Zbigniew Łukasik, Małgorzata Górska

Weryfikacja zastosowania otwartego środowiska programistycznego na wybranym zadaniu transportowym

JEL: L99 DOI: 10.24136/atest.2019.151
Data zgłoszenia: 05.04.2019 Data akceptacji: 26.06.2019

Celem badawczym omówionym w artykule była weryfikacja – jakie środowisko programowania jest bardziej użyteczne w wybranym zadaniu transportowym. I stąd – chcąc ukazać trajektorie przemieszczających pojazdów na zadanych sieciach dróg, wybrano m.in. otwarte środowisko programistyczne, w odróżnieniu do uprzednio stosowanego – w badaniu podobnego zagadnienia – środowiska o odmiennej licencji. Realizacja postawionego zadania nastąpiła poprzez konsekwentne stosowanie paradygmatu programowania obiektowego.

Słowa kluczowe: transport drogowy, programowanie obiektowe.

Wstęp

Aby zdefiniować cel niniejszego opracowania niezbędne jest scharakteryzowanie struktury uprzednio opracowywanego systemu (Systemu Poszukiwania Pojazdów Drogowych Według Zadanych Cech - Retina). Składa się z trzech głównych komponentów: centrum obserwacyjnego, sieci telekomunikacyjnej oraz zbioru punktów obserwacyjnych (czujników). Dodatkowymi składowymi są sieć dróg oraz zbiór poszukiwanych pojazdów. Dane uzyskane z obserwacji mają służyć śledzeniu wyznaczonego obiektu. O ile jeden pojazd za pomocą sieci sensorów zarejestrowano dwa lub więcej razy – można wyznaczyć kierunek i szybkość postępu. Na tej podstawie system przedłuża przeszłą drogę, by zapowiedzieć jego przyszłą pozycję. O ile taka ekstrapolacja jest oceniana jako pewna, może zostać podjęta decyzja o ewentualnej interwencji. Niektóre elementy systemu były opisane w [2-11]. W niniejszej pracy badawczej celem jest analiza możliwości podnoszenia poziomu wdrażalności systemu Retina poprzez wykorzystanie wybranego środowiska programistycznego.

W artykule ogólnie scharakteryzowano autorski program symulacyjny, a następnie jego wybrane elementy. Na koniec przedstawiono wyniki przeprowadzonych badań.

1 Charakterystyka aplikacji

1.1.Ogólny schemat działania

Do celów badań stworzono własny program symulacyjny – aplikację w środowiskach Delphi i Lazarus. Ogólny schemat działania przedstawiono na Rys. 1.

Program – o roboczej nazwie RadoSCI (RADOm SCIentific simulations) – pozwala śledzić równolegle trajektorie ruchu dużej liczby wirtualnych jednostek kierowca/pojazd. Do częściowej kalibracji modelu przeprowadzono własne badania empiryczne opisane w [2, 5], a także oparto się na oficjalnie dostępnych danych statystycznych opisujących ruch na wybranej sieci dróg. W trakcie symulacji wygenerowano duże liczby punktów wyjścia i docelowych, następnie zasymulowano odpowiednie ślady ruchu, bazując na modelu zachowania kierowcy, który podejmuje decyzje o losowym rozkładzie skoncentrowanym wokół optymalnego (pod względem odległości) wyboru kolejnych odcinków trasy.

Przed przystąpieniem do prac związanych z tworzeniem opisanego programu symulacyjnego założono wykorzystanie algorytmów optymalizacyjnych typu punktu wewnętrznego (biblioteki IPOpt), lecz w trakcie realizacji pracy badawczej okazały się one jednak zbędne – przyjęto rozwiązania własne za wystarczające.

Przyjmuje się, że poszczególne trajektorie pojazdów nie będą ze sobą bezpośrednio skorelowane, jak w [1]. Stąd – nie ma potrzeby synchronicznych obliczeń, można przeprowadzać kalkulacje sekwencyjnie.

Rys. 1. Ogólna charakterystyka stworzonej aplikacji [opracowanie własne]

Ze zbioru wierzchołków grafu sieci dróg (omawianego w [17]), wraz z modelem obszaru, wybrany jest podzbiór obejmujący początki tras oraz równoliczny zbiór celów jazdy.

Następnie generuje się poruszającą się po sieci dróg flotę kierowców, od każdego punktu wyjścia do każdego punktu docelowego. Odbywa się to w ten sposób, że pojazd przemieszcza z aktualnego wierzchołka do jednego z sąsiadujących z nim wierzchołków. Aktualny czas jazdy oraz miejsce są zapisane w pliku wyjściowym do późniejszej analizy. Sprawdza się, czy cel jazdy został osiągnięty, i odpowiednio symulacja bieżącej trajektorii zostanie zakończona bądź kontynuowana, jak opisano w [19, 20].

Realnie każdy przebieg trasy, nawet przy identycznych punktach startu i celu, jest indywidualny ze względu na przyczyny losowe, takie jak przykładowo warunki pogodowe lub samopoczucie kierowcy. Czynniki te uwzględniane są w badaniach symulacyjnych, a model preferencji kierowcy omawiany w [7-9].

1.2.Elementy programu

Program umożliwia edycję parametrów opisujących preferencje kierowcy, generację i animację trajektorii pojazdów drogowych.

Projekt (*.dpr) ma postać jak na Rys.2:

```
program radosci;

uses
  Forms,
  unit_joy in 'unit_joy.pas' {Form_joy},
  unit_impex in 'unit_impex.pas',
  unit_scale in 'unit_scale.pas',
  unit_about in 'unit_about.pas' {Form_about},
  unit_sim in 'unit_sim.pas',
  unit_pref in 'unit_pref.pas' {Form_pref},
  unit_panel in 'unit_panel.pas' {Form_panel};

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TForm_joy, Form_joy);
  Application.CreateForm(TForm_about, Form_about);
  Application.CreateForm(TForm_pref, Form_pref);
  Application.CreateForm(TForm_panel, Form_panel);
  Application.Run;
end.
```

Rys. 2. Listing - projekt RadoSCI [opracowanie własne]

Wprowadzono manipulowanie grubością i kolorem krawędzi, tj. kolorami wszystkich kategorii dróg oraz wyłączanie zakrzywionych przebiegów ścieżek (przy animacji jest to opcja wskazana, ponieważ umożliwia zdecydowanie szybsze działanie aplikacji). Podobne zagadnienia opisano w [16-18].

Kod programu podzielono na moduły (unit) m.in.:

- `unit_joy` – główny moduł: obsługuje formę, sterowanie programem, menu,
- `unit_impex` odpowiada za import i export danych,
- `unit_scale` przelicza dane, np. współrzędne geograficzne na piksele,
- `unit_about` – wyświetla informacje o programie,
- `unit_sim` – obsługuje symulacje,
- `unit_pref` – ustawia właściwości wizualizacji.

Program wczytuje dane map. Obecnie – dla celów opisanych badań – kolekcja obejmuje kilka miast i regionów, a przede wszystkim obszar województwa mazowieckiego, razem z ścieżkami – śladami, które zostały wczytane z plików GPX (opis w [17]).

Zdefiniowano pliki *Vertex.txt i *Edge.txt o wspólnym początku nazwy – są one ze sobą skorelowane. W momencie importu jednego z nich, także automatycznie zostanie załadowany drugi. Trasy zaś wczytuje się osobno lub generuje losowo – w zależności od wyboru celu badania.

Uwzględniony jest przebieg tras miedzy węzłami oraz klasyfikacja dróg z podziałem na trzy kategorie.

Poniżej przedstawiono wybrane fragmenty kodu istotnego modułu unit_impex (import/export).

unit_impex

```
unit unit_impex;

interface

uses SysUtils, Dialogs;

const
n  = 12000;    // max length of a track

type intpair = array [1..2] of integer;
type pair    = array [1..2] of double;
type trip    = array [1..3] of double;

type intbitab   = array of intpair;
type intmonotab = array of integer;
type bitab      = array of pair;
type tritab     = array of trip;
type tabmotab   = array of intmonotab;
type tabbitab   = array of intbitab;
```

Rys. 3. Moduł unit_impex – definicja typów danych [opracowanie własne]

Informacje przedstawione na Rys. 3 dotyczą typów danych, które były niezbędne do zdefiniowania w programie. Zostało wprowadzone ograniczenie, które miało na celu przerwanie symulacji w przypadku, gry liczba kroków – segmentów drogi (12000) zostanie przekroczona. Następnie pary liczb niezbędne do opisu punktów w grafice pikselowej zostały zdefiniowane jako całkowite, a punkty i trasy realne jako liczby rzeczywiste. Klasy dróg - tabele jednowymiarowe oraz tabele współrzędnych (dwu- i trzy-kolumnowe) zostały również odpowiednio zadeklarowane.

unit_impex

```
var
statusstr : AnsiString ='';

track     : tritab;              // stores track data, polygon
V         : bitab;               // vertices, table with two columns, double
E         : intbitab;            // edges, table with two columns, integer
C         : bitab;               // contour vertices

Vpix      : intbitab;            // vertices, in pixels on googlemap
Epix      : intbitab;            // edges according to google
Opix      : intbitab;            // positions of photo radars

Typix     : intmonotab;          // types of edge
Pathes    : tabbitab;            // pathes of edges

hitype    : integer = 1;         // maximal street type on map

glatcen, gloncen : double;
glatfac, glonfac : double;

zoom             : integer;
reso             : integer;
```

Rys. 4. Moduł unit_impex – definicja zmiennych [opracowanie własne]

Na Rys.4 widoczny jest opis dotyczący zmiennych wykorzystanych do stworzenia programu RadoSCI. V – wierzchołki, E – krawędzie oraz C – kontury obszaru zainteresowania są wczytane jako domyślne, czyli są widoczne w momencie uruchomienia programu. Wierzchołki mają także swoje odpowiedniki w postaci elementów ciągłych – Vpix. Przybliżenie mapy (zoom) zostało określone – wzorując się na mapach Goole – jako zmienna całkowita.

unit_impex

```
function gpxread(filename   : string) : tritab;    // read directly gpx track
function trxtxtread(filename : string) : tritab;   // read text with track data

function Vread(filename : string) : bitab;         // read vertices, double, from txt
function Eread(filename : string) : intbitab;      // read roads, strains, from txt
function Cread(filename : string) : bitab;         // read closed curve double data

function Vpixread(filename : string) : intbitab;   // read output, Vertices
function Epixread(filename : string) : intbitab;   // read output, Edges

function trx2trxtxt(filename : string) : integer;  // convert trx track data to txt
procedure Ewrite(filename : string; E : intbitab); // export edge file - if edited

implementation
```

Rys. 4. Moduł unit_impex – definicja kluczowych funkcji [opracowanie własne]

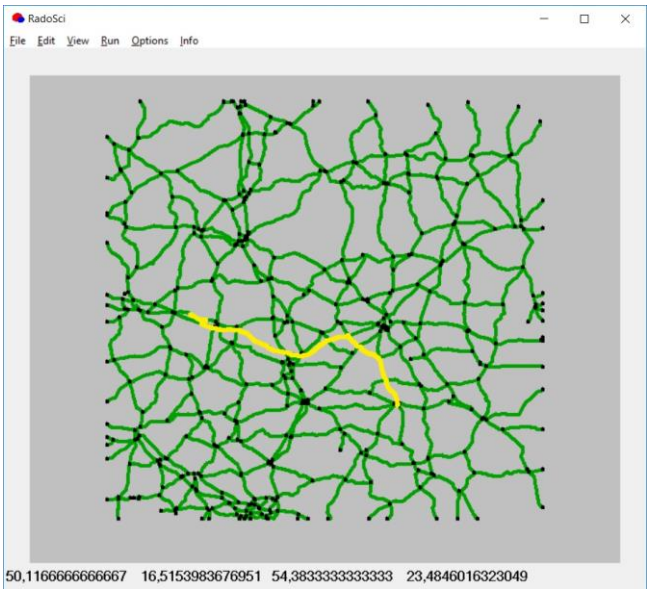
Funkcja gpxread pozwala na wczytanie do programu śladów zapisanych w formacie *.gpx, zaś trxtxtread importuje dane zapisane w formacie *.txt (ASCII). Następne funkcje wczytują wierzchołki oraz krawędzie grafów dotyczących dróg na wybranym terenie oraz – opcjonalnie – jego kontur. Program pozwala również na konwersję plików odnoszących się do trajektorii pojazdów na format *.txt.

2.Badania symulacyjne ruchu pojazdów na wybranej sieci dróg

W poprzednim rozdziale opisano autorski programu symulacyjny RadoSCI, który pozwala na monitorowanie trajektorii ruchu dużej liczby wirtualnych obiektów zainteresowania, a jednocześnie jest otwarty dla analizy obserwacji, przesyłu [15, 16] i interpretacji danych o ruchu pojazdów.

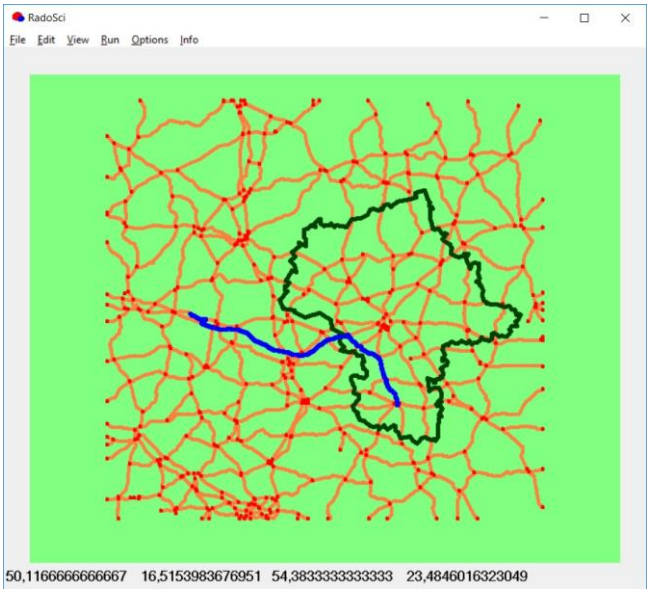
Poniższy rysunek (Rys. 6) przedstawia widok stworzonej aplikacji. Ustawienia domyślne zdefiniowano tak, aby zostały automatycznie wczytane wierzchołki i krawędzie grafów przedstawiających drogi na wybranym obszarze (zaznaczone kolorem zielonym), a

także zadaną trasę – wcześniej zarejestrowaną i zapisaną w formacie GPX.



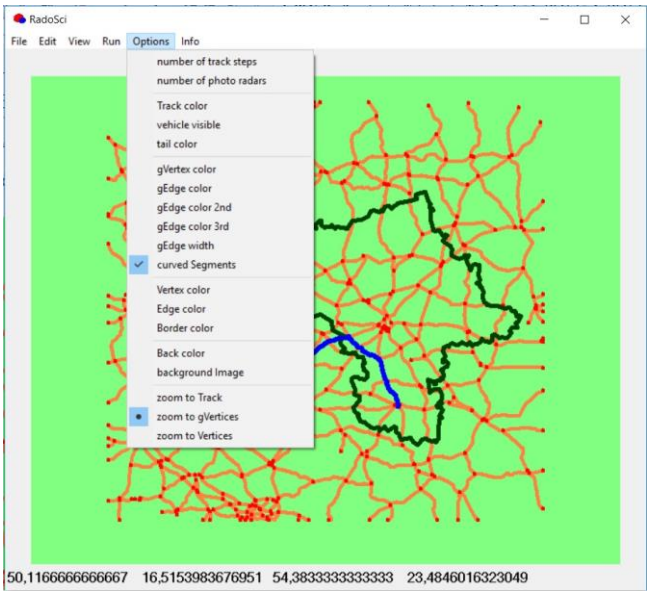
Rys. 6. Ustawienia domyślne programu symulacyjnego [opracowanie własne]

Wierzchołki, krawędzie i brzeg województwa są to pliki zawierające współrzędne geograficzne jako dane typu *double*, Rys.7. Są one zawsze wczytane do programu, ale ukazane jedynie po wybraniu z menu programu odpowiedniej pozycji.



Rys. 7. Wczytanie krawędzi województwa mazowieckiego [opracowanie własne]

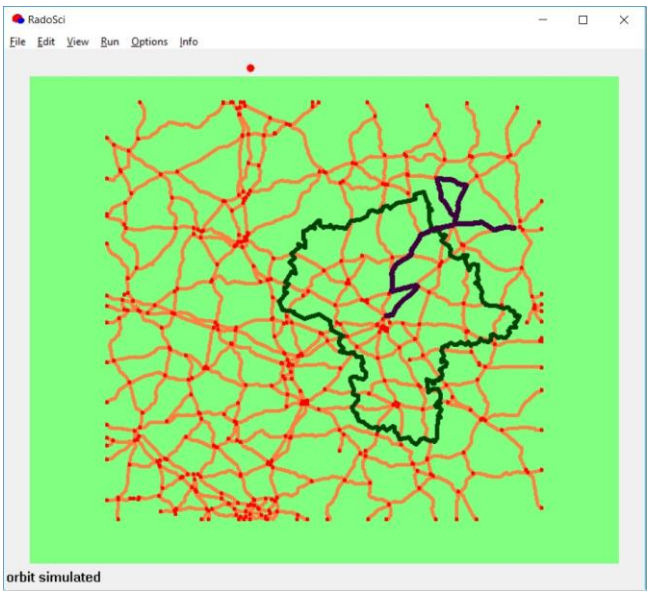
W menu *options* wybiera się także miedzy innymi kolory tła w aplikacji, kolory dróg oraz ich przecięcia, a także zarys województwa czy też trajektorii przemieszczającego się pojazdu.



Rys. 8. Menu 'options' aplikacji RadoSCI [opracowanie własne]

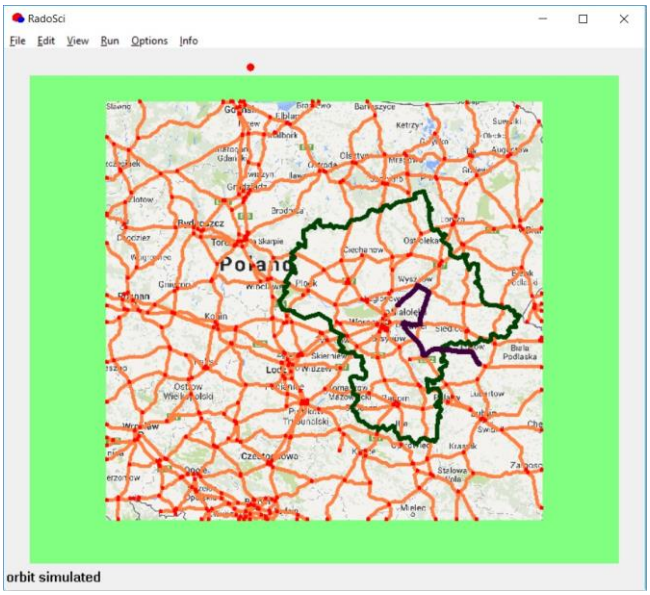
Rysunki 6, 7 oraz 8 przedstawiają jedną z dwóch możliwości wizualizacji trajektorii. Polega ona na wczytaniu do programu pliku GPX, który zawiera dane dotyczące zarejestrowanych tras. Na wymienionych grafikach ukazano konkretny ślad - drogę z Poznania do Radomia. Możliwa jest również animacja tychże tras.

Kolejną opcją jest możliwość zasymulowania – wedle ustalonych kryteriów – trajektorii przemieszczających się po drogach pojazdów, co przedstawia Rys. 9.



Rys. 9. Symulowanie trajektorii pojazdu drogowego [opracowanie własne]

Symulowana w programie RadoSCI trasa (zobrazowana kolorem fioletowym na Rys. 9) może przebiegać wedle różnych kryteriów np. może być całkowicie losowa lub mocno zdeterminowana. Preferencje „wirtualnego kierowcy" są możliwe do kalibracji. Trajektoria może przedstawiać zarówno kuriera, który przemieszcza się po określonym terenie, czasami wielokrotnie powielając niektóre odcinki trasy, a także osobę jadącą do pracy z Radomia do Warszawy, wybierającą optymalną – pod względem długości odcinków – trasę.



Rys. 10. Wczytywanie grafiki – tła zadanego obszaru

Opcjonalnie możliwe jest również wczytanie – po wybraniu odpowiednich opcji z menu programu – grafiki, która posłuży jako tło wybranego obszaru. Przygotowano kilka wybranych segmentów mapy – jak na Rys.10 fragmentu dotyczącego województwa mazowieckiego. Grafiki wykonano posługując się otwartymi źródłami informacji, jakimi są GoogleMaps. W powyższym przypadku, wybierając inaczej skalibrowany sektor – zmniejszając przybliżenie (zoom) sieć dróg zawierałaby znacznie mniej odcinków dróg, zaś powiększając zoom – konieczne by było wczytanie bardziej dokładnej sieci dróg, zawierającej większą liczbę wierzchołków i krawędzi grafów.

Podsumowanie

Realizacja postawionego w pracy badawczej zadania nastąpiła poprzez konsekwentne stosowanie paradygmatu programowania obiektowego. Do stworzenia opisanego programu symulacyjnego wybrano środowisko Delphi, a także otwarte środowisko programistyczne Lazarus. Aplikacja o roboczej nazwie RadoSCI pozwala przeprowadzenie obliczeń na podstawie danych dotyczących wielu obszarów Polski, w tym – województwa mazowieckiego.

Celem pośrednim badań symulacyjnych była weryfikacja – jakie środowisko programowania jest bardziej użyteczne w zadaniach transportowych. I stąd – chcąc ukazać trajektorie przemieszczających pojazdów na zadanych sieciach dróg, środowisko Lazarus zdecydowanie lepiej spełnia oczekiwania od (także często wykorzystywanego w zadaniach inżynieryjnych) języka MatLab. Przede wszystkim obiektowy język programowania pozwolił na znacznie szybsze generowanie symulacji, w porównaniu z prowadzonymi w ubiegłych latach na Wydziale Transportu i Elektrotechniki UTH Rad. zbliżonymi badaniami formułowanymi w języku MatLab.

Bibliografia

1. Chowdhury D., Santen L., Schadschneider A.: Statistical Physics of Vehicular Traffic and Some Related Systems, Physics Reports 329, str. 199-329, North-Holland, 2000
2. Górska M.: Application of dash-cams in road vehicle location systems, Czasopismo Autobusy – Technika, Eksploatacja, Systemy Transportowe, ISSN 1509- 5878, 12/2017
3. Górska M., Jackowski S.: Selected aspects of road vehicle localization, Czasopismo Autobusy – Technika, Eksploatacja, Systemy Transportowe, ISSN 1509- 5878, 12/2016
4. Górska M., Jackowski S.: Retina – a surveillance tool for road traffic, Czasopismo TTS Technika Transportu Szynowego vol. 12/2015,
5. ISSN 1232-3829
6. Górska M.: Analiza przydatności mobilnych źródeł informacji w systemie RETINA, Czasopismo Logistyka vol. 3/2015, ISSN 1231-5478
7. Górska M.: Podstawy modelowania sieci dróg w Systemie Poszukiwania Pojazdów Drogowych Według Zadanych Cech, Czasopismo Autobusy – Technika, Eksploatacja, Systemy Transportowe 3/2013
8. Górska M., Jackowski S.: Wybrane techniki sieciowej obserwacji ruchu pojazdu drogowego, Czasopismo TTS – Technika Transportu Szynowego 9/2012
9. Górska M., Jackowski J.: Analiza wybranych czynników wpływających na trajektorie ruchu poszukiwanych pojazdów, Czasopismo Logistyka 3/2012
10. Górska M.: Assessment of observer positions for given behavior of drivers, Czasopismo Logistyka 6/2011
11. Górska M.: Tracing multiple mobile objects by a network of intelligent detectors, Czasopismo Logistyka 3/2011
12. Górska M.: Course prediction for mobile object tracing network, Proceedings of the 4th International Interdisciplinary Technical Conference of Young Scientists, Poznań 2011
13. Jackowski S.: Telekomunikacja – część 1, Wydawnictwo Politechniki Radomskiej, Radom 2005
14. Jackowski S.: Telekomunikacja – część 2, Wydawnictwo Politechniki Radomskiej, Radom 2005
15. Jackowski S., Chrzan M.: Współczesne systemy telekomunikacyjne – część 1, Wydawnictwo Politechniki Radomskiej, Radom 2008
16. Jackowski S., Chrzan M.: Współczesne systemy telekomunikacyjne – część 2, Wydawnictwo Politechniki Radomskiej, Radom2008
17. Łukasik Z., Górska M.: Analysis of prediction methods applied to a selected road vehicle tracing system, Czasopismo Autobusy – Technika, Eksploatacja, Systemy Transportowe, ISSN 1509- 5878, 6/2018
18. Łukasik Z., Górska M.: The use of open geo-information bases for defining road nets in a given programming environment, Archives of Transport System Telematics, ISSN 1899-8208, str.46-52, 11/2018
19. Łukasik Z., Górska M.: Analiza probabilistyczna wirtualnego modelu ruchu pojazdów drogowych, Czasopismo Autobusy – Technika, Eksploatacja, Systemy Transportowe, ISSN 1509- 5878, 12/2018
20. Meng X., Chen J.: Moving objects management: models techniques and applications, Wydawnictwa Tsinghua University Press, Beijing oraz Springer-Verlag Berlin Heidelberg, 2010
21. Reveillac J.-M.: Modeling Road Traffic, in Modeling and Simulation of Logistics Flows 1: Theory and Fundamentals, Wydawnictwo: John Wiley & Sons, Nowy Jork 2017

Verification of the application an open programming environment on a selected transport task

The article presents results of a study of possibilities to improve the implementation of a road vehicle tracking system previously developed at WTiE UTH Rad. Consequent application of the object-oriented programming paradigm is a key requirement of the task. An intermediate aim of the research is the asseccment of available programming environments in the context of scientific problems in the field of transport.

Keywords: road transport, object-oriented programming

Autorzy:

prof. dr hab. inż. **Zbigniew Łukasik** – Uniwersytet Technologiczno-Humanistyczny im. K. Pułaskiego w Radomiu, Instytut Automatyki i Telematyki

dr inż. **Małgorzata Górska** – Uniwersytet Technologiczno-Humanistyczny im. K. Pułaskiego w Radomiu, Instytut Automatyki i Telematyki