WYBRANE ARCHITEKTURY KOGNITYWNE W ROBOTYCE – PORÓWNANIE I ZASTOSOWANIA

Maciej SŁOWIK[*], Daniel OŁDZIEJ[*]

[*]Katedra Automatyki i Robotyki, Wydział Mechaniczny, Politechnika Białostocka, ul. Wiejska 45 C, 15-351 Białystok

maciejslowik@gmail.com, danoj@wp.pl

Streszczenie: Autonomiczne roboty mobilne są wykorzystywane w wielu nowych zastosowaniach. Zastosowania te wymagają autonomii trudnej do przewidzenia przez projektanta systemów robotycznych. Stąd też potrzeba zastosowania architektur sterowania zwanych kognitywnymi. Architektury kognitywne mają zaimplementowane mechanizmy, dzięki którym potrafią autonomicznie rozwiązywać nowe problemy oraz uczyć się na podstawie rozwiązań problemów już napotkanych. W artykule omówiono trzy różne architektury kognitywne oraz implementacje dwóch z nich oraz dokonano symulacji.

1. WSTĘP

Współcześnie wymaga się od autonomicznych robotów mobilnych realizacji coraz bardziej złożonych zadań w złożonych środowiskach. Zwykłe algorytmy działania oparte na planowaniu oraz interakcji danych z sensorów z elementami wykonawczymi są w wielu wypadkach niewystarczające. Ponadto w złożonych, nieznanych środowiskach, w których wymagamy interakcji z otoczeniem, nie sposób przewidzieć i zaplanować odgórnie, reakcji robota na zdarzenia, które będą wymagać jego reakcji. Wymaga to więc innego podejścia, opartego nie na zaprogramowanych rozwiązaniach, lecz działania „inteligentnego" w sytuacjach wcześniej dla robota nieznanych. Z pomocą przychodzą nam tu architektury sterowania przejawiające zachowania zwane kognitywnymi. Ich wspólne cechy wg Newella (Newell i inni, 1989) to:

- zorientowanie na cel;
- działanie w złożonym, bogatym i kompleksowym środowisku;
- wymóg posiadania dużej wiedzy;
- możliwość używania symboli oraz abstrakcji;
- elastyczność;
- wymóg uczenia się przy zmianach środowiska oraz przez doświadczenia.

Wymienione cechy realizują architektury kognitywne (nie wszystkie). Ponadto charakteryzuje je podejście całościowe do modelowanych systemów, odporność na błędy (wynikające z braku uwzględnienia jakiejś sytuacji), uczenie się oraz budowa warstwowa. Użycie tych architektur w sterowaniu autonomicznym robotem mobilnym wymaga opisania środowiska (podania ram ograniczających, wprowadzenia wiedzy, symboli, abstrakcji) oraz stworzenia interfejsu pośredniczącego pomiędzy oprogramowaniem sterującym sensorami oraz elementami wykonawczymi robota, a implementowaną architekturą.

Uogólniając, architektury te cechują się dużą różnorodnością, od takich, które, mają emulować cechy ludzkiej psychiki do sterowania autonomicznymi aparatami powietrznymi. Tak wielka rozpiętość implementacji powoduje trudności we właściwej analizie oraz użyciu konkretnej architektury. W poniższym referacie przedstawiono wybrane architektury kognitywne oraz ich implementacje na rzeczywistych autonomicznych robotach bądź symulowanych robotach mobilnych.

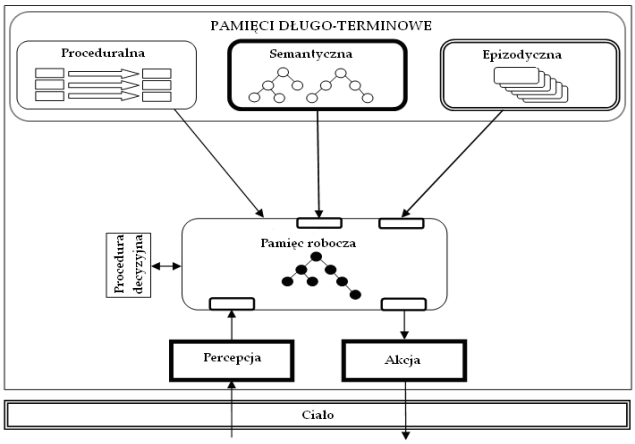
2. SOAR

Soar, wcześniej znane jako SOAR (od State Operator And Result) to architektura kognitywna stworzona przez J. Laird'a, P. Rosenbloom'a oraz A. Newell'a na uniwersytecie Carnagie Mellon, od 2000 roku wspierana i rozwijana na uniwersytecie Michigan. Aktualna wersja architektury oraz oprogramowania to Soar 9.0. Podstawowe właściwości:

- Posiadanie stanów i operatorów – stany cechują właściwości np.: stan początkowy, stan zamierzony. Ponadto opisują one całą obecną sytuację, uwzględniając aktualne informacje (percepcje) oraz opis aktualnych celów i przestrzeni problemu, gdzie operatory opisują kolejne kroki (opisują znaczenie kolejnych kroków) w przestrzeni stanów.
- Rodzaje pamięci: a) pamięć robocza WM (Working Memory) – zawiera percepcje oraz hierarchie stanów i powiązane z nimi operatory. b) pamięć długoterminowa (LTM-Long-Term Memory) jest miejscem tworzenia się wiedzy z posiadanych danych. W Soar możemy wyszczególnić następujące rodzaje wiedzy: proceduralną - zapisaną jako reguły używane automatycznie podczas realizacji cyklu decyzyjnego, semantyczną - zapisaną jako struktury deklaratywne oraz epizodyczną - zapisaną jako epizody. Do dwóch ostatnich rodzajów wiedzy dostęp uzyskujemy poprzez adnotację w pamięci roboczej (WM). Dostęp do LTM uzyskać możemy tylko poprzez WM (Lehman i inni, 2006).
- Interfejs percepcji/ruchu – interfejs ze światem zewnętrznym, dzięki niemu percepcja i akcja mogą odbywać się równoległe.
- Cykl decyzyjny- podzielony na trzy etapy: 1) faza wy-

pracowania operatora z LTM, sugerowania nowego operatora oraz oceny operatorów 2) stan spoczynku oznacza rozpoczęcie fazy drugiej tj. fazy decyzyjnej, opartej na preferencjach. Rezultatem jest zmiana (wybór) operatora lub impas, w przypadku konfliktu lub braku preferencji 3) ostatnia fazą jest faza aplikacji, gdzie reguły zmieniają wartości stanów (Newell i inni, 1989).

- Impasy – oznaczają brak wiedzy, a tym samym stan wymagający nauki nowego rozwiązania. W razie wystąpienia impasu architektura generuje automatycznie podstany w celu rozwiązania impasu.
- Cztery mechanizmy uczenia – porcjowanie (ang. chunking), uczenie wzmocnione, uczenie epizodyczne oraz uczenie semantyczne. Porcjowanie tworzy nowe reguły podczas rozwiązywania impasów. Uczenie wzmocnione (reinforcement learning) dodaje nowe wartości do preferencji operatorów, uczenie epizodyczne korzysta z poprzednich doświadczeń natomiast semantyczne produkuje bardziej abstrakcyjne schematy deklaratywne (Newell i inni, 1989).



Rys. 1. Schemat architektury Soar, z wyszczególnionymi rodzajami pamięci (Lehman i inni, 2006)

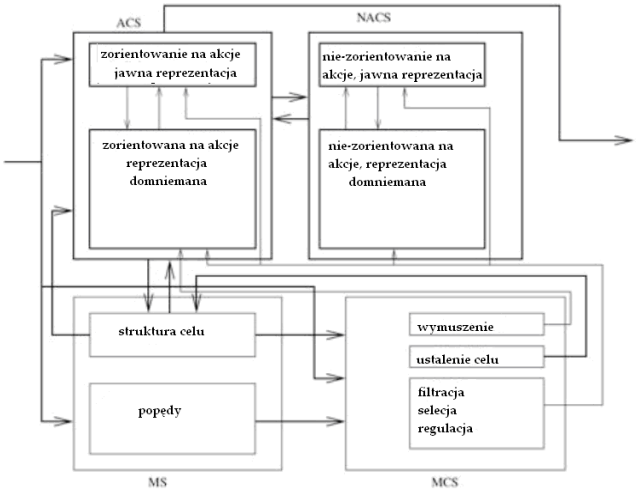
W ostatnich latach (od 2006 roku) wprowadzono dodatki do architektury Soar, mające na celu jeszcze wierniejsze symulowanie ludzkiej kognitywności. Są to: aktywacja pamięci roboczej, dodanie emocji, grupowanie danych oraz wizualny zbiór obrazów (Laird, 2008).

Ze względu na powyższe właściwości architektura Soar została wielokrotnie zaimplementowana, np.

- TacAirSoar, symulacja pilota jako agenta podczas teatru lotniczych działań wojennych;
- sterowanie zadaniami sześcio-nożnego robota mobilnego;
- sterowanie mobilnym robotem SuperDroid (Hanford i inni, 2009).

3. CLARION

CLARION (Cennectionist Learning with Adaptive Rule Induction ON-line), jest architekturą kognitywną rozwijaną przez prof. Ron'a Suna'a z Rensselaer Polytechnic Institute. Wprowadza podział na warstwy: na procesy ukryte, domniemane (ang. implicite) oraz procesy jawne (ang. explicit). Ze względu na zajmowanie się interakcjami pomiędzy tymi procesami architektura CLARION sprawdziła się w symulowaniu zadań psychologii kognitywnej oraz w zastosowaniach sztucznej inteligencji np. sterowanie inteligentnymi agentami. W architekturze CLARION możemy wyróżnić cztery podsystemy, z których każdy bazuje zarówno na procesach ukrytych (warstwa dolna) jak i jawnych (warstwa górna).



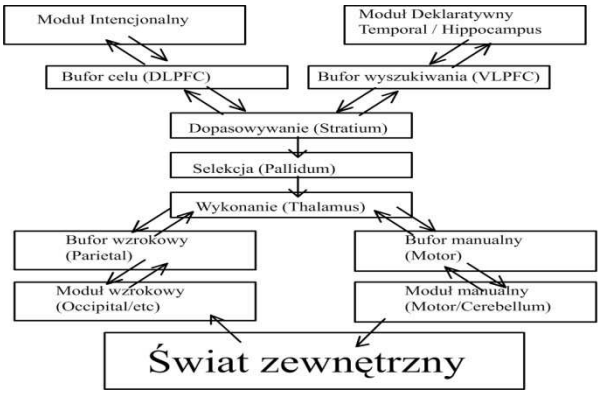
Rys. 2. Schemat budowy i wymiany informacji w architekturze CLARION (Sun, 2007)

- Podsystem skupiony na akcji (ang. action centered subsystem – ACS) – skoncentrowany na sterowaniu wyborem akcji. Nauczenie „w górę", tj. zapis wiedzy jawnej na podstawie wiedzy ukrytej. Ponadto w podsystemie tym zawarte jest nauczanie „w dół" oraz wykorzystanie zewnętrznej wiedzy jako struktur pojęciowych (zasady, reguły, plany), która to może zostać powiązana z wiedzą niskiego bądź wysokiego poziomu.
- Podsystem „nieskupiony" na akcji (ang. non-action centered subsystem – NACS) – zajmujący się reprezentacją ogólnej wiedzy, zwanej semantyczną (Sun, 2007), podporządkowany podsystemowi skupionemu na akcji, przeprowadzającemu zróżnicowane odczyty oraz łączenia na pamięci. Na poziomie wiedzy ukrytej, podsystem ten koduje wiedzę nie zorientowaną na akcję w „pamięć asocjacyjną", z rozproszoną reprezentacją mikro-elementów. W warstwie wiedzy jawnej, wiedza ogólna zapisywana jest jako węzeł (koncept) oraz łączona z odpowiadającymi jej mikro-elementami. Ponadto mamy tu do czynienia z możliwością podejmowania decyzji na zasadzie podobieństwa (np. do poprzednich) oraz łączenia reguł.
- Podsystem motywujący (ang. motivational subsystem MS) – fundament dla motywacji, o który oparte są percepcja, akcja oraz kognitywność. Skupia się na akcjach, które motywują agenta, poprzez maksymalizację wzmocnień (ang. gains), nagrody za wybór danej akcji. Dotyczy wyboru popędów oraz ich interakcji prowadzących do akcji. Możemy podzielić go na niższe (psychologiczne), np.: głód, pragnienie oraz wyższe (społeczne). Możemy też wyróżnić dodatkowe popędy wynika-

jące z procesu realizacji popędów głównych.
– Podsystem meta-kognitywny (ang. meta-cognitive subsystem MCS) – jego rola sprowadza się do sterowania, monitorowania oraz modyfikacji operacji pozostałych podsystemów. Podsystem realizuje dążenie do zachowań poprzez wybór celów (przedstawienie celów) oraz dobór funkcji wzmacniających (ang. reinforcment functions), filtrację informacji (skupienie się na wymiarach wejść podukładów ACS i NACS). Ponadto steruje metodami akwizycji informacji poprzez wybór metod nauczania dla podsystemów ACS i NACS (Sun, 2007). Steruje wykorzystaniem informacji oraz wyborem informacji zwrotnej z podsystemów ACS i NACS. Następnie steruje kognitywnością wyboru węzłów, zarówno w warstwie niższej jak i wyższej lub ich kombinacji.

4. ACT-R

ACT-R (ang. Adaptive Control of Thought – Rational) architektura kognitywna rozwijana głównie przez John'a Robert'a Andersona. Tak jak dwie poprzednie architektury dotyczy teorii oraz narzędzi służących do symulacji modelów architektury. Od wydania w 1993 roku przy pomocy ACT-C stworzono ponad 100 modeli kognitywnych, dla takich zagadnień jak pamięć implikatywna, przetwarzanie metafor, emocje oraz naukę gry tryk-traka (ang. Backgammon). Najaktualniejsza wersja to ACT-R 6.0. Wersja ta jest bardziej odporna na błędy, osiąga większą zgodność dla parametrów dla różnych zadań, posiada więcej mechanizmów generowania produkcji. Podsumowując, bardziej odpowiada naszej wiedzy o funkcjonowaniu mózgu.



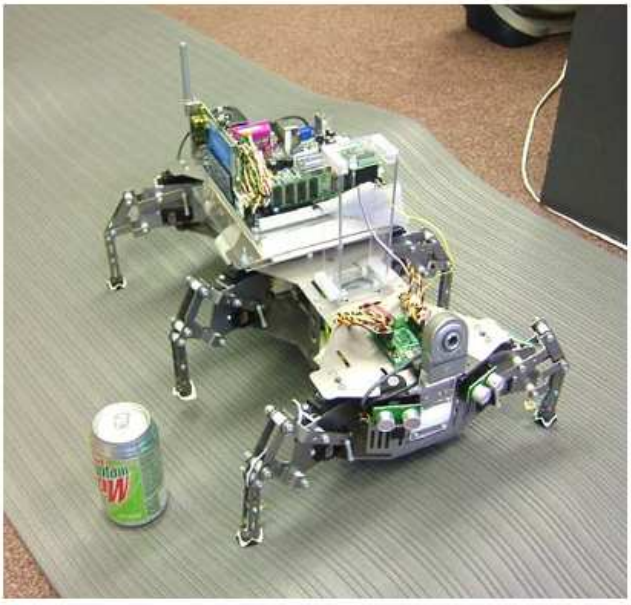
Rys. 3. Schemat architektury ACT-R jako modelu ludzkiego mózgu, nazwy w nawiasach – nazwy obszarów odpowiedzialnych wg teorii ACT-R za realizację tych zadań w mózgu człowieka (Anderson i inni, 2004)

Wybrane właściwości architektury ACT-R:
– nauka na przykładach – polega na uczeniu się poprzez pozyskiwanie z pamięci dawnych doświadczeń;
– nauka użyteczności – nauka wyboru, która z dostępnych strategii jest najkorzystniejsza ze względu na koszty i prawdopodobieństwo sukcesu;
– pojemność pamięci roboczej – modele, w których mamy różną ilość aktywacji, różnią się ilością zajmowanej pamięci w przestrzeni roboczej;
– percepcyjne/motoryczne przetwarzanie powiązań – w modelach realizujących ludzkie zachowania, percepcja i motoryka znajdują się w ramach ograniczających te zjawiska, w ich ograniczeniach (Anderson i inni, 2004).

Ponadto architekturę ACT-R wyróżnia możliwość dołączania modułów np. ACT-R PM, ACT-R R/E (Anderson i inni, 2004), które poszerzają jej zastosowania oraz funkcjonalność, jednocześnie zachowując zgodność z teorią, na której bazuje ACT-R.

5. IMPLEMENTACJA SOAR NA PRZYKŁADZIE STEROWANIA SZŚCIONOŻNYM AUTONOMICZNYM ROBOTEM MOBILNYM HEXACRAWLER

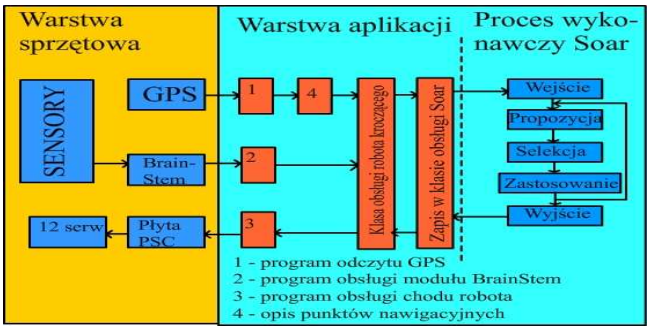
Architektura Soar została użyta do kontroli chodu robota 6-nożnego na podstawie danych sensorycznych z dwóch czujników nacisku oraz dwóch sonarów. Ponadto w sterowaniu użyto innych danych sensorycznych. Cele - to omijanie przeszkód i dążenie do określonego celu, poprzez kontrolę chodu. Elementy budowy robota:
– moduł odbiornika GPS – służący do lokalizacji celu oraz samego robota;
– dwa sonary;
– dwa czujniki nacisku (siły);
– kompas elektroniczny;
– kamera internetowa;
– każde „odnóże" wyposażone jest w dwa serwomechanizmy, jeden umieszczony poziomo a drugi pionowo, razem dwanaście serw.



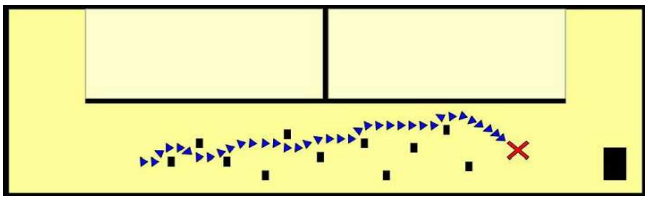
Rys. 4. Robot Hexcrawler (Janrathitikikarn i Long, 2008)

Ze schematu na Rys. 5 wynika sposób działania oraz interakcji, warstwy sprzętowej, warstwy pośredniczącej (ang. middleware) oraz implementacji architektury Soar. Warstwa pośrednicząca przesyła, tłumaczy oraz filtruje odpowiednie sygnały, po czym wprowadza je na wejście imple-

mentacji architektury Soar. Z wejścia sygnały przechodzą poprzez proces decyzyjny. Wyjście z procesu decyzyjnego jest przesyłane z powrotem do warstwy pośredniczącej, tam są odpowiednio dekodowane, po czym trafiają do warstwy sprzętowej, która decyduje o ruchu robota. W badaniach nad ruchem robota, zadaniem było przejście przez teren parkingu oraz ominięcie przeszkód (Janrathitikikarn i Long, 2008).



Rys. 5. Schemat interakcji warstwy sprzętowej, warstwy pośredniczącej oraz implementacji architektury Soar



Rys. 6. Rysunek przedstawiający rzeczywiste testy na parkingu Uniwersytetu Stanowego Pennsylvanii (Janrathitikikarn i Long, 2008)

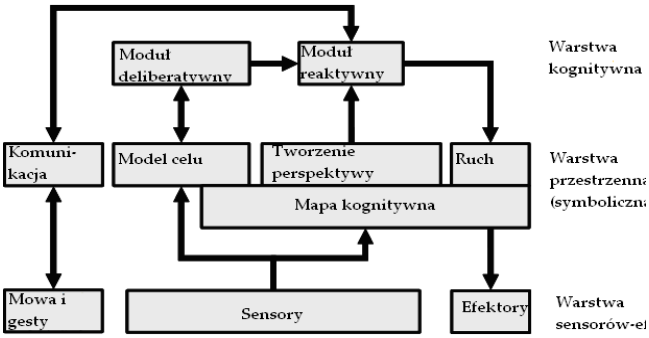
Na Rys. 6 poniżej widać, drogę robota podczas testu polegającego na omijaniu przeszkód oraz na zatrzymaniu się w odległości 3 m od celu. Test miał miejsce na parkingu o powierzchni 36 na 160 metrów.

6. IMPLEMENTACJA ARCHITEKTURY KOGNITYWNEJ ACT-R NA PRZYKŁADZIE KOOPERACJI ROBOTA Z CZLOWIEKIEM PODCZAS MISJI ZWIADOWCZEJ

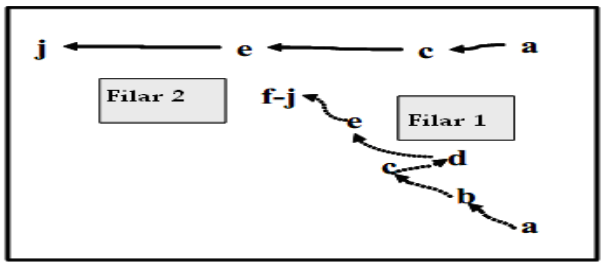
Kolejnym przykładem na implementacje architektury kognitywnej jest użycie architektury ACT-R jako warstwy kognitywnej w sterowaniu robotem mobilnym. Zadaniem drużyny jest interakcja robota z człowiekiem podczas (ang. HRI – Human-Robot Interaction) wykonywania zwiadu. Zadaniem drużyny człowiek-robot jest użycie wiedzy na temat pozycji celu, pola widzenia celu oraz przeszkód w środowisku realizacji działania w celu maksymalnego zbliżenia się do celu, będąc jednocześnie maksymalnie ukrytym (Kennedy i inni, 2007). Architektura robota została zbudowana w oparciu o model Stealthbot, składający się z trzech warstw (Rys. 7).

Warstwa pierwsza składa się z elementów sprzętowych, czujników, efektorów oraz systemu wizji i rozpoznawania mowy. Warstwa druga, nazywana warstwą przestrzenną (symboliczną), stanowi warstwę pośrednią pomiędzy warstwą pierwszą, a warstwą trzecią tj. warstwą kognitywną (z podziałem na moduły: decyzyjny, reaktywny oraz deliberatywny). Warstwa pierwsza dostarczała danych w systemie metrycznym, które to były odpowiednio przetwarzane przez warstwę drugą. Przetwarzanie to miało na celu zapis środowiska symbolicznego jako mapy kognitywnej. Mapa ta jest zapisem w postaci wzajemnych relacji pomiędzy elementami środowiska, w którym znajduje się robot. Ponadto warstwa druga opisuje właściwości ruchu celu.

Rys. 6. Scenariusz misji zwiadowczej: robot , cel (człowiek stojący) oraz członek drużyny (osoba kucająca) (Kennedy i inni, 2007)



Rys. 7. Warstwy robota w oparciu o model Stealthbot (Kennedy i inni, 2007)



Rys. 8. Schemat drogi celu (górna trasa od a do j) oraz robota Stealthbot (prawy dolny róg od a do j)

Przesłanie informacji z warstwy drugiej odbywa się poprzez porcjowanie danych (ang. chunking) i umieszczanie ich w pamięci deklaratywnej architektury ACT-R. Przesłanie występuje podczas zmian położenia celu lub kierunku ruchu w warstwie mapy kognitywnej. Wpływa to na wygenerowane produkcje oraz na reakcje systemu Stealthbot.

Wiedzę w implementacji architektury ACT-R możemy podzielić na dwa rodzaje: wiedzę deklaratywną oraz proce-

duralną. Wiedza deklaratywna zawiera porcje informacji opisane atrybutami. Informacje te pobierane są z warstwy przestrzennej (symbolicznej). Opisują one zmiany w ruchu celu oraz ogólne postrzeganie przestrzenne robota. Wiedza proceduralna to opis reguł oraz metod tworzenia produkcji odpowiedzialnych za, m.in.: przewidywanie następnego położenia celu, komunikacje z członkiem drużyny oraz wybór miejsca do ukrycia przed celem.

Na Rys. 8 widzimy poruszającego się człowieka za filarami z punktu a do punktu j (górna ścieżka). Jednocześnie robot porusza się dolną ścieżką (a-f). W pewnym momencie (punkt c) wykrył, że jest widoczny i ukrył się za filarem w pozycji d. Następnie wykonał ruch do pozycji e, od której ponownie śledził cel do pozycji f-j (Kennedy i inni, 2007).

Podsumowując, naukowcom z Naval Reaserch Laboratory, udało się zaimplementować architekturę ACT-R w systemie Stealthbot. Robot poprawnie realizował zadanie śledzenia celu oraz uniknął wykrycia dzięki ukrywaniu się za przeszkodami (filar 1, filar 2). Ponadto, robot komunikował się z ludzkim członkiem drużyny (człowiek kucający).

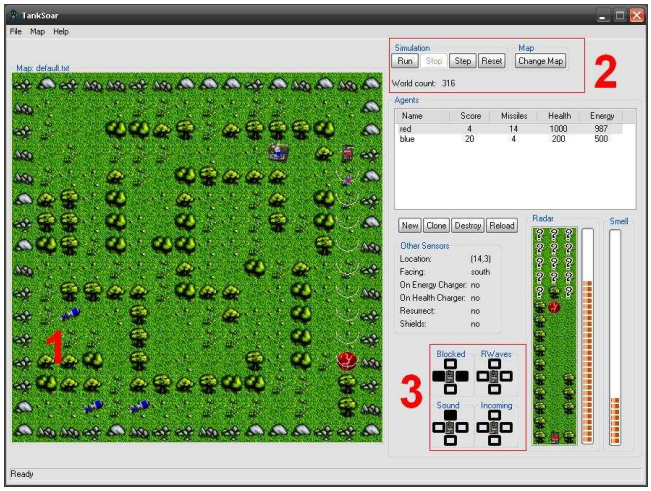
7. SYMULACJE W ŚRODOWISKU SOARSUITE

Badania symulacyjne architektury SOAR zostały przeprowadzone dla scenariusza o nazwie Tanksoar w środowisku Soar-Suite-9.3.0-win-x86 (Rys. 9). Scenariusz ten symuluje walkę dwóch lub więcej czołgów (agentów) na mapie z przeszkodami (na wzór gry komputerowej Tanks! z lat siedemdziesiątych). Celem symulacji oznaczającym zwycięstwo danego agenta jest osiągnięcie 50 punktów. Agenci posiadają następujące atrybuty – ilość punktów, liczba pozostałych pocisków, ilość punktów zdrowia oraz ilość energii osłon. Agent zdobywa punkty poprzez trafienie lub zniszczenie przeciwnika. Analogicznie agent traci punkty życia oraz osłony gdy zostanie trafiony pociskiem przeciwnika. Parametry symulacji które można ustawić to również m.in:

Parametr	**Wartość domyślna**
maksymalna liczba pocisków	15
maksymalna liczba energii osłon	1000
maksymalna liczba zdrowia	1000
liczba punktów za trafienie przeciwnika	2
liczba punktów za zniszczenie przeciwnika	3
liczba punktów za zostanie trafionym pociskiem przeciwnika	– 1
liczba punktów za zostanie zniszczonym przez pocisk przeciwnika	– 2
ilość energii oraz punktów życia po trafieniu przez pocisk przeciwnika	odpowiednio – 250 oraz – 400

W scenariuszu Tanksoar, Agent posiada źródła informacji takie jak: radar, dźwięk oraz potrafi wykryć zbliżający się pocisk przeciwnika. Potrafi również określić istnienie przeszkód w jego najbliższej odległości (Rys. 9. Blocked – zablokowany z prawej i lewej strony). W celu pokonania przeciwnika (osiągnięcia 50 punktów) agent korzysta z następujących trybów działania:

– tryb ataku – gdy przeciwnik został wykryty na radarze w polu rażenia oraz gdy liczba punktów życia oraz energia osłon wynosi więcej niż 200;
– tryb pogoni – gdy agent „usłyszy" przeciwnika oraz został on wykryty przy pomocy radaru;
– tryb ucieczki – gdy agent „usłyszy" przeciwnika, jeśli wykryje go przy pomocy radaru lub zauważy zbliżający się pocisk a poziom energii oraz punktów życia wynosi mniej niż 200;
– tryb wędrówki – gdy agent nie wykrywa przeciwnika przy pomocy radaru ani przez dźwięk (oba źródła są aktywne ich zmiana wprowadza zmianę trybu agenta) oraz zbliżającego się do niego pocisku.



Rys. 9. Program symulacyjny, z wybranym scenariuszem Tanksoar, 1 – obszar symulacji, 2 – panel sterowania symulacją, 3 – informacje dotyczące aktualnego stanu agentów oraz informacji które posiadają

W symulacjach przeprowadzonych przez autorów zbadano wpływ pominięcia informacji sensorycznej na rozgrywkę pomiędzy agentami. Autorzy środowiska Soar-Suite dostarczają trzy rodzaje agentów:

– obscure-bot – według autorów Saor najbardziej zaawansowany oraz służący do testowania oraz współzawodnictwa z innymi agentami;
– simple-bot – agent posługujący się tylko radarem;
– simple-sound-bot – agent posługujący się radarem oraz dźwiękiem

W symulacjach po pozbawieniu agentów wybranych źródeł informacji zewnętrznych tj.:dźwięku dla trybu wędrówki oraz radaru dla trybu pogoni, określano ilość iteracji potrzebnych do osiągnięcia 50 punktów przez jednego z dwóch agentów. Każdy wariant symulacji powtarzano piętnastokrotnie. Poniżej porównano wyniki symulacji:

Rezultaty symulacji:

Nr symulacji	**Nazwa agenta (zablokowane źródła sygnałów)**	**Przedział iteracji**	**Średnia iteracji**
1	Obscure-bot	476-1320	786
2	Simple-bot	1189-2390	1641
3	Simple-sound-bot	512-1923	1372
4	Simple-sound-bot (radar)	855-2712	1712
5	Simple-sound-bot (radar, dźwięk)	1146-3825	2376

Odebranie źródeł informacji sensorycznych powoduje wzrost czasu (kolejnych iteracji wykonania zadania przez agenta). Doprowadza również do sytuacji, w których

agent działa w sposób nieprzewidziany (zderzenia ze ścianami, brak ucieczki lub powstawanie impasów). W pierwszych trzech symulacjach, zachowano wszystkie źródła sygnałów zewnętrznych. W czwartej odebrano możliwość korzystania z informacji udostępnianych przez radar w trybie pogoni. W symulacji piątej usunięto sygnał pochodzący z radaru oraz możliwość reagowania na dźwięki pochodzące od przeciwnika w trybie wędrówki. Wydłużyło to prawie dwukrotnie liczbę iteracji potrzebną do osiągnięcia 50 punktów – celu misji w scenariuszu Tanksoar.

8. PORÓWNANIE I PODSUMOWANIE

Możemy zauważyć wspólne płaszczyzny na których możemy porównać wybrane właściwości wyżej wymienionych architektur, dodając, że ich wybór był trudny ze względu na różnice w ich budowie.

1) Rodzaje pamięci – wszystkie trzy pamięci architektur posiadają podział na pamięć roboczą oraz pamięć długotrwałą z informacjami zapisanymi w postaci symbolicznej. Różnią się natomiast ilością oraz rodzajami danych zapisywanych w poszczególnych pamięciach.

Tab. 1. Porównanie rodzajów pamięci architektur kognitywnych

Architektura	**Pamięć robocza**	**Pamięć długoterminowa**
Soar	informacje z aktualnego wejścia (np. czujniki), aktualne stany	Wiedza proceduralna, deklaratywna oraz epizodyczna
ACT-R	Cel, aktualne informacje ze świata zewnętrznego dostępne przez różne bufory	Wiedza deklaratywna w module deklaratywnym oraz proceduralna w proceduralnym
CLARION	Informacje tymczasowe	Wiedza proceduralna w warstwie górnej, deklaratywna w warstwie dolnej

2) Uczenie się – każda z architektur realizuje to na swój własny sposób.

Tab. 2. Porównanie mechanizmów uczenia się architektur

Architektura	**Mechanizmy uczenia**
Soar	Porcjowanie (ang. chunking) – metoda główna oraz uczenie wzmocnione
ACT-R	Kompilacja najlepszej reguły (produkcji) ze zbioru posiadanych reguł
CLARION	Nauczanie funkcją Q (warstwa dolna), ekstrakcja reguł (warstwa górna)

W powyższej pracy przedstawiono różne architektury kognitywne Soar, ACT-R oraz CLARION. Każda z nich reprezentuje różne podejście do modelowania oraz implementacji kognitywności jako systemu opisu oraz implementacji sztucznej inteligencji. Celem było modelowanie oraz symulacja kognitywności człowieka, jako odpowiedzi na problem modelowania ludzkiej inteligencji w systemach agentowych, autonomicznych robotach mobilnych oraz tworzeniu coraz lepszych jej modeli. W zależności od rozwoju prac nad badanymi architekturami zauważamy coraz większe możliwości ich implementacji w robotyce. Szczególnie w aplikacjach wieloagentowych, gdzie występuje współpraca wielu agentów-robotów oraz współpraca robota z człowiekiem (Masłowski i Ulatowski, 2005). Zwłaszcza w środowiskach dynamicznych, zmiennych, gdzie człowiek musi estymować pewne wartości, opierając się na życiowym doświadczeniu. Podczas realizacji coraz bardziej złożonych celów w złożonych środowiskach, to samo zadanie będą musiały spełniać roboty autonomiczne pozbawione jednak bagażu wiedzy wynikającej z doświadczenia. W zastosowaniu tym sprawdzają się systemy oparte na architekturach kognitywnych, które z założenia tworzą „doświadczenie”(bazę wiedzy), które robot sterowany taką architekturą wykorzysta dla rozwiązania problemu.

LITERATURA

1. **Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., Qin, Y.** (2004), An Integrated Theory of the Mind, *Psychological Review*, 111, (4), 1036-1060.
2. **Hanford S. D, Janrathitikarn O., Long L. N.** (2009), Control of Mobile Robots Using the Soar Cognitive Architecture, *Journal of Aerospace Computing Information and Communication*, Vol. 6,strony 69-91 .
3. **Janrathitikarn O., Long L. N.** (2008), Gait Control of a Six-Legged Robot on Unlevel Terrain Using a Cognitive Architecture, *IEEE Aerospace Conference*, Big Sky, Montana, 1-9.
4. **Kennedy W. G., Bugajska, M. D., Marge, M., Adams, W., Fransen, B. R., Perzanowski, D., Schultz, A.C., Trafton, J. G.** (2007), Spatial Representation and Reasoning for Human-Robot Collaboration, *Proceedings of the Twenty-Second Conference on Artificial Intelligence*, Vancouver, Canada: AAAI Press, 1554-1559
5. **Laird J. E.** (2008), Extending the Soar Cognitive Architecture, *Artificial General Intelligence Conference*, Memphis, 224-235.
6. **Lehman J. F., Laird J. E., Rosenbloom P.** (2006), A Gentle Introduction To Soar, *An Architecture For Human Cognition*, http://ai.eecs.umich.edu/soar/.../GentleIntroduction-2006.pdf
7. **Masłowski. A, Ulatowski. W.** (2005), Modelowanie działań operatora w sterowaniu wieloagentowym podsystemem transportowym w systemie wytwarzania, *Pomiary Automatyka Sterowanie*, 2, strony 6-11.
8. **Newell A., Rosenbloom, P. S., Laird J. E.** (1989) Symbolic architectures for cognition, strony 91-131.
9. **Posner M. I.** (Ed.) (1989), *Foundations of Cognitive Science*, Cambridge, MA: Bradford Books/MIT Press.
10. **Sun R.** (2007), The Importance of Cognitive Architectures: An Analysis Based on CLARION, *Journal of Experimental & Theoretical Artificial Intelligence Archive*, Vol. 19 Issue 2, strony 159-193.

COGNITIVE ARCHITECTURES SURVEY OF METHODS AND IMPLEMENTATIONS

Abstract: Autonomous mobile robots are used in different new applications. These applications put different requirements on the autonomy tasks of robotic systems. The more advances in autonomy solutions are connected with these cognitive architectures. Cognitive architectures have build-in mechanisms, which can cope with new problems arriving in uncertain environments. In the paper three different cognitive architectures, implementations and simulation of them have been described and compared.