem jest algorytm sprawdzania lekcji. Poniżej poszczególne elementy tego algorytu:

lessonErrorList – lista zawierająca komunikaty o błędach

lessonWarningList – lista zawierająca komunikaty o podpowiedziach

checkNamesClass – metoda sprawdzająca zarówno czy brak jak i nadmiarowość klas. W obu przypadkach do listy lessonErrorList dodawany jest odpowiedni komunikat z nazwą klasy z listy names uzyskanej po wykonaniu metody checkNameClass.

checkNameClass – wywoływana w metodzie checkNamesClass dwukrotnie z parametrami w postaci listy klas we wzorcowym diagramie i listy klas z rysowanego diagramu. Sprawdza czy kolejne klasy z jednej listy mają taką samą nazwę jak którakolwiek klasa z drugiej listy. Pomija wielkość liter. Jeżeli brak odpowiadającej klasy, nazwa dopisywana jest do listy names, która jest dalej wykorzystywana w metodzie checkNamesClass.

checkAttributesAndOperations - metoda sprawdzająca zarówno czy brak jak i nadmiarowość atrybutów lub operacji. Do listy lessonErrorList dodawany jest odpowiedni komunikat z nazwą atrybutu lub operacji z listy names uzyskanej po wykonaniu metody checkAttributesNames lub checkOperationsNames.

checkAttributesNames – wywoływana w metodzie checkAttributesAndOperations dwukrotnie z parametrami w postaci listy atrybutów we wzorcowym diagramie i listy atrybutów z rysowanego diagramu. Porównywane są atrybuty dla klas o takich samych nazwach w obu listach. Sprawdza czy kolejne atrybuty z jednej listy mają taką samą nazwę jak którykolwiek atrybut z drugiej listy. Pomija wielkość liter. Jeżeli brak odpowiadającego atrybutu, nazwa dopisywana jest do listy names, która jest dalej wykorzystywana w metodzie checkAttributesAndOperations.

checkOperationsNames – wywoływana w metodzie checkAttributesAndOperations dwukrotnie z parametrami w postaci listy operacji we wzorcowym diagramie i listy operacji z rysowanego diagramu. Porównywane są operacje dla klas o takich samych nazwach w obu listach. Sprawdza czy kolejne operacje z jednej listy mają taką samą nazwę jak którakolwiek operacja z drugiej listy. Pomija wielkość liter. Jeżeli brak odpowiadającej operacji, nazwa dopisywana jest do listy names, która jest dalej wykorzystywana w metodzie checkAttributesAndOperations.

checkNamesInheritance - metoda sprawdzająca zarówno czy brak jak i nadmiarowość dziedziczenia. W obu przypadkach do listy lessonErrorList dodawany jest odpowiedni komunikat z nazwą dziedziczenia z listy names uzyskanej po wykonaniu metody checkNameInheritance. Nazwy dziedziczenia mają postać: nadklasa ← klasa.

checkNameInheritance – wywoływana w metodzie checkNamesInheritance dwukrotnie z parametrami w postaci listy dziedziczeń we wzorcowym diagramie i listy dziedziczeń z rysowanego diagramu. Sprawdza czy kolejne dziedziczenia z jednej listy mają taką samą nazwę jak którekolwiek dziedziczenie z drugiej listy. Pomija wielkość liter. Jeżeli brak odpowiadającego dziedziczenia, nazwa dopisywana jest do listy names, która jest dalej wykorzystywana w metodzie checkNamesInheritance.

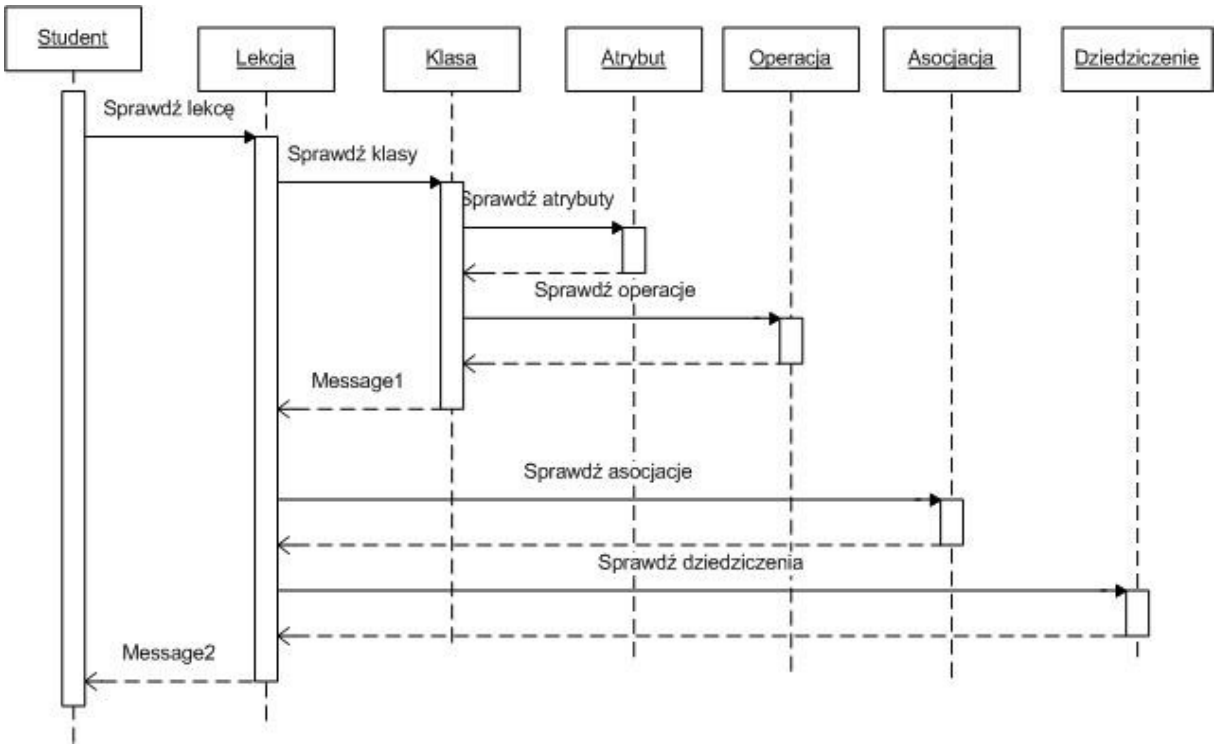
checkNamesAssociation - metoda sprawdzająca zarówno czy brak jak i nadmiarowość asocjacji. W obu przypadkach do listy lessonErrorList dodawany jest odpowiedni komunikat z nazwą asocjacji z listy names uzyskanej po wykonaniu metody checkNameAssociation. Nazwy asocjacji mają postać: nazwa[pierwszaKlasa:drugaKlasa]. Ponadto sprawdzany jest typ asocjacji, np. czy asocjacja jest skierowana w odpowiednią stronę.

checkNameAssociation – wywoływana w metodzie checkNamesAssociation dwukrotnie z parametrami w postaci listy asocjacji we wzorcowym diagramie i listy asocjacji z rysowanego diagramu. Sprawdza czy kolejne asocjacje z jednej listy mają taką samą nazwę jak którakolwiek asocjacja z drugiej listy. Pomija wielkość liter. Jeżeli brak odpowiadającej asocjacji, nazwa dopisywana jest do listy names, która jest dalej wykorzystywana w metodzie checkNamesAssociation.

checkNamesClassAndAts – metoda ta sprawdza czy którakolwiek klasa z diagramu wzorcowego odpowiada nazwą któremukolwiek atrybutowi z diagramu rysowanego oraz czy którykolwiek atrybut z diagramu wzorcowego odpowiada nazwą którejkolwiek klasie z diagramu rysowanego. Pomija wielkość liter. Powodem tego badania jest to, iż przy rysowaniu diagramu UML często zdarzają się pomyłki w postaci pomylenia klasy z atrybutem. Jeżeli zdarzy się taki przypadek, komunikat „podpowiedzi” dodawany jest do listy lessonWarningList.

Sprawdzenie lekcji może odbyć się w dowolnym czasie rysowania diagramu. Można tego dokonać dowolną ilość razy. Za każdym razem wyświetlą się wszystkie opisane wyżej komunikaty zarówno na temat błędów jak i na temat podpowiedzi.

Można go przedstawić za pomocą diagramu sekwencji:



Rysunek 9:

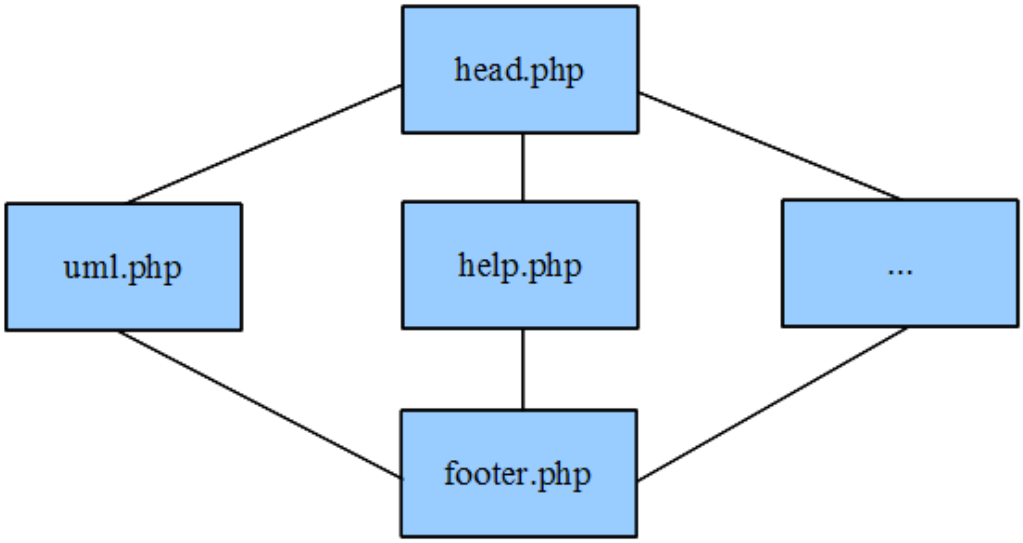
5.3. Strona internetowa

Na potrzeby tej pracy magisterskiej powstała strona internetowa mieszcząca się pod adresem http://endrizzt.net/educase. Można tam znaleźć między innymi konspekt, opis zagadnień dotyczących języka UML oraz pomoc programu EduCASE. Do pobrania są także kody źródłowe aplikacji, dokument stanowiący opis pracy i plik jnlp, dzięki któremu można zainstalować EduCASE na lokalnym komputerze. Niestety plik ten trzeba najpierw ściągnąć na dysk lokalny i dopiero uruchomić, gdyż nie było możliwości ustawienia typów mime serwera, przez co nie jest poprawnie obsługiwany.

Rysunek 10:

Strona powstała za pomocą technologii PHP, HTML, CSS. PHP używane jest praktycznie jedynie do dołączania plików zawierających nagłówek, spis treści oraz stopkę każdej ze stron. Zostało to rozwiązane w ten sposób, aby przy poprawce któregoś z tych elementów można było dokonywać zmian tylko w jednym pliku, a nie oddzielnie dla każdej ze stron. Mechanizm ten można porównać do master page, który używany jest w ASP.NET.

Ogólny sposób działania można zaprezentować na rysunku:



Rysunek 11:

Kod educase.jnlp:

```
<?xml version="1.0" encoding="utf-8"?>
<jnlp spec="1.5+"
      codebase="http://endrizzt.net/educase/"
      href="educase.jnlp">

    <information>
    <title>EduCASE</title>
        <vendor>endrizzt</vendor>
        <homepage href="http://endrizzt.net/educase/"/>
        <description>EduCASE</description>
        <offline-allowed/>
    </information>
     <security>
      <all-permissions/>
     </security>
```

```
    <resources>

        <j2se version="1.5+"
href="http://java.sun.com/products/autodl/j2se">
        </j2se>

        <jar href="educase.jar" main="true"/>
     <property name="java.security.policy"
value="wideopen.policy"/>
    </resources>

    <application-desc main-class="EduCase">

    </application-desc>
</jnlp>
```

Listing 3:

Jak widać w powyższym pliku jnlp, aplikacja uruchamiana jest poprzez educase.jar. JAR (Java ARchive) to format pliku, który pozwala łączyć wiele plików w jeden zarchiwizowany plik. Standardowo plik JAR zawiera pliki klas i dodatkowe zasoby związane z apletami i aplikacjami.

Format JAR dostarcza wielu korzyści:

- Bezpieczeństwo. Można cyfrowo podpisać zawartość pliku JAR. Użytkownicy, którzy rozpoznają sygnaturę mogą następnie ewentualnie udzielić programowi przywilejów bezpieczeństwa, których nie posiadałby w innym przypadku.
- Krótszy czas pobierania. Jeżeli aplet jest dostępny jako pojedynczy plik JAR, wszystkie pliki klas i dodatkowe zasoby mogą być pobierane przez przeglądarkę internetową w pojedynczej transakcji HTTP bez potrzeby tworzenia nowych połączeń dla każdego z plików.
- Kompresja. Format JAR pozwala na kompresję plików, dzięki czemu ich przechowywanie jest bardziej efektywne zajmując mniej miejsca.

- Wersjonowanie. Plik JAR może przetrzymywać dane na temat przechowywanych plików takie jak dostawca oprogramowania i informacje na temat wersji oprogramowania.
- Zabezpieczenie pakietu. Pakiety przechowywane w plikach JAR opcjonalnie mogą być zabezpieczone w ten sposób, że pakiet wymusza spójność wersji. Zabezpieczenie pakietu w pliku JAR oznacza, iż wszystkie pakiety zdefiniowane w tym pakiecie muszą zostać odnalezione w tym samym pliku JAR.
- Pakiety rozszerzeń. Framework rozszerzeń dostarcza środków, dzięki którym można dodać funkcjonalność do głównej platformy Javy, a format pliku JAR definiuje pakiety rozszerzeń. Java 3D i JavaMail to przykłady rozszerzeń wyprodukowanych przez firmę Sun. Dzięki użyciu formatu pliku JAR można także zamienić własną aplikację w rozszerzenie.
- Przenaszalność. Mechanizm obsługi plików JAR jest standardową częścią API głównej platformy Javy.

Adresy internetowe:

- http://endrizzt.net/educase – strona główna
- http://endrizzt.net/educase/help.php – pomoc programu EduCASE
- http://endrizzt.net/educase/EduCase.zip – źródła programu EduCASE
- http://endrizzt.net/educase/EduCase_PracaMgr.pdf – część opisowa
- http://endrizzt.net/educase/educase.jnlp – plik jnlp

Tutorial JNLP i Java Web Start:

http://java.sun.com/developer/technicalArticles/Programming/jnlp/

5.4. Format zapisu projektu i lekcji

Narzędzie EduCASE oferuje możliwość zapisywania tworzonego projektu lub lekcji. Używa do tego pliku xml o określonej składni, która w podpunktach dla łatwiejszego objaśnienia została opisana poniżej:

Ogólny schemat wygląda następująco:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<DIAGRAM>
     <CLASSES>
          <CLASS>
               ...
          </CLASS>
          <ASSOCIATION>
               ...
          </ASSOCIATION>
          <INHERITANCE>
               ...
          </INHERITANCE>
     </CLASSES>
     <LESSON>
          ...
     </LESSON>
</DIAGRAM>
```

Listing 4:

Może istnieć wiele elementów CLASS, ASSOCIATION i INHERITANCE. Element LESSON występuje lub nie w zależności czy zapisywana jest lekcja.

Opis elementu LESSON:

```
<LESSON>
     <TITLE>nazwa lekcji</TITLE>
```

```
     <CONTENT>treść lekcji</CONTENT>
</LESSON>
```

Listing 5:

Opis elementu CLASS:

```
<CLASS>
     <NAME>nazwa klasy</NAME>
     <ABSTRACT>true lub false</ABSTRACT>
     <X>punkt x</X>
     <Y>punkt y</Y>
     <ATTRIBUTE>
          ...
     </ATTRIBUTE>
     <OPERATION>
          ...
     </OPERATION>
</CLASS>
```

Listing 6:

Może być wiele elementów ATTRIBUTE i OPERATION.

Opis elementu ATTRIBUTE:

```
<ATTRIBUTE>
     <NAME>nazwa atrybutu</NAME>
     <ACCESS>specyfikator dostępu atrybutu</ACCESS>
     <TYPE>typ atrybutu</TYPE>
     <DEFAULTVALUE>wartość domyślna</DEFAULTVALUE>
     <AMOUNT>wartość, jeżeli atrybut jest powtarzalny</AMOUNT>
     <CLASSATTRIBUTE>czy atrybut jest klasowy</CLASSATTRIBUTE>
     <DERIVATEATTRIBUTE>czy atrybut jest
```

```
pochodny</DERIVATEATTRIBUTE>
</ATTRIBUTE>
```

Listing 7:

Opis elementu OPERATION:

```
<OPERATION>
     <NAME>nazwa operacji</NAME>
     <ACCESS>specyfikator dostępu</ACCESS>
     <TYPE>typ zwracany</TYPE>
     <CLASSOPERATION>czy jest operacją
klasową</CLASSOPERATION>
     <ABSTRACTOPERATION>czy jest operacją
abstrakcyjną</ABSTRACTOPERATION>
     <PARAMETER>
          ...
     </PARAMETER>
</OPERATION>
```

Listing 8:

Może być wiele elementów PARAMETER.

Opis elementu PARAMETER:

```
<PARAMETER>
     <NAME>nazwa parametru</NAME>
     <DEFAULTVALUE>wartość domyślna</DEFAULTVALUE>
     <TYPE>typ parametru</TYPE>
</PARAMETER>
```

Listing 9:

Opis elementu ASSOCIATION:

```
<ASSIOCIATION>
     <NAME>nazwa asocjacji</NAME>
     <AMOUNTFROM> </AMOUNTFROM>
     <AMOUNTTO> </AMOUNTTO>
     <AMOUNTFROM2> </AMOUNTFROM2>
     <AMOUNTTO2> </AMOUNTTO2>
     <FIRSTCLASS>nazwa pierwszej klasy, którą łączy
asocjacja</FIRSTCLASS>
     <SECONDCLASS>nazwa drugiej klasy, którą łączy
asocjacja</SECONDCLASS>
     <ASSTYPE>typ asocjacji</ASSTYPE>
     <POINTAX>0</POINTAX>
     <POINTAY>0</POINTAY>
     <POINTBX>0</POINTBX>
     <POINTBY>0</POINTBY>
</ASSIOCIATION>
```

Listing 10:

Elementy AMOUNTFROM, AMOUNTTO – liczności przy pierwszej stronie asocjacji

Elementy AMOUNTFROM2, AMOUNTTO2 – liczności przy drugiej stronie asocjacji

Elementy POINTAX, POINTAY, POINTBX, POINTBY – przesunięcia punktu początkowego i końcowego asocjacji w stosunku do lewego górnego rogu klasy.

Element ASSTYPE – wskazuje na typ asocjacji:

0 - zwykła

1 - skierowana do Klasy 1

2 - skierowana do Klasy 2

3 - skierowana w obu kierunkach

4 - agregacja do Klasy 1

5 - agregacja do Klasy 2

6 - kompozycja do Klasy 1

7 - kompozycja do Klasy 2

Opis elementu INHERITANCE:

```
<INHERITANCE>
      <UPCLASS>nazwa nadklasy</UPCLASS>
      <DOWNCLASS>nazwa podklasy</DOWNCLASS>
      <POINTAX>0</POINTAX>
      <POINTAY>0</POINTAY>
      <POINTBX>0</POINTBX>
      <POINTBY>0</POINTBY>
</INHERITANCE>
```

Listing 11:

Elementy POINTAX, POINTAY, POINTBX, POINTBY – przesunięcia punktu początkowego i końcowego dziedziczenia w stosunku do lewego górnego rogu klasy.

6. Rozwój oprogramowania

6.1. Generowanie kodu

Jedną z funkcjonalności, które warto byłoby rozwinąć jest generowanie kodu. Im więcej zostanie wygenerowane tym mniej czasu na programowanie aplikacji będzie musiał poświęcić developer. Jest to tym bardziej znaczące, ponieważ odnosi się to do części aplikacji, która jest powtarzalna, którą programista pisał już wiele razy. Z jego perspektywy przycisk „generuj" mógłby zastąpić monotonną pracę.

Tak więc obecną funkcjonalność generowania kodu można ulepszyć. Najlepszym rozwiązaniem byłoby dodanie kilku opcji, które programista wybierałby sam zależnie od preferencji. Oto niektóre propozycje:

- gettery, settery – publiczne metody pozwalające na zapisz/odczyt atrybutów. Opcja powinna pozwalać na wygenerowanie tych metod zarówno dla wszystkich jak i poszczególnych atrybutów
- konstruktor klasy – generowanie konstruktorów zarówno dla wszystkich jak i poszczególnych klas. Klasy abstrakcyjne są pomijane. Możliwość wyboru parametrów.
- zapisanie zmian klasy – pochodna powyższych. Metoda, która jako parametr przyjmuje wszystkie lub tylko wybrane atrybuty i zastępuje nimi stare wartości.

Generowanie kodu w desktopowej aplikacji jest tematem lekko ograniczonym. Inaczej jednak wygląda sprawa, jeżeli spojrzymy na webowe aplikacje bazodanowe, z czym miałem do czynienia w pracy zawodowej. W tamtym przypadku generator tworzył dużą część kodu, pozwalał skrócić czas na jego pisanie oraz zmniejszał cierpienia programisty przy tworzeniu powtarzalnego kodu.

Dzięki temu narzędziu w krótkim czasie mogły powstać Business Entities – klasy odpowiadające tabelom z bazy danych oraz DAL (Data Access Layer) - warstwa dostępu do

danych. Generowane były metody powodujące wykonanie procedur CRUD (Insert, Select, Update, Delete) na bazie danych. Najlepszym rozwiązaniem jest tworzenie procedur Select dla każdego klucza głównego i obcego w danej tabeli.

Następną funkcjonalnością było generowanie powtarzających się formularzy. W aplikacjach bazodanowych działających na przeglądarkach internetowych, np. dla przemysłu bankowego stosuje się wiele podobnie wyglądających formularzy. Tu po raz kolejny można użyć generatora eliminując znużenie programisty i przyspieszając tworzenie oprogramowania.

Obecnie EduCASE jest aplikacją stricte desktopową, jednak po przystosowaniu jej do warunków webowych mogłaby realizować także wyżej wymienione zadania.

6.2. Format eksportowania danych

W chwili obecnej EduCASE oferuje pliki xml jako jedyny format eksportowania projektów lub lekcji. Inne tego typu narzędzia oferują różnorodne formaty.

Można tutaj wymienić formaty graficzne:

- BMP – Windows Bitmap
- GIF – Compuserve GIF
- JP2 – JP2 Format
- JPG – JPG/JPEG Format
- PNG – Portable Network Graphics
- TIFF – Tagged Image File Format

Został już wspomniany xml. Warto zauważyć, że istniej format do zapisu diagramów UML. Nazywa się on XMI – XML Metadata Interchange.

Wiele narzędzi korzysta z bardzo popularnego formatu pdf, który może przechowywać zarówno treści graficzne jak i tekstowe.

Microsoft Visio, w którym także można stworzyć wiele rodzai diagramów UML pozwala zapisać powstały projekt jako stronę internetową.

Warto także zwrócić uwagę na prostsze formaty jak zwykły plik tekstowy lub plik CSV (Comma Separated Values – wartości rozdzielone przecinkiem), który służy do przechowywania danych i jest często używany w arkuszach kalkulacyjnych, czy też aplikacjach bazodanowych.

Różne programy używają wielu typów formatów do zapisu danych, więc przy rozwijaniu EduCASE warto przemyśleć dodanie kilku z nich.

6.3. Algorytm sprawdzania lekcji

Obecny algorytm został omówiony w rozdziale „Przyjęte rozwiązania implementacyjne”. Sprawdza on w tym momencie częściową poprawność diagramu klas UML. Ta część EduCASE pozostawia moim zdaniem największe pole do popisu. Algorytm można bardzo rozbudować w wielu kierunkach, posługując się nawet technikami sztucznej inteligencji. Trzeba jednak pamiętać, że komputerowe sprawdzanie diagramu nigdy nie będzie idealne. Dany problem można rozwiązać na wiele sposobów i jedynie człowiek jest w stanie ocenić, czy dane rozwiązanie jest zasadne.

6.4. Inne diagramy UML

W narzędziu EduCASE istnieje możliwość rysowania jedynie diagramu klas. Jednakże UML udostępnia wiele innych diagramów, które można utworzyć za pomocą podobnych aplikacji. Te diagramy to:

- diagram przypadków użycia
- diagram komunikacji i diagram przebiegu
- diagram stanów i diagram aktywności
- diagram pakietów i diagram komponentów

- diagram strukturalny
- diagram opisu interakcji
- diagram przebiegów czasowych i diagram wdrożenia

6.5. Limit czasowy

Obecnie po rozpoczęciu lekcji w EduCASE, diagram można tworzyć nie będąc ograniczonym czasowo. Ze względów szkoleniowych warto wprowadzić limit czasowy. Można byłoby wtedy utworzyć, np. sprawdzian, który trwałby przykładowo 45 minut. Aplikacja mogłaby automatycznie zapisywać plik z projektem studenta uniemożliwiając dalsze poprawki.

6.6. Poprawki graficzne

EduCASE jest jedynie prototypem. Aby mógł zaistnieć w większym stopniu potrzebne byłyby poprawki w implementacji GUI aplikacji. Oto niektóre z proponowanych udoskonaleń:

- rysowanie przy przeciąganiu – przyjemnym dla oka efektem byłoby, np. przy przeciąganiu klasy ciągłe odwzorowanie jej położenia. Realizowane poprzez rysowanie klasy o nieco jaśniejszych kolorach niż standardowo.
- wiele pól do rysowania – w chwili obecnej aplikacja składa się z jednego ekranu, na którym można rysować. W przyszłości warto byłoby to zmienić i umożliwić otwieranie wiele takich okien.
- zoom – obraz wyświetlany jest w stuprocentowej widoczności. Udoskonaleniem byłaby możliwość zoomu – funkcjonalność dostępna w wielu narzędziach graficznych, czy też edytorach tekstu.

6.7. Nowe typy

Kolejnym spostrzeżeniem przy porównywaniu EduCASE do innych narzędzi jest to, iż wiele aplikacji oferuje po stworzeniu nowej klasy, wybranie jej jako np. typ atrybutu kolejnej klasy. EduCASE niestety nie posiada tej funkcjonalności pozwalając jedynie na wybranie typu ze stałej listy zawierającej: byte, short, int, long, float, double, char, String, boolean oraz typ odpowiadający wszelkim tablicom czy kolekcjom – array.

6.8. Obsługa wielu języków

EduCASE został stworzony w języku Java i z myślą o generowaniu kodu źródłowego klas także w tym języku. Znaczącym usprawnieniem byłoby dodanie obsługi dla innych języków. Użytkownik powinien mieć wybór dla jakiego danego języka programowania chce wygenerować źródła.

6.9. Format do obsługi nowych języków

Jest to rozwinięcie poprzedniego podpunktu. Został już wspomniany pomysł obsługi wielu języków programowania. Na dzień dzisiejszy w aplikacji można by na stałe wpisać możliwość generowania kodu źródłowego do kilku wybranych języków. Jednakże istnieje jeszcze inne rozwiązanie tego problemu. Można przygotować format xml, który byłby czymś w rodzaju wtyczki dla EduCASE. W tym pliku zapisywane byłyby translacje poszczególnych tworów języka UML na dany język programowania. Jest to o tyle ciekawe rozwiązanie, że znacząco przeciwdziała procesowi starzenia się aplikacji. Z doświadczenia wiadomo, że technologie przemijają i pojawiają się nowe. Dzięki takiej wtyczce po pojawieniu nowego, dominującego języka programowania pasjonat mógłby stworzyć wtyczkę, która by go obsługiwała. Dzięki temu EduCASE zdezaktualizowałby się dopiero bo wypuszczeniu nowej wersji UML, którego jednak nowe wersje pojawiają się zdecydowanie rzadziej niż w przypadku innych technologii.

6.10. Aplikacja internetowa

Obecnie coraz większą popularność zdobywają bogate aplikacje internetowe. Wśród nich wyróżniają się między innymi produkty firmy Google lub też plany organizacji Microsoft dotyczące przyszłości pakietu Office. W początkowej fazie analizy rozważano budowę EduCASE jako program webowy. Jednakże nieznajomość odpowiednich technologii mogłaby znacząco wydłużyć czas potrzebny na implementację. W przyszłości warto byłoby przepisać aplikację w technologii internetowej. A jeżeli jesteśmy przy rozwiązaniach firmy Sun, jedną z pierwszych propozycji byłoby wykorzystanie jednego z jej najnowszych pomysłów, a mianowicie JavaFX.

JavaFX to technologia stworzona przez firmę Sun Microsystems. Służy podobnie jak konkurencyjne Flash firmy Adobe, AJAX firmy Microsoft do tworzenia RIA – Rich Internet Application (bogatych aplikacji internetowych).

Strona domowa technologii JavaFX: http://javafx.com/

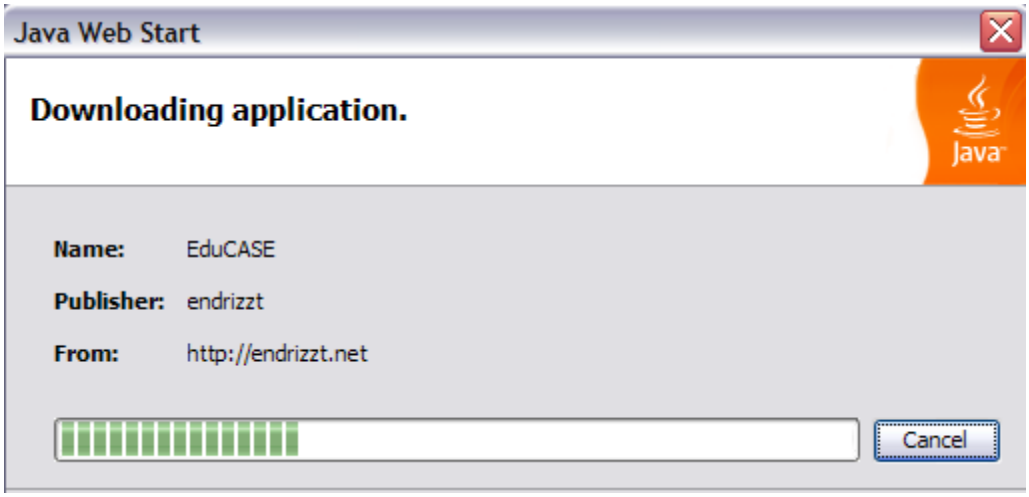
6.11.Generowanie diagramu z kodu źródłowego

EduCASE umożliwia generowanie kodu źródłowego, który odwzorowuje stworzony diagram klas. Sprzyja to procesowi błyskawicznego tworzenia oprogramowania. Na podstawie innych narzędzi CASE można jednak zauważyć odwrotną funkcjonalność. W niektórych aplikacjach możliwe jest odtworzenie diagramu klas z kodu źródłowego.

7. Instalacja i odinstalowanie EduCASE

7.1. Instalacja

Aby zainstalować aplikację EduCASE należy zapisać na dysku lokalnym plik spod adresu http://endrizzt.net/educase/educase.jnlp (dostęp ze strony http://endrizzt.net/educase/source.php). Po odpaleniu na lokalnym komputerze pliku jnlp ukazuje nam się okno Java Web Star, które pobiera z internetu najnowszą wersję oprogramowania.

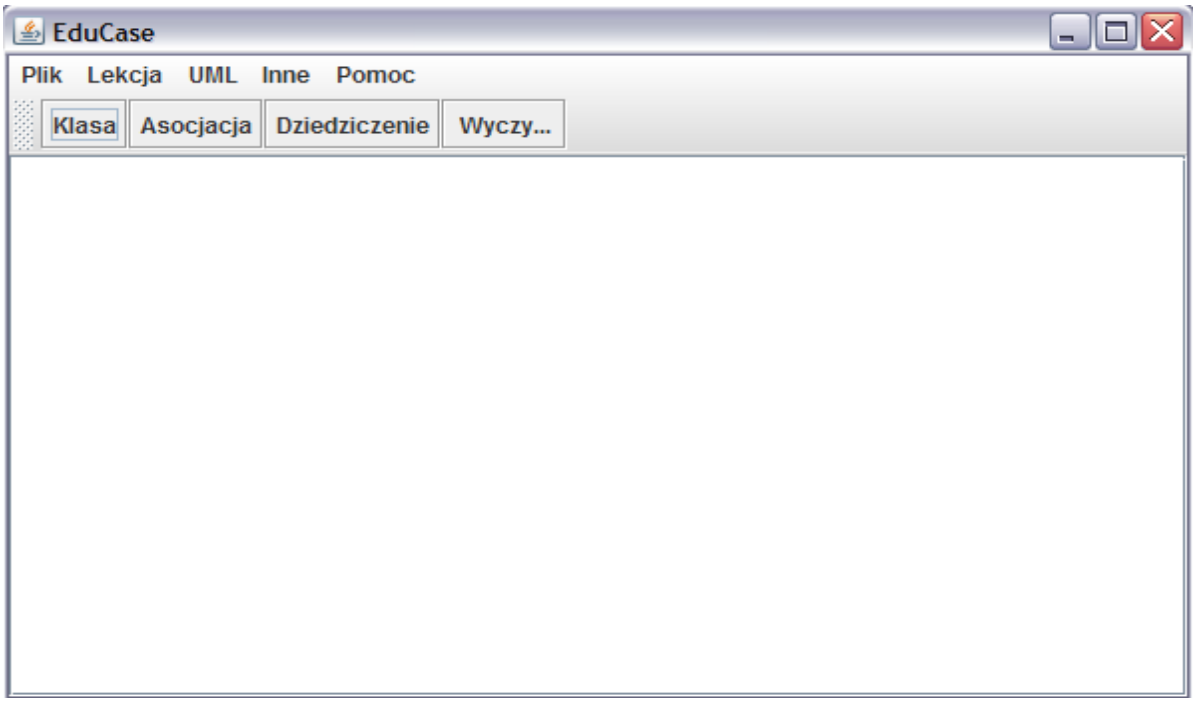


Rysunek 12:

Na początku mogą wystąpić ostrzeżenia bezpieczeństwa. Należy kliknąć przycisk Run.

Rysunek 13:

Po czym ukazuje nam się główne okno programu.

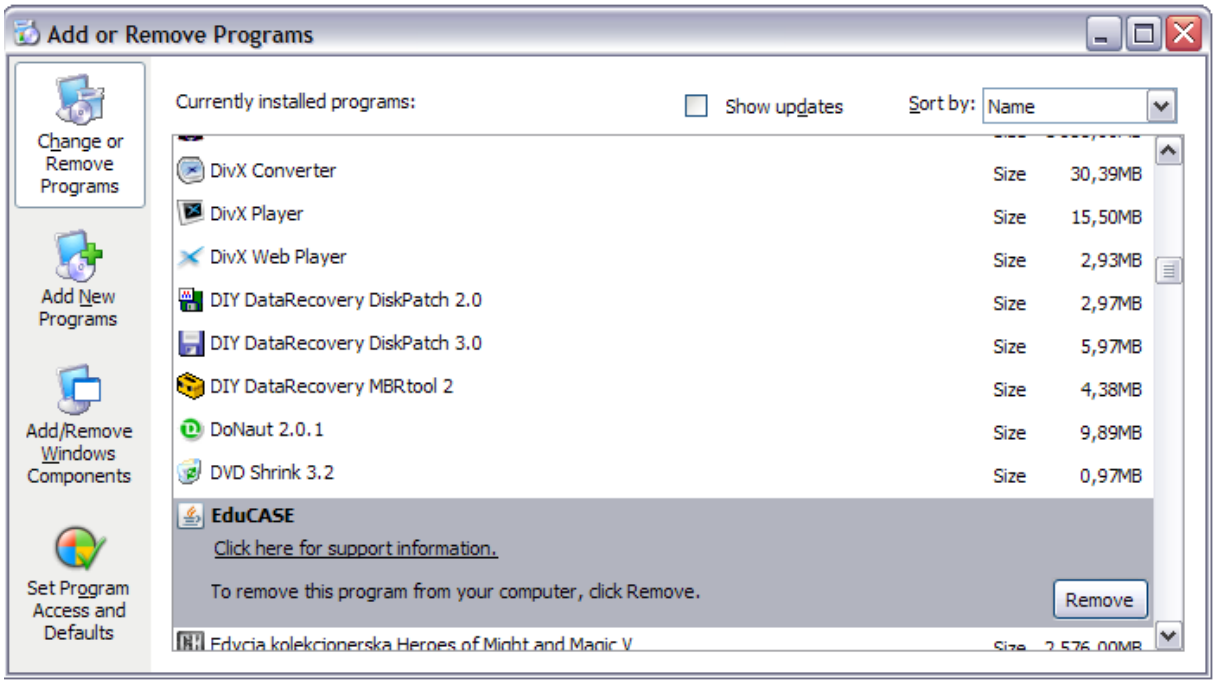


Rysunek 14:

Każdorazowe uruchomienie programu odbywa się za pomocą pliku jnlp.

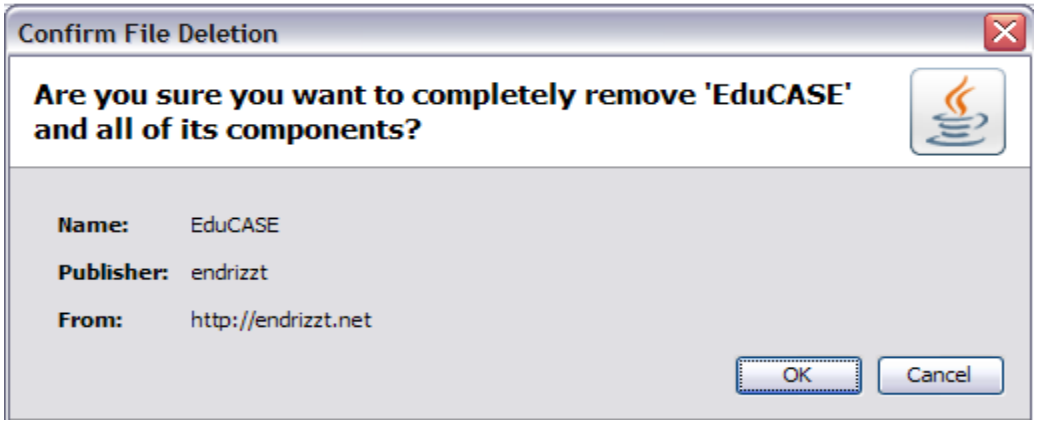
7.2. Odinstalowanie

Należy wybrać *Start/Ustawienia/Panel sterowania/Dodaj lub usuń programy*



Rysunek 15:

Wybrać aplikację EduCASE, kliknąć przycisk Usuń.



Rysunek 16:

Ostatnim krokiem jest kliknięcie Ok.

8. Dodatki

8.1. Tworzenie plików jar

Aby otrzymać plik educase.jar ze wszystkich klas znajdujących się w aktualnym katalogu wykonujemy polecenie:

```
jar cvf educase.jar *.class
```

Listing 12:

Parametry:

c – oznacza, że chcemy stworzyć plik JAR

v – wypisuje na konsoli nazwy wszystkich plików dołączanych do tworzonego pliku JAR

f – wyjściem dla polecenia jest plik, a nie standardowe wyjście

Następnie korzystamy z programu keytool, który jest narzędziem do zarządzania kluczami i certyfikatami. Keytool przechowuje klucze i certyfikaty w tak zwanym keystore. Korzystamy z polecenia:

```
keytool -genkey -alias endrizzt -keypass mypass -keystore
mykeys -storepass abc123
```

Listing 13:

Kolejnym krokiem jest użycie jarsigner, który generuje sygnatury dla plików JAR i weryfikuje sygnatury już podpisanych plików JAR.

```
jarsigner -keystore mykeys -storepass abc123 -keypass mypass
educase.jar endrizzt
```

Listing 14:

Parametry podobne do używanych przy korzystaniu z narzędzia keytool. Oznaczają kolejno:

keystore – specyfikuje ścieżkę do zlokalizowania keystore

storepass – hasło potrzebne, aby uzyskać dostęp do keystore

keypass – hasło chroniące klucz prywatny z wpisu z keystore odnoszący się do aliasu podanego w linii poleceń

Ostatnią rzeczą, którą możemy zrobić to sprawdzić poprawne podpisanie pliku JAR. Odbywa się to za pomocą polecenia:

```
jarsigner -verify -verbose educase.jar
```

Listing 15:

Podsumowanie

Zadaniem niniejszej pracy było przetestowanie rynku narzędzi CASE oraz aplikacji wspomagających edukację. Zarówno w jednym, jak i drugim obszarze można znaleźć wiele dopracowanych, profesjonalnych produktów. Skupiliśmy się na kilku z nich prezentując wady i zalety poszczególnych aplikacji. Zostały one jeszcze poddane dogłębnej analizie w celu wyszukania funkcjonalności, która mogłaby stać się nową jakością. Po wysegregowaniu najbardziej przydatnych właściwości powstał prototyp zwany EduCASE. Pozwala on na tworzenie diagramów klas UML, co jest esencją narzędzi do komputerowego projektowania oprogramowania. Pozwala na wygenerowanie kodu źródłowego – ważny element w teorii błyskawicznego tworzenia oprogramowania. Dodatkowo posiada szereg funkcji, ułatwiających użytkownikowi pierwsze spotkanie z diagramem klas, co było nieosiągalne w dotychczasowych produktach. Abstrahując od samego programu, powstała także strona internetowa znajdująca się pod adresem http://endrizzt.net/educase, która stanowi uzupełnienie pracy nad aplikacją.

Bibliografia

- http://office.microsoft.com
- http://www.gentleware.com/products.html
- http://www.w3schools.com
- http://www.w3c.org
- http://www.sun.com
- http://www.java.com
- http://www.omg.org
- http://pl.wikipedia.org
- http://www.php.net
- J. Płodzień, E. Stemposz – „Analiza i projektowanie systemów informatycznych”
- J. Cheesman, J. Daniels – „Komponenty w UML”