Tomasz Ciszewski, Mieczysław Kornaszewski, Waldemar Nowakowski

Zastosowanie RailML do opisu kolejowych systemów nastawczych

JEL: L92 DOI: 10.24136/atest.2018.415
Data zgłoszenia: 19.11.2018 Data akceptacji: 15.12.2018

Pierwotna specyfikacja języka RailML pochodzi z 2002 roku i została zdefiniowana za pomocą XML. Pliki w standardzie RailML mogą służyć do przechowywania i wymiany danych pomiędzy licznymi interfejsami kolejowych systemów teleinformatycznych. Obecna wersja 2.4 obejmuje tabor, systemy zarządzania ruchem kolejowym, rozkładami jazdy, informacji pasażerskiej, oraz rezerwacji i sprzedaży biletów. W nadchodzącej wersji standardu RailML 3.1 dołączono podschemat <interlocking>, który przeznaczony jest do opisu kolejowych systemów nastawczych. Ponadto zapewniona zostanie zgodność ze zorientowanym obiektowo, abstrakcyjnym modelem danych infrastruktury kolejowej zdefiniowanym w standardzie UIC IRS 30100 (RailTopoModel, RTM). W artykule autorzy omawiają aktualny stan standardu RailML i jego perspektywy. Szczególną uwagę kładą na wykorzystanie RailML do opisu kolejowych systemów nastawczych. Autorzy przedstawiają również własne oprogramowanie umożliwiające pracę z plikami RailML.

Słowa kluczowe: RailML, XML, kolejowe systemy nastawcze, standardy wymiany danych, kolejowe systemy teleinformatyczne

Wprowadzenie

Nowoczesne systemy kolejowe powszechnie wykorzystują technologie informacyjne [9, 11, 19, 21, 22]. Mimo, że nie istnieją bariery technologiczne, wciąż nieliczne są standardy w zakresie mechanizmów reprezentacji danych i ich wymiany, co sprawia, że producenci systemów i urządzeń stosują w tym zakresie własne, firmowe rozwiązania [11]. Skutkuje to trudnościami we współpracy systemów pochodzących od różnych producentów i wymaga opracowania licznych interfejsów [4, 5, 9, 10, 12, 13, 14]. Sytuację tę starają się zmienić twórcy standardu kolejowego IRS 30100 (International Railway Standard, RailTopoModel) [5, 7, 24], którzy w postaci diagramów UML [25] opisali topologię i strukturę wielu elementów i systemów kolejowych. Pierwszą praktyczną implementacją tego standardu jest RailML 3.1 [15, 16, 17], którego zastosowanie pozwala rozwiązać wiele problemów w zakresie przechowywania i wymiany danych. Wcześniejsze wersje RailML [19] pozwalały kolejowym systemom IT wymieniać dane dotyczące taboru, zarządzania ruchem kolejowym, rozkładów jazdy, informacji pasażerskiej, jak również rezerwacji i sprzedaży biletów. Nowym elementem RailML jest podschemat <interlocking> służący do opisu kolejowych układów zależnościowych. W niniejszym artykule autorzy korzystając z autorskiego oprogramowania RailML Editor przygotowali przykłady definicji wybranych aspektów systemów zależnościowych za pomocą RailML 3.1 dla prostego systemu stacyjnego. W przykładach pokazano sposób definiowana wybranych zależności i sygnałów.

1.RailML jako bazujące na XML narzędzie do opisu systemów kolejowych

RailML (Railway Markup Language) to otwarty standard opracowywany od 2002 roku, początkowo przez grupę naukowców z niemieckiego Instytutu Fraunhofera ds. Systemów Transportowych i Infrastruktury w Dreźnie oraz Szwajcarskiego Federalnego Instytutu Technologii Instytutu Planowania i Systemów Transportowych [3, 4, 5]. Jako narzędzie mające na celu rozwiązanie problemu wymiany danych pomiędzy interfejsami systemów kolejowych poprzez ujednolicenie ich struktur uwzględnia nowoczesne standardy danych stosowane w technologiach informacyjnych – przede wszystkim XML (Extensible Markup Language) [3, 4, 5] – inny otwarty standard, który stworzony został przez konsorcjum W3C (World Wide Web Consortium) i jest powszechnie używany jako metajęzyk opisujący dane i pozwalający tworzyć różnorodne aplikacje XML. Przykładami takich aplikacji są m.in.: RSS, MathML, SVG, jak i oczywiście RailML. Zastosowano XML Schema Definition do zdefiniowania mechanizmu przestrzeni nazw języka, co zapewnia weryfikację poprawności dokumentów i łatwość ich rozbudowy przy zachowaniu wstecznej kompatybilności. Definicja określa również liczbę elementów podrzędnych, typy danych i wartości domyślne dla elementów i atrybutów [3, 4]. Dzięki cechom języka XML, które dziedziczą wszystkie jego aplikacje, w RailML jednocześnie przechowujemy dane i opisujemy ich strukturę, co jest niezwykle efektywne. Najnowszą obecnie wersją produkcyjną jest wersja 2.4, opublikowana w październiku 2018 roku. Publikacja wersji railML 3.1, która będzie uwzględniała standard RailTopoModel opracowany pod auspicjami UIC, zaplanowano na koniec 2018 r. [5, 23, 24].

Podobnie jak w wypadku innych dokumentów XML do definiowania zawartości dokumentu RailML używa się hierarchicznej struktury drzewiastej, w której korzeniem jest znacznik <railml>. Jest on rodzicem wszystkich wewnętrznych elementów, które hierarchicznie mogą zawierać podelementy (dzieci). Każdy element może zawierać predefiniowane atrybuty, w celu zapewnienia dokładniejszego opisu. W wersji deweloperskiej RailML 3.1 korzeń może zawierać cztery rodzaje podelementów: <infrastructure>, <rollingstock>, <timetable> i <interlocking>.

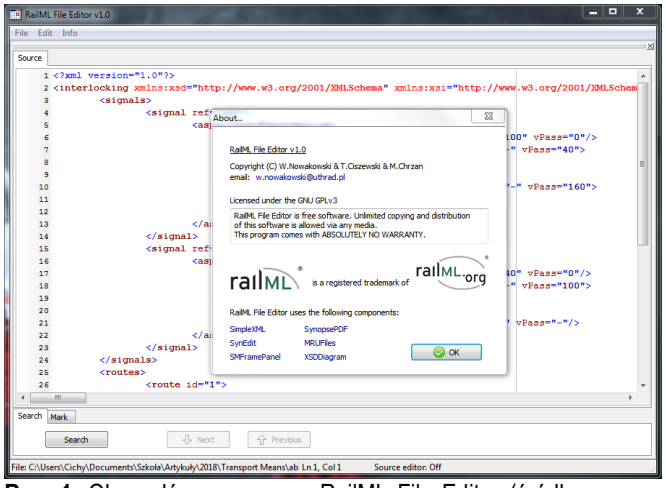
Każdy z nich definiuje swoje podelementy i ich atrybuty. Przykładowo podschemat <interlocking> zawiera między innymi następujące elementy [1, 2, 8, 14]: <signals>, <signal>, <signalAspects>, <signalAspectGroups>, <aspectSpeedDependencies>, <aspectSpeedDependency>, <routeGroups>, <routeGroup>, <routes>, <start>, <target>, </segments>, <elements>, <element>, <trackSection>, </derailerRef>, </levelCrossingRef>, <switchRef>, <switch>, <levelcrossingRef>, <levelcrossing>, <trainDetectorRef>, <trackCircuitBorder>, <flankProtection> , <flankElements>, <routePriority>, <interfaces>, <interface>.

2.RailML File Editor

Do prowadzenia prac będących przedmiotem rozważań w niniejszym opracowaniu autorzy zbudowali własne oprogramowanie railML File Editor (Rys. 1). To programowe narzędzie umożliwia wygodny podgląd, edycję i zapis plików railML (.xml) oraz XML Schema Definition (.xsd). Możliwy jest także eksport danych do formatów HTML, Rich Text Format (.rtf) i Adobe Portable Document Format (.pdf). Wygodny podgląd plików XML Schema pozwala na prezentację dokumentu w postaci drzewa oraz podgląd atrybutów i wartości wybranego elementu schematu. W trybie edycji edytor RailML koloruje składnię dokumentu, oraz umożliwia walidację zarówno dokumentów railML, jak i schematów XSD.

Podczas pracy z dokumentami XML Schema Definition oprogramowanie umożliwia współpracę z otwartoźródłowym programem

XSDDiagram [6], który służy do graficznej wizualizacji struktur dokumentów. Zbudowane autorskie oprogramowanie, jest wciąż intensywnie rozwijane i zostało wykorzystane przez autorów do przygotowania wszystkich dokumentów RailML, które będą prezentowane w dalszej części pracy.



Rys. 1. Okno główne program RailML File Editor (źródło: opracowanie własne)

3.Systemy nastawcze

Bezpieczne przemieszczanie się pojazdów szynowych jest realizowane przy użyciu systemów sterowania ruchem kolejowym (srk). Istotną grupę tych systemów stanowią systemy nastawcze, które sterują ruchem na pojedynczych posterunkach. Współczesne systemy nastawcze realizowane są w technice komputerowej, dlatego też w dalszych rozważaniach, system srk będzie oznaczać komputerowy system nastawczy srk [9, 20].

W kolejnictwie przyjęto zasadę dzielenia sieci kolejowej na odcinki i wyposażania ich w sygnalizatory, informujące maszynistę o możliwości dalszej jazdy. Odcinek taki nosi nazwę drogi jazdy i wraz z drogą zbliżania i drogą ochronną stanowi drogę przebiegu. Określenie dróg przebiegów na każdej stacji jest bardzo istotne dla procesu sterowania ruchem kolejowym. Podział na drogi przebiegów powinien być tak przeprowadzony, aby jednocześnie zapewnić jak najwięcej możliwych do realizacji jazd. W przypadku, gdy przebiegi nie mogą się równocześnie odbywać mówimy o przebiegach sprzecznych. Ma to zazwyczaj miejsce, gdy drogi jazd albo drogi ochronne pokrywają się lub krzyżują. Elementami drogi przebiegu są zwykle odcinki torowe i zwrotnicowe po których porusza się pojazd szynowy oraz sygnalizatory. Tak więc przez przebieg rozumie się zbiór uporządkowanych stanów, w jakich powinny znajdować się urządzenia sterowania ruchem kolejowym, które nastawiają, zabezpieczają i kontrolują drogę przebiegu. Systemami odpowiedzialnymi za nastawianie przebiegów są właśnie nastawcze systemy srk. Odpowiadają one za spełnienie następujących warunków bezpieczeństwa:

- wykluczenia sprzecznych dróg przebiegów,
- niezajętości odcinków torowych i zwrotnicowych,
- właściwego położenia zwrotnic (w tym również kontroli ich stanu w trakcie realizacji przebiegu),
- utwierdzania (unieruchomienia) elementów drogi przebiegu na czas jego realizacji,
- inne (powiązanie z blokadą liniową, urządzeniami zabezpieczenia przejazdu, itp.).

Spełnienie tych warunków wymagane jest do wyświetlenia sygnału zezwalającego na semaforze, umożliwiającego wjazd pojazdu szynowego na przygotowaną drogę jazdy. Pojazd szynowy przemieszczając się po drodze jazdy zajmuje i zwalnia kolejne elementy drogi kolejowej, czyli odcinki torowe i zwrotnicowe. Przejazd pojazdu szynowego jest kontrolowany (i odpowiednio wizualizowany) przez nastawczy system srk.

Podstawowymi dokumentami projektowymi systemu nastawczego srk są:

- plan schematyczny (rys. 2),
- tablica zależności (rys. 3).

Plan schematyczny urządzeń srk, sporządzony według określonych przepisami reguł, stanowi odwzorowanie układu torowego stacji i rozmieszczenia urządzeń srk, określa drogi przebiegów oraz podstawowe właściwości elementów tj.: rodzaje wskazań sygnalizatorów, lokalizację sygnalizatorów, numerację torów i zwrotnic itp. Plan schematyczny stanowi podstawę opracowania tablicy zależności, która jest istotnym składnikiem bezpieczeństwa zawierającym:

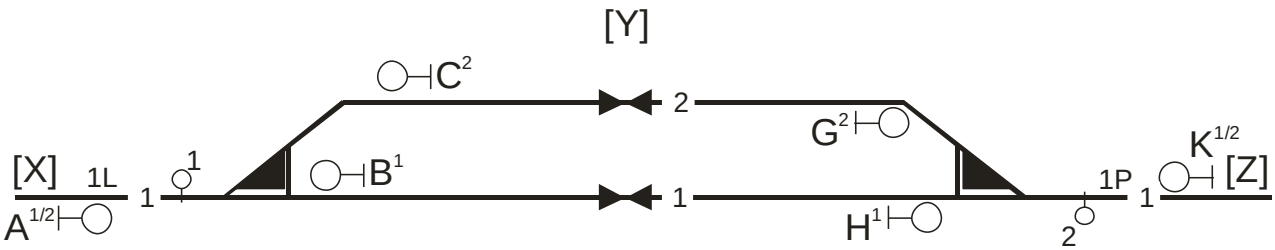
- oznaczenia (symbole) poszczególnych dróg przebiegów,
- wzajemne relacje między drogami przebiegów (sprzeczność, wykluczenie specjalne, brak sprzeczności),
- uporządkowane listy odcinków (torowych i zwrotnicowych) należących do każdej drogi przebiegu wraz z wymaganym ich stanem.

4.Opis systemów nastawczych za pomocą RailML3.1

By odzwierciedlić wybrane możliwości RailML w zakresie opisu systemów nastawczych rozważmy prosty układ torowy pokazany na rys. 2. Dla przejrzystości na tym planie schematycznym nie oznaczono odcinków kontroli niezajętości, a w powiązanej z planem tabeli zależności (rys. 3) nie uwzględniono przebiegów manewrowych. Pokazana tabela zależności ma formę stosowaną typowo w polskim kolejnictwie, a obowiązujące reguły bez trudu można adoptować do formatów stosowanych w innych kolejach europejskich [9].

W omawianym przykładzie zgodnie z regulacjami stosowanymi przez polskie koleje system sygnalizacji będzie wymagał przekazywania następujących sygnałów:

- dla semaforów $A^{1/2}$ i $K^{1/2}$ (rys. 4a)
  - prędkość na pierwszej drodze jazdy
    - stój,
    - V_{max} (jazda na wprost, tor 1),
    - 40km/h (jazda na odgałęzienie, tor 2),
  - prędkość na drugiej drodze jazdy
    - stój
    - 40km/h (jazda na odgałęzienie)
    - V_{max} (jazda na wprost)



Rys. 2. Przykładowy plan schematyczny (źródło: opracowanie własne)

- dla semaforów B^1 i H^1 (rys. 4b)
  - prędkość na pierwszej drodze jazdy
    - stój,
    - V_{max},
  - prędkość na drugiej drodze jazdy
    - stój
    - V_{max}
- dla semaforów C^2 i G^2 (rys. 4c)
  - prędkość na pierwszej drodze jazdy
    - stój,
    - 40km/h
  - prędkość na drugiej drodze jazdy
    - stój
    - Vmax

i użycia semaforów w konfiguracji pokazanej na rys. 4.

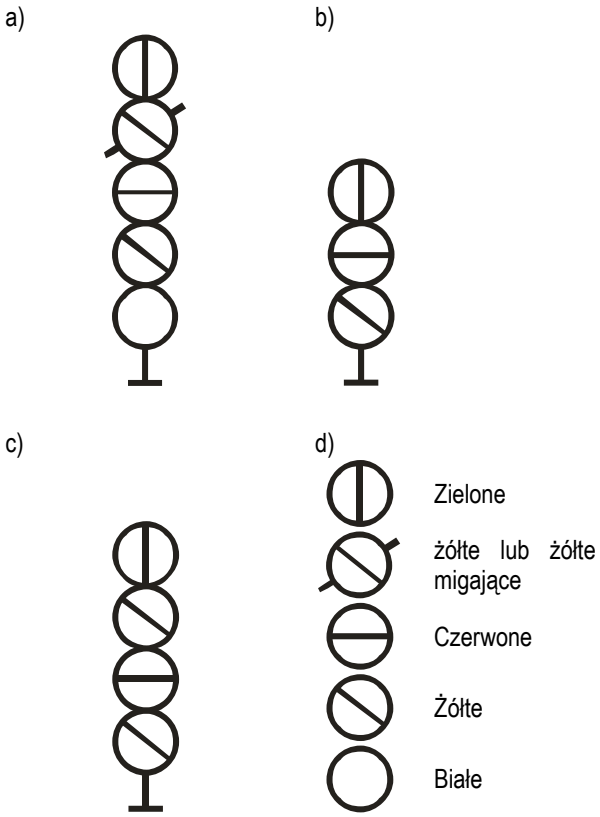
Opis przedstawionych sygnałów za pomocą RailML pokazano na rys. 5. W opisie tym dla uproszczenia przykładów pominięto możliwość jazdy na tzw. sygnał zastępczy, generowany w wypadku niemożności wypracowania właściwego sygnału.

Na rys. 6 pokazano fragmentarycznie formalny zapis zależności sygnałów dla układów zależnościowych bazujących na eliminacji przebiegów sprzecznych.

Wnioski

Kolejowe systemy komputerowe od lat korzystają z własnościowych i wzajemnie niekompatybilnych rozwiązań oferowanych przez producentów. Całościowe podejście do działalności kolejowej, pozwoliło opracować otwarte i coraz bardziej kompletne standardy modelowania topologii i opisu infrastruktury kolejowej, takie jak RailML i RailTopoModel. Ich kolejne wersje stopniowo zwiększają liczbę możliwych zastosowań. Wciąż rośnie też liczba praktycznych implementacji standardu. Istnienie wspólnego modelu danych ułatwia bowiem projektowanie oprogramowania, procesów i serwisów dla potrzeb kolei, a dzięki ujednoliceniu struktur danych, RailML, ma spore szanse stać się uniwersalnym formatem ich wymiany danych w transporcie kolejowym.

Zainspirowani tym trendem autorzy rozpoczęli prace nad własnym oprogramowaniem do pracy z plikami RailML. Powstały w wyniku podjętych prac RailML File Editor wciąż wymaga intensywnego rozwoju, jednak już teraz narzędzie to jest użyteczne dla użytkowników, którzy pracują z plikami railML i schematami XSD. Autorzy planują między innymi wyposażyć oprogramowanie w możliwość generowania i wizualnej edycji plików dla elementów infrastrukturalnych i zależnościowych.



Rys. 4. Konfiguracja semaforów dla przykładu z rys. 2. a) semafory $A^{1/2}$ i $K^{1/2}$; b) semafory B^1 i H^1; c) dla semaforów C^2 i G^2 d) objaśnienia (źródło: opracowanie własne)

Sygnały	Przebiegi	Przebiegi pociągowe								Zwrotnice
		A^1	A^2	B^1	C^2	G^2	H^1	K^1	K^2	1/2
A^1	[X] → Tor 1	-	+	+	+		+	+		+
A^2	[X] → Tor 2	+	-	+	+	+			+	-
B^1	Tor 1 → [X]	+	+	-	+			+		+
C^2	Tor 2 → [X]	+	+	+	-				+	-
G^2	Tor 2 → [Z]		+			-	+	+	+	-
H^1	Tor 1 → [Z]	+				+	-	+	+	+
K^1	[Z] → Tor 1	+		+		+	+	-	+	+
K^2	[Z] → Tor 2		+		+	+	+	+	-	-

Przebiegi

+	Przebiegi sprzeczne
+	Przebiegi sprzeczne wykluczony w sposób specjalny
-	Przekątna tablicy zależności

Zwrotnice

+	Zwrotnica zamknięta w przebiegu w położeniu zasadniczym
-	Zwrotnica zamknięta w przebiegu w położeniu przełożonym

Rys. 3 Tablica zależności dla przykładu z rys. 2 (źródło: opracowanie własne)

```
<signalAspects>
   <signalAspect="R" name="red" signalspeed="0" targetspeed="0"/>
   <signalAspect="YLFL" name ="yellow flashing" targetspeed="40"/>
   <signalAspect="YL" name="yellow" targetspeed="0"/>
   <signalAspect="GR" name="green" targetspeed="max"/>
   <signalAspect="YLGR" name="yellow green" signalspeed="40" targetspeed="vmax"/>
   <signalAspect="YLYLFL" name="yellow yellow flashing" signalspeed="40"
                                                        targetspeed="40"/>
   <signalAspect="YLYL" name="yellow yellow" signalspeed="40" targetspeed="0"/>
   <signalAspect="RW" name="red white" signalspeed="40"/>
</signalAspects>
<signalAspectGroups>
   <signalAspectGroup="1">
      <signalAspectRef="R"/>
      <signalAspectRef="YL"/>
      <signalAspectRef="YLFL"/>
      <signalAspectRef="GR"/>
      <signalAspectRef="YLYL"/>
      <signalAspectRef="YLYLFL"/>
      <signalAspectRef="YLGR"/>
   </signalAspectGroup>
   <signalAspectGroup="2">
      <signalAspectRef="R"/>
      <signalAspectRef="YL"/>
      <signalAspectRef="GR"/>
   </signalAspectGroup>
   <signalAspectGroup="3">
      <signalAspectRef="R"/>
      <signalAspectRef="YLGR"/>
      <signalAspectRef="YLYL"/>
   </signalAspectGroup>
  </signalAspectGroups>
```

Rys. 5. Opis konfiguracji semaforów i sygnałów w RailML (źródło: opracowanie własne)

```
<interlocking>
   <signals>
      <signal refid="A1/2" Cvps="Vmax" Cvns="Vmax">
            <signalAspectDependencies>
                  <signalAspectDependency code="R" Vp="Vmax" Vg="0">
                        <targetSignalTypeRef refid="G2"/>
                        <targetSignalTypeRef refid="H1"/>
                  </signalAspectDependency>
                  <signalAspectDependency code="YL" Vp="Vmax" Vg="40">
                        <targetSignalTypeRef refid="G2">
                              <segment refid="A1/2_G2"/>
                              <routePriorities/>
                        </targetSignalTypeRef>
                        <targetSignalTypeRef refid="H1">
                              <segment refid="A1/2_H1"/>
                              <routePriorities/>
                        </targetSignalTypeRef>
                  </signalAspectDependency>

                  (...)

            </signalAspectDependencies>
      </signal>
   </signals>
   <segments>
      <segment refid="A1/2_G2">
            <elements>
                  <signalRef/>
                  <trackSection/>
                  <switchRef/>
                  <crossingRef/>
                  <derailerRef/>
                  <trainDetectorRef/>
                  <levelCrossingRef/>
            </elements>
            <flankProtection/>
      </segment>

      (...)

   </segments>
  </interlocking>
```

Rys. 6. Fragment formalnego zapisu zależności sygnałów w RailML (źródło: opracowanie własne)

W niniejszym artykule proponowany edytor plików został użyty do opracowania przykładów opisu danych w stacyjnych systemach nastawczych. Autorzy bazowali na developerskich wersjach RailML, ponieważ produkcyjna wersja standardu nie jest jeszcze dostępna. Niemniej podjęte badania pokazały, że powstaje bardzo użyteczna technologia, która z czasem może zrewolucjonizować europejski rynek kolejowy w kolejnym obszarze, ułatwiając projektowanie i weryfikację stacyjnych systemów nastawczych.

Bibliografia:

1. Bosschaart M., Lean Engineering Design of Rail Interlocking Systems with RailML. Thesis Report Master TIL, Delft University of Technology, Delft, Nederland, 185p., 2013.
2. Bosschaart M., Quaglietta E., Janssen, B. i inni., Efficient formalization of railway interlocking data in RailML, Information Systems, Vol. 49: 126-141, 2015.
3. Ciszewski T., Nowakowski W., RailML as a tool for description of data model in railway traffic control systems. Transport Means - Proceedings of the International Conference, (2), pp. 935–940, 2018.
4. Ciszewski T., Nowakowski W., Interoperability of it systems in the international railways. Proceedings of the 16th International Scientific Conference Globalization and its socio-economic consequences, pp. 312-320, 2016.
5. Ciszewski T., Nowakowski W., Chrzan, M., RailTopoModel and RailML – data exchange standards in railway sector. Archives of Transport System Telematics, Vol.10 Issue 4 pp. 10-15, 2017.
6. Cosnier, R. XSD Diagram http://regis.cosnier.free.fr/?page=XSDDiagram
7. IRS 30100, RailTopoModel Railway infrastructure topological model, http://www.railtopomodel.org, 2016.
8. Khan S. A., Railway interlocking design support tools. Master's Thesis. University of Oslo. Norway, 90p. 2016.
9. Kornaszewski M., Chrzan M., Charakterystyka systemów kierowania i sterowania ruchem w kolejnictwie polskim. Technika Transportu Szynowego, (9), pp. 2573-2581, 2012.
10. Kornaszewski M., Chrzan M., Olczykowski Z., Implementation of New Solutions of Intelligent Transport Systems in Railway Transport in Poland. Smart Solutions in Today's Transport, pp. 282-292, 2017.
11. Kornaszewski M., Pniewski, R., Komputerowe wspomaganie procesu eksploatacji systemów srk. Logistyka, (6), pp. 5688-5695, 2014.
12. Lewiński A., Perzyńskiqa T., Toruń A., The Analysis of Open Transmission Standards in Railway Control and Management. Communications in Computer and Information Science. Vol.329 pp. 10-17, 2012.
13. Łukasik L., Ciszewski T., Wojciechowski J., Power supply safety of railway traffic control systems as a part of international transport safety. Proceedings of the 16th International Scientific Conference Globalization and Its Socio-Economic Consequences. Part III, pp. 1212-1219, 2016.
14. Łukasik Z., Nowakowski W., Application of TTCN-3 for testing of railway interlocking systems, Communications in Computer and Information Science. Vol. 104 pp. 447-454, 2010.
15. Łukasik Z., Nowakowski, W., Ciszewski, T., Extensible language for data description in diagnostic of traffic control systems. Prace Naukowe Politechniki Warszawskiej. Transport, (113), pp. 311–318, 2016.
16. Łukasik Z., Nowakowski W., Ciszewski T., Definition of data exchange standard for railway applications, Prace naukowe Politechniki Warszawskiej, (113), pp. 319-326, 2016.
17. Łukasik Z., Nowakowski W., Ciszewski T., Ujednolicenie struktur danych stosowanych w diagnostyce systemów sterowania ruchem kolejowym. Autobusy. Technika, Eksploatacja, Systemy Transportowe, (6), pp.995–998, 2016.
18. Maciejewski M., Zablocki, W., Basis of the Formalization and the Algorithmisation of the Control Functions in ATC Systems. Mikulski Jerzy (Eds.), Communications in Computer and Information Science. Vol. 104 pp. 253-262, 2010.
19. Nash A., Huerlimann D., Schuett, J., Krauss V. P., RailML – a standard data interface for railroad applications, Proceedings of the Ninth International Conference on Computer in Railways (Comprail IX), Dresden: 233-240, 2004.
20. Nowakowski W., Łukasik Z., Bojarczak P., Technical Safety in the process of globalization. Proceedings of the International Scientific Conference Globalization and its Socio-Economic Consequences. pp. 1571-1578, 2016.
21. Perzyński T., Wojciechowski J., Łukasik Z., Rail Transport Infrastructure on the Example of Level Crossing System. International Journal of Engineering & Technology (IJET), 7(4), pp.228–231, 2018.
22. Pniewski R., Metoda oceny bezpieczeństwa cyfrowych systemów automatyki kolejowej, Radom: Uniwersytet Technologiczno-Humanistyczny w Radomiu, 2013.
23. railML The XML interface for railway applications http://www.railml.org.
24. UIC. Feasibility study. UIC RailTopoModel and data exchange format. http://www.railtopomodel.org, 2013.
25. UML, http://www.omg.org/spec/UML/2.5

RailML application for description of railway interlocking systems

The original RailML specification comes from 2002 and was defined using XML. Files in the RailML standard can be used for storing and exchanging data between numerous interfaces of railway ICT systems. Current version 2.4 includes rolling stock and railway traffic management, timetables, passenger information as well as booking and selling tickets. In the upcoming version of the RailML 3.1 standard, the subscheme <interlocking> is included, which is intended for the description of railway interlocking systems. In addition, compliance with the object-oriented, abstract data model for the railway infrastructure, defined in the UIC IRS 30100 standard (RailTopoModel, RTM), will be ensured. In the paper, the authors discuss the current state of RailML and its perspective. Particular attention is paid to the use of RailML for the description of railway interlocking systems. The authors also present their proprietary software which allow to edit, visualize and verify RailML files.

Keywords: RailML, XML, interlocking, data exchange standards, railway IT systems

Autorzy:

dr inż. **Tomasz Ciszewski** – Uniwersytet Technologiczno-Humanistyczny im. Kazimierza Pułaskiego w Radomiu, Wydział Transportu i Elektrotechniki, t.ciszewski@uthrad.pl

dr hab. inż. **Mieczysław Kornaszewski** – Uniwersytet Technologiczno-Humanistyczny im. Kazimierza Pułaskiego w Radomiu, Wydział Transportu i Elektrotechniki, m.kornaszewski@uthrad.pl

dr hab. inż. **Waldemar Nowakowski** – Uniwersytet Technologiczno-Humanistyczny im. Kazimierza Pułaskiego w Radomiu, Wydział Transportu i Elektrotechniki, w.nowakowski@uthrad.pl