

Návrh softwarových systémů - mobilita

Jiří Šebek *Návrh softwarových systémů*
(B6B36NSS)



```
public final void onSensorChanged(SensorEvent event)
{
    m_flightIntensity = event.values[0];
    m_etAmblight.setText("" + m_flightIntensity + " lx");
}

... resume()
... light, ... (NORMAL);
```

Co to je mobilita a jak se projevuje v návrhu softwaru?



Mobilita

- Jedna z vlastností systému/ podsystemu/ algoritmu
- Založena na migrování výpočtu z jednoho uzlu sítě na další
 - Jak ? → podle typu
- Důležitá u distribuovaných výpočtů (dost zásadní)
- Využití « agentů »



Agenti

- Program, který se chová jako jiný uživatel
- Snaží se splnit « svůj » cíl
- **Nepotřebuje dohled**, reagují na prostředí
- Někdy označované jako «bots »
- Využití pro umělou inteligenci



Typy agentů

- Autonomní agenti
- Multiagentní systémy
- Distribuovaní agenti
- Mobilní agenti



Typy agentů

- Autonomní agenti
 - AI - modifikace chování pro dosažení optima
- Multiagentní systémy
 - AI - kooperace jinak autonomních agentů
- Distribuovaní agenti
 - spolupráce komponent výpočtu v distribuovaném prostředí
- Mobilní agenti
 - distribuovaní agenti s možností migrace

Co umí ?

- Provádět cílené vyhledávání na internetu.
- Zkontrolovat a upřednostnit příchozí e-maily.
- Otestovat nové počítačové hry.
- Vyplňovat e-formuláře.
- Provádět online hledání práce.
- Synchronizovat profilů sociálních sítí.
- Sestavit přizpůsobené zprávy.
- Najít dobré nabídky v e-commerce.
- ...

Mobilní agenti

- mobilní programová komponenta
 - specifické výpočetní paradigma
- autonomní rozhodování o migraci
 - mobilní agent přeruší výpočet a odmigruje na jiný počítač
- omezená komunikace se zdrojem
 - agent doručí výsledek výpočtu migrací

Typy mobility

- Mobilní kód
 - migrace kódu
- Mobilní objekt
 - migrace kódu a dat
- Mobilní agent
 - migrace kódu, dat a stavu

Typy mobility (jiný pohled)

- Slabá mobilita - Weak Mobility
 - migrace kódu a dat
- Silná mobilita - Strong Mobility
 - migrace kódu, dat a stavu výpočtu

Mobilní agenti

- Výhody
 - redukce datových toků
 - lokální komunikace agenta se serverem
 - autonomní práce
 - vhodné pro mobilní zařízení s omezenými možnostmi
 - přesun náročnějšího výpočtu na výkonné prvky
 - možnost práce i při odpojení klienta
- Problémy
 - bezpečnost
 - uzavřenost - typicky u skriptových řešení

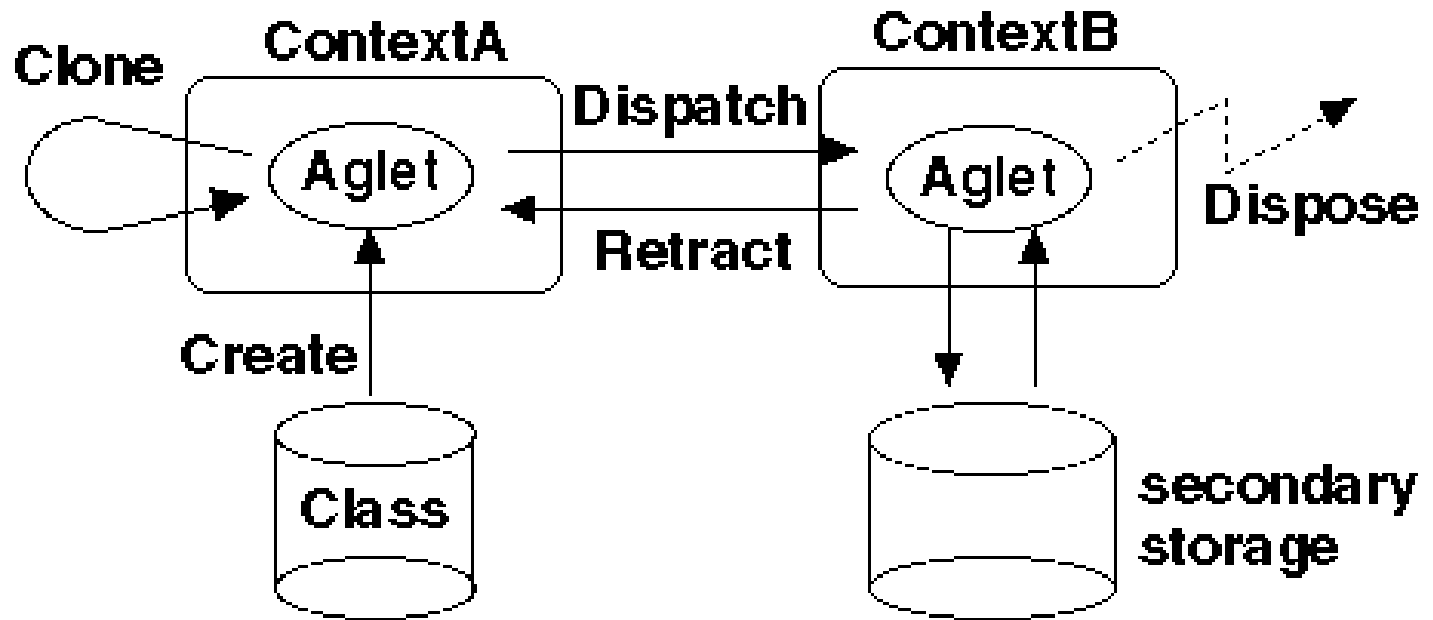
Technologie pro práci s agetama

- Telescript - General Magic - speciální jazyky
- Agent Tcl / D'Agents - Dartmouth College
- Voyager - Object Space - JVM
- Odyssey - General Magic - JVM
- Aglets - IBM - JVM

Aglet example

- Aglet = Agent+applet
- Aglet je mobilní Java objekt migrující mezi servery Aglets
- Výpočet omezený kontextem – Context
- Více kontextů na serveru
- Aglet = Aglet state (variable values) + Aglet code (Java class)

Aglet example

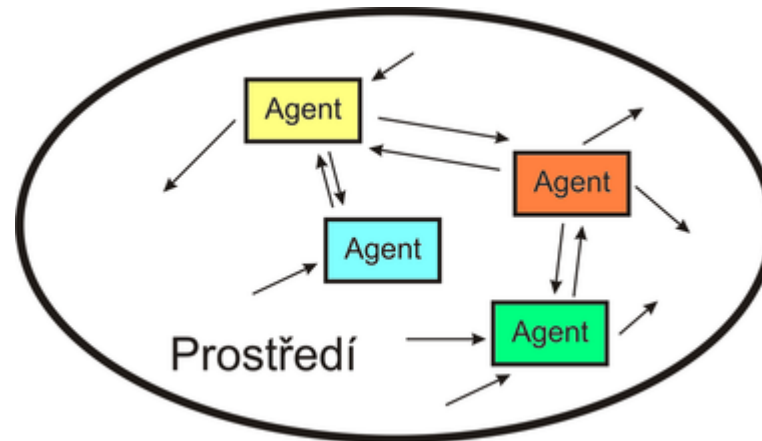


Autonomní agenti

- Autonomní agent může představovat jak fyzický (zvíře, stroj) tak virtuální (software)
- Autonomní agent se pohybuje v určitém prostředí
- Na základě podnětů z prostředí (kontext) se samostatně rozhoduje

Multiagentní systémy (MAS)

- Každý agent má nekompletní informaci nebo schopnosti pro řešení problému a disponuje jen limitovaným úhlem pohledu
- Neexistuje globální řízení systému
- Data jsou decentralizovaná
- Výpočty jsou prováděny asynchronně



Distribučování agentů

- Spolupráce komponent výpočtu v distribuovaném prostředí
- Požadavky na distribuované agenty:
 - Bezpečnost (Safety) = Nothing bad happened yet. Přechody mezi jednotlivými stavy systému způsobené algoritmem nesmí způsobit nežádoucí stav (zhroucení systému).
 - Živost (Liveness) = Something good eventually happens. Posloupností stavových přechodů je postupně dosahováno žádoucího stavu.
 - Férovost = Musí se dostat na každého, resp. každý uzel v konkrétní prioritní třídě (pokud je definována) musí být obslužen férově.

Distribučování agentů

- Důležitou vlastností je zachování jeho funkčnosti při výpadku jednoho nebo několika uzlů – zdroje, resp. úkoly jsou přerozděleny mezi ostatní uzly systému. Výpadek může zhoršit chování systému. Aby se zachovala funkčnost, musí být distribuována i data v několika kopiích
- Struktura systému (topologie, počet uzlů nebo latence) nemusí (a často není) známa předem (tedy před jeho vytvořením). Tato struktura se také může měnit v čase běhu programu (například počet uzlů se během času často mění)

Děkuji za pozornost

Jiří Šebek

