Metodyka projektowania systemów robotycznych w oparciu o metamodele EARL i MeROS

Tomasz Winiarski, Jan Kaniuka, Jakub Ostrysz
Politechnika Warszawska, Instytut Automatyki i Informatyki Stosowanej

Streszczenie: Inżynieria systemów odgrywa obecnie kluczową rolę w procesie wytwarzania, wdrażania oraz utrzymania systemów cyberfizycznych. Coraz częściej staje się ona nieodłącznym narzędziem podczas projektowania, chociażby systemów robotycznych. W artykule zaproponowano metodykę projektowania systemów robotycznych w oparciu o dwa metamodele: EARL na poziomie niezależnym od platformy implementacji oraz MeROS dedykowany dla ROS/ROS 2. Procedura została zaprezentowana poprzez analizę reprezentatywnego zastosowania aplikacyjnego: heteregonicznego systemu wielorobotowego z centralnym koordynatorem.

Słowa kluczowe: robot, inżynieria systemów, ROS 2, EARL, MeROS

1. Wprowadzenie

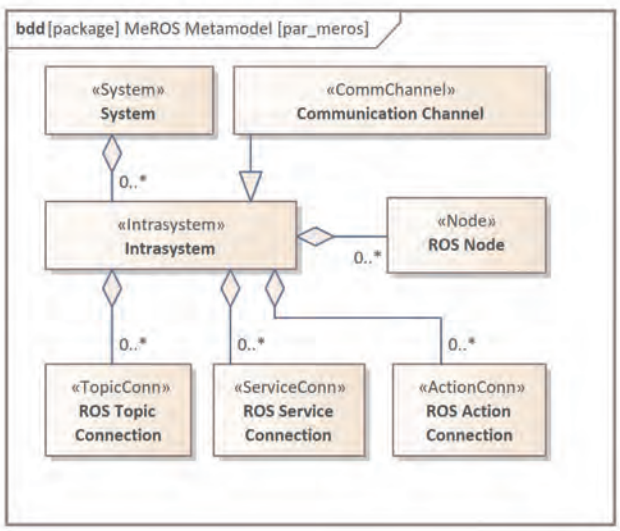
Podobnie jak w innych dziedzinach inżynierii, w robotyce podstawowym problemem jest jak opracowywać systemy w warunkach różnorodnych wyzwań i ryzyk projektowych. Procedura powinna być możliwie niezawodna i minimalizować ryzyko błędów czy też niepowodzenia projektu. Tradycyjną odpowiedzią jest inżynieria systemów oparta na modelu MBSE (ang. *Model-Based Systems Engineering*) [16]. Pytaniem, na jakie odpowiada ten artykuł, jest postać i sposób realizacji procedury projektowej nawiązującej do powyższej problematyki. Procedura opiera się współczesnych narzędziach, a do jej ilustracji wybrano przykład nawiązujący do robotyki przemysłowej.

Charakter systemów przemysłowych powoduje, że w wielu zastosowaniach uzasadnione jest użycie centralnej koordynacji [26–28]. Choć tego typu rozwiązania znane są od wielu lat, obecnie coraz częściej roboty wdrażane w zakładach przemysłowych zgodnie z koncepcją Przemysłu 4.0 są silnie zróżnicowane i obok robotów manipulacyjnych pojawią się roboty mobilne autonomicznie przewożące towary w ramach procesów intralogistycznych.

Przemysłowe systemy robotyczne cechowała znaczna rozmaitość języków programowania [29]. Ponadto frameworki usprawniające proces programowania robotów tworzone były pierwotnie głównie w środowisku akademickim [30]. Wśród frameworków na pierwszy plan wysunął się ROS (ang. *Robot Operating System*), obecnie rozwijany i zalecany do użycia w wersji drugiej [10]. Naturalną koleją rzeczy jest postępująca próba standaryzacji sposobu programowania robotów przemysłowych w oparciu o ROS 2 i społeczność ROS Industrial [4].

Efektywny opis systemów wspierany jest przez graficzny język SysML [11]. Do tego wskazane są metamodele dziedzinowe [1], stąd w ostatnim czasie dla ROS powstał metamodel bazujący na SysML o nazwie MeROS. Artykuł [18] zawiera szczegółowe omówienie sensu i znaczenia stosowania podobnych metamodeli, natomiast bieżąca wersja MeROS wraz z dodatkowymi materiałami instruktażowymi umieszczona jest na stronie tego projektu[1]. Rysunek 1 przedstawia kluczowe dla dalszej prezentacji bloki MeROS. Odnoszą się one do poziomu abstrakcyjnego opisu systemu poprzez Intrasystemy i Kanały Komunikacyjne oraz konkretne aspekty implementacji jak Węzły i metody komunikacji ROS.

[1] https://github.com/twiniars/MeROS



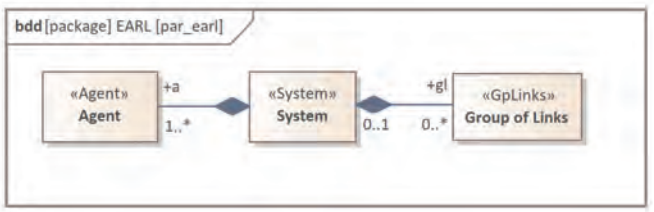
Rys. 1. Kompozycja systemu MeROS: Intrasystem i związane z nim bloki
Fig. 1. MeROS System Composition: Intrasystem and associated blocks

Autor korespondujący:
Tomasz Winiarski, tomasz.winiarski@pw.edu.pl

Artykuł recenzowany
nadesłany 01.11.2023 r., przyjęty do druku 16.04.2024 r.

Zezwala się na korzystanie z artykułu na warunkach licencji Creative Commons Uznanie autorstwa 3.0

Sam proces opracowywania systemu cyberfizycznego, jakim jest system robotyczny, wymaga osadzenia logiki działania w sprzęcie, stąd obok logicznej koncepcji działania potrzebne jest upostaciowienie [9]. Taką możliwość daje EARL [22, 23]. Podobnie jak MeROS, EARL oparty jest na SysML, co ułatwia ich wspólne wykorzystanie, będące przedmiotem tego artykułu. Mimo tego że EARL jest nowym rozwiązaniem, był już szeroko stosowany m.in. dla robotów manipulacyjnych i mobilnych [2]. EARL wywodzi się z Teorii Agentowej Szkoły Warszawskiej [7, 9, 25], podejścia ugruntowanego od lat i stosowanego współcześnie w wyjściowej postaci m.in. w tak rozbudowanych frameworkach jak TaskER [3]. W artykule korzystamy z EARL i MeROS, kontynuując i rozszerzając wcześniejsze prace nad metodami projektowania sterowników systemów robotycznych [17, 31] przez synergiczne stosowanie metamodeli zależnych i niezależnych od platformy implementacyjnej. W tym celu koncentrujemy się na blokach EARL wskazanych na diagramie 2, które z jednej strony odnoszą się do ogólnej struktury systemu, ale co jest kluczowe dla znaczenia EARL w proponowanym podejściu, mają upostaciowiony charakter.



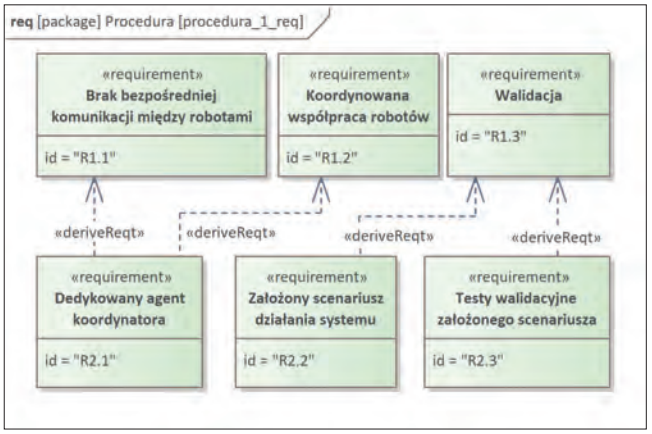
Rys. 2. Kompozycja systemu EARL: Agenty i Grupy Połączeń
Fig. 2. EARL System Composition: Agents and Groups of Links

Postać dalszej części artykułu jest następująca. Rozdział 2 prezentuje procedurę projektową. Rozdział 3 opisuje jej przykładową realizację. Artykuł kończy podsumowanie w rozdziale 4.

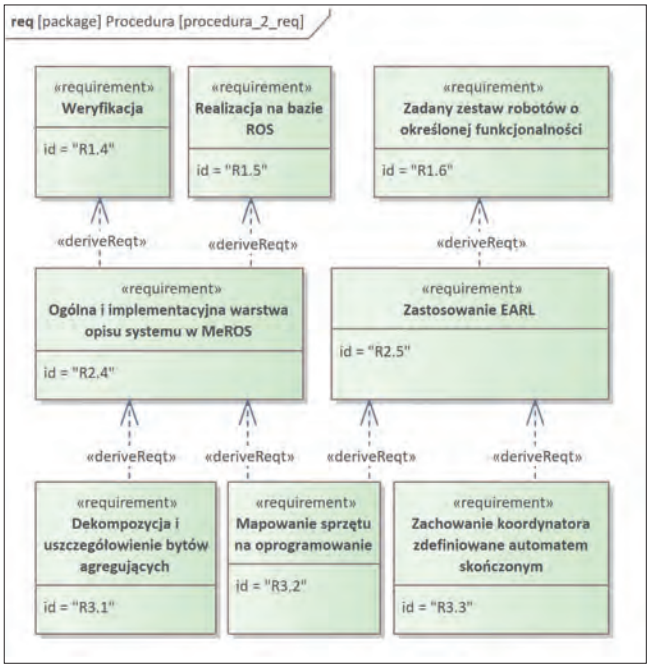
2. Procedura projektowa

Kluczowe wymagania dla procedury projektowej przedstawiono na rys. 3 i 4. Diagramy te, podobnie jak inne części procedury, zostały zdefiniowane w SysML [13]. Wymagania zaetykietowano według wzorca [RX.Y], gdzie X oznacza poziom wymagań, a Y numer wymagania na określonym poziomie. Poziom pierwszy odnosi się do wyjściowych wymagań, natomiast kolejne poziomy są wywiedzione i wiążą się z decyzjami projektowymi. Kluczowa w procesie opracowywania procedury projektowej jest alokacja wymagań, co uwidocznione jest na kolejnych diagramach, tym razem aktywności. Poczynając od rys. 3. Założono, że procedura będzie dotyczyła pewnej podklasy systemów robotycznych, gdzie brak jest bezpośredniej komunikacji między robotami [R1.1], ale i ich działania są koordynowane [R1.2]. Wynika z tego dedykowany Agent koordynatora [R2.1]. Walidacja [R1.3] jest jednym z kluczowych elementów przebiegu projektu w inżynierii systemów [6]. W rozważanej klasie systemów robotycznych założono, że system ma realizować scenariusz określony na wstępie [R2.2] i podlegający walidacji podczas testów systemu [R2.3].

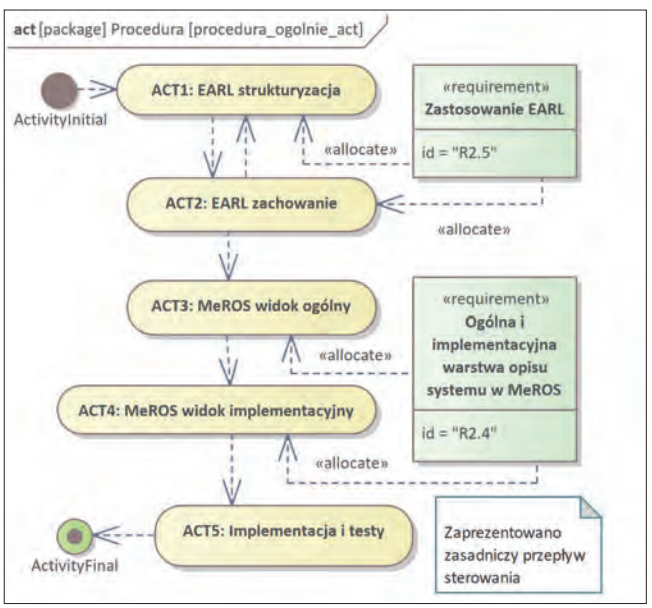
Kolejny zestaw wymagań przedstawiono na rys. 4. Weryfikacja [R1.4] jest newralgicznym elementem procesu projektowego [6] i wspiera jego poprawny przebieg. Zanim do niej przejdziemy, należy wspomnieć o wymaganiu [R1.5], gdyż w tej pracy zajmujemy się systemami na bazie ROS 2. Weryfikacja wiąże się w praktyce z zapewnieniem czytelnych przejść pomiędzy różnymi sposobami/poziomami definicji projektowanego systemu. Stąd zdecydowano o zastosowaniu MeROS w wersji 1.2.7, który to zapewnia, i, co również było wymagane, odnosi się do ROS. Przejście od ogólniejszych do bliższych imple-



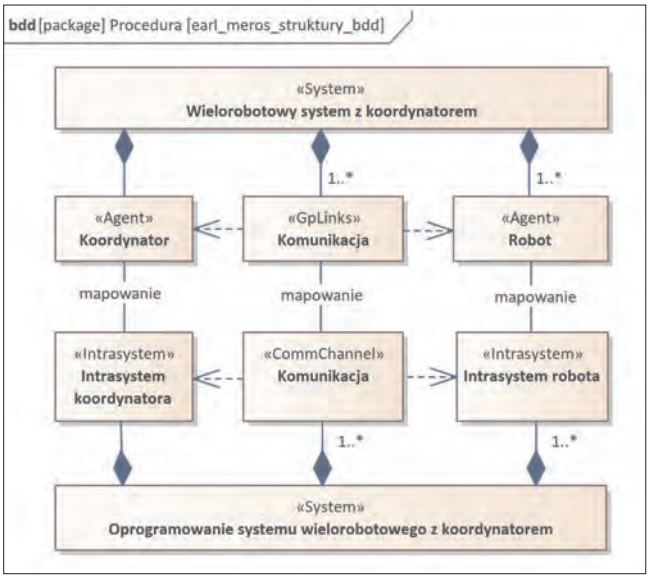
Rys. 3. Wymagania dla procedury projektowej – część 1
Fig. 3. Design procedure requirements – part 1



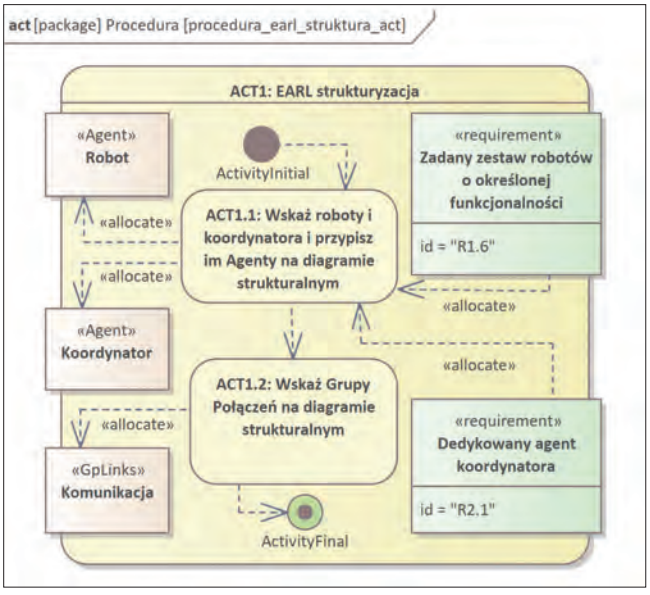
Rys. 4. Wymagania dla procedury projektowej – część 2
Fig. 4. Design procedure requirements – part 2



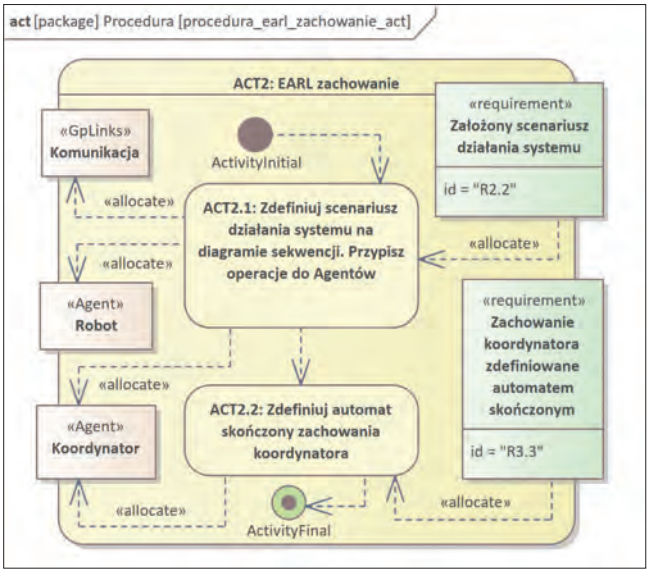
Rys. 5. Procedura projektowa – widok ogólny
Fig. 5. Design procedure – general view



Rys. 6. Struktura systemu wyrażona w EARL i MeROS
Fig. 6. System structure expressed in EARL and MeROS



Rys. 7. Procedura projektowa – ACT1: EARL strukturyzacja
Fig. 7. Design procedure – ACT1: Structurisation with EARL



Rys. 8. Procedura projektowa – EARL zachowanie
Fig. 8. Design procedure – Behaviour with EARL

mentacji opisów systemu w MeROS wymaga dekompozycji i uszczegóławiania bytów agregujących [R3.1].

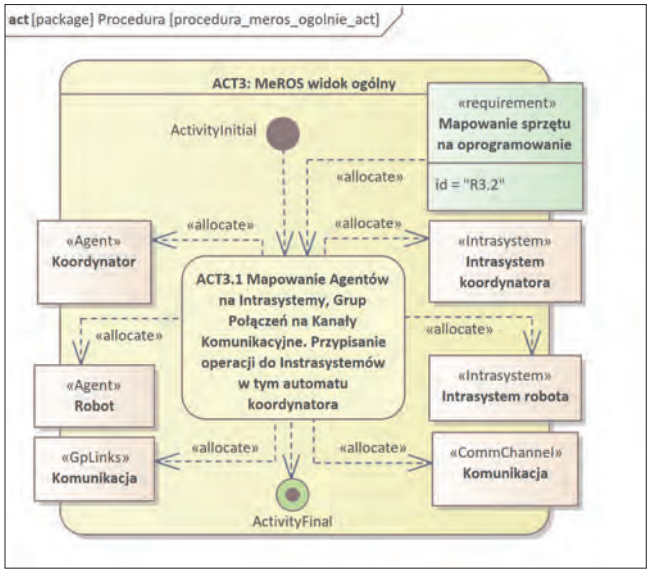
Powracając do wymagań wyjściowych, zakładamy pewien zadany zestaw robotów o określonej funkcjonalności [R1.6], z którego zostanie skomponowany system. Roboty mają w naturalny sposób fizyczną naturę, co skłania do zastosowania EARL[2] [R2.5] (wybrano jego najnowszą wersję 1.3) do modelowania systemu o fizycznej naturze, podczas gdy MeROS dotyczy konkretnej platformy programistycznej. W EARL zachowanie Agentów definiowane jest na ogólnym poziomie poprzez automat skończony i tak też zdefiniowany jest koordynator [R3.3], który w przeciwieństwie do robotów nie ma z góry określonej funkcjonalności. Połączenie EARL z MeROS w jednym procesie projektowym wymaga mapowania sprzętu na oprogramowanie [R3.2].

Procedura projektowa została przedstawiona w ogólnej postaci poprzez diagram aktywności na rys. 5. Wyróżniono pięć głównych aktywności oznaczonych etykietami [ACTX.Y], gdzie X to numer aktywności głównej, a Y to numer podaktywności. Na diagramie 5 nakreślono zasadniczy przepływ sterowania oraz alokację wymagań. Każda z aktywności głównych została szczegółowo omówiona na dedykowanym dla siebie diagramie aktywności w dalszej części tej sekcji. Zastosowanie EARL [R2.5] pociąga za sobą aktywności: [ACT1] – gdzie należy nakreślić strukturę systemu w EARL, a następnie [ACT2] – gdzie definiowane jest zachowanie systemu w EARL. Kolejny etap procedury zachodzi w MeROS [R2.4], w którym najpierw definiowany jest widok ogólny [ACT3] a następnie implementacyjny [ACT4]. Projekt kończą implementacja i testy [ACT5]. Naturalnie w praktyce możliwe, a nawet niekiedy wskazane, są powroty do poprzednich faz procedury, stąd mowa jest o zasadniczym przepływie sterowania. Dodatkowe dwie aktywności: ActivityInitial oraz ActivityFinal są standardowymi aktywnościami, którymi oznacza się początek i koniec procedury.

Dla dalszych, szczegółowych rozważań kluczowe jest sformułowanie struktury opracowywanego systemu poprzez diagram definicji bloków na rys. 6.

Diagram przedstawia obraz systemu z dwóch perspektyw: EARL (u góry) i MeROS w widoku ogólnym (u dołu). Obie części podlegają mapowaniu. Aktywność [ACT1] przedstawiona jest na diagramie 7. W pierwszej kolejności należy zaproponować diagram strukturalny w EARL, składający się z Agentów robotów i koordynatora [ACT1.1] oraz połączeń między nimi [ACT1.2].

[2] https://www.robotyka.ia.pw.edu.pl/projects/earl/

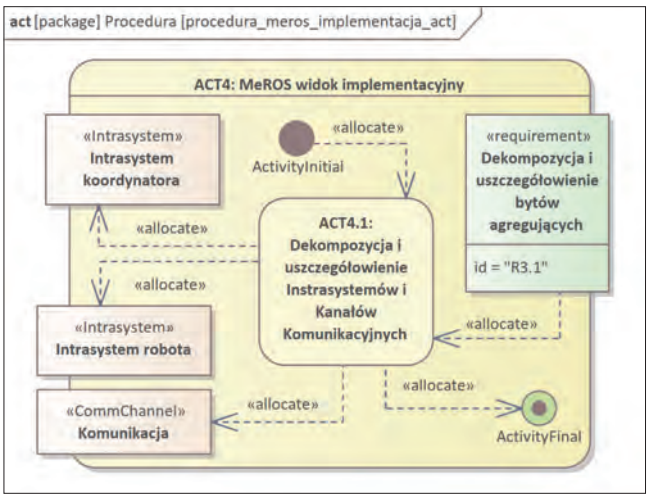


Rys. 9. Procedura projektowa – ACT3: MeROS widok ogólny
Fig. 9. Design procedure – ACT3: MeROS general view

W kolejnym etapie (rys. 8) następuje w pierwszej kolejności sformułowanie scenariusza działania systemu na diagramie sekwencji [ACT2.1], gdzie podmiotami są komponenty z diagramu strukturalnego opracowanego w etapie pierwszym [ACT1]. W oparciu o scenariusz oraz założoną na wstępie funkcjonalność robotów należy opracować automat skończony dla koordynatora [ACT2.2].

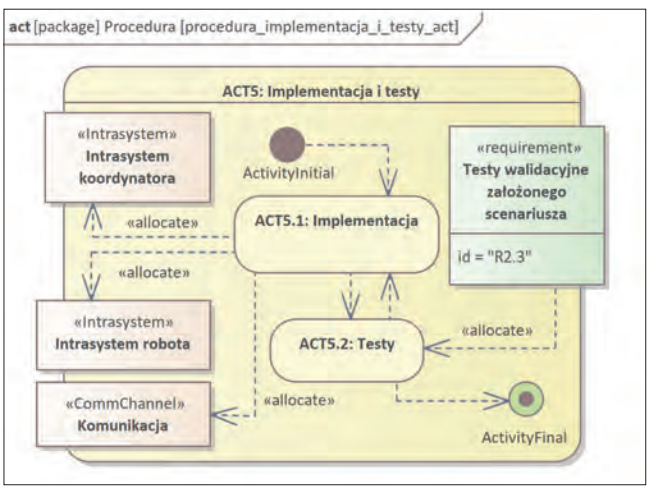
Po definicji systemu w EARL, przychodzi czas na specyfikację w MeROS. W pierwszej kolejności opracowywany jest widok ogólny (rys. 9). Jest to stosunkowo proste [ACT3.1] i sprowadza się do mapowania bloków EARL na ich odpowiedniki w MeROS. W ogólności ROS i MeROS mogą być zastąpione przez inne specyficzne rozwiązania, co jest kolejną zaletą wykorzystania modelowania niezależnego od platformy, jakie zapewnia EARL.

W kolejnej fazie (rys. 10) realizowana jest dekompozycja i uszczegółowienie komponentów [ACT4.1] określonych uprzednio w widoku ogólnym MeROS. Choć to pojedyncza aktywność, to jest to jeden z najbardziej czasochłonnych i złożonych etapów projektowania systemu.



Rys. 10. Procedura projektowa – ACT4: MeROS widok implementacyjny
Fig. 10. Design procedure – ACT4: MeROS implementation view

Na koniec (rys. 11) następują implementacja [ACT5.1] i testy [ACT5.2]. Naturalnie ich przebieg może wpływać na poprzednie aktywności. Wskazana jest też wspomagana generacja kodu w celu ułatwienia zapewnienia jego zgodności ze specyfikacją. W [ACT5.2] domykana jest procedura tworzenia systemu poprzez walidację pierwotnie założonego scenariusza jego działania.



Rys. 11. Procedura projektowa – ACT5: Implementacja i testy
Fig. 11. Design procedure – ACT5: Implementation and tests

3. Realizacja przykładowego systemu

3.1. Założenia

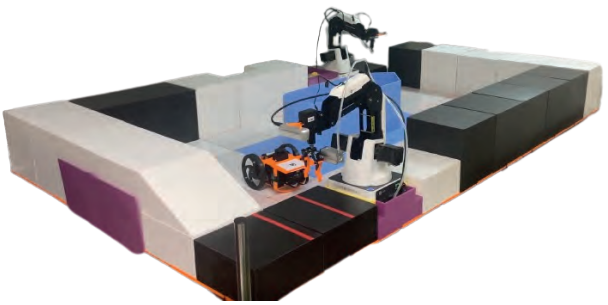
Podstawowym wymaganiem odnośnie realizacji systemu jest [R1.2] z diagramu 3. Współpraca wymaga współdziałania co najmniej dwóch robotów. Postanowiono wykorzystać roboty dostępne w Laboratorium Robotyki Instytutu Automatyki i Informatyki Stosowanej WEiTI PW, tzn. dwa manipulatory Dobot Magician i robota mobilnego MiniRyś. Opisom wspomnianych platform robotycznych poświęcono osobne sekcje w dalszej części tego artykułu. Drugą istotną kwestią było zdefiniowanie zadania, którą mają wspólnie wykonywać roboty. Zdecydowano się na realizację zadania transportu detalu przez robota mobilnego pomiędzy dwoma robotami manipulacyjnymi.

3.2. Opis środowiska testowego

Środowiskiem operacyjnym dla robotów jest modularna plansza o wymiarach 2 m × 1,2 m [20] (rys. 12).

Konfiguracja tej planszy może być modyfikowana przez zmianę położenia poszczególnych elementów oraz dodanie ruchomych przeszkód. Podczas budowy planszy wykorzystano elementy modularnego środowiska do rywalizacji robotów sportowych śledzących linię [24]. Wszystkie kable zasilające, sygnałowe i sieciowe poprowadzono pod planszą. Roboty manipulacyjne Dobot Magician zostały rozmieszczone w dwóch przeciwległych rogach planszy. Natomiast robot mobilny MiniRyś może swobodnie poruszać się po całej planszy. 3.3. Robot manipulacyjny Dobot Magician Robot manipulacyjny wykorzystany podczas testów opracowanego systemu to Dobot Magician – wielofunkcyjny, szeregowy manipulator dydaktyczny o czterech stopniach swobody (rys. 13) produkowany przez firmę Dobot.

Zasięg manipulatora wynosi 320 mm, a maksymalny udźwig to 0,5 kg. Robota Dobot Magician można łatwo rekonfigurować i dodawać nowy osprzęt np. przyssawkę pneumatyczną lub dwupalczasty chwytak równoległy, dostosowując go pod konkretne



Rys. 12. Środowisko testowe
Fig. 12. Test environment

Rys. 13. Manipulator Dobot Magician – wizualizacja [20]
Fig. 13. Dobot Magician manipulator – visualization [20]

zastosowanie aplikacyjne. Manipulator jest sterowany pozycyjnie. Przy komunikacji z manipulatorem korzysta się z portu szeregowego i interfejsu USB.

Robot został wyposażony w system sterowania opracowany w ramach pracy inżynierskiej "*System sterowania robota manipulacyjnego Dobot Magician na bazie frameworka ROS 2*" [8]. System sterowania daje możliwość interakcji z robotem z wykorzystaniem mechanizmu akcji (ruch manipulatora), usług (sterowanie efektorami) i tematów (odczyt pozycji). Użytkownik ma również możliwość wizualizacji stanu robota, sterowania z poziomu GUI oraz diagnostyki. Struktura kinematyczna tego robota sprawia, że przystaje on do realizacji zadań typu podnieś i umieść. Robot wykorzystany podczas testów posiada dodatkowe komponenty wykonane w technologii druku 3D – nakładki na chwytak oraz uchwyt do kamery głębi Intel RealSense D435i.

3.4. Robot mobilny MiniRyś

MiniRyś to dwukołowy mobilny robot o zmiennym sposobie lokomocji, który rozwijany jest w ramach działalności Koła Naukowego Robotyki "Bionik" od 2013 r. (rys. 14).

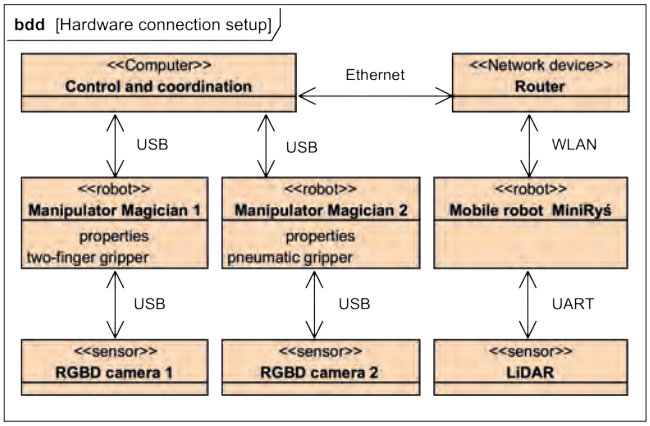
Rys. 14. Robot mobilny MiniRyś (model CAD)
Fig. 14. MiniRyś mobile robot (CAD model)

Pierwsze prototypy [19] powstały na podstawie badań nad platformą robota o zmiennym trybie lokomocji [14, 15, 21]. W kolejnych wersjach rozwojowych robota badano jego poszczególne komponenty sprzętowe, rozwiązania programowe jak również systemy operacyjne. Aktualnie robot posiada system sterowania oparty o ROS 2 stworzony w ramach prac nad platformą sprzętowo-programową czasu rzeczywistego [5]. W ramach ostatnich prac nad najnowszą wersją robota skupiono się na diagnostyce, testach i wizualizacji działania [12]. Platforma sprzętowa robota została wyposażona w obrotowy skaner LiDAR, a system sterowania został zintegrowany ze stosem nawigacyjnym Nav2. Oprócz poruszania się w trybie poziomym, w którym trzecim punktem podparcia jest zderzak, ma on również możliwość przemieszczania się w trybie pionowym, w którym robot balansuje na dwóch kołach. Robot jest platformą dydaktyczno-badawczą, umożliwiającą szkolenie z zakresu sterowania robotami mobilnymi, systemów wizyjnych, systemów czasu rzeczywistego, a także prowadzenie badań z zakresu zarządzania systemami wielorobotowymi i zachowania roju robotów.

3.5. Konfiguracja sprzętowo-sieciowa systemu podczas testów

Elementy systemu (m.in. roboty i ich sensory) wraz z interfejsami do ich wzajemnej komunikacji zobrazowano na diagramie 15.

Celem zachowania zgodności językowej modelu systemu z wytworzonym oprogramowaniem, na diagramach prezentujących realizację systemu zastosowano anglojęzyczne oznaczenia. Manipulatory komunikują się z kamerami RGBD oraz z jednostką sterująco-koordynującą z wykorzystaniem interfejsu USB. Komunikacja z robotem mobilnym odbywa się poprzez interfejs WLAN, a jego eksteroreceptor tzn. LiDAR, komunikuje się z komputerem robota z wykorzystaniem interfejsu UART.



Rys. 15. Diagram konfiguracji sprzętowo-sieciowej
Fig. 15. Hardware and network configuration diagram

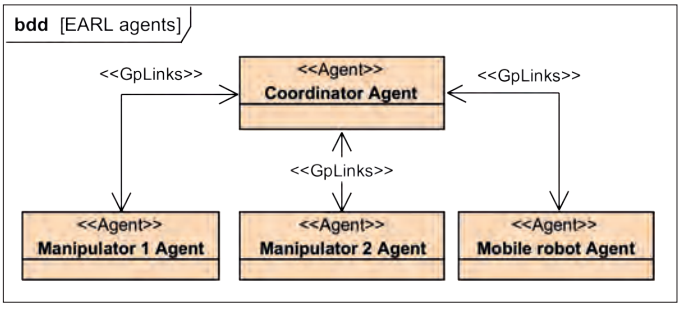
3.6. Przykład zastosowania procedury projektowej

Pierwszym krokiem procedury projektowej jest strukturyzacja z wykorzystaniem dziedzinowego języka opisu systemów cyberfizycznych, którym jest EARL [ACT1]. Należy wskazać roboty i koordynatora oraz przypisać im Agenty [ACT1.1]. W naszym systemie występują cztery Agenty, których typy oraz zadania podano poniżej.

1) Agent koordynatora (CT): koordynacja.
2) Agent manipulatora pierwszego (CERT): stawianie kostki na robocie mobilnym.
3) Agent manipulatora drugiego (CERT): podnoszenie kostki z robota mobilnego.
4) Agent robota mobilnego (CERT): transport kostki między manipulatorami.

Po wskazaniu Agentów można przejść do wskazania Grup Połączeń <GpLinks> między nimi [ACT1.2], pamiętając przy tym o założeniu [R1.1] mówiącym o braku bezpośredniej komunikacji między robotami. Na diagramie 16 zobrazowano grupy połączeń między Agentami. Agent koordynatora wymienia informacje z Agentami każdego z robotów, pośrednicząc tym samym w komunikacji między poszczególnymi robotami.

W drugim etapie opracowanej procedury projektowej również wykorzystywany jest język EARL. Ten etap rozpoczęto od sformułowania scenariusza działania systemu w formie diagramu sekwencji [ACT2.1], który widoczny jest na rysunku 17. Bloki występujące na tymże diagramie to Agenty z diagramu strukturalnego z rysunku 16.

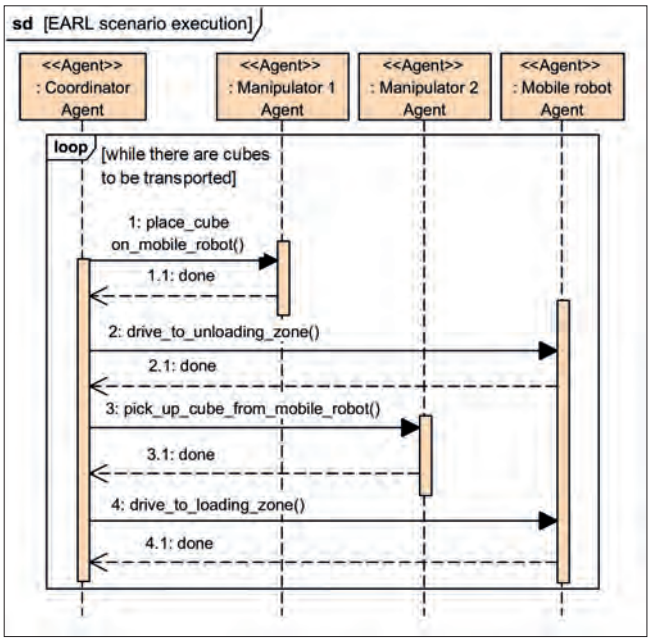


Rys. 16. Ogólna struktura omawianego systemu wielorobotowego
Fig. 16. Overall structure of considered multi-robot system

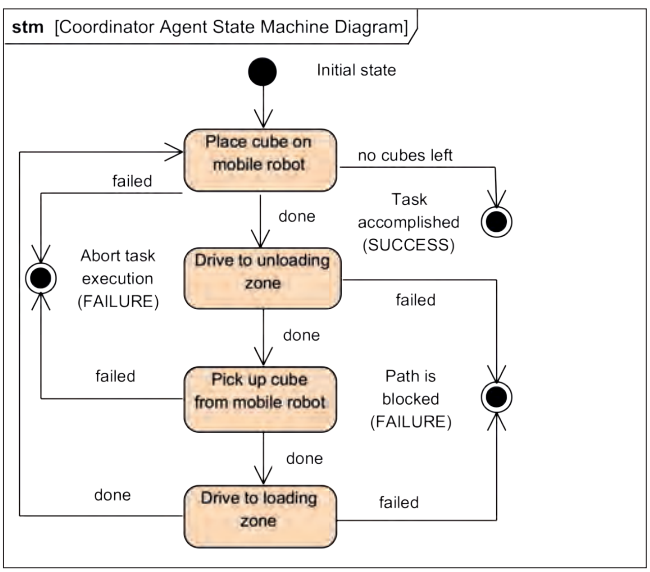
Diagram sekwencji reprezentuje jedynie konkretny scenariusz realizacji danego zadania. Jest on przejrzysty i czytelny dlatego, że nie są na nim rozpatrywane wszystkie sytuacje, które potencjalnie mogą zachodzić w trakcie działania systemu. Do zilustrowania wszystkich stanów systemu i tranzycji pomiędzy nimi wykorzystywany jest diagram automatu stanów (ang. *State Machine Diagram*), którego utworzenie jest częścią drugiego etapu procedury projektowej [ACT2.2]. Automat skończony zdefiniowany dla naszego scenariusza testowego widoczny jest na rysunku 18.

Na diagramie automatu skończonego znajdują się także stany terminalne, które są osiągane w momencie niepowodzenia na etapie wykonywania danego podzadania przez robota. W tym miejscu warto także wspomnieć o założeniu dotyczącym stanu początkowego systemu. Zakładamy, że robot mobilny nie ma na sobie kostki i znajduje się on w zasięgu manipulatora, który ma nim tę kostkę położyć.

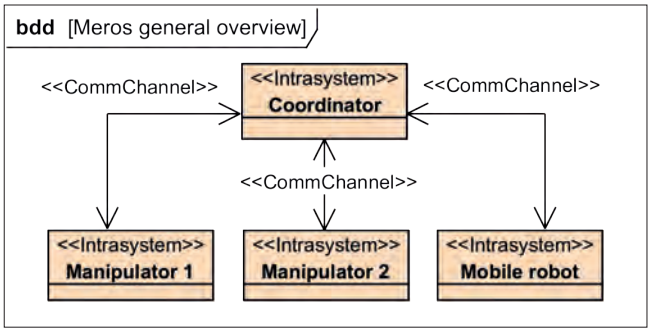
Kolejny krok procedury to specyfikacja systemu z wykorzystaniem metamodelu MeROS [ACT3.1]. Rezultatem realizacji tego etapu procedury projektowej jest diagram strukturalny



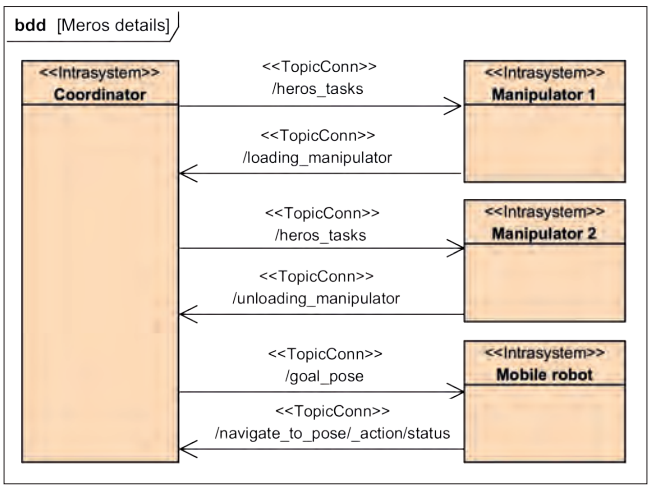
Rys. 17. Scenariusz działania systemu
Fig. 17. System operation scenario



Rys. 18. Automat skończony zachowania koordynatora
Fig. 18. Finite state automata of coordinator behaviour



Rys. 19. Specyfikacja systemu z wykorzystaniem metamodelu MeROS
Fig. 19. System specification using MeROS metamodel



Rys. 20. Interfejsy do komunikacji pomiędzy Intrasystemami
Fig. 20. Communication interfaces between Intrasystems

o ogólnym charakterze widoczny na rysunku 19. Powstał on przez mapowanie bloków EARL na ich odpowiedniki w MeROS.

Najbardziej rozbudowanym krokiem procedury projektowej jest etap czwarty, w którym następuje dekompozycja i uszczegółowienie komponentów [ACT4.1] określonych na diagramie MeROS z widokiem ogólnym. Na diagramie bloków wewnętrznych (rys. 20) opisano, jakie mechanizmy komunikacji dostarczane przez ROS 2 zostały wykorzystane do wymiany informacji między Intrasystemami.

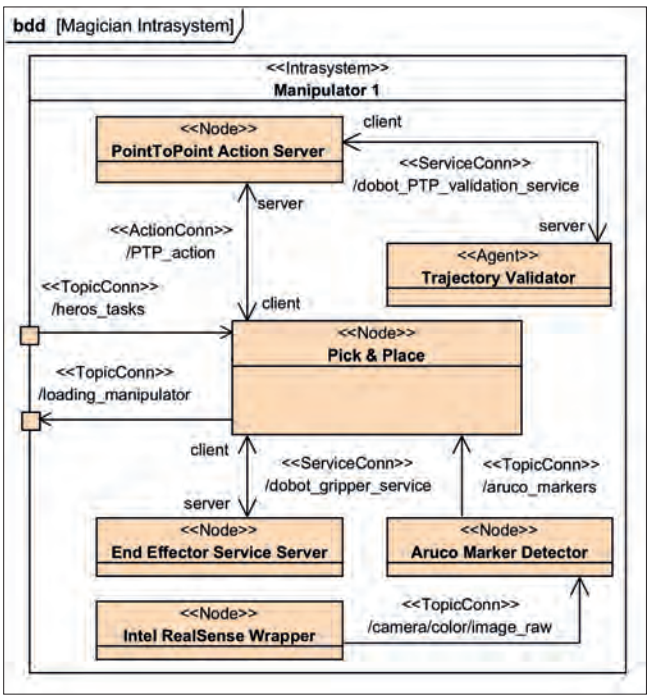
Intrasystemy obydwu manipulatorów otrzymują zadania od koordynatora poprzez odebranie wiadomości typu `std_msgs.String` na temacie `/heros_task` (nazwa *heros* to skrót od "***He****terogenous robots in* ***ROS***"). Manipulatory otrzymują komendy wysokopoziomowe (*load* lub *unload*), które odpowiadają sekwencji akcji wykonywanych w ramach Intrasystemu danego manipulatora. Informacja zwrotna mówiąca o zakończeniu wykonywania zadania lub wystąpieniu błędu jest przesyłana do koordynatora odpowiednio na tematach `/loading_manipulator` i `/unloading_manipulator`.

Robot mobilny otrzymuje od koordynatora zadania przez przesłanie na temacie `/goal_pose` współrzędnych punktu, do którego ma dojechać. Koordynator oczekuje na odpowiedź zwrotną na temacie `/navigate_to_pose/_action/status`. Zadanie dla robota mobilnego MiniRyś może zakończyć się sukcesem, tzn. robot dojechał do celu lub niepowodzeniem – na drodze robota pojawiła się przeszkoda, której nie może ominąć.

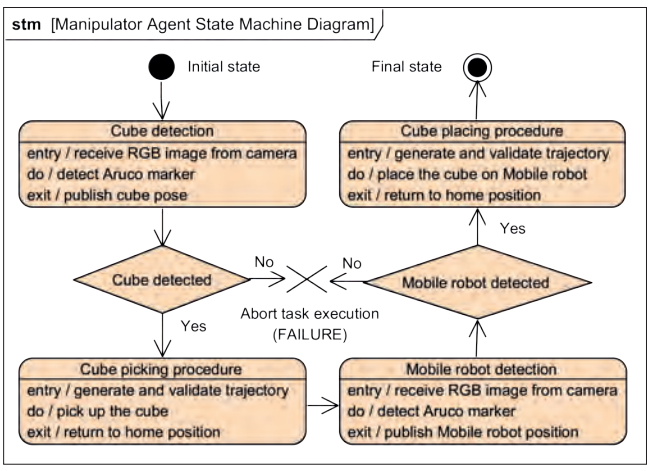
Intrasystem koordynatora został już opisany w formie automatu skończonego na diagramie 18. Wykorzystywane podczas testów manipulatory różnią się jedynie narzędziem roboczym oraz zadaniem, które wykonują. Manipulator 1 ma chwytak dwupalczasty równoległy i odpowiada za odkładanie kostki na robota mobilnego, z kolei Manipulator 2 ma przyssawkę pneumatyczną i zdejmuje nią kostki z robota. Struktura Intrasystemu Manipulator 1 została przedstawiona na diagramie 21.

Węzeł `Pick&Place` odbiera polecenia od Węzła Koordynatora, wykonuje sekwencję operacji pozwalających podnieść kostkę, a następnie przesyła informację zwrotną o statusie wykonania zadania. Węzeł `Intel RealSense Wrapper` przesyła obraz z kamery RGBD do Węzła `Aruco Marker Detector`, który lokalizuje kostkę i przesyła informację o jej pozycji do Węzła `Pick&Place`. Na podstawie tej informacji wykonywana jest sekwencja operacji otwarcia/zamknięcia chwytaka oraz ruchu manipulatora, które pozwalają na podjęcie obiektu. Opisane powyżej operacje przedstawiono również w formie diagramu behawioralnego widocznego na rysunku 22.

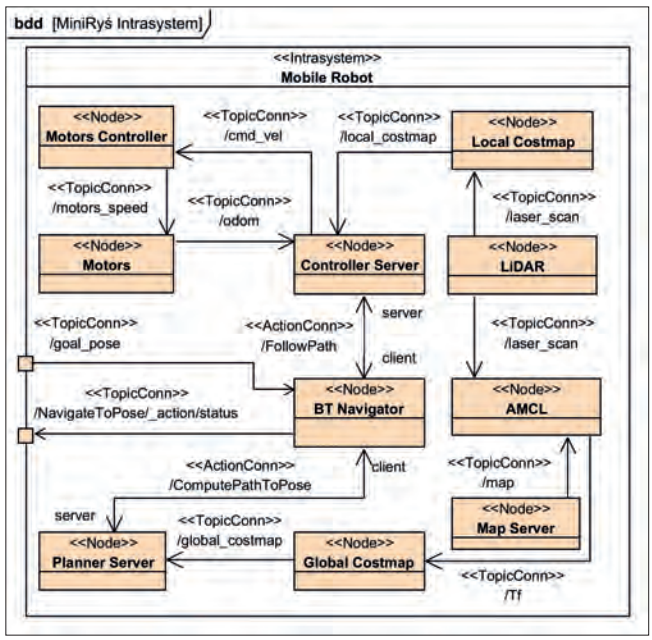
Na diagramie 23 przedstawiono strukturę Intrasystemu robota mobilnego. Głównym elementem jest Węzeł `BT Navigator`, pełniący rolę interfejsu między Węzłem koordynatora a robotem mobilnym. Węzeł ten otrzymuje od koordynatora informację dotyczące punktu na mapie modularnej planszy, do którego dotrzeć ma robot mobilny. Po otrzymaniu wiadomości, `BT Navigator` przekazuje informację zwrotną o statusie wykonania zadania. Węzeł `AMCL` odbiera odczyt z dwuwymiarowego skanu laserowego od Węzła `Lidar`. Dodatkowo otrzymuje mapę środowiska testowego od Węzła `Map Server`. Dzięki tym danym możliwe jest zlokalizowanie robota przez określenie układu współrzędnych bazy robota w układzie współrzędnych mapy. Następnie Węzeł `Global Costmap`, wykorzystuje te dane podczas tworzenia globalnej mapy kosztu. Ta mapa jest następnie przekazywana do Węzła `Planner Server`, na podstawie której wyznaczany jest globalny plan i ścieżka, którą robot powinien podążać. Następnie na podstawie akcji ścieżka ta przetwarzana jest na pozycje na mapie i przekazywana do Węzła `BT Navigator`. Informacja o skanie laserowym 2D wykorzystywana jest również do tworzenia lokalnej mapy kosztu w Węźle `Local Costmap`, która następnie przesyłana jest do Węzła `Controller Server`. Węzeł ten wykorzystuje również dane odometryczne z Węzła silników robota, oraz informację o ścieżce przesyłane za pomocą akcji z Węzła `BT Navigator` w celu wyznaczenia prędkości, z którą powinien poruszać się robot mobilny, aby pokonać skorygowaną ścieżkę uwzględniającą informację o potencjalnych przeszkodach wykrywanych przez lokalną mapę kosztu. Ostatecznie wiadomość dotycząca zadanej prędkości przetwarzana jest przez Węzeł `Motors Controller`, który wyznacza sterowania na poszczególne silniki robota i przekazuje je do Węzła efektorów. Opisane powyżej operacje przedstawiono również w formie diagramu behawioralnego widocznego na rysunku 24.



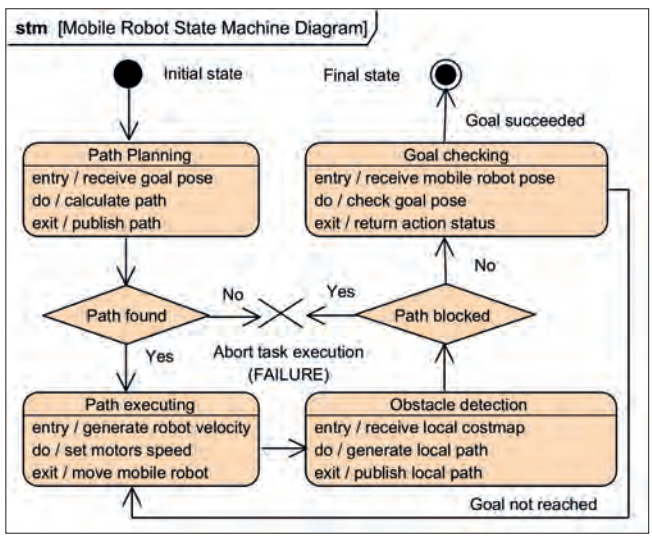
Rys. 21. Struktura Intrasystemu manipulatora
Fig. 21. Diagram of manipulator's Intrasystem



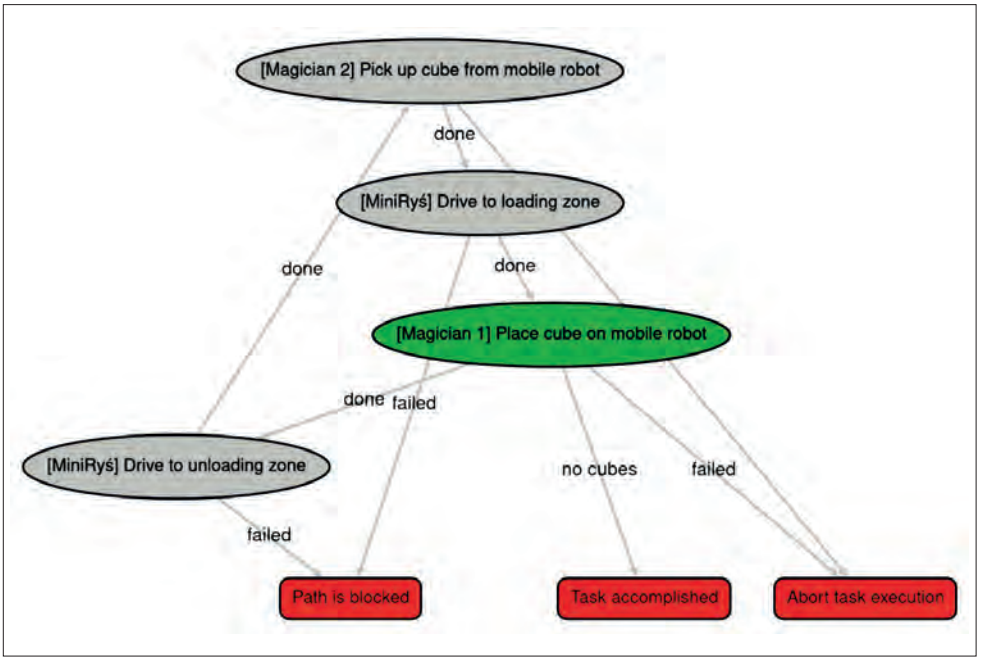
Rys. 22. Diagram behawioralny Intrasystemu manipulatora
Fig. 22. Behavioural diagram of manipulator Intrasystem



Rys. 23. Struktura Intrasystemu robota mobilnego
Fig. 23. Diagram of mobile robot Intrasystem



Rys. 24. Diagram behawioralny Intrasystemu robota mobilnego
Fig. 24. Behavioural diagram of mobile robot Intrasystem



Rys. 25. Widok automatu skończonego koordynatora z poziomu YasminViewer
Fig. 25. View of coordinator's finite state automaton from YasminViewer

4. Podsumowanie

4.1. Ocena działania systemu

Ostatni etap procedury projektowej to implementacja [ACT5.1] i testy [ACT5.2]. Zaplanowany scenariusz testowy wykonano wielokrotnie. We wszystkich przypadkach koordynator w odpowiednich chwilach czasu przydzielał zadania właściwym robotom. Celowo wykorzystano roboty heterogeniczne podczas weryfikacji, aby pokazać, że opracowana procedura projektowa nie zakłada homogeniczności systemu robotycznego. Podczas jednej z prób manipulatorowi z przyssawką pneumatyczną nie udało się pobrać kostki z robota mobilnego. Test ten przeprowadzany był już późną porą i na skutek niedostatecznego oświetlenia sceny system wizyjny miał problem z detekcją obiektu do podjęcia. Wymóg zapewnienia dostatecznego oświetlenia nie został uwzględniony na etapie specyfikacji wymagań.

Film dokumentujący jeden z testów wraz z jednoczesną wizualizacją automatu stanu koordynatora dostępny jest pod linkiem[3].

Podczas implementacji automatu stanów koordynatora wykorzystano otwartoźródłową bibliotekę YASMIN (ang. *Yet Another State MachINe*). Dostarcza ona klasy oraz metody, które pozwalają w prosty i przejrzysty sposób skojarzyć zachowania (akcje) z poszczególnymi stanami. Dodatkowym atutem tej biblioteki jest możliwość bieżącej wizualizacji stanu automatu (rys. 25) przez program *YasminViewer*, co ułatwia identyfikację ewentualnych błędów poczynionych na etapie implementacji.

4.2. Perspektywy rozwoju

Opracowana procedura projektowa dla systemów opartych o ROS 2 charakteryzuje się dużym potencjałem rozwojowym. Przykładowo możliwe jest jej rozbudowa przez wykorzystanie innych diagramów behawioralnych, np. diagramów przypadków użycia (ang. *Use case diagram* [*uc*]).

Etap implementacji systemu opracowanego z wykorzystaniem zaproponowanej procedury projektowej można także usprawnić poprzez dodanie wspomaganej generacji kodu w [ACT5.1]. Przyspieszy to proces tworzenia oprogramowania oraz pozwoli na wyeliminowanie części błędów na początkowym etapie implementacji. Do testów walidacyjnych systemu można także przygotować bardziej skomplikowany scenariusz. Warte rozważenia wydaje się rozbudowanie modularnej planszy, aby umożliwić współdziałanie kilku robotów mobilnych. Istotnym rozszerzeniem etapu projektowania systemu może być także uwzględnienie współbieżności wykonywania zadań przez roboty.

3 https://vimeo.com/865928183

Podziękowania

Projekt został zrealizowany w ramach grantu Koła Naukowego Robotyki Bionik "Rozwój metod współdziałania robotów manipulacyjnych i mobilnych wykorzystujących framework ROS 2". Sprzęt potrzebny do realizacji projektu współfinansowano ze środków programu IDUB – Inicjatywa Doskonałości: Uczelnia Badawcza.

Autorzy pragną podziękować Danielowi Giełdowskiemu i Jakubowi Sadowskiemu za udział w tworzeniu modularnego środowiska testowego dla heterogeniczego systemu robotycznego.

Bibliografia

1. de Araújo Silva E., Valentin E., Carvalho J.R.H., da Silva Barreto R., *A survey of model driven engineering in robotics*. "Journal of Computer Languages", Vol. 62, 2021, DOI: 10.1016/j.cola.2020.101021.
2. Dudek W., Miguel N., Winiarski T., *SPSysML: A metamodel for quantitative evaluation of Simulation-Physical Systems*. arXiv preprint arXiv:2303.09565, 2023, DOI: 10.48550/arXiv.2303.09565.
3. Dudek W., Winiarski T., *Scheduling of a Robot's Tasks With the TaskER Framework*. "IEEE Access", Vol. 8, 2020, 161449–161471, DOI: 10.1109/ACCESS.2020.3020265.
4. D'Avella S., Avizzano C.A., Tripicchio P., *ROS-Industrial based robotic cell for industry 4.0: Eye-in-hand stereo camera and visual servoing for flexible, fast, and accurate picking and hooking in the production line*. "Robotics and Computer-Integrated Manufacturing", Vol. 80, 2023, DOI: 10.1016/j.rcim.2022.102453.
5. Giełdowski D., *Struktura i implementacja systemu robotycznego zawierającego robota MiniRyś*. Master's thesis, WEiTI, 2021.
6. Grady J.O., *System validation and verification*, CRC Press, 1997.
7. Janiak M., Zieliński C., *Control system architecture for the investigation of motion control algorithms on an example of the mobile platform Rex*. "Bulletin of the Polish Academy

of Sciences – Technical Sciences", Vol. 63, No. 3, 2015, 667–678, DOI: 10.1515/bpasts-2015-0078.
8. Kaniuka J., *System sterowania robota manipulacyjnego Dobot Magician na bazie frameworka ROS 2*. Bachelor's thesis, WEiTI, 2023.
9. Kornuta T., Zieliński C., Winiarski T., *A universal architectural pattern and specification method for robot control system design*. "Bulletin of the Polish Academy of Sciences – Technical Sciences", Vol. 68, No. 1, 2020, 3–29, DOI: 10.24425/bpasts.2020.131827.
10. Macenski S., Foote T., Gerkey B., Lalancette C., Woodall W., *Robot operating system 2: Design, architecture, and uses in the wild*, "Science Robotics", Vol. 7, No. 66, 2022, DOI: 10.1126/scirobotics.abm6074.
11. Open Management Group. *OMG Systems Modeling Language* – Version 1.7, December 2022. https://www.omg.org/spec/SysML/1.7/Beta1/PDF (dostęp: 2024-03-23).
12. Ostrysz J., *Badania, modernizacja oraz wizualizacja robota mobilnego MiniRyś*. Bachelor's thesis, WEiTI, 2023.
13. Salado A., Wach P., *Constructing True Model-Based Requirements in SysML*. "Systems", Vol. 7, No. 2, 2019, DOI: 10.3390/systems7020019.
14. Seredyński D., Winiarski T., *Robot mobilny o zmiennym sposobie lokomocji – wyniki badań*. „Pomiary Automatyka Robotyka", R. 17, Nr 7–82013, 107–115.
15. Seredyński D., Winiarski T., Banachowicz K., Walęcki M., Stefańczyk M., Majcher P., *Robot mobilny o zmiennym sposobie lokomocji – konstrukcja mechaniczna i elektroniczna*. „Pomiary Automatyka Robotyka", R. 17, Nr 1, 2013, 162–167.
16. The International Council on Systems Engineering. *INCOSE systems engineering handbook: a guide for system life cycle processes and activities*. John Wiley & Sons, 2023.
17. Trojanek P., Zieliński C., Kornuta T., Winiarski T., *Metoda projektowania układów sterowania autonomicznych robotów mobilnych. Część 2. Przykład zastosowania*. „Pomiary Automatyka Robotyka", R. 15, Nr 10, 2011, 84–90.
18. Winiarski T., *MeROS: SysML-Based Metamodel for ROS-Based Systems*. "IEEE Access", Vol. 11, 2023, 82802–82815, DOI: 10.1109/access.2023.3301727.
19. Winiarski T., Bogusz M., Giełdowski D., Foryszewski K., *Miniaturowy robot mobilny o zmiennym sposobie lokomocji MiniRyś*. XV Krajowa Konferencja Robotyki – Postępy robotyki, Vol. 1, 2018, 251–260.
20. Winiarski T., Giełdowski D., Kaniuka J., Ostrysz J., Sadowski J., *HeROS: a miniaturised platform for research and development on Heterogeneous RObotic Systems*. arXiv:2403.04384, 2024, DOI: 10.48550/arXiv.2403.04384.
21. Winiarski T., Seredyński D., *Robot mobilny o zmiennym sposobie lokomocji – system sterowania*. „Pomiary Automatyka Robotyka", R. 17, Nr 5, 2013, 93–99, 2013.
22. Winiarski T., Seredyński D., *EARL – dziedzinowy język opisu systemów cyberfizycznych*. XVI Krajowa Konferencja Robotyki – Postępy robotyki, Vol. 1, 2022, 223–232.
23. Winiarski T., Węgierek M., Seredyński D., Dudek W., Banachowicz K., Zieliński C., *EARL – Embodied Agent-Based Robot Control Systems Modelling Language*. „Electronics", Vol. 9, No. 2, 2020, DOI: 10.3390/electronics9020379.
24. Węgierek M., Świstak B., Winiarski T., *Modularne środowisko do rywalizacji robotów sportowych śledzących linię*. „Pomiary Automatyka Robotyka", R. 19, Nr 3, 2015, 61–66, DOI: 10.14313/PAR_217/61.
25. Zieliński C., *Transition-function based approach to structuring robot control software*. K. Kozłowski, redaktor, Robot Motion and Control, Vol. 335 serii Lecture Notes in Control and Information Sciences, 2006, 265–286. Springer-Verlag, DOI: 10.1007/978-1-84628-405-2_17.
26. Zieliński C., *Robotyka: techniki, funkcje, rola społeczna Cz. 1. Techniczne podstawy inteligencji i bezpieczeństwa robotów*. „Pomiary Automatyka Robotyka", R. 26, Nr 4, 2022, 5–26, DOI: : 10.14313/PAR_246/5.
27. Zieliński C., *Robotyka: techniki, funkcje, rola społeczna Cz. 2. Aktualne możliwości robotów*. „Pomiary Automatyka Robotyka", R. 27, Nr 1, 2023, 5–18, 1 DOI: 10.14313/PAR_247/5.
28. Zieliński C., *Robotyka: techniki, funkcje, rola społeczna. Cz. 3. Roboty a problemy społeczne*. „Pomiary Automatyka Robotyka", R. 27, Nr 2, 2023, 5–20, 10.14313/PAR_248/5.
29. Zieliński C., Kornuta T., *Programowe struktury ramowe do tworzenia sterowników robotów*. „Pomiary Automatyka Robotyka", R. 19, Nr 1, 2015, 5–14, DOI: 10.14313/PAR_215/5.
30. Zieliński C., Kornuta T., Stefańczyk M., Szynkiewicz W., Trojanek P., Walęcki M.. *Języki programowania robotów przemysłowych*. „Pomiary Automatyka Robotyka", R. 16, Nr 112012, 10–19.
31. Zieliński C., Kornuta T., Trojanek P., Winiarski T., *Metoda projektowania układów sterowania autonomicznych robotów mobilnych. Część 1. Wprowadzenie teoretyczne*. „Pomiary Automatyka Robotyka", R. 15, Nr 9, 2011, 84–87.

Robotic Systems Development Method Based on EARL and MeROS Metamodels

Abstract: Systems engineering is currently playing a key role in the manufacture, implementation and maintenance of cyber-physical systems. Increasingly, it is becoming an integral tool when designing, for example, robotic systems. This paper proposes a methodology for the design of robotic systems based on two metamodels: EARL at the implementation platform-independent level and MeROS dedicated to ROS/ROS 2. The procedure is demonstrated by analysing a representative application: a heteregonous multi-robot system with a central coordinator.

Keywords: robot, systems engineering, ROS 2, EARL, MeROS

dr inż. Tomasz Winiarski
tomasz.winiarski@pw.edu.pl
ORCID: 0000-0002-9316-3284

Jest adiunktem w Instytucie Automatyki i Informatyki Stosowanej Politechniki Warszawskiej oraz opiekunem Koła Naukowego Robotyki „Bionik". Jego zainteresowania naukowe koncentrują się obecnie wokół inżynierii systemów cyberfizycznych, którą aplikuje w obszarze robotyki. W swojej pracy kierował licznymi grantami badawczymi poświęconymi powyższej tematyce. Jest członkiem polskich sekcji stowarzyszeń IEEE i INCOSE.

inż. Jan Kaniuka
jan.kaniuka.stud@pw.edu.pl
ORCID: 0009-0009-9379-4906

Student kierunku Automatyka i Robotyka na Wydziale Elektroniki i Technik Informacyjnych Politechniki Warszawskiej. W 2023 r. uzyskał tytuł inżyniera z wyróżnieniem. W ramach pracy inżynierskiej zajmował się implementacją systemu sterowania 4-osiowego robota manipulacyjnego na bazie ROS 2. Aktywny członek Koła Naukowego Robotyki "Bionik".

inż. Jakub Ostrysz
jakub.ostrysz.stud@pw.edu.pl
ORCID: 0009-0006-8178-0134

Student kierunku Automatyka i Robotyka na Wydziale Elektroniki i Technik Informacyjnych Politechniki Warszawskiej. W 2023 r. uzyskał tytuł inżyniera. Jego zainteresowania badawcze koncentrują się wokół systemów nawigacji robotów mobilnych i systemów sterowania rojem robotów. Aktywny członek Koła Naukowego Robotyki "Bionik".