Portál pre odborné publikovanie ISSN 1338-0087

Agentovo orientované riadenie systémov diskrétnych udalostí

Koššuth František · Elektrotechnika, Študentské práce

10.05.2010

Článok sa zaoberá možnosťami využitia agentového prístupu k riadeniu systémov diskrétnych udalostí. Využíva agentový prístup k tvorbe riadiaceho systému, zabezpečujúceho novú kvalitu riadenia aj v prípade vzniku poruchy riadením robotov s obmedzenou pracovnou schopnosťou. Experimentálne overenie návrhu je vykonané na príklade kolektívne sa chovajúcich robotov až po plnenie misie s využitím modelu riadeného systému vytvoreného vo virtuálnej realite. Prezentované sú aktuálne poznatky o členení a kategorizácii prístupov a implementácii agentového prístupu v oblasti riadenia.

Úvod

Príčinu vzniku oblasti teórie riadenia, ktorá sa zaoberá systémami diskrétnych udalostí, možno hľadať v snahe riadiť systémy s vyšším stupňom organizácie. Klasická teória riadenia sa zaoberá riadením procesov, ktoré sú opísateľné diferenciálnymi, resp. diferenčnými rovnicami – matematický model systému. Následne postačuje nájsť jeho analytické riešenie pre jeho riadenie.

Článok je zameraný na prezentáciu možností agentovo orientovaného prístupu v riadení takých systémov diskrétnych udalostí, ktoré je problematické opísať deterministicky matematickým modelom (či už z dôvodu zložitosti alebo dynamickej štruktúry systému). Tento prístup možno aplikovať na široké spektrum oblastí výskumu a aplikácie riadiacich systémov DES – Discrete Event System, systém diskrétnych udalostí.

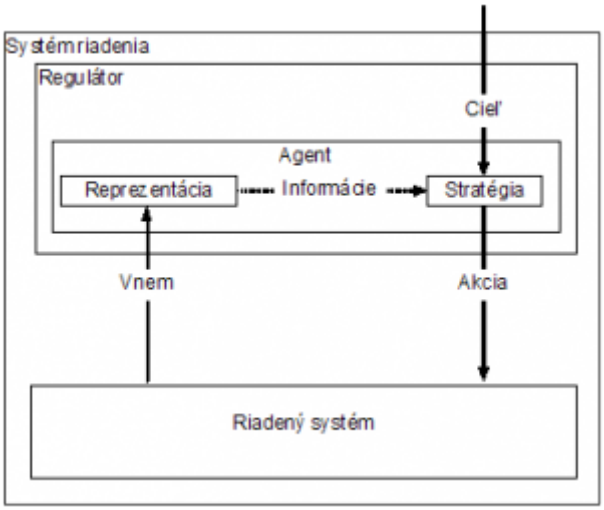
1. Agentovo orientované riadenie systémov diskrétnych udalostí

Za zakladateľov teórie DES sa považujú Peter J.G. Ramadge a W. Murray Wonham. V práci [Ramadge 89] špecifikujú DES ako dynamický systém s diskrétnym priestorom stavov a konštantnými stavovými trajektóriami, pričom prihliadajú len na postupnosť, v akej sa menia stavy a nezaujíma ich čas, kedy sa tieto zmeny udiali (toto zjednodušenie vedie k tzv. logickým modelom DES). Pokiaľ je čas zaujímavý a dôležitý pri opise správania systému, vedie to k tzv. časovaným alebo prevádzkovým (z angličtiny performanced) modelom. Tie sa navyše delia na deterministické a stochastické, v závislosti od toho, či sú známe časy výskytu udalostí, alebo ich len nahrádzame štatistickými predpokladmi.

1.1 Agent

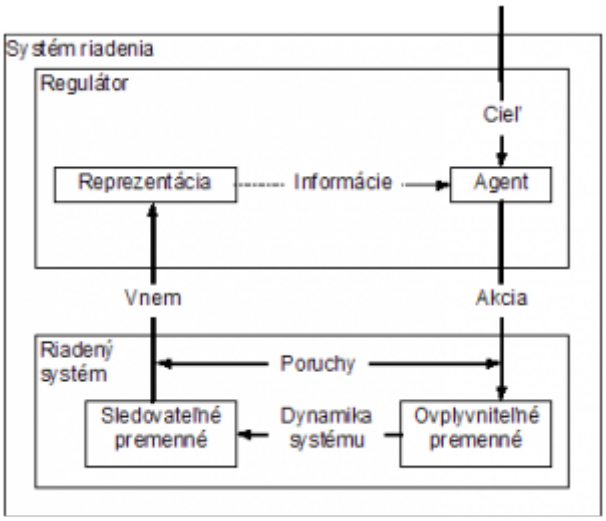
V našom prípade je pre riadenie systémov diskrétnych udalostí (t.j. oblasť, v ktorej je najväčšia miera použitia výpočtovej techniky a programových systémov riadenia) pojem agent používaný v zmysle riadiaceho systému. Existuje totiž množstvo implementácií, kde je agent použitý pre modelovanie riadeného systému (napr. modelovanie ekonomických tokov určitej komunity agentmi a vyvodzovanie vhodných stratégií pre ich ovplyvňovanie inými prostriedkami).

Na to aby agent mohol cielene ovplyvňovať prostredie, musí vedieť reprezentovať stav sledovaného systému (na základe hodnôt získaných snímačmi) a podľa toho voliť vhodnú akciu pre riadenie systému (inými slovami, musí poznať model systému, alebo očakávané reakcie systému na jeho zásahy – System Behavior).



Obr.1 Systém riadenia – prvý stupeň dekompozície

Do úvahy treba vziať aj fakt, že agent (ako každý iný riadiaci subjekt) má schopnosť vnímať a ovplyvňovať len niektoré z parametrov systému. Navyše môže nastať situácia, že nie je možné analyticky nájsť model systému a zabudovať ho do agenta. Vtedy ale možno vybudovať agenta s minimom informácií o systéme a so schopnosťou adaptácie a učenia sa (v zmysle osvojiť si správanie konkrétneho regulovaného systému) bez potreby existencie exaktného vnútorného modelu prostredia. Adaptívny agent sa opiera najmä o skúsenosti získané interakciou so systémom, čím sa tieto skúsenosti stávajú modelom prostredia.



Obr.2 Systém riadenia – druhý stupeň dekompozície

1.1.1 Členenie agentov

Agentov možno charakterizovať rozdelením do (väčšinou dvojice komplementárnych) skupín. Tie sú definované podľa poznatkov z rôznych teoretických prác a z existujúcich implementácií.

1.1.1.1 Agent osobný/verejný

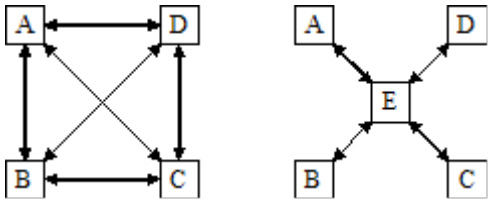
Osobný agent je prispôsobený jednému užívateľovi, ktorý ako jediný má právo zadávať príkazy. Tento typ agenta je najčastejšie realizovaný ako súčasť systému elektronickej pošty (napr. Eudora [WWW MIT], Lotus Notes, MS Outlook).

Verejný agent môže byť voľne prístupná verejná služba – napríklad knižničný vyhľadávací systém, alebo systém riadenia vozidiel individuálnej dopravy, ktorý je zdanlivo prispôsobený osobným potrebám užívateľa, avšak plní globálne definovaný cieľ, napr. optimalizáciu celkovej spotreby energie vozidiel individuálnej prepravy aj za cenu mierneho zdržania prepravovaných osôb.

1.1.1.2 Heterogénny/Homogénny

Homogénny agent je schopný komunikácie len s agentmi toho istého typu ako je on sám.

Heterogénny agent je schopný komunikácie s ostatnými agentmi (iného typu) vďaka znalosti jazyka ostatných agentov. Jediným spoločným znakom heterogénnych agentov je spoločný komunikačný jazyk.



Obr. 3 Definovanie univerzálneho jazyka

1.1.1.3 Agent Mobilný/Statický

Statický agent pri svojej činnosti operuje z jedného miesta a má prístup k svojmu prostrediu prostredníctvom statických senzorov.

Mobilný agent je schopný premiestňovať sa vo svojom prostredí a vykonávať činnosť na (geograficky) rôznych miestach.

Obr. 4 Delenie agentov podľa spôsobu použitia

2. Základné entity agenta

Agent – softvérová entita, ktorá komunikuje s ostatnými agentmi a ovládačmi za účelom riadenia robotov podľa kritérií definovaných v návrhu agenta (správanie).

2.1 Centralizované a decentralizované riadenie

2.1.1 Centralizovaný subsystém

Centralizované riadenie je charakterizované existenciou jedného komponentu – agenta, ktorý zabezpečuje koordináciu činností ostatných agentov a komunikácia je obmedzená výlučne na kanály

centrálny agent <=> agent

Na poruchu agenta musí reagovať centrálny komponent, ktorý zabezpečí prerozdelenie cieľov/plánu poškodeného agenta na ostatné agenty. Rizikom centralizovaného riadenia je zlyhanie riadenia v prípade nedostatočnej autonómie v spojení s výpadkom komunikačných kanálov.

2.1.2 Decentralizovaný subsystém

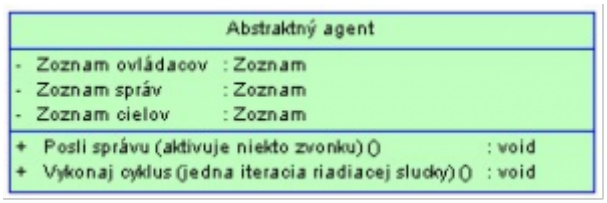
Decentralizované riadenie je charakterizované vzájomnou komunikáciou medzi agentami v zmysle P2P (peer to peer), kde je porucha agenta riešená ich vzájomnou komunikáciou a proaktívnym (negotiation) prístupom k problému. Agenty si rozdelia povinnosti poškodeného s ohľadom na minimalizáciu ceny zmeneného plánu.

2.2 Rozdiel medzi centrálnym a decentralizovaným riadením

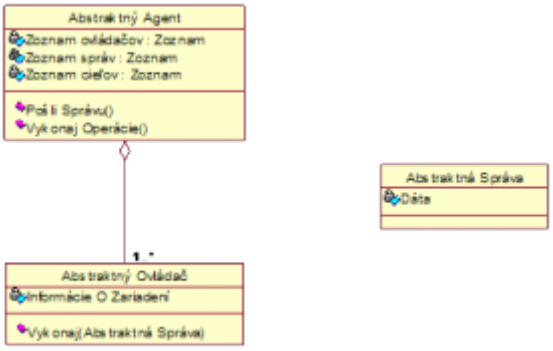
Pri hierarchickom riadení, kde sú agenti riadení centrálnou entitou (ktorá je najčastejšie tiež agentom), riadenie sa podobá typu **klient-server**. Centrálny agent len posiela príkazy po spracovaní požiadaviek klientských agentov. Keďže komunikácia cez rádio s obmedzeným dosahom nie je vhodná pre tento typ riadenia, roboty sú vybavené satelitnou komunikačnou jednotkou (agent obsahuje príslušný ovládač pre toto zariadenie). Agenti nevlastnia zoznam plánov do budúcnosti, ani algoritmus na optimalizáciu vybavovania jednotlivých cieľov. Všetky svoje požiadavky (napr.: cieľ vybavený, nový cieľ, chyba jednotlivých zariadení, odpovede na testovacie „otázky“

servera) posielajú na centrálu a po doručení odpovede vlastne len plnia rozkazy. Úlohy ako optimalizácia priebehu misie, evidencia poškodených robotov, rozdeľovanie cieľov má na starosti centrálny agent. Dôsledkom neobmedzeného dosahu komunikácie a centrálneho riadenia je vždy aktuálna „mapa“ misie (centrála vždy vie, kto je kde a čo robí).

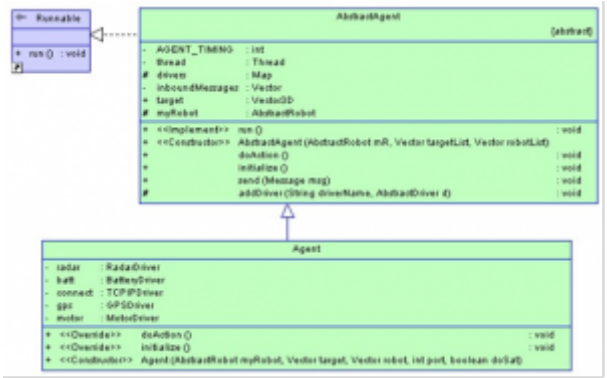
Nevýhodou je problém, ako nahradiť poškodeného robota, koho a odkiaľ tam poslať (roboty si nevymieňajú informáciu medzi sebou, aj keď sú blízko seba, nevedia o tom). Pri rovnocennom riadení, kde neexistuje nadradená úroveň, riadenie je typu **peer-t--peer**. Každý agent má rovnakú „hodnotu“, nie sú závislé od centrálnej jednotky. Vlastnia svoje plány misie (zoznam a poradie cieľov) a snažia sa optimalizovať svoju cestu. Pre takýto typ riadenia stačí komunikačné zariadenie s obmedzeným dosahom (rádio), čo má vplyv aj na spotrebu energie robota. Agenti si navzájom môžu vymieňať informácie a takto dosiahnuť lepšie výsledky. Nevýhodou riadenia peer-to-peer je to, že nie je k dispozícii okamžitý stav misie. Taktiež z dôvodu obmedzeného dosahu komunikácie nie je zaručené úspešné vybavenie všetkých cieľov (napr.: poškodený robot nemá v blízkosti nikoho, komu by mohol posielať svoj plán cieľov, aby nezostali nevykonané).



Obr. 5 Základné entity



Obr. 6 Základné triedy pre vrstvu agentov



Obr. 7 UML diagram tried balíka „agent“

Abstraktná trieda **AbstractAgent** definuje základné parametre (zoznam ovládačov,

zoznam prijatých správ, cieľ, riadený robot) a metódy agentov. Po vytvorení vlákna agenta (po vykonaní konštruktora triedy) sa riadenie dostane na metódu run(), kde sa vykonáva inicializácia agenta a potom nasleduje cyklické vykonávanie metódy doAction(), ktorá obsahuje riadiacu časť v zdedených triedach.

Trieda **Agent** je zdedená z triedy AbstractAgent . Je to už konkrétny typ agenta, ktorý má zabudovaný určitý mechanizmus správania sa. Riadenie jednotlivých zariadení (devices) robota sa uskutoční pomocou ovládačov (drivers), takže agent priamo nemá prístup k jednotlivým častiam robota. Informácie o stave zariadení sa dostanú do agenta cez ovládače pomocou správ. Spracovaním týchto správ agent eviduje stav jednotlivých zariadení a reaguje na zmeny stavov.

3. Ciele agenta

Za ciele agenta považujeme špecifickú množinu správ, ktoré určujú najbližšie vykonávané operácie. Konkrétnu definíciu správ (typ, argumenty) definuje až samotná implementácia správania agentov. Správy, ktorým agent „nerozumie“, sú ignorované.

3.1 Globálne ciele

Tieto ciele sú kladené na riadiaci systém ako celok. Napríklad v experimentálnej časti sú to súradnice miest, kam sa roboty majú presunúť. Tieto ciele roboty zdieľajú a môžu sa v ich splnení zastupovať.

3.2 Vnútorné ciele

Vnútorné ciele ovplyvňujú správanie robota, avšak nemajú vplyv na správanie iných agentov – je to napríklad cieľ „dobiť batériu“ pokiaľ jej stav nabitia klesne pod určitú úroveň. Experimentálne overenie riešenia.

4. Návrh štruktúry riešenia – riadiaca časť

Syntéza riadiaceho systému je postavená na štandardnej metodike UML [UML 05] používanej pre modelovanie procesov a softvérových systémov. Slúži na popis systémov, ich komponentov (objektov) a ich vzájomné interakcie a vzťahy použitím množstva diagramov a štruktúr. Navrhnutý agentovo orientovaný softvérový systém je následne realizovaný objektovo orientovaným jazykom Java. Objektový prístup bol použitý najmä kvôli flexibilite pri návrhu odlišných prvkov systému.

Komponenty vrstiev sú implementované ako podtriedy na báze dedičnosti z hlavnej sady tried. To dovoľuje definovať základné vlastnosti objektov a tie následne rozširovať o špecifické vlastnosti pre rôzne typy nových objektov. Tak sa tvorba nových typov robotov, zariadení alebo riadiacich algoritmov nemusí zaťažovať elementárnymi funkciami (napr. interakcia objektov alebo komunikácia). Zároveň to udržiava štruktúru celého systému konzistentnú voči požiadavkám – nie je dovolené predefinovať základné kritériá a tým narušiť štruktúru návrhu.

Proces dedenia vlastností môže byť aj viacstupňový, takže napr. malá úprava robota dovoľuje reflektovať zmenu novou podtriedou zdedenou z pôvodnej implementácie (zmena vlastností zariadenia, odstránenie alebo doplnenie zariadenia). Základná

schéma UML modelu je vytvorená z abstraktných tried, ktoré sú následne rozširované pre konkrétnu implementáciu a zabezpečuje len najnutnejšie funkcie – mechanizmus pre príjem a vysielanie správ bez akejkoľvek logiky spracovania.

4.1 Správy

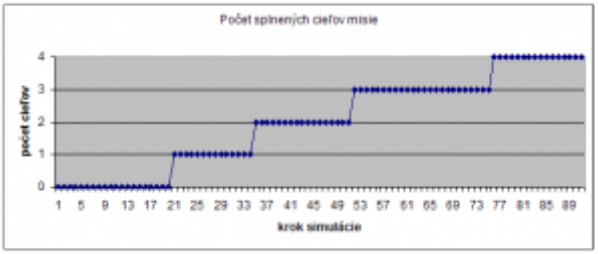
Správy ako ekvivalent udalostí používame pre komunikáciu jednotlivých komponentov. Na rozdiel od analyticky modelovaných udalostných systémov, SW implementácia umožňuje definovať rôzne typy správ s rôznou štruktúrou a doplnkovými informáciami (argumenty). Pre jednoduchosť implementácie je myšlienka správ odvodená zo štandardného princípu zasielania správ v sieti Internet (TCP/IP) pomocou datagramov (UDP). Každá správa obsahuje základnú päticu atribútov:

Odosielateľ	identifikácia objektu, ktorý správu odoslal;
Príjemca	identifikácia objektu, pre ktorý je správa určená;
Typ správy	klasifikácia správy;
Dáta	doplnkové údaje správy, ľubovoľného typu;
Príkaz	príznak že sa jedná o príkaz.

5. Priebeh experimentu

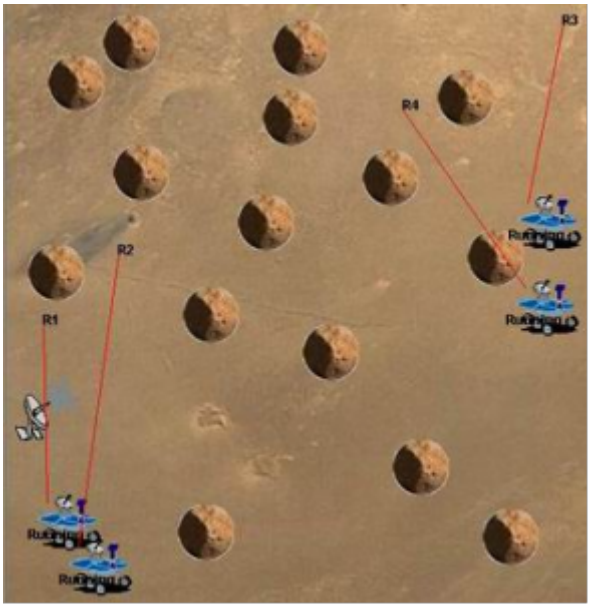
Simulačný program experimentu ako výstup generuje súbor vo formáte XLS, ktorý obsahuje všetky údaje potrebné k vykonaniu analýzy správania sa robotov. Umožňuje graficky znázorniť počet splnených cieľov v jednotlivých krokoch simulácie, čo je najdôležitejším faktorom pri vyhodnocovaní úspešnosti celej misie. Výstupný súbor obsahuje 3 + N hárkov (N je počet robotov), na ktorých sú zaznamenané nasledujúce údaje:

- **Systém** – informácie týkajúce sa samotného grafického prostredia vytvoreného v jazyku JAVA. Napr.: spracovanie vstupných súborov, priraďovanie objektov do prostredia simulácie, vytváranie objektov robotov a agentov, vytváranie zariadení jednotlivých robotov, spustenie jednotlivých časovačov. Údaje tohto hárku sú vedené v reálnom čase, aby bolo možné prípadne optimalizovať štart programu a dosiahnuť kratší čas inicializácie.
- **All Robots** – záznam udalostí celého experimentu. Je vedený na základe diskrétnych krokoch simulácie. Sú zaznamenané všetky udalosti týkajúce sa jednotlivých robotov. Tento hárok umožňuje analýzu správania sa robotov v rôznych situáciách, napr.: ako reaguje agent na chybu zariadení, atď.
- **Robot R(N)** – hárok n-tého robota simulácie. Sú to údaje podobné, ako v predchádzajúcom prípade, ale sú filtrované podľa čísla robotov. Umožňuje sledovať udalosti týkajúce sa len jedného konkrétneho robota.
- **Statistics** – najdôležitejšie informácie z hľadiska vyhodnotenia úspešnosti simulácie. Hárok obsahuje informáciu o počte splnených cieľov misie v každom kroku experimentu. Porovnaním týchto údajov v rôznych situáciách (iné štartovacie podmienky, iný terén, iný počet robotov, rôzne simulované poruchy zariadení atď.) je možné optimalizovať experiment a naplánovať misiu efektívnejšie.



Obr. 8 Graf splnených cieľov misie

Nasledujúci obrázok je fragmentom spojitého zobrazenia pohybu objektov reprezentujúcich robotov R1, R2, R3 a R4 na simulačnej ploche (povrch planéty Mars). Obrázok obsahuje štartovací stav experimentov.



Obr. 9 Obrázok simulácie

5.1 Všeobecné vlastnosti

1. Agent **komunikuje** s okolím (sníma stav systému a vykoná zásahy do systému) len na diskrétnej báze prostredníctvom **udalostí**. Z pohľadu agenta sú to abstraktné správy, ktoré prijíma alebo posiela. Problematika ovládačov už súvisí s implementáciou v konkrétnom prostredí.
2. Agent registruje svoje **„schopnosti“** ako množinu ovládačov, ktorými disponuje. Buď na základe prítomnosti (dostupnosti), alebo na základe ich stavov (indikácia poruchy) je schopný zmeniť svoje správanie a v prípade nemožnosti splnenia plánu/cieľa posunúť tento „problém“ na koexistujúce entity

Agenti disponujú dvoma režimami správania sa – v závislosti od preferovanej **komunikácie**. Jedným je striktne hierarchické usporiadanie, kde sú agenti riadení centrálnou entitou (tiež agentom) a komunikácia medzi podriadenými navzájom nie je použitá. Druhý režim spočíva v peer-to-peer (decentrálnom) komunikácii, kde je základom komunikácie každý s každým. Malo by byť možné v závislosti od okolností (napríklad nedostupnosť spojenia s „centrom“ aktivuje režim demokracie) prepínať medzi týmito pólmi správania sa, vrátane ich kombinovania – a to buď v čase (3periodicky 2sa1prepína) alebo podľa typu úlohy (niektoré ciele riešiť pre centrum, niektoré pre kolegov).

6. Záver

Článok, obsahuje krátky náhľad do oblasti riadenia robotov pomocou agentovho prístupu. Opisuje čo je vlastne agent a čo môžeme pomocou neho riadiť, že dokáže na neočakávané, chybné alebo protichodné údaje ovládač vyhodnotiť poruchu, na základe čoho agent prispôsobí plánovanie svojej činnosti, aby sa chybné zariadenie nevyužívalo. To je charakterizované ako strata schopností agenta/robota.

Príkladom využitia princípu „schopností“ sú multifunkčné obrábacie stroje, ktoré poškodením niektorého zo svojich komponentov sú naďalej schopné vykonávať obmedzenú sadu činností, čomu sa plán výrobnej linky dokáže dynamicky prispôsobiť [Kossuth 98]. Predpokladom využitia je redundancia schopností u viacerých robotov, ktorá je u multiagentových riešení samozrejmosťou.

Spouautormi článku sú Doc. Ing. Igor Hantuch, PhD., Ing. Alexandra Šlezárová,
Ústav riadenia a priemyselnej informatiky, Fakulta elektrotechniky a informatiky STU, Ilkovičova 3, 812 19 Bratislava 1