Nowoczesne metody wytwarzania oprogramowania systemów cyberfizycznych

Tomasz Szmuc[1], Rafał Mrówka[1], Paweł Skrzyński[1], Grzegorz Rogus[1], Wojciech Szmuc[1], Jacek Piwowarczyk[1], Marek Brańka[2], Jakub Ficoń[1]

[1] AGH Akademia-Górniczo-Hutnicza, Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej, Kraków
[2] OEConnection (OEC), Kraków

Streszczenie: Celem pracy jest przedstawienie nowoczesnych metod wspomagania wytwarzania oprogramowania systemów cyberfizycznych (SCF). Proponowane podejście bazuje na wprowadzeniu równoległej ścieżki formalnego modelowania i analizy właściwości w początkowych fazach wytwarzania oprogramowania. W dużym skrócie: artefakty modeli zapisanych w języku SysML lub AADL są tłumaczone na modele sieci Petriego, których właściwości mogą być analizowane bezpośrednio lub z zastosowaniem systemów dowodzenia logiki temporalnej. Dotychczasowe praktyczne zastosowanie metod formalnych ograniczone było złożonością generowanych modeli. W celu umożliwienia analizy złożonych struktur grafowych, tj. sieci Petriego, LTS (*Labeled Transition Systems*), drzew osiągalności, proponuje się zastosowanie wizualizacji 3D za pomocą urządzeń typu VR. Aktualne badania potwierdzają tezę, iż zastosowanie wizualizacji przestrzennej ułatwia interpretację, znajdowanie zależności, grupowanie, hierarchizację i dekompozycję wspomnianych struktur. Praktyczne wykorzystanie mechanizmu rzutowania grafów w przestrzeni 3D zostanie pokazane w zastosowaniu do wizualizacji sieci Petriego oraz na przykładzie systemu optymalizacji zarządzania współdzieleniem samochodów.

Słowa kluczowe: systemy cyberfizyczne, sieci Petriego, SysML, AADL, rzeczywistość wirtualna

MODERN METHODS IN SOFTWARE DEVELOPMENT OF CYBER-PHYSICAL SYSTEMS

Abstract: The paper aims to present modern methods supporting the development of Cyber-Physical Systems (CPS). The proposed approach is based on introducing a parallel path for formal modelling and feature analysis in the initial phases of software development. In a nutshell, artefacts written in SysML or AADL are translated into Petri net models, whose properties can be analysed directly or using temporal logic provers. Until now, the practical application of formal methods has been limited by the complexity of the generated models. 3D visualisation using VR devices is proposed to simplify the analysis of complex graph structures, i.e., Petri nets, LTS (Labeled Transition Systems), and reachability trees. Current research confirms the thesis that spatial visualisation facilitates interpretation, finding relationships, grouping, hierarchisation and decomposition. Practical use of the graph projection method in 3D space are shown in application to the visualisation of Petri nets and in the example of an optimisation system for car sharing management.

Keywords: Cyber-Physical Systems, Petri nets, SysML, AADL, virtual reality

https://doi.org/10.7494/978-83-67427-00-5_6

1. Wstęp

Systemy cyberfizyczne (SCF) są określone jako procesy obliczeniowe działające w ścisłej interakcji z fizycznym procesem (NSF 2006). Można zatem wyróżnić dwie powiązane dziedziny: warstwę fizyczną (procesy fizyczne, biologiczne itp.) oraz cyfrową. W warstwie cyfrowej realizowane są obliczenia na podstawie pomiarów/obserwacji procesów fizycznych. Wyniki tych obliczeń wpływają na procesy warstwy fizycznej. W tym ujęciu można zauważyć duże podobieństwo określenia SCF do istniejącej od wielu lat definicji systemów wbudowanych. Termin SCF pojawił się o wiele później w związku z gwałtownym rozwojem internetu rzeczy (*internet of things*), dużych zbiorów danych oraz zyskującą popularność koncepcją rzeczy inteligentnych (*smart things*). Innymi słowy SCF jest widziany raczej jako sieć kooperujących elementów.

Różnorodność oddziałujących procesów oraz rozszerzające się obszary zastosowań stawiają nowe wyzwania związane z konstrukcją systemów tej klasy. Dotyczy to modelowania, projektowania i implementacji oprogramowania oraz algorytmów przetwarzania często dużych zbiorów danych.

Prace grupy badawczej IMSEDA (Intelligent Methods in Software Engineering and Data Analysis) działającej w Katedrze Informatyki Stosowanej Wydziału EAIiIB koncentrują się w dwóch zasadniczych obszarach: modelowanie i analiza wspomagające implementację systemów SCF oraz nowe algorytmy analizy danych. W niniejszym artykule przedstawione są dwa wątki reprezentujące wymienione obszary: zastosowanie metod formalnych do modelowania i analizy oprogramowania oraz przykład systemu klasy *car sharing*, w którym zaimplementowano szereg algorytmów przetwarzających duże zbiory danych generowanych z IoT. W obu wątkach pokazano zastosowanie modelowania 3D. W pierwszym dotyczy to ułatwienia analizy grafów generowanych przez sieci Petriego, natomiast w drugim skupiono się na problematyce wizualizacji danych na potrzeby modułu inteligentnego rozmieszczenia pojazdów w celu zwiększenia dostępności usługi.

2. Modelowanie i analiza wspomagane metodami formalnymi

Implementacja systemów cyberfizycznych wymaga uwzględnienia specyfiki różnych dziedzin, co wiąże się zazwyczaj z koniecznością integracji różnorodnych modeli. W badaniach skupiono się na fazach analizy i projektowania systemów wbudowanych, będących podstawowymi elementami dla interakcji między opisywanymi wcześniej warstwami. Do konstrukcji modeli stosowane są dwa główne języki: SysML (Delligatti 2014) oraz AADL (Delange 2017). Języki te nie mają formalnej semantyki, stąd wykazanie poprawności poszczególnych modeli jest utrudnione. Jest to widoczne w przypadku systemów charakteryzujących się współbieżnością i dużą interakcją z otoczeniem, co zazwyczaj prowadzi do eksplozji kombinatorycznej wariantów, utrudniając

wykazanie poprawności przez testowanie. Udowodnienie poprawności struktur na poziomie projektu może być kluczowe dla powodzenia całego przedsięwzięcia.

W badaniach zaproponowano translację artefaktów wspomnianych języków na modele kolorowanych sieci Petriego (Jensen i Kristensen 2009). Umożliwia ona bezpośrednie dowodzenie własności z wykorzystaniem systemu CPN Tools (CPN Tools b.d.) lub potraktowanie utworzonej sieci (dokładniej jej grafu osiągalności) jako modelu, w którym bada się spełnialność żądanych właściwości zapisanych w postaci formuł logiki temporalnej (Szmuc T. i Szmuc W. 2020a, 2020b).

2.1. Zastosowanie języka SysML

SysML (*System Modelling Language*) jest językiem modelowania systemów wbudowanych, szeroko stosownym m.in. w przemyśle samochodowym. W podrozdziale zawarto krótkie wprowadzenie do tego języka oraz pokazano przykład translacji fragmentu scenariusza diagramu sekwencji.

Wprowadzenie do języka SysML

SysML wywodzi się z języka UML 2.0. Został utworzony i jest rozwijany przez Object Management Group. Język ten umożliwia specyfikację, projektowanie i analizę różnorodnych systemów, szczególnie tzw. *systems of systems.* SysML oferuje dziewięć typów diagramów. Cztery z nich, diagramy: przypadków użycia, sekwencji, maszyny stanowej i pakietów, są przeniesione z UML 2.0. Diagram czynności (*activity diagram*) jest rozszerzony, natomiast cztery poniższe są istotnie różne/nowe:

1) diagramy wymagań (RD – *requirements diagrams*) specyfikują wymagania oraz relacje z pozostałymi komponentami;
2) diagramy definiowania bloków (BDD – *block definition diagrams*) opisują relacje między blokami, np. *composition*, *association*, *specialization*;
3) diagramy bloków wewnętrznych (IBD – *internal block diagrams*) specyfikują wewnętrzną strukturę w kategoriach ich własności i połączeń;
4) diagramy parametryczne (PR – *parametric diagrams*) stosowane są do określenia ograniczeń (równań) między właściwościami liczbowymi.

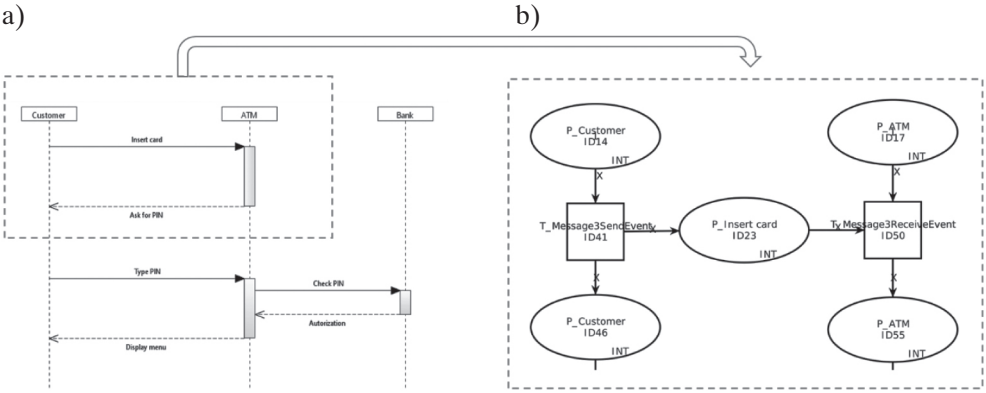
Dokładny opis SysML 1.4 oraz przykłady zastosowań można znaleźć w książce Delligattiego (2014). Aktualna wersja 1.6 jest opisana na stronie Object Management Group (OMG SysML 2019).

Translacja artefaktów SysML w modele CPN

Język SysML oferuje narzędzia do integracji modeli z różnych dziedzin i opisujących system z różnych perspektyw. W dotychczasowych badaniach skupiono się na modelach związanych z wytwarzaniem oprogramowania. Opracowano większość (z wyjątkiem diagramów PR) reguł translacji SysML w CPN (Szmuc W. 2014, Szmuc W. i Szmuc T. 2016a, 2016b).

Blok jest podstawowym elementem struktury aplikacji. Diagramy BDD opisują relacje między blokami, natomiast IBD ich wewnętrzną strukturę. Reguły translacji dla diagramów BDD i IBD są podobne. Modele CPN są tworzone przez rzutowanie bloków na przejścia podstawiane, gdzie odpowiednie miejsca są bramkami do strony niższego poziomu (Szmuc W. i Szmuc T. 2018, Szmuc T. i Szmuc W. 2020b).

W niniejszym podrozdziale opisano przykładową translację diagramu sekwencji na model CPN. Na wybór takiego tematu miały wpływ ograniczenia objętości artykułu oraz dążenie do pokazania w miarę reprezentatywnej translacji. Przykład diagramu sekwencji dla bankomatu oraz translację jego pojedynczej transakcji pokazano na rysunku 1.



Rys. 1. Diagram sekwencji dla bankomatu (a) oraz wynik translacji pojedynczej transakcji (b)
Źródło: opracowanie własne na podstawie Papyrus (b.d.) i Kałduś (2019)

Poniżej opisano reguły translacji diagramów sekwencji w CPN. Dokładny opis zawarty jest w publikacji W. Szmuca (2014).

1. Każda linia życia rozpoczyna się miejscem o nazwie odpowiadającej aktorowi/blokowi.
2. Przesłanie komunikatu jest modelowane przez miejsce umieszczone między przejściami w odpowiednich liniach życia.
3. Między kolejnymi przejściami w danej linii życia umieszcza się miejsca symbolizujące dany punkt na linii życia, jak również umożliwiające modelowanie aspektów czasowych.
4. Komunikaty znalezione/zgubione są modelowane odpowiednio przez pary miejsce wejściowe – przejście oraz przejście – miejsce wyjściowe.

Sieć pokazana na rysunku 1b jest wynikiem translacji wykonanej przez system SysML2CPN (Kałduś 2019). Jest to translator napisany w języku Java i może być zainstalowany jako wtyczka w środowisku Eclipse. Diagramy SysML są tworzone i edytowane z zastosowaniem systemu Papyrus (Papyrus b.d.), natomiast modele sieci Petriego są analizowane i wizualizowane z zastosowaniem CPN Tools (CPN Tools b.d.). Systemy

te stosują język XML do specyfikacji odpowiednich grafów. Prototypowa wersja systemu SysML2CPN (Kałduś 2019) umożliwia automatyczną translację sześciu diagramów: wymagań (RD), przypadków użycia, definicji bloków (BDD), bloków wewnętrznych (IBD), aktywności i sekwencji.

2.2. Zastosowanie języka AADL

AADL (*Architecture Analysis and Design Language*) jest przeznaczony głównie do fazy projektowania, lecz dostarcza dodatkowych możliwości analizy aspektów czasowych, bezpieczeństwa i zużycia energii. Język ten jest często stosowany przy wytwarzaniu oprogramowania systemów transportu lądowego, powietrznego i aplikacji militarnych.

Wprowadzenie do języka AADL

AADL został zaprojektowany jako język wspomagający modelowanie i weryfikację systemów czasu rzeczywistego. W odróżnieniu od SysML AADL skupia się na fazie projektowania, lecz integruje również modele utworzone przez inne języki modelowania, np. wygenerowane przez Mathworks®Simulink. Podstawowym środowiskiem tworzenia modeli i analizy jest Osate (Osate b.d.) – wtyczka w środowisku Eclipse. Osate wraz z modułem Ocarina umożliwia budowanie modeli (wersja tekstowa i graficzna), analizę ograniczeń czasowych i bezpieczeństwa (*safety*) oraz generację kodu w językach C, Ada i Java.

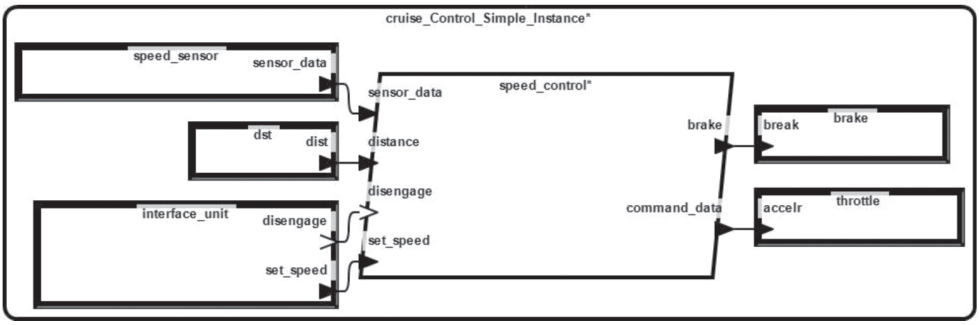
Podstawowymi elementami strukturalnymi są komponenty, czyli bloki połączone różnego rodzaju konektorami. Wyróżnia się trzy klasy komponentów, ich rodzaj jest określony przez słowa kluczowe poniżej zapisane pogrubioną czcionką:

1) Software: **process, thread, thread group, subprogram, subprogram group, data.**
2) Hardware: **device, processor, memory, bus.**
3) Integrator: **system.**

Zgodnie z ogólną zasadą systematycznego modelowania i analizy wskazać można następujące etapy postępowania:

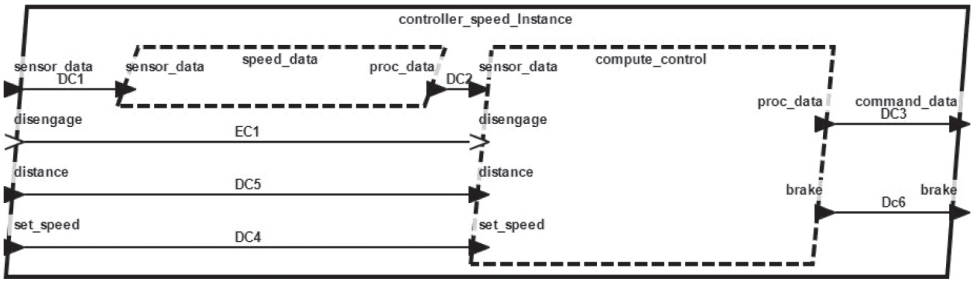
1. Specyfikacja otoczenia systemu przez deklaracje urządzeń i ich implementacji.
2. Ogólny opis struktury oprogramowania – procesy, podsystemy i ich połączenia z urządzeniami.
3. Uściślenia ogólnych opisów – wątki, podprogramy, kanały komunikacyjne.
4. Definicja sprzętu – procesory, pamięci, magistrale.
5. Określenie parametrów czasowych, reguł szeregowania i zarządzania zasobami.
6. Analiza bezpieczeństwa.
7. Wstawienie kodu do ram AADL i generacja kodu aplikacji.

Na rysunku 2 pokazano diagram AADL prostego tempomatu. System składa się z procesu sterującego (utrzymanie zadanej prędkości), trzech urządzeń wejściowych (*speed_sensor*, *dst*, *interface _unit*) i dwóch wyjściowych (*brake*, *throttle*).



Rys. 2. Model tempomatu w języku AADL
Źródło: opracowanie własne z wykorzystaniem Osate (b.d.)

Doprecyzowany model procesu *speed_control* przedstawiono na rysunku 3.



Rys. 3. Model procesu sterującego
Źródło: opracowanie własne z wykorzystaniem Osate (b.d.)

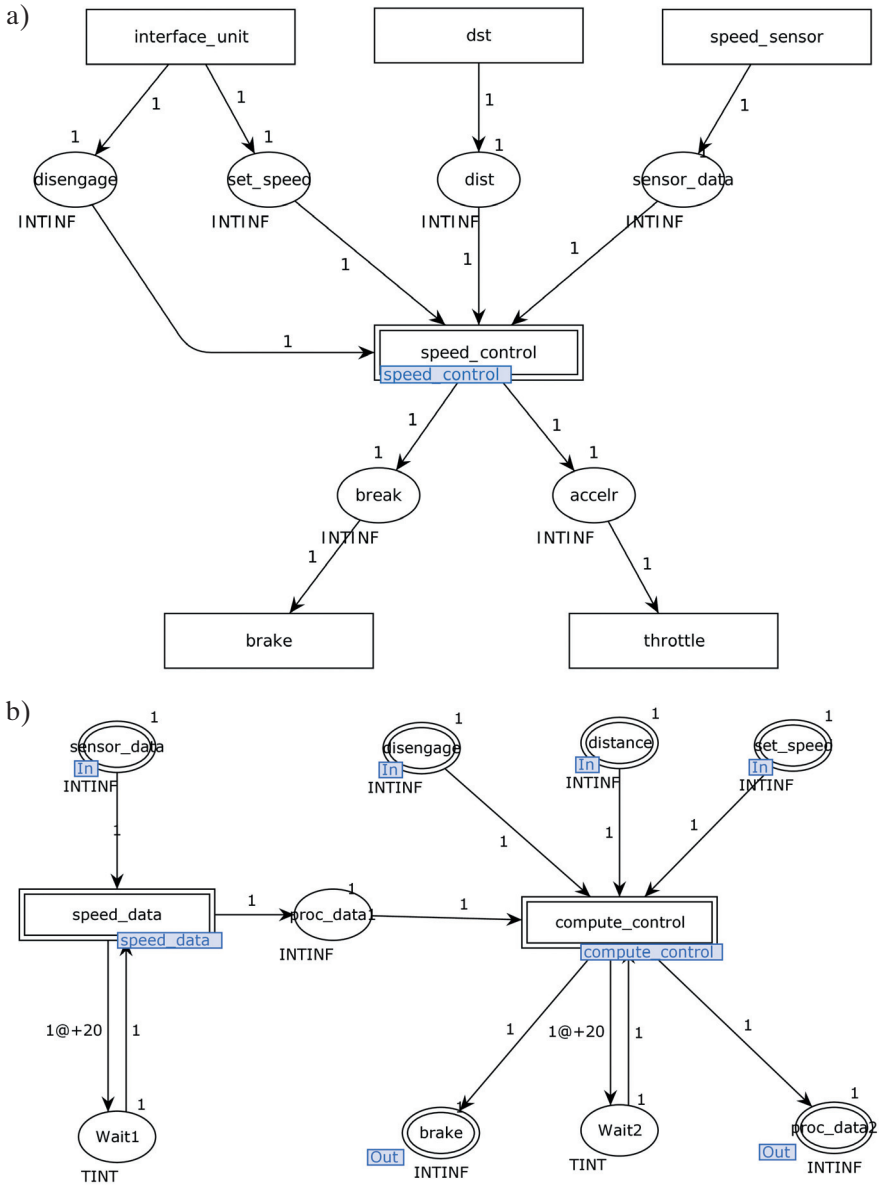
Proces sterujący składa się z dwóch wątków: *speed-data* i *compute_control*. Pierwszy przetwarza wstępnie dane z czujnika prędkości, a drugi oblicza decyzje sterujące dla przepustnicy i hamulca.

Translacja artefaktów AADL w modele CPN

AADL jest językiem strukturalnym, zatem translacja na hierarchiczne kolorowane sieci Petriego zachowuje w większości przypadków strukturę komponentów AADL. Ta obserwacja pozwala zastosować ogólną regułę translacji dla głównych komponentów AADL. Każdy proces, wątek, podprogram, urządzenie (*device*) może być rzutowany na odpowiednie przejście podstawiane (*substituted transition*), w którym miejsca gniazdowe (*socket places*) odpowiadają portom i/o (Szmuc T. i Szmuc W. 2020a).

Translacja komponentów AADL w modele CPN jest realizowana z zastosowaniem systemu AADL2CPN (Łukowicz 2020). Podobnie jak w przypadku SysML translacja jest wykonywana automatycznie, wymaga jednak ręcznego poprawienia rozmieszczenia elementów sieci. Wynik translacji modeli z rysunków 2 i 3 przedstawiono na rysunku 4. W sieci opisującej tempomat (rys. 4a) urządzenia są przedstawione jako zwykłe przejścia, gdyż w modelu AADL nie zdefiniowano struktury wewnętrznej tych

urządzeń. Odpowiednie uzupełnienie spowoduje ich zastąpienie przez przejścia podstawiane. Umieszczane na łukach (rys. 4b) pieczątki czasowe są wynikiem translacji sekcji **properties**, w której wyspecyfikowano cykliczne (okres 20 ms) wykonywanie wątku obliczającego sterowanie.



Rys. 4. Modele wygenerowane po translacji:
a) model tempomatu CPN; b) model procesu sterującego CPN
Źródło: opracowanie własne na podstawie Łukowicz (2020)

2.3. Analiza własności modeli CPN

Modele CPN będące wynikiem rzutowania artefaktów SysML lub AADL mogą być analizowane w domenie sieci Petriego z użyciem CPN Tools (CPN Tools b.d.). Umożliwia to badanie klasycznych właściwości sieci Petriego: osiągalności specyficznych stanów (np. zakleszczenia), powtórnego osiągania stanu (*home property*), bezpieczeństwa (*safety*).

Wymieniony obszar weryfikacji właściwości może być rozszerzony przez zdefiniowanie wymagań w postaci formuł logiki temporalnej i sprawdzanie spełnialności tych formuł w wygenerowanych modelach CPN (Szmuc T. i Szmuc W. 2020a, 2020b). Idea polega na generacji LTS (*labeled transition tree*) przez CPN Tools, które po przeformatowaniu (PetriNet2ModelChecker b.d., Szpyrka i in. 2016) będzie modelem wejściowym do systemu weryfikacji modelowej (nuXmv model checker b.d.).

3. Rzutowanie i analiza w przestrzeni 3D

Złożoność struktur grafowych (LTS w przypadku sieci Petriego) wymusza poszukiwania metod i narzędzi wspomagających grupowanie, dekompozycję czy odkrywanie zależności między fragmentami grafów. Propozycje rzutowania i analizy w przestrzeni 3D są przedstawione w odniesieniu do sieci Petriego oraz do optymalizacji rozmieszczenia pojazdów w systemie *car sharing*.

3.1. Zastosowanie technik wirtualizacji przestrzeni w inżynierii oprogramowania

Obecnie techniki wirtualizacji przestrzeni są powszechnie stosowanymi narzędziami wsparcia w inżynierii kosmicznej, motoryzacyjnej, lotniczej, medycynie, farmacji oraz architekturze. Dzięki modelowaniu 3D oraz możliwości interakcji z elementami modelu proces weryfikacji projektu jest znacznie tańszy i bardziej efektywny. Wirtualna rzeczywistość w tym zakresie nie tylko ułatwia zrozumienie struktur przestrzennie złożonych dzięki intuicyjnej ich prezentacji. Pozwala także na wizualizację procesów związanych z obliczeniami na tychże strukturach, jak również manipulowaniem ich elementami składowymi. Sukces w innych obszarach zachęca do zastosowania tych technik także w inżynierii oprogramowania.

Nowoczesne metody formalne stosowane w inżynierii oprogramowania operują na złożonych strukturach danych, reprezentowanych najczęściej w formie grafów lub hipergrafów (sieci Petriego, logiki temporalne, algebry procesów). W ogólnym przypadku własności modeli formalnych oprogramowania weryfikowane są na grafie stanów osiągalnych. Z tego względu zaproponowano, aby metody wirtualizacji przestrzennej w pierwszej kolejności zastosować do wizualizacji samych grafów. W celu

umożliwienia prezentacji grafu w sposób łatwy do analizy przez człowieka zaproponowana została metoda wprowadzająca hierarchizację grafu, oparta na algorytmach partycjonowania. Co istotne, założono, że wirtualizacja struktur grafowych powinna nie tylko pozwolić na ich analizę wizualną. Winna stanowić także podstawę do zaproponowania technik partycjonowania grafów, wspieranej przez sztuczne sieci neuronowe. Opracowane środowisko wirtualizacji struktur grafowych pozwala na ręczną manipulację wygenerowaną automatycznie hierarchią grafu. Dane zebrane podczas wykorzystania wspomnianego środowiska stanowią część zbioru uczącego dla opracowywanej sieci neuronowej służącej do partycjonowania grafów.

3.2. Algorytm rzutowania grafów w przestrzeni 3D

Głównym celem wprowadzenia technologii wirtualizacji przestrzeni 3D jest uzyskanie możliwości efektywnego prezentowania struktur grafowych o dowolnych rozmiarach. Metoda polegająca na generowaniu pojedynczego grafu o dowolnie dużej liczbie węzłów, które następnie użytkownik może oglądać, poruszając się w wirtualnej przestrzeni, musi zostać odrzucona jako niepraktyczna i nieefektywna. Proponuje się zastosowanie algorytmu wizualizacji wykorzystującej algorytmy partycjonowania grafów. W takim algorytmie użytkownik ma możliwość zmiany parametrów wpływających na jego działanie oraz manualnej zmiany przyporządkowania pojedynczego węzła lub partycji do tworzonego podgrafu. Podobny mechanizm stosuje się w hierarchicznych sieciach Petriego. W sieciach tych modelowanie zależności hierarchicznych (partycjonowanie) wykonywane jest manualnie przez projektanta sieci.

Partycjonowanie grafów pozwala na redukcję złożoności i bardziej intuicyjną wizualizację wybranych fragmentów modelu przez kapsułkowanie wybranych jego obszarów. Szukany jest optymalny sposób, by podzielić węzły grafu na partycje, przy jednoczesnym zminimalizowaniu liczby przecięć krawędzi pomiędzy partycjami. Dla grafu $G = (V, E)$ definiujemy partycje jako dwa zbiory rozłączne A, B spełniające warunki $A \cup B = V$ oraz $A \cap B = \emptyset$. Istnieje wiele algorytmów partycjonowania grafów, tj. *spectral partitioning* (Donath i Hoffman 1972), *graph growing* (Karypis i Kumar 1999), *flows* (Ford i Fulkerson 1956), *geometric partitioning* (Miller in. 1991), *streaming graph partitioning* (Stanton i Kliot 2012). W proponowanym rozwiązaniu zastosowano algorytm *normalized cuts* (Shi i Malik 2020). Pojęcie znormalizowanego cięcia grafu definiujemy jako:

$$N\mathrm{cut}(A,B) = \frac{\sum_{u\in A, v\in B} w(u,v)}{\mathrm{assoc}(A,V)} + \frac{\sum_{u\in A, v\in B} w(u,v)}{\mathrm{assoc}(B,V)},$$

gdzie $\mathrm{assoc}(A,V) = \sum_{u\in A, t\in V} w(u,t)$.

Algorytm wykorzystuje własności wektorów własnych macierzy Laplace'a grafu (Pothen i in. 1990). Badana jest miara podobieństwa dwóch węzłów w celu określenia ich przynależności do pojedynczej partycji. Problem sprowadza się do wyznaczenia rozwiązania wyrażenia:

$$\min_x N\mathrm{cut}(x) = \min_y \frac{y^T(D-W)y}{y^T D y},$$

gdzie D jest macierzą diagonalną $N \times N$ ($N = |V|$) oraz W jest macierzą symetryczną $N \times N$ i obie spełniają warunek $D^{-1/2}(D - W)D^{-1/2} = \mathcal{L}$. $\mathcal{L}$ jest macierzą Laplace'a grafu G. Podejście to jest metodą grupowania widmowego, gdzie węzły grafu to elementy, a wagi na krawędziach grafu łączące dwa węzły to mierzone podobieństwa między odpowiadającymi sobie elementami.

Proponowany algorytm wizualizacji składa się z następujących kroków:

1. Scena 3D początkowo jest dzielona na siatkę sześcienną 3 × 3 × 3. Przyjmujemy sześcian wewnętrzny o współrzędnych początkowych (0, 0, 0).
2. Dla danego grafu $G = (V, E)$ wykonujemy partycjonowanie (z wykorzystaniem algorytmu *NormilizedCut(V)*) na dwa zbiory (partycje) lub korzystamy z manualnego podziału grafu na struktury podrzędne (np. sieć Petriego).
3. Wyznaczamy partycję centralną przez wyliczenie $\max_{A,B}(\mathrm{assoc}(A, V), \mathrm{assoc}(B, V))$.
4. Partycję centralną umieszczamy w siatce o współrzędnych (0, 0, 0). Kolejną partycję umieszczamy zgodnie z formułą:

$$(x_{i+1}, y_{i+1}, z_{i+1}) = \begin{Bmatrix} x_i + 1, \text{ jeżli } i = \{1, 4, 6, 7, \ldots\}, \text{ inaczej } x_i \\ y_i + 1, \text{ jeżli } i = \{2, 5, 8, 11, \ldots\}, \text{ inaczej } y_i \\ z_i + 1, \text{ jeżli } i = \{3, 6, 9, 12, \ldots\}, \text{ inaczej } z_i \end{Bmatrix}$$

   dla $(x_0, y_0, z_0) = (-1, -1, -1)$.
5. Dodawanie nowej partycji jest powiązane z podziałem zajmowanej siatki na dwa identyczne sześciany. Dodanie sześcianu odbywa się przez powiększenie jednego wymiaru siatki o wartość 1.
6. Partycjonowanie grafu i podziały siatki kończymy po osiągnięciu wyznaczonej przez użytkownika granicy.

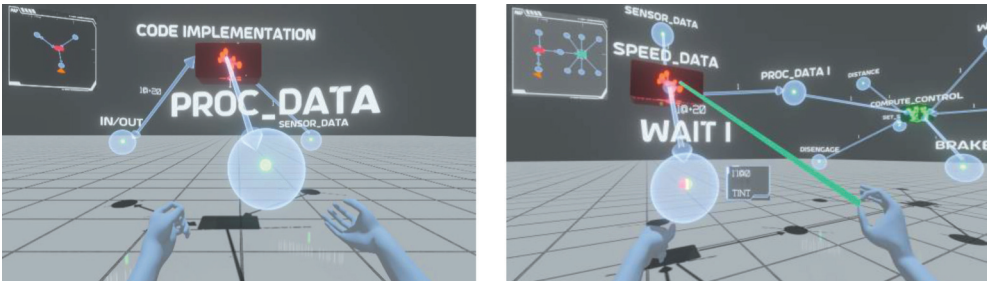
3.3. Wizualizacja i analiza sieci Petriego

Pomimo często nieskomplikowanej graficznej reprezentacji sieci Petriego analiza badanego systemu oraz interpretacja jego zachowania może być trudna. Istnieje wiele

opracowań podchodzących do zagadnienia wizualizacji sieci Petriego w przestrzeni dwuwymiarowej. Najbardziej typowe jest zastosowanie animacji ikon, wykorzystanie grafik wspomagających pozwalających na uchwycenie dynamicznego charakteru zmian zachodzących w badanym systemie. CPN Tools, oprócz wizualizacji, pozwala także na przeprowadzanie symulacji oraz weryfikację właściwości modelu za pomocą analizy przestrzeni stanów. Nową jakość w tym zakresie wprowadzają jednak możliwości związane z wirtualizacją przestrzeni 3D.

Rozszerzenie modelu wizualizacji sieci Petriego o trzeci wymiar jest realizowane przez wykorzystanie elementów standardu VRML (*Virtual Reality Modelling Language*) (Nadeau 1999), który systematyzuje zagadnienie budowy wirtualnych światów 3D, tworzenia interaktywnych scen i wykresów, a następnie nawigowanie nimi w czasie rzeczywistym. Otwiera to przed użytkownikiem nowe możliwości eksploracji wybranych obszarów sieci i analizowania ich szczegółowo bez utraty kontekstu całościowego modelu. W niniejszej pracy zaproponowano autorski model wizualizacji 3D sieci Petriego bazujący na algorytmie opisanym w podrozdziale 3.2. W sieciach Petriego nie musimy stosować partycjonowania, ponieważ zakładamy, że podział na poszczególne poziomy hierarchii został przeprowadzony przez projektanta. Wynik działania wizualizacji (rys. 5) zaprezentowany został na podstawie modelu tempomatu (rys. 4).

a) b)



Rys. 5. Wyniki działania oprogramowania do wizualizacji modelu tempomatu: a) widok procesu nadrzędnego; b) szczegóły podprocesu PROC_DATA

W zakresie wizualizacji 3D analizie poddano dwa poziomy modelu: system i proces. Na podstawie testów użyteczności zaproponowano notację 3D podstawowych elementów składowych, pozwalającą na ich prezentację graficzną z uwzględnieniem połączeń między nimi oraz informacji dodatkowych w postaci etykiet. Podstawowym wyzwaniem tego etapu jest zapewnienie efektywnego sposobu nawigacji po całości modelu, pozwalającego na ergonomiczne przemieszczanie się pomiędzy widokiem ogólnym oraz szczegółami poszczególnych obszarów i elementami w nich zawartymi. W tym celu wprowadzono okienko nawigacyjne, które pozwala użytkownikowi zachować kontekst całości bez względu na to, w której części modelu znajduje się w danym

momencie. W zastosowanym podejściu użytkownik z łatwością może obejrzeć wybrany obszar modelu, obracając lub powiększając go dowolnie.

Dzięki uwzględnieniu hierarchizacji przejść możliwe jest przeniesienie użytkownika do poszczególnych scen odpowiadających za dany poziom hierarchii. Można także wejść w szczegóły wybranych elementów, by zobaczyć szczegóły samej implementacji w języku ML.

3.4. Wizualizacja i analiza map rozmieszczenia pojazdów

Kolejnym przykładem zastosowania wirtualizacji przestrzeni 3D w inżynierii oprogramowania jest wizualizacja problemu optymalizacji rozmieszczenia pojazdów w systemie *car sharing* (Loose 2010, Shaheen i Cohen 2012, Jorge i Correia 2013). Głównym celem tej koncepcji jest zmiana podejścia do użytkowania samochodu i odejście od modelu, w którym samochód jest postrzegany jako dobro na rzecz traktowania go jako dzielonego narzędzia (zasobu) do wykonania usługi. Polska należy do pionierów wprowadzania takich rozwiązań na rynek. Przykładem jest system/usługa Traficar[1], wprowadzony przez firmę Express (największa wypożyczalnia samochodów w Polsce) w październiku 2016 roku. Platforma została stworzona i jest rozwijana we współpracy z AGH (grupa badawcza IMSEDA).

Kluczem do sukcesu systemów typu *car sharing* jest dostępność usługi, a w szczególności lokalizacji dostępnych pojazdów. Zwiększenie stopnia dostępności wymaga określenia przyszłego zapotrzebowania na pojazdy w danym miejscu i czasie. Optymalizacja rozmieszczenia pojazdów jest wersją problemu komiwojażera i należy do klasy problemów NP-trudnych. Na podstawie zadanych kryteriów, takich jak: dzień tygodnia, godzina odjazdu, pora roku, aktywności użytkowników, można przewidzieć przyszłe zapotrzebowania w danym miejscu. Narzędzie, które udostępni systemom typu *car sharing* możliwość badania miejsc największego zapotrzebowania, stałoby się jednym z kluczowych elementów dla każdego systemu i przyczyniłoby się do znacznego wzrostu dostępności współdzielenia samochodów. Badanie możliwości zwiększenia dostępności usługi dzięki zastosowaniu metod sztucznej inteligencji jest przedmiotem projektu realizowanego w grupie badawczej IMSEDA w ramach bonu innowacyjnego w ramach 1 Osi Priorytetowej Gospodarka Wiedzy Regionalnego Programu Operacyjnego Województwa Małopolskiego na lata 2014–2020 o numerze RPMN.01.02.03-12-0119/21-00.

Przygotowywana w ramach projektu usługa relokacyjna[2] bazuje m.in. na analizie podaż-popyt, która następnie jest zwizualizowana za pomocą zaproponowanego algo-

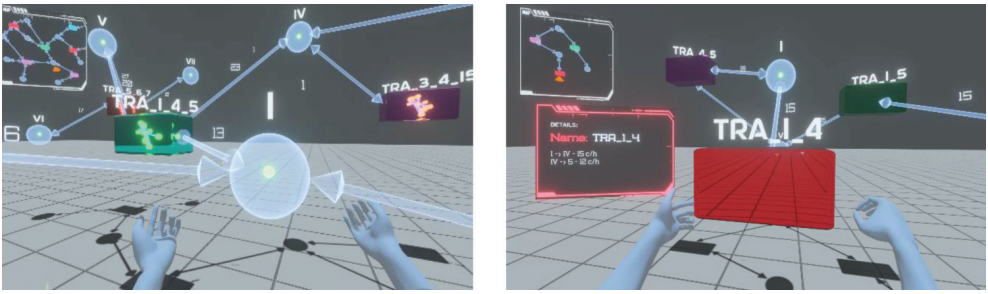
[1] Więcej informacji na stronie głównej usługi: https://www.traficar.pl.
[2] Wyniki badań i opis zastosowanego podejścia będą przedmiotem osobnego artykułu.

rytmu. W tym celu opracowano model podaż-popyt w postaci sieci Petriego zgodnie z poniższymi założeniami:

1. Modelowaną lokalizację (miasto Kraków) podzielono na odrębne obszary.
2. Obszary mają różną wielkość oraz nieregularny kształt.
3. Miejsca sieci reprezentują poszczególne obszary.
4. Wyróżniamy dwa kolory znaczników: pojazd (podaż) oraz zapotrzebowanie na pojazd (popyt).
5. Model sieci wykorzystuje hierarchię przejść – kolejne poziomy hierarchii sieci odpowiadają coraz mniejszym obszarom, na które dzielona jest lokalizacja.
6. Krawędzie reprezentują trasy, którymi można podróżować pomiędzy dwoma sąsiadującymi obszarami (miejscami).

Tak utworzony model podaży-popytu zasilono danymi historycznymi, co umożliwiło przeprowadzenie symulacji rozmieszczenia pojazdów stosownie do zapotrzebowania. Dane historyczne mają granulację 5-minutową. Dzięki zastosowaniu wizualizacji 3D (rys. 6) dokonano analizy efektywności wybranych algorytmów rozmieszczenia pojazdów. Własność określaną jako „sprawiedliwość" (*fairness*) zdefiniowano jako równomierne ilościowe rozłożenie znaczników popytu i podaży we wszystkich miejscach na każdym poziomie hierarchii.

a) b)



Rys. 6. Wyniki działania oprogramowania wizualizacji modelu podaży i popytu: a) widok procesu nadrzędnego; b) szczegółowa wizualizacja procesu TRA_1_4_5.

4. Podsumowanie

W artykule przedstawiono zastosowanie nowoczesnych metod wytwarzania oprogramowania na przykładzie systemów cyberfizycznych. Podstawowym założeniem stosowania opisanych metod jest możliwość dokonywania automatycznej translacji pomiędzy modelami oprogramowania. Umożliwia to analizę i dowodzenie własności w fazie projektowania systemu. Bardzo ważnym elementem procesu jest możliwość

wizualizacji modeli oraz ich analiza. Zaproponowano wykorzystanie wirtualizacji przestrzeni 3D jako narzędzia wizualizacji i symulacji. Opracowano algorytm prezentacji bardzo złożonych struktur grafowych z zastosowaniem partycjonowania i hierarchizacji. Wyniki prac zaprezentowano na przykładzie dwóch modeli sieci Petriego: tempomatu oraz modelu popytu i podaży w systemie *car sharing*. Wartością dodaną dla projektanta wykorzystującego proponowane rozwiązanie będzie jednoczesna prezentacja przejścia wraz ze wskazaniem zmiany w grafie stanów osiągalnych. Projektant obserwujący wykonywanie przejść w sieci Petriego będzie miał jednoczesny podgląd na to, jak zmienia się stan modelu w wyniku tej akcji. Będzie to miało istotne znaczenie w procesie partycjonowania grafu. Pozwoli również na określenie, które miejsca i przejścia w sieci mogą zostać zgrupowane ze względu na równoważność (z punktu widzenia badanych własności modelu) możliwych oznakowań tych miejsc w sieci. Dodatkowo weryfikując model w grafie stanów osiągalnych, projektant z łatwością dokona identyfikacji, które konkretne przejścia w sieci są powiązane ze stanami niespełniającymi badanej własności. Aktualnie trwają prace nad rozszerzeniem opracowanego narzędzia o powyższe funkcjonalności. Docelowo oprogramowanie powinno pozwolić na analizę bardzo złożonych struktur grafowych stosowanych w inżynierii oprogramowania. Ze względu na dynamiczny charakter prac ich graficzne wyniki będą sukcesywnie publikowane na platformie YouTube.

Literatura

CPN Tools, b.d., http://cpntools.org/ [dostęp: 14.04.2022].

Delange J., 2017, *AADL in Practice: Design and Validate the Architecture of Critical Systems*, Reblochon Development Company, USA.

Delligatti L., 2014, *SysML Distilled: A Brief Guide to the Systems Modelling Language*, Addison-Wesley Professional, USA.

Donath W.E., Hoffman A.J., 1972, *Algorithms for Partitioning of Graphs and Computer Logic Based on Eigenvectors of Connection Matrices*, IBM Technical Disclosure Bulletin, 15(3), s. 938–944.

Ford L.R., Fulkerson D.R., 1956, *Maximal Flow through a Network*, Canadian Journal of Mathematics, vol. 8, no 3, s. 399–404. https://doi.org/10.4153/CJM-1956-045-5.

Jensen K., Kristensen L.M., 2009, *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*, Springer, Heidelberg.

Jorge D., Correia G., 2013, *Carsharing systems demand estimation and defined operations: a literature review*, European Journal of Transport and Infrastructure Research, vol. 13(3), s. 201–220. https://doi.org/10.18757/ejtir.2013.13.3.2999.

Kałduś M., 2019, *Projekt i implementacja systemu translującego wybrane diagramy języka SysML na modele sieci Petriego*, Akademia Górniczo-Hutnicza, Kraków [praca magisterska, promotor T. Szmuc].

Karypis G., Kumar V., 1999, *A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs*, SIAM Journal on Scientific Computing, vol. 20, no 1, s. 359–392, 1998. https://doi.org/10.1137/S1064827595287997.

Loose W., 2010, *The State of European Car Sharing, Programme Européen MOMO Car--Sharing*, Final Report.

Łukowicz Ł., 2019, *Modelowanie i analiza oprogramowania z zastosowaniem języka AADL i sieci Petriego*, Akademia Górniczo-Hutnicza [praca magisterska, promotor T. Szmuc].

Miller G.L., Teng S.-H., Vavasis S.A., 1991, *A Unified Geometric Approach to Graph Separators*, [w:] *Proceedings 32nd Annual Symposium of Foundations of Computer Science*, IEEE, s. 538–547, https://doi.org/10.1109/SFCS.1991.185417.

Nadeau D.R., 1999, *Building virtual worlds with VRML*, IEEE Computer Graphics and Applications, vol. 19, no. 2, s. 18–29, https://doi.org/10.1109/38.749119.

National Science Foundation (NSF), 2006, *Cyber-Physical Systems (CPS)*, https://www.nsf.gov/pubs/2010/nsf10515/nsf10515.htm [dostęp: 11.04.2022].

nuXmv model checker, b.d., https://nuxmv.fbk.eu/ [dostęp: 11.04.2022].

OMG SysML, 2019, *Object Management Group*, https://www.omgsysml.org/ [dostęp: 11.04.2022].

Osate, b.d., https://osate.org/download-and-install.html [dostęp: 11.04.2022].

Papyrus, b.d., https://www.eclipse.org/papyrus/download.html [dostęp 11.04.2022].

PetriNet2ModelChecker, b.d., http://alvis.kis.agh.edu.pl/wiki/software [dostęp: 11.04.2022].

Pothen A., Simon H.D., Liou K.-P., 1990, *A Partitioning Sparse Matrices with Eigenvectors of Graphs*, SIAM Journal on Matrix Analytical Applications, vol. 11(3), s. 430–452. https://doi.org/10.1137/0611030.

Shaheen S., Cohen A., 2012, *Carsharing and Personal Vehicle Services: Worldwide Market Developments and Emerging Trends*, International Journal of Sustainable Transportation, vol. 7, no. 1, s. 5–34. https://doi.org/10.1080/15568318.2012.660103.

Shi J., Malik J., 2020, *Normalized cuts and image segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 8, s. 888–905. https://doi.org/10.1109/34.868688.

Stanton I., Kliot G., 2012, *Streaming Graph Partitioning for Large Distributed Graphs*, [w:] *KDD '12: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 2012*, s. 1222–1230. https://doi.org/10.1145/2339530.2339722.

Szmuc T., Szmuc W., 2020a, *Consistency Preserving Development of Embedded Systems Using AADL*, [w:] Napieralski A. (ed.), *MIXDES 2020: Proceedings of 27th International Conference Mixed Design of Integrated Circuits and Systems, Łódź, Poland, June 25–27, 2020*, Lodz University of Technology, Łódź, s. 245–248.

Szmuc T., Szmuc W., 2020b, *Rigorous Development of Embedded Systems Supported by Formal Tools*, [w:] Napieralski A. (ed.), *MIXDES 2020: Proceedings of 27th International Conference Mixed Design of Integrated Circuits and Systems, Łódź, Poland, June 25–27, 2020*, Lodz University of Technology, Łódź, s. 272–276.

Szmuc W., 2014, *Modelowanie wybranych diagramów UML 2.0 z zastosowaniem kolorowanych sieci Petriego*, AGH Akademia Górniczo-Hutnicza [praca doktorska, promotor M. Szpyrka].

Szmuc W., Szmuc T., 2016a, *From UML object behavior description into Petri net models. Towards systematic development of embedded systems*, International Journal of Microelectronics and Computer Science, vol. 7, no 2, s. 60–64.

Szmuc W., Szmuc T., 2016b, *Modeling UML object event handling with Petri Nets. Towards improvement of embedded systems analysis and design*, [w:] Napieralski A. (ed.), *MIXDES 2016: Mixed Design of Integrated Circuits and Systems: Łódź, Poland, June 23–25, 2016: book of abstracts of 23rd International Conference*, Lodz University of Technology, Department of Microelectronics and Computer Science, Łódź, s. 454–457.

Szmuc W., Szmuc T., 2018, *Towards embedded systems formal verification. Translation from SysML into Petri nets*, [w:] Napieralski A. (ed.), *MIXDES 2018: Mixed Design of Integrated Circuits and Systems: book of abstracts of 25th International Conference: Gdynia, Poland, June 21–23, 2018*, Lodz University of Technology, Department of Microelectronics and Computer Science, Łódź, s. 420–423.

Szpyrka M., Biernacki J., Biernacka A., 2016, *Tools and methods for RTCP-nets modeling and verification*, Archives of Control Sciences, vol. 26, no. 3, s. 339–365. https://doi.org/10.1515/acsc-2016-0019.