Úvod do problematiky multiagentních systémů

OBSAH:

© Ing. Arnoštka Netrvalová
(e-mail: netrvalo@kiv.zcu.cz, ZČU v Plzni, FAV, KIV - místnost: UK 412, tel.: 377632425)

1. Současný stav dané problematiky

1.1 Základní pojmy

Agentní systém (Agent Systems, AS)

Distribuovaná umělá inteligence (Distributed Artificial Intelligence, DAI)

Autonomní jednotky - aktéři (Actors) $\Longrightarrow$ **agenti (Agents)**

Autonomní agent (tzv. reaktivní) - Rodney Brooks (MIT)
Inteligentní agent (Intelligent Agent) - M. J. Wooldridge

Racionální agent - rozhoduje se na základě svých znalostí

Intenční systém (intence = záměr)

- agent sestavuje plán k dosažení vytýčeného cíle
- užitím **BDI** principu (**Beliefes, Desires and Intention**) - jednání řízeno mentálním stavem agenta

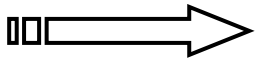
    - důvěra (v informace a v efekty akcí)
    - přání (tužby, záměry)

Nejpopulárnějším přístupem k řešení tohoto pojetí:
PRS (Procedural Reasoning Systém)

1.2 Motivace vzniku

Řešení složitých a rozsáhlých úloh:

- Zvýšením flexibility
- Zvýšením autonomie jednotlivých zdrojů znalostí v rámci modulární architektury
- Volnou integrací modulů pomocí komunikace formou zasílání zpráv
  ("peer-to-peer")
- Redukcí role centrálního prvku



multiagentní systémy

Multiagentní systém (Multi-agent system, MAS) je možno popsat jako skupinu volně propojených autonomních systémů (agentů), spolupracujících v zájmu dosažení společného cíle.

Použitím těchto systémů jako technického řešení přináší podobné výhody jako u týmové práce.

- **zkrácení doby řešení** - paralelní postup
- **snížení nároků na komunikaci** - jednotliví členové týmu si nepředávají dál všechny zjištěné údaje, ale především své závěry o nich (narozdíl od centrálně řízeného distribuovaného systému).
- **zvýšení operativnosti a spolehlivosti** - komunita může přibrat další členy (umožnit zástupy)

Poznámka:

termín "**peer-to-peer**"

v počítačových sítích

- rovnoprávné postavení při komunikace mezi dvěma procesy
- stejné právo navazování spojení
- bez rozdělení služeb na server a klient (poskytovatel a uživatel)

2. Agentní systémy

Agentní systém (Agent System, AS) je dán prostředím, kde působí agent vybavený určitou dávkou inteligence, jež je využita k řešení problému.
Agent (Agent, A) je definován jako aktivní prvek systému vytvořený člověkem za určitým účelem. Výsledkem činnosti agenta je pak změna AS z aktuálního do cílového stavu.

Rozeznáváme tři přístupy:

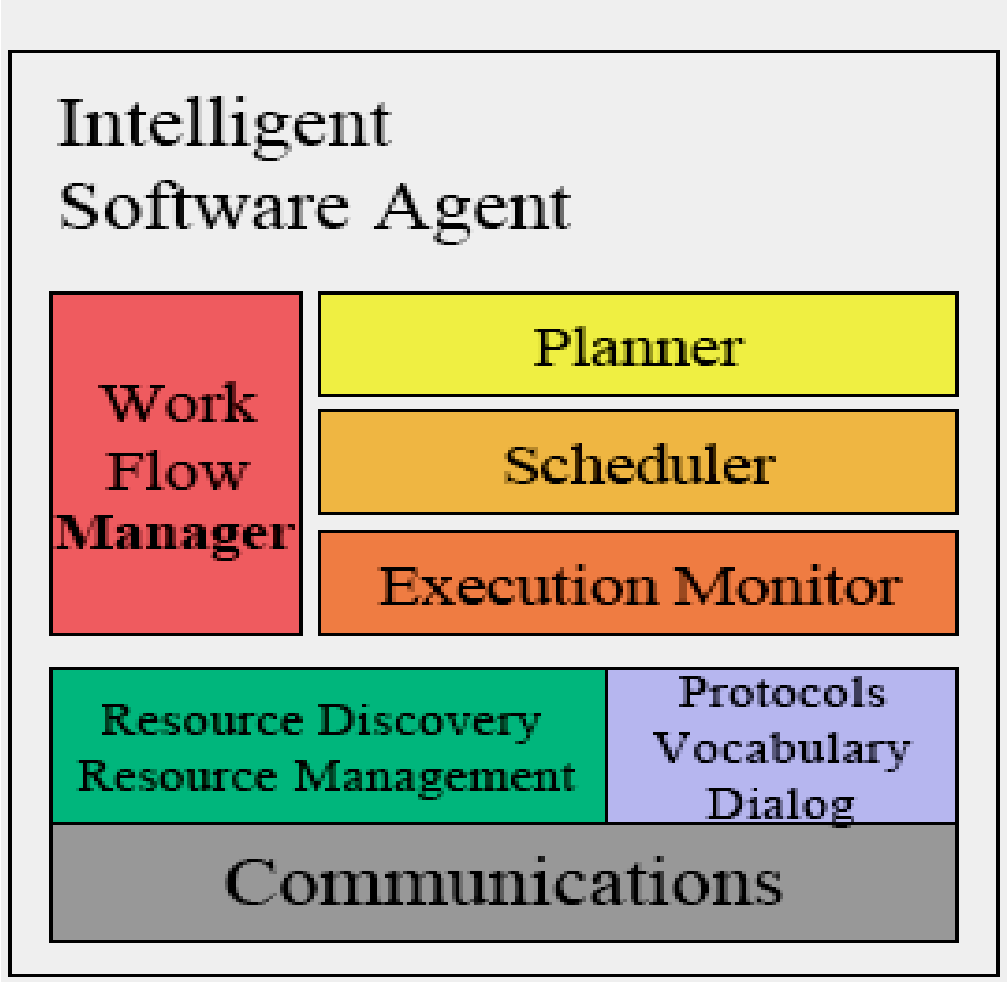
- Postupná transformace do požadovaného stavu
- Konzultace s expertem (člověkem) a pak postupná transformace do požadovaného stavu
- Zkoumání systému, hledání vnitřních zákonitostí, získání nových poznatků a pak postupná transformace do požadovaného stavu

2.1 Charakteristiky a typy agentů

Agenty lze rozdělit následovně:

1) Biologičtí agenti - lidé
2) Techničtí agenti – roboti
3) Programoví agenti – softboti (uplatnění Javy)

Softwaroví agenti - agenti v počítačových hrách, počítačové viry, agenti pro specifické úkoly, agenti – entity umělého života



Obr. 1: Schéma inteligentního softwarového agenta

Základní charakteristiky agenta:

- **Autonomnost** – agenti jsou aktivní, cílově orientované moduly, schopné samostatného řešení určitých úloh bez nezbytné komunikace s okolím, avšak schopní komunikace, koordinace činnosti či kooperace s jinými agenty v rámci určité komunity. Mají možnost vstoupit a opustit komunitu (dobrovolně), poskytnout či žádat výsledky

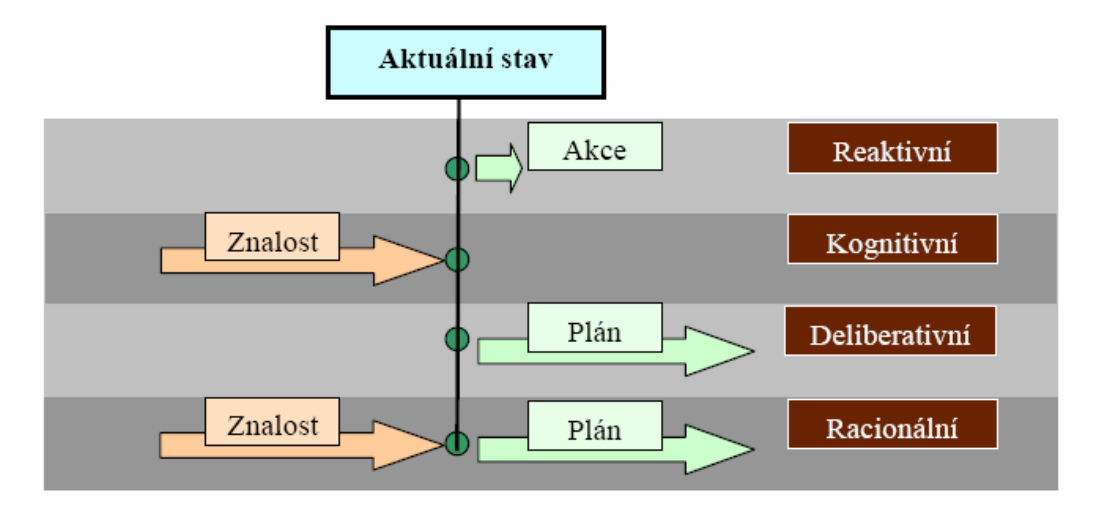
- **Reaktivita** – agenti jsou aktivováni událostmi, schopni reakce v souladu s vnímáním reálného času

- **Intencionalita** – schopnost mít na paměti dlouhodobé cíle, organizace chování k dosahování těchto cílů, formulace vlastních plánů a využití svých úsudků

- **Schopnost sociálního chování** – agenti jsou schopni spolupráce pro dosažení společných cílů, udržování informace o jiných agentech a vytváření úsudků o nich, sdružování do koalic a týmů, od nichž očekávají vzájemný prospěch

Agenty lze dále rozlišit na následující typy (obr. 2):

Agent inteligentní – má schopnost plnit cíle s využitím logické dedukce



Obr. 2 Typy agentů

- **Agent reaktivní** – má schopnost reakce na podněty
- **Agent deliberativní** (rozvážný) – má schopnost plánovat postup svých akcí, provádět výpočty, ovlivňovat prostředí tak, aby získal výhodu (proaktivita)
- **Agent kognitivní** – má schopnost vyvozovat logické závěry ze svých pozorování okolního prostředí. Je schopen se učit a vytvářet bázi znalostí (ukládá tam informace a znalosti získané dedukcí). Musí mít deliberativní schopnosti. Pak může provádět vnitřní akce jako je analýza scény, překlad a získávání dalších znalostí.
- **Agent racionální** – musí mít všechny výše uvedené schopnosti. Jeho struktura obsahuje plánovací jednotku i kognitivní jednotku včetně báze znalostí. Na základě svých poznatků je schopen se učit a pak racionálním způsobem plánovat svou činnost pro dosažení cílů.

2.2 Sociální znalosti a jejich úloha

Znalosti agenta lze rozdělit na:

- **Problémově-orientované znalosti** (problem oriented knowledge) – "asociální" typ znalostí, sloužící k lokálnímu, samostatnému řešení úloh (například poskytování expertizy, hledání ve vlastní databázi agenta atd.)
- **Znalosti o sobě samém** (self knowledge) – znalosti o vlastním chování, vnitřním stavu, závazcích apod.
- **Sociální znalosti** (social knowledge) – znalosti o chování jiných agentů, o jejich schopnostech, zatížení, zkušenostech, závazcích, o jejich znalostech, záměrech a víře

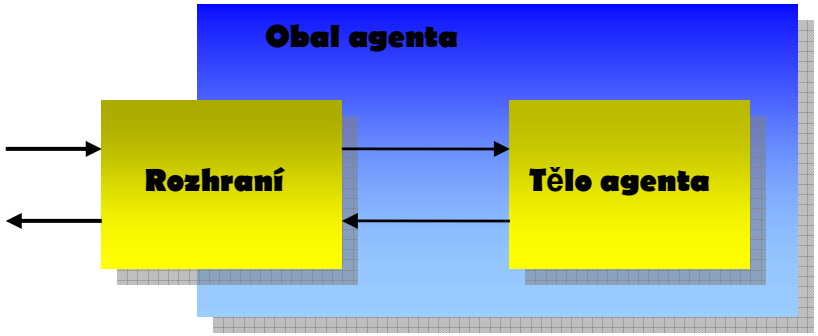
Sociální znalosti umožňují agentům:

- Delegovat odpovědnost
- Dekomponovat úlohu na jednodušší úlohy
- Kontrahovat optimálně spolupracující agenty
- Formovat týmy a koalice
- Vyhledávat chybějící informace

2.3 Architektura agenta

Agent má obvykle následující strukturu:

- **Obal** - je zodpovědný za plánování a realizaci sociálních interakcí a tvoří jej:
  - **Komunikační vrstva**
  - **Model sociálního chování** (acquaintance model)
- **Vlastní tělo** - nemá informace o komunitě



Obr. 3: Struktura agenta

Možnosti modelování sociálních znalostí:

- Centrální komunikační jednotka
- Všesměrové šíření zpráv (spojené např. s metodikou kontrahování agentů na bázi“contract-net-protocol”)
- Komunikace s využitím lokálně umístěných sociálních modelů (“acquaintance- model-based contract-bidding strategies)
  - je cílená, opírá se o znalosti udržované průběžnou, nenáročnou komunikací v nekritických časových intervalech

2.4 Prostředí (okolí)

Prostředím nazýváme vše, s čím agent přichází do styku.

Prostředí může být:

- **Plně nebo částečně pozorovatelné**
  (plně pozorovatelné je prostředí tehdy, může-li agent sledovat jeho kompletní stav)

- **Statické nebo dynamické**
  (statické prostředí se může měnit pouze s akcemi agenta)

- **Deterministické nebo nedeterministické**
  (v deterministickém prostředí je jeho stav dán pouze předchozím stavem a vykonanou akcí)

- **Diskrétní nebo spojité**
  (diskrétní prostředí má konečnou nebo spočetnou množinu stavů – například binární řetězec údajů ze senzorů)

2.5 Nástroje pro návrh

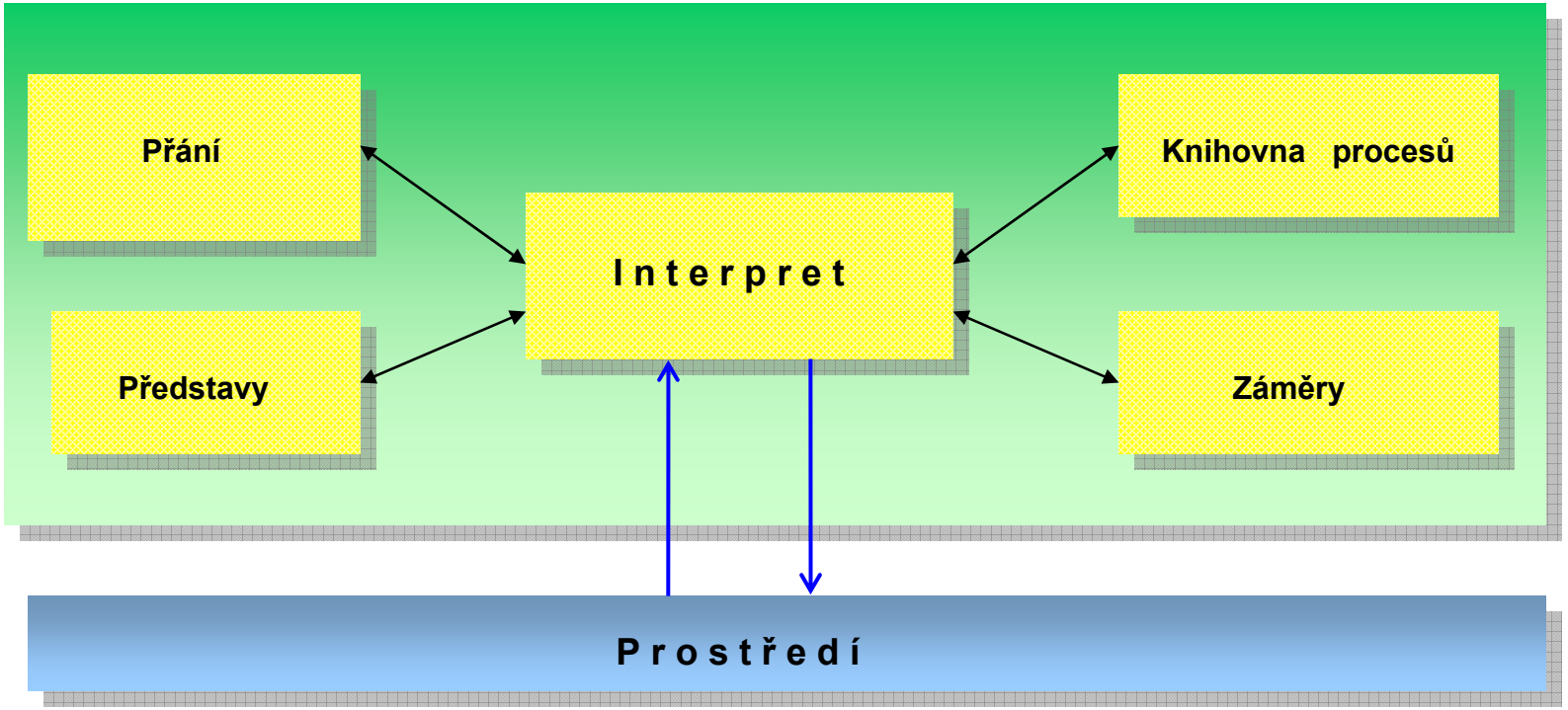
Pro účely popisu mentálních stavů agenta jsou používány **formální logiky** (např. BDI logika).
Algoritmus podle Wooldridge:
Plán je posloupnost akcí vedoucí k dosažení nějakého záměru. Záměr je vybrán dle agentova přání v závislost na aktuálním stavu prostředí. Může docházet k přehodnocování plánu, filtraci záměrů, kontrole, zda plán odpovídá záměru.

Základní algoritmus:

```
opakuj  {
    přijmi vjem z okolí
    uprav vnitřní model prostředí
    vyber záměr
    sestav plán pro dosažení záměru
    spusť plán
    }
```

Tento základní algoritmus je pouze ukázkou možného přístupu k realizaci agenta jako entity řízené nekonečným cyklem.

Jako nástroj pro návrh je používána PRS architektura. V ní je navíc zaveden pojem knihovna procesů (tj. aplikovatelných plánů). Parciální plány (procedurální znalosti) jsou pak posloupnosti akcí, transformující systém z počátečního do cílového stavu. Každý takovýto plán má podmínku spuštění, tělo (posloupnost akcí) a konečný efekt, kdy celý proces rozhodování agenta je řízen interpretem (viz obr.4).



Obr. 4. Blokové schéma architektury PRS

3. Standardizace pojmů, algoritmů a protokolů

FIPA (Foundation for Intelligent Physical Agent)

- sjednocení výzkumného úsilí.

Standardy:

- specifikace životního cyklu agenta
- mobilita agentů
- komunikační jazyky
- komunikační protokoly
- bezpečnost v MA systémech

Specifikace:

- FIPA architektura agenta
- realizace agenta jako internetové služby

Abstraktní architektura zahrnuje principy společné pro většinu vytvořených agentních systémů, ale **netýká se plánování a racionálního jednání**.

Na obr. 5 jsou zobrazeny základní bloky **specifikace správy agenta (AMS, Agent Management Specification)**.

Obr. 5: Blokové schéma specifikace správy agenta

V rámci jedné platformy existuje jedna AMS a agent v ní musí být **registrován**. Agenti mohou využívat vlastnosti systému pro registraci služeb v adresáři služeb a umístění v adresáři agentů a komunikovat s jinými agenty prostřednictvím jazyku a transportu zpráv.

Tři základní směry ve sjednocovacím úsilí:

3.1 Sociální chování agentů v MA komunitě

Každý agent se zapojuje do skupiny a respektuje určitá pravidla, sdílí prostředky, nabízí služby, přijímá závazky a koordinuje svou činnost v souladu s globálními cíli skupiny (**kooperace agentů**).

Mechanismus vzájemného dorozumívání podporuje vznik **komunikačních jazyků** jako KMQL (Knowledge Query Manipulation Language) a ACL (Agent Communication Language).

3.2 Návrh metodologie tvorby agentních systémů

Při vytváření modelu se používá **objektově orientovaný návrh** agenta. Zde je agent chápán jako objekt **rozšířený o schopnost vlastního řízení.**

Používají se **jazyky UML** (Unified Modelling Language), AUML (Agent-based Unified Modelling Language), nebo nástroj Gaia , kde je agent entitou s rolí v systému, zodpovědností a přidělenými zdroji.

3.3 Bezpečnost agentních systémů

Pro komerční využití je pak nezbytné řešení otázky **bezpečnosti agentních systémů**, zejména vzhledem k jejich zpřístupnění široké veřejnosti (důležitost autentikace).

4. Sociální aspekty v multiagentních systémech

Multiagentní systémy (Multi-agent systems, MAS) jsou takové systémy, kde se v prostředí pohybuje více než jeden agent.
Multiagentní problematika s sebou přináší nová témata, například **témata sociální**, jako je vytváření skupin na základě společných zájmů. Tudíž musí nutně docházet ke **spolupráci agentů** a jsou tedy potřebné prostředky pro vzájemnou **komunikaci**.

4.1 Společné schopnosti agentů a jejich zájmy

Agenti musí mít společné:

- **Vnímání pojmů**
- **Pravidla vzájemné komunikace**
- **Jazyk**

Agenti mohou navzájem **ovlivňovat svoje chování**, svou činnost provádět **v souladu s ostatními** nebo pouze činnost ostatních **nenarušovat** anebo dokonce **proti** ostatním **působit.**

Dále lze agenty dělit na:

- **Kooperativní** – mají společné cíle
- **Kompetitivní** – mají protichůdné cíle
- **Kolaborativní** – navzájem spolupracující

Nejčastěji mají agenti **partikulární zájmy** a dochází pak k **souladu** nebo **kolizím záměrů** jednotlivých agentů.

4.2 Strategie agentů

Strategie agenta udává, která akce z báze akcí bude vykonána jako reakce na aktuální stav prostředí.

Dominantní strategie je pro agenta nejlepší strategie nezávisle na zvolených strategiích ostatních agentů. Racionální agent pak volí vždy dominantní strategii.

Nashova rovnováha
Strategií skupiny je tzv. Nashova rovnováha, pokud každá ze strategií je nejlepší individuální strategií příslušného agenta vzhledem ke strategiím zvoleným ostatními agenty.

Volba strategie vedoucí k **optimálnímu zisku** celé skupiny:

- **koordinace** jednání agentů
- **komunikace**
- **vůle v prospěch** celé skupiny

Skupina agentů - společné mentální postoje – všem známé

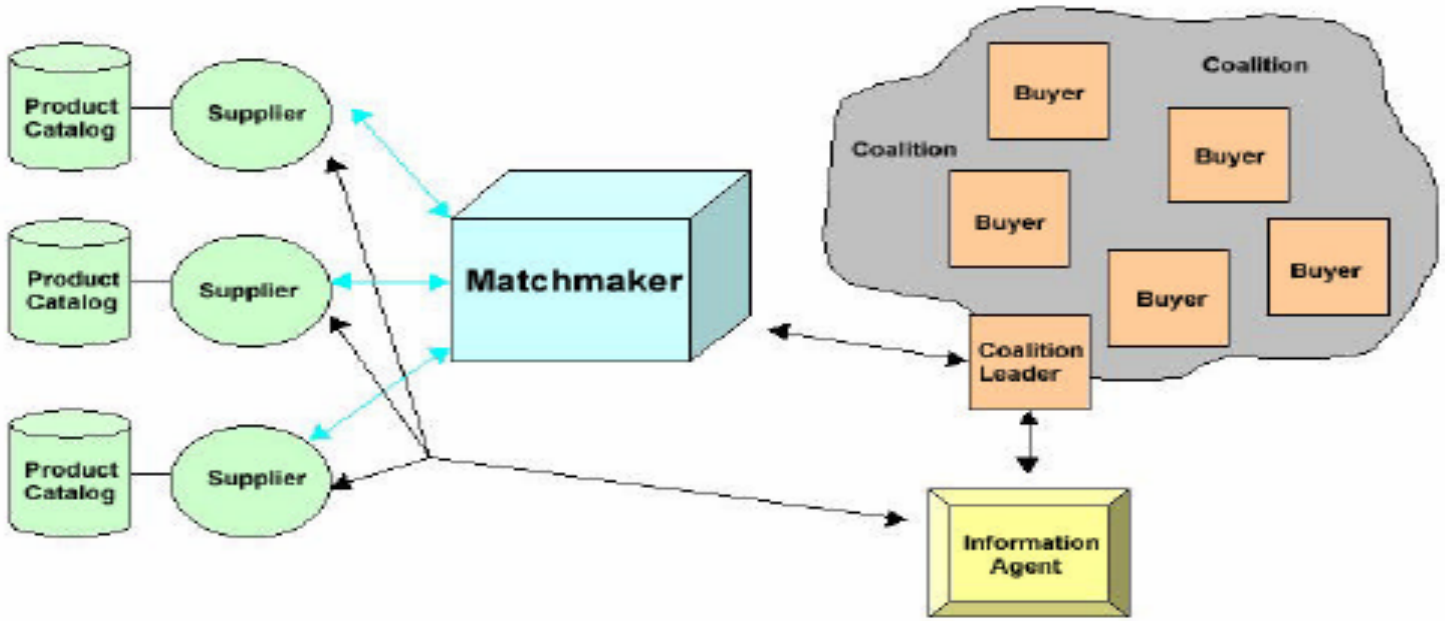
4.3 Dohody a koalice

Základ pro vytváření dohod a koalic:

- společné mentální postoje
- agenti se závazky a normami
- schopnost vzájemné komunikace
- koordinace individuálních jednání
- hledání společných strategií
- dosažení cílů ve společném zájmu
- bázích znalostí (základní pravidla spolupráce)
- plánování činnosti

Racionální agent vstupuje do závazku tehdy **očekává-li zisk**. Dohodou dosáhnou agenti lepšího stavu prostředí.
Spolupráce i kompromis se týká **sdílení cílů, sdílení prostředků a poskytování informací**

Coalition Formation for Volume Discounts



Obr. 6: Schéma funkce dohadovacího systému s vytvořenou koalicí

4.4 Závazky

Agenti vyjadřují vůli spolupracovat prostřednictvím závazků. Závazek je udržování mentálního postoje.

Typy závazků:

- **Dosažení cíle** – např. nalezení protiopatření na riziko
- **Status Quo** – např. závazek informovat při zjištění rizika
- **Obecně podmíněný**

Ve většině případů jsou mezi agenty sjednávány závazky podmíněné. Výjimku tvoří normy skupiny.

Sociální skupina agentů je dána **normami a agenty**, kteří tyto normy sdílí po celou dobu své existence v této skupině.

Skupiny lze kategorizovat dle důvodů zájmů o spolupráci na:

- **Skupiny se sdílením cílů**
- **Skupiny se sdílením prostředků**
- **Skupiny s poskytováním informací**

5. Komunikace v multiagentních systémech

V MAS tvoří okolí agenta i ostatní agenti, kteří jsou v tomto systému. **Záměrem** agenta tedy může být i změna vnitřního (mentálního) **stavu jiného agenta**. Existují dva způsoby ovlivnění:

- **Nepřímé** – agent **mění stav okolí** jiného agenta, aby při kontaktu s okolím změnil svůj postoj žádaným směrem
- **Přímé** – agent působí přímo a jediným možným způsobem je **komunikace**

5.1 Dialogy a zprávy

Základních typy dialogu:

- **Dotazování** – hledání informace tam, kde agent věří, že ji najde.
- **Hledání informace** - společné pátrání agentů po informaci, kterou nemají
- **Přesvědčování** - snaha agenta o získání jiného agenta pro svůj záměr
- **Vyjednávání** - agenti vyjednávají o podmínkách sdílení prostředků, o poskytnutí služeb pro maximální zisk
- **Porada** - cílem je nalezení řešení problému v zájmu všech(znalosti, schopnosti, usnášení se)
- **Eristický dialog** - expresivní výměna informace (hádka)

Komunikace – proces - dva nebo více agentů si vyměňuje informace ve formě elementárních komunikačních zpráv (řečových aktů). Každá elementární zpráva má odesílatele, příjemce, obsah a informaci o typu, který určuje význam obsahu zprávy.

Typy elementárních zpráv jsou:

- **Otázka**
- **Nabídka**
- **Zamítnutí**
- **Informování**

Nalézt partnera vhodného pro záměr - nástroje pro DS:

- **vysílání (broadcasting)**
- **nástěnka**
- **adresáře služeb**
- **speciální agenti pro zprostředkování komunikace**
- **sociální modely**

Vysílání uvnitř skupiny ---> vysílání ostatním skupinám

5.2 Komunikační jazyky

Každý jazyk je dán **abecedou, syntaxí a sémantikou**.

Sdílení informace probíhá na třech úrovních:

- **syntaktická** – problém je jednoduchý, syntax lze snadno definovat
- **sémantická** – tvorba znalostních ontologií
- **pragmatická** – s kým hovořit, kde ho nalézt, jak iniciovat komunikaci - není složité

V současné době je hlavním jazykem pro komunikaci agentů ACL(Agent Communication Language). Tento jazyk vychází z jazyka KQML(Knowledge Query Manipulation Language).

5.2.1 KQML

Jazyk KQML vyjadřuje jeden řečový akt, který se liší podle typu (dotaz, nabídka, otázka, souhlas, odmítnutí, …).

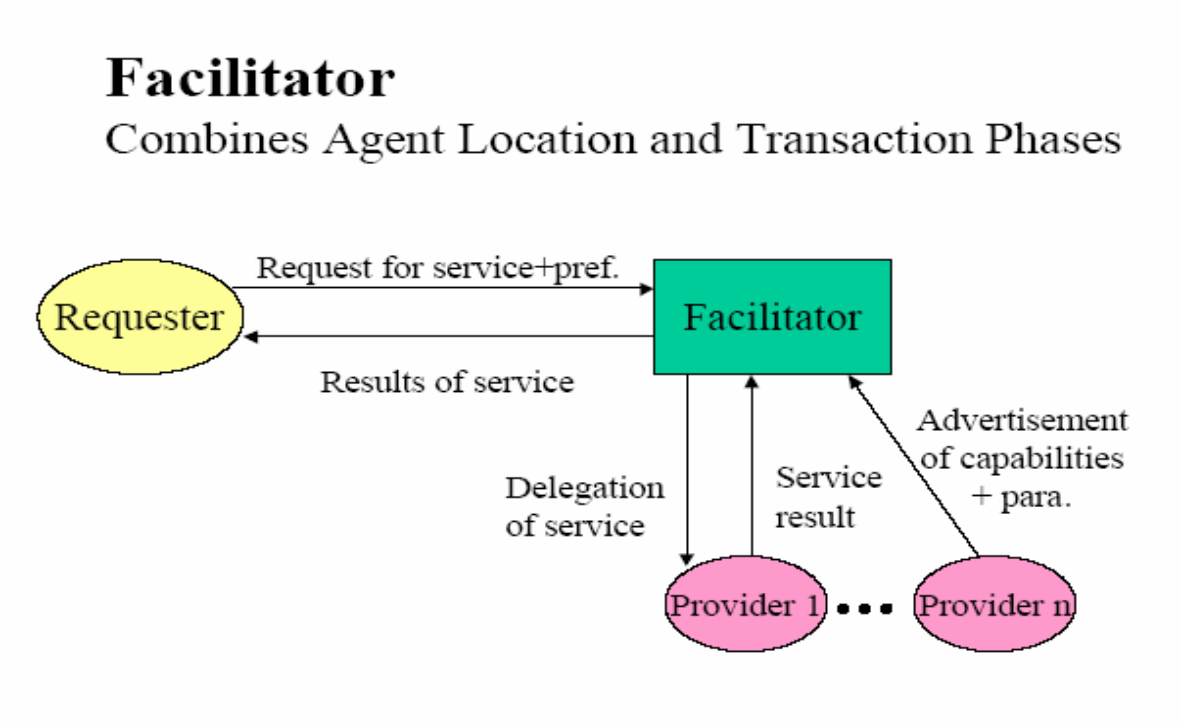
- první pokus o formalizaci a standardizaci

Zpráva:

- identifikátor určující, o jaký řečový akt se jedná
- jednotlivé prvky zprávy – dvojice ve tvaru identifikátor (klíčové slovo) a obsah.

Základní principy:

- **performativy** - základních komunikačních akty
- **facilitátoři** (tzv. "centrálních" agentů), kteří poskytují komunikační služby (obr.7)



Obr. 7: Schéma funkce facilitátora

Každá zpráva sestává ze jména performativu a jeho parametrů, vlastní obsah zprávy může být zapsán v libovolném jazyku a představuje obsah jednoho parametru

Typická zpráva:

```
(ask-one
:sender    agent A
:receiver  server_ceník
:language PROLOG
:ontology SCP_seznam
:content “cena(unipetrol,CENA).”)
```

Obsahem performativu může být další perfomativ jako požadovaná forma odpovědi, může dokonce docházet k vícenásobnému vnořování
Tři typy perfomativů:

- **Performativy pro vedení diskuse:**

  ask-if, ask-all, ask-one, tell, untell, deny, insert, uninsert, delete-one, delete-all, undelete, advertise, unadvertise, subscribe

- **Performativy pro zasahování do diskuse:**

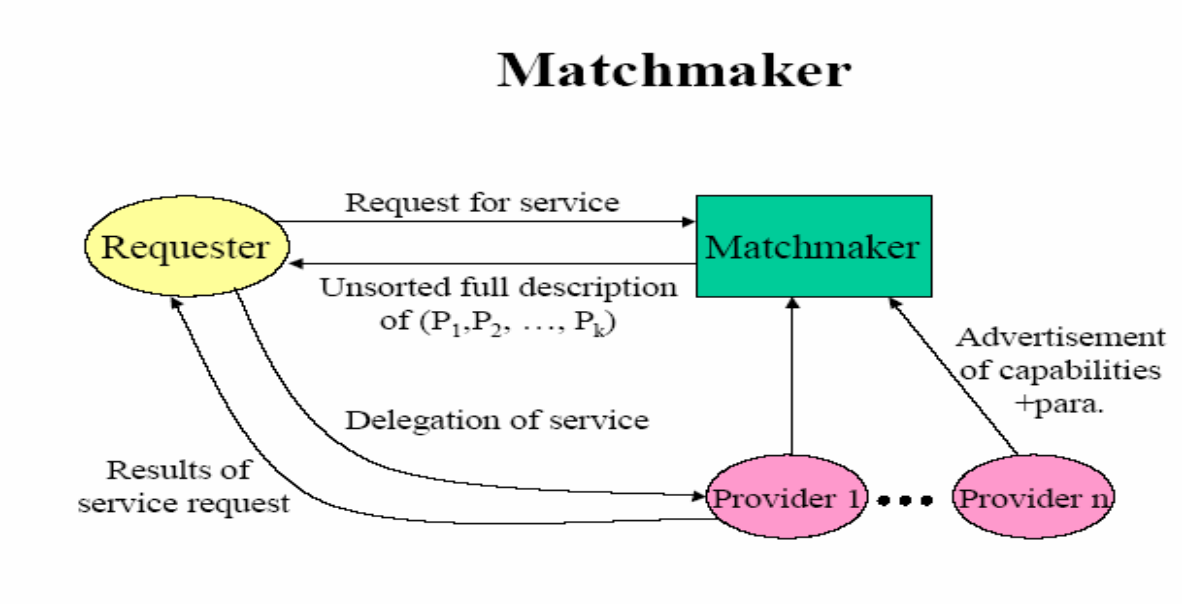
  error, sorry, standby, ready, next, rest, discard

- **Performativy pro síťování a podporu facilitátora:**

  register, unregister, broadcast, forward, transport-address, broker-one, broker-all, recommend-one, recommend-all, recruit-one, recruit-all

Jazyk KQML předpokládá, že většina dohadovacích procedur mezi agenty probíhá nepřímo, prostřednictvím facilitátora. Každý facilitátor zodpovídá za jednu doménu, veškerá komunikace vně domény jde výlučně přes příslušné facilitátory. Velmi často se používá kombinace performativů advertise-subscribe
Tři dohadovací mechanismy – (role agentů):

- **matchmaker (obr.8)**
- **broker**
- **mediator**



Obr. 8: Schéma funkce matchmakera

5.2.2 ACL

FIPA ACL - z principů KQML: opět se používá performativů a jejich syntaxe je téměř stejná jako v KQML. Obsah **performativů** je však jiný, FIPA jich stanovila pouze **uzavřenou množinu (20)** a nové perfomativy mohou vzniknout jen přípustnou kombinací těch základních
Pro každý z řečových aktů může existovat formální zápis, vyjadřující jeho efekt na komunikačního partnera. Každý typ zprávy má určenu podmínku použití a dopad na mentální stav příjemce, nazývaný **racionální efekt** (Racional Effect, RE). Racionálním efektem je tedy skutečnost, že **agent věří v platnost formule.**

Sémantika závisí na informacích typu ***belief***, které nebývají přístupné jiným agentům a často se mění. Ve FIPA-ACL nejsou k dispozici performativy pro síťování a administraci zpráv, FIPA tento problém přenechává implementátorům.

FIPA vydává standardy

- **normativní** - FIPA-AA (Abstract Architecture), FIPA-ACL, FIPA-SL, FIPA-KIF (Knowledge Interchange Format), FIPA-RDF (Resource Description Framework), FIPA-AMT (Agent Message Transport)
- **informativní** - Agent SW Integration, Personal Travel Assistance, Personál Assistant, Audio-visual Entertainment and Broadcasting, Network Management and Provisioning, AgentCities, úsilí v oblasti holonických výrobních systémů

Open-source implementace: JADE, FIPA-OS, APRIL, ZEUS

Protokoly spolupráce agentů

FIPA-ACL podporuje standardní protokoly spolupráce agentů:
Contract Net Protocol, anglická dražba, holandská dražba, obálková metoda, aukce typu Vickrey

Agent Oriented Programming

- besides logic component AOP is based on **societal view** on computation. It suggest of programming agents in terms of mentalistic notions.
- AGENT0 programming language allows you to represent agents **knowledge**, **capabilities**, **commitments** and **commitments rules**

```
committ
(
     (agent, request, do(time, action)),  message condition
     (       bel(now, Friend agent) and,   mental condition
             can(self, action) and
             not(time, commitment(self, anyaction),
     self,
     do(time, action)
 )
```

Obr. 9: Agentově orientované programování - ukázka

5.3 Metody navazování komunikace

Agent má k dispozici jazyk, dokáže formulovat zprávu a určit její význam.
Možné metody navazování komunikace

- **Vysílání**
- **Nástěnka**
- **Komunikace pomocí prostředníka**
- **Sociální model**

5.3.1 **Vysílání** (Broadcasting)

v distribuovaných systémech
V MAS agent rozešle žádost o navázání komunikace všem agentům, s nimiž má k dispozici komunikační kanál.

Výhody a nevýhody:

- **Snadné nalezení vhodného partnera (existuje-li)**
- **Vysílání zpráv zahlcuje informační kanály**
- **Zpracování zprávy zpomaluje činnost systému**

5.3.2 **Nástěnka (Black Board)**

Nástěnka je pasivní sdílený prostředek ke zveřejnění informace. Součástí strategie je pozorovat nástěnku a reagovat na vložené požadavky (moc se neužívá).

5.3.3 Komunikace pomocí prostředníka

Prostředníka - **mediator, broker** nebo **facilitator**.
Obecně agent - prostředník získává informace o schopnostech, záměrech a přáních jednotlivých agentů a pokud je některým z nich požádán, vyhledá mu dostupného partnera, který má požadované schopnosti nebo znalosti. Rolí prostředníka může být i moderování diskuse ve skupině, řízení hlasování a dražeb.

5.3.4 Sociální model

Spojuje přístupy - vysílání a komunikace pomocí prostředníka. Agent hledá spolupráci pouze uvnitř komunity nebo s jistou skupinou agentů s normami.
Princip hledání partnera:

- **Vysílání v rámci skupiny**
- **Přijetí závazku – role prostředníka**
- **Šíření požadavku v jiných sociálních skupinách**

Agent nejprve hledá partnera uvnitř skupiny prostřednictvím vysílání. Pokud takto partnera nenalezne, nabídne mu prostředník zprostředkování požadavku v jiné skupině. Pokud agent tuto nabídku akceptuje, prostředník přijme požadavek za svůj a vysílá jej v jiných skupinách, kterých je členem.

5.4 Interakční protokoly

- stanovení pravidel pro vedení dialogu

  - **kdy a za jakých podmínek lze v diskusi odpovídat**
  - **jakými typy zpráv**
  - **kdy dialog končí**
  - **s jakými výsledky**

Pravidla se liší podle druhu komunikace. Při vyjednávání o prostředcích nebo nabídce služeb bude komunikační protokol vycházet z principu **dražeb**. Pro rozhodování o koordinaci postupů je vhodné použít protokolů **hlasování**. Dalším typem komunikace je **argumentace**, kdy agenti podporují svoje argumenty proti argumentům jiného agenta.

Typy interakčních protokolů:

- **Kontraktní síť**
- **Dražba**
- **Hlasování**
- **Argumentace**

5.4.1 Kontraktní síť

Tento protokol se týká debaty o vykonání služby a nazývá se protokol kontraktní sítě (Contract Net Protocol, CNP). Podobá se vysílání. Princip komunikace v kontraktní síti je takový, že požadavek na službu je **vysílán iniciátorem** a část agentů na tento požadavek zareaguje tím, že navrhne jeho **vykonání za nějakou cenu**. Iniciátor informuje agenty na jejichž požadavky přistoupil a navrhne jim cenu. Část agentů s navrženou cenou **souhlasí**, ostatní vykonání požadavku **odmítnou**. Po vykonání požadavku je iniciátor informován příslušnými agenty či jejich skupinami o **výsledku** (úspěchu, neúspěchu, o hodnotě výsledku).

5.4.2 Dražba

Dražba je interakční protokol pro sjednávání služeb nebo sdílení prostředků. Problém cen a platidel se modeluje obvykle tak, že každý agent má v okamžiku svého vzniku přidělenu **počáteční sumu prostředků**, se kterou obvyklým způsobem hospodaří (za úplatu využívá nebo poskytuje služby a prostředky).

Specifikace protokolu dražeb vychází z principu dražeb, známých z reálného života:

- **Anglická dražba**
- **Holandská dražba**

V případě **anglické dražby** je zadána vyvolávací cena a účastníci dražby postupně tuto cenu zvyšují. To se opakuje tak dlouho, dokud nikdo nepodá návrh s vyšší cenou, než je aktuální. Protokol popisuje dražbu tak, že na začátku jsou agenti informováni o zahájení dražby a jsou žádáni o návrhy nových cen. Dražba pokračuje až do okamžiku, kdy již nikdo nezvyšuje nabídku. Pak dražba končí a účastníci jsou informováni o jejím výsledku.

V případě **holandské dražby** je na počátku stanovena nadnesená cena a ta se snižuje tak dlouho, dokud se nenajde zájemce nebo dokud cena nedosáhne úrovně, pod kterou už dražitel nesmí jít (NoBid).

5.4.3 Hlasování

Hlasování řeší problém strategického rozhodování v rámci skupiny, není však standardem FIPA. Hlasování předpokládá, že existuje množina prvků a cílem skupiny je **vybrat** z ní **jeden prvek**. Hlasování je rozděleno do dvou etap:

- **Shromažďování návrhů**
- **Hlasování o návrzích**

Předpokládejme rozhodování o strategii skupiny. Výsledkem musí být přijetí množiny plánů pro jednotlivé agenty, jejichž vykonáním skupina dosáhne některých ze svých záměrů.

V první etapě je **vybrán jeden agent** (elektor), který bude řídit volbu.
Tento agent **vyhlásí volbu**, požádá o návrhy strategií a informuje o výsledku (vytvoření báze strategií).
Po představení možných strategií informují jednotliví **agenti** elektora, které ze strategií **dávají svůj hlas**.
Elektor následně informuje všechny členy skupiny o tom, **která strategie byla vybrána**.

Uvedený postup neřeší problém rovnosti hlasů (elektor může vybrat jednu z vítězných strategií sám nebo náhodně).

5.4.4 Argumentace

Argumentace je dialog, kdy agenti diskutují o pravdivosti informace a tuto pravdivost podporují nebo vyvracejí **argumenty**.
Je třeba nalézt argument a specifikovat podmínky, za kterých má být informace přijata nebo zamítnuta.
Dialog probíhá tak dlouho, dokud jeden z aktérů nepřistoupí na argumenty druhého.

6. Správa sociálních znalostí a její modely

6.1 Správa znalostí

Kooperační báze - permanentní znalosti

Báze úloh:

- **Problémová sekce** - je permanentní
- **Plánovací sekce** - je spravována v průběhu přeplánování

Báze stavů - je udržována buď formou:

- **Periodických revizí** – jedná se o pravidelnou výměnu informací v časových úsecích, kdy není komunikační infrastruktura příliš zatížena
- **Performativů** - avdertise-subscribe (asynchronní údržba)

6.2 Vylepšování znalostí

- Vylepšování znalostí **na úrovni jednotlivých agentů** – agent se sám učí, optimalizuje, reorganizuje svoji činnost, samostatně vyvozuje a mění i permanentní znalosti
- Vylepšování **na meta-úrovni** – realizuje se za pomoci nezávislého agenta, zvaného meta-agent, jenž pozoruje aktivity celé komunity nebo její části, zobecňuje posbíraná data a poté zobecňuje získané poznatky

Vylepšování znalostí je jednou z forem "sociálního uvažování".

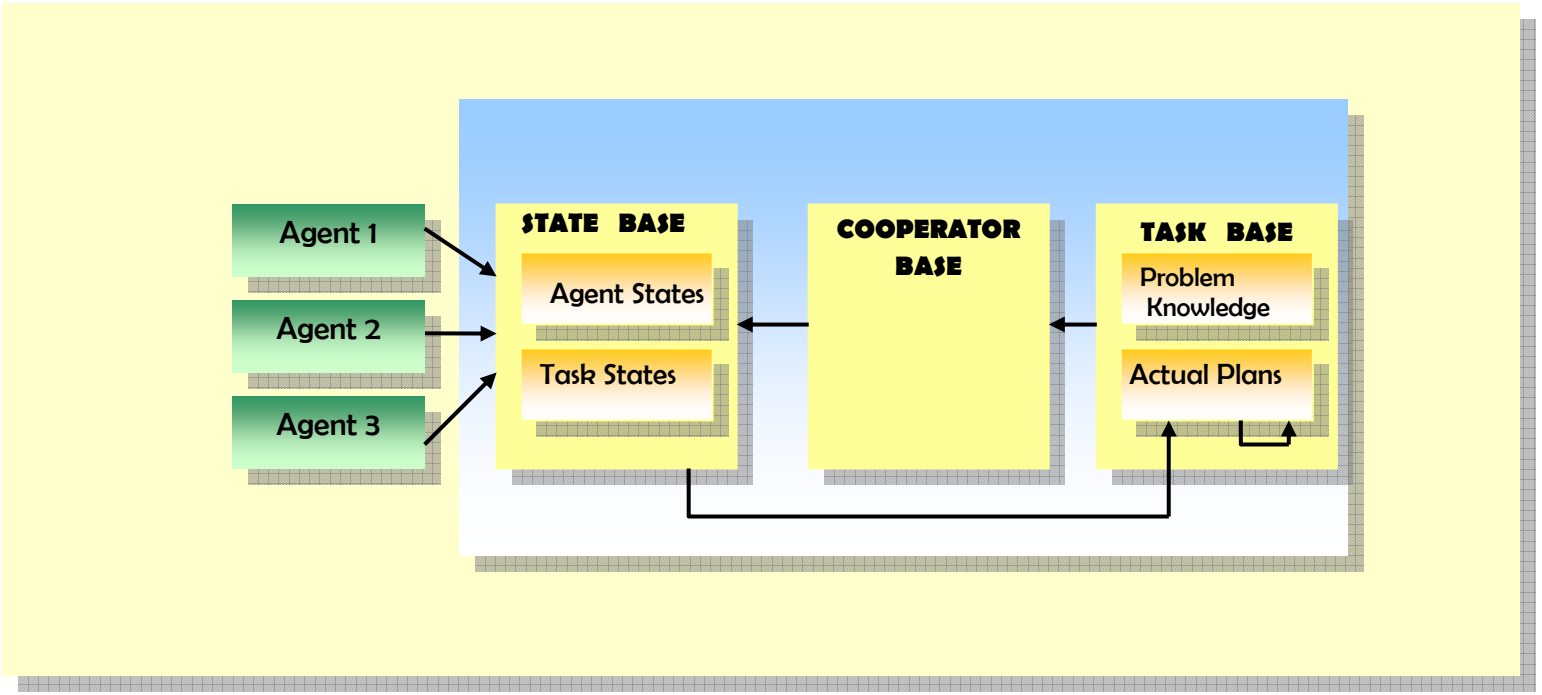
6.3 Modely

Používají se **dva** základní modely:

- **twin-base** model (Cao, Hartwigsen)
- **tri-base** model

Tri-base acquaintance model (3bA) - viz obr. 10:

- **kooperační báze** - permanentní/semipermanentní znalosti (adresa, jazyk, schopnosti agentů)
- **báze úloh** - sestává ze sekce problémové (obecné znalosti pro dekompozice úloh) a ze sekce plánovací (konkrétní instance pravidel s ohledem na aktuální zatížení a schopnosti)
- **báze stavů** - obsahuje stále se měnící informace o spolupracujících agentech a o stavu řešení úloh



Obr. 10: Tri-bázový model

7. Plánování (Planning, Scheduling)

Jednou z instancí chování inteligentního agenta je vzájemný vztah mezi:

Obr. 11: Vzájemný vztah agent, prostředí, úloha

Plán není nic jiného než úvaha o hypotetickém vzájemném vztahu mezi agentem a prostředím při respektování dané úlohy. Motivací tohoto procesu je **vytvoření možného sledu akcí**, které **pro dosažení záměru** změní prostředí.

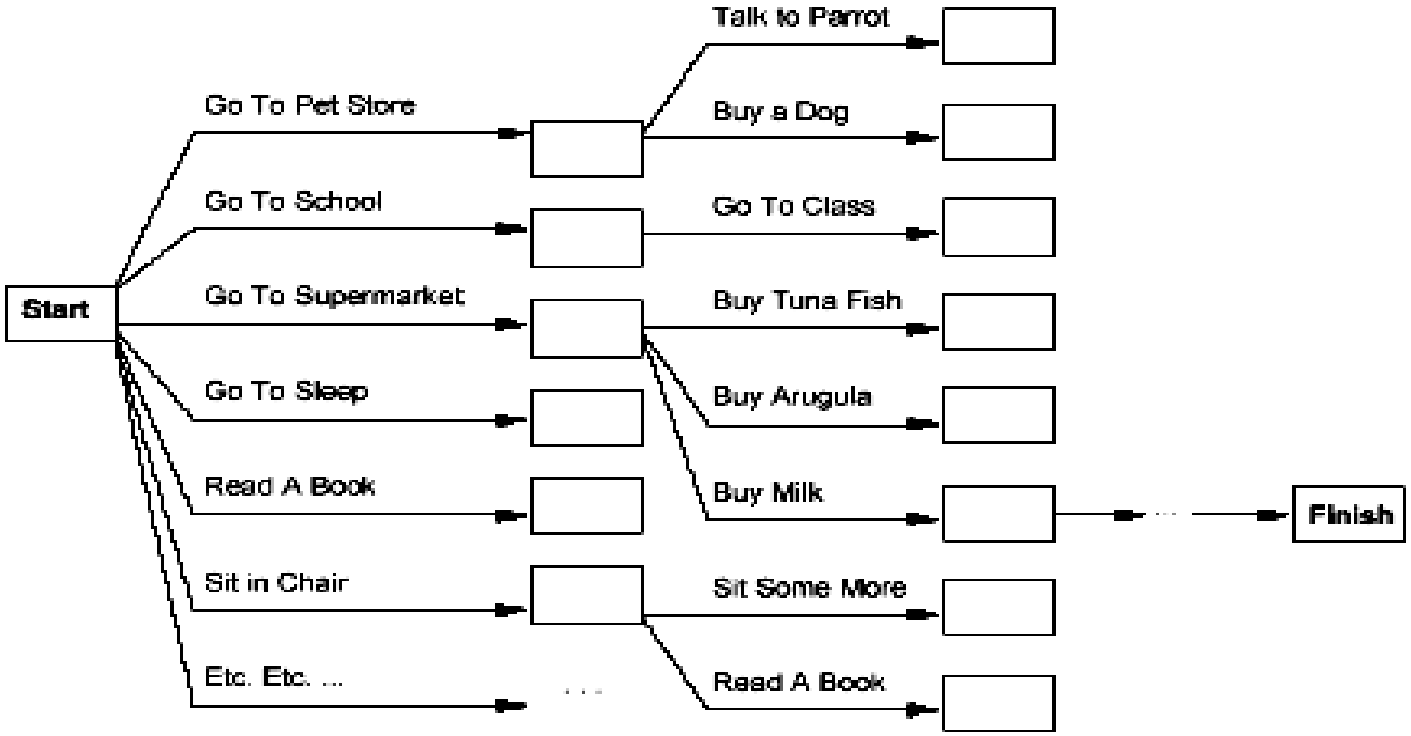
7.1 Planning versus Scheduling

- **plánování (scheduling)** - přiřadí prostředky v čase jednotlivým procesům
- **plánování (planning)** - bere v úvahu možné interakce mezi komponentami.

Na následujícím obrázku (obr. 12) je uveden příklad plánování:

- Uzly - stavy
- Šipky - akce
- Start a Finish - počáteční a koncový stav

Cesta řešení pak určuje plán



Obr. 12: Plánování

Vlastnosti plánu

Plán je **správný** tehdy, poskytuje-li **řešení záměru** a je-li **celistvý a konzistentní**.

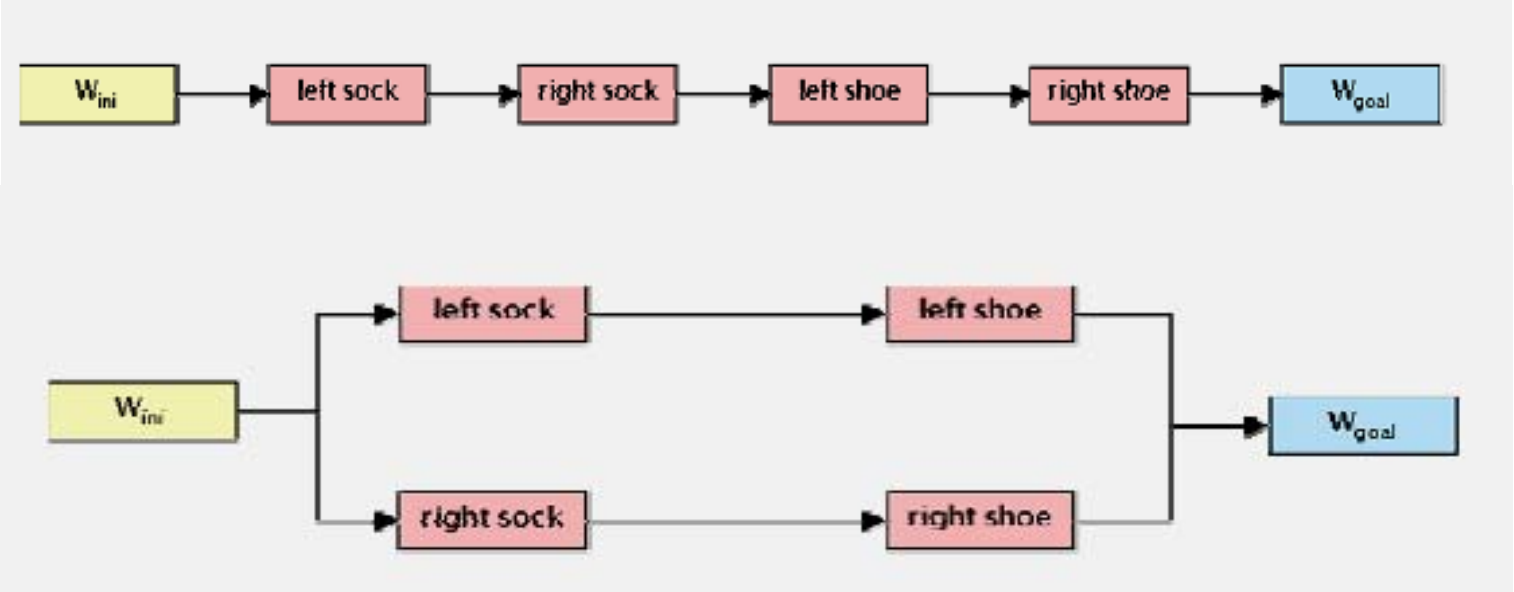
7.2 Reprezentace plánu

Výsledkem může být plán:

- **částečný** (parciální)
- **úplný**

Další dělení (viz obr. 13):

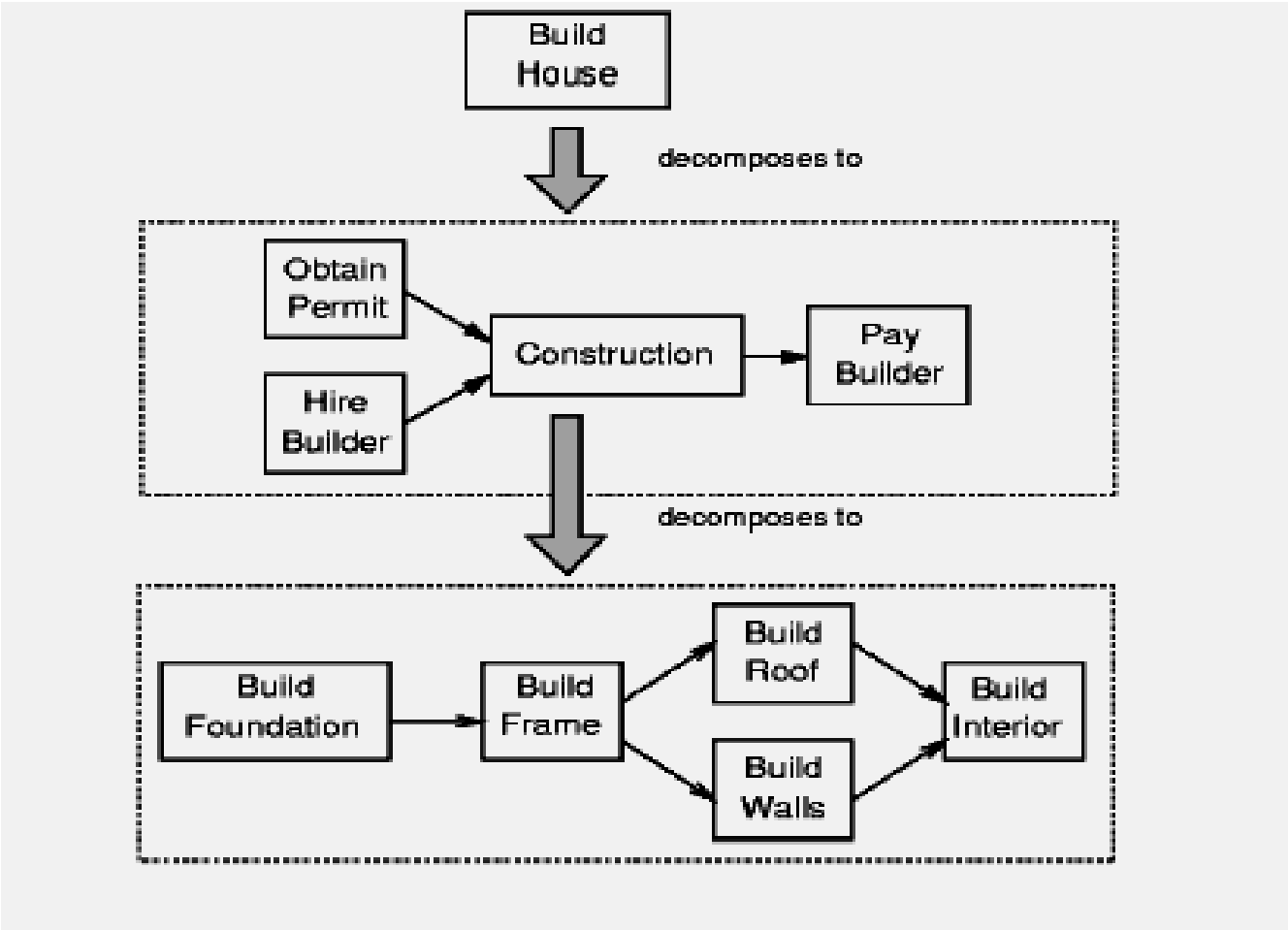
- **lineární**
- **nelineární**



Obr. 13: Příklad lineárního a nelineárního plánu

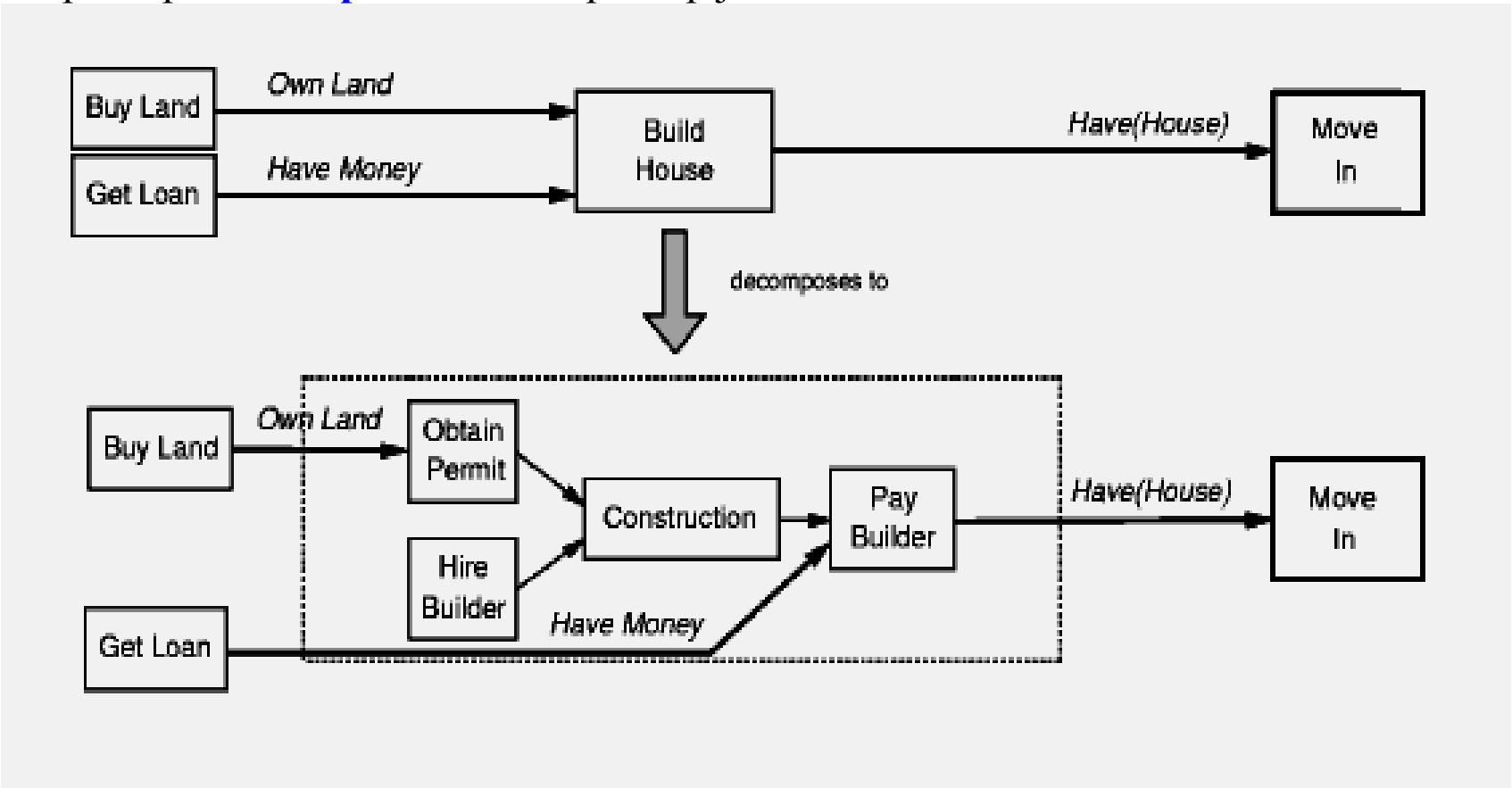
7.3 Hierarchické plánování

Příklad principu hierarchického plánování je uveden na obr 14.



Obr. 14: Příklad hierarchického plánování

Užívá s principu **dekompozice**. Tento princip je znázorněn na Obr.15.

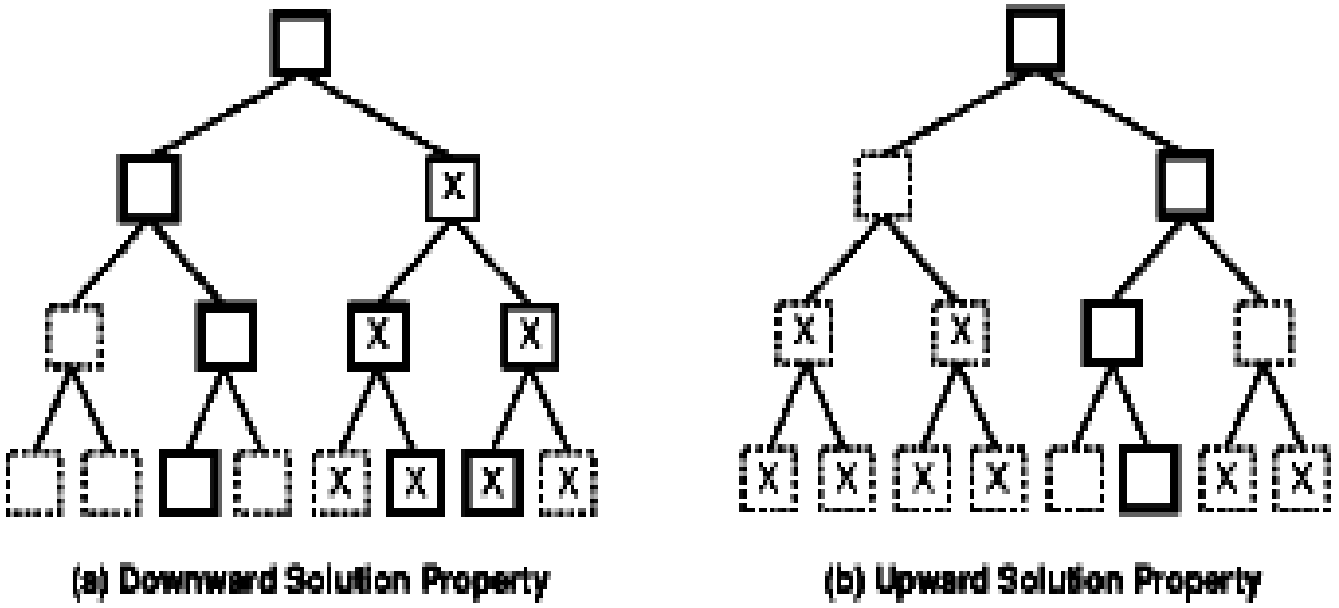


Obr. 15: Příklad dekompozice problému

K vytvořenému abstraktnímu plánu, který je úplný a konzistentní, ale obsahuje abstraktní operátory, lze **nalézt tzv. primitivní plán** (bez abstrakcí). Vytvoří se **binární strom**, ve kterém pak hledáme řešení metodami shora dolů či zdola nahoru.

Ty jsou realizovány algoritmy (viz obr. 16):

- depth – first-search
- length-first-search



Obr. 16: Algoritmy (a) Depth–First-Search a (b) Length-First-Search

7.3 Podmínkové plánování

Pokud **nemá** agent **úplnou a kompletní informaci**, jak se mění okolní prostředí, pak se používá **podmínkové plánování** (vyžaduje akce smyslů). Provádí se tzv. monitorování prostředí.
Plán je zapotřebí opravit – přeplánovat.

7.4 Přeplánování (Replanning)

- **Po kontrole předpokladů a podmínek**
- **Jednoduché přeplánování** – Backtracking
- **Situační plánování** – oprava podle hraničních podmínek či redundantními kroky
- Vytvořením tzv. **Triangular Table** (v případech neočekávaných jevů či u velkých rozměrů oblastí)

7.5 Plánování v multiagentních systémech

Je vytvářen **plán** akcí skupiny (**týmový**) jako výsledek **vyjednávání mezi agenty** a vytváření **dohod** se společnými **závazky**.

Množina plánů je:

- **správný** (correct) – v případě, že všichni kooperující agenti jsou schopni provést úkol v daném čase a za danou cenu
- **akceptovaný** (accepted) – v případě, že všichni agenti dostanou za úkol závazné provedení úlohy v daném čase a s danou cenou

Plán je

- **dosažitelný** (achievable) – existuje-li množina správných plánů
- **naplánovaný** (planned) – existuje-li množina akceptovaných plánů

8. Modelování multiagentních systémů

8.1 Motivace

Vytváření modelů a simulace činnosti modelovaných systémů jsou základním **přístupem k analýze** těchto systémů. Sledovaný systém je prostřednictvím modelu **zkoumán**, jsou zjišťovány jeho **reakce na vstupní podněty**, prováděna **analýza jeho chování**. Simulační modelování se využívá v mnoha technických, biologických, ekonomických, sociálních a jiných systémech.

Současné nástroje reflektují požadavky na modelování složitých systémů a směr vývoje je zaměřen na vytváření **modelů paralelních a otevřených systémů**.

8.2 Model MAS

Pokud se má vytvořit model MAS, je nutné vycházet z **agenta jako prvku** tohoto systému. Agent – prvek **mění vnitřní stav** a **vykonává akce podle** vlastních **rozhodnutí**, která vždy směřují ke **splnění** některého z **cílů**. Modelování multiagentních systémů má proto hodně společného s modelováním otevřených a nedeterministických systémů. Pokud agent vykoná akci v nějakém stavu systému, nemusí být tento systém transformován tak, jak agent předpokládal. To se vztahuje k **omezeným znalostem agenta**, na jejichž základě se agent **rozhoduje**. Agent si vytváří vnitřní model prostředí, avšak tento model je pro něj otevřený, protože neobsahuje všechny prvky, které mají vliv na celkovou činnost multiagentního systému.

Další významnou vlastností MAS je, že **agent může měnit svoji pozici v systému**. Z toho plyne požadavek **změny struktury** v průběhu simulačního běhu.

8.3 Skupiny modelů MAS

- **Fyzický model** (model dostupnosti)

  Relace představuje **dostupnost** (vzájemnou fyzickou) dvou **prvků.** Není definovaná sémantika těchto relací. Během simulace se mění struktura modelu a tím i vzájemná dostupnost prvků.
- **Sémantický model**

  Relace umožňuje postihnout vliv prvku na chování jiného prvku. Jde o **propojení senzorů a efektorů.** Obecně platí, že dva prvky spolu mohou inter-reagovat, pokud jsou ve fyzickém kontaktu a existuje mezi nimi funkční vazba.
- **Sociální model**

  Agenti jsou schopni **vytvářet sociální skupiny** se společnými **normami**. Agent může být součástí jedné nebo více sociálních skupin, nebo také žádné. Relace jsou dány vztahy mezi agentem a skupinou.
- **Kulturní model**

  MAS je definován jako množina agentů s rozdílnými rolemi, pravomocemi a cíli a jejich vztahem vzhledem k těmto rolím. Příkladem je modelování hierarchické struktury rozhodování nebo autoritativní vliv některých agentů vůči jiným.

Sociální a kulturní modely existují jen v mentálních stavech agentů. Tyto modely si vytváří agenti samostatně během své činnosti v MAS a **jakékoliv nastavení vazeb** (sociálních a kulturních) při vytváření modelu nebo během simulace by bylo **nepřípustným zásahem do agentovy autonomie**.

9. Nástroje pro implementaci

9.1 Implementační nástroje založené na agentních principech

Posledním krokem k realizaci BDI agentních systémů jsou implementační nástroje, které mají zabudované **principy tvorby báze znalostí, záměrů, přání a plánování**. Všechny uvedené nástroje jsou vytvořeny **na základě jazyku Java**, což jim umožňuje využívat prostředky pro **síťovou komunikaci** i **objektový přístup**.

9.1.1 JAM

Jazyk JAM nepatří mezi příliš známé agentní implementační nástroje. Má tři části – znalosti, cíle a plán. **Znalosti jsou zapsány ve formě faktů** (jako u jazyku PROLOG) a **cílem** je stav, kterého se snaží agent dosáhnout. Během vykonávání plánu může být vyvolán **podplán**. Definice procedur obsahují povinně cíl, jméno, tělo a nepovinně podmínku spuštění, kontext, efekt po vykonání plánu, efekt při selhání a prioritu. Tělo plánu je tvořeno JAM akcemi (vkládáním, rušením, změnami faktů, podmínky testování stavu prostředí, iteracemi, větvením atd. …).

Dále v něm lze definovat **primitivní akce jako metody** a tím využívat komfortu jazyku Java. Od logického programování se "agentní" programování v jazyku JAM liší tím, že v **případě neúspěchu plánu zůstává prostředí transformované** a **nový cíl** je vybrán a **plán** k jeho splnění je sestaven vzhledem k tomuto novému stavu prostředí.

9.1.2 JADE

JADE je dalším nástrojem pro tvorbu racionálních agentů. Není to přímo jazyk, ale **knihovna tříd pro jazyk Jav**a. Obsahuje prostředky pro implementaci nástrojů a zahrnuje **balíky tříd pro vytváření agentních platforem**, pro komunikaci v ACL jazyku, pro definici agentova životního cyklu, pro adresáře služeb, atd. JADE je vhodným nástrojem pro implementaci **softwarových mobilních agentů** a velice populárním prostředkem pro jejich realizaci.

9.1.3 ZEUS

ZEUS je nástroj pro vytváření **multiagentních systémů**. Zahrnuje možnosti **reprezentace znalostí plánování, komunikace a sociálních vazeb**. Jeho silnou stránkou je **vizualizace návrhu**.

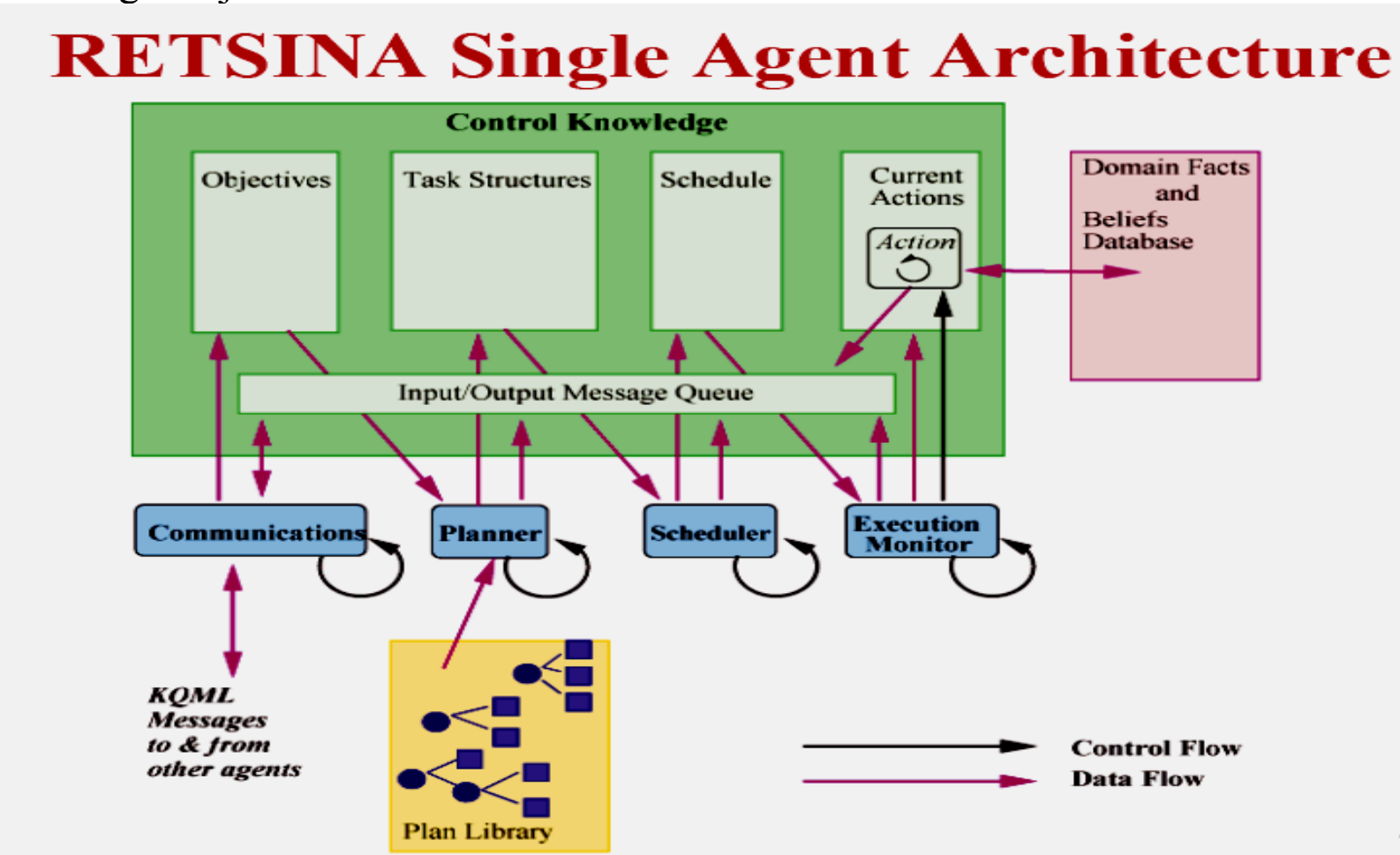
MAS je vytvářen **hierarchicky** ve třech vrstvách – **definiční, sociální a organizační**. Na definiční úrovni se **definují agenti**, jejich znalosti, cíle, prostředky a schopnosti. V sociální vrstvě se **určuje postavení** agentů, které zahrnuje znalosti i o jiných agentech v této skupině a informace o jejich dostupnosti. Organizační vrstva **určuje způsob komunikace** a vyjednávání, strategie rozdělování úloh, apod.

9.2 RETSINA

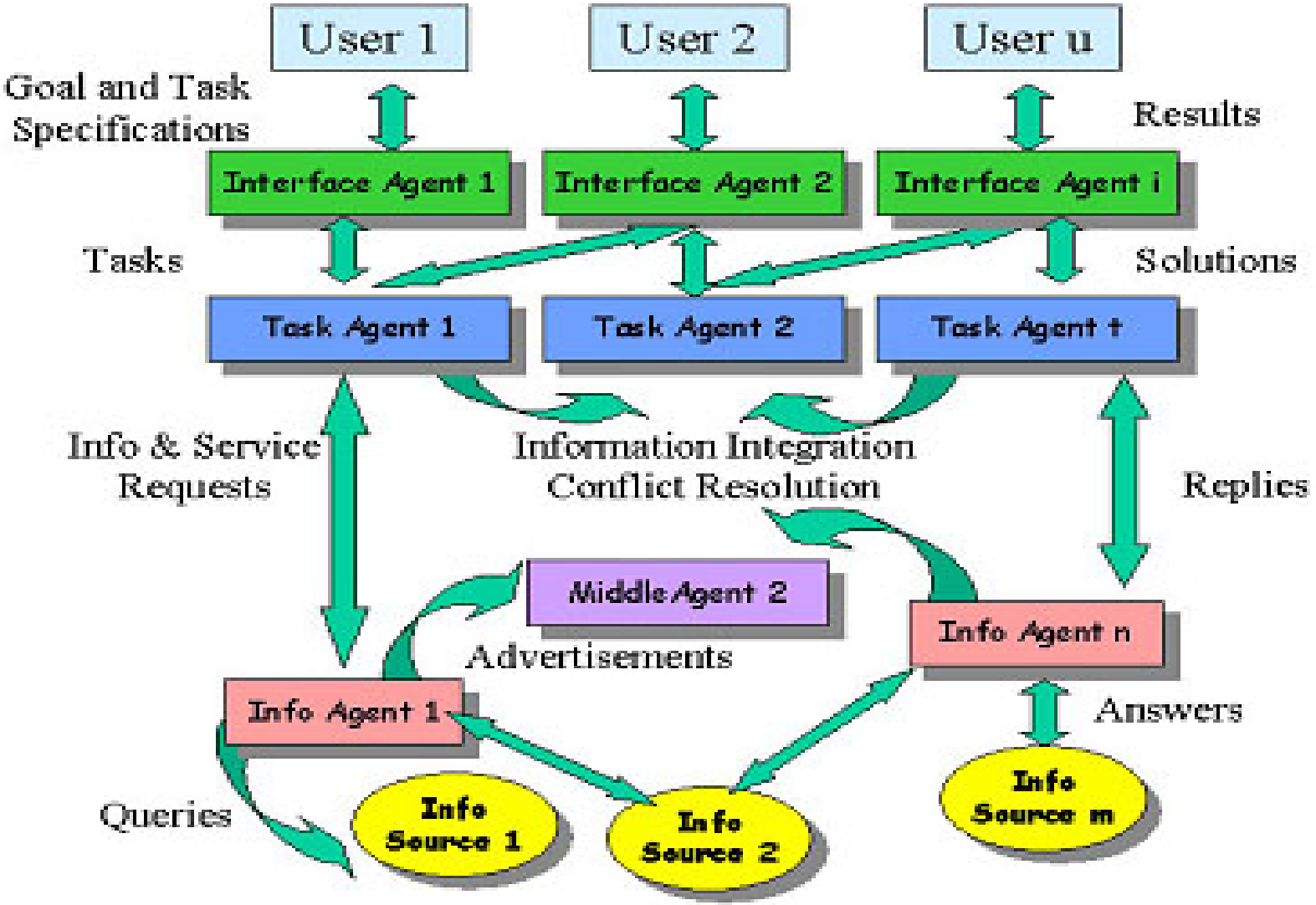
RETSINA (Reusable Environment for Task-Structured Intelligent Networked Agents) je otevřený multiagentní systém (viz obr.18), který podporuje skupiny heterogenních agentů.

Je založen na předpokladu, že agenti v systému mohou **vytvářet skupiny**, které vstupují do **peer -to-peer interakcí**. Koordinace struktur ve skupině agentů může vycházet ze vztahů mezi agenty spíše než jako výsledek vynuceného nátlaku samotné infrastruktury.

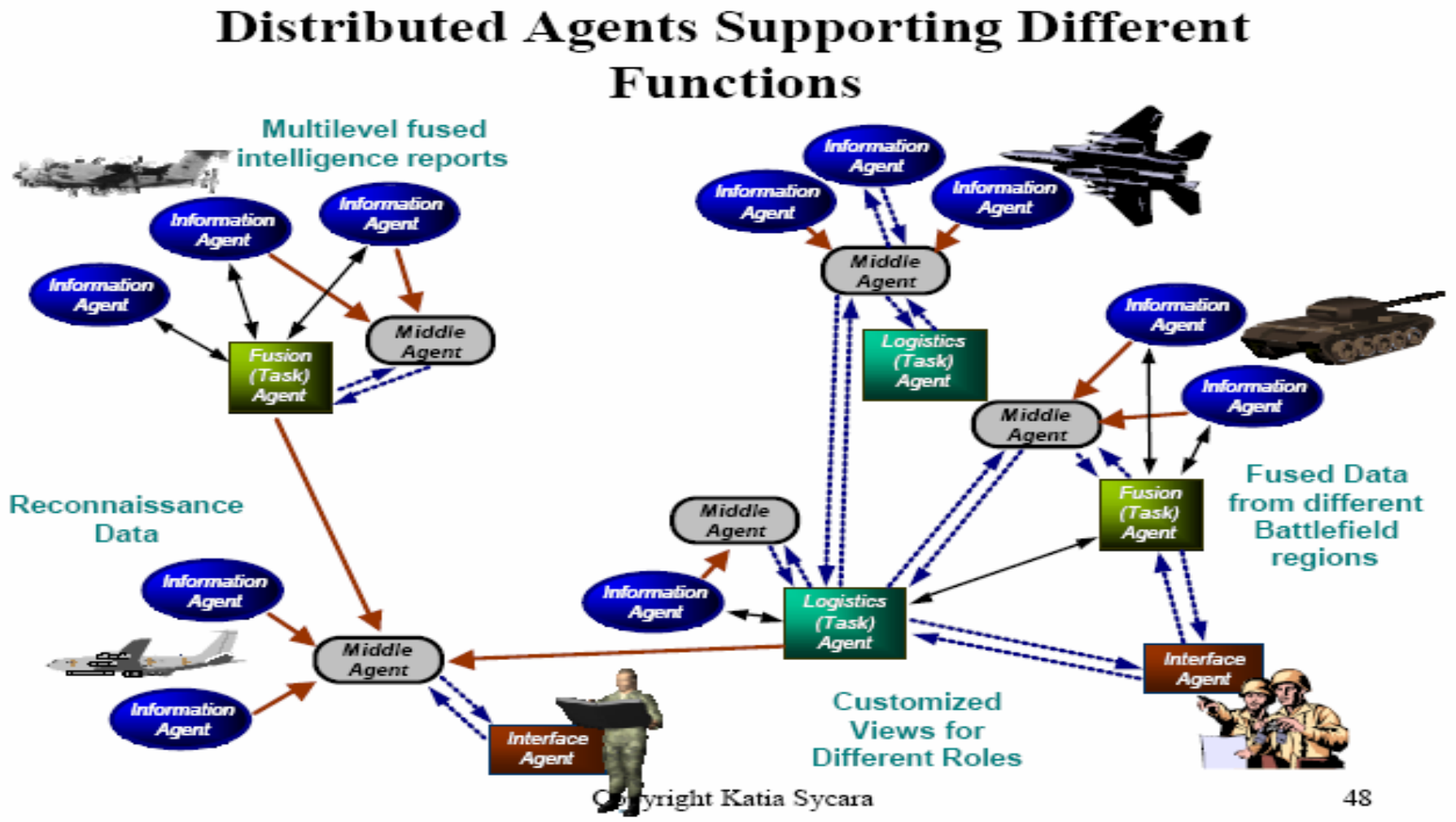
V souladu s tímto předpokladem, RETSINA nevytváří centralizované řízení bez MAS, ale raději implementuje distribuované služby infrastruktury, které usnadní interakce mezi agenty. Architektura agenta je znázorněna schématem na obr.17.



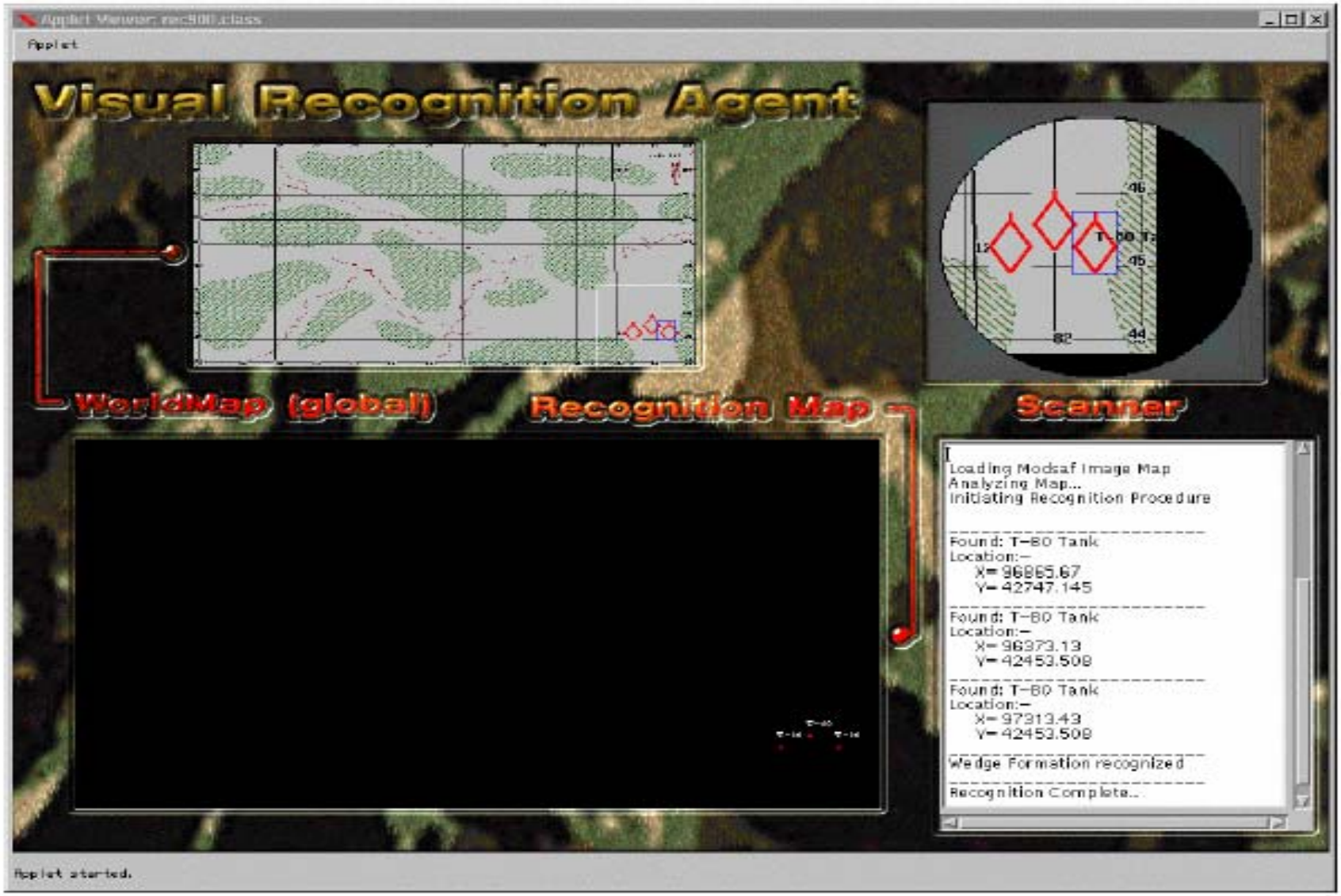
Obr 17. RETSINA architektura agenta



Obr. 18: Grafická reprezentace RETSINA MAS

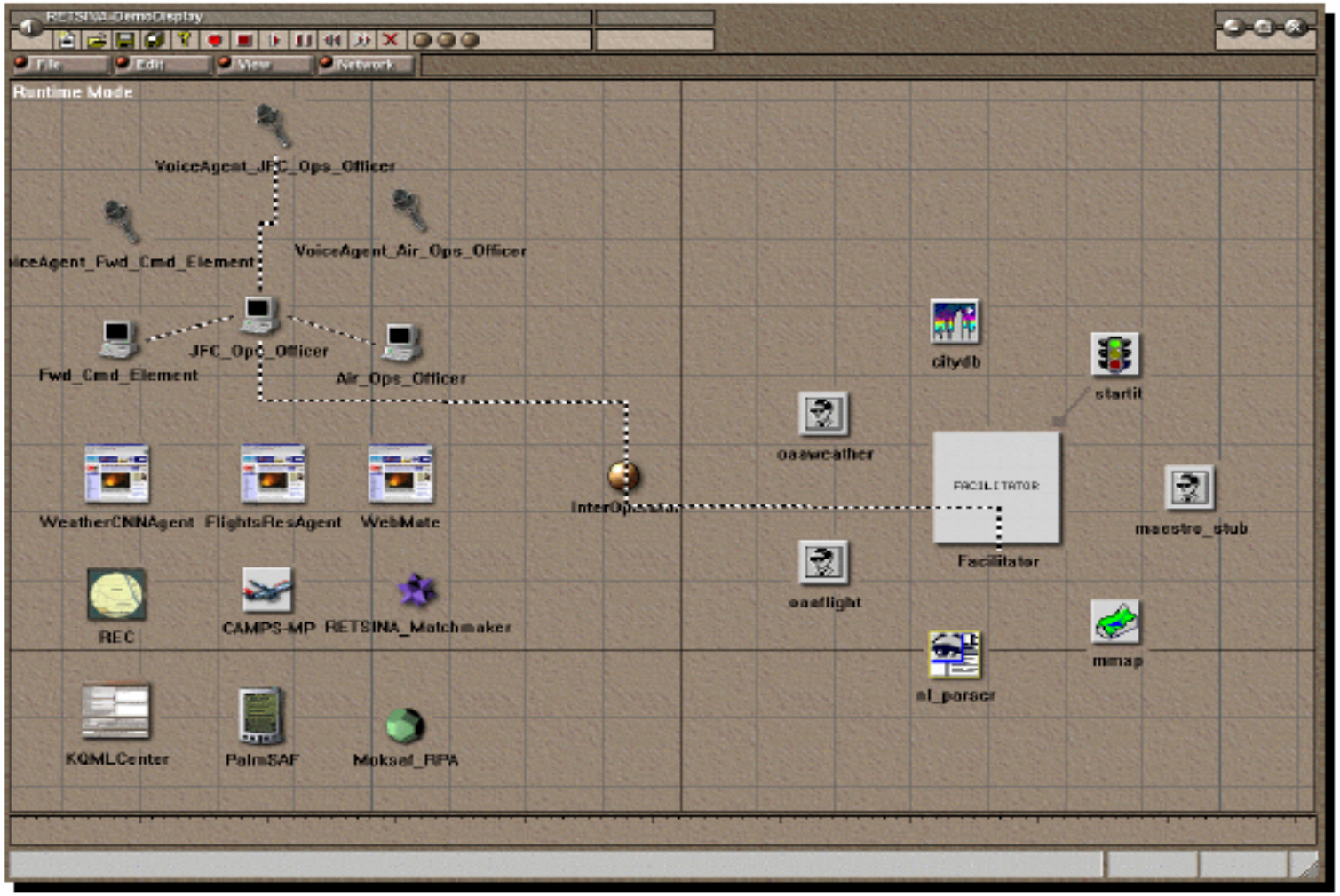


Obr. 19: Příklad různých funkcí podpory distribuovaných agentů



Obr. 20: Příklad využití agenta pro vizualizaci rozpoznávání

RETSINA Agent Visualization Tool



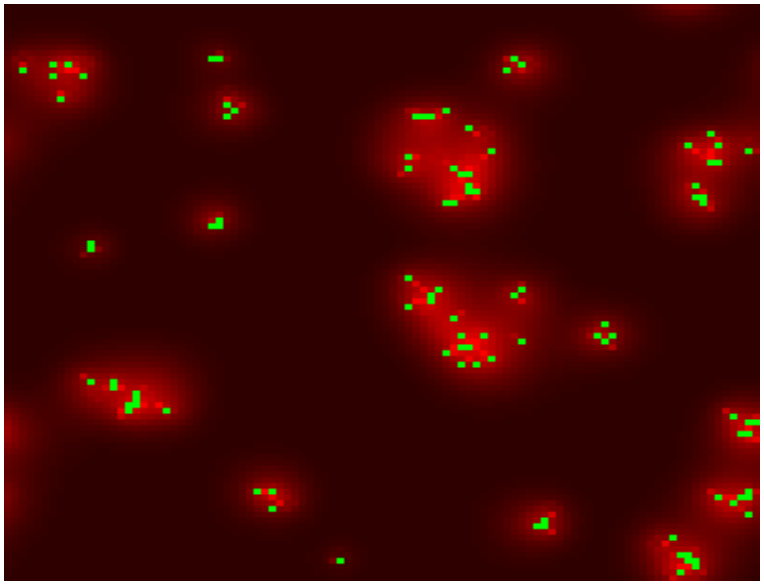
Obr. 21: Ukázka nástroje pro vizualizaci

9.3 Swarm

Pro simulaci multiagentního systému byl vytvořen software pracovníky **Institutu v Santa Fe** (Santa Fe Institute) s názvem ***Swarm*** – pojem „inteligence davu“ (Swarm Intelligence).
Je to nástroj pro výzkumné pracovníky, použitelný v různých oborech. Základní architekturou je simulace skupiny vzájemně působících interaktivních agentů. Tímto experimentálním softwarem lze implementovat rozsáhlou oblast modelů založených na agentech.

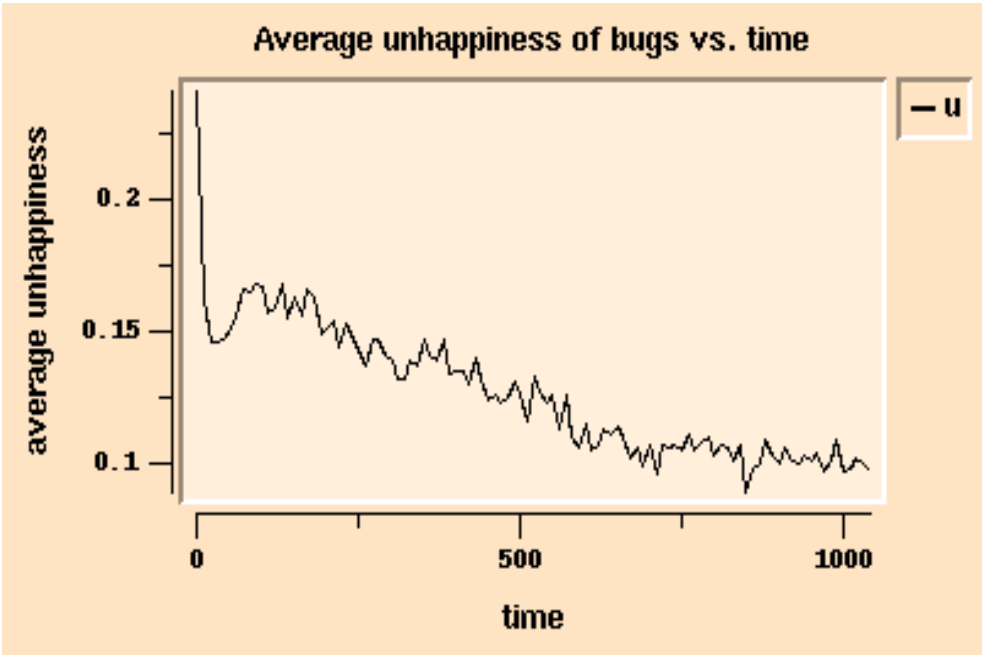
9.3.1 Příklad Heatbugs (heat=teplo, bug=hmyz, nepřeložitelné- uměle vytvořené slovo)

Heatbugs je jeden z ilustračních příkladů, globálního chování agentů, reagujících pouze na lokální informaci.



Obr.22: Svět heatbugů

Každý agent v tomto modelu je heatbug (bug = hmyz, lightning bug=světluška, amer.). Svět má prostorové vlastnosti, teplo, které se šíří a mizí s časem. Na tomto obrázku (obr.22) reprezentují zelené skvrny heatbugy, jasně červené pak teplejší místa ve světě.
Každý heatbug produkuje malé množství tepla, a ke své existenci má též určitou ideální teplotu. Systém sám je jednoduchý v čase krokovaný model, v každém časovém kroku se heatbug snaží přiblížit k sousedovi, aby byl "šťastnější" a vyprodukoval trochu tepla. Heatbug sám o sobě nemůže udržovat dostatečnou teplotu, takže po čase má tendenci se v zájmu udržení teploty a "štěstí" sdružovat.

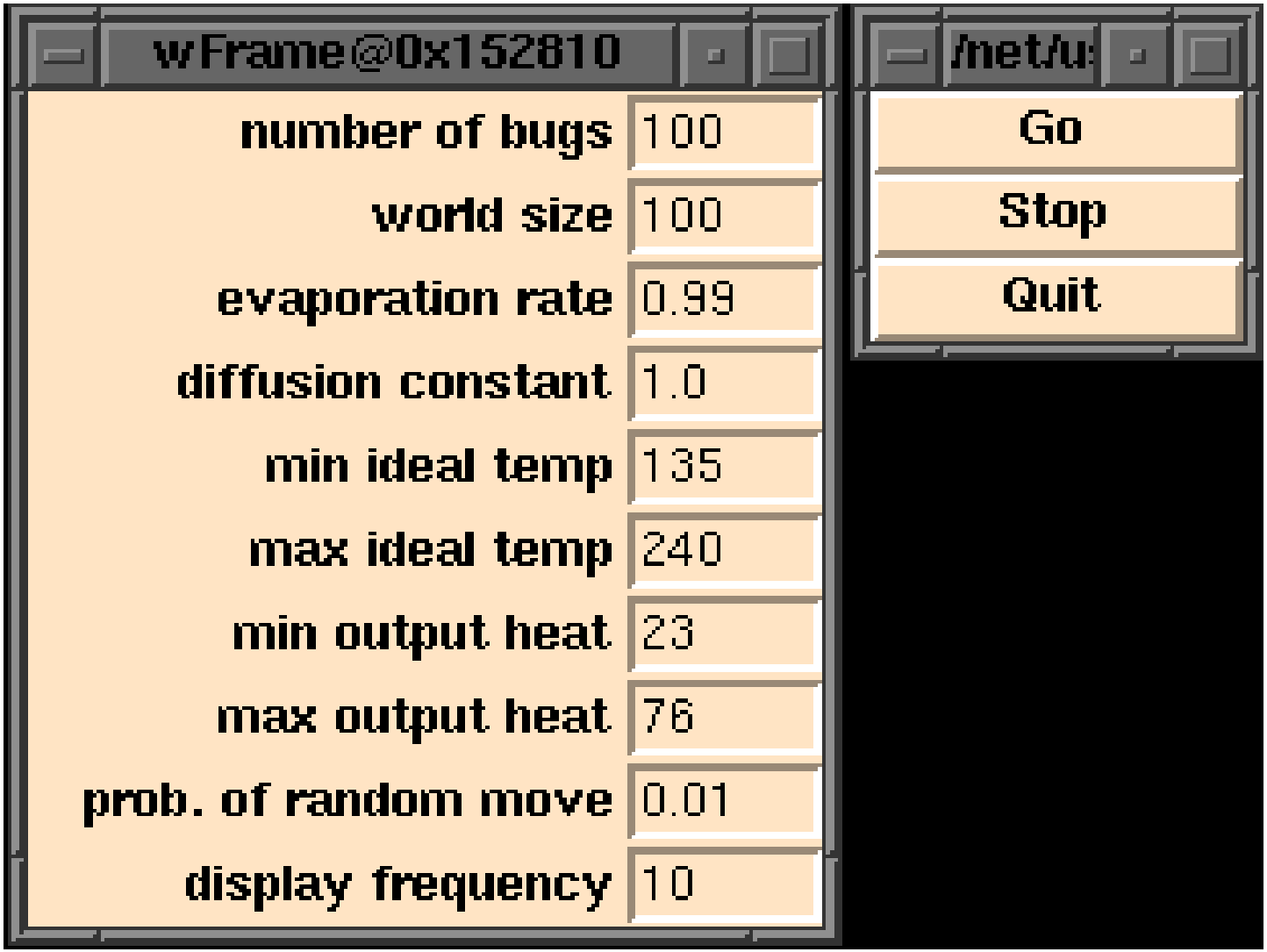


Obr.23: Graf s daty získanými ze systému

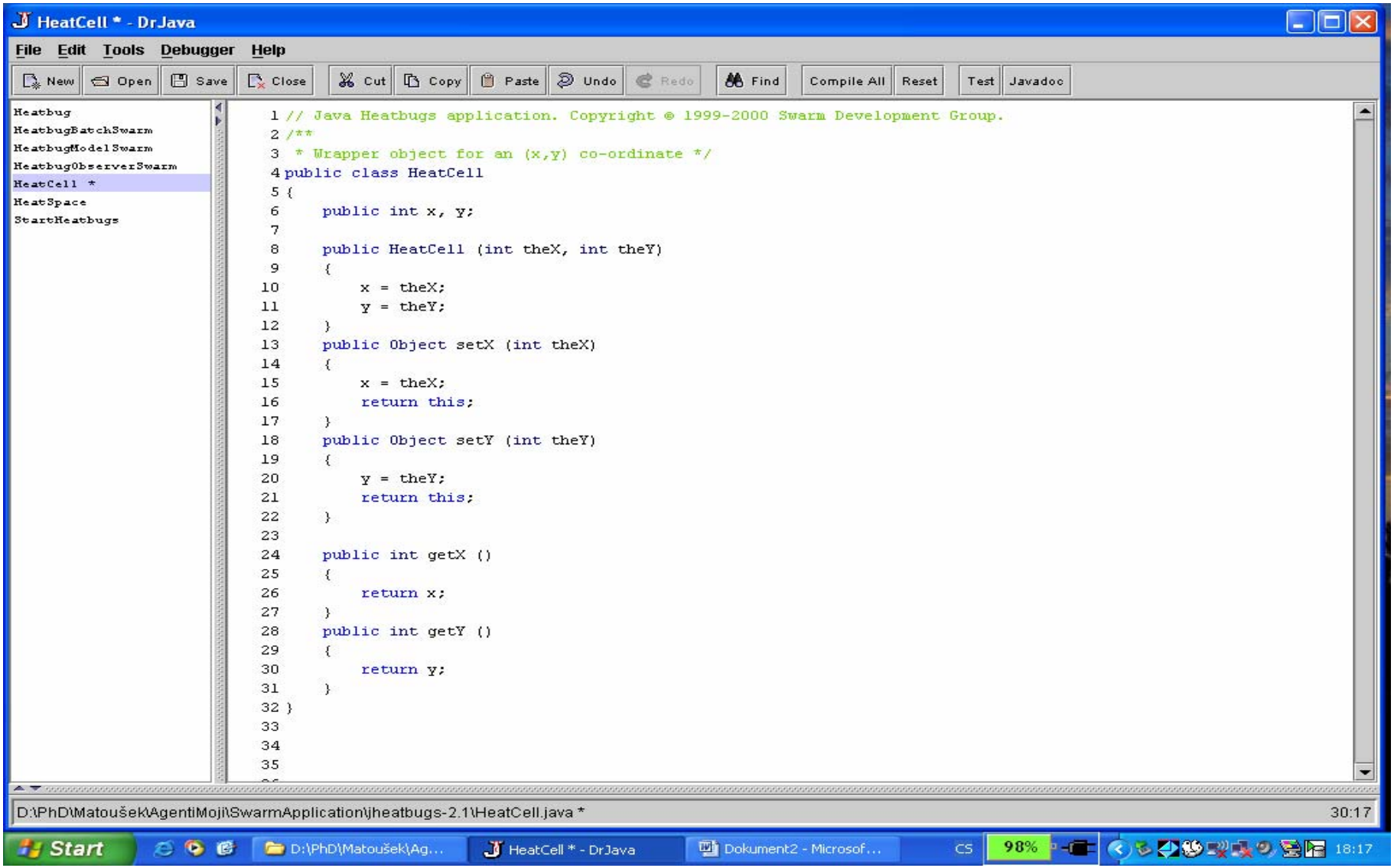
O tomto systému lze uvažovat jako o optimalizační úloze, každý heatbug se snaží minimalizovat svoje “neštěstí”. Graf (obr. 23) ukazuje jak „dobře“ (v čase) je průměr neštěstí všech bugů optimalizován. V tomto systému, přestože každý bug jedná samostatně, dochází v průměru ke zmenšování “neštěstí”.

Tento graf reprezentuje jenom malou část souboru dat, graf s jednoduchými časovými řadami. Swarm systém podporuje paletu generických metod pro získání dat z komponent systému, spojením dat prostřednictví statistických filtrů a pak zobrazení generickými objekty vizualiazace a úschovu do souborů. Velká část této podpory je stále ještě ve vývoji a tak zde není prezentována.

Jedním z důvodů vytváření simulací je, že chceme **poopravit několik parametrů** a **sledovat**, k jakým **změnám** dochází. Obrázek uvedený níže (obr.24) je snímek některých řídicích charakteristik: formulář pro uživatelský vstup parametrů a řídicí panel zahájení a zastavení simulace. Modely Swarm mohou být také konfigurovány a řízeny skriptů jazyku Lisp.



Obr. 24: Řízení systému

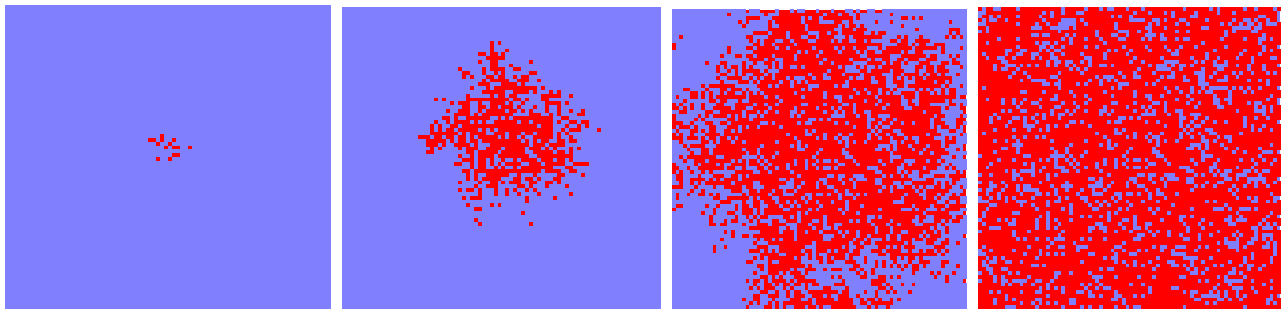


Obr.25: Java kód obalu objektu pro heatbug

9.3.2 Příklad - **Mousetrap** (past na myši)

Úloha je starou školní ukázkou řetězové reakce (jaderné štěpení). Na podlaze jsou rozloženy v pravidelné mřížce pasti na myši a každá past je zatížena dvěma pimpongovými míčky. Jeden míček spustíme na střed a sledujeme, co se bude dít.

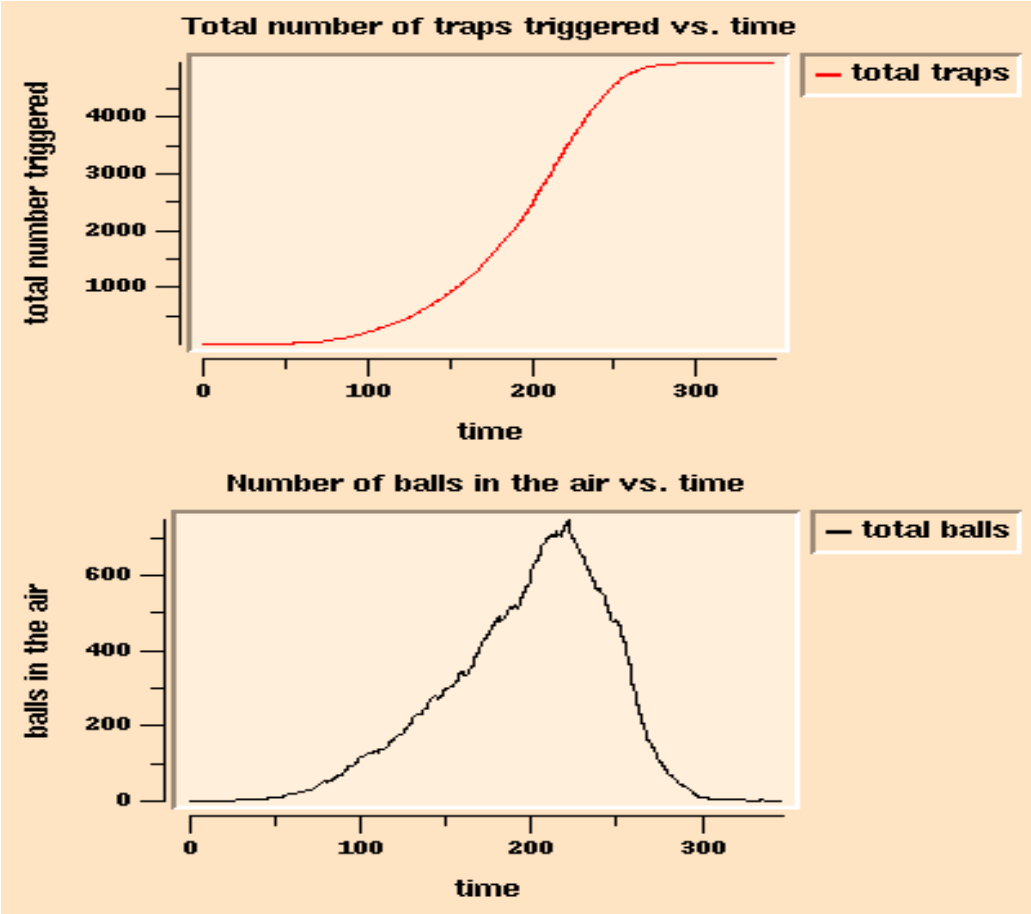
Tato úloha je implementována jako příklad jednoduché diskrétní událostní simulace. Časové krokování každé pasti může být neuvěřitelně neefektivní, pokud velká část pastí po většinu času „nic nedělá“. Místo toho se simuluje každý pimpongový míček jako událost v časovém plánu. Tato metoda je mnohem účinnější: výpočet proběhne pouze, když se něco vykoná. Síla Swarmu je v tom, že podporuje diskrétní události a časově krokované modely.



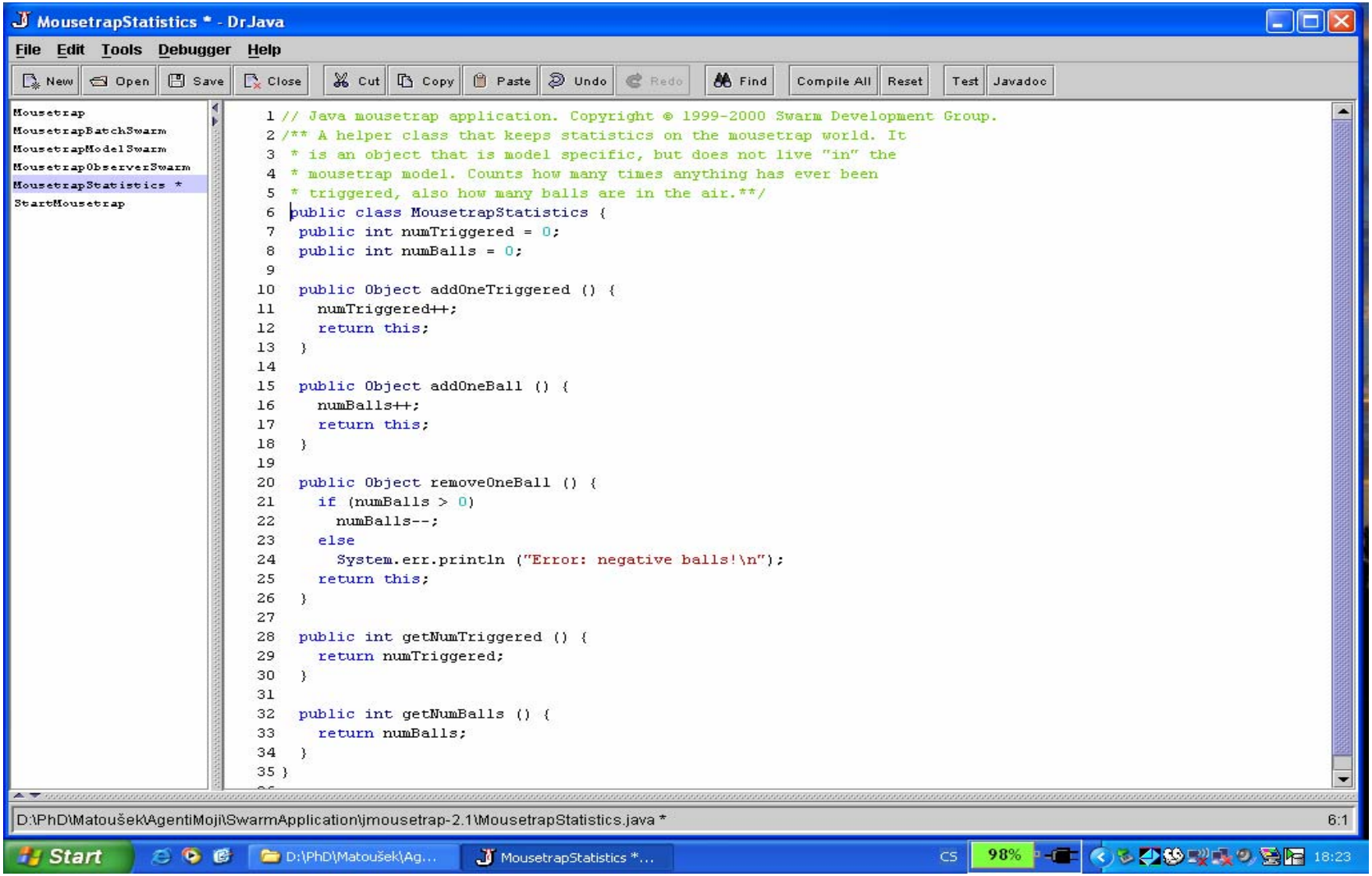
Obr. 26: Světová evoluce v čase.

Na obr. 26 jsou časové snímky (od počátku do konce reakce). Když past sklapne, barva pasti se změní z modré na červenou. V čase sklapnutí pasti se plánují dvě události sklapnutí pro dvě náhodné blízké pasti někdy v blízké budoucnosti. Míčky letí pouze do omezené vzdálenosti, a tak se reakce rozrůstá.

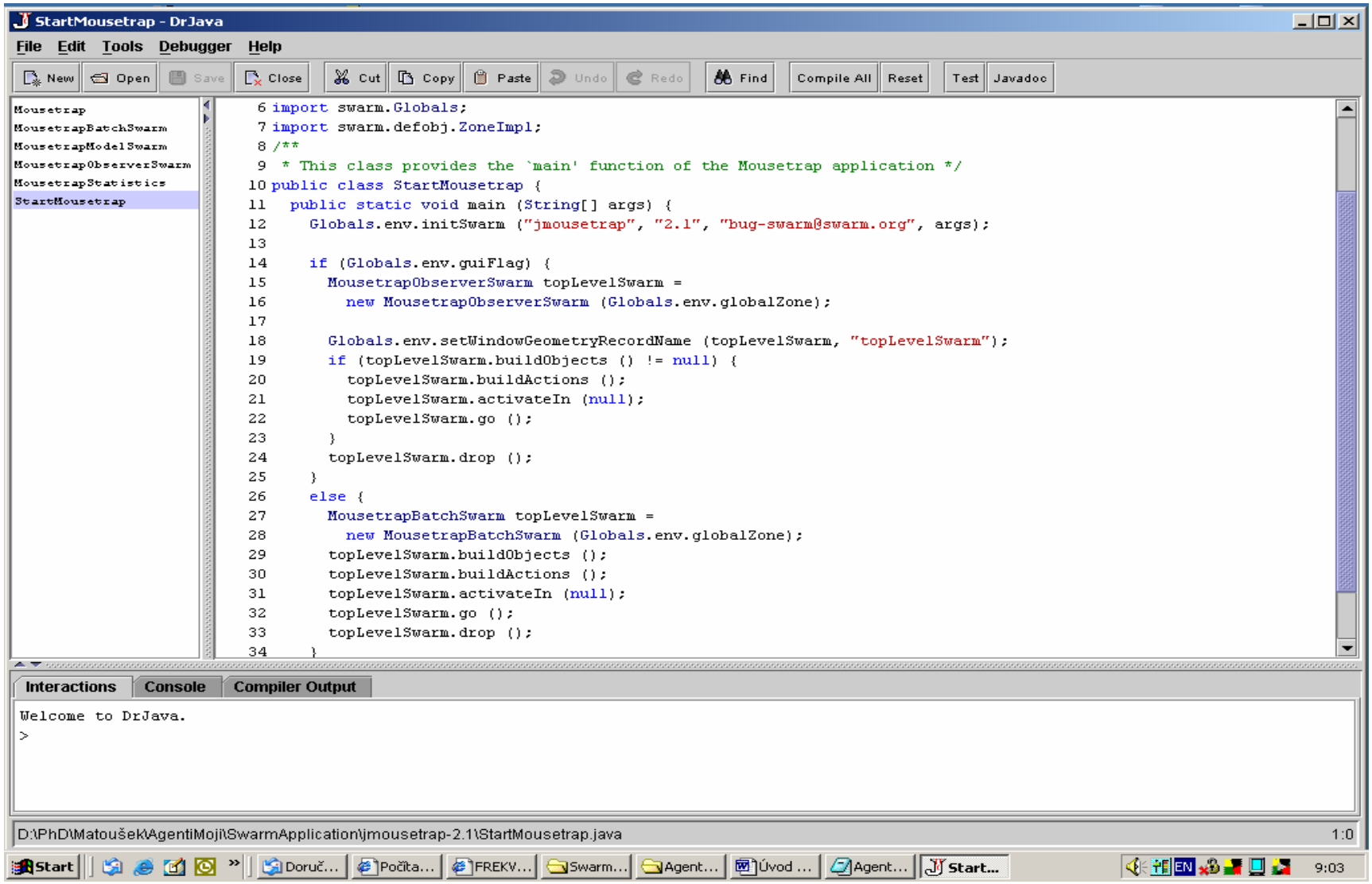
Tyto dva grafy (obr. 27) ukazují detail reakce. Horní graf, ukazuje počet pastí sklapnutých v čase, poskytuje klasický výsledek: reakce se na počátku rapidně rozrůstá, ale klesá a zastaví se, když dochází palivo (rozložené pasti). Spodní graf zachycuje počet událostí neukončených v časovém rozvrhu, to koresponduje s počtem pimpongových míčků právě ve vzduchu.



Obr. 27: data modelu



Obr. 28: Kód pomocné třídy statistiky pro úlohu Mousetrap (Java)

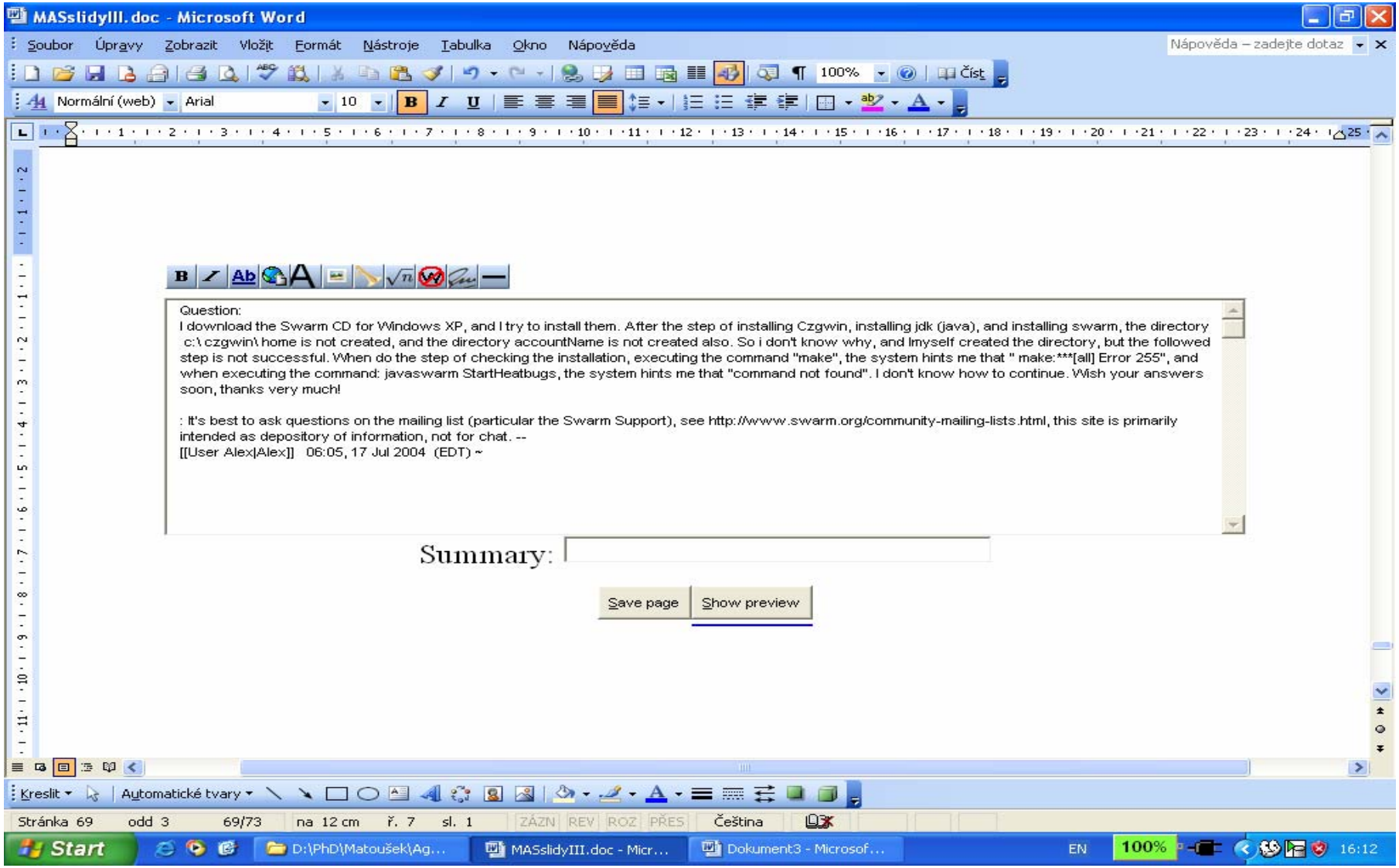


Obr. 29: Třída s metodou main pro úlohu Mousetrap

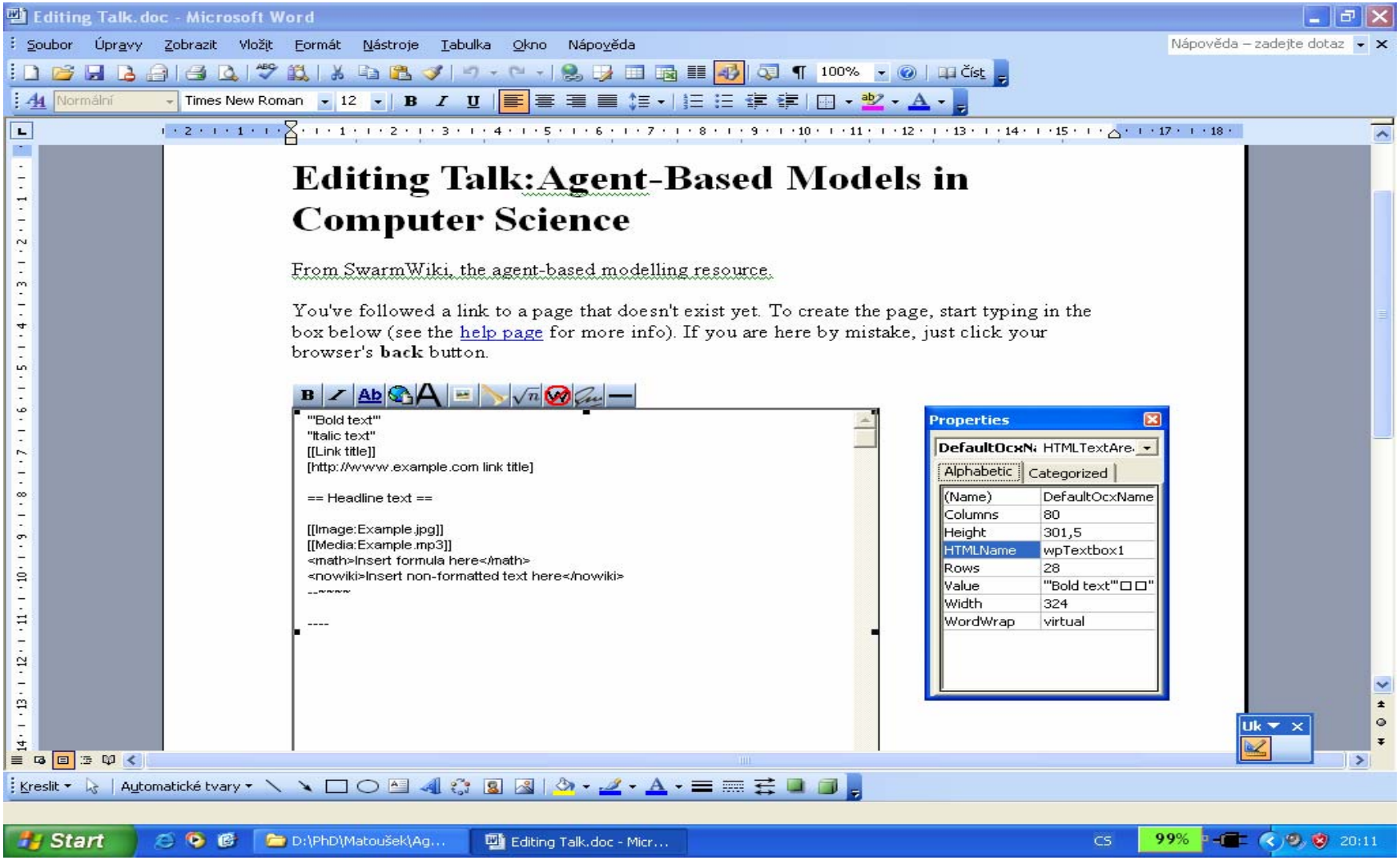
9.3.2 Příklad - Editing Talk: Agentově orientované modely v počítačových vědách

SwarmWiki – příklad agentově orientovaného modelování editace zdrojových souborů.

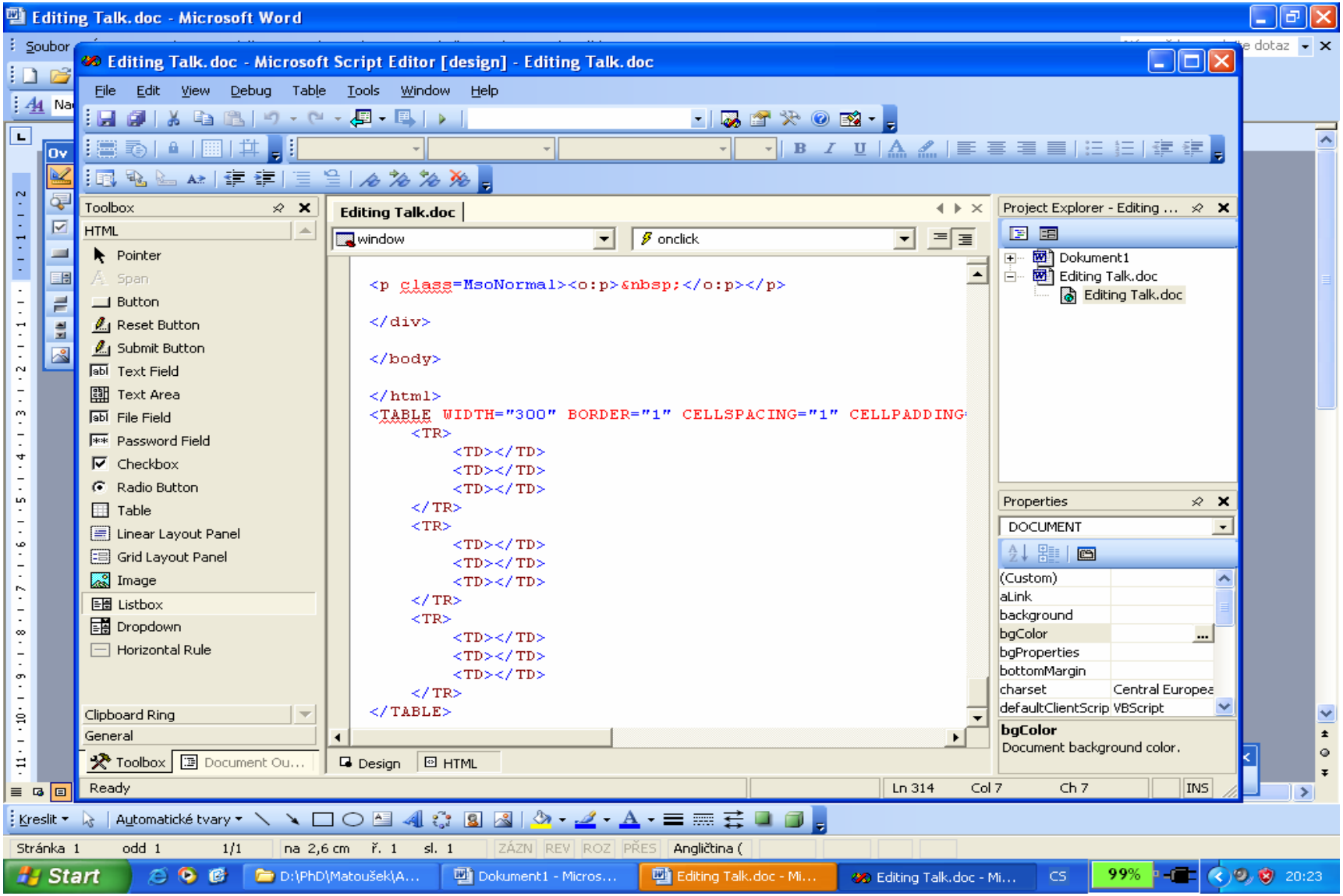
Následující odkaz je na stránku, která zatím neexistuje.
K vytvoření stránky je třeba provést zápis do editačního boxu (obr. 30)
- viz více informace na: help page.



Obr. 30: Editační box



Obr. 31: AOP – editační box a menu Vlastnosti



Obr. 32: Editing Talk s Microsoft Skript Editorem

9.4 Některé další nástroje

Dalším prostředkem je *Stella* – program pro dynamické systémy, který dovoluje vytváření modelů se statickou strukturou. Tento nástroj vyhovuje pro modelování vztahů mezi agenty, ale je nevyhovující pro tvorbu a implementaci modelů s dynamickými vztahy.
Squeak a ostatní dialekty Smalltalku je vhodné pro rychlé vytvoření prototypu a pohotové implementaci vzájemných vztahů.

Další informace:
Prof. Ing. Václav Matoušek, CSc.:
Předmět: PT
Přednáška: Inteligentní počítačové prostředky - Inteligentní agenti

http:// www.kiv.zcu.cz/~Matousek/uir

10. Zdroje

[1] *Zbořil F.*: Plánování a komunikace v multiagentních systémech, Disertační práce, FIT VUT Brno, 2004

[2] *Mařík V.*: MAS-řešení složitých a rozsáhlých úloh, přednášky Umělá inteligence, FEL ČVUT, Praha, 2002

[3] *Pěchouček M.*: Planning in Multi-Agent Systems, Agent Technology Group, Gerstner Laboratory, FEL ČVUT, Praha, 2004

[4] *RETSINA* - The Robotics Institute Carnegie-Mellon University*:* http://www.cs.cmu.edu/~softagents

[5] *Sycara K.*: Multiagent Infrastructure for Agent Interoperatibility in Open Computational Environments , School of Computer Science, Carnegie Mellon University, Pittsburgh, 2004

[5] *Swarm* - www stránky: http://www.swarm.org/

[6] *Urbánek Š.*: Modelling and Simulation of Trust Evolution in Complex System, Master Thesis, STU Bratislava, 2004

[7] *Hodík J.*: Multi-agentní systém jako model tržního hospodářství, Diplomová práce, ČVUT Praha, 2001