Autonomia organizacji oprogramowania

* Zastosuj definicję architektur Levisa do konceptualizacji oprogramowania w inteligentnym robocie i wyrażaj różne poziomy abstrakcji w architekturach operacyjnych, systemowych i technicznych.

* Nazwij warstwy (reaktywną, deliberatywną, interaktywną) w kanonicznej architekturze operacyjnej inteligentnego robota i opisz je pod kątem wartości pięciu wspólnych atrybutów (prymitywów, zdolności percepcyjnych, horyzontu planowania, skali czasu, wykorzystania modeli).

* Opisz cztery kategorie funkcjonalności deliberatywnej (generowanie, monitorowanie, wybieranie, wdrażanie) i wyjaśnij, czym te funkcje różnią się od funkcjonalności reaktywnej pod względem modelu świata, skal czasowych i horyzontu planowania.

* Zidentyfikuj architekturę systemów jako implementującą hierarchiczny, reaktywny lub hybrydowy paradygmat deliberatywno/reaktywny, jeśli weźmie się pod uwagę dwie wyróżniające cechy architektur systemów robotycznych: i) wzorzec relacji trzech prymitywów robota AI oraz ii) drogę wykrywania .

* Oznacz i zdefiniuj pięć podsystemów, które pojawiają się w większości architektur systemów: planowanie, kartograf, nawigacja, schemat motoryczny i percepcja.

* Powiąż operacje reaktywne i deliberatywne z każdym z pięciu podsystemów i wyjaśnij, w jaki sposób operacje będą wykorzystywać te podsystemy.

* Opisz podobieństwa i różnice w zatwierdzaniu wykonania i realizacji zadań oraz podaj dwa powody, dla których nie zapewniają one bezpiecznego lub poprawnego działania robota.

Przegląd

W poprzedniej części ustalono, że autonomia różni się koncepcyjnie od podejścia opartego na teorii sterowania - badacze sztucznej inteligencji postrzegają robotykę jako kwestię zwiększania zdolności adaptacyjnych (robienie „właściwej rzeczy" w otwartym świecie), podczas gdy wielu badaczy inżynierii postrzega robotykę jako rozszerzenie kontroli (tworzenie większy model świata i działania gwarancyjne). Dyskusja na temat tych różnic była ogólna i nie dała żadnych rzeczywistych wskazówek, jak zaprojektować inteligentnego robota. W tej części podjęto próbę udzielenia szerokiej odpowiedzi na pytania: Biorąc pod uwagę, że autonomia ma inny styl programowania, co to jest? oraz Czy inteligencję można dodawać warstwami, na przykład uaktualnianie do „wersji pro" lub pobieranie „aplikacji" w razie potrzeby? Rozpoczyna się od omówienia stylu programowania w kategoriach trzech typów architektur inżynierii oprogramowania, a następnie skupienia się na architekturach operacyjnych i systemowych, które dominują w programowaniu robotyki AI. Aby zrozumieć organizację oprogramowania, rozdział rozpoczyna się od zdefiniowania „architektury". Definicje Levisa dotyczące architektury operacyjnej, systemowej i technicznej są wykorzystywane do pomocy w omówieniu projektowania oprogramowania. Architektury operacyjne są analogiczne do opisywania domu pod kątem ważnych cech. Na przykład wykaz nieruchomości może opisywać dom pod względem stylu, na przykład ranczo, domek, posiadłość, przyczepa stacjonarna itp., lub funkcji, na przykład dwupoziomowy, jednorodzinny itp., co daje potencjalny nabywca wskazanie, czy będzie pasował do rodziny i stylu życia nabywcy. Architektury systemów oddają to, z czego składa się dom, na przykład trzy sypialnie, dwie łazienki, duża kuchnia, zadaszony garaż i basen. Architektury techniczne to rzeczywiste przykłady projektu, w których plan piętra domu jest deklaracją funkcji. Weź pod uwagę, że dom z trzema sypialniami i dwiema łazienkami może mieć różne konfiguracje pomieszczeń i sposoby ich łączenia, takie jak drugie piętro, liczba i zasięg korytarzy, wspólne łazienki itp. na. Architektury techniczne składają się również z algorytmów do implementacji funkcji, które są podobne do

rzeczywistych szczegółów dotyczących kadrowania, hydrauliki, malowania, kształtowania krajobrazu i dekorowania. Następnie przedstawiono kanoniczną architekturę operacyjną wykorzystywaną w sztucznej inteligencji, która grupuje inteligencję w trzy szerokie warstwy: reaktywną, deliberatywną i interaktywną. Ta kanoniczna architektura operacyjna będzie stanowić podstawę do zorganizowania reszty tekstu, z częścią II dotyczącą systemów i technicznych szczegółów architektonicznych warstwy reaktywnej, częścią III dotyczącą warstwy deliberatywnej, i częścią IV dotyczącą warstwy interaktywnej. Kanoniczna architektura operacyjna nie jest dobrze znana poza społecznością robotyki AI, dlatego opisuje trzy inne architektury operacyjne używane przez automatykę przemysłową, sterowanie i ogólną społeczność AI. Kanoniczna architektura operacyjna jest z definicji wysokopoziomowa i abstrakcyjna, dlatego opisano kolejną, bardziej namacalną architekturę, architekturę systemową. Istnieje pięć powszechnych podsystemów wykorzystywanych we wdrożeniach robotyki AI: Planowanie, Nawigacja, Kartograf, Schemat Motorowy, Percepcja. Odpowiadają one rodzajom pomieszczeń w domu. Istnieją trzy paradygmaty lub plany pięter, w których dominuje hybrydowa architektura systemów deliberatywno-reaktywnych, często skracana do architektury hybrydowej. Architektura hybrydowa będzie podstawą pozostałej części , która zagłębi się w podsystemy i algorytmy składające się na architekturę techniczną lub konkretny dom. Perspektywa architektoniczna podkreśla, że programowanie inteligentnego robota wymaga wielu komponentów sprzętowych i programowych, aby idealnie ze sobą współpracowały. Jeśli nie, robotowi grozi awaria. Jednym ze sposobów zmniejszania ryzyka jest włączenie człowieka do cyklu wykonania w celu sprawdzenia i zatwierdzenia tego, co robot zamierza zrobić, zanim to zrobi. Jak zostanie omówione, badania nad czynnikami ludzkimi sugerują, że aprobata człowieka w rzeczywistości nie zapewnia lepszej wydajności robota. Patrząc w przyszłość, następna część obejmie telesystemy, które są często uważane za substytuty projektowania i budowy „kompletnego" systemu robotyki AI lub „w pełni autonomicznego" robota. To zakończy przegląd robotyki AI w Big Picture. W pozostałej części omówione zostaną różne algorytmy i metody programowania dla różnych modułów oprogramowania, które składają się na widok architektury technicznej, podobnie jak rejestrowanie szczegółów instalacji wodno-kanalizacyjnych, ram i krajobrazu dla konkretnego domu. Dzięki tej części można rozpocząć odgórne projektowanie robota AI.

Trzy typy architektur oprogramowania

Termin „architektura" jest często używany w robotyce w odniesieniu do ogólnego stylu projektowania lub organizacji. Arkin w swojej książce „Behavior-Based Robots" podaje kilka definicji. Dwie z definicji, które cytuje od innych badaczy, oddają sposób, w jaki termin ten będzie tutaj używany. Zgodnie z Mataric, architektura zapewnia zasadniczy sposób organizacji systemu sterowania. Jednak oprócz zapewniania struktury, architektura nakłada ograniczenia na sposób rozwiązania problemu sterowania. Za Deanem i Wellmanem, architektura opisuje zestaw elementów architektonicznych i ich interakcje. My jesteśmy zainteresowani wspólnym rdzeniem komponentów w architekturach robotów, ponieważ są to podstawowe elementy do programowania robota, a także zasadami i regułami łączenia tych komponentów ze sobą. Aby zobaczyć, jak ważna jest architektura, rozważ budowę domu lub samochodu. Nie ma „właściwego" projektu domu, chociaż większość domów ma te same elementy (kuchnie, łazienki, ściany, podłogi, drzwi itp.) oraz infrastrukturę (elektryczność, hydraulika, ogrzewanie i klimatyzacja itp.). Jednak wielkość pomieszczeń, wyrafinowanie infrastruktury, a także rozkład pomieszczeń wpływają na to, czy jest to „właściwe" dla kupującego.

Rodzaje architektury

Architektury przekuwają wcześniejsze doświadczenia w szablony do tworzenia nowych inteligentnych robotów. Termin „architektura" był używany w Siłach Powietrznych Stanów Zjednoczonych tak często

w różnych znaczeniach, że w 2001 r. główny naukowiec Alex Levis sformalizował taksonomię architektur: operacyjną, systemową i techniczną. Zostały one opisane poniżej.

ARCHITEKTURA OPERACYJNA

architektura operacyjna: opisuje, co robi system lub jego funkcjonalność na wysokim poziomie, ale nie opisuje, jak to robi. Ten opis jest podobny do wykazu nieruchomości dla nowego domu opisującego dom w powszechnie używanych terminach, takich jak styl domu (np. bungalow, kolonialny, ranczo itp.) oraz liczba sypialni i łazienek. Architektura operacyjna określa, w jaki sposób siedem odrębnych obszarów sztucznej inteligencji, każdy z własnymi algorytmami i strukturami danych, jest połączonych razem i może być użyty do określenia, czy projekt zapewnia zamierzoną funkcjonalność.

ARCHITEKTURA SYSTEMÓW

architektura systemów: opisuje, w jaki sposób system jest rozłożony na główne podsystemy. Ten opis jest podobny do planu domu. Architektura systemu określa konkretne systemy oprogramowania (pomieszczenia), które zapewniają pożądaną inteligentną funkcjonalność i opisuje sposób ich połączenia (korytarze? otwarty plan piętra? piętro lub piwnica?). Architekturę systemu można wykorzystać do określenia, czy projekt spełnia dobre zasady inżynierii oprogramowania, zwłaszcza modułowość i rozszerzalność.

ARCHITEKTURA TECHNICZNA

architektura techniczna: opisuje szczegóły implementacji systemu. Jest to podobne do opisywania, w jaki sposób dom jest faktycznie zbudowany, gdzie budowniczy umieszcza rury, jakie przepisy budowlane mają zastosowanie do dachu lub fundamentu, jakie materiały zostaną użyte, jak drogie będą oprawy oświetleniowe i szafki i tak dalej. Architektura techniczna określa algorytmy, a nawet języki, w których są zaprogramowane moduły, i może być używana do określenia, czy projekt korzysta z najbardziej odpowiednich algorytmów. Architektury techniczne podlegają ciągłej ewolucji, odzwierciedlając postęp w sztucznej inteligencji.

Architektury wzmacniają dobre zasady inżynierii oprogramowania

Sztuczna inteligencja jest przedsiębiorstwem programistycznym, a dobra inżynieria oprogramowania jest niezbędna do pomyślnego przedsiębiorstwa programistycznego. Myślenie o projekcie autonomicznego robota w kategoriach trzech architektur zachęca do tworzenia projektów spełniających cztery ogólne zasady inżynierii oprogramowania. Jednym z nich jest abstrakcja, gdzie portale architektury operacyjnej i systemowej pozwalają projektantom ignorować szczegóły, aby skupić się na myśleniu o ogólnej organizacji inteligencji. Kolejną zasadą jest modułowość, gdzie systemy i architektury techniczne zachęcają projektanta do myślenia w kategoriach programowania obiektowego. Moduły powinny charakteryzować się wysoką spójnością, gdy każdy moduł lub obiekt wykonuje jedną rzecz dobrze, a niepowiązane ze sobą funkcje są umieszczone gdzie indziej, oraz niskim sprzężeniem, gdy moduły lub obiekty są niezależne. Testowanie i debugowanie jednostek pomocniczych o wysokiej spójności i niskim sprzężeniu. Trzecią zasadą jest przewidywanie zmian z przyrostowością, ponieważ systemy i architektury techniczne zapewniają projektantowi wgląd w to, czy kod został zaprojektowany tak, aby wspierać aktualizację algorytmu do nowszego i dodawanie nowych możliwości bez konieczności przepisywania kodu dla istniejącego moduły i ewentualnie wprowadzić błędy. Ta zasada jest związana z modułowością. Czwartą zasadą jest zasada ogólności. Architektury operacyjne, systemowe i techniczne zapewniają projektantowi ramy do określenia, czy podstawowa organizacja, podsystemy i implementacja pozwolą na użycie kodu w innych aplikacjach. W robotyce ta zasada jest często nazywana przenośnością. Intencją tego rozdziału jest dostarczenie

szablonów dla architektur operacyjnych i systemowych, aby nie trzeba było ich za każdym razem wymyślać na nowo.

CELOWALNOŚĆ DO NISZ

KRZEPKOŚĆ

Oprócz czterech ogólnych zasad inżynierii oprogramowania, Arkin dodaje jeszcze dwie zasady inżynierii oprogramowania dla robotyki AI: możliwość kierowania do niszy i niezawodność. Możliwość celowania w niszę pokazuje, jak dobrze robot działa w zamierzonym zastosowaniu. Należy pamiętać, że możliwość kierowania do nisz i łatwość przenoszenia często są ze sobą sprzeczne. W tym miejscu architektura techniczna może wymagać algorytmów, które są wysoce specyficzne dla konkretnej aplikacji. Na przykład robot może być zaprojektowany do pracy tylko w ciągu dnia. Ale jeśli architektura techniczna była dobrze zorganizowana w podsystemy w architekturze systemów, algorytmy specyficzne dla światła dziennego można łatwo zastąpić algorytmami, które działały w nocy. Solidność określa, gdzie system jest podatny i jak konstrukcja systemu samoistnie zmniejsza tę podatność. Ta luka może pojawić się na poziomie abstrakcji architektury operacyjnej lub systemowej, gdzie projektant może sprawdzić, czy istnieje podsystem monitorujący wykonanie. Może też pojawić się poprzez architekturę techniczną służącą jako portal, przez który projektant może zbadać, jak kod obsługuje wyjątki i warunki awarii.

Kanoniczna architektura operacyjna robotyki AI

TRZY WARSTWY

Po serii „wojen paradygmatów" w latach 1967 (Shakey) i 2000 robotyka AI przeszła na hybrydową, deliberatywno-reakcyjną architekturę operacyjną. Pod względem organizacyjnym architektura operacyjna składa się z trzech warstw: reaktywnej, deliberatywnej i interaktywnej, zgodnie z metaforą inspirowaną biologią. Można myśleć, że inteligencja biologiczna składa się z czterech głównych funkcji: reakcji, deliberacji, konwersji sygnałów na symbole, która łączy reakcję i deliberację oraz interakcję robota z innymi czynnikami zewnętrznymi. Ta abstrakcja inteligencji biologicznej prowadzi do trzech różnych celów i stylów komputerowych. Te trzy warstwy nie tylko zawierają różne filozoficznie kategorie funkcjonalności, ale mają też pięć atrybutów, które wpływają na informatykę: elementy podstawowe, zdolności percepcyjne, horyzont planowania, skala czasowa i wykorzystanie modeli. Jak zostanie omówione bardziej szczegółowo później, warstwy mogą wykorzystywać znacząco różne struktury i języki programowania. W tej sekcji najpierw przedstawiono pięć atrybutów opisujących warstwy, omówiono każdą warstwę indywidualnie, tworząc diagram podsumowujący.

Atrybuty do opisywania warstw

Opis i uzasadnienie tego, jakie funkcje oprogramowania trafiają do każdej warstwy, opierają się na pięciu atrybutach: elementach podstawowych, zdolności percepcyjnej, horyzoncie planowania, skali czasu i wykorzystaniu modeli.

CZTERY ELEMENTY PODSATAWOWE

Elementy podstawowe. Podobnie jak w przypadku funkcji w architekturach LOA i ACL, robotyka AI postrzega autonomiczne możliwości jako składające się z czterech podstawowych elementów: SENSE, PLAN, ACT i LEARN. Jeśli funkcja pobiera informacje z czujników robota i generuje dane wyjściowe przydatne dla innych funkcji, to ta funkcja należy do kategorii SENSE. Jeśli funkcja pobiera informacje (albo z czujników, albo z własnej wiedzy o tym, jak działa świat) i produkuje jedno lub więcej zadań do wykonania przez robota (idź korytarzem, skręć w lewo, przejdź trzy metry i zatrzymaj się), ta funkcja

znajduje się w kategorii PLAN. Funkcje, które wytwarzają polecenia wyjściowe do siłowników silnikowych, mieszczą się w ACT (obrót 98°, zgodnie z ruchem wskazówek zegara, z prędkością obrotu 0,2 m/s). Inteligencja jest ogólnie postrzegana jako proces SENSE, PLAN i ACT, podobny do pętli OODA. LEARN jest ważnym mechanizmem, dzięki któremu agent maksymalizuje swoje szanse na sukces, ale wykracza poza pozostałe trzy prymitywne; agent może nauczyć się lepiej wyczuwać lub planować, zdobywać więcej działań lub umiejętności, a nawet uczyć się dla konkretnej misji, co wyczuwać, co planować i jak działać. Podobnie jak w przypadku ACL, bardziej inteligentne systemy mają coraz bardziej wyrafinowane egzemplarze jednego lub więcej z tych czterech elementów podstawowych.

ZDOLNOŚĆ PERCEPCYJNA

Zdolność percepcyjna. Robotyka AI postrzega postrzeganie potrzebne dla określonej zdolności jako bezpośrednie lub wymagające rozpoznania. Dzieli to zdolności na te, które mogą pracować bezpośrednio z bodźcami przychodzącymi oraz te, które wymagają przekształcenia bodźca w symbol.

HORYZONT PLANOWANIA

Horyzont planowania. Robotyka AI postrzega zdolność jako posiadającą horyzont planowania albo z obecnych; przeszłość, teraźniejszość; lub przyszłość, przeszłość, teraźniejszość. Na przykład warstwa reaktywna dotyczyła tylko bodźców z teraźniejszości, podczas gdy zdolności deliberacyjne wymagają rozumowania na temat przeszłości lub przewidywania, jak świat będzie wyglądał w przyszłości w wyniku działania. Horyzont planowania ogranicza wybór struktur danych (np. jeśli robot wnioskuje o przeszłości lub przyszłości, to musi przechowywać stan przeszły i obecny), a także algorytmów (np. jeśli algorytmy działają tylko w teraźniejszości). , robot nie może używać metod rozumowania, które rzutują przyszłe konsekwencje).

SKALA CZASU

Skala czasu. Robotyka AI uwzględnia również skalę czasową funkcji — czy musi być bardzo szybka (odruch), szybka (wybór działań), czy może być stosunkowo wolna (rozumowanie problemu). Skala czasu pomaga określić asynchroniczne działanie składników oprogramowania. Jedna architektura techniczna, RCS5, określa skale czasowe dla różnych funkcji, tworząc warstwy funkcji wykonywanych w podobnych skalach czasowych. Model świata. Robotyka AI dzieli możliwości na te, które wykorzystują informacje zebrane tylko dla tej funkcji, zwane lokalnym modelem świata, oraz te, które opierają się na wszystkich informacjach z czujników, które mają być najpierw przetworzone w globalny model świata. Podział ten pomaga określić stopień zaawansowania zdolności.

Warstwa reaktywna

Funkcjonalność reaktywna, zwana także behawioralną, odpowiada pętli reaktywnej w ośrodkowym układzie nerwowym. Ta pętla jest związana z funkcjonalnością, która występuje w rdzeniu kręgowym i „dolnej części mózgu", w szczególności odpowiedziami i umiejętnościami opartymi na pamięci motorycznej. Odruchy mimowolne, takie jak odruch kolanowy na uderzenie w rzepkę i wyuczone umiejętności, takie jak podnoszenie filiżanki kawy lub jazda na rowerze, pozwalają agentowi na szybkie wykonywanie działań. Odpowiedzi i umiejętności to wzorce działania ogólnie określane jako zachowania. Funkcjonalność behawioralna jest wystarczająca dla inteligencji zwierząt, takiej jak ryby i owady. W robotyce warstwa reaktywna składa się z funkcjonalności zbudowanej z elementów podstawowych SENSE i ACT bez PLAN. Ta warstwa odpowiada inteligencji zwierząt, gdzie działanie jest generowane bezpośrednio przez wyczuwane lub wewnętrzne bodźce i nie ma świadomości większego świata ani monitorowania misji. SENSE i ACT są ściśle powiązane w konstrukty zwane zachowaniami,

które wytwarzają zdolności. Jedno lub więcej zachowań jest wyzwalanych przez bodźce zewnętrzne lub wewnętrzne, a ogólne zachowanie wyłania się w oparciu o mechanizm łączenia wyników każdego zachowania. Zdolność percepcyjna zachowań nazywana jest percepcją bezpośrednią, co zostanie opisane w dalszych rozdziałach. Chociaż zachowanie nie tworzy i nie utrzymuje globalnego modelu świata, zachowanie może mieć lokalny model świata, który służy jako pamięć krótkotrwała. Nie ma planowania, a zatem horyzont planowania jest teraźniejszością. Skala czasowa funkcji jest bardzo szybka; Zachowania bodziec-odpowiedź mają zwykle szybki cykl aktualizacji, rzędu 15–30 cykli na sekundę, odpowiadający potrzebom sterowania i szybkości aktualizacji czujnika.

Warstwa deliberatywna

Funkcjonalność deliberatywna odpowiada pętli poznawczej związanej z korą mózgową. Mózg niezależnie przyjmuje jako dane wejściowe te same sygnały, które są wykorzystywane przez dolny ośrodkowy układ nerwowy i dodaje dodatkowe przetwarzanie czujników, aby podejmować świadome decyzje. Pętla poznawcza może modyfikować pętlę reaktywną, aby tworzyć instancje nowych działań, które zostałyby wykonane przez funkcję reaktywną (np. wchodzenie po schodach do budynku) lub je modyfikować (chodź wolniej niż normalnie, ponieważ na powierzchni może znajdować się niewidoczny lód). schody). Funkcjonalność deliberatywna jest wystarczająca dla bardziej wyrafinowanych aspektów rozwiązywania problemów i rozumowania inteligencji, a także planowania. Na przykład pętla deliberatywna polega na wnioskowaniu o tym, gdzie znaleźć filiżankę kawy i aktywuje pętlę reaktywną, aby przenieść robota do kuchni, a następnie podnieść filiżankę z kawą. Te dwie pętle różnią się szybkością przetwarzania i zawartością tego, co jest przetwarzane. Odruchy i zdolności motoryczne są bardzo szybkie, podczas gdy rozważanie problemu może być powolne; to może wyjaśniać, dlaczego pętle są asynchroniczne. Pracownik biurowy może złapać spadającą filiżankę kawy, a jednocześnie zastanawiać się, jak zbilansować konto finansowe. Funkcjonalność reaktywna wydaje się opierać na sygnałach, to jest bezpośrednich bodźcach sensorycznych, podczas gdy deliberatywna funkcyjność wydaje się opierać na symbolach, to znaczy bodźcach, które zostały zsyntetyzowane, oznaczone i połączone z pamięcią i wiedzą. Jak mózg przekształca sygnały w symbole, jak rozpoznaje obiekt, któremu powinien zostać przypisany symbol (np. filiżanka kawy) lub jak przypisuje temu obiektowi etykietę semantyczną (np. moja filiżanka kawy), nie jest dobrze zrozumiane. W AI określa się to jako rozpoznawanie obiektów i stanowi problem uziemienia symbolu. Warstwa deliberatywna obsługuje działania PLAN robota, ma Plannera, który generuje plan przy użyciu Modelu Świata, a następnie tworzy instancje odpowiednich zachowań SENSE-ACT w warstwie reaktywnej w celu wykonania planu. Zachowania te działają do momentu osiągnięcia następnego kroku w planie i utworzenia instancji nowego zestawu zachowań lub wykrycia problemu z planem przez monitorowanie deliberatywne. Zdolność percepcyjna zachowań to rozpoznawanie, w którym robot buduje globalne lub trwałe modele świata. Horyzont planowania wykorzystuje informacje z przeszłości i teraźniejszości do prognozowania konsekwencji planu w przyszłości. Czas potrzebny na planowanie jest wolniejszy niż na reakcję, a funkcje w warstwie deliberatywnej mogą być aktualizowane z częstotliwością rzędu od 15 cykli na sekundę w celu aktualizacji Modelu Świata do kilku minut w celu skonstruowania złożonych planów. Nawet jeśli algorytm planowania działa prawie tak szybko, jak zachowania w warstwie reaktywnej, ciągłe ponowne planowanie i przywracanie zachowań może nie przynieść korzyści.

CZTERY FUNKCJE DELIBERATYWNE

Warstwa deliberatywna jest podzielona na dwie podwarstwy połączone Modelem Świata, które zawierają cztery funkcje deliberatywne:

GENEROWANIE

* generowanie planów, które odpowiadają planowaniu, rozumowaniu i rozwiązywaniu problemów w AI

WYBIERANIE

* wybór określonych zasobów do realizacji planu, co odpowiada planowaniu, alokacji zasobów i reprezentacji wiedzy o możliwościach w AI

REALIZOWANIE

* realizacja planu, co odpowiada realizacji

MONITOROWANIE

* monitorowanie realizacji planu w celu ustalenia, czy osiąga cel, uczenie się, co jest normalne i przewidywanie potencjalnych niepowodzeń, co odpowiada planowaniu i rozumowaniu w AI.

Górna warstwa rozważań zawiera funkcje generowania i monitorowania misji, które działają w przeszłych, obecnych i przyszłych horyzontach czasowych. Niższa warstwa deliberatywna zawiera funkcje wyboru i implementacji, które przekształcają plan w zachowania. Niższa podwarstwa działa w przeszłych i teraźniejszych horyzontach czasowych. Podwarstwy działają asynchronicznie. Model świata jest ograniczony, reprezentując to, co jest ważne na świecie dla ogólnych zamierzonych możliwości robota. Na przykład wiele robotów tworzy i utrzymuje mapę 2D lub 3D fizycznego układu świata na potrzeby autonomicznej nawigacji, ale nie uwzględnia zmian w czasie, aby wesprzeć zrozumienie semantyczne potrzebne do monitorowania planu (np. „Dlaczego droga jest nagle zablokowana? Czy zaraz wpadnę w zasadzkę?"). Funkcje w warstwie deliberatywnej są często programowane w języku funkcjonalnym, takim jak Lisp, co upraszcza planowanie i rozumowanie.

Warstwa interaktywna

OSOBA

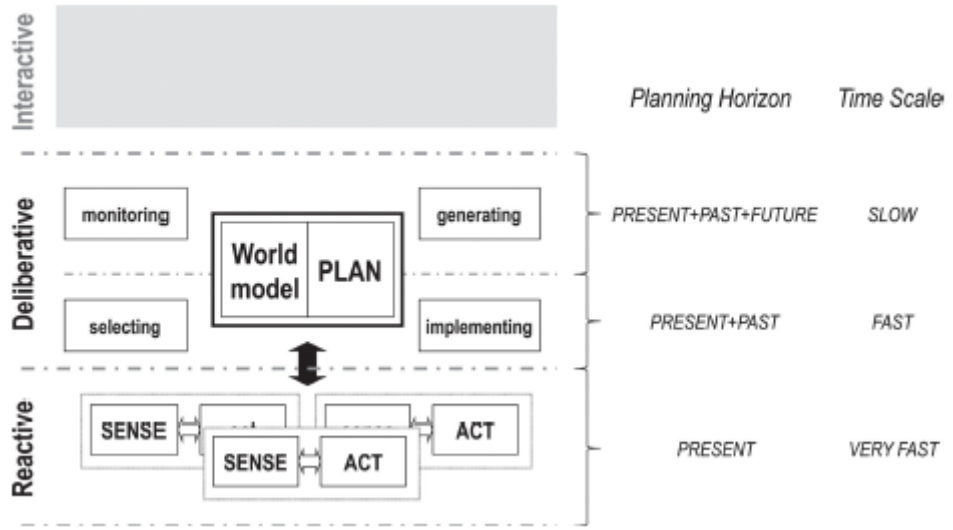
ZACHOWANIA ROJOWE

Interaktywna funkcjonalność jest potrzebna do interakcji z innymi agentami, ludźmi, innymi robotami lub agentami oprogramowania. Interakcja może być regulowana przez pętle reaktywne lub poznawcze. Reakcje wewnętrzne na sytuacje lub na innych mogą generować negatywne emocje, takie jak strach lub złość, które mogą objawiać się zmianami postawy i gestów (szybsze i bardziej ekstremalne ruchy), bliskością innych podmiotów (zbliż się, aby grozić) oraz komunikacją werbalną (dobór słów i mówienie głośniej); jest to czasami określane jako inteligencja społeczna i zostanie opisane w Części 18. Interakcje można również uregulować w sposób przemyślany, na przykład w przypadku decyzji dotyczących bezpieczeństwa i prywatności. Wyświetlacze interaktywnej funkcjonalności zasadniczo otaczają osobę i tworzą jej osobowość. Jednak podczas pracy w zespołach, jak opisano w Części 17, mogą pojawić się interakcje. Interakcje te obejmują zachowania roju, w których kolonie mrówek znajdują pożywienie, stada ptaków i ławice ryb, lub, w przypadku bardziej wyrafinowanych zespołów, gdzie członkowie wyraźnie negocjują, które części zadania wezmą odpowiedzialność za. Warstwa interaktywna to zasadniczo opakowanie na indywidualne programowanie robota, które umożliwia robotowi pracę z innymi agentami — ludźmi, robotami lub agentami oprogramowania. Funkcjonalność w warstwie interaktywnej jest tematem aktywnych badań nad tym, jak roboty komunikują się i pracują ze sobą i z ludźmi, jak zachowują bezpieczeństwo i prywatność oraz jakie są dobre interfejsy użytkownika. Brak jest danych dotyczących wewnętrznej organizacji warstwy interaktywnej, ponieważ obecnie nie ma jasnego wzorca atrybutów warstwy. Na przykład język naturalny i inne formy komunikacji mogą być tworzone w sposób reaktywny lub deliberatywny. Podobnie emocje, które dostarczają informacji

zwrotnej na temat stanu robota, mogą być generowane reaktywnie lub deliberatywnie, w zależności od okoliczności. Warstwa interaktywna zaczęła pojawiać się jako temat zainteresowania około 2006 r., inspirowana badaniami, w jaki sposób zespoły robotów mogą wyraźnie współpracować ze sobą oraz badaniami interakcji człowiek-robot w konkretnych zastosowaniach robotów społecznych do rozrywki, pomocy i zdrowia opieka. Warstwa Interaktywna może być programowana w językach proceduralnych, funkcjonalnych lub ontologicznych, takich jak OWL.

Kanoniczny schemat architektury operacyjnej

Kanoniczną architekturę operacyjną można przedstawić tak, jak pokazano na rysunku 4.5, który pokazuje organizację prymitywów, zdolność percepcyjną, horyzont planowania, skalę czasową i wykorzystanie modeli w warstwach reaktywnej i deliberatywnej. Warstwy odpowiadają pętlom przetwarzania, a Model Świata zajmuje się konwersją sygnałów na symbole.



Inne architektury operacyjne

Kanoniczna architektura organizacyjna preferowana przez społeczność robotyczną AI opiera się na metaforze biologicznej, ale istnieją co najmniej trzy inne odrębne style architektur operacyjnych zapewniających różne, prowokujące do myślenia sposoby myślenia o organizowaniu inteligencji. Społeczność automatyzacji stworzyła taksonomie określające, w jaki sposób funkcje mogą być przypisane do maszyny lub człowieka, co prowadzi do poziomów automatyzacji. Jednak te poziomy były traktowane przez niektórych programistów jako architektura operacyjna. Społeczność kontrolerów, zwłaszcza w lotnictwie i kosmonautyce, próbowała kategoryzować rosnące poziomy zrozumienia i kontroli nad sytuacją, tworząc wariant poziomów automatyzacji zwany poziomami autonomicznej kontroli. Społeczność wieloagentowa w ramach sztucznej inteligencji zaczęła myśleć o autonomii w kategoriach rosnącego poziomu zrozumienia i kontroli nad sytuacją, a dokładniej jako ilości inicjatywy, jaką robot może wykazać, ujmowanej jako poziomy inicjatywy. W tej sekcji omówiono te inne architektury i omówiono ich związek z kanoniczną architekturą robotyki AI. Poziomy automatyzacji i ramy poziomów autonomicznej kontroli pojawiają się w literaturze robotyki, więc szczególnie pomocne jest ich zrozumienie i ich ograniczenia dla organizacji oprogramowania AI.

Poziomy automatyzacji

POZIOMY AUTOMATYZACJI (LOA)

POZIOMY AUTONOMII

Podczas gdy inteligencja biologiczna zapewnia wgląd w ogólną organizację inteligencji, społeczność systemów człowiek-maszyna oferuje inny punkt widzenia. Społeczność zajmująca się systemami człowiek-maszyna przyjrzała się automatyzacji eksploracji kosmosu, kontroli zakładów przetwórczych i autopilotom w samolotach. Począwszy od przełomowej pracy Thomasa Sheridana, społeczność tradycyjnie określała stan automatyzacji w dowolnym momencie w kategoriach funkcji, które człowiek obecnie deleguje na komputer. Ta klasyfikacja stanu w systemie naturalnie prowadzi do hierarchii, w których maszyna odpowiada za więcej funkcji, a zatem byłaby uważana za bardziej autonomiczną. Te hierarchie taksonomiczne są często używane jako architektury operacyjne lub jako de facto miary automatyzacji systemu, gdzie poziom B jest bardziej zautomatyzowany niż poziom A. Hierarchie są ogólnie określane jako poziomy automatyzacji lub poziomy autonomii (LOA), jako automatyzacja i autonomia są często używane jako synonimy. Książka skupi się na ogólnych tematach wspólnych dla wariantów poziomów automatyzacji. Powszechnym sposobem wyrażania stanu automatyzacji w systemie jest podzielenie możliwości funkcjonalnych na cztery szerokie kategorie.

WSPÓLNA KONTROLA

KONTROLA WYMIANY

Funkcję może pełnić człowiek, komputer, oba współpracujące ze sobą (nazywane wspólną kontrolą), lub części mogą być wykonywane przez człowieka lub komputer, a pozostałe części przez drugiego agenta (nazywane kontrolą w obrocie). . Stan automatyzacji procesu zależy od tego, czy funkcję wykonuje człowiek, czy maszyna. Żaden konkretny poziom automatyzacji nie jest akceptowany przez wszystkich badaczy, a różni eksperci kłócą się o funkcje i kolejność hierarchii. Jednak różne warianty poziomów automatyzacji mają tę samą zasadę, że istnieje zestaw funkcji, a im więcej funkcji delegowanych do komputera, oznacza wyższą automatyzację. Wynikające z tego poziomy podejścia do automatyzacji są często wykorzystywane jako operacyjne architektury, z oczekiwaniem, że inteligentny robot będzie budowany przyrostowo w warstwach odpowiadających wierszom w tabeli. Niezamierzoną konsekwencją stosowania poziomów automatyzacji jako architektury operacyjnej jest to, że zakłada ona, że celem projektu jest pełna automatyzacja, a nie dopasowanie odpowiedniego poziomu automatyzacji do misji i możliwości robota. Główną zaletą korzystania z poziomów organizacji automatyki jest to, że cztery funkcje i poziomy oferują bardziej specyficzną modułowość niż trzy warstwy inteligencji biologicznej. Taksonomia potwierdza, że misja może wymagać współpracy zarówno robotów, jak i ludzi. Sugeruje, że można dodać do robota funkcjonalność, aby zmniejszyć jego zależność od ludzi, zasadniczo, że pojedynczy robot może z czasem zostać zmodernizowany z nieinteligentnej, całkowicie zależnej od ludzkiego operatora formy automatyzacji do inteligentnej, całkowicie niezależnej od człowieka. Formularz. Główną wadą tej organizacji jest to, że poziomy autonomii początkowo miały stanowić zbiór definicji do oznaczania aktualnego stanu aktywnej zdolności, a nie operacyjną architekturę do koordynowania wszystkich zdolności. Jeśli taksonomia zostanie zastosowana jako oznaczenie inteligencji dla całego systemu, może to zakłócać możliwości projektowe robota, które zmieniają się dynamicznie podczas misji. Na przykład bezzałogowy statek powietrzny może mieć zdolność autonomicznego startu, która nie wymaga udziału człowieka-pilota, jednak później w misji może być potrzebny człowiek-pilot do rozpoznawania celów. W przypadku pierwszej zdolności robot odpowiada za wszystkie funkcje, podczas gdy w przypadku drugiej funkcje są wspólne dla robota i człowieka. Również stan może zmieniać się dynamicznie na skutek awarii wymagającej ingerencji człowieka, zmniejszając w ten sposób liczbę funkcji, za które odpowiada robot. Jednak projektanci często będą odnosić się do tego, że cały robot znajduje się na poziomie najwyższego stanu automatyzacji, a nie najniższego. Ta niefortunna sytuacja powoduje, że użytkownicy mają nierealistyczne oczekiwania co do ogólnego zaawansowania robota i pozwala projektantom ignorować

sposoby wspierania wzajemnego oddziaływania między poziomami. Istnieją dwie inne wady. Jednym z nich jest to, że traktowanie taksonomii poziomów autonomii jako architektury operacyjnej skutkuje powstaniem systemu, który uwzględnia tylko cztery funkcje deliberatywne, ignorując zachowania reaktywne. Inną wadą jest to, że taksonomia poziomów autonomii uwzględnia sposób interakcji z ludzkim przełożonym, ale nie uwzględnia interakcji z innymi agentami ani interakcji nienadzorczych.

Poziomy autonomicznej kontroli (ACL)

<b>AUTONOMICZNE POZIOMY KONTROLI (ACL)

Architektura operacyjna autonomicznych poziomów kontroli (ACL) jest czasami wykorzystywana przez Departament Obrony USA jako mapa drogowa rozwoju systemów bezzałogowych. Opiera się ona na pętli OODA wymawianej „ewe-dah", którą John Boyd, słynny pilot sił powietrznych i strateg, używany do skodyfikowania sposobu, w jaki odnoszący sukcesy pilot podejmuje decyzje. Boyd postawił hipotezę, że pilot przechodzi przez ciągły cykl obserwacji, orientacji, decydowania i działania (OODA), aby ukończyć misję. Podobnie jak LOA, ACL dzieli autonomię na funkcje i poziomy, ale w przeciwieństwie do LOA, poziomy nie uwzględniają wyraźnie delegowania człowiek/maszyna. ACL ma dziesięć poziomów odzwierciedlających ranking ad hoc, który łączy zarówno rosnące zaawansowanie tego, które części każdej fazy pętli OODA są wykonywane autonomicznie przez bezzałogowy system powietrzny, jak i ich wartość dla kategorii misji wojskowych. Każdy poziom można podzielić na trzy funkcje: percepcja/świadomość sytuacyjna, analiza/podejmowanie decyzji oraz komunikacja/współpraca. Poziomy to:

9 Inteligencja roju w przestrzeni bitewnej. Grupa robotów może przypisywać sobie role i stosować taktykę rozproszoną w aktualnej sytuacji.

8 Znajomość przestrzeni bitewnej. Robot rozumie lokalizacje i trajektorie innych agentów i potrafi oportunistycznie wybierać cele.

7 Wiedza o przestrzeni bitewnej. Robot lub grupa robotów może wykrywać i projektować działania wroga.

6 Współpraca wielu pojazdów w czasie rzeczywistym. Grupa robotów współpracuje i optymalizuje swoje działania grupowe, aby osiągnąć cele taktyczne.

5 Koordynacja wielu pojazdów w czasie rzeczywistym. Grupa robotów pracuje blisko siebie, realizując plan taktyczny pod nadzorem człowieka.

4 Pojazd adaptacyjny do usterek/zdarzeń. Robot otrzymuje plan misji i zasady zaangażowania (tj. zasady) i może dostosowywać plan misji taktycznej do działań wroga lub awarii systemu.

3 Solidna reakcja na usterki/zdarzenia w czasie rzeczywistym. Robot monitoruje swój stan zdrowia, wykrywa problemy i decyduje, czy dostosować plan, czy przerwać misję.

2 zmienna misja. Robot otrzymuje plan misji taktycznej i może dostosować plan w granicach.

1 Wykonaj wcześniej zaplanowaną misję. Robot wykonuje misję w środowisku z dala od innych agentów.

0 Pojazd zdalnie sterowany. Człowiek jest zasadniczo odpowiedzialny za wszystko oprócz stabilności lotu.

Podstawowymi zaletami ACL jest to, że obejmuje szeroki zakres misji wojskowych i wiąże funkcjonalność związaną z inteligencją ze znaną formułą pętli OODA. Te trzy funkcje mają coraz większy

poziom zaawansowania; na przykład nie ma funkcji analizy/podejmowania decyzji robota na poziomie 0, ale na poziomie 3 funkcja analizy/podejmowania decyzji obejmuje identyfikację, wykrywanie i naprawę błędów. Drugą zaletą jest to, że ACL uwzględnia zespoły robotów, chociaż nie uwzględnia interakcji robotów pracujących z ludźmi. Wyższe poziomy współdzielą funkcjonalność z aplikacjami rozrywkowymi, w szczególności rozumieją, gdzie znajdują się inni agenci i wnioskują o ich zamiarach. Wady ACL są podobne do tych z LOA, z dwoma dodatkowymi wadami. Jednym z nich jest to, że uzasadnienie rankingu poziomów jest niejasne i może opóźnić zastosowanie robotów do misji, ponieważ poziomy implikują warunki wstępne dotyczące zaawansowania wymaganej inteligencji. Rozważ na przykład rój do zastosowań, takich jak humanitarne rozminowywanie obszaru min lądowych lub pobieranie próbek jakości powietrza lub wody. Wiele zwierząt roi się. Osy mogą gromadzić się i koordynować działania, aby zaatakować intruza, co odpowiadałoby poziomom 5, 6 i 9. Ale również giną, wielokrotnie próbując wyjść przez zamknięte okno, ponieważ osa nie monitoruje swojego postępu, co wymagałoby funkcjonalności, która ACL umieszcza się na poziomie 3 i 4. Biologia sugeruje, że rój agentów może odnieść sukces przy mniej wyrafinowanej inteligencji, podczas gdy ACL uważa, że rój wymaga większej inteligencji niż monitorowanie zadań. Ta różnica między biologią a ACL jest ważna, ponieważ w ramach ACL rój nie zostałby zaprojektowany i wdrożony, dopóki technologia nie rozwinie się do punktu wykonywania misji monitorowania. Jednak mogą istnieć misje, w których setki tanich robotów będą w stanie ukończyć misję bez monitorowania misji, nawet jeśli duży procent z nich zawiedzie. Drugim problemem jest to, że lista ACL nie wydaje się odpowiednia do zastosowań wspomagających, ponieważ nie ma poziomów ani funkcji, które sugerują pomaganie, a nie zastępowanie pilota lub osób znajdujących się w pobliżu robota

Poziomy inicjatywy

Inny styl architektury operacyjnej opiera się na ilości inicjatywy, jaką daje robotowi do wykonania zadania. W tym przypadku autonomia opiera się na politycznej konotacji autonomii jako samorządności, a nie na konotacji mechanicznej kontroli omówionej w rozdziale 3.3. W przeciwieństwie do wcześniej omówionych architektur operacyjnych, inicjatywa jest konceptualizowana na podstawie ról, a nie funkcji.

POZIOMY INICJATYWY

Colman i Han stanowią jeden przykład architektury operacyjnej poziomu inicjatywy. W swojej pracy zastanawiają się, jak robot może być inteligentnym piłkarzem. Robot miałby co najmniej dwie role, jedną, powiedzmy, napastnika, a drugą, by być dobrym kolegą z drużyny. Na najniższym poziomie robot grający w piłkę nożną z minimalną inteligencją byłby w stanie wykorzystać swoją percepcję i umiejętności do wypełnienia swojej roli, ale nic więcej; ogólnie rzecz biorąc, robot nie miałby żadnej inicjatywy w celu zmiany strategii. Na wyższym poziomie robot-pomocnik byłby w stanie zrozumieć, że kolega z drużyny pełniący rolę obrońcy jest kontuzjowany i spontanicznie zmienić swoją rolę, aby objąć zarówno role pomocnika, jak i obrońcę. Pięć poziomów inicjatywy to:

* Brak autonomii. Robot podąża za sztywnym programowaniem, aby wykonać zadanie lub osiągnąć cel. Robot może zawsze osłaniać przeciwnika w ten sam sposób lub zawsze podawać piłkę innemu graczowi. Ten poziom jest podobny do zachowań bodziec-odpowiedź w architekturze biologicznej i 10 poziomów na poziomach architektury automatyzacji.

* Autonomia procesu. Robot może wybrać algorytm lub proces, aby osiągnąć swoje cele zadania. Robot może przyjąć inną strategię, aby objąć różnych graczy. Ten wybór jest podobny do wyboru funkcji na poziomach architektury automatyki.

* Autonomia systemowo-państwowa. Robot może generować i wybierać między opcjami, aby osiągnąć swoje cele. Robot może zdecydować się na zatrzymanie piłki, zamiast podawać ją zgodnie z podręcznikiem gry. Ten poziom jest podobny do funkcji generowania na poziomach architektury automatyki, ale implikuje zwiększoną nieprzewidywalność i niedeterminizm.

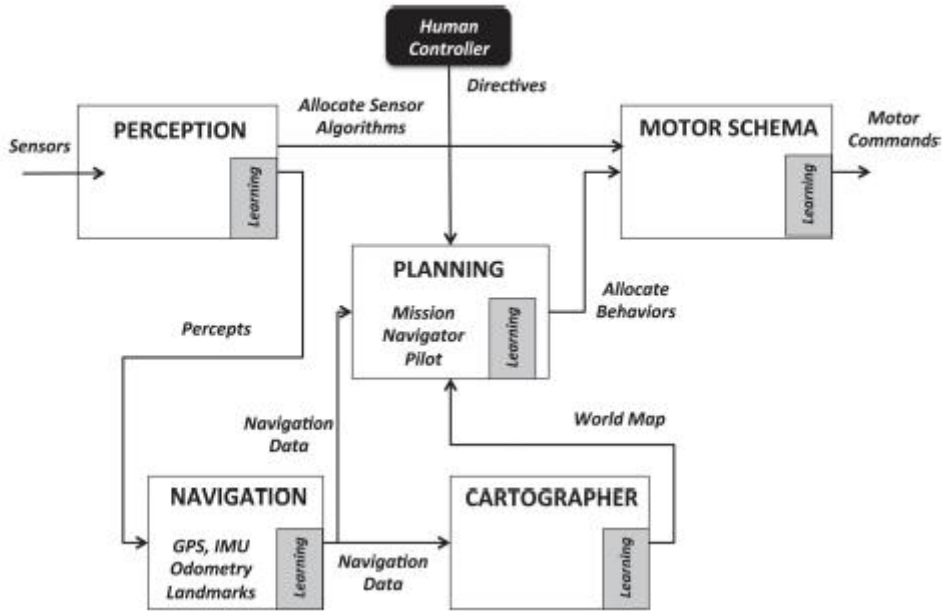
* Zamierzona autonomia. Robot może zmienić swoje cele, aby spełnić intencje swoich ról w drużynie, na przykład zastąpić innego gracza, gdzie chęć bycia dobrym graczem zespołowym jest ważniejszym priorytetem niż zamiar bycia dobrym pomocnikiem. W tym celu robot może negocjować z innymi. Ta funkcjonalność wprowadza pojęcie bycia świadomym innych i bezpośredniej interakcji z nimi, podobnie jak warstwa interfejsu w architekturze inspirowanej biologią.

* Autonomia ograniczenia. Robot może tworzyć własne role i cele, co w sztucznej inteligencji może wiązać się z rozluźnieniem ograniczeń dotyczących jego celów lub sposobu, w jaki osiąga swoje cele. Na tym najwyższym poziomie inicjatywy robot może zadać sobie pytanie, czy w ogóle chce się bawić. Ograniczona autonomia odzwierciedla poziom inicjatywy związany z robotami w filmach, gdzie nagle decydują się przejąć kontrolę nad światem. Jednak autonomia ograniczeń nadal podlega ograniczonej racjonalności.

Poziomy architektury operacyjnej inicjatywy są intuicyjnie atrakcyjne, z wieloma zaletami koncepcyjnymi, ale co najmniej jedną poważną wadą. Grupowanie funkcji według inicjatywy jest zgodne z definicją inteligencji z rozdziału 1 jako maksymalizacji sukcesu; bardziej inteligentni agenci próbują nowych sposobów na osiągnięcie sukcesu. Podejście oparte na poziomach inicjatywy ma podobieństwa z ACL w ustalaniu priorytetów adaptacji robota do pojawiających się sytuacji, ale jest bardziej skoncentrowane na relacji jeden-do-wielu robota z innymi agentami oraz celami i zadaniami. Inicjatywa obejmuje wybór i generowanie alternatyw, więc istnieje wpływ LOA. Jednak funkcje używane do opisu ACL (percepcja/świadomość sytuacyjna, analiza/podejmowanie decyzji oraz komunikacja/współpraca) są wymagane w jakiejś formie dla każdego poziomu inicjatywy, a zatem nie są dyskryminatorem. Jedną z wad poziomów architektury inicjatywy jest brak specyfiki LOA i ACL w odniesieniu do tego, które funkcje są potrzebne do osiągnięcia wymaganej autonomii na każdym poziomie.

Pięć podsystemów w architekturach systemów

Architektury operacyjne koncentrują się na ogólnym stylu, podczas gdy architektury systemowe koncentrują się na ogólnych komponentach. Można oczekiwać, że dom będzie miał kuchnię, jedną lub więcej sypialni i łazienek, salon lub pokój i szafy. Podobnie oprogramowanie dla inteligentnego robota zwykle zawiera co najmniej pięć podsystemów, które są hermetyzowane jako biblioteki obiektów lub podobne repozytoria programowania wielokrotnego użytku. Projektowanie architektury systemów zachęca projektantów do myślenia w kategoriach tworzenia bibliotek algorytmów i struktur danych dla każdego podsystemu, podobnych do modułów i funkcji w MATLAB, Mathematica i Maple, bibliotek ISML w Fortran i Standard Template Library w C + + . Biblioteki te służą jako ogólne izby informacyjne, z których projektant może wybrać konkretne funkcje dla konkretnego robota w celu dostosowania go do nowej aplikacji. Poniżej wymieniono najpopularniejsze podsystemy w literaturze robotyki AI. Typowa konfiguracja podsystemów jest również pokazana na rysunku w układzie podobnym do datagramu/aktygramu.



Zauważ, że diagram architektury systemów bardzo różni się od kanonicznej architektury operacyjnej; pierwszy to wysokopoziomowy schemat programowania robota. Poniżej wymieniono podsystemy.

* Podsystem Planowania jest powiązany z generowaniem i monitorowaniem ogólnych celów misji oraz przekazywaniem komponentu geoprzestrzennego tych celów do podsystemu nawigacyjnego, wybieraniem zasobów motorycznych i percepcyjnych, wdrażaniem lub tworzeniem instancji tych zasobów oraz monitorowaniem wykonania misji. Ten podsystem jest gospodarzem wielu klasycznych algorytmów planowania i alokacji zasobów.

* Podsystem Nawigacji jest powiązany z generowaniem i monitorowaniem ścieżek ruchu oraz wybieraniem zasobów do realizacji ruchów. Podsystem ten zazwyczaj zawiera wiedzę nawigacyjną potrzebną do wyboru zasobów, przejścia z punktu A do punktu B, decydowania o parametrach implementacji, jak szybko bezpiecznie podróżować i monitorować postępy. Podsystem może zawierać algorytmy planowania ścieżki, w szczególności planery topologiczne, algorytmy planowania ścieżki metryczne A* i D* lub algorytmy jednoczesnej lokalizacji i mapowania. Alternatywnie te algorytmy mogą być przechowywane w podsystemie Kartograf, ponieważ są one połączone ze strukturami danych dla Modelu Świata.

* Podsystem Kartograf, znany również jako Model Świata lub Mapa Świata, jest powiązany z konstruowaniem i utrzymywaniem reprezentacji wiedzy o świecie. Dla robotów zorientowanych na nawigację ten podsystem jest kluczową strukturą danych, która umożliwia generowanie, monitorowanie, wybieranie i wdrażanie działań. Te reprezentacje wiedzy są często mapami geoprzestrzennymi, ale mogą zawierać bardziej abstrakcyjne informacje symboliczne, takie jak przekonania o stanie świata lub intencje innych agentów. W bardziej rozważnych robotach ten podsystem wypełnia lukę między sygnałami a symbolami. W robotach interaktywnych podsystem monitoruje również własne modele umysłowe robota oraz przekonania, pragnienia i intencje innych agentów.

* Podsystem Schematu Motorycznego, znany również jako Biblioteka schematów motorycznych lub bardziej ogólnie jako podsystem zachowań, jest powiązany z wyborem najlepszych procedur motorycznych i wdrażaniem działań. Ten podsystem zawiera funkcje, które łączą funkcje deliberatywne i reaktywne z wymaganymi algorytmami prowadzenia, nawigacji i sterowania dla siłowników i efektorów. Ten podsystem zazwyczaj nie ma funkcji rozważania, ale raczej dotyczy

wykonania. Zazwyczaj nie zawiera algorytmów, które monitorują ogólny sukces akcji w kategoriach sztucznej inteligencji, na przykład, czy robot utknął w pętli „latania w oknie”. Ten rodzaj monitorowania jest obsługiwany przez podsystem planowania, który jako „dom” planu misji robota jest normalnym miejscem określania, czy plan jest realizowany. Algorytmy w podsystemie Schemat silnika mogą być zaprogramowane do monitorowania ich wykonania w pętli sterowania.

* Podsystem Percepcji, znany również jako Sensing, Perceptual Schema lub Perceptual Schema Library, jest powiązany z wyborem najlepszych czujników do działań motorycznych i implementacją algorytmów przetwarzania czujników. Jak zostanie opisane w rozdziale 10, podsystem ten zawiera funkcje, które kontrolują czujniki i wydobywają percepcję, czyli to, co jest postrzegane. Obejmuje to algorytmy, które wyodrębniają specjalny rodzaj percepcji zwany afordancjami lub te, które wykonują rozpoznawanie obiektów i rozumienie obrazów.

Istnieją dwie ważne uwagi dotyczące podsystemów. Po pierwsze, podsystemy nie są niezależne i nie reprezentują sekwencji działań programistycznych. Zamiast tego są one zasadniczo głównymi obiektami lub klasami w programowaniu obiektowym, które odzwierciedlają decyzje inżynierów oprogramowania dotyczące sposobu grupowania podobnych funkcji i danych. Program „główny” (część architektury technicznej) wykorzystuje podzbiory funkcji i struktur danych zawartych w podsystemach do tworzenia możliwości. Na przykład określone zachowanie, takie jak unikanie przeszkód, pobiera dane wyjściowe funkcji z podsystemu Percepcja (np. wyłaniający się obiekt) do wykorzystania przez schemat silnika (np. skręt w najbardziej otwartym kierunku) z podsystemu Schemat silnika. Po drugie, lista podsystemów niekoniecznie jest kompletna. Pięć wspólnych podsystemów podkreśla historyczne zainteresowanie społeczności robotycznej autonomiczną nawigacją platformy i mapowaniem. Te podsystemy mogą być wystarczające dla autonomicznego samochodu, ale nie dla robota chirurgicznego lub rozrywkowego. Pięć podsystemów opisanych powyżej nie ma wyraźnego komponentu do interakcji z innymi agentami lub do manipulacji. Inteligentny robot w służbie zdrowia, rozrywce, a nawet w ruchu miejskim prawdopodobnie miałby dodatkowe podsystemy.

Trzy paradygmaty architektury systemów

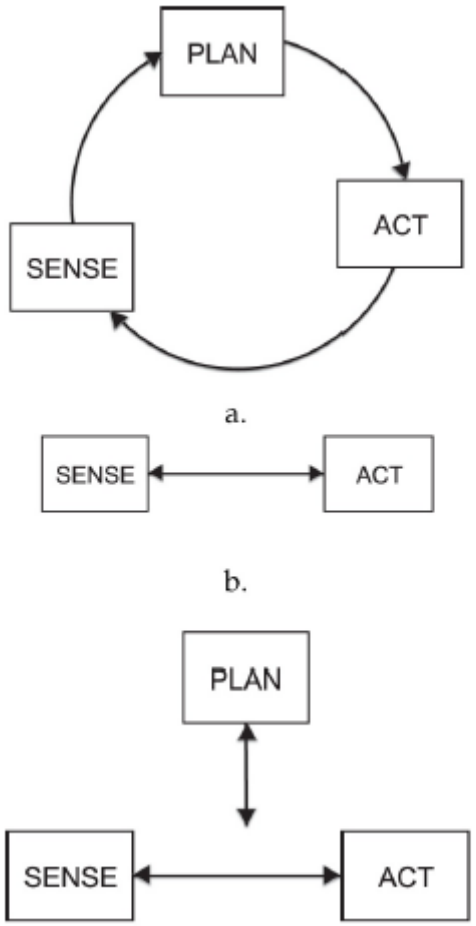
Jeśli pięć podsystemów odpowiada rodzajom pomieszczeń i przestrzeni w domu, paradygmaty architektoniczne odpowiadają podstawowym planom pięter łączących podsystemy. Historycznie, architektury systemów dla robotyki AI dzielą się na trzy kategorie zwane paradygmatami: hierarchiczne, reaktywne lub hybrydowe deliberatywne/reaktywne. Te paradygmaty porządkują przepływ danych i kontrolę w robocie. Ten przepływ danych i kontroli dla każdego paradygmatu można jednoznacznie opisać za pomocą dwóch cech: wzorca interakcji między trzema prymitywami robota: SENSE, PLAN i ACT oraz trasy wykrywania. Ta sekcja rozpoczyna się od zdefiniowania dwóch cech, a następnie wykorzystuje te cechy do formalnego scharakteryzowania trzech określonych stylów architektury systemu.

Cecha 1: Interakcja między elementami podstawowymi

Przypomnij sobie z Części 3, że istnieją trzy podstawowe elementy podstawowe: SENSE, PLAN i ACT, które podsumowano w tabeli 4.3.

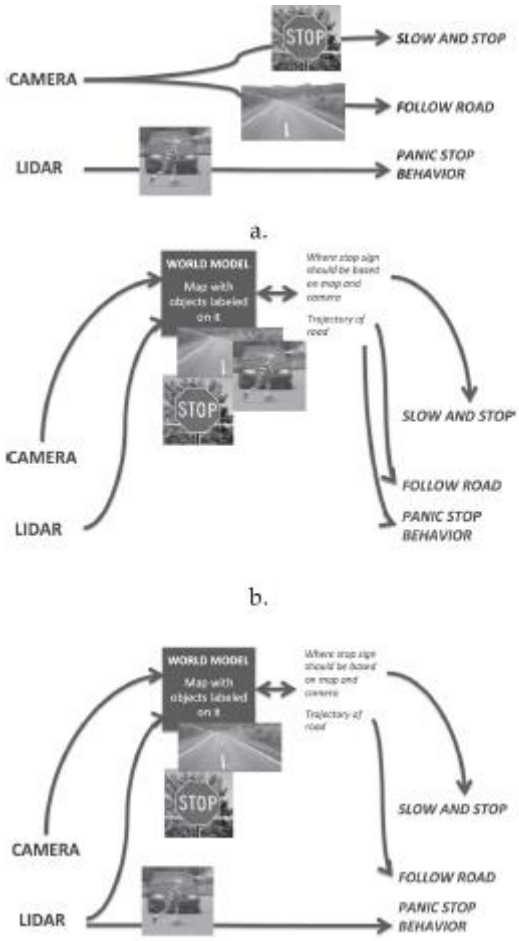
ROBOT PRIMITIVES	INPUT	OUTPUT
SENSE	Sensor data	Sensed information
PLAN	Information (sensed and/or cognitive)	Directives
ACT	Sensed information or directives	Actuator commands

Inteligentne roboty wykorzystują jeden z trzech cyklicznych wzorców interakcji między tymi prymitywami. Jednym z nich jest SENSE, potem PLAN, a na końcu DZIAŁANIE, w którym działanie zmienia stan świata, co prowadzi do nowej iteracji SENSE, PLANU, ACT i tak dalej. Jest to oznaczane jako SENSE, PLAN, ACT, gdzie „,” między SENSE, PLAN i ACT wskazuje sekwencję prymitywów. Ale rozdział 3 opisuje również reaktywną inteligencję zwierząt, która wykorzystuje wzorzec SENSE i ACT w połączeniu z zachowaniami bez PLANU. Zachowanie jest oznaczone jako SENSE-ACT, gdzie „-” wskazuje na sprzężenie lub współbieżność między prymitywami. Modele neurofizjologiczne faworyzują inny wzorzec inteligencji, w którym wyższy mózg funkcjonuje PLANUJ asynchronicznie, podczas gdy zachowania SENSE i ACT zarówno realizują plany, jak i reagują na otoczenie. Ten wzorzec inteligencji jest oznaczony jako PLAN, SENSE-ACT, gdzie istnieje zarówno sekwencja (plan, a następnie instancja wykrywania i działania) i sprzężenie (wykonywanie wykrywania i działanie do momentu zakończenia lub utworzenia nowego planu). Trzy style interakcji pokazano na rysunku



Cecha 2: Ścieżka wykrywania

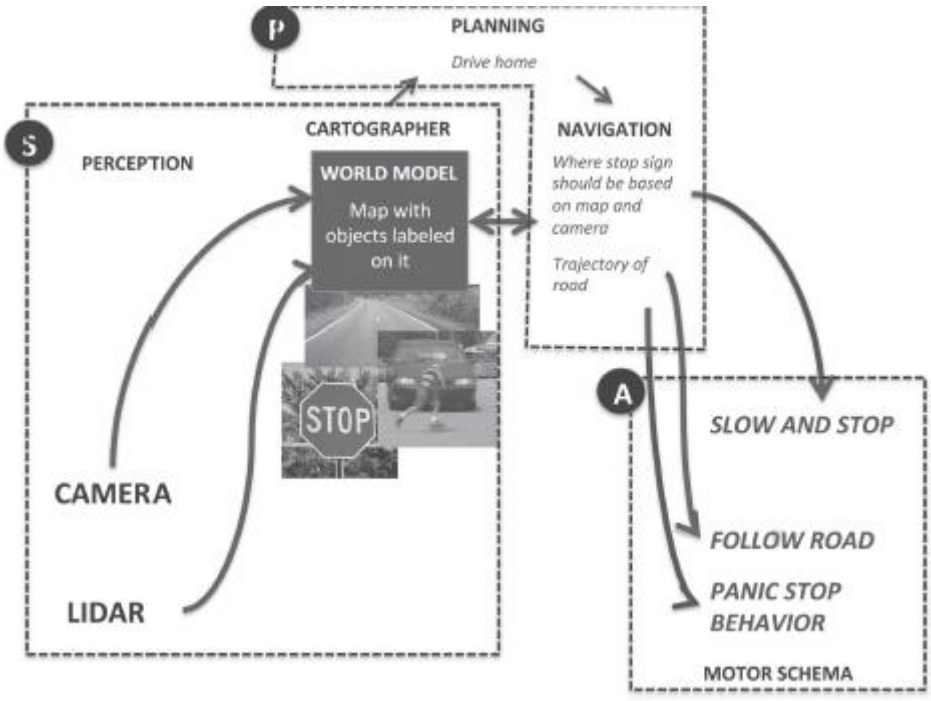
Inteligencja opiera się na wykrywaniu, a architektury systemów zazwyczaj wykorzystują jedną z trzech opisanych poniżej dróg, za pomocą których wykrywanie dociera do funkcji reaktywnych, deliberatywnych lub interaktywnych, które stanowią autonomiczną zdolność. Trzy ścieżki reprezentują wybory projektowe dotyczące zaawansowania obliczeniowego i kosztów, opóźnienia oraz tego, czy naśladować organizację wyczuwania u zwierząt. Lokalna trasa wykrywania prowadzi bezpośrednio od czujnika do zachowania lub funkcji wykorzystującej dane czujnika. Funkcja odbioru odpowiada za wszelkie przekształcenia lub konwersje surowych danych do odpowiedniej struktury danych. Przekształcenia lub konwersje pozostają lokalne, to znaczy w zakresie programowania tej funkcji. Lokalne wykrywanie może być mapowaniem jeden-do-jednego lub jeden-do-wielu, w którym dane z jednego czujnika mogą przejść do jednej lub więcej niezależnych funkcji.



Rysunek a przedstawia lokalną trasę wykrywania od dwóch czujników do trzech wyjść dla hipotetycznego samochodu bez kierowcy. Te same dane z kamery trafiają do dwóch różnych i niezależnych funkcji, jednej do wyszukiwania znaków stopu, które umożliwiłyby zatrzymanie robota, a drugiej do szukania białych linii do podążania drogą. To jest relacja jeden-do-wielu. Wyjście lidaru przechodzi do trzeciej funkcji, która wyzwala panikę lub zatrzymanie awaryjne, jeśli coś znajduje się przed samochodem. To jest relacja jeden na jednego. Globalna ścieżka wykrywania to sposób przesyłania danych pochodzących ze wszystkich czujników do funkcji, która przekształca i łączy dane w globalny model świata lub ujednoliconą strukturę danych. Globalne wykrywanie można traktować jako mapowanie wiele do jednego, w którym dane z wielu czujników trafiają do jednej funkcji. Rysunek 4.8b przedstawia trasę wykrywania z dwóch czujników połączonych w model świata, a następnie model świata, przedstawiający lokalizację znaku stopu, trajektorię białych linii i obecność czegokolwiek przed samochodem. To jest relacja wielu do jednego. Fuzja z modelem świata stanowi kolejny krok i może mieć znaczenie obliczeniowe, powodując w ten sposób opóźnienie w przetwarzaniu czujnika. Hybrydowa trasa wykrywania jest kombinacją, w której te same dane z czujnika mogą trafiać do jednej lub większej liczby funkcji, które wykonują lokalne przekształcenia, oraz do globalnej funkcji wykrywania. Wykrywanie hybrydowe można traktować jako mapowanie wiele do wielu, w którym dane z wielu czujników mogą trafiać do wielu niezależnych funkcji. Rysunek 4.8c pokazuje hybrydową ścieżkę wykrywania, w której dane z dwóch czujników są łączone w model świata, ale są przesyłane bezpośrednio do funkcji, takich jak zatrzymanie paniki, które nie tolerują opóźnień w przetwarzaniu modelu świata lub nie potrzebują świata Model. To jest relacja „wielu do wielu”. Ta hybrydowa ścieżka wykrywania najbardziej naśladuje ludzkie procesy wykrywania, w których sygnały neuronowe wędrują po ścieżkach aferentnych i rozdzielają się.

Hierarchiczny paradygmat architektury systemów

Hierarchiczne architektury systemów organizują pięć podsystemów w celu obsługi wzorca SENSE-PLAN-ACT, który opiera się na globalnej trasie wykrywania. Jak zauważono w Albus i Mystel4, hierarchie są naturalnym sposobem organizowania funkcjonalności, a jeśli priorytety i cele są jasne, mogą być wydajne obliczeniowo, ponieważ mogą ograniczyć obliczenia poprzez określenie ramy i zdefiniowanie zamkniętego świata. Systemy hierarchiczne są używane w przypadku wdrożeń, w których misja lub aplikacja jest dobrze zrozumiana i nie oczekuje się dalszego dodawania funkcji lub większych aktualizacji. Zautomatyzowana produkcja i systemy naprowadzania, nawigacji i kontroli są zwykle hierarchiczne, tracąc pracę w szeroko otwartym świecie za możliwość ścisłej kontroli przepływu realizacji. Systemy hierarchiczne architektury są często używane tam, gdzie nie oczekuje się ponownego użycia kodu lub późniejszego rozszerzenia funkcjonalności. Dlatego celem jest optymalizacja programowania. Przepływ typowej architektury hierarchicznej wykorzystuje podsystemy w sekwencji. Faza SENSE zbiera dane ze wszystkich czujników przy użyciu procedur z podsystemu Percepcja, a następnie łączy dane z czujników w Model Świata przy użyciu procedur z podsystemu Kartograf. Proces fuzji to coś więcej niż stopienie bieżącego zestawu odczytów; łączy bieżące odczyty ze starszymi odczytami i wszelką wiedzą a priori, taką jak mapy. Faza PLAN stosuje procedury z podsystemu Planner do Modelu Świata w celu określenia wszelkich zmian w misjach lub ograniczeniach, a następnie planuje trasy nawigacyjne i ruchy przy użyciu procedur z podsystemu Nawigacja. Wynikiem jest zestaw Akcji. Faza ACT wykonuje działania przy użyciu procedur z podsystemu schematu silnika. Działania mogą również obejmować podsystem Percepcji, jeśli czujniki mają siłowniki do poruszania. Sekwencja SENSE-PLAN-ACT jest wykonywana w ciągłej pętli. Rysunek



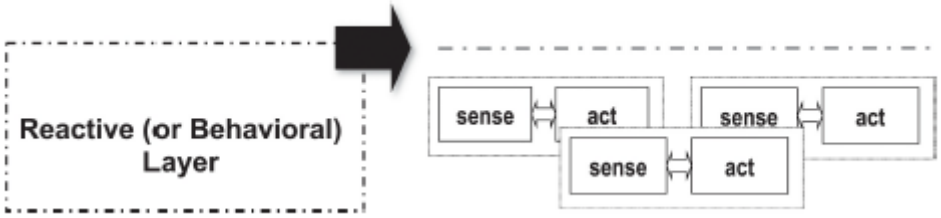
ilustruje, jak wyidealizowana hierarchiczna architektura systemów może działać w przypadku samochodu bez kierowcy. Przepływ programu rozpocznie się od fazy SENSE. Dane byłyby zbierane z kamery i czujników LIDAR, a następnie łączone z wiedzą a priori w Model Świata. Z Modelu Świata system wyciągnąłby odpowiednią percepcję, w tym przypadku, gdzie jest droga, że jest tam znak stopu i że na drodze znajduje się nieoczekiwany obiekt. Faza PLANu wykorzystywałaby model świata do generowania, aktualizowania lub modyfikowania planu misji, w którym ograniczenia miały dotrzeć do punktu P, przestrzegać wszystkich przepisów ruchu drogowego i nie uderzać w nic na drodze.

Zaplanowałby wtedy działania nawigacyjne, w tym przypadku natychmiastowe zatrzymanie, aby uniknąć uderzenia w obiekt na drodze, ale pozostanie na drodze. Podsystem nawigacyjny odnotowałby również, że znak stopu wskazywał, że samochód znajduje się w punkcie P na trasie i dobrze posuwa się na zaplanowanej trasie. Faza ACT realizowałaby zaplanowane działania, zatrzymując się w celu uniknięcia nieoczekiwanego obiektu na drodze. System sterowania w czasie rzeczywistym NIST (RCS) jest najlepiej znanym z hierarchicznych systemów architektonicznych. Jim Albus z National Bureau of Standards (później przemianowanego na National Institute of Standards and Technology lub NIST) przewidział potrzebę inteligentnych manipulatorów przemysłowych, nawet gdy inżynierowie i badacze sztucznej inteligencji podzielili się na dwie grupy. Zauważył, że jedną z głównych przeszkód w stosowaniu sztucznej inteligencji w produkcji robotów był brak wspólnych terminów i wspólnego zestawu standardów projektowych. To spowodowało, że przemysł i producenci sprzętu obawiali się sztucznej inteligencji z obawy przed zakupem drogiego robota, który nie byłby kompatybilny z robotami kupowanymi w przyszłości. Na początku lat 80. opracował bardzo szczegółową architekturę zwaną Architekturą Systemu Sterowania w Czasie Rzeczywistym (RCS), która ma służyć jako przewodnik dla producentów, którzy chcieli zwiększyć inteligencję swoich robotów. Został przyjęty przez armię USA do kilku dużych projektów, wpływanie na sposób organizacji kodu przez wykonawców obronnych do dnia dzisiejszego. Główną zaletą paradygmatu hierarchicznego było to, że zapewniał uporządkowanie relacji między odczuwaniem, planowaniem i działaniem. Był intuicyjnie atrakcyjny i zgodny ze sterowaniem pętlą OODA. Podstawową wadą było planowanie. W każdym cyklu aktualizacji robot musiał aktualizować globalny model świata, a następnie dokonywać pewnego rodzaju planowania. Dzisiejsze algorytmy wykrywania i planowania były bardzo powolne (a wiele z nich nadal działa), więc wprowadziło to znaczące wąskie gardło. Zauważ też, że wyczuwanie i działanie są zawsze oddzielone. To skutecznie wyeliminowało wszelkie rodzaje działań typu bodziec-odpowiedź („skała na mnie wali, powinienem się ruszyć gdziekolwiek”), które są obserwowane w naturze.

Paradygmat systemów reaktywnych

Systemy reaktywne wykorzystują tylko dwa podsystemy, Percepcję i Schemat Motoru, z pięciu kanonicznych podsystemów, aby wspierać wzorzec SENSE-ACT, który opiera się na lokalnej trasie wykrywania. Programowanie przez zachowanie ma wiele zalet, większość z nich jest zgodna z dobrymi zasadami inżynierii oprogramowania. Zachowania są z natury modułowe i łatwe do przetestowania w oderwaniu od systemu (tj. wspierają testowanie jednostkowe). Zachowania wspierają również stopniowe rozszerzanie możliwości robota. Robot staje się bardziej inteligentny dzięki większej liczbie zachowań. Dekompozycja behawioralna skutkuje implementacją, która działa w czasie rzeczywistym i jest zwykle niedroga obliczeniowo. Ogólnie rzecz biorąc, szybkości reakcji robota reaktywnego są równoważne czasom reakcji na bodziec u zwierząt, chociaż jeśli wybory architektury technicznej są słabo zaimplementowane, implementacja reaktywna może być powolna. Roboty czysto reaktywne przydają się w prostych systemach autonomicznych, takich jak odkurzacz iRobot Roomba, roboty kosiarki i roboty rozrywkowe, gdzie urządzenie wykorzystuje biomimikę do naśladowania owadów lub bardzo podstawowe zachowania zwierząt, w których planowanie i optymalność nie są niezbędne. Architektury systemów reaktywnych mogą być również wykorzystywane do dodawania autonomicznych możliwości do istniejącego systemu człowiek-maszyna, w którym ważna jest szybka reakcja, na przykład gdy samochód hamuje autonomicznie, aby uniknąć zderzenia z innym samochodem lub przeszkodą. Naukowcy mają tendencję do postrzegania systemów reaktywnych jako niższej warstwy w szerszej hybrydowej architekturze deliberatywnej/reaktywnej. Przepływ typowej architektury reaktywnej wykorzystuje równolegle wiele instancji podsystemów Percepcji i Działania. Faza SENSE zbiera dane ze wszystkich czujników za pomocą procedur z podsystemu Percepcji i bezpośrednio dostarcza je do schematów motorycznych, które wytwarzają działania (faza ACT). Może istnieć wiele par SENSE-ACT lub zachowań, a te zachowania działają jednocześnie i niezależnie. Każde

zachowanie jest pętlą lokalną i może mieć inny czas obliczeń, a zatem zachowania działają asynchronicznie. Wyczuwanie jest kierowane do każdego zachowania bez żadnego filtrowania ani przetwarzania, a każde zachowanie wydobywa bezpośrednią percepcję, która jest dla niego unikalna. Czujnik może dostarczać ten sam odczyt czujnika jednocześnie do różnych zachowań, a następnie każde zachowanie oblicza inną percepcję. Przetwarzanie czujnika jest lokalne dla każdego zachowania bez globalnego modelu świata. Nie ma fazy PLANOWANIA, ponieważ zachowania wywołują Akcje tylko wtedy, gdy wykrywanie wywołuje reakcję. Rysunek 4.10 ilustruje, jak wyidealizowana reaktywna architektura systemu może działać w przypadku samochodu bez kierowcy.



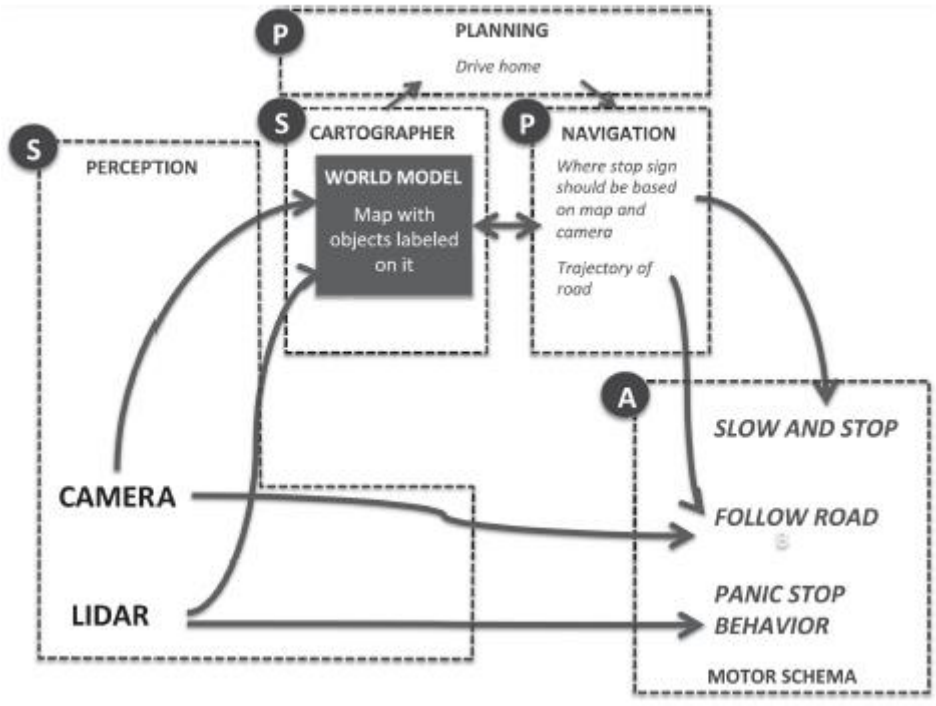
Przepływ programu rozpocznie się od fazy SENSE. Dane z czujnika z kamery byłyby kierowane do zachowania „zatrzymaj się na znakach stop", a zachowanie to określiłoby, czy należy użyć hamulców. W tym samym czasie dane czujnika z kamery byłyby rozdzielane i kierowane do zachowania „podążania za drogą", a zachowanie obliczałoby korekcję kierowania. Dane czujnika dla zachowania „zatrzymania napadowego" byłyby również zbierane z LIDAR i obliczane w razie potrzeby. Faza ACT zastosowałaby funkcję koordynacji, ponieważ oba zachowania przyczyniają się do działania hamulców. Na przykład zachowanie „zatrzymaj się na znakach stop" może oznaczać „brak hamulców", ponieważ żaden znak stopu nie jest widoczny, podczas gdy „zatrzymaj się w panice" może oznaczać „wciśnij hamulce!" ponieważ na drodze jest przeszkoda. Funkcja koordynacji rozwiązałaby spór między zachowaniami i umożliwiłaby zatrzymanie paniki. Robot zbudowany wyłącznie z warstwy reaktywnej jest często nazywany robotem reaktywnym lub behawioralnym. iRobot Roomba jest prawdopodobnie najbardziej znanym przykładem robota reaktywnego. Warstwa reaktywna, której orędownikami byli Rodney Brooks i Ron Arkin, była głównym przedmiotem badań robotyki AI w latach 1986–1996. Naukowcy zbadali, w jaki sposób agent może zmaksymalizować swoje szanse na sukces, korzystając tylko z informacji bezpośrednio dostrzegalnych na świecie i bez planowania. Architektura subsumpcji Brooksa30 to najbardziej znany styl wdrażania reaktywności jako architektury systemów autonomicznych. Subsumpcja organizuje zachowania w warstwy kompetencji i wykorzystuje mechanizmy subsumpcji i hamowania, aby służyć jako funkcja koordynacji dla wielu warstw. Podsumowanie zostanie omówione dalej w rozdziale 8. Pola potencjalne są kolejnym popularnym mechanizmem koordynacji zachowań i zostały spopularyzowane przez Arkina na podstawie oryginalnej pracy Khatiba. Warstwa reaktywna jest na ogół programowana w języku proceduralnym, takim jak C, C + + lub Java, ale można to zaprogramować w Lispie. Robot behawioralny ma co najmniej trzy zalety. Po pierwsze, bezpośrednia percepcja jest zwykle łatwa do zaprogramowania. Po drugie, zachowania są wysoce modułowe. Ta modułowość oznacza, że nowe zachowania mogą być dodawane bez przeprogramowywania istniejących zachowań roboczych. Roboty behawioralne mają tendencję do wdzięcznej degradacji, ponieważ awaria jednego zachowania nie powoduje niepowodzenia innych zachowań, chociaż zależy to od architektury technicznej. Po trzecie, robot reaktywny lub behawioralny może dobrze współpracować z człowiekiem w telesystemach, ponieważ może reagować na bodźce, takie jak bliskość, wywołując zachowania paniki (stop!) lub strzeżone zachowania ruchowe (robot nie zbliży się do jest bezpieczny bez względu na wkład człowieka). Zachowania często pojawiają się jako automatycznie wyzwalane „makra", takie jak bezprzewodowy robot powracający do domu po utracie sygnału, samonaprawa i ładowanie. Robot z tylko warstwą reaktywną ma trzy wady. Jedną wadą jest

to, że niemożność PLANOWANIA prowadzi do efektu „latania w oknie” przedstawionego w Części 3. Efekt ten występuje, ponieważ robot wyczuwa i działa lokalnie na to, co może dostrzec w danej chwili. Nie ma możliwości monitorowania się w czasie i odkrycia, że utknął w pętli. Drugą wadą jest wyłanianie się całościowego zachowania z indywidualnych zachowań przy użyciu bezpośredniej percepcji, co utrudnia precyzyjne przewidzenie, co zrobi robot; to znaczy, dodaje do niedeterminizmu systemu. Na przykład zachowanie unikania może prowadzić robota w lewo lub w prawo od przeszkody, w zależności od odczytów czujnika. Po trzecie, zachowanie może nie być optymalne, ponieważ opiera się na lokalnym, natychmiastowym wykrywaniu, a nie na planie wygenerowanym z globalnego modelu świata.

Paradygmat hybrydowych systemów deliberatywnych/reaktywnych

HYBRYDOWE SYSTEMY DELIBERATYWNE/REAKTYWNE

Hybrydowe architektury systemów deliberatywno-reaktywnych organizują pięć podsystemów w celu obsługi wzorca „PLAN, a następnie SENSE-ACT” w połączeniu z hybrydowym wykrywaniem. Idea PLANU, a następnie wzoru SENSE-ACT, wyewoluowała zarówno z biomimiki, jak iz dwóch założeń inżynierii oprogramowania. Po pierwsze, planowanie obejmuje długi horyzont czasowy i wymaga globalnej wiedzy, dlatego powinno być oddzielone od wykonywania w czasie rzeczywistym na zasadzie koherencji inżynierii oprogramowania (różne funkcje powinny być umieszczone w różnych obiektach). Po drugie, algorytmy planowania i łączenia czujników są kosztowne obliczeniowo, dlatego należy je oddzielić od wykonywania zachowań w czasie rzeczywistym, ponieważ spowalniają one szybkość reakcji. Model Świata jest budowany przez procesy niezależne od wyczuwania specyficznego dla zachowania. Systemy hybrydowe wykorzystują hybrydowe trasy wykrywania. W wielu architekturach hybrydowych procesy kartografa lub tworzenia modeli mogą mieć dostęp nie tylko do tych samych czujników, co zachowania, ale także do wyników wykrywania lokalnego. Kartograf może również mieć czujniki, które są przeznaczone do dostarczania obserwacji przydatnych do modelowania świata, ale nie są wykorzystywane do żadnych aktywnych zachowań. Systemy hybrydowe są najbardziej elastyczne i stały się domyślną architekturą systemów dla badaczy oraz do zastosowań, w których oczekuje się, że w przyszłości funkcjonalność robota zostanie zmodyfikowana lub rozszerzona. Rysunek



ilustruje, jak wyidealizowana architektura systemu hybrydowego może działać w przypadku samochodu bez kierowcy. Przepływ programu rozpocznie się od fazy SENSE. Dane z czujników z kamery byłyby dzielone na dwa strumienie, jeden przyczyniający się do powstania Modelu Świata w Kartografie, drugi skierowany do zachowania „podążania za drogą" w podsystemie schematu motorycznego. Podobnie dane z czujników z lidaru byłyby dzielone i przesyłane strumieniowo do Kartografa i „paniki zatrzymania". Zachowania „podążania za drogą" i „zatrzymania się w panice" byłyby wykonywane szybciej, ponieważ odruchowo wykorzystują lokalne wykrywanie. Kartograf byłby wolniejszy, ponieważ musiałby podjąć dodatkowe kroki w zakresie przetwarzania wyczuwania, a następnie zintegrować wydobytą percepcję z globalnym modelem świata. Z Modelu Świata system wyodrębniłby odpowiednią percepcję potrzebną do planowania w fazie PLANOWANIA oraz do wykorzystania w nawigacji i działaniu. Dane wyjściowe Modelu Świata mogą służyć jako wirtualny strumień czujników do zachowań. Architektura robotów mobilnych 3T, czyli 3-warstwowa, jest najlepszym przykładem systemu hierarchii stanów i jest stosowana głównie w NASA. Jak sama nazwa wskazuje, 3T dzieli się na trzy warstwy, jedną wyraźnie reaktywną, jedną wyraźnie deliberatywną i jedną służącą jako interfejs między nimi. 3T był używany głównie w łazikach planetarnych, pojazdach podwodnych i asystentach robotów dla astronautów. Systemy hybrydowe są najbardziej elastyczne i stały się domyślną architekturą systemów dla badaczy oraz do zastosowań, w których oczekuje się, że w przyszłości funkcjonalność robota zostanie zmodyfikowana lub rozszerzona. Wadą systemów hybrydowych jest dodatkowa złożoność programistyczna zarządzania procesami asynchronicznymi. Zachowania SENSE-ACT działają niezależnie i bardzo szybko podczas planowania procedur i globalnej aktualizacji wykrywania w wolniejszym tempie.

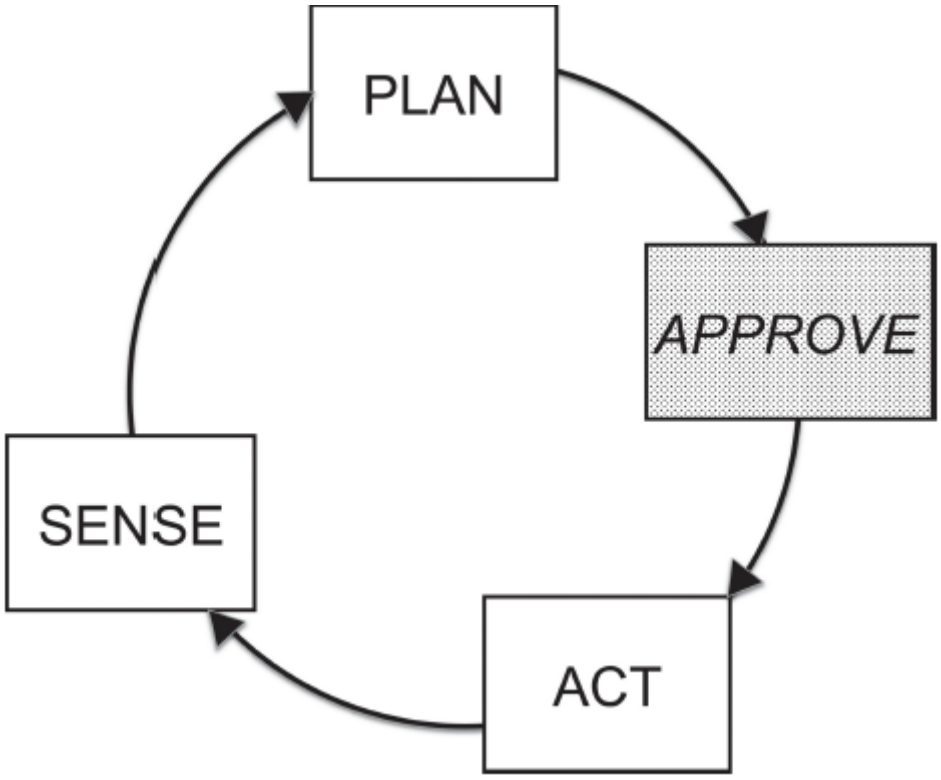
Zatwierdzenie wykonania i wykonanie zadania

Poprzednie sekcje pokazują, że zbudowanie inteligentnego robota to duże przedsięwzięcie. Wiele aspektów rozważań i reakcji budzi obawy, czy mamy algorytmy i metody koordynacji, aby rzeczywiście zbudować w pełni autonomicznego robota, który może pracować w otwartym świecie. Jedną ze strategii jest zbudowanie

robota, ale poproś człowieka, aby sprawdził zaplanowane działania robota, zanim je wykona; strategia ta jest znana jako zatwierdzanie wykonania i próba zadania w literaturze dotyczącej czynników ludzkich. Na przykład podczas DARPA Robotics Challenge oczekiwano, że zespoły zasymulują wykonanie planu, aby potwierdzić, że powinien on odnieść sukces po wdrożeniu. Zasadniczo zmienia hierarchiczny cykl SENSE-PLAN-ACT na układ SENSE-PLAN-APPROVE-ACT pokazany na rysunku 4.12, wprowadzając człowieka do pętli w celu obejrzenia symulacji i zatwierdzenia następnego działania. Niestety, SENSE-PLAN-APPROVE-ACT nie okazał się bardzo skuteczny, choć jest atrakcyjny.

ZATWIERDZENIE WYKONANIA

Ten rodzaj sprawdzania przedstawiony na rysunku jest często wdrażany jako zatwierdzenie wykonania, w którym agent poznawczy, człowiek lub komputer, rozważa następny krok i decyduje, czy jest on bezpieczny.



Jak opisuje Klein, zatwierdzenie wykonania przez człowieka zazwyczaj skutkuje niską wydajnością, ponieważ ludzie nie są dobrzy w rozważaniu wszystkich sposobów, w jakie zaplanowane działanie może się nie udać. Komputery mogą być bardziej wszechstronne, ale generalnie zatwierdzenie wykonania należało do kompetencji człowieka.

PRÓBA DO ZADANIA

Podobnym podejściem do weryfikacji planu jest rozważenie całej sekwencji działań, co jest znane jako próba zadania. Godnym uwagi przykładem próby zadania jest komórka przetwarzania jądrowego, w której znany jest układ budynku i każdy element wyposażenia. Ta wiedza jest wykorzystywana do wygenerowania komputerowej symulacji, czy określona ścieżka robota obsługującego materiał jądrowy zderzy się z innym sprzętem i spowoduje rozlanie. Operator może wyświetlić symulację i potwierdzić, że nie ma kolizji lub nieoczekiwanego zachowania. Niestety, ludzie nie są dobrzy w wyłapywaniu problemów podczas próby zadania. Klein pokazuje, że ludzie nie mogą zarządzać więcej niż trzema zmiennymi lub sześcioma przejściami. Niska wydajność ludzi w wyobrażaniu sobie wszystkich sposobów, w jakie coś może pójść źle, lub próbach, co zrobić, gdy coś pójdzie nie tak, sugeruje, że to robot powinien ostatecznie zweryfikować swój plan lub przynajmniej stworzyć bardziej godne zaufania plany. Na szczęście zatwierdzenie wykonania i próba zadania mogą nie być potrzebne, a nawet właściwe, dla wszystkich działań. Jeśli kawałek sufitu spada na robota, musi on natychmiast zejść z drogi lub zaryzykować zbyt późne przybycie z najlepszym miejscem do poruszania się, aby uniknąć zmiażdżenia. Opóźnienie w projekcji konsekwencji może dosłownie oznaczać śmierć robota.

Podsumowanie

W tej Części rozważono autonomię w kategoriach pożądanych zdolności i omówiono, jak te zdolności wpisują się w ogólne ramy programowania. Ramy programistyczne są przedstawiane jako architektura operacyjna, opis wysokiego poziomu tego, co robi robot, oraz architektura systemów, określająca ogólne systemy potrzebne do wdrożenia funkcjonalności. Ramy architektoniczne mają kilka zalet

koncepcyjnych. Opisuje ogólne rozmieszczenie funkcji w inteligentnym robocie przy użyciu zasad abstrakcji i modułowości inżynierii oprogramowania i służy jako półformalna specyfikacja projektu i lista kontrolna kompletności odpowiadająca na pytanie: Czy wszystkie funkcje są uwzględnione w konkretnym projekcie robota ? Jeśli niektórych brakuje, czy jest to akceptowalna luka w zabezpieczeniach, czy też należy się tym zająć? Społeczność robotyki AI skupiła się na kanonicznej architekturze operacyjnej składającej się z trzech warstw reakcji, rozważań i interakcji. Kanoniczna architektura operacyjna oddziela zachowania reaktywne (reakcje) od funkcji poznawczych generowania, wybierania, wdrażania i monitorowania planów i zasobów (deliberacja) oraz dodaje warstwę interakcji do komunikacji i koordynacji z innymi agentami. Ta organizacja programistyczna łączy wiedzę biologiczną z poziomami automatyzacji i autonomicznej kontroli oraz stylami architektury preferowanymi przez producentów, armię amerykańską i społeczności składające się z wielu agentów. Architektura systemu, która najlepiej wyraża tę architekturę operacyjną w kategoriach podsystemów, to hybrydowa architektura deliberatywna/reaktywna lub w skrócie „hybrydowa". Architektury systemów hybrydowych wykorzystują przepływ PLAN, a następnie SENSE-ACT z rozproszonym wykrywaniem oraz funkcjami reaktywnymi i deliberatywnymi, które działają w różnych skalach czasowych. Istnieje pokusa, aby wprowadzić człowieka do pętli, aby sprawdzić plany robota przed wykonaniem, przesuwając przepływ do sekwencji SENSE, PLAN, APPROVE, ACT, ale istnieją dowody od społeczności zajmującej się czynnikami ludzkimi, że rzadko działa to zgodnie z oczekiwaniami. W przeglądzie rozdziału postawiono dwa pytania, z których na każde udzielono częściowej odpowiedzi. Biorąc pod uwagę, że autonomia ma inny styl programowania, co to jest? Styl programowania autonomii polega na myśleniu w kategoriach metafor biologicznych, uchwyceniu reakcji, rozważań i interakcji potrzebnych do tego, aby robot odniósł sukces jako kompletny system. Metafora biologiczna prowadzi do kanonicznej architektury operacyjnej składającej się z trzech warstw: warstwy reaktywnej składającej się z zachowań SENSE-ACT często zaprogramowanych w architekturze technicznej przy użyciu języka proceduralnego, takiego jak C++; warstwa deliberatywna, która utrzymuje model świata, odwzorowując symbole na grunt i zapewniając funkcje generowania, wyboru, implementacji i monitorowania, które często są programowane w języku funkcjonalnym, takim jak Lisp; oraz warstwa interaktywna, która umożliwia robotowi komunikowanie się i pracę z ludźmi i innymi robotami lub agentami oprogramowania i często jest programowana w OWL lub XML, aby wykorzystać ontologie i relacje. Odpowiedź na drugie pytanie: Czy inteligencję można dodawać warstwami, takimi jak aktualizacja do „wersji pro" lub pobieranie „aplikacji" w razie potrzeby? to zarówno „tak", jak i „nie". Po stronie „tak" dyskusji, inteligencja ogólnego przeznaczenia jest zorganizowana w warstwy abstrakcji i funkcji modułowych. Robot może nie potrzebować wszystkich warstw do określonego zadania; na przykład robot reaktywny może nie wymagać rozważań i interakcji. Jednak odpowiedź brzmi „nie", ponieważ pomyślna zdolność autonomiczna, taka jak nawigacja, wymaga zazwyczaj uwzględnienia wszystkich warstw i funkcji w projekcie. Na przykład albo robot wewnętrznie zajmuje się monitorowaniem planu, ponieważ monitorowanie nie jest ważne dla tej aplikacji (np. nie przeszkadza nam, że od czasu do czasu robot odkurzający zacina się), albo człowiek musi monitorować ręcznie. Główna klasa niepowodzeń misji wynika z tego, że projektant zaniedbuje jawne uwzględnienie całego zestawu funkcji deliberatywnych, na przykład zapewnienie generowania planu, ale nie monitorowania planu. Nie tylko prowadzi to do niepowodzeń misji per se, ale człowiek musi się starać aby zapobiec lub odzyskać te brakujące możliwości. Prowadzi to do zwiększonego zapotrzebowania na siłę roboczą i nie ma gwarancji, że człowiek może wykonywać te funkcje lub ręcznie odzyskać siły po niepowodzeniu misji. Niestety, trzy warstwy w architekturze hybrydowej lub jakiejkolwiek operacyjnej, systemowej lub technicznej architekturze nie są mapą drogową do stopniowego budowania robota. Elegancja inżynierii oprogramowania modułowych architektur operacyjnych i systemowych to nie to samo, co mapa drogowa. Przemysł robotyki i wojskowi agenci zaopatrzenia zwykle traktowali architektury operacyjne jako model rozwoju przyrostowego, zasadniczo mówiąc, że branża powinna najpierw

skupić się na robotach, które mogą obsłużyć jedną warstwę, a następnie przejść do następnego zestawu modułów lub warstw i tak dalej. Ten sposób myślenia podważa ideę, że nawet „prosta autonomiczna" zdolność może wymagać reakcji, namysłu i interakcji. Warstwy odzwierciedlają funkcjonalność, a podsystemy i koordynacja odzwierciedlają ogólny układ, podobnie jak typy pomieszczeń i ogólne plany pięter. Warstwy nie sugerują, że najpierw należy zbudować jeden podsystem, potem robota używanego (lub zajętego domu), potem kolejny podsystem i tak dalej. Ludzie chcą działającego robota lub kompletnego domu. Roboty, podobnie jak domy, na ogół nie są projektowane z myślą o łatwym dodawaniu - jednak jeśli właściciel wie, że dom będzie z czasem rozbudowywany, to jest on projektowany i budowany inaczej niż typowy dom z przybudówką. W tej części mogło pojawić się trzecie pytanie, które zostanie odłożone do rozdziału 19 dotyczącego projektowania: Ile sztucznej inteligencji potrzebuje robot? Ile sztucznej inteligencji zależy od pożądanych możliwości robota. Niedrogi robot odkurzający może odnieść sukces, naśladując proste zachowania związane z wypasaniem zwierząt, podczas gdy robot warty wiele milionów dolarów, który ma podróżować w kosmos, może wymagać zaawansowanego planowania ścieżki i misji. Ten rozdział pomógł przygotować grunt pod rozdział 19, który postawi pięć pytań, które pomogą skoncentrować projekt na zapewnieniu wystarczającej inteligencji, aby spełnić oczekiwania robota, złożoność środowiska i możliwości robota. Jedno pytanie brzmi, jakie funkcje musi zapewniać robot? Te funkcje to generalnie zachowania, funkcje deliberatywne (generowanie, monitorowanie, wybieranie, wdrażanie) czy uczenie się? Drugie pytanie brzmi: Jakiego horyzontu planowania wymagają funkcje? W tym rozdziale opisano Teraźniejszość (praca tylko z bezpośrednią informacją), Przeszłość i Teraźniejszość (wykorzystywanie wiedzy o przeszłości z natychmiastową informacją) oraz Przyszłość (projekcja). Kolejne pytanie brzmi: Jak szybko algorytmy muszą się aktualizować? Zależy to od tego, czy funkcja służy do kontroli w czasie rzeczywistym, co może wymagać gwarantowanych wskaźników wykonania, czy też do planowania, które może trwać dłużej i nie musi być wykonywane tak często. Czwarte pytanie brzmi: Jakiego modelu potrzebuje robot? Modele mogą być lokalne dla określonego zadania lub wymagać zbudowania i utrzymania większego, niezależnego od zadania modelu. Często pomijanym pytaniem jest Skąd pochodzi człowiek? W rozdziale opisano interakcje, ale także wprowadzono domniemanie, że człowiek wykonuje szczegółowy monitoring robota. Układ kanonicznej architektury operacyjnej i głównych podsystemów w architekturze systemów hybrydowych zapewni podstawę lub szkielet do zakotwiczenia pozostałych algorytmów omówionych w tym tekście