Využitie multi-agentových platforiem pre systémy riadenia technologických procesov

(Diplomová práca)

Eva Vasilová

Univerzita komenského
Fakulta matematiky, fyziky a informatiky
Katedra informatiky

Vedúci práce: RNDr. Peter Fabo, PhD.

Bratislava, 2006

Čestne prehlasujem, že túto diplomovú prácu som vypracovala samostatne, a že som v zozname literatúry v závere práce uviedla všetky pramene, z ktorých som čerpala.

V Bratislave, 8.5.2006

Eva Vasilová

Poďakovanie

Chcem poďakovať svojmu diplomovému vedúcemu Petrovi Fabovi za jeho pochopenie pri rozpačitých začiatkoch tejto práce a za rady, ktoré pomohli doviesť výskum do úspešného konca.

Moja vďaka taktiež patrí kolegovi Petrovi Mišákovi, ktorý nešetril časom a poskytol mi užitočné materiály k výskumu a spolužiakovi Michalovi Pokornému za pomoc pri technických detailoch práce.

V neposlednom rade chcem poďakovať svojim rodičom, za ich dôveru a investície, ktoré neváhali vložiť do môjho štúdia.

Chcem tiež poďakovať všetkým, ktorí akýmkoľvek spôsobom prispeli k úspešnému ukončeniu práce, či už technickými poznámkami, alebo len psychickou podporou.

Abstrakt

Vývoj a nasadzovanie multiagentových systémov v reálnych podmienkach prináša mnoho problémových otázok, ktoré musí tento prístup riešiť, ak sa má stať naozajstným konkurentom pre tradičné spôsoby vytvárania softvéru. Medzi týmito otázkami je aj problematika real-time procesov riadenia techologických procesov a ťažkosti, pre ktoré je treba hľadať primerané riešenia. Za posledné roky sa vyvinulo mnoho nástrojov pre vytváranie takýchto systémov a v tejto práci predstavíme základné kritériá hodnotenia, ktoré je potrebné mať na zreteli pri výbere vhodnej agentovej platformy pre tvorbu týchto systémov.

Kľúčové slová: autonómny agent, multiagentové systémy, distribuovaná umelá inteligencia, multiagentové platformy, softvérové inžinierstvo, systémy riadenia technologických procesov.

Predhovor

Agentové technológie tvoria v súčasnosti širokú a rýchlo sa rozvíjajúcu oblasť distribuovaných počítačových systémov. Mnohí odborníci sa zhodujú v názore, že aplikácie založené na multiagentových systémoch majú obrovský potenciál v rozličných sférach reálneho života. Dôvodov tohoto rozmachu je viacero, ale najpodstatnejším je fakt, že tieto systémy zatiaľ najpresnejšie vystihujú charakter prirodzených biologických inteligentných systémov (t.j. sú obrazom prirodzenej distribúcie), a zároveň sa dokážu prblížiť požiadavkám tvorby dnešných zložitých systémov.

Už teraz nachádzajú tieto systémy naozaj široké uplatnenie a rozhodne si získavajú len veľmi ťažko nahraditeľné miesto v oblasti komplexných počítačových systémov.

Umelá inteligencia vidí možnosti využitia tejto novej paradigmy v tvorbe komplexných systémov, zložených z *interagujúcich inteligentných a autonómnych entít*, ktoré sú schopné spolupracovať pri dosahovaní jednotlivých vlastných cieľov. Softvérové inžinierstvo je po umelej inteligencii druhou oblasťou, v ktorej multiagentové systémy našli najväčšie uplatnenie. Využívajú sa v doménach ako sú napríklad telekomunikácie, riadenie priemyselných procesov a management obchodných procesov, ktoré patria medzi najzložitejšie, čo sa týka vývoja softvérových systémov. V medicíne sú už teraz tieto systémy využité v mnohých aplikáciách, veľa z nich má dokonca na starosti riešenie životne kritických úloh.

Jednou z obrovských výhod multiagentových systémov je, že nekladú žiadne obmedzenia na kompatibilitu interagujúcich agentov, ich komunikácia je teda nezávislá od návrhov jednotlivých agentov. Tento prístup je špeciálne vhodný pre tvorbu softvéru, ktorý pracuje v distribuovanom a otvorenom prostredí. Ponúka úplne nový pohľad na problematiku spätnej kompatibility dieľčích komponentov distribuovaných softvérových systémov, ktoré si vyžadujú časté rozširovanie. Distribúcia riadenia a súperenie jednotlivých komponentov o získanie kontroly zároveň zaručujú korektný beh systému aj v prípade zlyhania niektorého z komponentov a rieši problém systémového zaťaženia spôsobovaného 'zúženými hrdlami'. Toto nie je len výhodou pohodlnejšieho prístupu k informáciám a výsledkom, ale zároveň možnosťou riešenia životne kritických real-time systémov, v ktorých zlyhanie alebo preťaženie a následné spomalenie komponentu môže mať katastrofické následky. Práve takéto systémy inšpirovali vznik tohoto dokumentu.

Obsah

Kapitola 1

Úvod

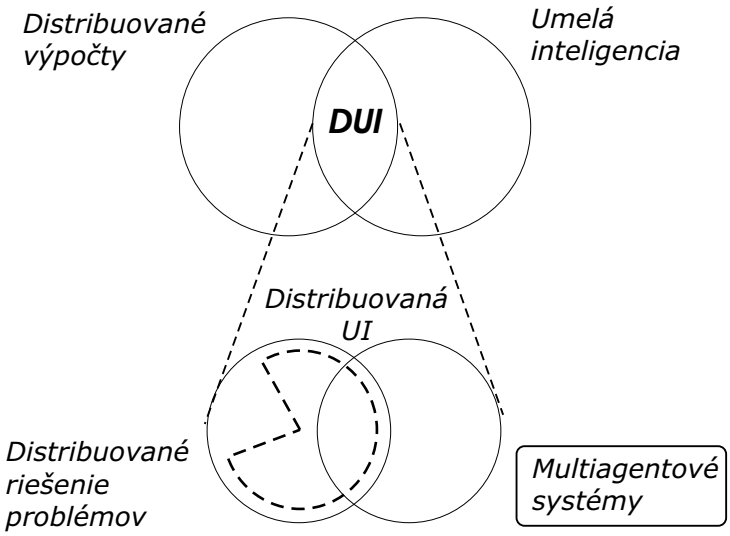
V tejto časti zľahka nahliadneme do histórie vzniku *multiagentových systémov*, a rovnako aj do širokého spektra potenciálnych aplikácií týchto systémov. Ako prvá je popísaná doména distribuovanej umelej inteligencie, z ktorej multiagentové systémy vzišli, a ďalej je v skratke rozobratý dosah tejto *novej paradigmy tvorby softvéru* na rozličné sféry počítačových systémov, ale aj reálnej ľudskej spoločnosti.

1.1 Distribuovaná umelá inteligencia

Oblasť distribuovaných výpočtov sa vyvíjala od chvíle, keď bolo umožnené využívať na riešenie problému viac ako jeden procesor a prvotným zámerom a hlavnou silou distribúcie bolo zdieľanie dát medzi procesormi, pracujúcimi na spoločnom probléme. V priebehu času sa začala v tejto problematike angažovať disciplína umelej inteligencie (UI) a vniesla do nej snahu o zdieľanie nielen dát, ale aj *kontroly*. Pozornosť sa presunula od synchronizácie a paralelizácie na nižšej úrovni k *riešeniu problémov, komunikácii a koordinácii* [18]. V druhej polovici 70-tych rokov teda vznikla *distribuovaná umelá inteligencia (DUI)* ako podoblasť umelej inteligencie, vo svojej podstate to bol prienik oblastí umelej inteligencie a distribuovaných výpočtov. Zatiaľčo vedci z oblasti klasickej umelej inteligencie sa venovali vývoju rôznych teórií, techník a systémov pre bližšie štúdium správania a uvažovania jednotlivej kognitívnej entity, cieľom tejto novej disciplíny bol vývoj *distribuovaných* riešení pre zložité systémy vyžadujúce inteligenciu.

Oblasť DUI sa v priebehu svojho vývoja rozvetvila na viaceré sféry záujmu a etablovala sa na veľmi sľubnú výskumnú disciplínu. Vo všeobecnosti môžeme jej záber rozdeliť na dve hlavné oblasti výskumu a využitia, tak ako

to vidíme na obrázku 1.1, a tými sú oblasť *multiagentových systémov* a oblasť "*ne*-multiagentových" systémov, z ktorých najväčšiu časť tvoria systémy *distribuovaného riešenia problémov (DRP)* [2].



Obrázok 1.1: Oblasti výskumu distribuovanej umelej inteligencie [31].

Nie všetky DUI systémy musia nutne obsahovať agentov. Nasledujúca definícia používa všeobecnejši pojem - entita:

> *Distribuovaná umelá inteligencia (DUI) sa zaoberá štúdiom a návrhom systémov, skladajúcich sa z niekoľkých interagujúcich entít, ktoré sú logicky a často aj priestorovo distribuované a v istom zmysle sa môžu považovať za autonómne a inteligentné. [35]*

Hlavným predmetom záujmu DRP je problematika *managementu informácií* vrámci distribuovaných systémov, konkrétne otázka *rozdeľovania úloh* na viaceré podúlohy a následnej *syntézy výsledného riešenia* z dieľčích výsledkov dosiahnutých distribuovaným výpočtom z podúloh. V multiagentových systémoch niekoľko agentov koordinuje svoje vedomosti a aktivity a uvažujú o procesoch koordinácie.

Zatiaľčo DRP systémy sú väčšinou systémy "šité na mieru" danému typu problému, a teda majú vždy vopred dané isté obmedzenia zaručujúce kompatibilitu jednotlivých spolupracujúcich entít, v multiagentových systémoch nie je zaručená žiadna kompatibilita. Rôzni agenti môžu mať úplne rozdielne

návrhy, a jednako musia byť schopní spolupracovať.

Vysvetleniu pojmu *agent* a prehľadu typológie a návrhov agentov sa venujú nasledujúce kapitoly tejto práce.

1.2 Ciele

Táto práca má za cieľ v prvej polovici ponúknuť prehľad problematiky multiagentových systémov s dôrazom na ich využitie v softvérovom inžinierstve, konkrétne pri vytváraní systémov riadenia technologických procesov.

Za posledné roky sa výskum v oblasti distribuovanej umelej inteligencie významne rozbehol, a už teraz existuje mnoho publikácií, zaoberajúcich sa problematikou riešenia problémov pomocou agentovej technológie. Prehľad, ktorý môžete nájsť v tejto práci, čerpá z mnohých prameňov najcitovanejších publikácií a snaží sa podať ucelený a komplexný pohľad na oblasť multiagentových systémov, ich miesto v disciplíne umelej inteligencie, ich využitie v procese vývoja softvéru a porovnanie s konvenčným prístupom tvorby softvéru.

Primárnym cieľom tejto práce je preskúmať oblasť systémov riadenia technologických procesov, sformulovať požiadavky na multiagentový prístup k tvorbe týchto systémov a na záver nájsť základnú množinu kritérií pre hodnotenie agentových platforiem, ktoré budú čo najpresnejšie odrážať požiadavky ukladané softvérovým inžinierstvom na využitie agentovej technológie pre tvorbu systémov riadenia procesov.

1.3 Štruktúra práce

Táto **prvá kapitola** je úvodom k práci, a zároveň úvodom do oblasti distribuovanej umelej inteligencie, z ktorej vyšli multiagentové systémy.

Druhá kapitola predstavuje pojem agent a zároveň ponúka dosť obšírny prehľad vlastností a delení agentov. V závere kapitoly sú rozobraté jednotlivé prístupy k tvorbe týchto entít.

V **tretej kapitole** je popísaná oblasť multiagentových systémov, porovnanie decentralizovaného prístupu distribuovanej umelej inteligencie a centralizovaného prístupu klasickej umelej inteligencie. Kapitola ďalej podáva pre-

hľad o základných charakteristikách multiagentových systémov a vnútrosystémovej komunikácie agentov. Na záver je vidieť výhody agentového prístupu.

Štvrtá kapitola sa venuje využitiu agentovej technológie v softvérovom inžinierstve, popisuje vlastnosti zložitých systémov a nástroje na zvládanie tejto zložitosti. Zároveň obsahuje zaujímavé porovnanie agentového a konvenčného prístupu k tvorbe softvéru a na záver uvádza niekoľko charakteristík aplikácií, pri ktorých je vhodné využiť agentovú paradigmu.

Piata kapitola definuje agentové platformy a predstavuje tri prístupy hodnotenia platforiem podľa rozličných systémov kritérií.

Posledná **šiesta kapitola** obsahuje výsledky skúmania domény systémov riadenia technologických procesov, predstavuje sformulované požiadavky na agentovo založené systémy riadenia a kritériá výberu vhodnej platformy pre tieto systémy.

Na záver je vyslovené záverečné zhrnutie práce a možné pokračovania výskumu.

Kapitola 2

Úvod do teórie agentov

Technológia budovania systémov pomocou agentov vyvolala v posledných rokoch všeobecné vzrušenie. Hlavným dôvodom je zvyšujúca sa náročnosť systémov, technologických procesov a výpočtových úloh, ktoré je potrebné zvládnuť, a tak distribuovaní agenti môžu byť považovaní za ďalší krok v evolúcii programovacích metodológií.

Na začiatku boli stroje a kompilačné jazyky. Tie sa vyvinuli do vyšších programovacích jazykov, schopných organizovať kroky programu do *podprogramov*. Ďalšie zovšeobecnenie umožnilo programátorom zoskupovať podprogramy do *knižníc* a *modulov*. Ďalšia novinka pridala pojem *objektovo-orientované programovanie*: dáta a procedúry mohli byť združené do jedinej programovej entity - *objektu*, ktorý zapuzdroval tieto programové zložky a zvyšoval tak modularitu a znovupoužiteľnosť jednotlivých komponentov. Distribuované objektové technológie, ako napr. CORBA, urobili ďalej prielom v pravidle, že každý používaný objekt musí byť uložený na lokálnom počítači; objektové knižnice mohli distribuovane poskytovať služby cez tieto "dojednávacie" technológie a samotné objekty dokonca mohli byť napísané v rôznych programovacích jazykoch, samozrejme pokiaľ používali rovnaký jazyk pre definíciu rozhrania (Interface Definition Language).

Napriek distribúcii na vysokej úrovni, objekty dokážu pracovať len na základe vzájomnej zviazanosti, pričom každý objekt má vopred naprogramované volania metód nejakých ďalších, už existujúcich objektov. Naproti tomu prichádza nová koncepcia entít, ktoré pracujú veľmi dynamicky. Keď agent narazí na problém, ktorý nevie vyriešiť, nevolá už známe metódy, ale namiesto toho sformuluje svoj problém a postúpi ho ďalej komunite agentov. Tam už je zabezpečená vzájomná interakcia a následné pridelenie problémov agentom, ktorí sú ich schopní riešiť.

2.1 Pojem *agent*

Agentové technológie vznikli ako výsledok fúzie bádania v oblasti distribuovaných výpočtov a umelej inteligencie. Výskum tejto novej domény počítačových vied nám dal za posledné roky množstvo definícií pojmu *agent*. Uvediem tu niektoré z nich, ktoré sú najčastejšie citované v odborných článkoch, a ktoré majú zopár spoločných čŕt prisudzovaných agentom - tie sa dajú nakoniec zhrnúť do veľmi príjemnej definície. Takýto prehľad ponúka tiež pojednanie Franklina a Graessera [13].

Russell, Norvig [30]: *"Agent je čokoľvek, o čom môžeme povedať, že vníma svoje prostredie cez senzory a koná v ňom cez efektory."*

Táto definícia má nedostatky. Jej význam vo veľkej miere závisí od toho, čo nazveme *prostredím*, a čo presne znamená *vnímanie* a *konanie*. Ak si zadefinujeme prostredie ako to, čo poskytuje vstupy a prijíma výstupy, vnímanie prostredia ako prijímanie vstupu a konanie ako produkovanie výstupu, potom môžeme agentom nazvať každý program.

Maes [20]: *"Autonómni agenti sú výpočtové systémy, ktoré existujú v nejakom komplexnom dynamickom prostredí, v ňom autonómne vnímajú a konajú, a tak realizujú množinu cieľov alebo úloh, pre ktoré sú určení."*

Definícia Pattie Maesovej pridáva nový prvok, a to *autonómnosť* agentov. Ale aj napriek novým obmedzeniam na prostredie môže byť ako agent stále považovaný aj program.

Hayes-Roth [14]: *"Inteligentní agenti neustále vykonávajú tri funkcie:*

- *vnímanie dynamických podmienok prostredia;*
- *konanie, ktorým ovplyvňujú podmienky prostredia;*
- *uvažovanie, kvôli interpretácii vnemov, riešeniu problémov, vyvodzovaniu výsledkov a stanoveniu akcií."*

V prípade, že nový prvok *uvažovania* agentov bude ponímaný v dostatočne širokom zmysle, potom takýto agentový systém dáva priestor rovnako pre mimovoľné akcie, ako aj pre tie vopred plánované.

Brustoloni [9]: *"Autonómni agenti sú systémy schopné autonómnej, cieľavedomej akcie v reálnom svete."*

Podľa tejto definície sa agenti musia nachádzať v reálnom svete, navyše musia byť reaktívni, t.j. schopní interakcie s prostredím.

Wooldridge, Jennings [37, 38]: *"... hardvérový alebo (častejšie) softvérový počítačový systém, ktorý spĺňa nasledovné podmienky:*

- *autonómnosť: agenti konajú bez priameho ľudského alebo iného zásahu a majú istú kontrolu nad svojimi akciami a vnútorným stavom;*
- *sociálna schopnosť: agenti interagujú s ostatnými agentami (poprípade ľuďmi) pomocou nejakého komunikačného jazyka;*
- *reaktívnosť: agenti vnímajú svoje prostredie (ktorým môže byť skutočný svet, užívateľ cez grafické užívateľské rozhranie, skupina iných agentov, Internet alebo nejaká kombinácia týchto vecí) a reagujú v príhodnom čase na jeho zmeny;*
- *pro-aktivita: konanie agentov už nie je len jednoduchou reakciou na prostredie, ale sú schopní aj cieľavedomo prevziať iniciatívu."*

Napriek rozdielnym ponímaniam agentov, je užitočné mať stručné všeobecné zhrnutie, ktoré zachytáva podstatu tejto entity v najširšom zmysle. Stále ostane priestor pre možné rozširovanie, ktoré potom vedie k vytváraniu špecifických tried agentov podľa potrieb rôznych systémov. Franklin a Graesser [13] ponúkajú ako takéto zhrnutie nasledovnú formuláciu:

> **Autonómny agent** *je sytém nachádzajúci sa v istej časti prostredia, ktorý toto prostredie vníma a v priebehu času v ňom koná za účelom splnenia svojich cieľov, a tak ovplyvňuje to, čo bude vnímať v budúcnosti.*

Táto definícia zároveň ponúka zreteľné rozlíšenie medzi programom a agentom. Agent vôbec nemusí byť programom. Môže to byť robot, alebo učiteľ. Softvéroví agenti sú z definície programami, na druhú stranu program musí spĺňať viacero kritérií, aby mohol byť klasifikovaný ako agent.

2.2 Typológie agentov

V predchádzajúcej časti tejto kapitoly sme zadefinovali všeobecnú charakteristiku agenta. Je zrejmé, že táto charakteristika je príliš všeobecná a zahŕňa

v sebe príliš veľa typov agentov. Avšak teraz je už definovanie jednotlivých podtried len otázkou pridávania jednotlivých požiadaviek na agentov. Tak je možné vytvoriť vhodné špecifické skupiny, užitočné pri rôznych typoch aplikácií. Typológia - veda o typoch - tu teda znamená štúdiu typov agentov.

Rovnako, ako je mnoho definícií agenta, máme aj mnoho rôznych taxonómií, ktoré delia agentov na jednotlivé typy. V nasledujúcich riadkoch teda nepôjde o vyslovenie všeobecného "jediného" rozdelenia, ale prehľadom rôznych delení možno hlbšie pochopiť špecifické vlastnosti a črty agentov.

Hyacinth S. Nwana [27] uvádza delenie už *existujúcich* softvérových agentov. Jedným z kritérií je *pohyblivosť (mobilita)* agentov. Podľa toho poznáme agentov:

- *statických*
- *mobilných.*

Ďalším kritériom je spôsob, akým sa agenti rozhodujú. Na základe toho rozoznávame agentov:

- *reaktívnych*
- *zvažujúcich (deliberative agents).*

Agent *zvažujúci* má vnútorný model pre symbolické uvažovanie v prostredí a uskutočňuje svoju voľbu na základe analýzy možných následkov akcie, pričom vyberie tú, ktorá je podľa jeho uvažovania z hľadiska výsledku najlepšia. Naproti tomu *reaktívny* agent neuchováva žiaden vnútorný symbolický model svojho prostredia a koná spôsobom *"vnem/reakcia"*, ktorý je jeho odpoveďou na konkrétny stav prostredia, v ktorom sa nachádza.

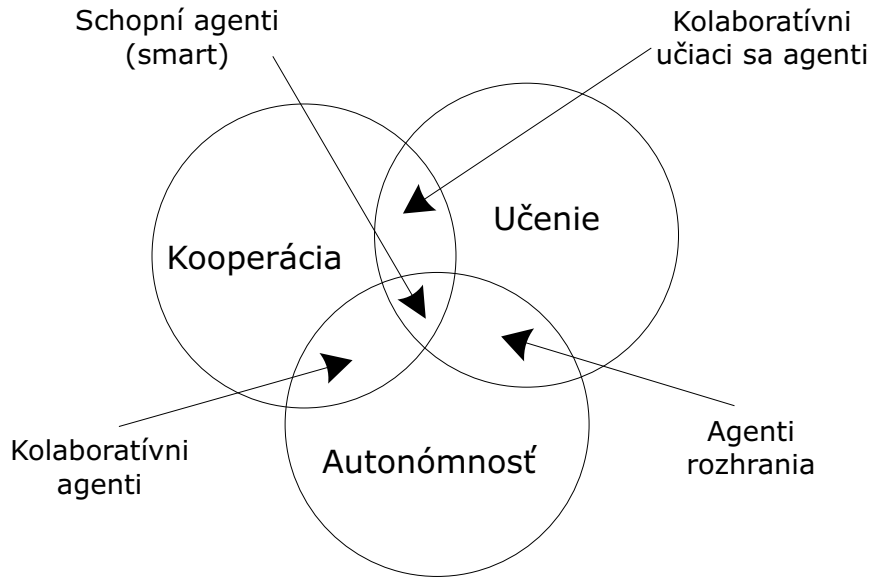
Ďalej môžeme agentov klasifikovať na základe istých primárnych *ideálnych* vlastností, ktoré by mal agent mať. Nwana [27] uvádza tri takéto vlastnosti, a to:

- *autonómnosť: agent dokáže pracovať bez ľudského vedenia; teda má osobitné vnútorné stavy a ciele a koná tak, aby tieto ciele dosiahol zastupujúc pritom užívateľa. Kľúčovou zložkou autonómnosti agenta je jeho pro-aktivita, t.j. schopnosť prevziať iniciatívu a nekonať len ako odpoveď na podmienky prostredia [38].*

- *učenie: agent sa učí popri reagovaní/interagovaní s prostredím. Agenti patria do oblasti umelej inteligencie a učenie je jednou z najpodstatnejších zložiek inteligencie. U agentov môže mať učenie formu napríklad zvyšovania výkonu v priebehu času.*

- *kooperácia: spolupráca s inými agentami je prvoradá - je to hlavný predpoklad pre využitie viacerých agentov namiesto len jediného. Aby mohli agenti spolupracovať, musia mať tzv.* sociálnu schopnosť, *t.j. musia byť schopní komunikovať medzi sebou, poprípade s ľuďmi pomocou nejakého komunikačného jazyka [38].*

Tieto vlastnosti samozrejme netvoria nutnú ani postačujúcu množinu vlastností ideálneho agenta, je to len návrh, ktorý vyplynul z predchádzajúceho výskumu a poskytol zaujímavú a užitočnú klasifikáciu agentov, ktorú ilustruje nasledujúci obrázok.



Obrázok 2.1: Typológia agentov podľa ich vlastností. (Nwana, 1996)

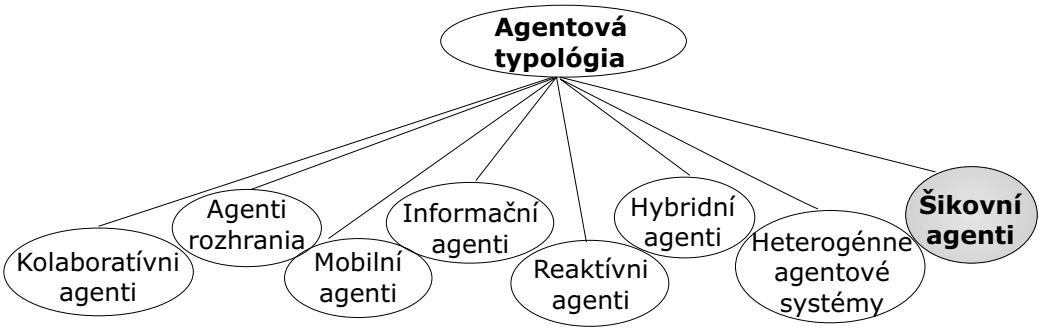
Toto rozdelenie ale nie je definitívne, takže napríklad u *kolaboratívnych* agentov sa kladie väčší dôraz na kooperáciu a autonómnosť, ale to neznamená, že sa v žiadnom prípade nebudú v priebehu času aj učiť. S ostatnými typmi je to analogické.

Ďalšia klasifikácia určuje agentov podľa *rôl*, v ktorých vystupujú - napríklad *world wide web informační agenti.* Tento typ agentov vo všeobecnosti

pomáha spravovať obrovské množstvá informácií vo WAN sieťach. Preto sa niekedy nazývajú tiež *internetoví agenti*.

Ako posledných spomína Nwana aj *hybridných* agentov, ktorí v sebe spájajú dve alebo viac agentových filozofií.

Výsledné delenie agentov ako ho uvádza Nwana môžeme vidieť na nasledujúcom obrázku.



Obrázok 2.2: Celková klasifikácia softvérových agentov. (Nwana, 1996)

Iné delenie ponúka Decker [12] vo svojej práci. Uvádza štyri dimenzie agentov:

- *granularita (silní/slabí agenti)*
- *heterogenita znalostí (redundantné/špecializované znalosti)*
- *metódy distribuovaného riadenia (benevolencia/spolupráca, tím/hierarchia)*
- *komunikačné možnosti (blackboard/správy, obsah)*

Komunikačné možnosti sú v tomto ponímaní čiste aspektom agenta, keďže sa jedná o schopnosti a úroveň komunikačných prostriedkov jednotlivých agentov a nie o komunikačný protokol vrámci multiagentového systému.

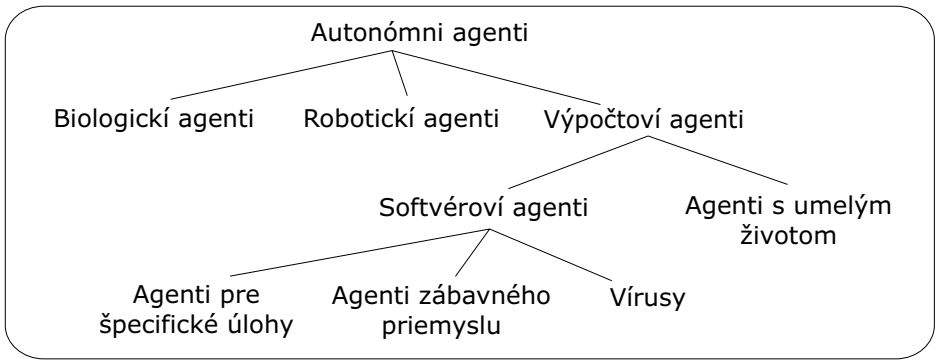
Posledné tu popísané rozdelenie agentov pochádza od Franklina a Graessera [13]. Uvádzajú nasledovný zoznam vlastností agentov:

- *reaktívny (vnímanie a konanie): reaguje v príhodnom čase na zmeny prostredia,*

- *autonómny: kontroluje svoje vlastné konanie,*
- *cieľovo-orientovaný (pro-aktívny): jeho akcie nie sú len reakciami na prostredie,*
- *existujúci v čase: je to súvisle bežiaci proces,*
- *komunikatívny (sociálna schopnosť): komunikuje s ostatnými agentami, poprípade ľuďmi,*
- *učiaci sa (adaptívny): mení svoje správanie na základe predchádzajúcich skúseností,*
- *mobilný: schopný sa presúvať z jedného stroja na iný,*
- *flexibilný: akcie nie sú explicitne naprogramované,*
- *charakter: vierohodná "osobnosť" a emocionálny stav.*

Agenti teraz môžu byť vhodne rozdelení podľa podmnožín vlastností, ktoré vykazujú. Podľa všeobecnej definície uvedenej na konci časti **2.1**, každý agent musí spĺňať prvé štyri vlastnosti. Pridávaním niektorých z ďalších vlastností vzniká prirodzene (na základe množinovej inklúzie) *hierarchické rozdelenie agentov*. Napríklad mobilní učiaci sa agenti sú podmnožinou *mobilných agentov*, atď.
Franklin a Graesser [13] ďalej uvádzajú klasifikáciu na základe analógie s prirodzeným delením biologických druhov do ríší, kmeňov, tried, atď. Takto vytvorili hierarchické rozdelenie agentov, ako je uvedené na obrázku 2.3. Hlbšie delenie potom môže byť určené podľa riadiacej architektúry, prostredia (data-báza, súborový systém, sieť, Internet), jazyka (v ktorom sú napísané) alebo aplikácie.

Existuje ešte mnoho iných typológií, a ich rozoberanie by bolo nad rámec tohoto dokumentu. Oblasť skúmania vlastností agentov je široká a otvorená, a samozrejme plná rôznych prístupv. Môžeme napríklad klasifikovať agentov podľa úloh, ktoré vykonávajú, alebo podľa ich riadiacej architektúry (agentovým architektúram sa venuje časť **2.3**). Niektoré ďalšie delenia môžu súvisieť s prostredím, v ktorom sa agenti nachádzajú, napríklad *softvéroví agenti* v protiklade k *agentom-robotom (umelý život)*.



Obrázok 2.3: Taxonómia agentov podľa Franklina a Graessera.

2.3 Agentové architektúry

V predchádzajúcich častiach sa tento dokument snažil ponúknuť predstavu o pojme *agent* a tiež stručný prehľad vlastností týchto entít, tak ako ich vnímajú rôzni odborníci zaoberajúci sa touto oblasťou. V tejto časti zájdeme zasa o čosi ďalej a priblížime si niektoré postupy vytvárania počítačových systémov, ktoré spĺňajú tieto vlastnosti. Budeme sa teda zaoberať *agentovými architektúrami*.

Na otázku, čo to je agentová architektúra, odpovedala Pattie Maes nasledovne:

> *... špecifická metodológia tvorby agentov. Určuje, ako môžeme agentov rozložiť na konštrukciu tvorenú množinou čiastkových modulov, a ako by mali tieto moduly vzájomne interagovať. Celková množina modulov a ich interakcií musí poskytovať odpoveď na otázku, ako určujú vnímané dáta a momentálny vnútorný stav agenta jeho akciu a jeho vnútorný stav v budúcnosti. Architektúra zahŕňa techniky a algoritmy, ktoré túto metodológiu podporujú. [20]*

Kaelbling považuje agentovú architektúru za:

> *... špecifický súbor softvérových (alebo hardvérových) modulov, spravidla označovaných rámčekmi so šípkami naznačujúcimi dátový a riadiaci tok medzi modulmi. Z abstraktnejšieho pohľadu je architektúra všeobecná metodológia tvorby špecifických modulárnych dekompozícií určených pre špecifické úlohy. [16]*

Mataric vo svojej práci uvádza:

Architektúra poskytuje množinu princípov pre zostavenie riadiacich systémov. Popri stanovovaní štruktúry ukladá zároveň aj obmedzenia spôsobov, ktorými môžu byť problémy riadenia riešené. [22]

Za základné typy architektúr sa považujú nasledovné:

- *deliberatívne architektúry,*
- *reaktívne architektúry,*
- *hybridné architektúry.*

Pomerne novým prístupom v tejto oblasti sa javia byť tzv.

- *"behavior-based" architektúry,*

ktoré sú rozšírením reaktívnych architektúr a spadajú niekam medzi extrémy čisto reaktívneho a čisto deliberatívneho (*"planner-based"*) prístupu.

V nasledujúcich odsekoch budú tieto typy architektúr podrobnejšie popísané.

2.3.1 Klasický prístup: deliberatívne architektúry

Klasický deliberatívny prístup je prístupom *zhora-nadol* a berie agentov ako špecifický typ znalostných (*"knowledge-based"*) systémov. Táto paradigma sa nazýva *symbolická umelá inteligencia.*

Deliberatívny agent (resp. agent s deliberatívnou architektúrou) využíva všetky informácie získané pomocou senzorov a všetky interne uložené vedomosti pri určovaní akcie, ktorú musí vykonať. Rozhoduje sa na základe logických (poprípade pseudo-logických) úvah.

Uvažovanie je spravidla realizované formou plánovania, zahŕňa hľadanie postupností akcií zodpovedajúcich danému stavu okolností a ich možných následkov. Toto je výpočtovo náročná úloha, ktorá sa môže ukázať nevhodnou, ak je priestor možných stavov príliš veľký, alebo ak sa o ňom nedá získať kompletná informácia, čo sú typické situácie, keď sa jedná o systémy nachádzajúce sa v reálnom svete (*"stelesnené" systémy*).

Plánovanie navyše vyžaduje, aby agent mal k dispozícii internú reprezentáciu sveta, ktorá mu umožňuje analyzovať následky akcií v rôznych stavoch. Ak je agent súčasťou nejakého multi-agentového systému, plánovanie zahŕňa aj schopnosť predpokladať reakcie ostatných, k čomu je potrebný ich model.

Wooldridge a Jennings [38] uvádzajú nasledovné problémy, ktoré treba vyriešiť v prípade využitia tohoto konceptu:

- ako preložiť reálny svet do presného adekvátneho symbolického popisu tak rýchlo, aby bol daný popis ešte využiteľný;
- ako symbolicky reprezentovať informáciu o zložitých entitách a procesoch z reálneho sveta a ako dosiahnuť, aby agent spracoval túto informáciu tak rýchlo, aby bol výsledok ešte využiteľný.

Prvý problém viedol k rozvoju práce v oblastiach obrazu, porozumenia reči, učenia a iných, zatiačo druhý viedol k práci v oblasti reprezentácie znalostí, automatizovanom uvažovaní, atuomatickom plánovaní, atď. Napriek tomu mnoho odborníkov uznáva, že tieto problémy zďaleka nie sú vyriešené.

Deliberatívny prístup je užitočný v prípadoch, keď je cena za vykonanie nesprávnej akcie veľmi vysoká, napríklad keď sa jedná o riskantné prostredie. Umožňuje naprogramovať *všeobecný postup* na určovanie riešení pre celé triedy problémov, teda môže byť veľmi efektívny pri riešení nových problémov.

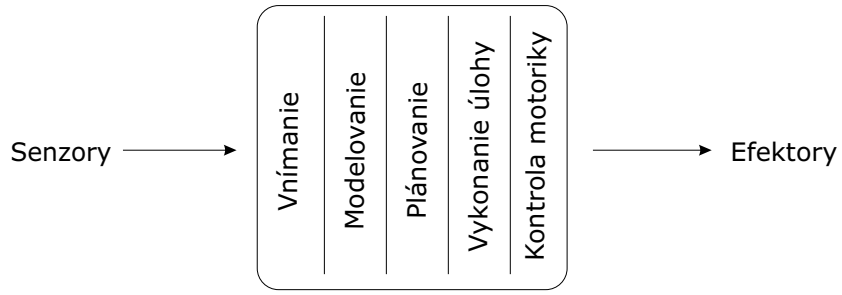
2.3.2 Alternatívny prístup: reaktívne architektúry

Problémy, ktoré sa spájajú s klasickým prístupom symbolickej umelej inteligencie, viedli niektorých výskumníkov k vývoju tzv. *reaktívnych* architektúr. Reaktívna architektúra neuchováva žiaden symbolický model sveta a nevyužíva symbolické uvažovanie, naproti tomu ukotvuje riadiacu stratégiu agenta do súboru predprogramovaných dvojíc *podmienka-akcia* s minimálnym vnútorným stavom.

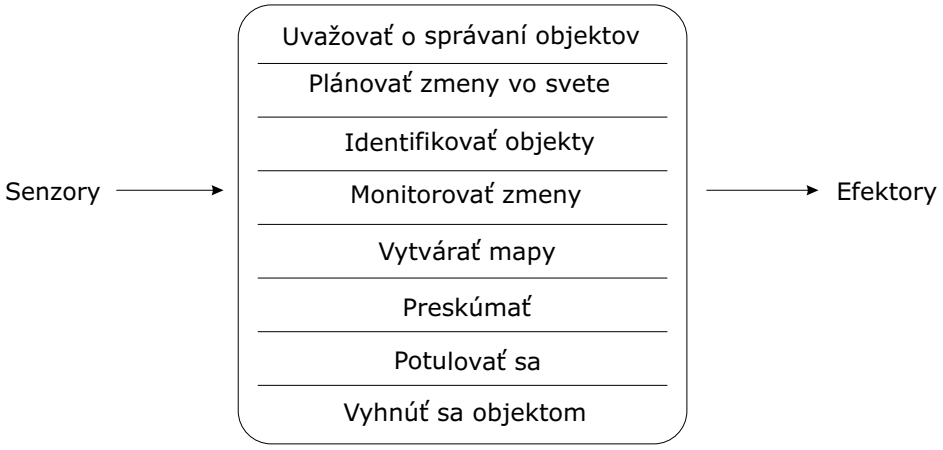
Pravdepodobne najväčším kritikom klasických prístupov umelej inteligencie bol za posledné obdobie Rodney Brooks, ktorý vo svojej práci rozvil oblasť reaktívneho prístupu a vo svojej práci načrtol novú *subsumpčnú* architektúru [5].

Klasický prístup budovania riadiaceho systému agentov rozdeľoval problém do série funkčných častí, ako je zobrazené na obrázku 2.4. Brooks využil pre svoju architektúru nový prístup, a to dekompozíciu podľa správania vyvíjaného na dosahovanie cieľov úloh (viď obrázok 2.5). Pri dekompozícii podľa funkcie sú jednotlivé komponenty explicitne naprogramované a následne pospájané do jedného riadiaceho systému. Nový prístup prezentovaný Brooksom umožňuje využiť radikálne odlišné stratégie pri implementácii na úrovni hardvéru.

Subsumpčná architektúra je hierarchiou úlohovo-orientovaných "správaní". Každé správanie súperí s ostatnými o uplatnenie kontroly nad robotom. Nižšie vrstvy reprezentujú primitívnejšie správania (ako napríklad *vyhýbanie sa objektom*) a majú väčšiu prioritu oproti vyšším vrstvám v hierarchii. Výsledné systémy boli až extrémne jednoduché, čo sa týka výpočtovej zložitosti a Brooks prezentoval roboty vykonávajúce veľmi pôsobivé úlohy.



Obrázok 2.4: Tradičná dekompozícia riadiaceho systému mobilného robota do funkčných modulov. (Brooks, 1986)



Obrázok 2.5: Dekompozícia riadiaceho systému mobilného robota podľa konania. (Brooks, 1986)

Počas svojej práce v nasledujúcich rokoch ([6, 7, 8]) sformuloval Brooks tieto tri kľúčové tvrdenia:

1. Inteligentné správanie môže byť utvárané aj *bez* explicitných reprezentácií ako ich ponúka symbolická umelá inteligencia.

2. Inteligentné správanie môže byť utvárané aj *bez* explicitného abstraktného uvažovania ako ho ponúka symbolická umelá inteligencia.

3. Inteligencia je *emergentná* vlastnosť určitých komplexných systémov.

Brooks zároveň identifikoval dve základné myšlienky, ktoré prenikali jeho výskum:

1. Situovanosť a stelesnenie: 'skutočná' inteligencia je situovaná vo svete a nie v nehmotných systémoch ako sú dokazovače alebo expertné systémy.

2. Inteligencia a emergencia: 'inteligentné' správanie vzniká ako výsledok interakcie agenta s prostredím. Inteligencia je skôr 'otázkou vkusu pozorovateľa' a nie nejakou vrodenou, izolovanou vlastnosťou.

Rozvoj tohoto prístupu tvorby riadiacich systémov podnietilo najmä poznanie, že mnohé prostredia sú do istej miery zložené z rutín a tým pádom sa zdĺhavé procedúry uvažovania ukazujú byť zbytočnými.

2.3.3 Hybridné architektúry

Hybridný prístup si osvojuje najlepšie aspekty z deliberatívnych a reaktívnych architektúr: pokúša sa kombinovať 'real-time' odpoveď reaktívneho prístupu s racionálnosťou a optimálnosťou deliberatívneho prístupu.

Zrejmým riešením je postaviť agenta, ktorý v sebe spája dva (alebo viac) subsystémov:

- *deliberatívny subsystém, ktorý obsahuje smbolický model sveta a ktorý plánuje a uvažuje systémom symbolickej UI;*

- *reaktívny subsystém, ktorý je schopný reagovať na udalosti z prostredia bez zapojenia zložitého uvažovania.*

Reaktívny subsystém má často určitú prioritu pred deliberatívnym, aby mohol agent poskytnúť rýchlu odozvu na dôležité udalosti.

Tento spôsob konštrukcie prirodzene vedie k tzv. "vrstvenej" architektúre (*layered architecture*), v ktorej sú jednotlivé riadiace subsystémy daného agenta hierarchicky usporiadané. V tejto hierarchii riešia vyššie vrstvy daný problém na zvyšujúcej sa úrovni abstrakcie. Takže napríklad najnižšia vrstva môže len jednoducho priamo mapovať dáta získané zo senzorov na výstup efektorov, zatiaľčo najvyššia vrstva sa zaoberá sledovaním dlhodobých cieľov.

Jednou z možných aplikácii tohoto prístupu je *trojvrstvová architektúra*, ktorá má na najnižšej úrovni reaktívny subsystém, potom jeden prechodný a na najvyššej úrovni je deliberatívny subsystém.

Pri týchto architektúrach ale ostáva kľúčovým problémom návrh rámca, ktorý by riadil všetky interakcie medzi jednotlivými vrstvami danej architektúry.

2.3.4 "Behavior-based" architektúry

Behavior-based riadenie je inšpirované biológiou. Názov "behavior-based" je odvodený od základného konceptu, okolo ktorého je teória tohoto prístupu vybudovaná - z anglického slova *behavior*, čo znamená "správanie". Správania sú pozorovateľné schémy aktivít, ktoré vyplývajú z interakcie agenta s prostredím (v ktorom sa môžu potenciálne nachádzať ďalší agenti).
Takéto systémy sú budované spôsobom *zdola-nahor*, začínajúc množinou správaní dôležitých pre prežitie agenta a postupne pokračujúc až ku správaniam určujúcim komplexnejšie schopnosti. Správania sú inkrementálne pridávané, až pokiaľ systém nedosahuje želané vlastnosti a schopnosti. Výsledkom je hierarchická štruktúra systému, ktorá je veľmi podobná reaktívnej *subsumpčnej* architektúre.

V literatúre sa behavior-based architektúry často priradzujú k reaktívnym architektúram, avšak Maja Mataric vo svojej práci ([22, 23, 24]) tvrdí, že medzi týmito dvoma prístupmi existujú základné odlišnosti. Behavior-based architektúry zahŕňajú niektoré vlastnosti reaktívnych architektúr a obyčajne obsahujú reaktívne komponenty, zároveň však ich výpočet nie je obmedzený na vyhľadanie a vykonanie jednoduchého mapovania *"senzor-efektor"*.

Každý model správania obyčajne obsahuje pravidlá pre transformáciu vstupných dát, získaných zo senzorov alebo od iných správaní, na výstupné informácie pre efektory alebo iné správania. Behavior-based systémy teda môžeme chápať ako štruktúrované siete vzájomne interagujúcich správaní (tzv. *behaviorálne siete*). Na základe svojho vstupu sa dané správanie môže aktivovať na úroveň, ktorá bola vstupom určená. Úrovne aktivácie sú formulované ako podmienky, ktoré musia byť splnené, aby sa dané správanie začalo prejavovať. Rozlišujeme dva základné typy podmienok:

- *Podmienky prostredia: aktivujú správanie na základe aktuálneho stavu prostredia.*
- *Podmienky sekvencie: zvyčajne sú definované pre konkrétnu úlohu a najčastejšie sú reprezentované výstupmi iných správaní. Keďže je možné*

zreťazením vytvoriť sekvenciu jednotlivých správaní (môže byť vnímaná aj ako jednoduchá stratégia, alebo plán), konkrétne správanie sa aktivuje až vtedy, keď nasleduje v aktuálne spracovávanej sekvencii. To sa obyčajne realizuje odovzdaním aktivity od jedného správania druhému.

Vo všeobecnosti nemajú tieto systémy centrálnu reprezentáciu, s ktorou pracuje jeden alebo viac uvažujúcich mechanizmov. Namiesto toho sa zvyčajne spoliehajú na rôzne formy distribuovaných reprezentácií a vykonávajú na nich distribuované výpočty. V niektorých prípadoch tieto reprezentácie nie sú statické, manipulovateľné štruktúry, ale aktívne procedurálne procesy [7, 21]. Takže agent v prípade potreby môže plánovať, ale namiesto centralizovaného plánovača (deliberatívny prístup, viď časť **2.3.1**) využíva sieť komunikujúcich správaní. Táto odlišnosť v reprezentácii so sebou prináša významné výpočtové a výkonnostné dôsledky.

Vlastnosti behavior-based systémov zhŕňa Mataric [22] takto:

- *správania sú relatívne jednoduché a sú do systému pridávané inkrementálne;*
- *vykonávanie správaní nie je serializované;*
- *správania sú náročnejšie na čas, ako obyčajné atomické akcie daného agenta;*
- *správania interagujú s ostatnými správaniami cez svet a nie interne cez systém.*

Behavior-based systémy dominujú riadeniu multi-agentových systémov, pretože prístup využívajúci súbor správaní vrámci systému sa ukazuje byť pri týchto systémoch výhodný; vedie k robustnému a adaptívnemu skupinovému správaniu. Vo všeobecnosti sa tento typ riadenia hodí pre systémy nachádzajúce sa v prostredí, v ktorom sa odohrávajú významné dynamické zmeny, a teda rozhodujúca je rýchla odpoveď, ale zároveň je dôležitá schopnosť plánovania do budúcnosti a vyhýbanie sa predchádzajúcim chybám.

Kapitola 3

Multi-agentové systémy

Keď zvolíme agentovo-orientovaný pohľad na svet, veľmi skoro bude jasné, že na modelovanie reálnych problémov nebude stačiť jeden agent. Kapacita inteligentného agenta je obmedzená jeho vedomosťami, výpočtovými prostriedkami a stanoviskom k problému[1]. Táto obmedzenosť inteligentných entít viedla k vytvoreniu systémov, v ktorých viacero týchto entít vzájomne interaguje.

Katia Sycara uvádza vo svojej práci nasledovné zdôvodnenie multiagentového prístupu:

> *Najsilnejšími prostriedkami na zvládanie zložitosti problému sú modulárnosť a abstrakcia. Multiagentové systémy (MAS) ponúkajú modulárnosť. Ak je doména problému obzvlášť rozsiahla, komplikovaná alebo nepredvídateľná, potom jediný spôsob ako ju zvládnuť je vytvoriť niekoľko funkčne špecifických a (takmer) modulárnych komponentov (agentov), ktoré sa špecializujú na riešenie určitého aspektu problému. Táto dekompozícia umožňuje každému agentovi využiť najvhodnejší spôsob riešenia jeho problému. Keď sa vyskytnú vzájomne závislé problémy, agenti v systéme musia zjednotiť svoje úsilie tak, aby sa podarilo závislosti zvládnuť. [32]*

Navyše, reálne problémy často zahŕňajú distribuované, otvorené systémy. Tieto systémy majú nasledovné črty: ich komponenty sa nedajú dopredu určiť; môžu sa v priebehu času meniť; a môžu sa skladať z heterogénnych agentov implementovaných rôznymi ľuďmi, v rôznom čase a s použitím rôznych nástrojov a techník. Efektívne riešenie problémov v takýchto prostrediach teda vyžaduje, aby boli agenti schopní interagovať a koordinovať svoju prácu

[1] Agent nemusí mať nutne kompletný obraz o probléme, dokáže ho riešiť len zo svojho pohľadu. To môže ovplyvniť optimalitu a správnosť riešenia.

s ostatnými. Takáto funkcionalita si vyžaduje techniky založené na *vyjednávaní*, ktoré sú doménou disciplíny multiagentových systémov.

Táto časť distribuovanej umelej inteligencie je však relatívne nová, existuje vpodstate len od 80-tych rokov, preto je veľmi obtiažne poskytnúť celistvý a jednotný obraz o výskumoch a teóriách v oblasti MAS. Napriek tomu sa v nasledujúcej časti pokúsime urobiť si o tejto oblasti aspoň základný obraz. Budeme sa venovať charakteristikám a výhodám týchto riešení, ich základnými kategorizáciami a pojmami, ktoré s touto oblasťou súvisia.

3.1 Jedno-agentové vs. multiagentové systémy

Predtým než začneme podrobnejšie skúmať a kategorizovať multiagentové systémy, nesmieme zabudnúť na ich najsamozrejmejšiu alternatívu: centralizované jednoagentové systémy. Centralizovaný systém obsahuje jedného agenta, ktorý robí všetky rozhodnutia, a ak sú v systéme prítomní viacerí agenti, tak slúžia len ako vzdialené pridružené zariadenia pod kontrolou centrálneho riadiaceho agenta.

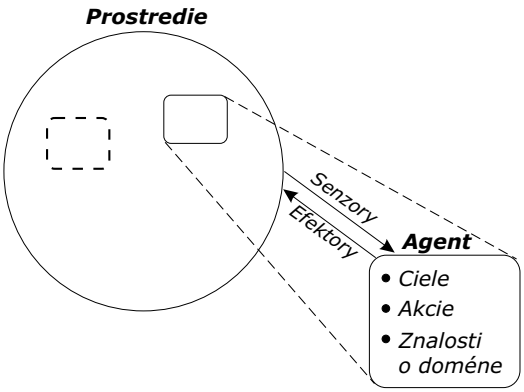
3.1.1 Jedno-agentové systémy

Na prvý pohľad by sa mohlo zdať, že centralizované systémy pozostávajúce vo svojej podstate len z jediného autonómneho agenta[2] sú jednoduchšie, avšak ak sa jedná o komplikovanejšiu problémovú doménu, opak býva pravdou. Distribúcia kontroly medzi viacerých agentov zároveň umožňuje, aby bol každý agent jednoduchší - žiaden nemusí byť schopný samostatne vyriešiť danú úlohu.

Vo všeobecnosti agent v centralizovanom systéme modeluje seba, prostredie a vzájomné interakcie. Je to nezávislá entita s vlastnými cieľmi, akciami a znalosťami a nerozoznáva v prostredí žiadne ďalšie takéto entity. Takže aj keď sa v prostredí nachádzajú iní agenti, sú považovaní len za súčasť prostredia, nie sú modelovaní ako autonómne entity. Jedinou autonómnou entitou, ktorú agent modeluje, je on sám, tak ako to vidíme na obrázku 3.1.

Vo svojej podstate spadajú jednoagentové systémy do oblasti klasickej umelej inteligencie, keďže neobsahujú prvok distribuovanej kontroly, avšak niektorí autori argumentujú, že tieto systémy sú intuitívne najjednoduchším modelom multiagentového systému [31].

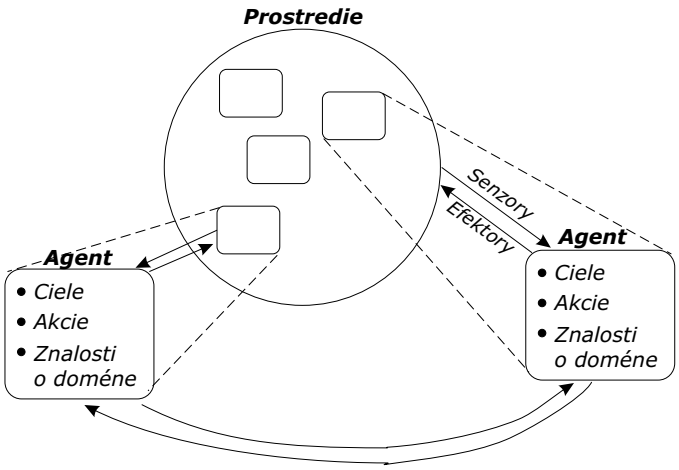
[2]Ako už bolo spomenuté vyššie, centralizovaný systém sa môže skladať z viacerých výpočtových entít, z ktorých sa však len jedna dá klasifikovať ako autonómny agent, keďže zvyšné nevykonávajú žiadne samostatné akcie; sú prítomné len ako výpočtová sila.



Obrázok 3.1: Všeobecný model jednoagentového sytému [31].

3.1.2 Multiagentové systémy

Multiagentové systémy sa skladajú z viacerých autonómnych entít, každá z nich modeluje seba a aj tie ostatné ako entity s cieľmi, akciami a znalosťami. V úplne všeobecnom modeli decentralizovaného systému môže byť priama komunikácia medzi jednotlivými agentami. Aj keď by sme mohli takúto komunikáciu považovať za podnet z prostredia, bez ujmy na všeobecnosti budeme priamu interakciu agentov modelovať oddelene (obrázok 3.2).



Obrázok 3.2: Všeobecný model multiagentového sytému [31].

Z hľadiska agenta ako jednotlivca je najvýznamnejším rozdielom medzi jednoagentovými a multiagentovými systémami fakt, že dynamika prostredia môže byť ovplyvňovaná akciami ostatných agentov. To prakticky znamená,

že okrem implicitnej neistoty zahrnutej v doméne, ďalším zdrojom neistôt pre agenta sú nepredvídateľné akcie zámerne vykonávané ostatnými agentami.

Na obrázku 3.2 vidíme, že agenti sú časťami prostredia, ale zároveň sú modelovaní ako autonómne oddelené entity.

3.2 Základné charakteristiky

V predchádzajúcej časti sme oddelili centralizované systémy od všeobecného modelu multiagentových systémov. Teraz teda môžeme sformulovať základné charakteristiky multiagentových systémov:

(1) každý agent má nekompletné informácie alebo schopnosti na vyriešenie problému, a teda má obmedzené stanovisko;

(2) neexistuje glóbalne riadenie systému;

(3) dáta sú decentralizované;

(4) výpočet je asynchrónny.

Keď sa jedná o decentralizované distribuované dáta, tie sa nachádzajú v informačných systémoch, ktoré sú rozsiahle a zložité v nasledujúcom zmysle: (1) môžu byť geograficky distribuované, (2) môžu mať mnoho komponentov, (3) môžu mať obrovskú kapacitu, aj čo sa týka konceptov, aj čo sa týka množstva informácií o každom koncepte a (4) môžu mať široký záber, t.j. pokrývať veľkú časť významnej domény.

Jedným z najzákladnejších konceptov, ktorý je nevyhnutnou súčasťou multiagentových systémov, je *prostredie*. V nasledujúcom odseku si priblížime základné charakteristiky agentových prostredí a ich kategorizáciu.

3.2.1 Prostredie

Prostredie poskytuje výpočtovú infraštruktúru pre vzájomné interakcie agentov, zahŕňa komunikačné protokoly (umožňujúce agentom, aby rozumeli vzájomným správam) a protokoly pre výmenu informácií (určujúce scenáre vzájomných konverzácií, napríklad vyjednávania a pod.). Prostredia, ktoré sú vhodné pre aplikáciu multiagentového prístupu, sú zvyčajne otvorené a nemajú centralizovaný návrh.

Nasledujúci prehľad vlastností čerpá z práce Huhnsa a Stephensa v [35] a je vcelku užitočný pre pochopenie vplyvu prostredia na konanie jednotlivých agentov a napokon aj systému agentov ako celku.

- *Pozorovateľnosť*

Aby mohol byť agent naozaj považovaný za agenta, časť prostredia, v ktorom sa nachádza, musí byť pozorovateľná (konkrétne časť relevantná pre uvažovanie nad prípadnou akciou). V niektorých prípadoch (obzvlášť ak ide o softvérových agentov) je pozorovateľné celé prostredie, ale aj keď je to pre agenta výhodné, môže to nastať len v relatívne jednoduchých prostrediach.

- *Poznateľnosť*

Táto vlastnosť hovorí o tom, do akej miery dokáže agent modelovať dané prostredie, iné slovo pre túto črtu by mohlo "pochopiteľnosť".

- *Predpovedateľnosť*

Do akej miery dokáže agent predpokladať nasledujúci stav prostredia. Táto vlastnosť veľmi úzko súvisí s dynamickosťou prostredia.

- *Kontrolovateľnosť*

Hovorí o tom, do akej miery dokáže agent prostredie zmeniť.

- *Historickosť*

Táto vlastnosť hovorí o tom, či budúce stavy prostredia závisia od celkovej histórie, alebo len od aktuálneho stavu.

- *Teleologickosť*

Vlastnosť, ktorá hovorí o tom, či sa v prostredí nachádzajú nejaké *cieľavedomé* časti, napríklad ďalší agenti.

Prostredie, ktorého nasledujúci stav závisí výlučne od aktuálneho stavu a od akcie, ktorú vykoná agent, sa nazýva *deterministické*. Ak sa v prostredí vyskytne nejaký zásah zvonka alebo prvok neistoty[3], potom toto prostredie nazývame *stochastické*. Všimnime si, že stochastické prostredie, ktoré je len čiastočne pozorovateľné, sa môže agentovi javiť ako deterministické.

Prostredie, ktorého nasledujúci stav závisí výlučne od aktuálneho stavu a akcií viacerých agentov, sa nazýva *strategické*.

- *"Real-time"-ovosť*

[3]Črta "neistoty" je popísaná v nasledujúcom odseku.

Hovorí o tom, či sa prostredie mení v čase, keď sa agent rozhoduje o postupe svojho konania. *Statické* prostredie, ako napovedá samotný názov, je prostredie, ktorého stav sa počas agentovho uvažovania nemení. Inými slovami, jediné zmeny prostredia sú tie, ktoré spôsobuje samotný agent svojím konaním. *Dynamické* prostredie sa mení, takže ak agent nezareaguje včas, je to považované za rozhodnutie nevykonať žiadnu akciu.

- *Počet možných stavov prostredia*

Množina možných stavov prostredia môže mať konečný alebo nekonečný počet prvkov. Prostredie s nekonečným počtom možných stavov nazývame *spojité*. *Diskrétne* prostredie má konečný počet možných stavov, avšak ak je tento počet dostatočne veľký, potom sa prostredie stáva z pohľadu agenta prakticky *spojitým*.

3.2.2 Pojem "neistoty" v prostredí

Neistota je významným faktorom, ktorý ovplyvňuje spoločenský vývoj v prirodzených aj umelo vytvorených prostrediach. V nedávnej publikácii [1] sa spomínajú tri aspekty neistoty prostredia:

(1) *neistota pochádzajúca z prostredia:* zmeny prostriedkov v prostredí;

(2) *vnímaná neistota:* zmeny distribúcie prostriedkov tak, ako ich vníma agent;

(3) *efektívna (skutočná) neistota:* zmeny prostriedkov, ktoré agent skutočne využíva.

V dynamickom prostredí, ako je napríklad reálny svet, je mnoho zdrojov neitoty. Nasledujúce príklady sú niektorými z nich:

- môže nastať množstvo udalostí, nad ktorými agent nemá kontrolu; niektoré z týchto udalostí môžu mať priamy alebo nepriamy vplyv na agentov aktuálny plán, môžu sťažiť jeho vykonávanie, znížiť pravdepodobnosť úspechu alebo zvýšiť jeho cenu;

- kroky, ktoré agent vykonáva podľa aktuálneho plánu, nemusia byť úplne úspešné pri dosiahnutí svojich cieľov, a tak môžu znížiť zisk z dosahovania cieľa;

- agent si nie je istý informáciami, ktoré získava z prostredia; táto neistota môže byť spôsobená nedostatkami senzorov alebo jednoducho nejakou implicitnou neistotou obsiahnutou v informácii.

Určitý provk neistoty je súčasťou takmer každého zložitejšieho prostredia a ovplyvňuje výkonnosť jednotlivých agentov, aj celých systémov.

3.3 Komunikácia agentov

V multiagentových systémoch je jednou z najdôležitejších záležitostí aj schopnosť agentov komunikovať. Táto schopnosť sa skladá z vnímania (prijímanie správ) a vykonávania akcií (posielanie správ). U čiste počítačového agenta je možné, že prijímanie a posielanie správ budú jediné schopnosti agenta, čo sa týka vnímania a konania.

3.3.1 Spôsoby komunikácie

V [10] autori uvádzajú nasledovné rozdelenie spôsobov komunikácie:

- *Interakcia prostredníctvom prostredia*

Tento spôsob je najjednoduchší a najviac obmedzený spôsob komunikácie. Komunikačným médiom je samotné prostredie a neexistuje žiadna explicitná forma vzájomnej kooperácie. Tento spôsom sa niekedy nazýva aj "kooperácia bez komunikácie".

- *Interakcia prostredníctvom senzorov*

Agenti spolu môžu komunikovať aj tak, že sa navzájom snímajú senzormi, pričom sa však opäť nejedná o žiadnu explicitnú formu komunikácie. Tento spôsob predpokladá schopnosť rozoznať agenta od iných objektov v prostredí. Senzorová interakcia je často nevyhnutná pri riešení problému modelovania iných agentov.

- *Interakcia ako komunikácia*

Tento tretí spôsob využíva explicitné formy komunikácie medzi agentami pomocou správ. Architektúry, ktoré poskytujú komunikačnú infraštruktúru pre posielanie správ, sa nijak podstatne nelíšia od klasických komunikačných sietí. Využívajú komunikačné protokoly a protokoly pre interakciu, avšak tu je treba dbať na problém, že agenti využívajúci pre komunikáciu istý špecifický protokol nebudú schopní komunikovať s agentami využívajúcimi nejaké iné protokoly. Preto boli snahy o aspoň čiastočné zjednotenie komunikácie a tieto snahy vyústili do vytvorenia široko akceptovaných komunikačných jazykov, akými sú napríklad *KQML* alebo *FIPA*[4]. Cieľom tohoto dokumentu nieje popisovanie komunikačných štruktúr, viac o tejto problematike je možné nájsť napríklad v [36].

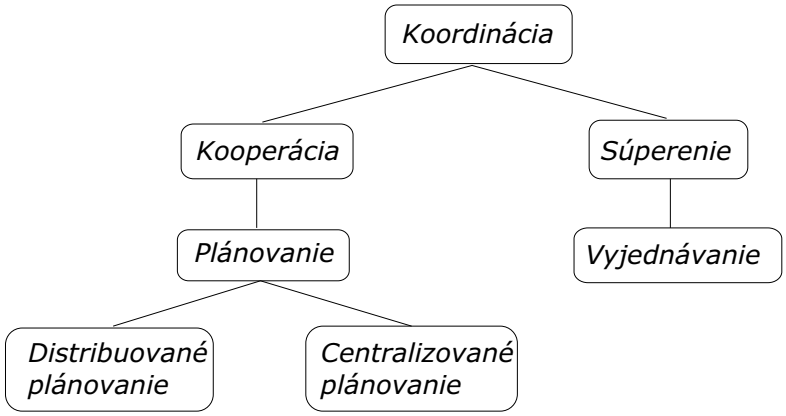
[4] viď Slovník pojmov na konci dokumentu.

3.3.2 Koordinácia

Komunikácia slúži agentom na zlepšenie výkonu v dosahovaní svojich vlastných cieľov alebo cieľov spoločnosti agentov. Vďaka komunikácii môžu agenti koordinovať svoje aktivity a správanie, čo potom vedie k väčšej súvislosti a koherencii systému.

Koordinácia je jednou z vlastností agentových systémov, ktoré vykonávajú nejakú činnosť v zdieľanom prostredí. Stupeň koordinácie sa meria schopnosťou agentov vyhýbať sa problémom redukovaním sporov o prostreidky, vyhýbaním sa 'livelocku' a 'deadlocku' a udržovaním vhodných bezpečnostných podmienok.

Kooperácia je koordinácia medzi agentami, ktorí nemajú protichodné záujmy, zatiaľčo *vyjednávanie* je koordinácia medzi konkurujúcimi agentami, alebo agentami, ktorí sledujú jednoducho len vlastné záujmy. Aby mohli agenti kooperovať, musia mať zvyčajne model ostatných agentov a tiež musia vyvíjať model budúcich interakcií, čo predpokladá sociálnu schopnosť.



Obrázok 3.3: Taxonómia spôsobov, ktorými agenti koordinujú svoju činnosť [35].

3.4 Organizácia

Organizácia poskytuje rámec pre interakcie agentov formou rolí, očakávaných správaní a vzťahov autority. Je to vzor informačných a riadiacich vzťahov medzi agentami a distribúcia spôsobilostí pre riešenie problémov medzi jednotlivých agentov [32].

Organizácia ukladá obmedzenia na spôsob, akým agenti komunikujú a koordinujú svoju činnosť. Katia Sycara [32] uvádza nasledovné príklady organizácií multiagentových systémov:

Hierarchia: Autorita rozhodovania a riadenia je sústredená v jednej entite (alebo v špecializovanej skupine) na každej úrovni hierarchie. Interakcia má formu vertikálnej komunikácie od nadradeného agenta k podriadenému a naopak. Nadradený agenti majú kontrolu nad prostriedkami a nad procesom rozhodovania.

Spoločenstvo expertov: Plochá (flat) organizácia, v ktorej každá entita je špecializovaná pre nejakú oblasť. Agenti komunikujú podľa pravidiel správania a koordinujú svoju činnosť vzájomnými úpravami svojich riešení tak, aby bola dosiahnutá celková koherencia.

Trh: Kontrola je distribuovaná medzi agentov, ktorí súperia o úlohy alebo prostriedky prostredníctvom licitovania a zmluvných mechanizmov. Agenti interagujú cez jednu premennú - cenu, ktorá sa používa na ohodnotenie služieb. Agenti sa koordinujú cez vzájomné úpravy cien.

Vedecké spoločenstvo: Model fungovania pluralistickej spoločnosti. Riešenia problémov sú konštruované lokálne, a potom sú prezentované ostatným riešiacim entitám, ktoré môžu testovať, spochybňovať a vybrusovať riešenie.

V otvorených prostrediach je kľúčová adaptabilita organizácie. Organizácie, ktoré sa dokážu prispôsobiť meniacim sa okolnostiam pozmenením interakcií agentov, majú potenciál vyústiť do koherentného systému v meniacom sa otvorenom prostredí.

3.5 Využitie multiagentových systémov

Keď uvažujeme o ľubovoľnej technológii, prirodzene vyvstáva otázka: *Pre aké aplikácie sa hodí a v čom je lepšia ako predchádzajúce použité prístupy?*. V prípade MAS sa na túto otázku pokúsi dať odpoveď nasledujúci odsek, ktorého primárnym zdrojom je práca Katie Sycary [32].

Multiagentové systémy sú schopné riešiť problémy, ktoré sú príliš rozsiahle na to, aby ich riešil centralizovaný agent. Dôvodom môžu byť obmedzené

výpočtové prostriedky alebo fakt, že jeden centralizovaný systém by sa mohol stať výkonnostným *zúženým hrdlom* alebo by mohol v kritickom momente zlyhať.

Umožňujú prepojenie a spoluprácu viacerých existujúcich *zastaralých systémov*. Zastaralé systémy musia byť pravidelne aktualizované podľa meniacich sa potrieb firmy. Kompletne prepísať takéto systémy sa môže ukázať príliš nákladné, a preto viac-menej jedinou možnosťou je začleniť ich do spoločnosti spolupracujúcich agentov, v ktorej môžu byť využívané inými softvérovými časťami. Aby mohol byť zastaralý softvér začlenený do systému, môžeme pre neho napríklad vytvoriť *agentový obal*, ktorý umožní jeho prepojenie s ostatnými systémami.

Poskytujú riešenie problémov, na ktoré sa môžeme prirodzene dívať ako spoločnosť autonómnych interagujúcich komponentov - agentov. Príkladom takýchto problémov je napríklad letový dispečing alebo vyjednávanie pri kupovaní a predávaní tovaru na internete.

Poskytujú riešenia, ktoré efektívne využívajú informačné zdroje, ktoré sú priestorovo distribuované. Príkladom takýchto aplikácií je napríklad monitorovanie seizmickej činnosti alebo zhromažďovanie informácií z internetu.

Poskytujú riešenia v situáciách, kde je distribuovaná odbornosť. Tieto situácie zahŕňajú napríklad paralelné softvérové inžinierstvo, zdravotnú starostlivosť alebo výrobný proces.

Zlepšujú systémovú výkonnosť po nasledujúcich stránkach:

(1) *výpočtová efektivita:* paralelnosť výpočtu je využitá (pokiaľ je komunikácia minimalizovaná - napríklad posielaním iba výsledkov a informácií vysokej úrovne namiesto posielania informácií nízkej úrovne);

(2) *spoľahlivosť:* elegantné zotavenie sa zo zlyhaní systemových komponentov - dynamicky sa nahradí agentom s nadbytočnými schopnosťami alebo vhodnou koordináciou viacerých agentov;

(3) *rozšíriteľnosť:* množstvo agentov alebo ich schopnosti môžu byť zmenené;

(4) *robustnosť:* schopnosť systému tolerovať neistotu - agenti si vymieňajú vhodné informácie;

(5) *udržovateľnosť:* systém zložený z viacerých komponentov-agentov je jednoduchší na údržbu vďaka vysokej modulárnosti;

(6) *schopnosť reakcie:* modulárnosť dokáže zvládnuť anomálie lokálne, nepotrebuje ich posúvať do systému;

(7) *flexibilita:* agenti s rôznymi schopnosťami sa dokážu adaptívne zorganizovať kvôli riešeniu aktuálneho problému;

(8) *znovupoužiteľnosť:* funkčne špecifickí agenti môžu byť využití v iných zoskupeniach agentov pri riešení iných problémov.

Kapitola 4

Agenti v softvérovom inžinierstve

Softvérové inžinierstvo je disciplína, ktorá sa zaoberá tvorbou počítačových systémov čo najvyššej kvality pre priemyselné aj iné využitie. Pri zvyšujúcej sa náročnosti problémových domén, je potrebné skúmať stále nové možnosti a paradigmy, ktoré by zvládli tieto požiadavky bez výrazných nežiadúcich dôsledkov, čo sa týka zložitosti a robustnosti softvéru. Každá zo širokého spektra úspešných paradigiem (ako sú napríklad procedurálne programovanie, štrukturované programovanie, deklaratívne programovanie, objektovo-orientované programovanie a iné) prichádza s novým prelomovým prístupom, ktorý má zjednodušiť proces vývoja, alebo dokonca umožňuje vytvárať zložitejšie aplikácie. Avšak rýchly vývoj si vyžaduje neustále nové, lepšie vývojové prostiedky.

4.1 Prístup k zložitým systémom

Tvorba zložitých systémov, ktoré si vyžaduje dnešná priemyselná situácia, sa stáva stále naliehavejšou oblasťou softvérového inžinierstva. Je potrebné poskytovať štruktúry a techniky, ktoré pomáhajú zvládať zložitosť problémových domén. Našťastie táto zložitosť má niekoľko dôležitých pravidelností (prevzaté z [15]):

- Komplexita má často formu hierarchie. Systém je zložený z prepojených subsystémov, ktorých vnútorná štruktúra je tiež hierarchická, až pokiaľ sa nedosiahne najnižšia úroveň elementárneho subsystému. Presná povaha týchto hierarchických vzťahov sa mení od systému k systému, aj keď je možné určiť niektoré špecifické formy (napr. client/server, peer, atď.). Tieto vzťahy nie sú statické a často sa v priebehu času menia.

- Výber primárnych komponentov systému je relatívne ľubovoľný a je definovaný cieľmi a plánmi pozorovateľa.
- Hierarchické systémy sa vyvíjajú rýchlejšie ako tie nehierarchické porovnateľnej veľkosti (zložitejšie systémy sa vyvinú z jednoduchších rýchlejšie, ak existujú nejaké jednoznačne identifikovateľné *stabilné prechodné formy*).
- Dá sa rozlišovať medzi interakciami medzi subsystémami a interakciami vrámci subsystémov. Toto dáva perspektívu, že zložité systémy sú takmer dekomponovateľné, t.j. môžeme pristupovať k jednotlivým subsystémom akoby boli nezávislé, aj keď v skutočnosti sú potrebné nejaké interakcie medzi nimi.

Na základe týchto pozorovaní vyvinuli odborníci z oblasti softvérového inžinierstva základné nástroje, pomocou ktorých je možné lepšie zvládať zložitosť [3]:

Dekompozícia: Najzákladnejšou technikou pre zvládanie zložitosti je rozdeliť problém na menšie časti, ktoré potom môžu byť riešené relatívne nezávisle. Dekompozícia uľahčuje prácu analytikovi, ktorý sa tak potrebuje zaoberať menšou problémovou doménou.

Abstrakcia: Proces definovania zjednodušeného modelu systému, ktorý dáva do popredia niektoré detaily alebo vlastnosti systému, zatiaľčo iné potláča. Rovnako ako dekompozícia, abstrakcia zužuje oblasť, ktorou sa musí analytik v danom čase zaoberať.

Organizácia: Proces definovania a organizovania vzájomných interakcií medzi jednotlivými komponentami systému. Schopnosť špecifikovať a zavádzať vzťahy vrámci organizácie systému pomáha analytikovi nasledovne: je možné zoskupovať jednotlivé komponenty do väčších celkov, ktoré potom môžu byť považované za jednotky vyššej úrovne analýzy, a následne špecifikovať komunikáciu na vyššej úrovni medzi jednotlivými celkami.

4.2 Porovnanie agentovej technológie a klasického prístupu tvorby softvéru

Multiagentové systémy boli inšpirované prirodzenou biologickou distribúciou. Umožňujú vytvárať softvér, ktorý nemá také striktné obmedzenia ohľadne centralizovaného, plánovaného a sekvenčného riadenia, aj keď nie všetky

multiagentové systémy tento potenciál využívajú. Použitím agentovej technológie je možné vytvárať systémy, ktoré sú:

- "*decentralizované radšej ako centralizované;*

- *emergentné radšej než plánované;*

- *paralelné radšej než sekvenčné." [28]*

Parunak vo svojej práci [28] porovnáva technológiu multiagentových systémov s klasickým konvenčným prístupom tvorby softvéru. Jeho úvahy sú zhrnuté v tabuľke 4.1.

Problém	**Autonómni agenti**	**Konvenčný prístup**
Model	Ekonomika, biológia	Armáda
Pre konvenčný prístup		
Teoretické optimum	Nie	Áno
Úroveň predpovede	Ako celkok	Jednotlivo
Výpočtová stabilita	Nízka	Vysoká
Pre autonómnych agentov		
Vystihue realitu	Vysoká útoveň	Nízka úroveň
Vyžaduje centralizované dáta	Nie	Áno
Reakcia na zmeny	Robustná	Nestála
Rekonfigurovateľnosť systému	Ľahká	Zložitá
Povaha softvéru	Krátky, jednoduchý	Dlhý, zložitý
Čas potrebný na plánovanie	Real-time	Veľa času

Tabuľka 4.1: Agentová technológia vs. konvenčný prístup [28].

Ako hovorí hneď prvý riadok tabuľky, systémy autonómnych agentov sú inšpirované biológiou (ekosystémy) a ekonomikou (trhy), na rozdiel od tradičného prístupu, ktorý využíva vzor hierarchickej organizácie, akú je možno vidieť napríklad v armáde.

Agentová technológia ako súčasť distribuovanej umelej inteligencie využíva decentralizáciu dát a riadenia, každý agent má samostatný logický pohľad na prostredie a samostatne vykonáva akcie ako odpoveď na zmenu prostredia. Sekvenčný proces, v ktorom sa akcie najprv naplánujú a potom vykonávajú,

je nahradený iným, v ktorom plánovanie emergentne vyplýva z akcií jednotlivých agentov a toto plánovanie je v real-time.

Z agentového prístupu môžu vyplynúť isté nevýhody. Nieje garantovaná teoretická optimálnosť a predpovede pre výpočet systému autonómnych agentov je možné sformulovať len na úrovni systému ako celku. Vyplýva to z návrhu jednotlivých komponentov - agentov, ktorí majú emergentné správanie. Multiagentové systémy sa zároveň môžu prejaviť ako výpočtovo nestále. Avšak zo skúsenosti badať, že teoretické optimá vypočítané pre tradičné výpočtové systémy často nie sú realizovateľné v praxi, a detailné predpovede, ktoré umožňuje konvenčný prístup, sú často vyvrátené aplikáciou v skutočnom svete.

Vo všeobecnosti však multiagentové systémy oveľa lepšie modelujú realitu a flexibilnejšie sa adaptujú na komplikované a otvorené prostredia.

4.3 Charakteristiky vhodných aplikácií

Agenti, rovnako ako ostatné technológie, majú svoje spôsobilosti, a sú vhodné pre systémy a situácie, ktoré tieto spôsobilosti vyžadujú.

Nie vždy je vhodné použiť pre vývoj softvéru práve multiagentový prístup. V tejto časti si priblížime situácie, v ktorých sa tento prístup žiada, alebo je priam nevyhnutný.

Distribúcia: Agentový prístup je najprirodzenejším modelom distribúcie, takže je to sľubný prístup pre aplikácie, pri ktorých je centralizácia nemysliteľná alebo nerealizovateľná. Najevidentnejším príkladom takýchto systémov je internet. Neustále rozširovanie a zavádzanie nových služieb robí z internetu vynikajúceho adepta pre aplikáciu distribuovaného riadenia. Ďalšou oblasťou uplatnenia sú systémy, v ktorých zlyhanie centralizovanej kontroly môže viesť k nežiadúcim, až katastrofálnym následkom.

Prirodzená dekompozícia: Systémy, ktoré sú vo svojej podstate zložené z interagujúcich entít, ktorých charakter prirodzene vyplýva z domény. Klasický prístup tvorby softvéru často aplikuje na problém rozdelenie, ktoré absolútne neodráža prirodzene sa vyskytujúce entity, a to vedie k neefektívnym systémom.

Vysoká konfigurovateľnosť alebo modifikovateľnosť: Agentový systém je schopný sa rekonfigurovať za behu a preto je výhodným riešením pre

systémy, ktoré musia reagovať na širokú škálu rozličných podmienok.

Častá zmena: Vyššie v dokumente už boli spomínané tzv. zastaralé (legacy) systémy, ktoré sa veľmi rýchlo po svojej aplikácii do reálneho prostredia stávajú neaktuálnymi a nestíhajú odpovedať na prúd stále nových funkčných požiadaviek.

Autonómne jednanie: Agentový prístup je efektívnym riešením aj pre systémy, ktoré musia autonómne jednať. Príkladom môže byť napríklad letový dipečing, keď je potrebné riešiť možné kolízie a byť pripravený sa vysporiadať s nečakanými (teda vopred nepredpokladanými) situáciami; neschopnosť adekvátne riešiť také situácie môže vyústiť v nepríjemnosti.

Nie vo všetkých situáciách je vhodné použiť agentový prístup. O takýchto situáciách si môžete prečítať v práci Wooldridgea a Jenningsa [39].

Kapitola 5

Agentové platformy

Ako bolo v priebehu tohoto dokumentu už mnohokrát spomínané, agentová technológia - špeciálne *mobilní agenti* - sa stávajú stále zaujímavejšími, dokonca aj v komerčnej sfére. Ruka v ruke s každým úspešným prístupom v oblasti tvorby počítačových systémov samozrejme prichádza vznik rôznych štandardizovaných prístupov, vývojových nástrojov a platforiem, ktoré uľahčujú tvorbu daných systémov. Mnoho z nich je typu *open source*, čo zaručuje neustály vývoj, vcelku rýchle vychytávanie chýb a nedostatkov a hlavne spojenie viacerých dobrých nápadov v štýle "viac hláv, viac rozumu".

Agentové platformy definujú:

- Všeobecný návrh vývojových vzorov a vývojový rámec (framework).
- Middleware slúžiaci na medzi-agentovú komunikáciu.
- Organizáciu riadenia výpočtu, posielania správ, adresárov a životného cyklu komponentov v systéme.

5.1 Kritériá hodnotenia

Pri takej významnej paradigme, akou agentový prístup bezpochyby je, môžete nájsť a využiť pre tvorbu systémov skutočne obdivuhodné množstvo vývojových nástrojov. Je preto dôležité skúmať do hĺbky ich silné stránky a prípady, v ktorých bude ich využitie efektívnym riešením.

Z dlhodobého výskumného hľadiska (na rozdiel od konkrétnych situácií, v ktorých sa rozhodujete pre nejaký konkrétny produkt) treba brať pri analýze platforiem do úvahy rýchly vývoj oblasti agentových systémov a závažné zmeny, ktoré tento vývoj nevyhnutne čas od času prináša. Preto je

priveľmi špecifická analýza, alebo porovnanie platforiem, veľmi nespoľahlivé, keďže nikdy neviete vpored presne odhadnúť, ako dlho bude aktuálne. To však rozhodne neznamená, že takéto konkrétne porovnania nemajú význam. Pri porovnaní viacerých aktuálnych riešení veľmi jasne vyniknú prednosti a slabosti jednotlivých nástrojov a vývoj môže ísť dopredu.

Na druhej strane, všeobecnejšia analýza nám poskytuje niečo ako smernice, ktoré ukazujú, akým smerom by sa mal vývoj uberať. Tento všeobecnejší prístup väčšinou určuje nejakú množinu kritérií, vzťahujúcu sa na niektorú aplikačnú problémovú doménu, na základe ktorých by jednotlivé platformy mali byť posudzované a hodnotené. Už teraz existuje mnoho kritérií popísaných v literatúre a v nasledujúcich riadkoch si niektoré z nich predstavíme.

5.1.1 Kritériá podľa Nguyena a spol.

V práci odborníkov zo SAV[1] [26], je uvedené porovnanie a zhodnotenie viacerých voľne dostupných agentových platforiem. Na hodnotenie určili nasledovné kritériá:

- *Kompatibilita použitých štandardov:* všeobecné štandardy pre agentovú technológiu sú FIPA, (OMG) MASIF, atď.
- *Komunikácia:* podpora posielania správ medzi rôznymi platformami.
- *Mobilita agentov:* silná - schopnosť systému prenášať zdrojový kód a aktuálny výpočtový stav vykonávanej jednotky; slabá - systém prenáša len kód, vykonávané programové vlákna musia byť znova spustené (spustenie vykonávajú "daemoni", ktorí tieto kódy očakávajú).
- *Bezpečnostná politika:* bezpečná vnútro-plaformová a homogénna mimoplatformová komunikácia.
- *Dostupnosť*
- *Použiteľnosť a dokumentácia:* Úroveň prijateľnosti pre používateľov a vývojárov.
- *Vývojové otázky:* praktické aplikácie, vývojové projekty.

Pre zaujímavosť, hodnotené boli patformy, odporúčané organizáciou FIPA a podľa hodnotenia v tejto práci obstáli najlepšie platformy Grasshopper, JADE a Aglety.

[1] Slovenská akadémia vied

5.1.2 Kritériá podľa Ricordela a Demazeaua

Ricordel a Demazeau [29] rozširujú proces vývoja multiagentového systému na 4 etapy:

- *Analýza:* proces zisťovania, konkretizovania a popisovania typu problému a okolitej domény.
- *Návrh:* proces definovania architektúry riešenia daného problému, špecifikácia princípov (napr. pomocou UML).
- *Vývoj:* proces konštrukcie riešenia, programovanie zdrojového kódu.
- *Uvedenie do reality (Deployment):* proces realizácie riešenia na reálnom probléme, spustenie softvéru na počítačovej sieti a poprípade upravovanie a rozširovanie funkcionality.

Vo svojej práci určili zároveň 4 kritériá, ktoré sú podľa nich relevantné v každej z uvedených etáp, a ktoré sa dotýkajú čo najväčšieho množstva problémových aspektov vývojového procesu. Týmito kritériami sú:

- *Kompletnosť:* Úroveň, na ktorej špecifická platforma pokrýva danú etapu. Týka sa kvality aj kvantity dokumentácie a nástrojov, ktoré pre danú etapu poskytuje.
- *Aplikovateľnosť:* Škála možností, ktoré táto platfoma ponúka a obmedzenia, ktoré platforma v danej etape ukladá.
- *Zložitosť:* Aké zložité je splniť etapu. Zahŕňa to požiadavky na kompetencie človeka, ktorý sa podieľa na vývoji danej etapy, a zároveň množstvo práce, ktoré si vyžaduje splnenie úloh.
- *Znovupoužiteľnosť:* Množstvo práce získané opätovným využitím predchádzajúcich výsledkov.

Pri hodnotení platforiem je možné aplikovať ešte aj nasledujúce 2 kritériá. Tieto kritéria sú vhodné pri výbere platformy pre konkrétny projekt:

- *Dostupnosť:* Existuje "trial" verzia? Je zdrojový kód dostupný?
- *Podpora:* Ako sa bude platforma ďalej vyvíjať? Je táto platforma využívaná vo veľkom rozsahu?

5.1.3 Kritériá podľa Casagniho a Lyellovej

Casagni a Lyell [11] vyvinuli zložitú a obsiahlu štruktúru porovnávacích kritérií, ktorú aplikovali na porovnanie dvoch prístupov: multiagentovej platformy podľa FIPA štandardov a web-centrickej J2EE platformy. Kritériá sú rozdelené do troch množín:

(1) Črty návrhových vlastností.

(2) Črty vplyvu vývojového rámca na riešenie.

(3) Atribúty kvality.

Medzi *črty návrhových vlastností* zaradili nasledovné kritériá:

- **Aplikačná úroveň:**
  - *Rozsah návrhu:* počet komponentov aplikácie.
  - *Funkčná súdržnosť:* nakoľko poskytujú jednotlivé komponenty uživateľovi jednotnú aplikáciu.
  - *Prepojenie:* interakcia medzi komponentami tvoriacimi aplikáciu.
- **Komponentová úroveň:**
  - *Dedenie/Vzory:* pravidlá pre vytváranie inštancií komponentov.
  - *Zapúzdrenie:* ako veľmi je komponent zameraný na jednu "tému".
  - *Zložitosť:* hĺbka a šírka funkfionality komponentu.

Medzi *črty vplyvu vývojového rámca* patria:

- **Črty podporujúce usídlenie komponentov:**
  - *Podpora konfigurovateľnosti platformy; popisné súbory; rozdelenie do balíkov; zavedenie do reálneho problému; služby, ktoré platforma poskytuje komponentom.*
- **Dôsledky pre návrh komponentov vyplývajúce z faktorov vývojového rámca:**
  - *Prístup ku komponentom; obmedzenia prostriedkov, ktoré na komponenty ukladá platforma; využívanie návrhových vzorov.*

Do tretej množiny *atribútov kvality* zaradili nasledovné:

- **Dostupnosť:** do akej miery je systém schopný plniť svoju funkcionalitu.
- **Použiteľnosť:** platforma je hodnotená z troch pohľadov.
  - *Pohľad vývojového pracovníka:* črty, ktoré ovplyvňujú vývoj komponentov.
  - *Pohľad pracovníka, ktorý zavádza systém do praxe:* črty, ktoré ovplyvňujú zavádzanie komponentov do reálneho riešenia.
  - *Pohľad uživateľa:* črty, ktoré ovplyvňujú používanie aplikácie.
- **Upraviteľnosť:**
  - *Udržovateľnosť:*
    - *Rozširovateľnosť:* črty, ktoré ovplyvňujú upraviteľnosť systému, keď sa jedná o zakomponovávanie nových požiadaviek.
    - *Znovupoužiteľnosť:* črty, ktoré ovplyvňujú využiteľnosť komponentu v novom probléme bez prílišnej námahy.
  - *Flexibilita:* črty, ktoré ovplyvňujú schopnosť systému odpovedať na podnety prostredia bez zásahu do definovanej funkcionality aplikácie.
    - *Faktory prostriedkov:* prostriedky poskytované infraštruktúrou, ktoré by mala aplikácia využívať.
    - *Výkonnosť:* časové merania.

Všetky tieto spomínané kritériá pomáhajú lepšie pochopiť požiadavky na multiagentové systémy a na nástroje tvorby týchto systémov. Zároveň určujú črty, ktoré by mali dané systémy spĺňať a odlišujú tak agentovú technológiu od ostatných prístupov, ktorých požiadavky navrhuje a skúma tradičné softvérové inžinierstvo.

Kapitola 6

Systémy riadenia technologických procesov

Predchádzajúce kapitoly sa zaoberali využitím agentovej technológie v softvérovom inžinierstve ako takom, a priblížením niektorých požiadaviek na agentové platformy pre tvorbu zložitých softvérových systémov. V tejto kapitole si povieme o systémoch riadenia procesov. Pre ne sú práve agentové systémy veľmi prirodzeným riešením, pretože riadenie procesov je samo o sebe *autonómny a reaktívny systém.*

Skôr než pôjdeme ďalej analyzovať doménu systémov riadenia, zadefinujeme si pojem *real-time systém.*

> **Real-time systém** *je systém, ktorý musí spĺňať explicitné (ohraničené) časové obmedzenia na reakciu, inak riskuje kritické nepriaznivé následky, vrátane zlyhania (kde zlyhanie znamená nesplnenie jednej alebo viacerých požiadaviek, sformulovaných vo formálnej špecifikácii systému) [17].*

6.1 Analýza domény riadenia procesov

Keď sa sféra vplyvu technológií presunula z výpočtových stredísk a počítačových laboratórií do skutočného prostredia, vyvstala potreba pre real-time riadenie systémov. Vpodstate každý priemyselný produkt je aspoň v slabom zmysle real-time. Aby bol softvér na trhu úspešný, musí poskytovať čo najrýchlejšie služby; obmedzenie v každom prípade ukladá minimálne trpezlivosť zákazníka. U systémov riadenia technologických procesov však môže mať nesplnenie časového obmedzenia oveľahoršie následky ako len stratu klienta.

Na životne kritické systémy sa vzťahuje nasledujúca definícia:

> **Hard real-time systém** *je systém, v ktorom nesplnenie jediného časového obmedzenia môže viesť k úlpnému a katastrofálnemu systémovému zlyhaniu [17].*

Soft real-time systémy potom môžeme definovať ako systémy, u ktorých nesplnenie nejakého časového obmedzenia nevedie k zlyhaniu celého systému, ale len k degradácii výkonu.

Keď ide o systémy riadenia technologických procesov, môžeme ich z definície zaradiť do množiny hard real-time systémov, vzhľadom na to, že nesplnenie nejakej časovej podmienky rozhodne môže viesť k chybe v procese, ktorá pri reálnych aplikáciách ako sú tie spomínané v úvode znamená katastrofálne následky.

Príkladom takýchto systémov je riadenie letovej prevádzky, kde musia byť dáta spracovávané v pravidelných dosť krátkych časových intervaloch, aby bolo zabezpečené riešenie možných kolízií lietadla (pod dátami sa rozumejú informácie o letovej výške lietadla, rýchlosti, smeru a informácií z radaru). Ďalším príkladom je napríklad riadenie prevádzky jadrovej elektrárne, kde napríklad neskoro spracovaná informácia z tepelného ventilu môže znamenať veľké nepríjemnosti, ktoré môžu vyústiť až vo výbuch.

6.2 Požiadavky na systém riadenia

Vezmime si systém riadenia jadrovej elektrárne. Normálny scenár tohoto systému bude bežná správa nameraných hodnôt a vykonanie príslušnej akcie. Napríklad keď systém dostane zvýšenú hodnotu nameranú z teplomera, otvorí nejaký príslušný ventil, alebo spustí chladiace zariadenie. Tieto akcie sú reaktívneho charakteru, systém ich očakáva a má vopred pripravené zodpovedajúce akcie.

Pri pohotovostných situáciách je to už o dosť zložitejšie, pretože je potrebné brať do úvahy veľké množstvo obmedzení a systém má málo informácii na to, aby vedel jednoducho vyvodiť odpoveď na dané okolnosti. Pohotovostné situácie môžu vzniknúť z rôznych dôvodov:

- zlyhanie hardvéru alebo iných technických zariadení,

- klamné dáta, ktoré je potrebné detekovať - to sa dá napríklad pomocou systému viacerých senzorov; pravidelným porovnávaním hodnôt je potom možné nielen zistiť chybu dát, ale aj lokalizovať chybné zariadenie,

- zlyhanie softvérového komponentu - v prípade multiagentového riešenia tento problém nie je nijak zvlášť dramatický, vzhľadom na distribuované riadenie a redundanciu funkcionality jednotlivých agentov.

Tieto situácie je potrebné čo najrýchlejšie riešiť, aby sa z pomerne malého problému nestala značnejšia pohroma. Z predchádzajúceho popisu rizikových udalostí už teda vieme vyvodiť požiadavky na činnosť a rozhodovanie riadiaceho systému. Pre korektné riešenie chýb musí zvládať nasledovné akcie:

(1) Detekcia poruchy: systém musí zároveň vedieť rozlíšiť medzi skutočnými poruchami a náhodnými nezrovnalosťami v dátach alebo v prostredí, spôsobených nejakým jednorázovým neškodným faktorom.

(2) Lokalizácia poruchy a následne notifikácia príslušných subsystémov, poprípade ľudského obsluhovateľa, zodpovedného za vyriešenie problému.

(3) Zmapovanie situácie v systéme, akonáhle sa znova stabilizuje, a príprava a realizácia plánu na odstránenie následkov a uvedenie systému do pôvodného bezporuchového stavu.

Tento management neočakávaných situácií je vitálnym pre riadenie a automatizáciu procesov, keďže manuálne odstraňovanie týchto problémov sa ukazuje veľmi zložitým. Najviac havárií spôsobených poruchou technologického procesu má ešte stále na svedomí ľudský faktor.

Vo viacerých výskumných prácach boli teda navrhované rôzne techniky a systémy na vysporiadavanie sa s týmito pohotovostnými situáciami. Niektorými z riešení sú nasledovné [34]:

- integrácia rôznych paradigiem riešenia problémov, schém na reprezentáciu vedomostí a prehľadávacích techník,

- uchovávanie globálnych databáz, obsahujúcich dáta procesov a ich vedomosti,

- uvažovanie o činnosti procesov bez vyžadovania presných modelov,

- zvládanie návalu dát a potreba efektívnej kompresie a interpretácie,

- porozumenie a následná reprezentácia správania procesov na rôznych úrovniach abstrakcie.

Na základe týchto požiadaviek na systémy riadenia technologických procesov sa v nasledujúcej časti pokúsime sformulovať niekoľko základných kritérií výberu vhodnej agentovej platformy pre tvorbu takýchto systémov.

6.3 Určenie kritérií hodnotenia platforiem

Ak sa chystáte vytvárať systém riadenia technologických procesov, potrebujete spoľahlivý rámec, v ktorom by naimplementovaní agenti korektne fungovali a kooperovali, a to spĺňajúc časové obmedzenia, ktoré by ste mu uložili. Agenti musia tvoriť stabilný a efektívny systém, ktorý bude schopný spoľahlivo riadiť želané procesy a bude si vedieť poradiť s nečakanými pohotovostnými situáciami, ktoré počas behu systému vyvstanú.

Ak systém nebude schopný riešiť nečakané situácie v čase spĺňajúcom real-time obmedzenia, môžete na to škaredo doplatiť. Preto je potrebné zvážiť výber vhodnej platformy, spĺňajúcej žiadané kritériá v čo najväčšej miere.

Je dobré vybrať si niektorú z platforiem spĺňajúcich štandardy určené nadáciou FIPA, alebo OMG (MASIF štandardy). Prehľady týchto platforiem je možné nájsť na internete, mnohé z nich sú open source-ové projekty.

Prvou skupinou kritérií budú najstrohejšie obmedzenia, ktoré sú životne kritické pre korektný beh systému:

- *Globálna databáza a prístup k nej:* platforma musí podporovať vytváranie a správu globálnych databáz, a zároveň musí poskytovať
- *Učenie:* úroveň získavania a využívania vedomostí o realite a o procesoch, ktoré systém riadi, a ich aplikácia pri rozširovaní reaktívnych odpovedí na podnety prostredia.
- *Komunikácia:* efektivita posielania správ, kvalita použitého middlewareu.
- *Mobilita komponentov:* efektivita a podpora presúvania komponentov.
- *Zotavenie z chyby:* do akej miery ovplyvní zlyhanie systémového komponentu výkon a korektnosť celého systému.
- *Výkonnosť:* časové merania.

Ďalšou skupinou sú kritériá *vývoja a údržby* systému:

- *Použité štandardy a paradigmy:* nakoľko je platforma hardvérovo a softvérovo nezávislá?
- *Rozšírovateľnosť:* ako platforma podporuje pridávanie nových funkčných komponentov bez vplyvu na robustnosť systému?
- *Znovupoužitenosť:* ako všeobecne sú navrhnuté komponentové vzory?
- *Reaktívnosť:* do akej miery vie systém reagovať na podnety z prostredia čisto využitím nadefinovanej funkcionality?
- *Konfigurovateľnosť:* nakoľko je platforma variabilná, do akej miery poskytuje uživateľovi možnosti?
- *Zložitosť:* nakoľko zložité je z vývojového hľadiska vytvoriť fungujúci multiagentový systém?

Nakoniec nesmieme zabudnúť na kritériá *úrovne "priateľskosti"* platformy. Napriek tomu, že možno nemajú taký priamy dopad na kvalitu eventuélne implementovaného systému riadenia, je dobré vyhýbať sa zbytočným komplikáciám v priebehu vývoja. Kritériami kvality vývoja sú:

- *Dostupnosť:* je daný systém voľne dostupný, je štýlu open source? aké zložité je dostať sa k jeho inštalačným súborom?
- *Použiteľnosť:* čo všetko je potrebné, aby mohol byť systém úspešne nainštalovaný, spustený a používaný?
- *Dokumentácia:* kvalita dokumentácie pre vývojového pracovníka a pre používateľa.
- *Spoľahlivosť:* nakoľko je skutočná funkcionalita daného systému spoľahlivá?
- *Podpora:* vyhliadky vývoja platformy v budúcnosti, sféry jej využívania a možnosti konzultovania a debugovania.
- *Aplikovateľnosť na reálne prostredie:* ako odrážajú vlastnosti platformy reálne prostredie, do ktorého bude nasadená.

Ak bude platforma spĺňať uvedené kritériá, nie je ešte zaručený bezproblémový beh systému riadenia, väčšina totiž závisí od samotnej implementácie systému. Avšak výber platformy s čo najlepšími kritériami značne uľahčí celý proces vývoja softvéru, následného nasadenia do reality, udržiavania a rozširovania.

Kapitola 7

Záver

Agentová technológia pritiahla nemálo výskumníkov v oblasti softvérového inžinierstva, keďže poskytuje novú nádejnú paradigmu vývoja distribuovaných softvérových systémov. V prvom rade poskytuje agentová technológia väčsiu spoľahlivosť, bezpečnosť, flexibilitu a real-time-ovosť v systémoch riadenia procesov.

Vyvinuli sme štruktúru kritérií, podľa ktorých je vhodné vyberať agentovú platformu pre tvorbu multiagentového systému riadenia technologických procesov. Táto štruktúra súvisí s požiadavkami na systémy riadenia. Najinovatívnejším kritériom je prístup ku globálnej databáze, mierne porušenie decentralizácie dát.

7.1 Možné rozvinutia práce

Stálo by za zmienku porovnanie implementácie systému riadenia procesov s čiste decentralizovaným ukladaním dát jednotlivými agentami a implementácie s globálnou databázou. Bolo by zaujímavé zistiť všetky plusy s mínusy oboch prístupov, pretože zatiaľ zastávam názor, že by bolo veľmi ťažké, možno aj nemožné určiť jeden prístup ako lepší. Očakávam, že by v rôznych situáciách reagovali rôzne a každá implementácia by mala svoju oblasť dobre riešených problémových udalostí.

Druhé zaujímavé rozvinutie tejto práce by bolo vo vybratí konkrétnej platformy na základe uvedených kritérií a implementácia nejakého systému riadenia procesov, napríklad systému riadenia jednoduchého modelu jadrovej elektrárne, a následné zhodnotenie výhod multiagentového prístupu.

Obe tieto možnosti ostávajú otvorené pre ďalší výskum.

Literatúra

[1] Andras, P., Lazarus , J., Roberts, G., Lynden, S. J. (2005) *"Uncertainty and Cooperation: Analytical Results and a Simulated Agent Society"* Journal of Artificial Societies and Social Simulation, 2005, vol. 9

[2] Bond, A. H., Gasser, L. (1988) *"An analysis of problems and research in DAI"* In Alan H. Bond and Les Gasser, editors, Readings in Distributed Artificial Intelligence, pp 3–35, Morgan Kaufmann Publishers, San Mateo, CA

[3] Booch, G. (1994) *"Object-Oriented Analysis and Design with Applications"* Addison Wesley

[4] Bradshaw, J. M., editor (1997) *"Software Agents"* MIT Press, Cambridge, MA, USA

[5] Brooks, R. A. (1986) *"A Robust Layered Control System for a Mobile Robot"* IEEE Journal of Robotics and Automation, Vol. RA-2

[6] Brooks, R. A. (1990) *"Elephants Don't Play Chess"* In Designing Autonomous Agents (Maes, P., editor), pp 3–15. The MIT Press: Cambridge, MA

[7] Brooks, R. A. (1991) *"Intelligence Without Reason"* In Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91), pp 569–595, Sydney, Australia

[8] Brooks, R. A. (1991) *"Intelligence Without Representation"* Artificial Intelligence, Vol. 47, pp 139–159

[9] Brustoloni, Jose C. (1991) *"Autonomous Agents: Characterization and Requirements"* Carnegie Mellon Technical Report CMU-CS-91-204, Pittsburgh: Carnegie Mellon University

[10] Cao, Y. U., Fukunaga, A. S., Kahng, A. B. (1997) *"Cooperative mobile robotics: Antecedents and directions"* Autonomous Robots, Vol. 4, No. 1, pp 7-23, March 1997

[11] Casagni, M., Lyell, M. (2003) *"Comparison of Two Component Frameworks: The FIPA-Compliant Multi-Agent System and the Web-Centric J2EE Platform"* Proceedings of the 25th International Conference on Software Engineering, Portland, Oregon

[12] Decker, K. S. (1987) *"Distributed Problem Solving: A Survey"* IEEE Transactions on Systems, Man, and Cybernetics, Vol. 17, No.5, Sept. 1987, pp 729-740

[13] Franklin, S., Graesser, A. (1996) *"Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents"* Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag

[14] Hayes-Roth, B. (1995) *"An Architecture for Adaptive Intelligent Systems"* Artificial Intelligence: Special Issue on Agents and Interactivity

[15] Jennings, N. (1991) *"An Agent-Based Approach for Building Complex Software Systems"* SIGART Bulletin, Vol. 2, No. 4, pp 85-88

[16] Kaelbling, L. P. (1991) *"A Situated Automata Approach to the Design of Embedded Agents"* SIGART Bulletin, Vol. 2, No. 4, pp 85-88

[17] Laplante, P. A. (1993) *"Real-Time Systems Design and Analysis: An Engineer's Handbook"* IEEE Press, New York

[18] Lesser, V. R. (1995) *"Multiagent Systems: An Emerging Subdiscipline of AI"* ACM Computing Surveys, Vol. 27, pp 340-342, September 1995

[19] Maes, P. (1991) *"The Agent Network Architecture (ANA)"* SIGART Bulletin, Vol. 2, No. 4, pp 115-120

[20] Maes, P. (1995) *"Artificial Life Meets Entertainment: Life like Autonomous Agents"* Communications of the ACM

[21] Mataric, M. J. (1990) *"A Distributed Model for Mobile Robot Environment Learning and Navigation"* Technical Report AI-TR-1228, MIT Artificial Intelligence Laboratory

[22] Mataric, M. J. (1997) *"Behavior-Based Control: Examples from Navigation, Learning and Group Behavior"* Journal of Experimental and Theoretical Artificial Intelligence 9

[23] Mataric, M. J. (1999) *"Behavior-Based Robotics"* in the MIT Encyclopedia of Cognitive Sciences, Robert A. Wilson and Frank C. Keil, eds., MIT Press, April 1999, pp 74-77

[24] Mataric, M. J. (2001) *"Learning in Behavior-Based Multi-Robot Systems: Policies, Models and Other Agents"* Cognitive Systems Research, special issue on Multi-Disciplinary Studies of Multi-Agent Learning, ed., Vol. 2, No. 1, pp 81-93

[25] Moreno, A. (2002) *"Medical Applications of Multi-Agent Systems"* Computer Science and Mathematics Department, Universitat Rovira i Virgili ETSE, Spain

[26] Nguyen, G., Dang, T. T., Hluchý, L., Laclavík, M., Balogh, Z., Budinská, I. (2002) *"Agent Platform Evaluation and Comparison"* Pellucid 5FP IST-200134519, Institute of Informatics, Slovak Academy of Sciences

[27] Nwana, H. S. (1996) *"Software Agents: An Overview"* Knowledge Engineering Review, Vol. 11, No. 3, pp 1-40, Cambridge University Press

[28] Van Dyke Parunak, H. (1996) *"Applications of Distributed Artificial Intelligence in Industry"* In Foundations of Distributed Artificial Intelligence, (G.M.P. O'Hare, N. R. Jennings, eds.) Wiley

[29] Ricordel, P. M., Demazeau, Y. (2000) *"From Analysis to Deployment: A Multi-Agent Platform Survey"* In Proceedings of the Workshop on Engineering Societies in the Agents' World. Springer-Verlag

[30] Russel, S. J., Norvig, P. (1995) *"Artificial Intelligence: A Modern Approach"* Englewood Cliffs, NJ: Prentice Hall

[31] Stone, P., Veloso, M. (2000) *"Multiagent systems: A Survey from a Machine Learning Perspective"* in Autonomous Robotics, Vol. 8, No. 3, July, 2000

[32] Sycara, K. (1998) *"Multiagent systems"* Artificial Inteligence Magazine, Vol. 19, No. 2, pp 79–92

[33] Torngren, M. (1995) *"Modeling and Design of Distributed Real-Time Control Applications"* Ph.D. thesis, Department of Machine Design, The Royal Institute of Technology, Stockholm, Sweden

[34] Venkatasubramanian, V., Rengaswamy, R., Kavuri, S. N., and Yin, K. (2005) *"A Review of Process Fault Detection and Diagnosis, part 1, 2, 3"* Computer and Chemical Engineering, Vol. 27, No. 3, pp 293-346

[35] Weiss, G. , editor (1999) *"Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence"* MIT Press, Cambridge, MA, USA

[36] Werner, E., (1989) *"Co-operating Agents: A unified theory of communication and social structure"* In M. Huhns and L. Gasser, editors, Distributed Artificial Intelligence, vol. 1I, pp 3-36, Kaufman and Pitman, London

[37] Wooldridge, M., Jennings, N. R. (1995) *"Agent Theories, Architectures, and Languages: a Survey"* in Wooldridge and Jennings Eds., Intelligent Agents, Berlin: Springer-Verlag

[38] Wooldridge, M., Jennings, N. R. (1995) *"Intelligent Agents: Theory and Practice"* The Knowledge Engineering Review

[39] Wooldridge, M., Jennings, N. R. (1999) *"Software Engineering with Agents - Pitfalls and Pratfalls"* In IEEE Internet Computing, Vol. 3, No. 3, pp 20-27

[40] Wooldridge, M., Jennings, N. R. (2000) *"Agent-Oriented Software Engineering"* In Handbook of Agent Technology (ed. J. Bradshaw) AAAI/MIT Press

[41] Zbořil, F. (2004) *"Plánování a komunikace v multiagentních systémech"* Dizertačná práca k získaniu akademického titulu Ph.D., Vysoké učenie technické v Brně, Fakulta informačných technológií

Dodatok A

Slovník pojmov

A

Agenti pre špecifické úlohy (Task-Specific Agents) Agenti so špecifickými spôsobilosťami, určený na riešenie špecifických úloh.

Agenti rozhrania (Interface Agents) Kolaboratívni agenti implementovaní pomocou reaktívnej technológie.

Agenti zábavného priemyslu (entertainment agents) Agenti, určení do počítačových hier ako virtuálni súperi, a.i.

B

Behaviorálne siete (Behavioral Networks) Štrukturované siete skladajúce sa zo vzájomne interagujúcich komponentov, ktorými sú modely jednotlivých správaní.

Bezpečnostná politika (Security Policy) Plán na riešenie bezpečnostných záležitostí, alebo množina obmedzení, potrebných pre zachovanie istej úrovne bezpečnosti.

Biologický agent (Biological Agent) Biologický (živý) objekt vnímajúci svoje prostredie a autonómne v ňom konajúci za splnenia svojich cieľov (táto definícia zahŕňa napríklad zvieratá, aj ľudí).

C

CORBA - Common Object Request Broker Architecture Štandard dojednávacích technológií, uvedený a spravovaný konzorciom OMG (Object Management Group). Definuje rozhranie, komunikačný protokol

a informačné modely, ktoré umožňujú vzájomnú interakciu heterogénnym aplikáciám napísaným v rôznych programovacích jazykoch a bežiacim na rôznych platformách.

D

Distribuovaná umelá inteligencia (Distributed Artificial Intelligence) Disciplína, ktorá sa vyvinula z oblasti umelej inteligencie. Predmetom záujmu tejto oblasti je riešenie distribuovaných problémov, ktoré vyžadujú umelú inteligenciu.

Distribuované riešenie problémov (Distributed Problem Solving) Odvetvie DUI, ktoré sa zaoberá managementom informácií.

Distribuované výpočty (Distributed Computing) Oblasť, ktorá sa venuje paralelným výpočtom využívajúcim viacero počítačov, ktoré spolu komunikujú cez sieť, aby mohli vyriešiť daný problém.

Dojednávacie technológie (Broker Technologies) Softvér, ktorý sprostredkuváva interakciu medzi dvoma objektami: zvyčajne medzi *klientom* a *serverom*, alebo medzi databázou a objektom, ktorý od nej žiada dáta.

E

Efektor (Effector) Mechanizmus, ktorým agent vykonáva v prostredí akcie.

F

FIPA - Foundation for Intelligent Physical Agents Zoskupenie určené na vývoj a stanovovanie štandardov pre heterogénnych interagujúcich agentov a agentové systémy.

I

Inteligentný agent (Intelligent Agent) Softvérový agent, ktorý je schopný autonómneho a inteligentného správania.

K

Kolaboratívny agent (Collaborative Agent) Pojem uviedol Nwana. Reprezentuje autonómneho agenta, ktorý kladie dôraz na kooperáciu.

Kompilačný jazyk (Assembly Language) Pre človeka zrozumiteľný zápis strojového jazyka.

KQML - Knowledge Query and Manipulation Language Komunikačný jazyk a protokol založený na SGML, ktorý je určený na výmenu informácií a znalostí.

O

Otvorený systém (Open System) Systém, ktorého štruktúra sa môže dynamicky meniť.

Open Source Spôsob vývoja, ktorý umožňuje pristupovať k zdrojovým kódom vyvíjaného produktu.

P

Paralelné softvérové inžinierstvo (Concurrent Engineering) Proces vývoja softvéru, v ktorom jednotlivé etapy nie sú vyvíjané postupne, ale zároveň.

Pridružené zariadenie (Slave Device) Zariadenie, ktoré nekoná samostatné rozhodnutia, ale čaká na príkaz riadiaceho zariadenia.

Prostredie (Environment) Priestor, v ktorom agent vykonáva svoju činnosť a s ktorým interaguje.

R

Reaktívny agent (Reactive Agent) Agent bez symbolickej reprezentácie sveta, ktorý rozhodovanie uskutočňuje na základe vopred daných dvojíc *podnet/reakcia*.

Riadiaca architektúra (Control Architecture) Architektúra návrhu vnútornej stavby agenta.

Robotický agent (Robotic Agent) Robotický systém, ktorý spĺňa definíciu agenta, teda je identifikovateľný ako autonómna entita s vlastnými cieľmi.

S

Senzor (Sensor) Zariadenie, ktorým agent vníma signál alebo podmienky z prostredia.

Šikovný agent (Smart Agent) Tento pojem uviedol Nwana. Reprezentuje autonómneho genta, ktorý je schopný kooperovať a učiť sa.

Softvérový agent (Software Agent) Abstrakcia, logický model popisu softvéru, ktorý koná za uživateľa alebo iný program ako zástupca (agent).

Subsumpčná architektúra (Subsumption Architecture) UI koncept, pochádzajúci z "behavior-based" robotiky. Spôsob dekompozície jedného zložitého správania na viacero jednoduchších vrstiev so zvyšujúcou sa abstrakciou.

Z

Symbolická umelá inteligencia (Symbolic Artificial Intelligence) Disciplína umelej inteligencie, založená na idey modelovania reprezentácie sveta pomocou symbolov.

Symbolické uvažovanie (Symbolic Reasoning) Uvažovanie, využívajúce symbolický model sveta.

U

Umelá inteligencia - UI (Artificial Intelligence - AI) Inteligencia vykazovaná umelou entitou. Za takúto entitu je vo všeobecnosti považovaný počítačový systém.

UML - Unified Modeling Language Jazyk pre modelovanie a špecifikáciu objektov využívaný v softvérovom inžinierstve. Obsahuje štandardizované grafické zápisy objektov, pomocou ktorých je možné zostaviť abstraktný model systému.

V

Výpočtový agent (Computational Agent) Agent určený pre počítačové výpočty.

Zastaralý systém (Legacy System) Počítačový systém alebo aplikácia, ktorá sa ďalej používa, lebo uživateľ (zvyčajne organizácia) nechce tento systém prerobiť, alebo vytvoriť namiesto neho nový.

Zúžené hrdlo (Bottleneck) Časť systému, ktorá počas výpočtu dosahuje limit svojich schopností reagovať (kvôli systémovému zaťaženiu) a kvôli tomu spôsobuje spomalenie behu celého systému.

Zvažujúci agent (Deliberative Agent) Agent, ktorý používa vnútorný symbolický model sveta na uvažovanie o primeranej odpovedi na zmenu prostredia.