# Architektura systemu wielorobotowego w oparciu o paradygmat SOA*

*Stanisław Ambroszkiewicz*[1], *Waldemar Bartyna*[2]

**Streszczenie**

W pracy przedstawiono nowe podejście do agenta softwarowego oraz systemów wieloagentowych oparte na paradygmacie Service Oriented Architecture. Na tej podstawie zaproponowano nowe podejście do architektury robota i systemów wielorobotowych. Te architektury mają służyć do współdziałania heterogenicznych robotów w wykonywaniu złożonych zadań. Podstawowym założeniem jest traktowanie poszczególnych możliwości robota jako indywidualnych usług, które mogą być dynamicznie komponowane, synchronizowane i wywoływane w celu realizacji złożonego zadania.

## 1. KONCEPCJA AGENTA

Programowy agent powinien obejmować trzy charakterystyki: autonomiczność, adaptacyjność i współpracę. Autonomiczność oznacza to, że agent posiada własne cele a jego zachowanie jest ukierunkowane na realizację tych celów. Zdolność przystosowania się do środowiska jest widziana jako zdolność agenta do uczenia się z wcześniejszych doświadczeń tak, aby udoskonalać swoje możliwości osiągnięcia celów w zmieniającym się otoczeniu. Współpraca i koordynacja powinna angażować, co najmniej dwóch agentów tworzących system wieloagentowy (MAS). W przeciwieństwie do pojedynczych agentów, agenci w MAS współpracują wzajemnie ze sobą dążąc do osiągnięcia celu. Ogólnie mówiąc, MAS może być traktowany jako zbiór agentów usytuowanych w środowisku, wchodzących w interakcje nawzajem ze sobą i ze środowiskiem w celu wypełnienia określonego zadania. Zatem, MAS posiada cztery podstawowe komponenty: agentów, środowisko, interakcje i cele. Interakcje odgrywają zasadniczą rolę w MAS, ponieważ definiują one relację między działaniami wykonywanymi przez agentów. Komunikacja (w języku symbolicznym) pomiędzy agentami może być również traktowana jako specyficzny rodzaj interakcji. Podobnie sprawa ma się z postrzeganiem środowiska przez agenta; w zależności od sensorów interakcja taka może być mniej lub bardziej skomplikowana.

Złożoność MAS wyznaczana jest przez następujące, powiązane ze sobą czynniki: Środowisko + Agenci + Interakcje + Cele. Tak, więc złożoność MAS

*Wyniki prezentowane w tej pracy zostały zrealizowane w ramach grantu MEiN 3 T11C 038 29.
[1]Instytut Informatyki Akademii Podlaskiej, ul. Sienkiewicza 51, 08 – 110 Siedlce, sambrosz@ipipan.waw.pl, www.ipipan.waw.pl/mas/stan/ .
[2]Instytut Podstaw Informatyki Polskiej Akademii Nauk, ul. Ordona 21, 01-237 Warszawa.

może być rezultatem złożoności celów nawet, jeśli środowisko i interakcje są stosunkowo proste. MAS może być definiowany w oparciu o te cztery czynniki. Z punktu widzenia realizacji MAS, głównym zagadnieniem jest wybór architektury agenta będącej w stanie zapewnić efektywne funkcjonowanie całego systemu MAS. Istnieją dwa podstawowe podejścia do architektury robota: reaktywny agent i deliberatywny agent. Agent reaktywny jest zdefiniowany poprzez funkcję bodziec – reakcja, taką, że dla danego bodźca ze środowiska i obecnego stanu agenta wyznaczana jest reakcja. Reaktywny agent nie dysponuje mechanizmami rozumowania i planowania i nie jest proaktywny.

Deliberatywny agent może być postrzegany jako podejście „top down" do tworzenia architektury agenta. Posiada on symboliczną reprezentację świata, jak również kilka dodatkowych mentalnych atrybutów związanych ze świadomością, motywacją i intencjami. Na podstawie tych atrybutów agent przeprowadza planowanie, rozumowanie itp. Architektura agenta, zwana BDI, może tutaj stanowić najbardziej znany przykład. Inspiracja dla pojęć Belief, Desire oraz Intentions (przekonanie, pragnienie i intencje) jest zapożyczona z psychologicznego podejścia. Te mentalne atrybuty (przekonanie, pragnienie i intencje) traktowane są jako prymitywne pojęcia, na których oparte jest rozumowanie i planowanie. Architektura ta jest powrotem do tradycyjnego nurtu symbolicznej Sztucznej Inteligencji.

Istnieje szeroki wachlarz potencjalnych architektur agenta mogących oscylować między reaktywnym i celowym agentem. W rzeczywistości, reaktywna architektura jest bardzo prosta, podczas gdy celowa bardzo złożona. Koncepcja architektury hybrydowej próbuje przezwyciężyć ograniczenia celowej i reaktywnej architektury poprzez ich połączenie. Zazwyczaj, architektury te złożone są z kilku warstw. Każda warstwa odpowiedzialna jest za inne działania. Dolna, *oparta na zachowaniu*, warstwa implementuje reaktywne zachowanie i wiedzę proceduralną. Środkowa, *oparta na planowaniu*, warstwa zawiera mechanizmy umożliwiające konstruowanie indywidualnych planów dla agenta uwzględniające jego aktualną wiedzę o otaczającym świecie. Górna warstwa *współdziałania* zdolna jest do generowania planów współpracy wykorzystując odpowiednia wiedzę. Hybrydowa architektura nie przezwycięży złożoności architektury agenta celowego. Właściwie rozszerza deliberatywną architekturę poprzez dodanie reaktywnych komponentów. Dlatego, dla niektórych bodźców agent hybrydowy zachowa się reaktywnie według najniższej warstwy architektury, podczas gdy dla pozostałych jego zachowanie będzie ustalone na podstawie części celowej.

Architektura Agenta ACT [1, 2] i architektura M-agenta [6] mogą być przykładami całkowicie różnych podejść do reprezentacji ogólnej struktury i funkcjonowania agenta.

# 2. ARCHITEKTURA AGENTA I ZŁOŻONOŚĆ MAS

Współdziałanie jest kluczowym pojęciem w MAS, ponieważ może służyć jako test do oceny architektury agenta. Architektury BDI odznaczają się dobrą wydajnością w stosunkowo prostym środowisku – z małą ilością homogenicznych agentów. Techniki współdziałania (takie jak budowanie formacji i rekonfiguracja zespołów robotów), wypracowane dla tego rodzaju architektury, oparte są na zbiorowej wiedzy, zbiorowych celach, zbiorowych intencjach i zaangażowaniu. Architektury agenta reaktywnego są dobrym rozwiązaniem dla prostego, dużego i niezbyt złożonego strukturalnie środowiska z dużym, jednolitym zespołem agentów. W takich środowiskach można zaobserwować pojawianie się wzorów współdziałania tych agentów.

Z tych powodów, architektura reaktywnego agenta, jak również deliberatywnego oraz hybrydowego, może być stosowana w MAS charakteryzujących się stosunkowo małą złożonością. Oznacza to, że istnieje krytyczny poziom złożoności MAS, po przekroczeniu, którego żadna z powyżej opisanych architektur nie będzie odpowiednia; agenci, stworzeni według tych architektur będą niewystarczający i/lub niezdolni do realizacji ich celów. Ten właśnie krytyczny poziom może być najwyraźniej zauważony w środowisku, wielorobotowym, gdzie stosunkowo proste zadania (aby mogły zostać wykonane) wymagają pewnej liczby heterogenicznych robotów, które muszą sporządzić wspólny plan a następnie koordynować swoje działania. Prosty sposób reprezentacji robotów jako agentów (reaktywnych, celowych lub hybrydowych) czyni system za prostym do realizacji zadań w przypadku reaktywnych agentów, lub czyni mechanizm robota (odpowiedzialny za postrzeganie, rozumowanie, planowanie i wykonywanie zadań) zbyt złożonym dla efektywnej implementacji, np. do działania w czasie rzeczywistym. Wydaję się, że nie istnieje bezpośrednie odwzorowanie pomiędzy robotami a agentami, tzn. nie jest jasne, czy robot może być reprezentowany przez jednego agenta. Tak więc, problem jest następujący: *Jak (i dlaczego) agenci powinni być zlokalizowani w systemie wielorobotowym?*

# 3. AGENCI, SERWISY INTERNETOWE ORAZ SIECI SEMANTYCZNE

Współdziałanie heterogenicznych, kognitywnych robotów w otwartym (z założenia nieznanym) środowisku jest wciąż jednym z najbardziej wyzywających problemów w robotyce. Współpraca ta oznacza zdolność do wykonywania nowych złożonych zadań przez roboty. Ponieważ roboty mogą być heterogeniczne, sposób wykonania zadania nie może być wpisany w zachowanie robota. Dlatego też, roboty muszą się komunikować i rozumieć się wzajemnie. To zrozumienie (semantyczne współdziałanie) powinno być osiągnięte na poziomie protokołów i ogólnej, symbolicznej reprezentacji *świata rzeczywistego*. Współdziałanie jest konieczne do

koordynowania działań robotów, synchronizacji wykonywania zadań, do sprawdzania rezultatów i przekazywania ich pozostałym robotom.

Większość z obecnych badań (zobacz [7, 8, 9, 10, 11, 12]) w tej dziedzinie próbuje osiągnąć współdziałanie na poziomie oprogramowania poprzez tworzenie pakietów oprogramowania i API (zazwyczaj w C++) do realizacji funkcji robota np. (bezprzewodowa) komunikacja, generowanie i interpolowanie ruchu, kinematyka i dynamika, itp. Mimo tego, że problem semantycznej współpracy w kontekście robotów jest trudny i złożony, ma on wiele wspólnego z ideą Semantic Web. Doświadczenia z realizacji Semantic Web jasno pokazują, że współdziałanie nie może być osiągnięte na poziomie oprogramowania. Kluczowym elementem zapewnienia semantycznej współpracy jest skonstruowanie ogólnej reprezentacji rzeczywistego środowiska; jest to jednakże bardzo trudny problem.

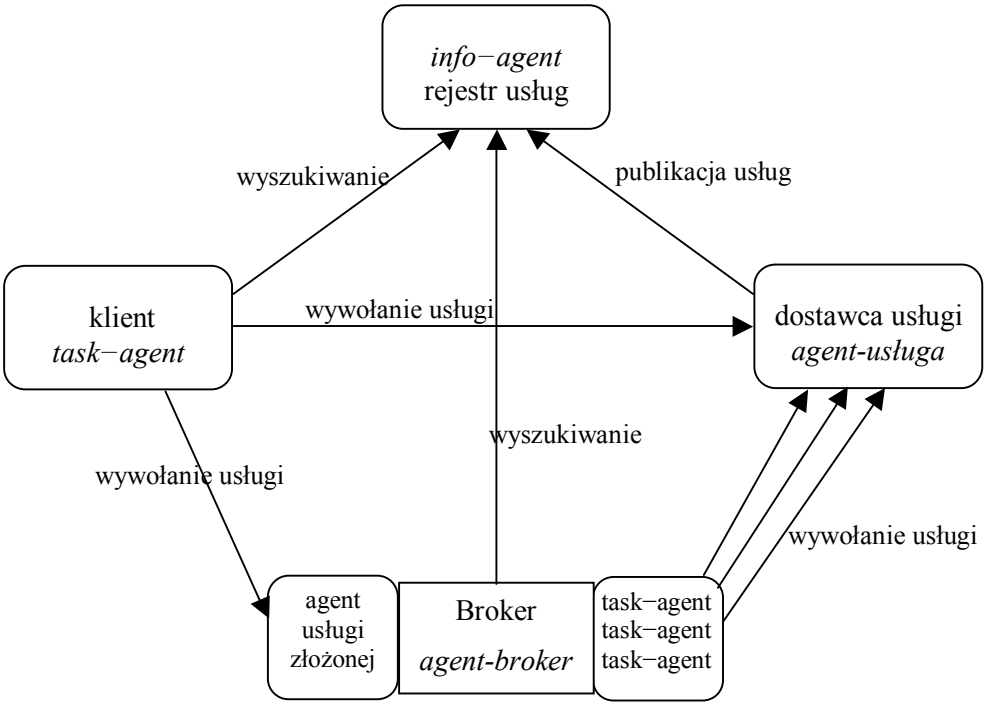
W systemie wielorobotowym każdy indywidualny robot posiada pewne możliwości wykonywania specyficznych dla niego działań i zadań. Każda z takich możliwości może być postrzegana jako *usługa*, która może być udostępniona do wykorzystania przez inne roboty. Przy realizacji złożonego zadania, roboty mogą współdziałać i połączyć niektóre z ich usług w odpowiedniej kolejności, tak aby wykonać to zadanie. Dlatego też współdziałanie w systemie wielorobotowym ma dużo wspólnego z ideą Web serwisów, jednakże, jest dużo bardziej złożone, ponieważ dotyczy rzeczywistego środowiska.
W kontekście systemu wielorobotowego, wydaje się sensownym połączenie koncepcji MAS z koncepcją SOA tj. Service Oriented Architecture (architektura zorientowana na usługi), która jest podstawą do realizacji Web serwisów.

# 4. MAS I ARCHITEKTURA ZORIENTOWANA NA USŁUGI

Service Oriented Architecture (patrz rys. 1) jest paradygmatem programistycznym dostarczającym standardowego modelu programowania, który pozwala, z jednej strony, aplikacjom rezydującym w sieci (nazwanych tutaj usługami sieciowymi), na automatyczne opublikowanie tego co one wykonują, a z drugiej strony, innym aplikacjom (nazwanych tutaj klientami) na odkrycie tych usług i ich zdalne wywołanie (uruchomienie). Tak wiec są trzy zasadnicze komponenty w SOA: Service Provider (dostawca usługi), Service Requester lub Client (klient korzystający z usługi) i Service Registry (rejestr usług). Dostawca stawia usługę, kontroluje do niej dostęp i jest odpowiedzialny za opublikowanie opisu jego usługi w rejestrze usług. Klient jest komponentem programowym szukającym innego komponentu, który mógłby wywołać w celu zrealizowania żądania. Rejestr usług jest centralnym repozytorium, które ułatwia wykrywanie usług klientom. Istnieć może również czwarty komponent zwany Brokerem (zachowujący się jako pośrednik), który zachowuje się jako usługa złożona wobec klienta, a jako klient

wobec elementarnych usług. Broker może zwiększyć efektywność działania systemu.

Wyróżnić można cztery rodzaje agentów w SOA (zobacz rys. 1): *service-agent* reprezentujący usługę, *task-agent* reprezentujący klienta mającego (złożone) zadanie do wykonania, *info-agent* reprezentujący rejestr, i *broker-agent* reprezentujący brokera. Interakcje między agentami są jasno zdefiniowane w SOA, jednakże musi istnieć język komunikacji i protokoły implementujące wspomniane interakcje. Dostępnych jest kilka propozycji języka i protokołów, wystarczy wspomnieć te proponowane przez IBM i Microsoft (SOAP+WSDL+UDDI++), DAML-S proponowany przez DRAPA i naszą własną propozycję zwaną enTish [4, 5].




**Rys. 1**. Service Oriented Architecture i SO-MAS.

System wieloagentowy wyspecyfikowany powyżej przy użyciu architektury zorientowanej na usługi, może być postrzegany jako specjalny rodzaj MAS. Taki rodzaj systemu wieloagentowego może być nazywany Service Oriented MAS, w skrócie: SO-MAS.

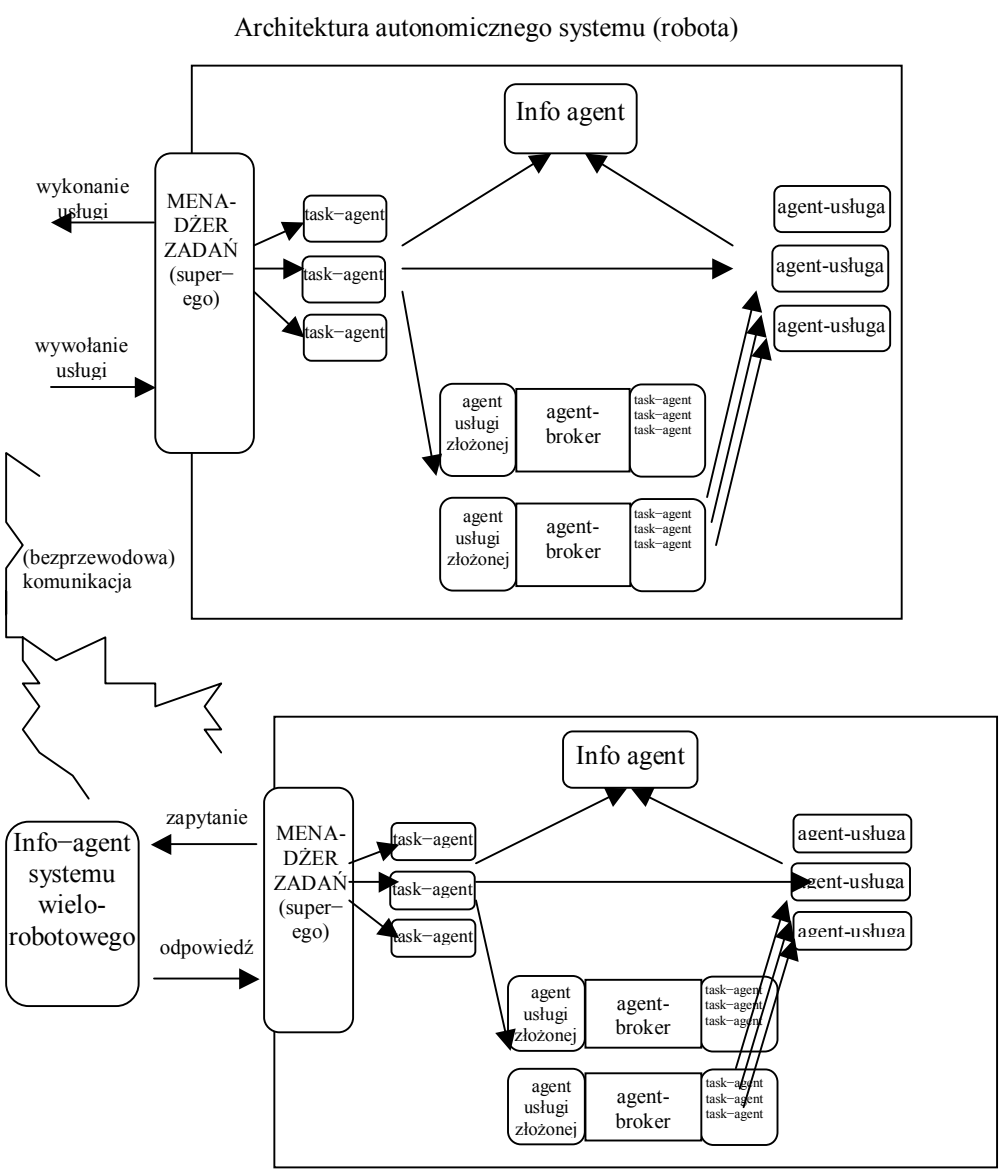
W kontekście wielorobotowym (zobacz rys. 2), robot może być traktowany jako zbiór agentów-usług (odpowiadających jego usługom), jak również, w tym samym czasie, jako agent-klient wykonujący zadanie, agent-broker lub nawet jako info-agent. Kluczowym pojęciem w architekturze robota jest jego autonomia i cele. Pomimo tego, że możliwości robota mogą być wystawione jako usługi dla innych

robotów, nadal kontrolowane są przez robota właściciela. Z drugiej strony, robot (aby mógł zrealizować swój cel) może potrzebować usług kontrolowanych przez inne roboty. Aby zrealizować taką kontrolę usługi, sensownym staje się wprowadzenie specjalnego rodzaju agenta zwanego menadżerem zadań (implementującego autonomię robota, zobacz rys. 2) odpowiedzialnego za wykonanie zadania wewnątrz robota tj. poprzez użycie jego własnych usług. Z punktu widzenia wewnętrznych usług robota, menadżer zadań zachowuje się jako task-agent. I faktycznie do realizacji specyficznego zadania menadżer powołuje dedykowanego task-agenta. Z zewnętrznego punktu widzenia (tj. systemu wielorobotowego) menadżer zadań reprezentuje robota i zachowuje się jako zbiór agentów-usług, wykonujących złożone usługi dla innych agentów. Ponieważ ci agenci-usługi są kontrolowani przez robota (tzn. przez jego menadżera zadań), menadżer zadań może być postrzegany z zewnątrz jako specjalna instancja agenta-brokera.

Architektura wewnętrzna robota może zawierać kilka typów agentów: agentów-usługi, task-agentów a nawet info-agentów i agentów-brokerów, jeśli funkcjonalność danego robota jest wystarczająco złożona. Biorąc pod uwagę autonomiczność robota, dobrym pomysłem jest potraktowanie wewnętrznej architektury robota jako zamkniętego SO-MAS z ustalonym zbiorem agentów-usług, zobacz rys. 2. Wówczas, z zewnętrznego punktu widzenia, robot (jako autonomiczny system) może być postrzegany jako agent-broker oferujący złożone usługi dla zewnętrznych robotów, lub jako task-agent posiadający pewne (złożone) zadanie, które wymaga wykonania zewnętrznych usług (oferowanych przez inne roboty). Ponieważ menadżer zadań implementuje autonomiczność robota, może być uważany za super-agenta tego robota, lub jako jego super-ego.

Spróbujmy się teraz zastanowić, czym jest system wielorobotowy w intuicyjnym znaczeniu. Składa się on z wielu robotów, być może heterogenicznych, umieszczonych w pewnym wspólnym środowisku. Najbardziej interesujące systemy to takie, w których roboty nie znają a priori tego środowiska oraz system jest otwarty, to znaczy, nowe roboty oraz nowe komponenty systemu mogą pojawiać się w systemie i znikać dynamicznie. Roboty, a w szerszym kontekście również urządzenia, którymi można sterować, mogą być widziane w systemie jako mniej lub bardziej złożone usługi. Przeznaczeniem systemu wieloobrotowego jest niewątpliwie realizacja złożonych zadań. Te zadania powinny być wykonywane poprzez kompozycje i skoordynowane wykonanie niektórych z usług dostępnych w systemie. Poszczególne zadanie powinno mieć agenta odpowiedzialnego za jego wykonanie, i oczywiście można go nazwać task-agentem. Taki task-agent powinien mieć możliwość uzyskania informacji czy są i gdzie są usługi potrzebne do jego zadania. A więc potrzebne będzie repozytorium gdzie usługi będą mogły być z jednej strony rejestrowane (przez nowe roboty pojawiające się w systemie) oraz odkrywane przez task-agentów. Do rejestracji oraz odkrywania potrzebne są specjalne protokoły. Również potrzebny jest osobny protokół do kompozycji i wywoływania usług. Niektóre wyspecjalizowane roboty mogą działać w systemie jako brokerzy wyspecjalizowani w realizacji pewnych specyficznych zadań, nie

koniecznie angażujących ich własne możliwości (usługi), ale również możliwości innych urządzeń dostępnych w systemie. Wniosek z powyższych rozważań jest następujący: otwarty i heterogeniczny system wielorobrotowy można konstruować w oparciu o architekturę SO-MAS.




**Rys. 2**. Architektura SO-MAS robota i systemu wieloagentowego.

Tak, więc, idea zastosowania architektury SO-MAS systemach wieloobrotowych jest bardzo ciekawa, można ja stosować zarówno do konstrukcji wewnętrznej struktury robota, jak również struktury całego systemu wielorobotowego. Różnica może polegać na tym, że architektura wewnętrzna robota widziana jest raczej jako zamknięty SO-MAS, podczas, gdy system wielorobotowy jest traktowany jako otwarty SO-MAS.
Nie koniecznie tak być musi, ponieważ równie dobrze można traktować architekturę robota jako otwartą zakładając, że można do takiego robota dokładać dodatkowe urządzenia (usługi) na zasadzie plug-and-play. Z drugiej strony nic nie stoi na przeszkodzie, żeby zamknąć system wielorobotowy ustalając środowisko i komponenty systemu.

Autonomia (robota) może być widziana jako istotną różnica pomiędzy robotem a systemem wielorobotowym. Ale ta autonomia sprowadza się w architekturze SO-MAS do agenta menadżera, który wyznacza zadania realizowane przez robota. Ponieważ również w systemie wielorobotowym, pojawiają się zadania (do wykonania przez ten system) założone przez projektanta tego systemu, więc celowe jest powołanie specjalnego dedykowanego agenta do wyznaczania takich zadań. Nic nie stoi na przeszkodzie żeby nazwać takiego agenta menadżerem zadań dla takiego systemu.

Powyższe argumenty mogą sugerować, że z punktu widzenia architektury nie istnieje zasadnicza różnica pomiędzy robotem a systemem wielorobotowym. Teza ta może to wydawać się nieco zaskakująca, ale jest ona w pełni uzasadniona biorąc pod uwagę funkcjonalność zarówno robota jak i systemu wieloobrotowego zamierzoną przez ich projektantów. Ta funkcjonalność to oczywiście realizacja złożonych zadań w często nieznanych i nieprzewidywalnych środowiskach.

Warto zauważyć jeszcze jedną ciekawą cechę naszego podejścia. Otóż, architektura SO-MAS jest rekurencyjna – agent-broker osadzony we wewnętrznej architekturze robota może być uważany za pół-autonomiczny system. Tym sposobem wewnętrzna architektura agenta-brokera może być reprezentowana również jako zamknięty SO-MAS. Z drugiej strony wyposażając system wielorobotowy w agenta menadżera zadań zamykamy ten system. Wówczas ten menadżer zadań reprezentuje na zewnątrz ten system wielorobotowy jako agent-broker potrafiący wykonywać specyficzny typ złożonych zadań. Właściwie nic nie stoi na przeszkodzie, żeby takiego agenta-brokera uważać za robota. Takie rekurencyjne podejście do architektury złożonych systemów może być widziane jako propozycja na przezwyciężenie problemu złożoności w systemach wielorobotowych.

# 5. KONLKUZJE I WSTĘPNE WNIOSKI

Bardzo często utożsamia się system autonomiczny (np. robota) z pojedynczym agentem. Praca [14] wyraźnie wskazuje, że tak być nie musi, a wręcz przeciwnie. Biorąc pod uwagę efektywność działania systemu, warto zrewidować tę koncepcję i

spojrzeć na agenta (jako na podmiot kontrolujący działanie systemu) zupełnie inaczej. Zamiast pojedynczego agenta może to być rozproszony system wieloagentowy. Również w ramach takiego systemu warto reprezentować niektórych agentów również jako złożone systemy wieloagentowe. Prowadzi to do hierarchicznej architektury w systemach wieloagentowych, która w znaczny sposób może redukować złożoność takich systemów.
Z drugiej strony w tej pracy została przedstawiona propozycja oparcia architektury MAS na paradygmacie SOA. W sumie stanowi to ciekawą propozycję nowego podejścia do architektury systemów wieloagentowych i samego agenta.
Architektura jest rekurencyjna (może być zagnieżdżona), ponieważ jej wewnętrzna architektura może zawierać agentów-brokerów, którzy są także skonstruowani według proponowanej architektury. Jednakże proces zagnieżdżania agentów nie może iść w nieskończoność; musi istnieć prymitywny, elementarny agent, na którym kończy się rekurencja.

Zaskoczeniem na pewno jest fakt, że jedna ogólna i jednolita architektura SO-MAS może służyć zarówno jako podstawa do konstrukcji struktury pojedynczego robota jak i systemu wielorobotowego. Jednakże środowiska reprezentowane przez SO-MAS są całkowicie różne. W pierwszy przypadku jest to zamknięta, sztuczna wewnętrzna struktura autonomicznego systemu, podczas, gdy w drugim, jest to nieprzewidywalne, otwarte i złożone rzeczywiste środowisko.

Interesującą kwestią jest tworzenie zadań w systemie wielorobotowym. Zadanie może powstawać na podstawie stałego celu zadanego robotowi do realizacji. Może ono pochodzić od nadzorcy, zaimplementowanego na zdalnym stanowisku w Internecie. Wykonanie takich zadań, może stać się nowym celem robota. Wspomniany nadzorca może być zaimplementowany jako menadżer zadań w systemie wielorobotowym. Dlatego też, ponownie, możemy uważać system wielorobotowy z nadzorcą jako pojedynczego agenta, który ten system reprezentuje na zewnątrz.

Podsumowując, w pracy zaprezentowano kilka nowych pomysłów dotyczących architektury agenta i systemu wieloagentowego w kontekście robotyki. Pomysły te bazowane są na naszych wieloletnich badaniach i doświadczeniach w teorii i technologiach związanych z agentami. Mimo tego, że są one jasno określone, istnieje wciąż wiele do zrobienia, aby móc im zapewnić kompletną, teoretyczną strukturę, i co ważniejsze, stworzyć technologię realizującą i weryfikującą te pomysły.

Ponieważ istnieją standardy języków i protokołów do realizacji SO-MAS, ta idea może być zweryfikowana. Weryfikacja musi być przeprowadzona na prawdziwych robotach w rzeczywistym środowisku. W tym celu wykorzystujemy roboty Pioneer firmy Activ Media, oraz system enTish [4, 5] do implementacji architektury SO-MAS na indywidualnych robotach jak i dla systemu wielorobotowego. Kluczowym zagadnieniem jest tutaj ogólna reprezentacja rzeczywistego środowiska, ponieważ zadania i opisy usług muszą być wyrażone w języku, który korzysta z takiej reprezentacji. Jest to przedmiotem rozważań równoległej pracy [13] zgłoszonej również na IX KKR.

# LITERATURA

[1] W. Truszkowski, Ch. Rouff. New Agent Architecture for Evaluation in Goddard's Agent Concepts Testbed. NASA Goddard Space Flight Center. http://agents.gsfc.nasa.gov/papers.

[2] W. Truszkowski, H. Hallock. Agent Technology from a NASA Perspective. NASA Goddard Space Flight Center. http://agents.gsfc.nasa.gov/papers.

[3] S. Ambroszkiewicz.. Agent Based Approach to Service Description and Composition. *Innovative Concepts for Agent-Based Systems: First International Workshop on Radical Agent Concepts* WRAC 2002 McLean, VA, USA, January 16-18, 2002 Revised Papers. Springer LNCS 2564, ISSN: 0302-9743, s. 135 – 149, 2003.

[4] S. Ambroszkiewicz.. enTish: *An Approach to Service Description and Composition*. Instytut Podstaw Informatyki Polskiej Akademii Nauk, Warszawa 2003, ISBN 83-910948-7-1, http://www.ipipan.waw.pl/mas.

[5] S. Ambroszkiewicz.. Entish: A Language for Describing Data Processing in Open Distributed Systems. *Fundamenta Informaticae* Vol. 60, No. 1-4, April 2004, ISO Press, s. 41-66.

[6] K. Cetnarowicz, E. Nawarecki, M. Zabinska. M-agent architecture oriented technology. Proc. of Distributed Artificial Intelligence, Russia, 1997.

[7] Software for Distributed Robotics, DARPA, http://www.darpa.mil/ipto/programs/sdr/index.htm.

[8] Joint Robotics Program. http://www.jointrobotics.com.

[9] Joint Architecture for Unmanned Systems. http://www.jauswg.org.

[10] Orocos project. http://www.orocos.org.

[11] The Player/Stage Project. http://sourceforge.net/projects/playerstage.

[12] Orca project. http://orca-robotics.sourceforge.net.

[13] S. Ambroszkiewicz. Reprezentacja przestrzenna środowiska na podstawie kognitywnych map obiektowych. Materiały z IX Krajowej Konferencji Robotyki, KKR9, Piechowice 13 - 16 września 2006.

[14] S. Ambroszkiewicz, K. Cetnarowicz. On the concept of agent in multi-robot environment. In Proc. WRAC-2005. To be published by Springer LNAI.

## A CONCEPT OF MULTIROBOT SYSTEM BASED ON THE PARADIGM OF SOA

A new approach (based on the paradigm of Service Oriented Architecture) to agent as well as to multiagent system is presented. This gives rise to define a new architecture for robot as well as for multirobot system.