Pomiary Automatyka Robotyka, ISSN 1427-9126, R. 23, Nr 3/2019, 41–54, DOI: 10.14313/PAR_233/41

Agentowa struktura wielomodalnego interfejsu do Narodowej Platformy Cyberbezpieczeństwa, część 1.

Włodzimierz Kasprzak, Wojciech Szynkiewicz, Maciej Stefańczyk, Wojciech Dudek, Maksym Figat, Maciej Węgierek, Dawid Seredyński, Cezary Zieliński
Politechnika Warszawska, Wydział Elektroniki i Technik Informacyjnych, Instytut Automatyki i Informatyki Stosowanej, Nowowiejska 15/19, 00-665 Warszawa

Streszczenie: Ten dwuczęściowy artykuł przedstawia interfejs do Narodowej Platformy Cyberbezpieczeństwa (NPC). Wykorzystuje on gesty i komendy wydawane głosem do sterowania praca platformy. Ta część artykułu przedstawia strukturę interfejsu oraz sposób jego działania, ponadto prezentuje zagadnienia związane z jego implementacją. Do specyfikacji interfejsu wykorzystano podejście oparte na agentach upostaciowionych, wykazując że podejście to może być stosowane do tworzenia nie tylko systemów robotycznych, do czego było wykorzystywane wielokrotnie uprzednio. Aby dostosować to podejście do agentów, które działają na pograniczu środowiska fizycznego i cyberprzestrzeni, należało ekran monitora potraktować jako część środowiska, natomiast okienka i kursory potraktować jako elementy agentów. W konsekwencji uzyskano bardzo przejrzystą strukturę projektowanego systemu. Część druga tego artykułu przedstawia algorytmy wykorzystane do rozpoznawania mowy i mówców oraz gestów, a także rezultaty testów tych algorytmów.

Słowa kluczowe: Narodowa Platforma Cyberbezpieczeństwa, rozpoznawanie obrazu, rozpoznawanie gestów, rozpoznawanie mowy, rozpoznawanie mówcy

1. Wprowadzenie

Termin *cyberbezpieczeństwo* odnosi się do zbioru zagadnień związanych z zapewnieniem bezpieczeństwa użytkowania sieci komputerowych, a w szczególności Internetu. Dotyczy on zabezpieczenia infrastruktury sieciowej używanej przez agendy państwa, prywatne korporacje oraz indywidualnych użytkowników przed cyberatakami. Działania związane z cyberbezpieczeństwem przede wszystkim koncentrują się na zapobieganiu cyberatakom, ale jeżeli one już nastąpią, to na maksymalnym ograniczeniu ich skutków. Podstawowym celem jest zapobieganie wyciekom danych oraz udaremnianie ataków polegających na uniemożliwianiu realizacji usług dostarczanych przez sieć.

Stopień zależności współczesnych państw od systemów teleinformatycznych osiągnął taki poziom, że przypadkowe lub zamierzone uszkodzenie tej infrastruktury może pociągnąć za sobą katastrofalne skutki. Dlatego potrzebne są narzędzia służące wykrywaniu, zapobieganiu oraz minimalizacji skutków działań naruszających bezpieczeństwo infrastruktury teleinformatycznej istotnej dla funkcjonowania państwa oraz jego obywateli. W tym celu tworzona jest Narodowa Platforma Cyberbezpieczeństwa (NPC), która umożliwi bieżące monitorowanie funkcjonowania sieci oraz zapewni koordynację w skali kraju działań służących wykrywaniu i zapobieganiu niepożądanym sytuacjom w cyberprzestrzeni. Mowa tutaj o sieciach wykorzystujących protokoły TCP/IP, mobilnych sieciach bezprzewodowych, sieciach tworzących Internet Rzeczy (IoT) oraz sieciach automatyki przemysłowej.

Narodowe centra cyberbezpieczeństwa powstały w większości państw wysokorozwiniętych, np. w Wielkiej Brytanii, USA czy Holandii [1–3]. Ich zadaniem jest monitorowanie, gromadzenie i analiza danych o podatnościach, zagrożeniach i naruszeniach cyberbezpieczeństwa oraz zapobieganie cyberatakom i reagowanie na incydenty na poziomie państwa. Przykładem systemu monitorowania bezpieczeństwa sieci jest prototypowy system SEQUESTOR zainstalowany w Pacific Northwest National Laboratory (USA) [5]. System korzysta z wielu źródeł danych, w tym programów antywirusowych, narzędzi sieciowych, analizy zdarzeń systemowych i ruchu sieciowego. Integracja danych z tak różnorodnych źródeł stanowi prawdziwe wyzwanie projektowe. Szczególnie istotna jest czytelna prezentacja tych danych, a to się łączy bezpośrednio ze sterowaniem przez operatorów sys-

Autor korespondujący:
Cezary Zieliński, c.zielinski@ia.pw.edu.pl

Artykuł recenzowany
nadesłany 15.07.2019 r., przyjęty do druku 16.09.2019 r.

temu, jak dane mają być przedstawione. Kompleksowa analiza pracy sieci wymaga wykrywania korelacji zachodzących zdarzeń, analizy sytuacyjnej oraz dynamicznej i statycznej analizy ryzyka. Aby narzędzie do takiej analizy było użyteczne, potrzebne są odpowiednie metody i techniki do wizualizacji wielowymiarowych danych [29, 30].

W systemie SEQUESTOR wizualizacja danych została zorganizowana za pomocą SEQViz [5]. Dane są przedstawiane graficznie w postaci dwóch skoordynowanych widoków: ogólnego widoku sieci (w postaci grafu) oraz widoku modelu behawioralnego. Modele są wzorcami anomalii, czyli zachowań wskazujących na potencjalne zagrożenie lub atak. Aby wspomóc analityków sieci w wykrywaniu wzorców w danych dotyczących działania sieci i wykrywaniu anomalii opracowano panel BubbleNet do ich wizualizacji [21]. Panel ten przedstawia dane w różnych widokach: widoku lokalizacji w postaci mapy z przestrzenną informacją zakodowaną z użyciem tzw. pseudo-kartogramów Dorlinga [7], widoku czasowego w postaci wykresów słupkowych oraz widoku atrybutów przedstawiających graficznie różne atrybuty danych.

W pracy [12] przedstawiono wizualny system interaktywny do wykrywania anomalii w danych czasoprzestrzennych zebranych z różnych źródeł danych strumieniowych. Interaktywny interfejs graficzny użytkownika tego systemu składa się z ośmiu głównych widoków, w tym makro- i mikromapy, widoku czasowego wzorca oraz widoku inspekcji cech. Użytkownik za pomocą myszki może wybrać obszar na makromapie i wyświetlić na mikromapie listę anomalii wykrytych dla tego obszaru. Może również wykonać operacje powiększania i przesuwania oraz zmiany trybu wyświetlania na obu mapach.

System VisIDAC [32] umożliwia w czasie rzeczywistym wizualizacje 3D danych o zdarzeniach dotyczących cyberbezpieczeństwa, zebranych przez systemy wykrywania włamań zainstalowane w różnych sieciach. Dane o zdarzeniach są wyświetlane w postaci graficznej na trzech panelach: dla wejściowego, wyjściowego i docelowego ruchu sieciowego. W celu łatwego rozróżnienia rodzajów zdarzeń są stosowane różne kształty i kolory.

Opracowanie systemu do monitorowania sieci pod kątem jej cyberbezpieczeństwa pociąga za sobą konieczność stworzenia wielomodalnego interfejsu operatora umożliwiającego mu inteligentne sterowanie obrazowaniem zjawisk zachodzących w cyberprzestrzeni. Niniejszy artykuł poświęcony jest opisowi takiego interfejsu, który umożliwia sterowanie zobrazowaniem za pomocą głosu i gestów operatora. Gesty tworzone są przez operatora – ruchy sylwetki, rąk i głowy. Oprogramowanie do rozpoznawania mowy i weryfikacji mówcy zapewnia bezpieczny sposób zadawania komend głosowych przez uprzywilejowanego operatora systemu obrazowania.

Prace badawcze poświęcone wielomodalnym interfejsom człowiek-komputer są prowadzone od ponad 40 lat [15, 16, 23, 33]. Celem tych badań jest opracowanie metod i technik interakcji ludzi z komputerem w pełni wykorzystujących sposoby naturalnej komunikacji i interakcji człowieka z otoczeniem. Interfejsy wielomodalne charakteryzują się dwiema podstawowymi cechami: łączeniem wielu typów danych oraz przetwarzaniem tych danych w czasie rzeczywistym przy określonych ograniczeniach czasowych [15].

Sposób prezentacji operatorowi oraz analitykom zebranych danych o działaniu sieci w istotny sposób wpływa na efektywność ich pracy. Wielomodalny interfejs umożliwia im dostosowanie sposobu prezentacji do ich wymagań. Sterowanie prezentacją danych za pomocą komend głosowych i gestów umożliwia obsługę systemu przez osoby, które nie zostały specjalnie do tego przeszkolone, a co więcej, ułatwia prezentację stanu sieci osobom podejmującym decyzje, które nie muszą być zaznajomione ze sposobem działania interfejsu. Do codziennej obsługi systemu przez przeszkolony personel używane będą standardowe urządzenia wejściowe, takie jak klawiatury i myszki. Ponadto interfejs umożliwia rozpoznanie mówcy i automatyczną jego autoryzację w systemie.

Ten artykuł poświęcony jest sposobowi tworzenia interfejsu do Narodowej Platformy Cyberbezpieczeństwa (NPC), ale sposób ten może być wykorzystany także do budowy innych interfejsów. Ogólne metody tworzenia oprogramowania są określone przez inżynierię oprogramowania [28]. Niewątpliwie należy się do nich stosować, ale konkretne systemy mają swoją specyfikę, która również wpływa na architekturę powstającego oprogramowania. Interfejs stanowi styk między otoczeniem fizycznym, w którym znajduje się operator, a platforma NPC. Podobne cechy mają systemy robotyczne, które także operują w środowisku fizycznym, z jednej strony wpływając na nie, a z drugiej pozyskując z niego informacje. Podobnie interfejs oddziałuje na operatora, z jednej strony przekazując mu niezbędne dane, a z drugiej pozyskuje od niego informacje, jak konkretnie ma platforma zaprezentować zebrane dane. Dlatego postanowiono skorzystać z bogatego doświadczenia w konstrukcji systemów sterowania robotami i zastosować podejście wykorzystujące agenty upostaciowione [37–40] zarówno do specyfikacji jak i implementacji interfejsu. Prezentacja tej metody zastosowanej do budowy interfejsu jest głównym celem tego artykułu.

2 Wymagania i ogólna koncepcja rozwiązania

W procesie projektowania systemu pierwszym krokiem jest jego wyodrębnienie ze środowiska. Tutaj system stanowią platforma NPC wraz z przedstawianym tu interfejsem do niej. Założono, że kontakt operatora z systemem będzie najbardziej naturalny, jeżeli będzie się można z nim porozumiewać tak, jak ludzie porozumiewają się ze sobą, a więc za pomocą głosu i gestów. Dlatego interfejs NPC wyposażono w kamery i mikrofony. Ponadto operator w procesie powoływania systemu do życia musi go skonfigurować, a następnie nim administrować. W tym celu używa standardowych urządzeń wejściowych komputera, takich jak myszka, klawiatura czy ekran dotykowy (touchpad). Informacja zwrotna wytwarzana przez system przedstawiana jest na ekranie monitora, a konkretnie w okienkach pojawiających się na nim, a ponadto interakcja następuje za pomocą kursorów przemieszczanych po ekranie. Przyjęto, że operator działa w środowisku fizycznym rozszerzonym o ekran monitora. System porozumiewa się z operatorem za pomocą wymienionych urządzeń i okienek pojawiających się na ekranie. Z uwagi na fakt, że dotychczas operator kontaktował się z platformą NPC za pomocą urządzeń wejściowych, takich jak: ekrany dotykowe, myszki lub klawiatury, przyjęto iż interfejs będzie przekształcał komendy głosowe i te wydawane za pomocą gestów do formatu komend wprowadzanych z urządzeń wejściowych.

Gesty związane są z ruchem rąk, a więc obrazy uzyskiwane z kamer muszą być zbierane z dostatecznie dużą częstotliwością. Przyjęto, że przetwarzanie zarówno obrazów jak i mowy powinno być na tyle szybkie, by dla operatora czas między wydaniem komendy a inicjacją jej wykonywania był niezauważalny. W analizie mowy powstaje opóźnienie między zakończeniem wypowiadania komendy a wynikiem analizy. Przyjęto, że dopuszczalne opóźnienie wynosi 1 s, gdyż po zakończeniu wypowiedzi trzeba odczekać, aby zdecydować, czy pojawi się pauza czy wypowiedź będzie kontynuowana. Na to potrzeba około 200 ms, a do tego dochodzi jeszcze czas analizy polecenia. W przypadku przetwarzania obrazu użytkownik współpracuje dużo ściślej z systemem – sterując położeniem kursora oczekuje dużo szybszych reakcji i ciągłości jego przemieszczania. Z tego względu przyjęto, że

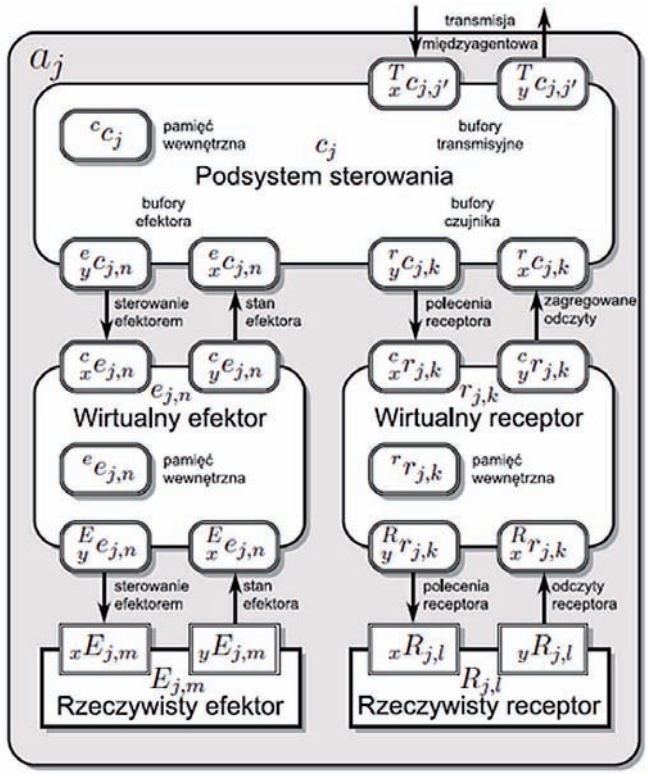
analiza obrazu powinna odbywać się z szybkością co najmniej 20 klatek na sekundę, natomiast opóźnienie reakcji nie powinno być większe niż 100 ms.

Struktura interfejsu operatora musi być dostosowana do wyżej sformułowanych wymagań. Ponieważ interfejs jest konfigurowany za pomocą standardowych urządzeń wejściowych komputera lub pobudzany do działania przez operatora wydającego komendy głosowe lub gestykulacje, a ponadto oddziałuje na operatora wyświetlając dane, można jego strukturę określić za pomocą agentów upostaciowionych [18, 36], które realizują swoje zadanie zbierając dane z otoczenia za pomocą receptorów, by przekształcić je na oddziaływania na to otoczenie za pomocą efektorów. Należy jedynie adekwatnie zdefiniować receptory i efektory interfejsu. Dotychczas agenty upostaciowione były wykorzystywane do tworzenia systemów sterowania systemami robotycznymi [37–40]. Niniejszy artykuł pokazuje ich użyteczność w projektowaniu systemów, które nie są związane z robotyką.

3. Agenty upostaciowione

W klasycznym podejściu agent definiowany jest jako twór działający autonomicznie, postrzegający swoje środowisko, mający charakter trwały, a ponadto mający zdolność do dostosowywania się do zmian w swoim środowisku, a co więcej mogący przejąć cel innego agenta [26]. Niemniej jednak tutaj przyjęto definicję bardziej zwięzłą, a jednocześnie ogólniejszą. Przyjęto, że agent jest tworem dążącym do realizacji zadania, postrzegającym swoje środowisko i wpływającym racjonalnie na nie. Zadanie stanowi wewnętrzny imperatyw agenta do działania. Jest ono realizowane w środowisku, zgodnie z aktualnie panującymi w nim warunkami, dzięki informacji uzyskiwanej z otoczenia. Tam gdzie środowisko ma charakter fizyczny, agent oddziałuje na nie przez swoje efektory i postrzega je za pomocą swoich receptorów. Agent działający w fizycznym otoczeniu musi mieć fizyczną powłokę (postać), by na nie oddziaływać. W związku z tym zwykło się go zwać agentem upostaciowionym [18, 36]. Agentowa struktura systemu będzie określona dalej. Tutaj wystarczy zauważyć, że działania operatora działającego w środowisku rejestrowane są za pomocą takich receptorów jak kamery czy mikrofony, natomiast system NPC oddziałuje na operatora, traktowanego jako element otoczenia, poprzez okienka oraz kursory pojawiające się na obrazie wyświetlanym na ekranie monitora, a więc zarówno okienka jak i kursory mogą być traktowane jako efektory. Wewnętrznym imperatywem systemu jest rozpoznawanie komend operatora i przekształcanie ich w odpowiednie zobrazowanie danych.

Wykorzystanie agentów do określenia struktury systemu ma tę zaletę, że można użyć formalnej notacji, opracowanej specjalnie do opisu zarówno struktur systemów, jak i sposobów ich działania. Główne elementy tej notacji zostaną opisane poniżej. **Agent** a_j, gdzie j jest jego oznaczeniem, składa się z: **efektorów rzeczywistych** E_j oddziałujących na środowisko (w przypadku interfejsu NPC są to okienka oraz kursory pojawiające się na ekranie monitora), rzeczywistych receptorów R_j (ściśle mówiąc chodzi o eksteroceptory, gdyż proprioceptory sprzężone są bezpośrednio z efektorami) zbierających dane o stanie otoczenia oraz **systemu sterowania** C_j realizującego zadanie przez wpływanie na efektory na podstawie odczytów uzyskiwanych z receptorów (w interfejsie NPC są nimi mikrofony i kamery). System sterowania C_j składa się z trzech typów podsystemów: **wirtualnych efektorów** e_j, **wirtualnych receptorów** r_j oraz **podsystemu sterowania** c_j. Agent może zawierać wiele receptorów rzeczywistych $R_{j,l}$ i wiele receptorów wirtualnych $r_{j,k}$, gdzie l i k są ich oznaczeniami. Podobnie w jego skład może wchodzić wiele efektorów rzeczywistych $E_{j,m}$ i wiele efektorów wirtualnych $e_{j,n}$, gdzie m i n są ich oznaczeniami. Wirtualne efektory $e_{j,n}$



Rys. 1. Ogólna struktura agenta upostaciowionego
Fig. 1. General structure of an embodied agent

i receptory $r_{j,k}$ prezentują podsystemowi sterowania c_j urządzenia wykonawcze i środowisko w postaci właściwej do określenia sposobu realizacji zadania – w tym przypadku są to polecenia dotyczące sposobu prezentacji wyników analizy, które ma wykonać platforma NPC. Wirtualne efektory dokonują transformacji złożonych poleceń na elementarne rozkazy sterujące sprzętem (w przypadku NPC są to okienka i kursory), natomiast wirtualne receptory przekształcają odczyty z czujników (w przypadku NPC są to zapisy dźwięku i obrazy) na abstrakcyjne pojęcia niezbędne do zwięzłego wyrażenia realizowanego zadania. Agenty mogą się również komunikować między sobą. Na rys. 1 przedstawiono wewnętrzną strukturę agenta upostaciowionego.

Podsystemy agenta a_j, a więc: podsystem sterowania c_j, wirtualne efektory $e_{j,n}$, wirtualne receptory $r_{j,k}$, rzeczywiste efektory $E_{j,m}$ i rzeczywiste receptory $R_{j,l}$ (rys. 1), kontaktują się ze sobą przez bufory komunikacyjne. Bufory oznaczane są przednim indeksem dolnym: x – wejściowe, y – wyjściowe, natomiast pamięć wewnętrzna podsystemu pozbawiona jest takiego indeksu. Górny lewy indeks przy oznaczeniu bufora określa typ podsystemu, z jakim ten bufor współpracuje.

Wszystkie podsystemy agenta działają według tego samego wzorca [37–40]. Każdy zawiera automat skończony FSM (ang. *Finite State Machine*), z którego stanami skojarzono zachowania. Każde zachowanie jest sparametryzowane funkcją przejścia oraz warunkami końcowymi i błędu. Zachowanie jest powtarzane do momentu spełnienia dowolnego z wymienionych warunków. Zakończenie wykonania zachowania powoduje przejście automatu skończonego do stanu następnego, który jest wybierany na podstawie warunków początkowych etykietujących łuki skierowane grafu automatu. Czynności związane z zachowaniem zależą od postaci funkcji przejścia. Funkcja ta pobiera dane z buforów wejściowych oraz pamięci wewnętrznej podsystemu i oblicza nowe wartości dla buforów wyjściowych oraz pamięci wewnętrznej. Zachowanie, oprócz obliczenia funkcji przejścia, zajmuje się wczytaniem nowych wartości do buforów wejściowych oraz rozesłaniem danych z buforów wyjściowych do innych podsystemów.

4. Zadania interfejsu i kategorie jego operatorów

Interfejs do systemu obrazowania informacji NPC odgrywa trzy role, w zależności od kategorii operatora, z którym wchodzi w interakcje. Umożliwia:
- użytkownikowi (np. kierownikowi NPC) dogodne sterowanie wizualizacją informacji tworzonej przez platformę NPC,
- serwisantowi konfigurację danej instalacji interfejsu,
- administratorowi systemu zarządzanie zasobami (danymi o gestach, osobach i komendach głosowych) oraz tworzenie modeli: gestów, komend głosowych i użytkowników.

Role te związane są z trzema różnymi aspektami (trybami) pracy interfejsu: sterowanie, konfigurowanie, administrowanie (tworzenie modeli i zarządzanie danymi).

Użytkownik za pomocą interfejsu zleca platformie wykonanie swoich poleceń, a więc przedstawienie danych w odpowiedni sposób. Serwisant zajmuje się konfigurowaniem sprzętu i parametryzacją oprogramowania do analizy obrazów i mowy. Konfiguracja sprzętowo-programowa każdego z modułów interfejsu polega na doborze jego parametrów wewnętrznych dla danej instalacji systemu. Administrator zarządza modelami gestów, poleceń i użytkowników, ale przede wszystkim dostarcza dane treningowe i uruchamia uczenie się systemu (tworzenie modeli). Odnosi się to zarówno do zarządzania informacją dotyczącą użytkowników systemu, jak i danych dotyczących poleceń, które system może zrealizować. W tym pierwszym przypadku do systemu można wprowadzać nowych użytkowników bądź usuwać tych wcześniej wprowadzonych oraz edytować dane użytkownika. W drugim przypadku można dodawać i usuwać polecenia oraz je modyfikować (edytować).

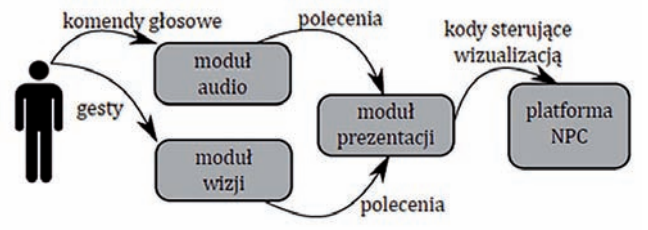
Uczenie w systemie zastosowano do zdefiniowania następujących przyporządkowań: gestów poleceniom, instrukcji wydawanych głosem komendom oraz rozpoznawania mówcy w celu określenia jego uprawnień. Przewidziano, że użytkownicy różnej rangi będą mieli różne uprawnienia do wydawania poleceń systemowi. Proces uczenia odbywa się w dwóch fazach. Wpierw zbierane są próbki o właściwej postaci, a następnie uczony jest odpowiedni klasyfikator. Nadzór nad procesem uczenia spoczywa na administratorze systemu.

Sterowanie polega na rozpoznawaniu poleceń użytkownika wydawanych głosem lub gestem. Polecenia te są przekształcane w bodźce zazwyczaj generowane przez myszkę (ruch kursora, kliknięcie), klawiaturę (wciśniecie klawisza) bądź ekran dotykowy (dotknięcia). Bodźce te oddziałują na okienka reprezentowane agentami lub na inne programy uruchomione na komputerze. Są one przez te agenty lub programy interpretowane, przykładowo jako: obrót obrazu, zbliżenie lub oddalenie widoku wyświetlanego w oknie (*zoom*). Agenty reprezentujące okna mogą być częściami modułów interfejsu lub platformy. Interfejs przekształca gesty oraz komendy wydawane głosem na polecenia dla platformy.

5. Modułowa struktura interfejsu

System NPC (rys. 2) składa się z platformy, która zbiera dane o aktywności w sieci, analizuje je i prezentuje wyniki, oraz z interfejsu, za pomocą którego użytkownik instruuje platformę, w jaki sposób te wyniki powinny zostać zaprezentowane. Aby móc sterować prezentacją wyników analizy, interfejs musi być odpowiednio skonfigurowany i zarządzany. Tutaj skoncentrowano się jedynie na sposobie konstrukcji interfejsu. Zawiera on trzy moduły: prezentacji, wizyjny i audio.

Każdy moduł skomponowany jest z agentów. Liczba modułów wynika ze sposobu interakcji operatora z interfejsem oraz



Rys. 2. Struktura systemu NPC
Fig. 2. Structure of the NCP system

sposobu interakcji interfejsu z platformą NPC. Użytkownik może wydawać polecenia albo za pomocą głosu albo za pomocą gestów, a więc istnieją dwa oddzielne źródła poleceń, a co za tym idzie dwa odrębne moduły je przetwarzające: audio i wizyjny. Niezależnie od źródła, wydawane polecenia muszą wpływać bezkonfliktowo na działanie platformy NPC. Dlatego pojedynczy dysponent platformy, agregujący polecenia otrzymywane z różnych źródeł, jest optymalny – jest nim moduł prezentacji. W konsekwencji powstała trójmodułowa struktura interfejsu. Moduł wizyjny służy do przetwarzania poleceń wydawanych gestami, natomiast moduł audio zajmuje się komendami wydawanymi głosem i identyfikacją mówcy. Moduł prezentacji jest odpowiedzialny za agregację rozkazów pochodzących z modułów przetwarzających polecenia użytkownika (tzn. modułów wizji i audio). Efektem działania tego modułu jest selekcja polecenia operatora, które ma być przekazane platformie do realizacji.

Opracowany interfejs umożliwia wydawanie poleceń za pomocą głosu i gestów, które muszą być przetłumaczone na postać akceptowaną przez platformę, czyli kody generowane przez tradycyjne urządzenia wejściowe (klawiatura, mysz, ekran dotykowy itp.). Odbywa się to dwuetapowo. Wpierw moduły wizyjny i audio zamieniają odpowiednio gesty bądź komendy głosowe na ich identyfikatory, a następnie moduł prezentacji zamienia te identyfikatory na kody poleceń wydawanych przez urządzenia wejściowe. Identyfikatory urządzeń w systemie operacyjnym komputera, w którym uruchomiono interfejs, mogą się zmieniać. Moduły audio i wizyjny odnoszą się do typów imitowanych urządzeń wejściowych, natomiast moduł prezentacji chcąc wykonać akcję musi się odwołać do identyfikatora konkretnego urządzenia obsługiwanego przez system operacyjny. Identyfikator ten jest zmienny i przez to musi być dostarczony modułowi prezentacji. Interfejs przekazuje prezentowanym oknom informacje o elementarnych zdarzeniach, np. dotyku i przesunięciu palca po ekranie. Informacja ma postać generowana przez wspomniane urządzenia wejściowe. Sterownik okna interpretuje te elementarne zdarzenia (a dokładniej sekwencje tych zdarzeń) powodując np. zoom lub obrót. Założono, że zbiór możliwych akcji użytkownika (gestów i komend głosowych) jest ustalony (w przypadku gestów) względnie ograniczony (w przypadku komend głosowych) przez twórcę systemu. Tablica asocjacji akcji, utworzona przez projektantów systemu, może być co najwyżej modyfikowana przez serwisanta systemu, aby ewentualnie lepiej dostosować konkretną instalację systemu do potrzeb danego użytkownika/administratora.

6. Role interfejsu

6.1. Kategorie operatorów interfejsu

Interfejs jest używany przez różne kategorie operatorów zgodnie z przypisanymi im rolami. Użytkownicy wykorzystują go do sterowania sposobem prezentacji danych generowanych przez platformę. Podstawową formą działania interfejsu jest przekształcanie poleceń użytkownika na pożądany sposób prezen-

tacji danych. Metoda sterowania okienkami została opisana przy okazji prezentacji poszczególnych modułów interfejsu.

Każdy z modułów ma interfejs administracyjno-serwisowy zawiadujący własnym oknem GUI (ang. Graphical User Interface). Moduł prezentacji wykorzystuje swoje okno do wyświetlania logów otrzymanych z wszystkich trzech modułów oraz do wysyłania instrukcji do każdego z modułów. W odpowiedzi na otrzymane z modułu prezentacji instrukcje, moduły wizji i audio: przechodzą w tryb wstrzymania, wznawiają działanie po wstrzymaniu lub wczytują nową konfigurację.

Administrator modułu audio ma możliwość tworzenia i zarządzania modelami komend głosowych i mówców. Do tego celu wykorzystuje interfejs administracyjno-serwisowy, który uaktywnia funkcje uczenia modułu audio. Komenda głosowa składa się z sekwencji 1–3 słów, wypowiadanych w sposób ciągły bez widocznych przerw między słowami. Pozwala to interpretować tę sekwencję jako pojedynczą izolowaną frazę zdaniową i rozpoznawać ją jako jedną z komend, dla których stworzono model akustyczny. Model akustyczny komendy ma postać sekwencji cech krótkookresowych ramek sygnału i jest oparty na cechach mel-cepstralnych [17] i pochodnych tych cech. Model głosowy mówcy ma postać mieszaniny rozkładów Gaussa w przestrzeni cech ramek. Szczegółowe omówienie tych modeli znajduje się w drugiej części tego artykułu w jego sekcji 2. Rola administratora modułu audio w procesie tworzenia modeli polega na zebraniu próbek głosowych odpowiednich komend i mówców, uruchomieniu procedur uczenia i na zdefiniowaniu tablicy **asocjacji treści komend i mówcy** z numerami klas udostępnianymi przez moduł audio. Administrator jest przy tym świadomy istnienia tablicy asocjacji komend i ograniczeń, co do maksymalnej liczby dostępnych komend i mówców. Jeśli z tablicy asocjacji komend wynika, że komenda nr 1 służy do wywołania „pochwycenia" elementu graficznego, to administrator może nazwać tę komendę jako „pochwyć" lub „drag" lub jeszcze inaczej, w sposób zrozumiały dla użytkownika. Wtedy ta klasa powinna być uczona na podstawie próbek głosowych zawierających treść „pochwyć" itp.

6.2. Role modułu wizji

Zarządzanie modelami przez administratora modułu wizyjnego polega na powiązaniu identyfikatorów gestów z ich klasą. Identyfikatory gestów przekazywane są do modułu prezentacji w trakcie użytkowania interfejsu. Zbiór gestów i ich forma są ustalone z góry, a system analizy gestów jest dostarczony w postaci już nauczonej. Administrator nie ma bezpośrednio możliwości definiowania i uczenia własnych gestów, tak jak to ma miejsce w przypadku modułu audio. Jednak może zlecić serwisantowi lub twórcy systemu wykonanie modyfikacji zbioru gestów. Uwzględniając powyższą możliwość, bezpośrednie zarządzanie modułem wizji przez administratora polega jedynie na ewentualnej modyfikacji tablicy asocjacji gestów, czyli odwzorowania: moja nazwa gestu ↔numer/klasa gestu w module wizji.

Moduł wizyjny, po wczytaniu plików konfiguracyjnych, przygotowany jest do współpracy z użytkownikiem. Aby sterowanie gestami było skuteczne, użytkownik musi być widoczny dla kamery. W tym celu moduł wizyjny wyświetla rysunek pokazujący, w jakim obszarze powinien stać użytkownik, by móc sterować systemem, np. kursorami. Kursory to odwzorowanie dłoni użytkownika na ekran.

Rolą serwisanta systemu wizyjnego jest, oprócz samej sprzętowej instalacji systemu, jego wstępna konfiguracja i kalibracja. Serwisant wprowadza (w plikach konfiguracyjnych) położenie i orientację kamery względem układu współrzędnych ustalonego dla punktu odniesienia (np. środka ekranu). Wykonuje też kalibrację stereo-pary kamer w celu uzyskania zestawu parametrów wewnętrznych (ang. intrinsic parameters) i zewnętrznych (ang. extrinsic parameters). Do jego zadań należy też wstępne ustawienie parametrów akwizycji tak, aby dopasować je do warunków oświetleniowych panujących w miejscu instalacji.

6.3. Role modułu audio

Moduł audio sprawuje kontrolę nad okienkami umożliwiającymi administratorowi przełączanie się między następującymi trybami pracy: trybem zarządzania zasobami (zarządzanie poleceniami, użytkownikami oraz nagraniami; zapisywanie plików konfiguracyjnych i nagrań mowy), uczeniem się modeli mówców i modeli komend (oraz zapisywaniem modeli do plików) i trybem sterującym, umożliwiającym rozpoczęcie rozpoznawania poleceń oraz identyfikacji użytkowników (wczytanie modeli mówców oraz komend, rozpoczęcie rozpoznawania komendy/mówcy). Po przejściu w tryb sterowania moduł audio rozpoczyna nasłuch sygnałów z mikrofonów oczekując na polecenia wydawane przez użytkownika. W trybie tym identyfikatory rozpoznanych poleceń oraz rozpoznanych użytkowników są przekazywane do modułu prezentacji.

Administracja modułem audio polega na: wprowadzaniu lub usuwaniu poleceń, wprowadzaniu lub usuwaniu użytkowników, nadzorze oraz zlecaniu rejestracji i przechowywania próbek głosu użytkowników, uaktualnianiu system plików konfiguracyjnych, a ponadto uruchamianiu dla danego użytkownika lub polecenia procesu uczenia. Podczas czynności administracyjnych użytkownik wykonuje polecenia administratora. Na jego żądanie użytkownik podaje swoje dane, umożliwiając w ten sposób swoją identyfikację, a następnie przechodzi do nagrywania próbek głosowych. Użytkownik wprowadzony do systemu może wydawać polecenia modułowi audio. Administrator modułu audio komunikuje się z nim za pomocą okienek GUI, wybierając za pomocą myszki pola okienek i wprowadzając informację tekstową do wybranych pól okienek.

6.4. Konfiguracja modułów

Serwisant oddzielnie konfiguruje każdy z modułów. Konfigurowanie modułu prezentacji polega na: ustaleniu rozdzielczości monitora (co wymusza zmianę wielkości emulowanego ekranu dotykowego), zmianie adresów modułów audio i wizji, zmianie parametrów emulowanego ekranu dotykowego oraz zmianie identyfikatorów przypisanych do bodźców inicjujących akcje systemu operacyjnego. Każdy identyfikator określa pewną akcję, która powinna zostać wykonana na oknie graficznym platformy NPC. Jest ona wynikiem polecenia otrzymanego od modułu prezentacji, pobudzonego przez moduł:

- audio, który rozpoznał komendę głosową, lub
- wizji, który rozpoznał gest.

Konfigurowanie modułu wizji sprowadza się do:

- wyboru obszaru, w którym użytkownik będzie mógł wykonywać gesty sterujące,
- kalibracji kamer,
- ustalenia związku między przestrzenią trójwymiarową a jej dwuwymiarowym zobrazowaniem,
- modyfikacji tablicy asocjacji akcji/bodźców (dodawanie lub usuwanie przyporządkowania rozpoznawanych gestów do identyfikatorów akcji/bodźców (i skrótów klawiszowych) generowanych przez imitowane urządzenie wejściowe.

Konfigurowanie modułu audio wymaga od serwisanta wykonania następujących czynności:

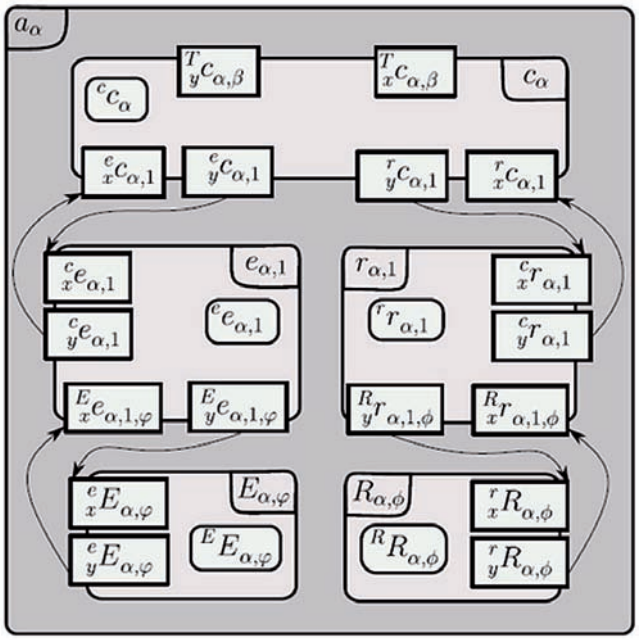
- instalacji sprzętu,
- ustawienia parametrów karty akwizycji – wybór kanałów akwizycji,
- ustawienia parametrów akwizycji sygnału audio, czyli: poziomu szumu, częstotliwości próbkowania, wielkości ramki sygnału, długości wektora cech itp.,
- modyfikacji tablicy asocjacji akcji/bodźców (dodania lub usunięcia przyporządkowania rozpoznawanych komend głosowych

do identyfikatorów akcji/bodźców (i skrótów klawiszowych) generowanych przez imitowaną myszkę.

Część z wymienionych czynności wykonywanych jest przez serwisanta bezpośrednio na sprzęcie wchodzącym w skład interfejsu.

7. Dekompozycja modułów na agenty

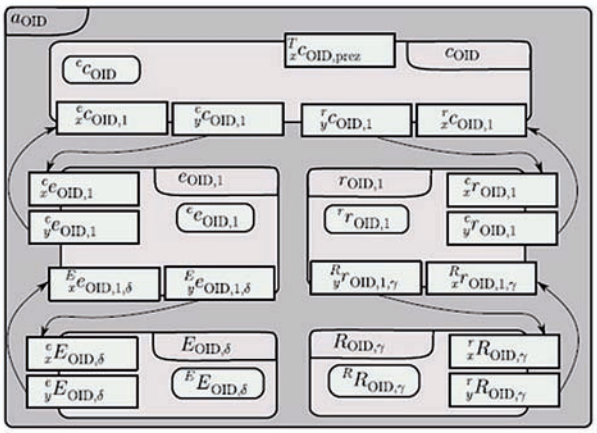
Podstawową jednostką kompozycji interfejsu jest moduł. Podział na moduły związany jest z trzema podstawowymi funkcjami, które realizuje interfejs w celu umożliwienia użytkownikowi łatwej interakcji z platformą NPC. Są nimi: przekształcanie gestów na polecenia dla platformy NPC, transformacja komend wydawanych głosem na polecenia dla tej platformy oraz integracją poleceń w celu spójnej prezentacji ich skutków na ekranie obserwowanym przez użytkownika. Każdy z tych modułów został skomponowany z agentów. Niektóre z tych agentów reprezentują okna ukazujące się na ekranie monitora traktowanego jako element środowiska.



Rys. 3. Ogólna struktura agentów reprezentujących okna uruchamiane na ekranie komputera; α – nazwa agenta reprezentującego okno, β – nazwa agenta, z którym ten agent się komunikuje

Fig. 3. General structure of agents representing windows appearing on the computer screen; α – name of the agent representing the windows, β – name of the agent with which this agent communicates

Ogólną strukturę agentów reprezentujących okna a_α, gdzie α to substytut nazwy agenta, określa rys. 3. Agentów tego typu w systemie jest wiele. Rzeczywistymi efektorami $E_{\alpha,\varphi}$ każdego takiego agenta są miejsca służące wyświetlaniu informacji w oknie (φ). Ta informacja pojawia się w środowisku i w ten sposób oddziałuje na operatora. Natomiast rzeczywistymi receptorami $R_{\alpha,\phi}$ tego agenta są obiekty wrażliwe na bodźce (ϕ) wytwarzane przez efektory agenta a_{OID} (ang. *Operator Interface Devices*). Agent a_α ma po jednym efektorze i receptorze wirtualnym, odpowiednio: $e_{\alpha,1}$ i $r_{\alpha,1}$. Współpracują one z wieloma efektorami i receptorami rzeczywistymi: $E_{\alpha,\varphi}$, $R_{\alpha,\phi}$. Liczba receptorów rzeczywistych zależy od liczby pól okna, z którymi użytkownik może wejść w interakcję (np. przycisk lub suwak). Natomiast liczba efektorów rzeczywistych zależy od liczby pól, w których może być wyświetlana informacja. Agenty okien a_α komunikują się bezpośrednio z innymi agentami a_β interfejsu. Istnieją następujące pary komunikujących



Rys. 4. Struktura agenta generującego bodźce; $\gamma \in$ {mysz, klawiatura, ekran dotykowy}, $\delta \in$ {mainPointer, secPointer, keyExecutor} ⇒ agent ten ma 3 efektory i 3 receptory

Fig. 4. Structure of the agent generating stimulus; $\gamma \in$ [mouse, keyboard, touchscreen}; $\delta \in$ {mainPointer, secPointer, keyExecutor} ⇒ this agent has 3 effectors and 3 receptors

się agentów: (α = prezGUI, β = prez), (α = audioGUI_nr, β = audio) (gdzie nr ∈ {init, user, command, control}), (α = visionGUI, β = vision).

Agent generujący bodźce a_{OID} (rys. 4) wyposażony jest w receptory rzeczywiste $R_{\mathrm{OID},1,\gamma}$, takie jak mysz, klawiatura lub ekran dotykowy – tu zbiorczo oznaczone jako γ. Pojedynczy wirtualny receptor $r_{\mathrm{OID},1}$ agreguje informacje uzyskane z tych urządzeń wejściowych. Efektorami rzeczywistymi tego agenta są kursory, a dokładniej rzecz ujmując, generatory bodźców zazwyczaj inicjowanych wspomnianymi urządzeniami wejścia. Agent a_{OID} ma trzy efektory rzeczywiste $E_{\mathrm{OID},\delta}$. Dwa efektory tworzą na ekranie monitora kursory: pierwotny (δ= mainPointer) i wtórny (δ = secPointer), a jeden efektor (δ = keyExecutor) imituje klawiaturę generując bodźce, takie jak rzeczywista klawiatura. Informacja do tych trzech efektorów rzeczywistych dostarczana jest przez efektor wirtualny $e_{\mathrm{OID},1}$.

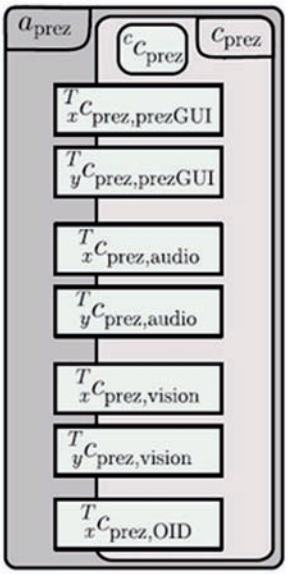
Agent a_{OID} oraz agenty a_α (reprezentujące okna) wchodzą w interakcje za pośrednictwem stygmergii [6], a więc przez zastosowanie komunikacji poprzez środowisko, którego częścią jest ekran monitora. Agenty a_α odczytują bodźce generowane przez a_{OID} za pomocą swoich rzeczywistych receptorów $R\alpha_{\alpha,1,\phi}$. W tym przypadku receptorami rzeczywistymi $R\alpha_{\alpha,1,\phi}$ są elementy okna (ϕ) czułe na bodźce zewnętrzne. Przykładowa interakcja między kursorem a elementami okna realizowana jest w następujący sposób. Agent a_{OID} wykorzystując jeden ze swoich rzeczywistych efektorów $E_{\mathrm{OID},\delta}$ (δ – element aktywny wytwarzający bodziec) oddziałuje na rzeczywisty receptor okna $R_{\alpha,1,\phi}$ (ϕ – element pasywny odbierający bodziec, np. przyciski lub suwak). Następnie agent a_α przekazuje uzyskaną informację do kolejnego agenta, który zrealizuje akcję wywoływaną tym bodźcem.

7.1. Moduł prezentacji

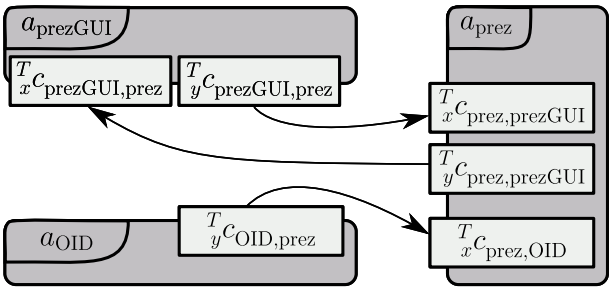
Moduł prezentacji utworzony jest z trzech agentów: a_{prez} (rys. 5), a_{prezGUI} (agent typu a_α – rys. 3) i a_{OID} (rys. 4). Komunikację między tymi agentami w module prezentacji przedstawiono na rys. 6.

Podsystem sterowania agenta a_{prez} ma jedynie bufory komunikacji międzyagentowej i pamięć wewnętrzną. Bufory komunikacji międzyagentowej umożliwiają interakcję nie tylko z dwoma pozostałymi agentami modułu prezentacji, ale również z agentami a_{audio} i a_{vision}. Agent a_{prez} wykonuje następujące zadania:

— tłumaczenie gestów otrzymanych od agenta a_{vision} na odpowiednie imitacje akcji urządzeń wejścia (myszy, klawiatury, ekranu dotykowego) i przesyłanie tych zleceń do agenta a_{OID}, aby ten je zrealizował,



Rys. 5. Struktura agenta prezentacji
Fig. 5. Structure of the agent presentation



Rys. 6. Komunikacja międzyagentowa w module prezentacji
Fig. 6. Interagent communication in the presentation module

– tłumaczenie identyfikatorów poleceń otrzymanych od agenta a_{audio} na odpowiednie skróty klawiaturowe i przesyłanie ich do agenta a_{OID}, aby ten je zrealizował,
– przesyłanie poleceń zmiany trybu pracy do agentów a_{vision} i a_{audio}. Polecenia te podsystem c_{prez} otrzymuje od agenta a_{prezGUI},
– przesyłanie logów do agenta a_{prezGUI}, aby ten je wyświetlił. Logi te pochodzą od samego podsystemu c_{prez} oraz od agentów a_{vision} i a_{audio}.

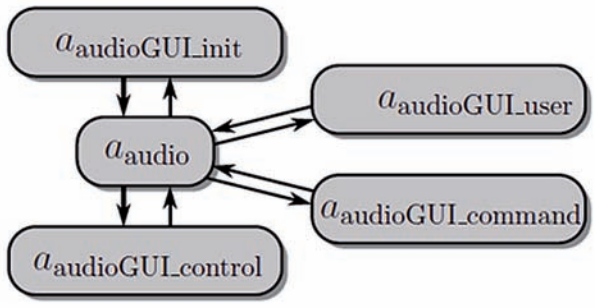
Zadaniem agenta a_{OID} jest wykonywanie odpowiednich operacji za pomocą swych efektorów rzeczywistych $E_{\text{OID},\delta}$ (kursorów), w celu oddziaływania na inne agenty reprezentujące okna, poprzez stygmergię. Operacje te wyzwalane są albo poleceniami agenta a_{prez} albo bodźcami generowanymi przez rzeczywiste receptory $R_{\text{OID},\gamma}$, gdzie $\gamma \in$ {mysz, klawiatura, ekran dotykowy}.

Agent a_{prezGUI} jest odpowiedzialny za wyświetlanie informacji otrzymanych od agenta a_{prez} oraz odczytywanie bodźców pochodzących ze środowiska, w tym przypadku ekranu komputera, a konkretnie kliknięć myszy lub naciśnięć klawiatury imitowanych przez agenta a_{OID}. Bodźce te są przechwytywane za pomocą przycisków okna – rzeczywistych receptorów agenta a_{prezGUI}.

7.2. Moduł audio

7.2.1. Struktura

Moduł audio składa się z pięciu agentów (rys. 7): podstawowego agenta a_{audio} oraz agentów pomocniczych zarządzających oknami:
– agenta $a_{\text{audioGUI\_init}}$ reprezentującego pojedyncze okno, służące do wyboru poniżej wymienionych okien,
– agenta $a_{\text{audioGUI\_user}}$ reprezentującego okno zarządzania użytkownikami,
– agenta $a_{\text{audioGUI\_command}}$ reprezentującego okno zarządzania poleceniami oraz

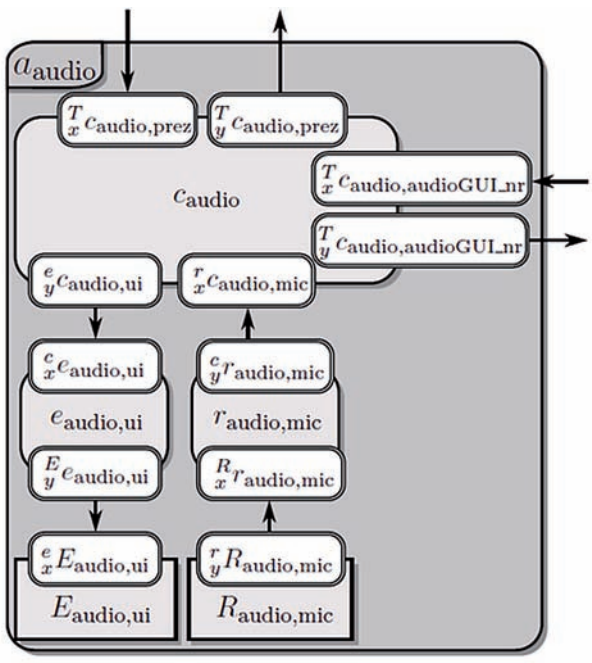


Rys. 7. Dekompozycja modułu audio na agenty
Fig. 7. Decomposition of the audio module into agents

– agenta $a_{\text{audioGUI\_control}}$ reprezentującego okno sterowania i przeprowadzania uczenia na podstawie posiadanych nagrań.

Cztery agenty reprezentujące okna mają strukturę agenta a_{α}, gdzie $\alpha \in$ {audioGUI_init, audioGUI_user, audioGUI_command, audioGUI_control}, a więc ich wewnętrzna struktura odpowiada tej z rys. 3. Każdy z czterech wspomnianych agentów GUI reprezentuje pojedyncze okno wyposażone w przyciski oraz elementy służące do wyświetlania odpowiednich informacji związanych z konkretnym agentem GUI modułu audio. Tylko jeden agent GUI modułu audio sposród: $a_{\text{audioGUI\_user}}$, $a_{\text{audioGUI\_command}}$ oraz $a_{\text{audioGUI\_control}}$ jest aktywny w danym momencie.

Struktura wewnętrzna agenta a_{audio} została przedstawiona na rys. 8. Agent a_{audio} składa się z podsystemu sterowania c_{audio} pozyskującego informacje z receptora wirtualnego $r_{\text{audio,mic}}$ i oddziałującego na efektor wirtualny $e_{\text{audio,ui}}$. Podsystem c_{audio} wyposażony jest w bufory transmisyjne umożliwiające komunikację z agentem a_{prez} oraz wspomnianymi agentami GUI modułu audio. Receptor wirtualny $r_{\text{audio,mic}}$ agreguje dane z receptora rzeczywistego $R_{\text{audio,mic}}$, który zawiera mikrofony, sprzętowy przetwornik analogowo-cyfrowy oraz sterownik programowy tego przetwornika zainstalowany w systemie operacyjnym. Natomiast efektor wirtualny $e_{\text{audio,ui}}$ steruje efektorem rzeczywistym $E_{\text{audio,ui}}$, który reprezentuje głośnik komputera, bądź system nagłośnienia pomieszczenia.



Rys. 8. Struktura agenta audio aaudio, gdzie nr ∈ {init, user, command, control}
Fig. 8. Structure of the audio agent, where nr ∈ {init, user, command, control}

Moduł audio, a dokładniej podsystem sterowania c_{audio} agenta a_{audio}, umożliwia:
— administratorowi systemu – tworzenie i zarządzanie modelami komend oraz użytkowników,
— użytkownikowi – sterowanie głosem platformą.

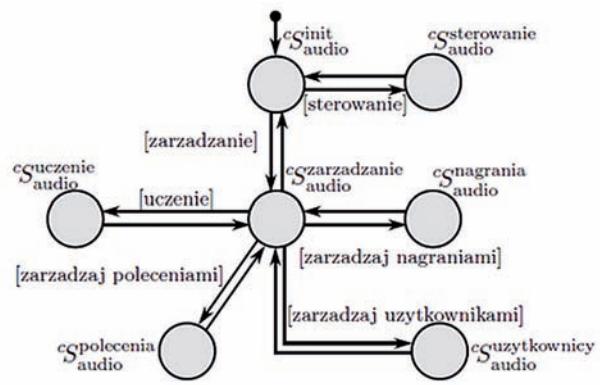
Administrator wpływa na moduł audio komunikując się z agentem a_{OID} za pomocą jego receptorów. Agent ten wykorzystując swoje efektory komunikuje się z odpowiednim agentem GUI stosując stygmergię. W dalszej kolejności podsystem sterowania wspomnianego agenta GUI przesyła odpowiednie polecenia do podsystemu c_{audio} agenta a_{audio}. Konfiguracja modułu audio polega na odpowiednim ustawieniu sprzętu oraz stworzeniu odpowiedniego pliku konfiguracyjnego, co wykonywane jest przez serwisanta bez udziału agenta a_{audio}.

7.2.2. Zachowania

Działanie agentów GUI a_{α}, tj. $a_{audioGUI\_init}$, $a_{audioGUI\_user}$, $a_{audioGUI\_command}$, $a_{audioGUI\_control}$ polega na odbieraniu za pomocą swoich rzeczywistych receptorów informacji ze środowiska (ekranu monitora), np. o wciśniętym przycisku w jednym z odpowiadających im okien, wysyłaniu powyższych informacji do agenta a_{audio}, odbieraniu danych z a_{audio} oraz wyświetlaniu ich za pomocą rzeczywistych efektorów (odpowiednich elementów okien).

Działanie agenta a_{audio}, a dokładniej jego podsystemu sterowania c_{audio}, opisuje automat skończony przedstawiony na rys. 9. Z każdym ze stanów automatu związane jest odpowiednie zachowanie. W stanie początkowym ${}^{c}S^{init}_{audio}$ podsystem c_{audio} oczekuje na polecenie otrzymane z $c_{audioGUI\_init}$. Polecenie to może dotyczyć albo przetwarzania komend głosowych albo zarządzania zasobami. Otrzymywane polecenia, powodując zmianę stanu automatu skończonego, powodują tworzenie agentów GUI lub zakończenie ich istnienia – w ten sposób okienka pojawiają się i znikają z ekranu. W konsekwencji prezentowany system ma zmienną strukturę [39], w której agenty pojawiają się i znikają.

Polecenie rozpoczęcia rozpoznawania komend, otrzymanie z $c_{audioGUI\_init}$ powoduje, że podsystem c_{audio} przechodzi do stanu ${}^{c}S^{sterowanie}_{audio}$, w którym realizuje zachowanie polegające na stworzeniu agenta $a_{audioGUI\_command}$ (reprezentującego okno służące do sterowania rozpoznawaniem), a następnie dokonuje ciągłej akwizycji sygnału audio i na bieżąco wykonuje detekcję mowy w sygnale oraz analizuje każdy izolowany fragment mowy w celu rozpoznania mówcy i komendy. W zależności od tego, czy rozpoznano mówcę czy też nie, stosowany jest dedykowany model komend takiego mówcy lub model ogólny. Zakończenie zachowania związanego ze stanem ${}^{c}S^{sterowanie}_{audio}$ powoduje unicestwienie agenta $a_{audioGUI\_command}$, a w związku z tym i okienka, którym zawiaduje.



Rys. 9. Automat skończony przełączający zachowania podsystemu sterowania c_{audio}
Fig. 9. Finite state automation switching behavioours of the control subsystem c_{audio}

W stanie ${}^{c}S^{zarzadzanie}_{audio}$ podejmowana jest decyzja, jakimi danymi (zasobami) ma zarządzać c_{audio}. Automat robi to na podstawie danych otrzymanych z $c_{audioGUI\_init}$. Po przejściu do stanu ${}^{c}S^{polecenia}_{audio}$ albo ${}^{c}S^{uzytkownicy}_{audio}$ uruchamiane jest zachowanie inicjujące odpowiedniego agenta, tj. $a_{audioGUI\_command}$ lub $a_{audioGUI\_user}$ i możliwe jest dodawanie, usuwanie i aktualizowanie danych zawartych w pliku konfiguracyjnym odpowiednio poleceń i użytkowników. Po zakończeniu działania zachowań następuje zakończenie działania odpowiedniego agenta GUI.

W stanie ${}^{c}S^{nagrania}_{audio}$ uruchamiany jest agent $a_{audioGUI\_control}$, dzięki czemu możliwe jest przeprowadzenie sesji nagraniowej dla wybranego użytkownika. Sesja nagraniowa polega na nagraniu po n próbek (powtórzeń) dla wszystkich poleceń wczytanych z pliku konfiguracyjnego, gdzie n jest liczbą próbek definiowaną przez administratora modułu audio. Podczas sesji nagraniowej c_{audio} komunikuje się z wirtualnym receptorem $r_{audio,mic}$, w celu akwizycji sygnału audio, oraz z podsystemem sterowania agenta $a_{audioGUI\_control}$, w celu kontrolnej wizualizacji oscylogramu nagrania, a ponadto z $e_{audio,ui}$ w celu odtworzenia przez głośniki rozpoznanego polecenia. Na koniec tworzony jest plik w formacie *wave* z nagraną próbką, będący elementem wewnętrznej pamięci podsystemu sterowania c_{audio}.

W stanie ${}^{c}S^{uczenie}_{audio}$ uruchamiany jest agent $a_{audioGUI\_control}$ i inicjowane jest uczenie, tj. automatyczne tworzenie modeli użytkowników i poleceń w oparciu o zebrane wcześniej próbki głosowe (podczas sesji nagraniowych z udziałem przyszłych użytkowników). Proces uczenia wykonywany jest prawie wyłącznie przez wirtualny receptor $r_{audio,mic}$, który realizuje funkcje:
— dekodowania danych w formacie *wave*,
— analizy wstępnej i parametryzacji sygnału mowy,
— tworzenia modeli mówców i modeli komend,
— serializacji modeli następnie zapisywanych do plików.

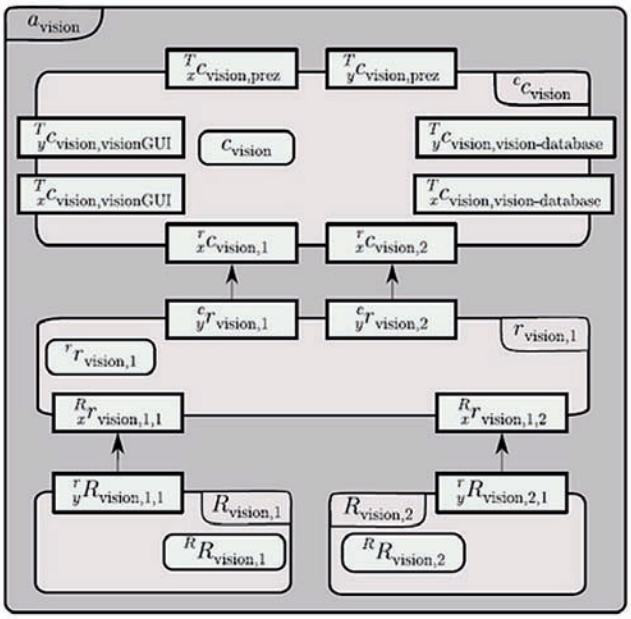
Zakodowane w formacie *wave* dane są przesyłane do podsystemu sterowania c_{audio}, a ten wypisuje je do rezydującego w jego pamięci wewnętrznej systemu plików. W razie potrzeby podsystem sterowania c_{audio} może pobrać ze swojej pamięci wewnętrznej uprzednio przygotowany plik, przesłać jego treść do wirtualnego receptora $r_{audio,mic}$, który zdekoduje dane w formacie *wave* do postaci wektora próbek sygnału (liczb) i w tej postaci zapamięta je na czas przetwarzania.

7.3. Moduł wizyjny

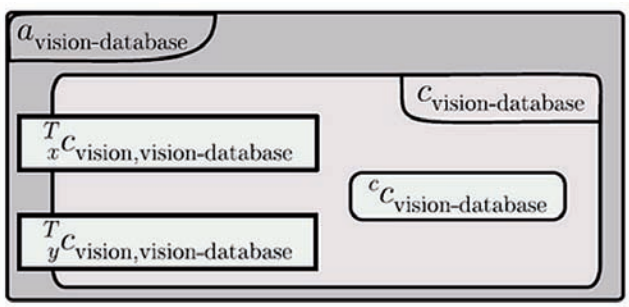
Moduł wizyjny składa się z trzech agentów. Głównym jest agent a_{vision} (rys. 10), który odpowiada za realizację zadania sterowania gestami. Komunikuje się on z agentami $a_{visionGUI}$ oraz $a_{vision-database}$ (rys. 11), które są odpowiedzialne za realizację odpowiednio graficznego interfejsu użytkownika oraz bazy danych.

Agent a_{vision} składa się z podsystemu sterowania c_{vision}, wirtualnego receptora r_{vision} oraz dwóch rzeczywistych receptorów $R_{vision,1}$ i $R_{vision,2}$ będących kamerami RGB zestawionymi w stereoparę. Pozostałe agenty mają jedynie podsystemy sterowania. Agent a_{vision} komunikuje się ponadto z agentem a_{prez}.

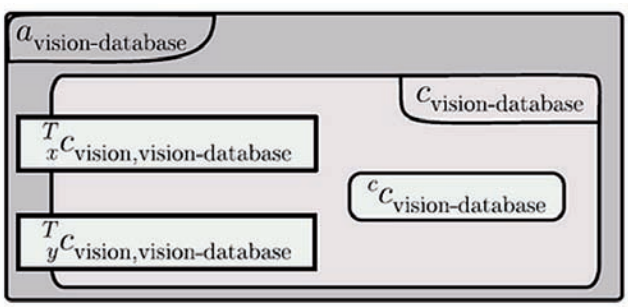
Automat skończony podsystemu sterowania c_{vision} agenta wizyjnego a_{vision} (rys. 12) w swym stanie początkowym ${}^{c}S^{init}_{vision}$ uruchamia zachowanie, które komunikuje się z agentem $a_{vision-database}$ w celu pobrania modeli gestów statycznych i dynamicznych oraz pozostałych danych potrzebnych do pracy agenta (m.in. zapisanych parametrów kamer), a następnie przechodzi do rozpoznawania gestów. Najpierw w stanie ${}^{c}S^{poszukiwanie\_twarzy}_{vision}$ uruchamiana jest procedura poszukiwania twarzy operatora. Jeśli zostanie ona odnaleziona w oczekiwanym miejscu przestrzeni roboczej, to automat przechodzi do stanu ${}^{c}S^{poszukiwanie\_dloni}_{vision}$, w którym ustalane jest położenie obu dłoni operatora. Po ich poprawnym zlokalizowaniu automat przechodzi do stanu ${}^{c}S^{rozpoznawanie\_gestow}_{vision}$, w którym rozpoznaje i śledzi dłonie oraz interpretuje gesty statyczne i dynamiczne. W przypadku utraty możliwości śledzenia



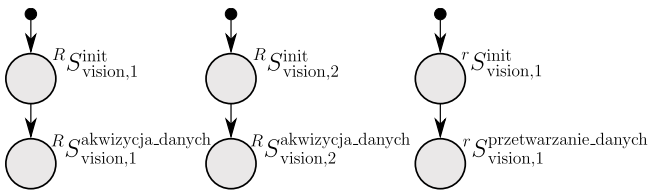
Rys. 10. Struktura agenta wizyjnego
Fig. 10. Structure of the agent vision



Rys. 11. Struktura agenta bazy danych modułu audio
Fig. 11. Structure of the agent audio data base



Rys. 12. Automat skończony przełączający zachowania podsystemu sterowania c_{vision}
Fig. 12. Finite state automaton switching the behaviours of the control subsystem c_{vision}



Rys. 13. Automaty skończone przełączające zachowania rzeczywistych receptorów $R_{\text{vision},1}$ i $R_{\text{vision},2}$ oraz wirtualnego receptora $r_{\text{vision},1}$
Fig. 13. Finite state automaton switching the behaviours of real receptor $R_{\text{vision},1}$ i $R_{\text{vision},2}$ and virtual receptor $r_{\text{vision},1}$ of the agent a_{vision}

dłoni bądź twarzy operatora automat wraca do stanu – odpowiednio $^{c}S^{\text{poszukiwanie\_dloni}}_{\text{vision}}$ lub $^{c}S^{\text{poszukiwanie\_twarzy}}_{\text{vision}}$. Trywialne automaty skończone wirtualnego receptora $r_{\text{vision},1}$ i rzeczywistych receptorów $R_{\text{vision},1}$ i $R_{\text{vision},2}$ agenta a_{vision} przedstawiono na rys. 13.

Podsystem sterowania agenta $a_{\text{visionGUI}}$, odpowiedzialnego za wyświetlanie okna interfejsu użytkownika modułu wizyjnego, realizuje tylko jedno zachowanie. Polega ono na wyświetlaniu w swoim oknie obrazów otrzymywanych od agenta a_{vision} oraz przesyłaniu do tego agenta poleceń będących efektem interakcji z agentami wytwarzającymi kursory na ekranie albo dostarczającymi informacji o wciśnięciu klawiszy klawiatury bądź innym zdarzeniu wywoływanym przez urządzenie wejściowe.

8. Charakterystyka agentów upostaciowionych interfejsu NPC

Agenty upostaciowione, z których skomponowano interfejs NPC, odpowiadają ogólnej definicji agenta przedstawionej w części 3. tego artykułu. Literatura dotycząca agentów, zarówno programowych, jak i stosowanych w robotyce, jest bogata i, jak już wspomniano, nie doczekała się jednoznacznej definicji tego pojęcia. Definicje tam przedstawione zazwyczaj odnoszą się do licznych cech różnorodnych typów agentów. Warto tutaj spojrzeć na agenta upostaciowionego pod kątem często wymienianych atrybutów różnych typów agentów.

Pojęcie agenta upostaciowionego zostało spopularyzowane przez Rodneya Brooksa (np. [9]) w trakcie debaty dotyczącej sztucznej inteligencji. Tradycyjna sztuczna inteligencja koncentrowała się na modelu środowiska (jego reprezentacji), by móc wyciągać wnioski dotyczące przyszłych akcji do wykonania przez system. Planowanie akcji było głównym przedmiotem zainteresowania badaczy. Brooks zakwestionował to podejście stwierdzając, że najlepszym modelem środowiska jest ono samo, a inteligentne działanie jest wynikiem ewolucyjnego procesu prowadzącego do doskonalenia sposobu działania systemu [8, 10, 11]. W rozumieniu Brooksa, dwoma podstawowymi cechami agenta jest posiadanie ciała/postaci (ang. *embodiment*) oraz usytuowanie (ang. *situatedness*), czyli egzystowanie w realnym otoczeniu, a nie w świecie abstrakcyjnym (uproszczonym). Wymienia on również inteligencję będącą następstwem wyłaniania (ang. *emergence*) jako efektu interakcji między agentem i jego otoczeniem. Spór między Brooksem i przedstawicielami klasycznej sztucznej inteligencji, dotyczący wyższości agentów reaktywnych nad deliberatywnymi, nie został jednoznacznie rozstrzygnięty, bo do inteligentnego działania potrzebne jest, w istocie, zarówno wnioskowanie prowadzone w oparciu o model świata, jak i zachowania reaktywne wykorzystujące uogólnianie (ang. s*ubsumption*) – powstały więc agenty hybrydowe (np. [4]). Co najmniej niektóre agenty wykorzystane do stworzenia interfejsu NPC są upostaciowione, gdyż posiadają ciało, a więc mikrofony i ekran monitora, służące do kontaktu z operatorem znajdującym się w środowisku. Ponadto są usytuowane, ponieważ przetwarzają sygnały dźwiękowe i świetlne bezpośrednio pochodzące z tego otoczenia. Sygnałami tymi są mowa oraz gesty operatora. Mamy tu jednak do czynienia z agentami hybrydowymi, bo sygnały te nie oddziałują w sposób bezpośredni na efektory, a więc nie jest to system czysto reaktywny. Wpierw te sygnały przetwarzane są na postać symboliczną komend, a więc wykorzystują modele wiążące sygnały z ich znaczeniem.

Spór dotyczący emergencji inteligencji w systemach doprowadził do rozpowszechnienia pojęcia agenta w środowisku naukowców zajmujących się sztuczna inteligencją. Klasyczny podręcznik tej dziedziny [26] zbudowany jest właśnie wokół pojęcia agenta. W najbardziej ogólnej postaci traktuje agenta jako coś, co działa. Taka definicja jest zbyt ogólna, więc dodatkowo wyróżnia pożądane cechy agentów: autonomiczne sterowanie,

czyli działanie bez udziału człowieka, zdolność do postrzegania środowiska, długotrwałość działania, adaptacja do zmian zachodzących w środowisku, zdolność do przejmowania celu działania innych agentów. Dodatkowo wymieniane są agenty racjonalne, a więc takie, których działanie przynosi najlepszy skutek lub najlepszy spodziewany skutek (jeżeli mamy do czynienia z niepewnością). Pojawiają się też agenty uczące się czyli takie, które polepszają wyniki swojego działania w miarę zdobywania nowych doświadczeń. Agenty interfejsu NPC, po dostrojeniu modeli przez uczenie, działają autonomicznie, dysponują zdolnością do postrzegania środowiska (zdolność do interakcji z operatorem), niektóre istnieją tak długo, jak długo istnieje interfejs, a agenty pomocnicze zarządzające oknami działają przez czas, kiedy są niezbędne do realizacji konkretnej operacji. Nie mają one zdolności adaptacyjnych i nie mogą przejmować celu działania innych agentów.

Różnorodność cech przypisywanych agentom wynika z ewolucji, która doprowadziła do pojawienia się tego pojęcia na gruncie informatyki. W podręczniku [25] autor wywodzi agenta z pojęcia aktora, które stanowi elementarne obliczenie wykonywane równolegle z innymi takimi obliczeniami. Aktory zmieniają swój stan wewnętrzny oraz komunikują się z innymi aktorami. Uogólnieniem aktora stał się aktywny obiekt, działający w swym własnym wątku. Uogólnienie aktywnego aktora doprowadziło do pojęcia agenta, który stanowi autonomiczną jednostkę działającą w środowisku. Jednostka ta dąży do realizacji swojego celu, co wymaga wykonywania pewnych czynności oraz komunikacji z innymi agentami. Efektem tych rozważań historycznych jest określenie agenta w bardzo podobny sposób, jak zdefiniowano agenta upostaciowionego.

Podjęto też próbę mariażu agentów działających w cyberprzestrzeni z tymi funkcjonującymi w przestrzeni fizycznej [13]. Rozwój systemów operacyjnych doprowadził do powstania pojęcia procesu. Początkowo procesy komunikowały się ze sobą jedynie wewnątrz systemu operacyjnego, ale wraz z rozwojem sieci komputerowych ta komunikacja musiała zostać rozszerzona, co doprowadziło do powstania koncepcji cyberprzestrzeni. Wspomniana praca pokazuje możliwości i konsekwencje przekroczenia granicy cyberprzestrzeni w celu wniknięcia do przestrzeni rzeczywistej, w której działają urządzenia, a więc i roboty. Ponieważ proces funkcjonuje w obrębie pojedynczego systemu operacyjnego, to uogólnienie tego pojęcia na cyberprzestrzeń dla odróżnienia zostało nazwane agentem. Cyberprzestrzeń stanowi środowisko, w którym agent działa, a więc postrzega ją i wpływa na nią. Dalsza ewolucja doprowadziła do powstania agentów mobilnych, które są zdolne do przemieszczania się w cyberprzestrzeni. Ponieważ roboty przemieszczają się w przestrzeni rzeczywistej, dostrzeżono tu, z jednej strony paralelę, a z drugiej strony pole ciekawych badań na styku tych przestrzeni. Podstawą tych rozważań był rozwój pojęć – od algorytmu przez obiekt do agenta. Motywacją była konieczność dekompozycji algorytmu, w celu umysłowego zrozumienia złożoności tworzonego systemu przez projektanta. Wspomniana ewolucja zwiększała stopień autonomii działania kolejno wymienionych pojęć. W pracy [13] zdefiniowano M-agenta, który na podstawie danych pozyskanych ze środowiska oraz wewnętrznego modelu służącego symulacji możliwych zachowań, wybiera optymalną strategię działania i te realizuje wpływając na środowisko. Specyfikacja M-agenta za pomocą opisanego tu agenta upostaciowionego została przedstawiona w [35]. Jednak do sterowania platformą za pomocą gestów i mowy nie jest potrzebny wyrafinowany model otoczenia – wystarczą wzorce gestów i komend.

Agentom programowym (software'owym) poświęcona jest praca [22], w której przytoczono różne kryteria, według których można sklasyfikować agenty. Wyróżnia się agenty statyczne i mobilne, w zależności od tego czy mogą się one przemieszczać po sieci. Agenty interfejsu NPC są statyczne. Natomiast praca [22] odnosi się do omawianych wyżej agentów reaktywnych, deliberatywnych i hybrydowych oraz autonomiczności agentów. Podnosi kwestie uczenia się i zdolności do współpracy między agentami. Agenty interfejsu NPC uczą się jedynie w trakcie ich konfiguracji, natomiast współdziałają ze sobą, by osiągnąć wspólny cel. Wspomniana praca sugeruje wykorzystanie jednego wspólnego języka do porozumiewania się agentów. Tu taki język nie został stworzony. Poszczególne agenty porozumiewają się ze sobą dzięki oddzielnym zestawom przesyłanych wiadomości. Wreszcie wymieniana jest oddzielna klasa agentów interfejsowych służących do porozumiewania się użytkowników z systemami. Interfejs NPC spełnia, jako całość, dokładnie tę rolę. Należy też zauważyć, że część agentów interfejsu NPC to agenty upostaciowione, a część to agenty czysto programowe.

Tekst [24] bazuje na definicji agenta zaproponowanej przez Wooldridge'a [34], mówiącej że agent stanowi system komputerowy, który jest usytuowany w pewnym środowisku i jest zdolny do autonomicznych działań w tym środowisku w celu realizacji zadań, dla których został zaprojektowany. Praca [24] wymienia pożądane cechy inteligentnego agenta: reaktywność, czyli zdolność do reagowania na zmiany zachodzące w środowisku; proaktywność, czyli dążenie do realizacji celu; socjalność, czyli zdolność do wchodzenia w interakcje z innymi agentami; elastyczność, czyli zdolność do osiągania celów różnymi sposobami oraz odporność, czyli umiejętność radzenia sobie z błędami. Agenty wchodzące w skład interfejsu NPC do pewnego stopnia mają wszystkie te cechy. Z kolei praca [31] traktuje o programowaniu zorientowanym agentowo, jako uogólnieniu programowania obiektowego, określa agenta przez takie właściwości jego „umysłu", jak: przekonania, zdolności, wybory oraz zobowiązania. To podejście wywodzi się z prac dotyczących architektury BDI (ang. *belief, desire, intension*) [24, 31]. Przekonania dotyczą aktualnego stanu środowiska i samego agenta, pragnienia wynikają z celu, który ma być osiągnięty, natomiast intencje dotyczą pragnień, do których osiągnięcia agent się zobowiązał, a więc dotyczą wyboru planu osiągnięcia celu. W przypadku agentów interfejsu NPC nie wprowadzono pojęcia „umysłu", gdyż jest on nadmiarowy w stosunku do potrzeb. W konsekwencji tu prezentowane agenty upostaciowione nie korzystają z architektury BDI. Na gruncie sztucznej inteligencji prowadzone są również prace dotyczące zdobywania nowej wiedzy przez agenty. Przykładowo praca [27] opisuje system wieloagentowy stosujący logikę niemonotoniczną do wnioskowania, w celu odpowiedzi na zapytania. Mógłby to być ciekawy kierunek rozwoju interfejsu NPC, dający użytkownikowi możliwość nie tylko sterowania prezentacją wyników działania platformy, ale również kierowania do niej zapytań.

Na gruncie informatyki rozpatrywane są również masywne systemy wieloagentowe (ang. *Massive Multi-Agent Systems*), złożone z olbrzymiej liczby agentów. W takich systemach głównym problemem jest skalowalność komunikacji agentów. Rozwiązanie tego problemu w dużej mierze zależy od technologii implementacji agentów jako wątków, oraz sposobu realizacji przesyłania wiadomości między agentami. Polem zastosowania takich systemów są, przykładowo, obliczenia ewolucyjne [20]. Chwilowo w systemach robotycznych tak olbrzymie liczby agentów nie występują, stopień współbieżności jest więc dużo niższy niż ten przedstawiony w pracy [20]. Należy się jednak spodziewać, że w przyszłości roje robotów mogą stać się ogromne. Wtedy wykorzystanie takich technologii może być koniecznością. Informatycy badają również ewolucyjne systemy wieloagentowe (ang. *Evolutionary Multi-Agent Systems*), w których każdy z agentów reprezentuje pojedyncze rozwiązanie zadania optymalizacji [19]. Agenty wchodzą w interakcje między sobą (rozmnażają się, współzawodniczą o zasoby) przekazując sobie skwantowany zasób zwany energią. Z punktu widzenia informatyki istotne jest odseparowanie obliczeń od modelu ich realizacji (sekwencyjnego, równoległego, typu prze-

pływ danych *dataflow*). Przedstawiany w tym artykule model agenta również abstrahuje od jego implementacji, a więc sposobu prowadzenia obliczeń.

Na zakończenie dyskusji dotyczącej relacji między prezentowanym tu agentem upostaciowionym i agentami definiowanymi w bardzo bogatej literaturze tematu warto jeszcze wspomnieć o inżynierii systemów wieloagentowych [14]. Podstawą jest rozdzielenie fazy specyfikacji systemu od fazy jego implementacji, czyli realizacja ogólnych zaleceń inżynierii oprogramowania [28]. Istotnym jest dodatkowe zalecenie mówiące, że specyfikację należy sporządzić dla każdego agenta oddzielnie, a działanie całości systemu powinno wynikać z interakcji tych agentów. Kolejnym zaleceniem jest przyporządkowanie każdej roli lub grupie ról pokrewnych, które system ma pełnić, jednego agenta realizującego te role. Ponadto zaleca się stosowanie automatów skończonych do opisu działania agentów. Specyfikacja i implementacja interfejsu NPC jest zgodna z tymi ogólnymi zaleceniami. Należy jednak podkreślić, że wśród zaleceń brak było wskazówek na temat wewnętrznej struktury i sposobu działania agentów. Prezentowany tu sposób tworzenia systemu wieloagentowego jest konkretniejszy, przez co ułatwia projektantowi stworzenie systemu wieloagentowego. Zarówno struktura, jak i sposób działania określone są wzorcami.

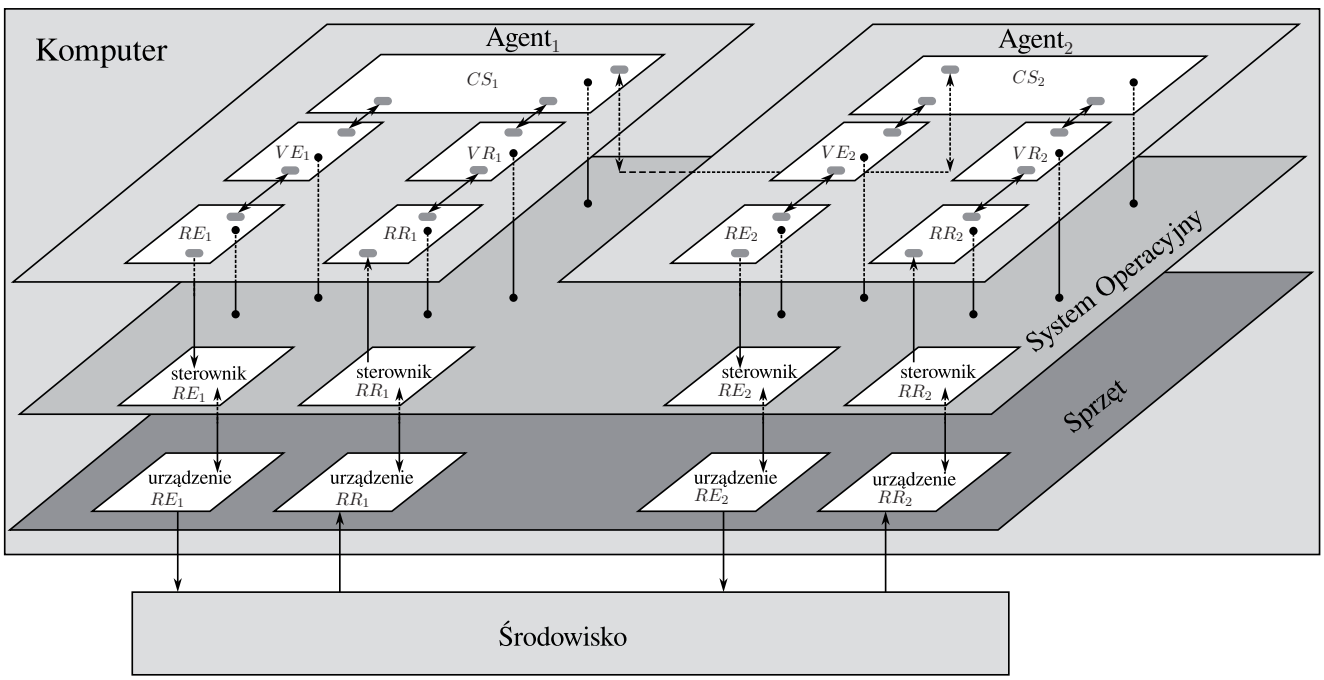
9. Interakcja interfejsu z systemem operacyjnym

Jedną z decyzji dotyczących implementacji jest ta odnosząca się do liczby komputerów, które mają realizować zadania systemu. Bezpośredni wpływ na nią ma obliczeniochłonność stosowanych algorytmów. W przypadku relatywnie niskiej obliczeniochłonności kod wszystkich agentów wchodzących w skład projektowanego systemu można ulokować na jednym komputerze. Z taką sytuacją mamy do czynienia w przypadku interfejsu NPC. Na rys. 14 przedstawiono ogólną zasadę implementacji komunikacji zarówno między agentami jak i podsystemami agentów, w przypadku gdy wszystkie agenty i podsystemy wykonywane są przez ten sam komputer. Komunikacja ta odbywa się za pośrednictwem systemu operacyjnego. Schemat ten pokazuje również sposób interakcji między podsystemami a sterownikami (ang. *driverami*) urządzeń oraz między sterownikami a urządzeniami wykorzystywanymi przez interfejs NPC.

Jak widać, implementacja zakłada powstanie struktury trójwarstwowej, w której górną warstwę bezpośrednio określa specyfikacja. Poniżej znajduje się warstwa odpowiadająca systemowi operacyjnemu komputera. W tej warstwie rezydują zarówno sterowniki urządzeń – receptorów i efektorów, a także oprogramowanie organizujące przekazywanie informacji między wątkami i procesami. Najniższą warstwę tworzą tylko urządzenia, które same w sobie mogą zawierać oprogramowanie, ale to tworzone jest przez producentów sprzętu i jest ukryte przed projektantem systemu.

10. Podsumowanie

Artykuł poświęcony jest specyfikacji i implementacji wielomodalnego interfejsu do sterowania prezentacją wyników wytwarzanych przez Narodową Platformę Cyberbezpieczeństwa. Ta część artykułu koncentruje się na opisie struktury i sposobu działania interfejsu. Do jego specyfikacji wykorzystano koncepcję agentów upostaciowionych. Dotychczas ta metoda specyfikacji stosowana była do systemów robotycznych, a więc do opisu urządzeń działających w środowisku fizycznym. Tutaj jednak mamy do czynienia z systemem działającym na styku środowiska fizycznego, w którym funkcjonuje operator, oraz cyberprzestrzeni, w której tworzone są wyniki analizy aktywności w sieci. Ten charakter środowiska wymagał specyficznego rozdzielenia systemu i środowiska. W szczególności zakwalifikowano ekran monitora do środowiska, natomiast elementy okienek i kursory pojawiające się na tym ekranie zaliczono do receptorów i efektorów reprezentujących je agentów. Ponadto agenty zostały pogrupowane w moduły, co nadało systemowi hierarchiczny charakter. Wyróżniono trzy takie moduły: audio, wizji oraz prezentacji. Dzięki zastosowaniu grup współpracujących agentów struktura powstałego interfejsu ma naturę otwartą. System może być rozszerzony o kolejne moduły, jeżeli trzeba dodać następne tryby porozumiewania się operatora z systemem, a ponadto każdy z modułów może być wzbogacany o kolejne agenty realizujące dodatkowe funkcje tych modułów. Obecne tryby porozumiewania się mogą być roz-



Rys. 14. Schemat komunikacji międzyagentowej i wewnątrzagentowej zastosowanej w interfejsie NPC
Fig. 14. Intra-agent and inter-agent communication employed by the NCP interface

szerzane przez dodawanie nowych komend głosowych oraz gestów sterujących, w tym wykonywanych głową oraz sterowanie mimiką twarzy. Wszystko to umożliwia względnie proste rozszerzenie struktury, a przez to zwiększenie jej możliwości. Rozszerzając tę strukturę należy wykorzystać to, co już istnieje, jako wzorzec. Zastosowana metoda projektowa ma swoje korzenie w inżynierii oprogramowania, w szczególności przykłada istotną wagę do dwóch faz tworzenia systemu: fazy specyfikacji odpowiadającej na pytanie, co ma być stworzone, oraz fazy implementacji definiującej, jak to ma być wykonane. To rozdzielenie zainteresowania (ang. *separation of concerns*) projektanta prowadzi do stworzenia hierarchicznego systemu o przejrzystej architekturze. Wspomniana hierarchia obejmuje następujące warstwy: modułu, agenta, automatu skończonego, funkcji przejścia.

Obecnie podstawowymi sposobami komunikacji operatora z interfejsem są polecenia wydawane albo głosem albo gestem. Wprowadzono też bezpieczny tryb dostępu do platformy wykorzystujący identyfikację mówcy. Wstępne testy wykazały wysoką skuteczność obecnych trybów porozumiewania się z systemem. Druga część artykułu poświęcona będzie zastosowanym algorytmom rozpoznawania: mowy, mówcy i gestów. Ponadto przedstawione zostaną wyniki przeprowadzonych testów.

Podziękowania

Praca wykonana w ramach projektu CYBERSECIDENT /369195/I/NCBR/2017, współfinansowanego przez Narodowe Centrum Badań i Rozwoju w ramach programu CyberSecIdent.

Bibliografia

1. The National Cyber Security Centre, Holandia, https://www.ncsc.nl [on-line: dostęp 5-04-2019].
2. The National Cyber Security Centre, Wielka Brytania, https://www.ncsc.gov.uk [on-line: dostęp 5-04-2019].
3. The National Cybersecurity and Communications Integration Center, USA, https://ics-cert.us-cert.gov [on-line: dostęp 5-04-2019].
4. Arkin R.C., *Behavior-Based Robotics*. MIT Press, 1998.
5. Best D.M., Endert A., Kidwell D., *7 key challenges for visualization in cyber network defense*. Proceedings of the Eleventh Workshop on Visualization for Cyber Security, VizSec '14, 33–40, DOI: 10.1145/2671491.2671497.
6. Bonabeau E., Dorigo M., Theraulaz G., *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York, Oxford, 1999.
7. Bostock M., *Pseudo-Dorling cartogram*, https://bl.ocks.org/mbostock/4055892, 2015. [on-line: dostęp 5-04-2019].
8. Brooks R.A., *Elephants don't play chess*. "Robotics and autonomous systems", Vol. 6, No. 1–2, 1990, 3–15, DOI: 10.1016/S0921-8890(05)80025-9.
9. Brooks R.A., Intelligence without reason. *Artificial intelligence: critical concepts*, 3:107–163, 1991.
10. Brooks R.A., Intelligence without representation. *Artificial Intelligence*, 47(1-3):139–159, January 1991.
11. Brooks R.A., *New approaches to robotics*. "Science", Vol. 253, No. 5025, September 1991, 1227–1232, DOI: 10.1126/science.253.5025.1227.
12. Cao N., Lin C., Zhu Q., Lin Y., Teng X., Wen X., *Voila: Visual anomaly detection and monitoring with streaming spatiotemporal data*. IEEE Transactions on Visualization and Computer Graphics, Vol. 24, No. 1, Jan 2018, 23–33, DOI: 10.1109/TVCG.2017.2744419.
13. Cetnarowicz K., *Inteligencja wokół nas. Współdziałanie agentów softwareowych, robotów, inteligentnych urządzeń*, Vol. 15, rozdział M–agent, 137–167. Wydawnictwo EXIT, 2010.
14. DeLoach S., Wood M., Sparkman C., *Multiagent systems engineering*. "International Journal of Software Engineering and Knowledge Engineering", Vol. 11, No. 3, 2001, 231–258, DOI: 10.1142/S0218194001000542.
15. Dumas B., Lalanne D., Oviatt S.. *Human Machine Interaction*, Vol. 5440 serii *Lecture Notes in Computer Science*, rozdział Multimodal Interfaces: A Survey of Principles, Models and Frameworks, 3–26, Springer, 2009.
16. Jaimes A., Sebe N., *Multimodal human–computer interaction: A survey*. "Computer Vision and Image Understanding", Vol. 108, No. 1–2, 2007, 116–134, Special Issue on Vision for Human-Computer Interaction, DOI: 10.1016/j.cviu.2006.10.019.
17. Kasprzak W., *Rozpoznawanie obrazów i sygnałów mowy*. Oficyna Wydawnicza Politechniki Warszawskiej, 2009.
18. Kornuta T., Zieliński C., *Robot control system design exemplified by multi-camera visual servoing*, "Journal of Intelligent & Robotic Systems", Vol. 77, No. 3–4, 2013, 499–524, DOI: 10.1007/s10846-013-9883-x.
19. Krzywicki D., Faber Ł., Dębski R., *Concurrent agent-based evolutionary computations as adaptive dataflows*. "Concurrency and Computation Practice and Experience", Vol. 30, No. 22, 2018, 1–29, DOI: 10.1002/cpe.4702.
20. Krzywicki D., Turek W., Byrski A., Kisiel-Dorohinicki M., *Massively concurrent agent-based evolutionary computing*. "Journal of Computational Science", Vol. 11, 2015, 153–162, DOI: 10.1016/j.jocs.2015.07.003.
21. McKenna S., Staheli D., Fulcher C., Meyer M., *Bubblenet: A cyber security dashboard for visualizing patterns*. Eurographics Conference on Visualization (EuroVis), Vol. 35, No. 3, June 2016, 281–290, DOI: 10.1111/cgf.12904.
22. Nwana H.S., Ndumu D.T., *A Brief Introduction to Software Agent Technology*, 29–47. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.
23. Oviatt S., Schuller B., Cohen P., Sonntag D., Potamianos G., Kruger A. (eds), *The Handbook of Multimodal-Multisensor Interfaces, Vol. 1: Foundations, User Modeling, and Common Modality Combinations*. ACM Books Series. Association for Computing Machinery (ACM), 2017.
24. Padgham L., Winikoff M., *Developing Intelligent Agent Systems: A Practical Guide*. John Wiley & Sons, 2004.
25. Pałka P., *Wieloagentowe systemy decyzyjne*. Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa 2019.
26. Russell S., Norvig P., *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, N.J., 1995.
27. Rybiński H., Ryzko D., Wiech P., *Learning of Defaults by Agents in a Distributed Multi-Agent System Environment*, 197–213. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
28. Sacha K., *Inżynieria oprogramowania*. Wydawnictwo Naukowe PWN, Warszawa 2010.
29. Sethi A., Wills G., *Expert-interviews led analysis of EEVi – a model for effective visualization in cybersecurity*. 2017 IEEE Symposium on Visualization for Cyber Security (VizSec), Oct 2017, 1–8, DOI: 10.1109/VIZSEC.2017.8062195.
30. Shiravi H., Shiravi A., Ghorbani A., *A survey of visualization systems for network security*. IEEE Transactions on Visualization and Computer Graphics, Vol. 18, No. 8, 2012, 1313–1329, DOI: 10.1109/TVCG.2011.144.
31. Shoham Y., *Agent-oriented programming*. "Artificial Intelligence", Vol. 60, No. 1, 1993, 51–92.
32. Song B., Choi J., Choi S.-S., Song J., *Visualization of security event logs across multiple networks and its application to a CSOC*. "Cluster Computing", 1–12, Nov 2017.
33. Turk M., *Multimodal interaction: A review*. "Pattern Recognition Letters", Vol. 36, 2014, 189–195, DOI: 10.1016/j.patrec.2013.07.003.
34. Wooldridge M., *Multiagent systems*. Ch. Intelligent Agents, 27–77. MIT Press, Cambridge, MA, USA, 1999.

35. Zieliński C., *Inteligencja wokół nas. Współdziałanie agentów softwareowych, robotów, inteligentnych urządzeń*, wolumen 15, rozd. Formalne podejście do programowania robotów – struktura układu sterującego, 267–300, EXIT, 2010.
36. Zieliński C., Figat M., Hexel R., *Communication within multi-FSM based robotic systems*. "Journal of Intelligent & Robotic Systems", Vol. 93, No. 3–4, 2019, 787–805, DOI: 10.1007/s10846-018-0869-6.
37. Zieliński C., Kornuta T., *Diagnostic requirements in multi-robot systems*. Korbicz J., Kowal M. (eds), *Intelligent Systems in Technical and Medical Diagnostics*, wolumen 230 serii *Advances in Intelligent Systems and Computing*, 345–356. Springer, 2014.
38. Zieliński C., Kornuta T., Winiarski T., *A systematic method of designing control systems for service and field robots*. 19th IEEE International Conference on Methods and Models in Automation and Robotics, MMAR, 1–14. IEEE, 2014.
39. Zieliński C., Stefańczyk M., Kornuta T., Figat M., Dudek W., Szynkiewicz W., Kasprzak W., Figat J., Szlenk M., Winiarski T., Banachowicz K., Zielińska T., Tsardoulias E.G., Symeonidis A.L., Psomopoulos F.E., Kintsakis A.M., Mitkas P.A., Thallas A., Reppou SE., Karagiannis G.T., Panayiotou K., Prunet V., Serrano M., Merlet J.-P., Arampatzis S., Giokas A., Penteridis L., Trochidis I., Daney D., Iturburu M., *Variable structure robot control systems: The RAPP approach*. „Robotics and Autonomous Systems", Vol. 94, 2017, 226–244, DOI: 10.1016/j.robot.2017.05.002.
40. Zieliński C., Winiarski T., Kornuta T., *Agent-based structures of robot systems*. J. Kacprzyk, et al. (eds), *Trends in Advanced Intelligent Control, Optimization and Automation*, Vol. 577 AISC, 493–502, 2017, DOI: 10.1007/978-3-319-60699-6_48.

Agent Structure of Multimodal User Interface to the National Cybersecurity Platform – Part 1

Abstract: This two part paper presents an interface to the National Cybersecurity Platform utilising gestures and voice commands as the means of interaction between the operator and the platform. Cyberspace and its underlying infrastructure are vulnerable to a broad range of risk stemming from diverse cyber-threats. The main role of this interface is to support security analysts and operators controlling visualisation of cyberspace events like incidents or cyber-attacks especially when manipulating graphical information. Main visualization control modalities are gesture- and voice-based commands. Thus the design of gesture recognition and speech-recognition modules is provided. The speech module is also responsible for speaker identification in order to limit the access to trusted users only, registered with the visualisation control system. This part of the paper focuses on the structure and the activities of the interface, while the second part concentrates on the algorithms employed for the recognition of: gestures, voice commands and speakers.

Keywords: National Cybersecurity Platform, image recognition, gesture recognition, speech recognition, speaker recognition

prof. dr hab. inż. Włodzimierz Kasprzak
W.Kasprzak@elka.pw.edu.pl
ORCID: 0000-0002-4840-8860

Jest profesorem na Wydziale Elektroniki i Technik Informacyjnych (WEiTI) Politechniki Warszawskiej (PW). W PW pracuje od 1997 r. Obecnie jest kierownikiem Zespołu Percepcji Maszyn w IAiIS. Jego zainteresowania badawcze obejmują algorytmy i techniki automatycznej analizy obrazów i sygnału mowy oraz metody rozpoznawania wzorców i uczenia maszynowego. Jest autorem/współautorem ponad 160 publikacji naukowych i członkiem towarzystw naukowych IEEE, IEEE CI Soc., IEEE SMC Soc., TPO (IAPR), DAGM.

dr hab. inż. Wojciech Szynkiewicz
W.Szynkiewicz@elka.pw.edu.pl
ORCID: 0000-0001-6348-1129

Adiunkt w Instytucie Automatyki i Informatyki Stosowanej na Wydziale Elektroniki i Technik Informacyjnych Politechniki Warszawskiej. Zajmuje się zagadnieniami sterowania i programowania robotów, autonomicznej nawigacji, planowania zadań i cyberbezpieczeństwa robotów. Autor i współautor ponad 90 publikacji w wymienionych obszarach badawczych.

mgr inż. Maciej Stefańczyk
maciej.stefanczyk@pw.edu.pl
ORCID: 0000-0001-9948-6319

Absolwent Wydziału Elektroniki i Technik Informacyjnych Politechniki Warszawskiej. W 2010 r. uzyskał tytuł inżyniera, w 2011 r. tytuł magistra inżyniera, oba z wyróżnieniem. W 2011 r. rozpoczął prace nad doktoratem dotyczącym zastosowania aktywnej wizji wraz z systemami opartymi na bazie wiedzy w systemie sterowania robotów. Od 2012 r. pracuje na Politechnice Warszawskiej, początkowo jedynie przy realizacji projektów, a obecnie na stanowisku asystenta. Główne zainteresowania naukowe obejmują zastosowanie informacji wizyjnej zarówno w robotyce, jak i systemach rozrywki komputerowej.

mgr inż. Wojciech Dudek
wojciech.dudek@pw.edu.pl
ORCID: 0000-0001-5326-1034

Jest doktorantem i asystentem naukowym w Politechnice Warszawskiej w Instytucie Automatyki i Informatyki Stosowanej. Pracował w międzynarodowych grantach badawczych, m.in. RAPP (7PR Komisji Europejskiej) i INCARE (Active and Assisted Living JP Komisji Europejskiej). Jego zainteresowania naukowe obejmują systemy sterowania usługowymi robotami mobilnymi, ich lokalizacje, nawigacje i harmonizacje ich zadań.

mgr inż. Maksym Figat
maksym_figat@pw.edu.pl
ORCID: 0000-0002-1898-0540

Absolwent Wydziału Matematyki i Nauk Informacyjnych. W 2012 r. uzyskał tytuł inżyniera, a w 2013 r. tytuł magistra (z wyróżnieniem) na specjalizacji „Projektowanie systemów CAD/CAM". Jest doktorantem na Wydziale Elektroniki i Technik Informacyjnych Politechniki Warszawskiej (PW). W PW pracuje od 2014 r. w projektach badawczych krajowych i międzynarodowych, a od 2018 r. na stanowisku asystenta badawczo-dydaktycznego. Jego zainteresowania badawcze koncentrują się na zagadnieniach związanych z językami dziedzinowymi, specyfikacja systemów robotycznych, automatyczna generacja kodu sterowników robotów na podstawie formalnej specyfikacji.

mgr inż. Maciej Węgierek
wegierek.maciej@gmail.com
ORCID: 0000-0003-0779-255X

Jest doktorantem w Politechnice Warszawskiej w Instytucie Automatyki i Informatyki Stosowanej. Pracował w międzynarodowym grancie badawczym INCARE (Active and Assisted Living JP Komisji Europejskiej). Jego zainteresowania naukowe obejmują specyfikacje wymagań, struktury oraz zasady działania sterowników robotów.

mgr inż. Dawid Seredyński
dawid.seredynski@pw.edu.pl
ORCID: 0000-0003-2528-6335

Jest doktorantem na Wydziale Elektroniki i Technik Informacyjnych (WEiTI) Politechniki Warszawskiej (PW). W PW pracuje od 2013 r. w projektach badawczych krajowych i międzynarodowych, a od 2018 r. na stanowisku asystenta naukowo-dydaktycznego. Jego zainteresowania badawcze obejmują zagadnienia związane z planowaniem i realizacja złożonych zadań przez roboty usługowe.

prof. dr hab. inż. Cezary Zieliński
c.zielinski@ia.pw.edu.pl
ORCID: 0000-0001-7604-8834

Jest profesorem zwyczajnym na Wydziale Elektroniki i Technik Informacyjnych (WEiTI) Politechniki Warszawskiej (PW). W PW pracuje od 1985 r., a od 2008 r. również w Przemysłowym Instytucie Automatyki i Pomiarów PIAP. W PW sprawował funkcje: prodziekana ds. nauki i współpracy międzynarodowej WEiTI (2002–2005), zastępcy dyrektora ds. naukowych Instytutu Automatyki i Informatyki Stosowanej (IAiIS) (2005–2008), dyrektora IAIS (2008–2016) oraz prodziekana ds. ogólnych WEiTI (2016–). Od 1996 r. jest kierownikiem Zespołu Robotyki w IAiIS. Od 2007 r. jest członkiem Komitetu Automatyki i Robotyki Polskiej Akademii Nauk. Jego zainteresowania badawcze koncentrują się na zagadnieniach związanych z programowaniem i sterowaniem robotów.