Metoda projektowania układów sterowania autonomicznych robotów mobilnych Część 1. Wprowadzenie teoretyczne

Cezary Zieliński*, **Tomasz Kornuta****, **Piotr Trojanek****, **Tomasz Winiarski****

*Przemysłowy Instytut Automatyki i Pomiarów,
**Instytut Automatyki i Informatyki Stosowanej Politechniki Warszawskiej

Streszczenie: Ten dwuczęściowy artykuł przedstawia metodę projektową umożliwiającą określenie zarówno struktury jak i sposobu działania układów sterowania autonomicznych robotów. Część pierwsza koncentruje się na aparacie formalnym i ogólnych przesłankach metody projektowej, natomiast część druga przedstawia przykład obrazujący sposób stosowania metody opisanej w części pierwszej artykułu.

Słowa kluczowe: układ sterowania robota, roboty autonomiczne

Obecne układy sterowania robotów w istotnej części składają się z oprogramowania. Stąd też opracowując metodę ich projektowania należy odnieść się do dokonań inżynierii oprogramowania, na gruncie której opracowano szereg metod postępowania w tym zakresie. Monografia [5] wyróżnia pięć części składowych procesu tworzenia oprogramowania:

- analizę wymagań użytkownika, prowadzącą do stworzenia modelu analitycznego opisującego pożądane działanie systemu,
- projektowanie systemu, czyli opracowanie modelu projektowego określającego schemat budowy oprogramowania, a więc definicje struktur danych, struktury programów, sposób współdziałania elementów składowych oraz wybór technologii, w której zostanie zrealizowane oprogramowanie,
- implementację systemu, a więc przekształcenie modelu projektowego w działający kod programu,
- weryfikację i zatwierdzenie, czyli zweryfikowanie, czy oprogramowanie działa zgodnie ze specyfikacją, oraz sprawdzenie, czy spełnia wymagania użytkownika,
- konserwację, czyli modyfikację oprogramowania związaną ze zmianami wymagań użytkowych oraz późnym wykryciem błędów.

Metoda projektowania zaprezentowana w tym artykule odnosi się do dwóch pierwszych faz projektowania. Ma ona na celu ułatwić opracowanie zarówno modelu analitycznego jak i modelu projektowego. Bazuje ona na podejściu spotykanym przy definiowaniu semantyki języków programowania, a konkretnie na metodzie operacyjnej [2], gdzie znaczenie konstrukcji językowych określane jest poprzez pokazanie jakie operacje wykona pewna maszyna abstrakcyjna. Dziedziną prezentowanej metody projektowej jest robotyka, a więc zostały uwzględnione zasadnicze różnice między robotami a komputerami. Komputery stanowią maszyny o bardzo dobrze zdefiniowanym sposobie działania – efekt wykonania każdego rozkazu jest ściśle określony. Roboty wchodzą w interakcję z niedeterministycznym otoczeniem, a model matematyczny tych urządzeń jest znany jedynie w pewnym przybliżeniu (proces identyfikacji parametrów kinematycznych i dynamicznych oraz uproszczenia strukturalne wprowadzają znaczne niedokładności). Formalizacja podejścia ma na celu redukcję prawdopodobieństwa popełnienia błędów we wczesnych fazach projektowania oprogramowania. Jak wiadomo, w im wcześniejszej fazie projektowania powstanie błąd i im później będzie wykryty tym kosztowniejsze jest jego usunięcie. Należy pamiętać, że roboty są systemami czasu rzeczywistego, a do tego bezpośrednio współpracują z ludźmi, więc niezawodność ich działania ma znaczenie nadrzędne.

1. Ogólne zasady projektowania złożonych układów sterowania

Główne fazy projektowania układów sterowania są następujące:

- wyodrębnienie systemu z otoczenia (zarówno obiektu sterowania jak i samego układu sterującego),
- dekompozycja – podział układu sterowania na części składowe według pewnego kryterium, np. realizowanych zadań lub sprzętu wchodzącego w skład systemu,
- projekt części składowych układu sterowania,
- synteza systemu – złożenie całości z części oraz zapewnienie komunikacji miedzy częściami składowymi.

Jak wspomniano, współczesne układy sterowania w dużym stopniu bazują na oprogramowaniu. Dlatego ten artykuł przede wszystkim koncentruje się na tym składniku układów sterowania. Obecnie powszechnie stosowanymi narzędziami do tworzenia takich układów są programowe struktury ramowe [3] bazujące na:

- wzorcach architekturowych i projektowych sugerujących strukturę projektowanego systemu,
- bibliotekach modułów składowych (ewentualnie ich wzorców), a więc budulcu.

Układy sterowania rozpatrywane w tym artykule składają się z wielu agentów upostaciowionych [8, 10], przy czym przyjmuje się tu dość pojemną definicję agenta, jako dowolnego urządzenia lub programu, mającego zdolność postrzegania swego środowiska i wpływania na jego stan [1, 4], a nadto posiadającego imperatyw wewnętrzny do osiągnięcia celu poprzez swe autonomiczne działanie oraz komunikację z innymi agentami. Ponieważ interesuje nas środowisko fizyczne, więc agenty (urządzenia) muszą mieć fizyczną powłokę cielesną, by być w stanie wpływać

na stan tego środowiska – stąd nazwano je agentami upostaciowionymi. Na wstępie rozważana jest ogólna struktura pojedynczego agenta upostaciowionego.

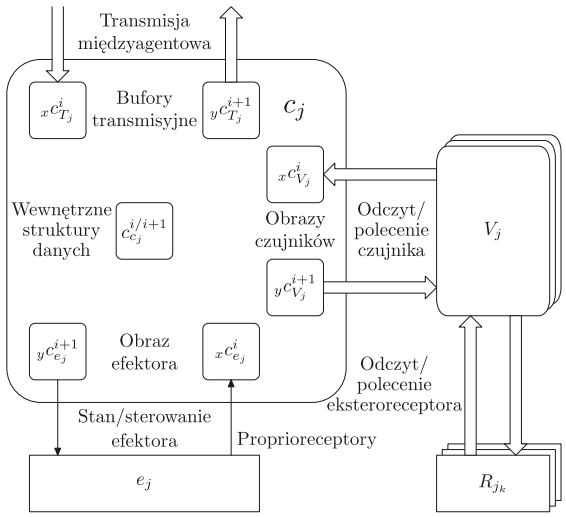
2. Agent upostaciowiony

Rozpatrywane będą systemy składające się z wielu agentów upostaciowionych a_j, $j = 0, \ldots, n_a$, gdzie $n_a + 1$ jest liczbą agentów tworzących system. Stan agenta a_j w chwili dyskretnej i określany jest przez:

$$s_j^i = \langle c_j^i, e_j^i, V_j^i, T_j^i \rangle \tag{1}$$

c_j – stan układu sterującego agenta (stan pamięci, czyli zmiennych, programu sterującego etc.),
e_j – stan efektorów agenta (elementów mechanicznych mogących wpływać na stan środowiska – np. manipulator bądź platforma mobilna),
V_j – zestaw odczytów czujników wirtualnych wykorzystywanych przez agenta,
T_j – informacja nadana/odebrana do/z innych agentów (czyli przesyłane wiadomości).

Dla zachowania zwięzłości zapisu nie odróżnia się tu oznaczeń poszczególnych części składowych agenta od oznaczeń ich stanu. Upływ czasu dyskretnego, czyli przejście od makrokroku i do makrokroku $i+1$ wyznaczany jest przez projektanta w zależności od charakteru obiektu sterowania. Przykładowo dla manipulatora będzie to niewielka wielokrotność okresu próbkowania serwomechanizmów sterujących ruchem efektora, czyli co najwyżej kilka milisekund, a dla autonomicznej platformy mobilnej minimalny czas realizacji przesłanego rozkazu, a więc zazwyczaj kilkaset milisekund. Należy podkreślić, że każdy z agentów, a w wielu przypadkach także czujników wirtualnych, inaczej odmierza upływ czasu, a więc ściśle ujmując powinniśmy operować oznaczeniem i_j, ale dla uproszczenia notacji pominiemy tu indeks j.



Rys. 1. Struktura wewnętrzna agenta upostaciowionego.
Fig. 1. Internal structure of an embodied agent

Czujniki wirtualne służą agregacji danych uzyskanych z receptorów (czujników):

$$v_{j_k}^{i+1} = f_{v_{j_k}}(c_j^i, v_{c_{j_k}}^i, R_{j_k}^i) \tag{2}$$

j – numer agenta,
k – numer czujnika wirtualnego,
R_{j_k} – zestaw odczytów z eksteroreceptorów,
$v_{c_{j_k}}$ – pamięć sensoryczna

$$R_{j_k} = \langle r_{j_{k_1}}, \ldots, r_{j_{k_n}} \rangle \tag{3}$$

n – liczba eksteroreceptorów wykorzystywanych przez k-ty czujnik wirtualny (ściślej n_{j_k}).

$$v_{c_{j_k}}^{i+1} = f_{vc_{j_k}}(c_j^i, v_{c_{j_k}}^i, R_{j_k}^i) \tag{4}$$

Strukturę układu sterującego agenta przedstawia rys. 1. Projektant tworzący układ sterowania operuje pewnymi pojęciami abstrakcyjnymi. Istnieje wiele użytecznych zestawów takich pojęć. W zależności od ich wyboru powstaje pewna ontologia. Stwarza ona obraz (widok) systemu. Ów obraz definiuje się w taki sposób, aby był najwygodniejszy w użyciu przez projektanta. Wspomniane pojęcia muszą być odwzorowane w strukturach danych agenta. Właśnie te struktury danych nazywane są obrazami lub widokami, stanowiącymi bufory interfejsujące układ sterujący agentem z innymi agentami, czujnikami wirtualnymi bądź efektorami. Ich wartości określane są poprzez interakcje układu sterowania z częściami składowymi systemu, dzięki czemu powstają obrazy wejściowe i wyjściowe. Obrazy wejściowe stanowią:

${}_x c_{e_j}$ – proprioreceptywne dane o stanie efektora,
${}_x c_{V_j}$ – eksteroreceptywne dane uzyskane z czujników wirtualnych,
${}_x c_{T_j}$ – informacje otrzymane od innych agentów,

obrazy wyjściowe tworzone są przez:

${}_y c_{e_j}$ – pożądany stan efektora,
${}_y c_{V_j}$ – polecenia dla czujników wirtualnych,
${}_y c_{T_j}$ – informacje przekazywane innym agentom,

natomiast c_{c_j} symbolizuje struktury danych związane z wewnętrzną pamięcią agenta.

Układ sterowania używa:

$${}_x c_j^i = \langle c_{c_j}^i, {}_x c_{e_j}^i, {}_x c_{V_j}^i, {}_x c_{T_j}^i \rangle \tag{5}$$

aby wytworzyć:

$${}_y c_j^{i+1} = \langle c_{c_j}^{i+1}, {}_y c_{e_j}^{i+1}, {}_y c_{V_j}^{i+1}, {}_y c_{T_j}^{i+1} \rangle \tag{6}$$

Do tego celu wykorzystuje funkcje przejścia:

$$\begin{cases} c_{c_j}^{i+1} &= f_{c_{c_j}}(c_{c_j}^i, {}_x c_{e_j}^i, {}_x c_{V_j}^i, {}_x c_{T_j}^i) \\ {}_y c_{e_j}^{i+1} &= f_{c_{e_j}}(c_{c_j}^i, {}_x c_{e_j}^i, {}_x c_{V_j}^i, {}_x c_{T_j}^i) \\ {}_y c_{V_j}^{i+1} &= f_{c_{V_j}}(c_{c_j}^i, {}_x c_{e_j}^i, {}_x c_{V_j}^i, {}_x c_{T_j}^i) \\ {}_y c_{T_j}^{i+1} &= f_{c_{T_j}}(c_{c_j}^i, {}_x c_{e_j}^i, {}_x c_{V_j}^i, {}_x c_{T_j}^i) \end{cases} \tag{7}$$

co można zapisać w zwartej postaci jako:

$${}_y c_j^{i+1} = f_{c_j}({}_x c_j^i) \tag{8}$$

Funkcja przejścia (8) stanowi program działania agenta. Jeżeli znany jest stan początkowy (5), zdefiniowana jest funkcja przejścia (8) oraz znane są przyszłe oddziaływania zewnętrzne, to jesteśmy w stanie określić przyszłe

zachowanie agenta. Innymi słowy, to jak agent będzie się zachowywał jest określane przez projektanta poprzez odpowiednie zdefiniowanie funkcji przejścia (8). Oczywiście pojedyncza funkcja, dla użytecznego agenta, musiałaby mieć bardzo złożoną postać, wiec trzeba ją zdekomponować na funkcje częściowe:

$$ {}_{y}c_j^{i+1} = {}^{m}f_{c_j}({}_{x}c_j^{i}), \qquad m = 1, \ldots, n_f \tag{9} $$

Istnieją różne rodzaje dekompozycji: sekwencyjna, współbieżna, hybrydowa. Każda z funkcji (9), będąca efektem dekompozycji współbieżnej, wytwarza pewien wynik swego działania. Otrzymane wyniki trzeba jakoś wykorzystać, a więc dokonać jakieś formy ich kompozycji w funkcję (8). Wyróżniamy następujące rodzaje kompozycji współbieżnej: promująca współzawodnictwo (antagonistyczna), wspierająca współdziałanie (kooperacyjna), mieszana – wykorzystująca elementy każdej z poprzednich dwóch skrajnych strategii. Różne metody dekompozycji, a następnie kompozycji funkcji przejścia rozważane są w pracach [6, 7, 10]. Dla ilustracji dalszych rozważań ograniczymy się do dekompozycji sekwencyjnej, gdzie w danej chwili tylko jedna cząstkowa funkcja przejścia będzie aktywna. Istnieją problemy wymagające dekompozycji równoległej. Jak sobie radzić w takich przypadkach opisano w [9].

Aby wywołać pożądane zachowanie agenta należy nie tylko zdefiniować odpowiednią funkcję przejścia, ale również spowodować wczytanie jej argumentów (5), wysłanie wartości obliczonych (6) oraz określić jak długo to zachowanie powinno być realizowane (przez ile makrokroków). W związku z tym ostatnim aspektem funkcja przejścia (8) zostaje rozłożona na dwie części składowe: ${}^{m}f'_{c_j}$ oraz ${}^{m}f_{\tau_j}$. Ta pierwsza określa zachowanie, a ta druga determinuje, jak długo owe zachowanie powinno być realizowane – stąd funkcja ta zwana jest warunkiem końcowym. Teraz można już zdefiniować semantykę instrukcji odpowiedzialnej za realizację zachowania. Ponieważ zazwyczaj zachowanie związane jest z wykonaniem czynności, a więc ruchem, więc zdefiniowana zostanie instrukcja ruchu (10) o następującej postaci.

```
loop
  // Sprawdź warunek końcowy (predykat)
  if ᵐf_τj( ₓc_j^i ) = true
      then break // Opuść pętlę
  endif
  // Określ następny stan układu sterowania agenta
  ᵧc_j^{i+1} := ᵐf'_cj( ₓc_j^i );
  // Roześlij wyniki obliczeń
  ᵧc_ej^{i+1} ⇀ e_j;   ᵧc_Vj^{i+1} ⇀ V_j;   ᵧc_Tj^{i+1} ⇀ c_Tj';
  i := i + 1;
  // Określ aktualny stan agenta
  e_j ⇀ ₓc_ej^i;   V_j ⇀ ₓc_Vj^i;   c_Tj' ⇀ ₓc_Tj^i;
endloop
```

(10)

gdzie $\rightharpoonup$ symbolizuje przesłanie danych. Należy zwrócić uwagę, że kod (10) wykorzystuje funkcję przejścia ${}^{m}f'_{c_j}$ zdefiniowaną dla jednego kwantu czasu $i \to i+1$ (makrokrok), co jest względnie łatwe do określenia. Ogólne zachowanie agenta określone jest poprzez zdefiniowanie jak ma się on zachować w małym kwancie czasu (makrokroku). Resztę załatwia powtarzanie w pętli tego mikrozachowania.

Aby zaprojektować układ sterowania, należy system podzielić na agenty, a następnie należy zdefiniować dla każdego z tych agentów zestaw par funkcji $({}^{m}f'_{c_j}, {}^{m}f_{\tau_j})$. Funkcje te wywoływane są przez instrukcję ruchu (10). Struktura programu agenta odpowiada strukturze automatu skończonego, który w każdym ze swych stanów wykonuje jedną instrukcję ruchu o postaci (10), natomiast przejścia pomiędzy tymi stanami wskazywane są na podstawie wartości pewnych dodatkowych predykatów, które stanowią funkcje logiczne określone na argumentach (5).

3. Podsumowanie

W pierwszej części artykułu przedstawiono koncepcję sformalizowanego opisu struktury oraz sposobu działania wieloagentowych układów sterowania robotami. Zastosowanie tego opisu wraz z ogólną metodą tworzenia programów powstałą na gruncie inżynierii oprogramowania prowadzi do tworzenia eleganckich i prostych struktur układów sterowania robotów. W drugiej części artykułu pokazany zostanie przykład zastosowania przedstawionej tu metody projektowej. Przykład będzie dotyczył projektu układu sterowania robota Scout umożliwiający mu autonomiczny dojazd do celu uprzednio wskazanego przez operatora.

Podziękowania

Praca była finansowana z funduszy statutowych PIAP. Pragniemy podziękować następującym studentom Wydziału Elektroniki i Technik Informacyjnych Politechniki Warszawskiej, członkom Koła Naukowego *Bionik*, którzy wzięli udział w implementacji przykładowego układu sterowania: Konradowi Banachowiczowi, Piotrowi Miedzikowi, Maciejowi Stafańczykowi, Kacprowi Szkudlarkowi.

Bibliografia

1. Cetnarowicz, K. (2010): *M–agent*. In: Ambroszkiewicz, S., Borkowski, A., Centarowicz, K., Zieliński, C. (Eds.), Inteligencja wokół nas. Współdziałanie agentów softwareowych, robotów, inteligentnych urządzeń, pp. 137–167, EXIT.
2. Dembiński, P., Małuszyński, J. (1981): *Matematyczne metody definiowania języków programowania*. Wydawnictwa Naukowo–Techniczne.
3. Kaisler, S. (2005): *Software Paradigms*. Wiley Interscience.
4. Russell, S., Norvig, P. (1995): *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, N.J.
5. Sacha, K. (2010): *Inżynieria oprogramowania*. Wydawnictwo Naukowe PWN, Warszawa.
6. Zieliński, C. (2005): *Formal approach to the design of robot programming frameworks: the behavioural control case*. Bulletin of the Polish Academy of Sciences – Technical Sciences 53(1), 57–67.
7. Zieliński, C. (2006): *Transition-Function Based Approach to Structuring Robot Control Software*. In: Kozłowski, K. (Ed.), Robot Motion and Control: Recent

Developments, Lecture Notes in Control and Information Sciences, Vol.335, pp. 265–286, Springer Verlag.
8. Zieliński, C. (2010): *Formalne podejście do programowania robotów – struktura układu sterującego*. In: Ambroszkiewicz, S., Borkowski, A., Centarowicz, K., Zieliński, C. (Eds.), Inteligencja wokół nas. Współdziałanie agentów softwareowych, robotów, inteligentnych urządzeń, pp. 267–300, EXIT.
9. Zieliński, C., Trojanek, P. (2009): *Stigmergic cooperation of autonomous robots*. Journal of Mechanism and Machine Theory 44, 656–670.
10. Zieliński, C., Winiarski, T. (2010): *Motion Generation in the MRROC++ Robot Programming Framework*. International Journal of Robotics Research 29(4), 386–413.

Method of Designing Autonomous Mobile Robot Control Systems. Part 1: Theoretical Introduction

Abstract: This two-part paper presents a general method of designing both the structure as well as the method of operation of autonomous robot control systems. The first part of the paper describes the notation and the general aspects of the design method. The second part presents an example of the utilization of the proposed method of designing robot controllers.

Keywords: robot control systems, autonomous robots

prof. nzw. dr hab. inż. Cezary Zieliński

Od 2008 roku pracuje w Przemysłowym Instytucie Automatyki i Pomiarów. Ponadto jest profesorem nadzwyczajnym Politechniki Warszawskiej na Wydziale Elektroniki i Technik Informacyjnych. W latach 2002-2005 sprawował na tym wydziale funkcję prodziekana ds. nauki i współpracy międzynarodowej, 2005-2008 zastępcy dyrektora Instytutu Automatyki i Informatyki Stosowanej (IAiIS) ds. naukowych, a od 2008 pełni funkcję dyrektora tego instytutu. Od uzyskania habilitacji w roku 1996 pełni rolę kierownika Zespołu Robotyki w IAiIS. Od 2007 roku jest członkiem i sekretarzem Komitetu Automatyki i Robotyki Polskiej Akademii Nauk. Jego zainteresowania badawcze koncentrują się na zagadnieniach związanych z programowaniem i sterowaniem robotów.

e-mail: c.zielinski@ia.pw.edu.pl

mgr inż. Tomasz Kornuta

Absolwent Wydziału Elektroniki i Technik Informacyjnych Politechniki Warszawskiej. W 2003 roku uzyskał tytuł inżyniera, w 2005 tytuł magistra inżyniera. Od 2008 roku pracuje na etacie asystenta w Instytucie Automatyki i Informatyki Stosowanej (IAiIS), w ramach którego prowadzi zajęcia dydaktyczne, a od 2009 roku pełni funkcję Kierownika Laboratorium Podstaw Robotyki. Dodatkowo od 2009 roku jest zatrudniony na etacie konstruktora, w ramach którego realizuje prace związane z grantem europejskim. Od 2005 roku w ramach doktoratu prowadzi badania związane z wykorzystaniem przez roboty paradygmatu aktywnego czucia do analizy otoczenia. Jego główne zainteresowania naukowe obejmują wykorzystanie informacji wizyjnej w robotyce.

e-mail: tkornuta@ia.pw.edu.pl

mgr inż. Piotr Trojanek

Jest doktorantem oraz pracuje jako konstruktor w Instytucie Automatyki i Informatyki Stosowanej Politechniki Warszawskiej. Swoje doświadczenie zdobywał pracując nad układami sterowania robotów mobilnych. Obecnie prowadzi badania dotyczące systemów wieloagentowych oraz zastosowań metod inżynierii oprogramowania w robotyce. Od lat związany ze studenckim kołem naukowym Bionik.

e-mail: piotr.trojanek@gmail.com

dr inż. Tomasz Winiarski

Jest adiunktem w Instytucie Automatyki i Informatyki Stosowanej Politechniki Warszawskiej. Sprawuje funkcje kierownika laboratorium robotyki w macierzystym instytucie, a także opiekuna studenckiego koła naukowego Bionik, które współtworzył. W 2010 roku otrzymał za osiągnięcia naukowe nagrodę indywidualną drugiego stopnia rektora PW. Jego zainteresowania badawcze dotyczą z jednej strony konstrukcji i nawigacji robotów mobilnych z drugiej strony specyfikacji zadań manipulatorów i chwytaków ze szczególnym uwzględnieniem sterowania pozycyjno-siłowego.

e-mail: tmwiniarski@gmail.com

Lista recenzentów artykułów punktowanych, zamieszczonych w miesięczniku PAR Pomiary Automatyka Robotyka

Grega Wojciech, prof. dr hab. inż.
Kaczanowski Stanisław, dr inż., prof. PIAP
Kaliczyńska Małgorzata, dr inż.
Korytkowski Jacek, dr inż., prof. PIAP
Masłowski Andrzej, prof. dr hab. inż.
Missala Tadeusz, prof. dr inż.
Szewczyk Roman, dr hab. inż., prof. PW
Tarczyński Wiesław, dr hab. inż., prof. PO
Trojnacki Maciej, dr inż.
Warsza Zygmunt, doc. dr inż.
Wrzuszczak Maria, dr inż.
Zator Sławomir, dr hab. inż., prof. PO