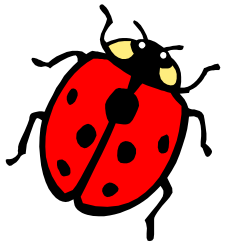
Architektúra Agent-Space

Andrej Lúčny

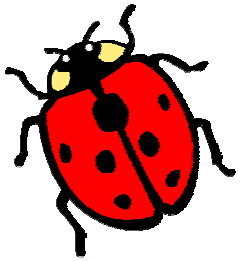
MicroStep-MIS & KAI FMFI UK Bratislava, SLOVAKIA

andy@microstep-mis.com

www.microstep-mis.com/~andy



Agent



- “an agent is a computational process than can collect information about its environment, can decide what actions to perform perhaps by reference to explicit goals, and can then act upon its environment” [Doran 1992]

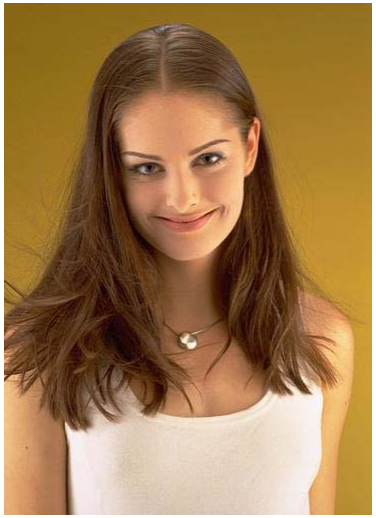
1. Vníma prostredie
2. Volí akcie
3. Vykonáva akcie
4. Jeho činnosti je možné pripísať cieľ
5. Robí to neustále (opakovane)

Klasifikácia agentov podľa voľby akcií

Reaktívna voľba

Keď chodievam na skúšku, obliekam si minisukňu

Idem na skúšku



Oblečiem si minisukňu

Deliberatívna voľba

Skúšku chcem urobiť, ale veľa toho neviem. Keď skúšajúci nebude dávať pozor, mám šancu. Je to muž a ja mám pekné nohy, preto si dám minisukňu

MÓDA

implicitný cieľ

ROZUM

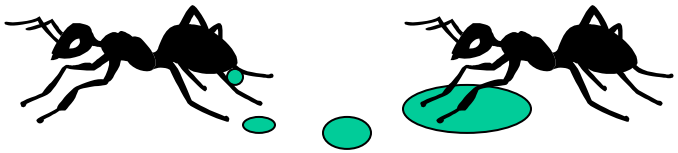
explicitný cieľ

Komunikácia medzi agentami

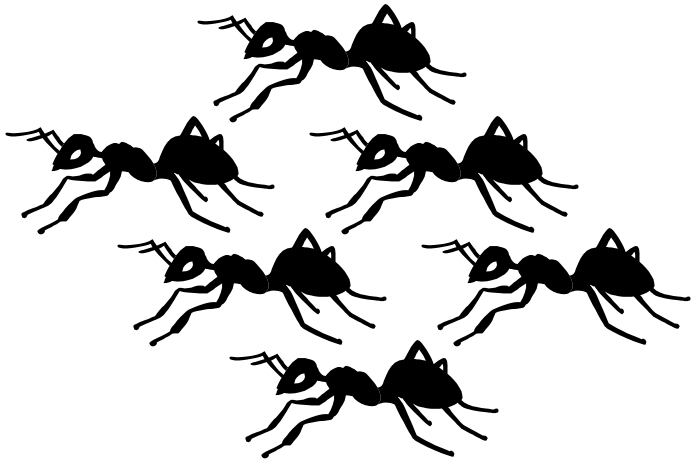
Komunikácia

priama - adresná

nepriama - cez prostredie



Multiagentové systémy (MAS)



- distribuované architektúry
- čo vložiť do každého mravčeka, aby dokopy vykonávali určitú činnosť ? (napr. RoboSoccer)

Vrstvové architektúry

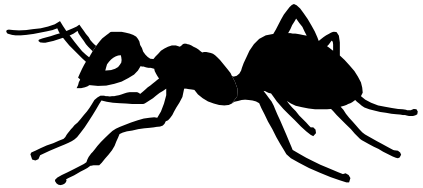
- moduly

reaktívne

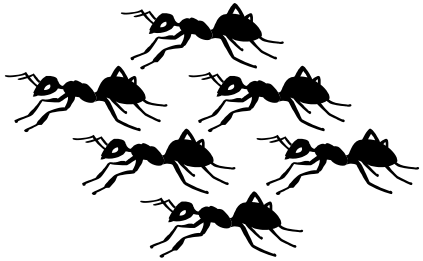
deliberatívne

hybridné

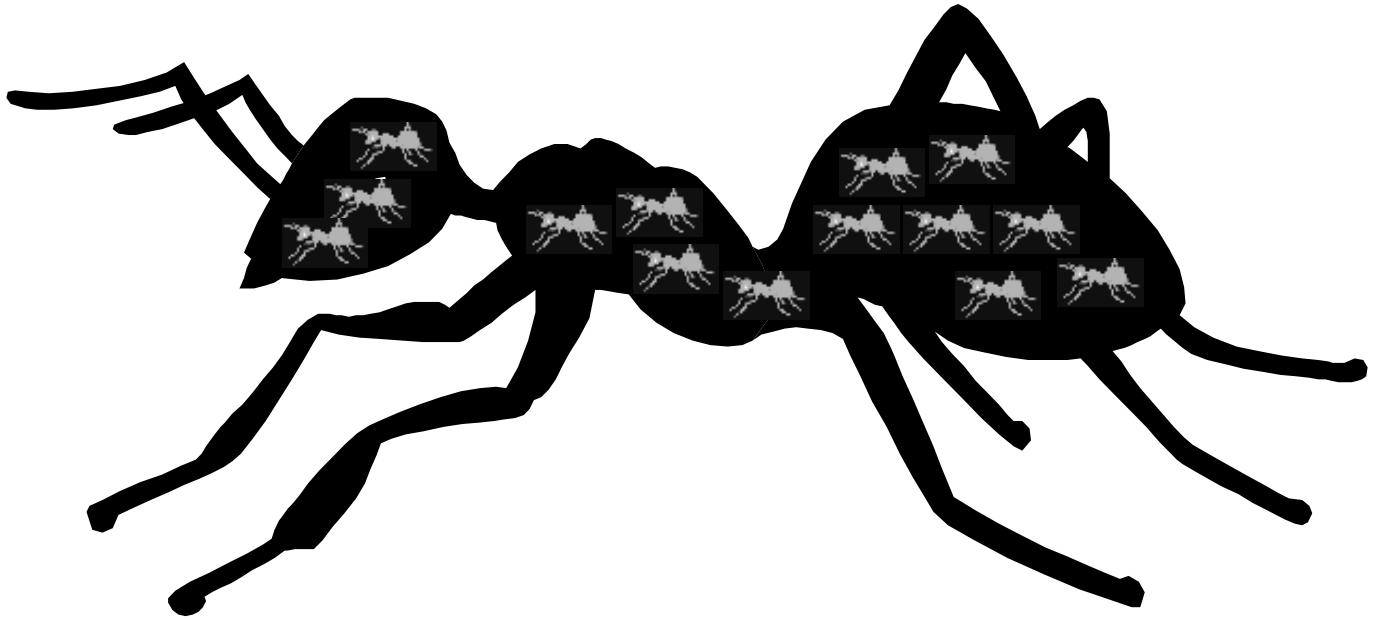
- agenty



- kolónia



Rekurzia v hierarchii

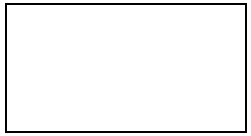


• Hľadali sme architektúru, ktorá by dokázala agenta poňať ako multi-agentový systém nižšej úrovne

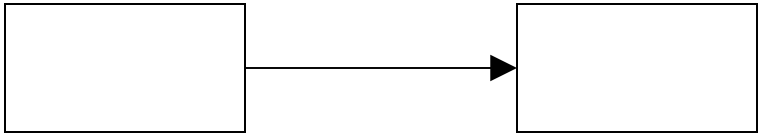
• Kelemenov princíp: povoliť určitým agentom vnímanie a vykonávanie akcií na oboch hierarchických úrovniach

Subsumpčná architektúra

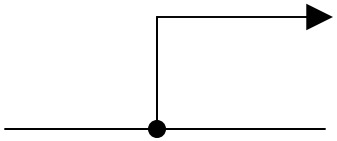
- [Brooks 1987] – základná inšpirácia, lebo moduly majú takmer agentovú povahu



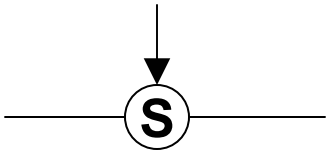
modul



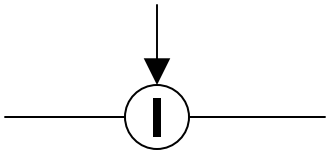
vedenie



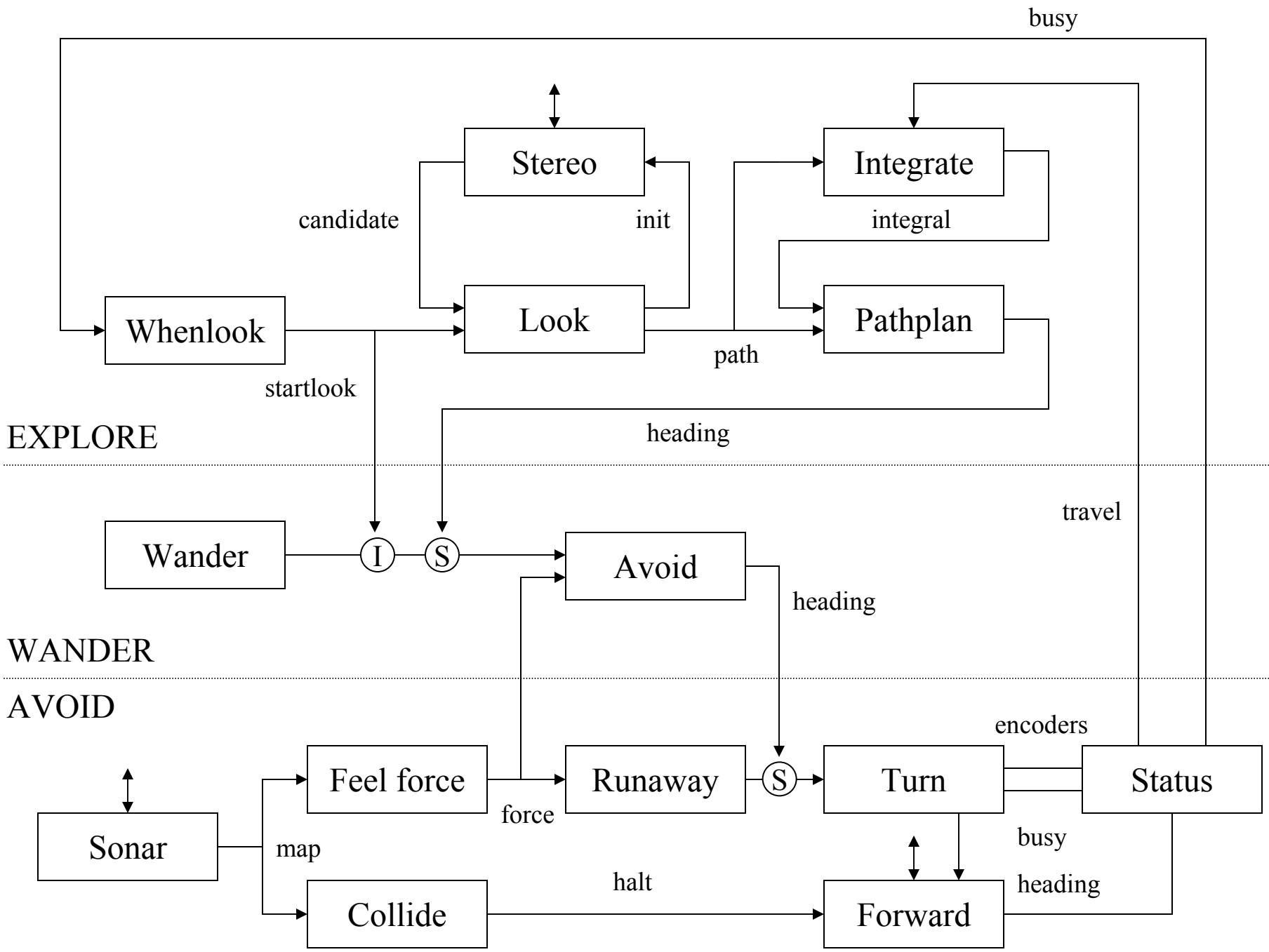
zdvojenie



supresia



inhibícia

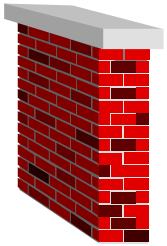

busy
Stereo
Integrate
candidate
init
integral
Whenlook
Look
Pathplan
path
startlook
heading
EXPLORE
travel
Wander
I
S
Avoid
heading
WANDER
AVOID
encoders
Feel force
Runaway
S
Turn
Status
Sonar
force
map
busy
halt
heading
Collide
Forward

Agent-Space Architecture

- reformulácia subsumpčnej architektúry
- nie je určená len pre mobilnú robotiku ale akýkoľvek interaktívny systém (monitorovací systém, riadiaci systém)
- vhodná pre systémy reálneho času a pre systémy vybavené senzormi a aktuátormi
- plynule nadväzuje na bežné programovanie a predstavuje univerzálny programovací prostriedok
- založená na agentovo-orientovanom programaní

Agentovo-orientované programovanie – jeden zo spôsobov modelovania reálneho sveta v počítači

- pasívna entita (záznam, štruktúra)

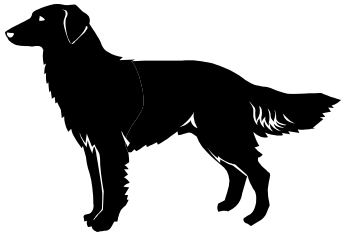


- štruktúrované programovanie

- reaktívna entita (objekt)

- objektovo-orientované programovanie

- proaktívna entita (agent)



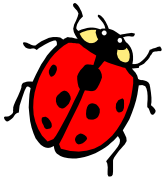
- agentovo-orientované programovanie

V rámci MAS Agent-Space volí

voľba akcií

reaktívna

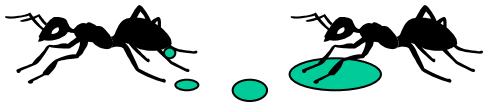
deliberatívna



komunikácia medzi agentami

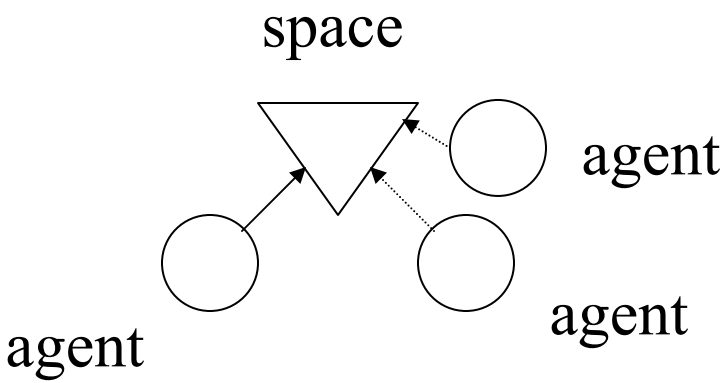
priama

nepriama



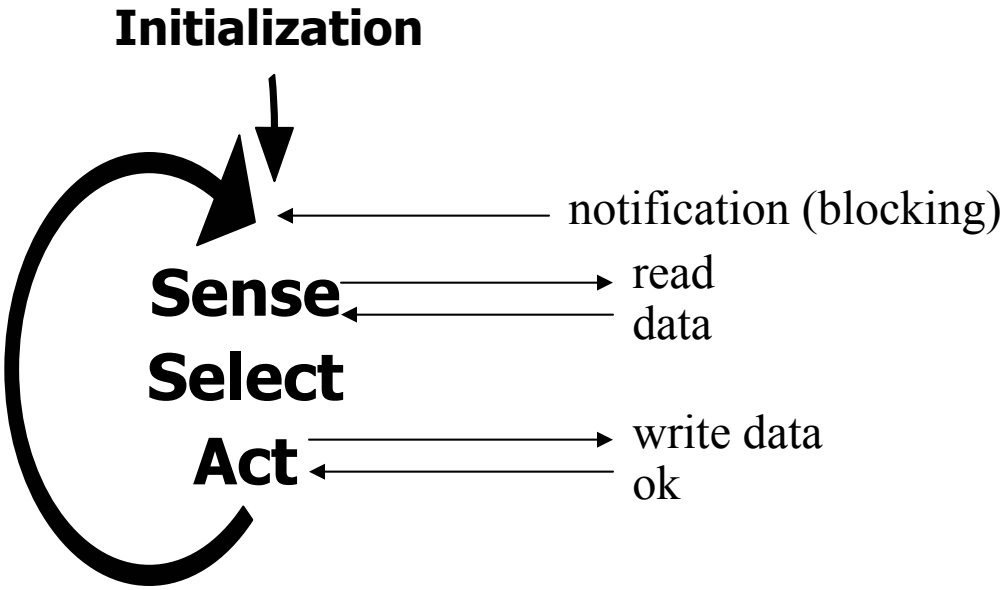
Agent-Space

- Systém sa skladá z *reaktívnych agentov,* ktoré komunikujú nepriamo cez tzv. *space*



Reaktívny agent

- vykonáva cyklus sense-select-act



Reaktívny agent

- Voľba akcií je realizovaná obyčajným kódom bežiacim v jedinom vlákne, ktorý nemá žiadnu špeciálnu formu ani nepoužíva nejaký špeciálny komponent
- Zvolené akcie sú iba púhou reakciou na stav prostredia a vnútorný stav agenta
- Vnútorný stav agenta zodpovedá tým premenným, ktorých obsah pretrváva medzi dvomi, po sebe idúcimi prechodmi cez cyklus sense-select-act

Čistá reaktivita

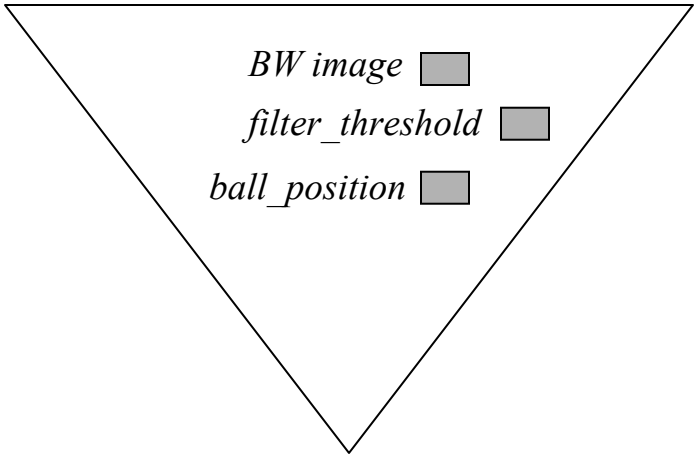
- Čisto reaktívny agent je taký reaktívny agent, ktorý nemá vnútorný stav
- t.j. žiadne globálne premenné, žiadne konštanty, nič, čo by prežilo prechod agentovým cyklom

Budenie agentov

- Primárny mechanizmus: agent má timer s určitou frekvenciou, ktorý ho pravidelne budí k vykonaniu jedného prechodu cyklom
- Doplnkový mechanizmus: Agent si môže u space zaregistrovať trigger, ktorý ho zobudí na základe určitej zmeny, ktorá sa v space odohrá

Space

- obsahuje a spravuje pomenované dáta – tzv. bloky



Space - služby

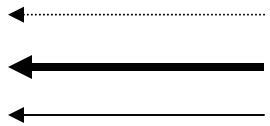
- Space poskytuje agentov služby, ktorými môžu manipulovať s blokmi

- Základné služby sú

  READ

  WRITE

  DELETE



Space – služby

- Ďalšie služby:

  - registrácia triggrov

  - hromadné manipulácie s blokmi na základe masky

- Synchronizácia: jednotlivá služba sa vždy vykonáva bez prerušenia inou, agent má možnosť vykonať aj sériu služieb bez prerušenia

Bloky

- Sú pomenované virtuálne miesta, do ktorých je možné uložiť dáta
- Blok môže byť prázdny, na jeho vytvorenie nie je potrebné zavolať žiadnu špecifickú službu
- Blok is fyzicky vytvorený prvým zápisom dát avšak čítaný môže byť už predtým (odporúčaná stratégia: každý agent si pri čítaní definuje vlastný default)

Bloky

- blok má definovanú časovú platnosť, po vypršaní ktorej je jeho obsah automaticky odstránený
- Space nemá žiadnu vedomosť o význame dát uložených v bloku
- Agent, ktorý blok číta, musí sám vedieť ako má prečítané dáta interpretovať

Bloky

- S blokom možno asociovať prioritu
- priorita je nepovinný parameter operácii zápisu a vymazania bloku
- priorita je reálne číslo, aby bolo aspoň teoreticky vždy možné vložiť medzi dve tretiu
- default je nula
- vyprší spolu časovou platnosťou bloku (dá sa na nulu)
- prepísať obsah bloku bude dovolené len takým zápisom, ktoré definujú priority aspoň takú, aká je asociovaná s blokom

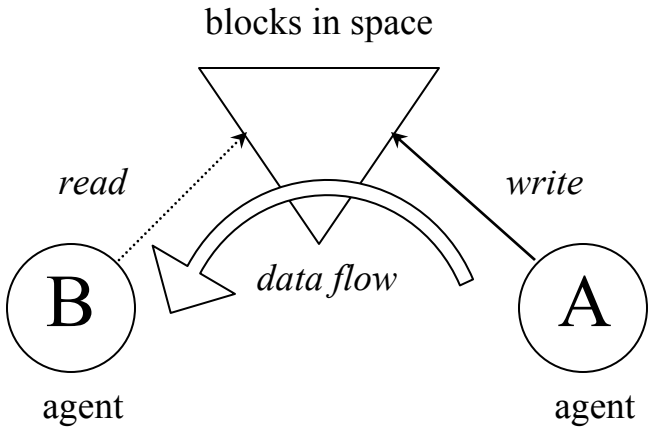
Podoby dát uložených v bloku

Záleží to od platformy, možné stratégie sú

- bufre obsahujúce binárne dáta
- Smerníky na objekty
- Klony objektov
- Marshalované objekty
- XML dokumenty
- XML dokumenty mapujúce jendoduché objekty
- **hybridné formáty**

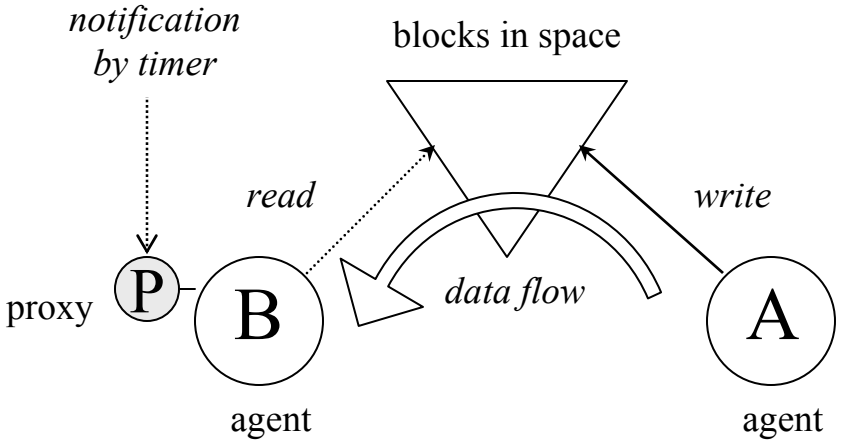
Nepriama komunikácia

Priama komunikácia medzi agentami nie je povolená. Jediný spôsob ako zrealizovať dátový tok medzi dvomi agentami je zaviesť v space blok, ktorý producent zapisuje a konzument číta



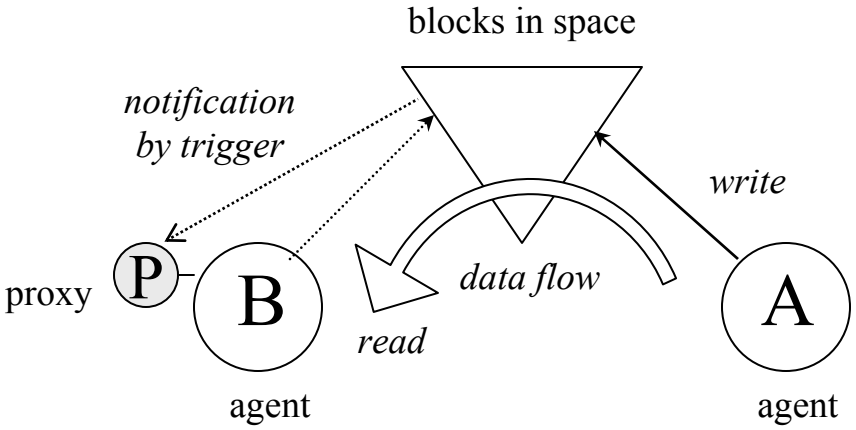
Timer

Prípad 1: konzument je budený timerom – t.j. Pravidelne kontroluje či sú nejaké nové dáta v space alebo proste spracúva tie, čo tam nájde



Trigger

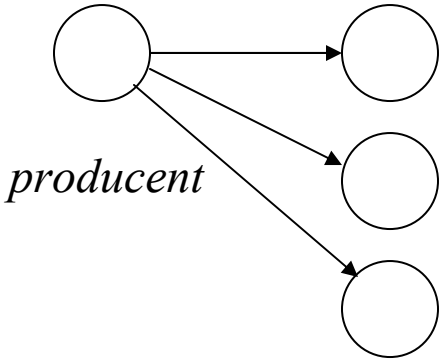
Prípad 2: agent je budený triggerom, keď sa zmení určitý blok v space (ale! Nepravidelnosť zmeny nie je dobrým dôvodom na použitie triggra. Tým je výlučne potreba spracovať dáta okamžite ako sú produkované)



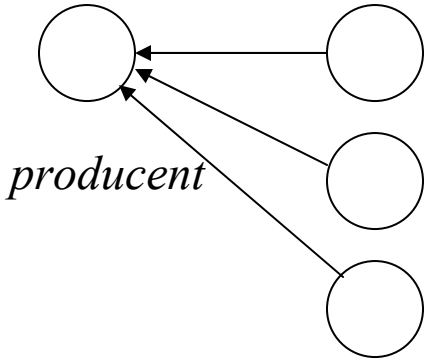
Dátové toky

výhodou je tok many:many

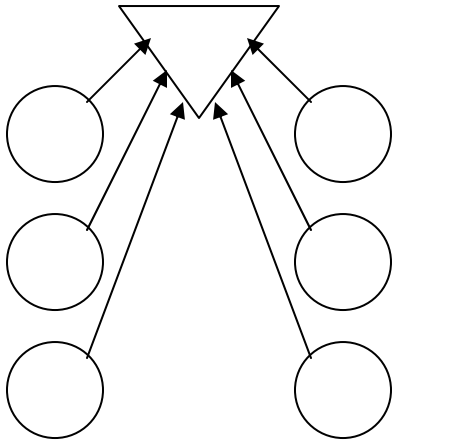
tradičný spôsob



klient-server



agent-space

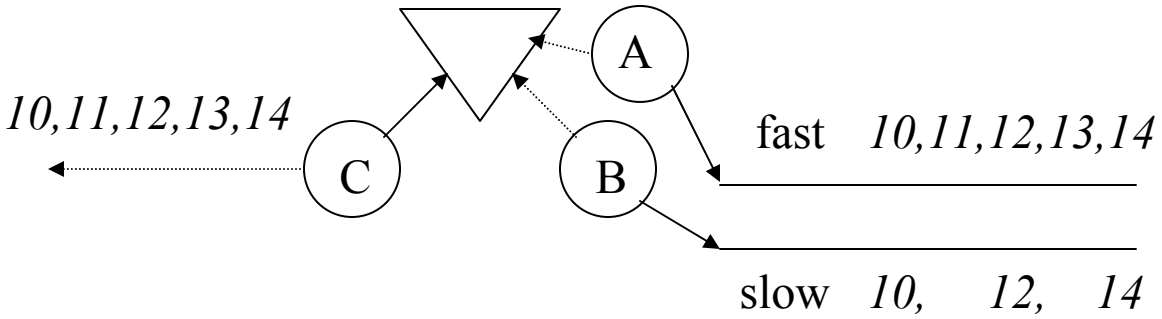


Dôležité pravidlá

- Parametrizácia: mená blokov, nad ktorými agent pracuje by mali byť doň odovzdané ako parametre. Podporuje to jeho znovu použiteľnosť.
- Čistá reaktivita: je dobré preferovať čistú reaktivitu, t.j. Všetky aplikačné konštanty a globálne premenné dávať do blokov. Vnútorný stav agentov tak otvoríme pre ostatných agentov
- Nikdy nezapisovať hodnoty typu „zlá hodnota“ do space. Miesto toho treba zapísať len rozumné hodnoty a stanoviť ich časovú platnosť

Implicitné vzorkovanie

Keďže každý zápis prepisuje dáta do bloku zapísané pri predchádzajúcom zápise a to bez ohľadu na to, či ich niekto stihol prečítať alebo nie, každý dátových tok je automaticky (potenciálne) vzorkovaný.



Základné vlastnosti Agent-Space

- Nepriama komunikácia
- Agentovosť modulov
- Dátové toky many:many
- Implicitné vzorkovanie

Odvodené vlastnosti

+ deadlocky nemožné bez ohľadu na prehľadnosť komunikačnej štruktúry
+ automatická propagácia zmien
- prechodné stavy (nekonzistencie)

Odvodené vlastnosti

- Stratovosť dát
+ Vzokovanie dát
+ podpora reálneho času
+ podporované systémom reálneho času

Odvodené vlastnosti

+ Reštarť ľubovoľnej časti možný
+ Zotavovanie sa z chýb
+ Ľahký štart
+ Dobrá konfigurovateľnosť

Odvodené vlastnosti

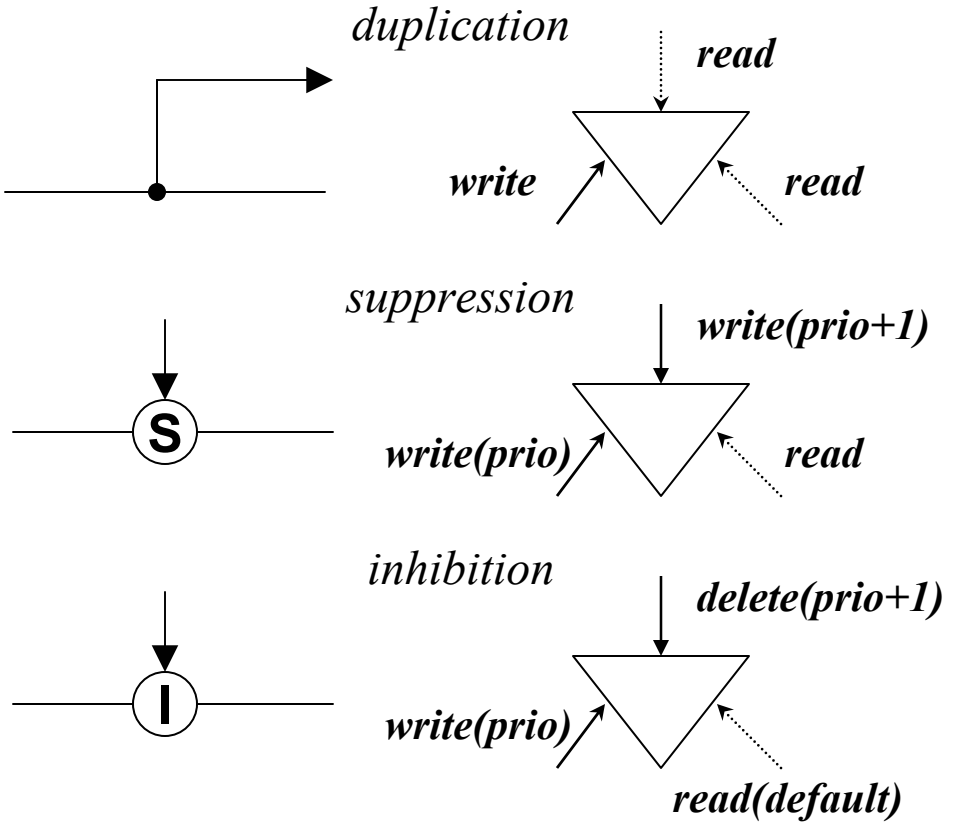
- Space je úzke hrdlo
+ Space je nezávislý od aplikačnej oblasti (všetok závislý kód je sústredený v reaktívnych agentoch)
+ Space je znovupoužiteľný komponent takže sa naň možno sústrediť natoľko, že bude spoľahlivý a bez skrytých chýb

Odvodené vlastnosti

+ systém je otvorený
+ môžeme doň pridať nového agenta, ktorý dokáže monitorovať činnosť pôvodných agentov a využívať výsledky ich práce
+ tento agent môže dokonca ovplyvňovať činnosť pôvodných agentov
+ dobrá modifikovateľnosť
- bezpečnosť

Vzťah ku sumbsumpčnej architetúre

- Riešenie vyjadrené v subsumpčnej architektúre možno poľahky transformovať do agent-space



Vzťah ku sumbsumpčnej architetúre

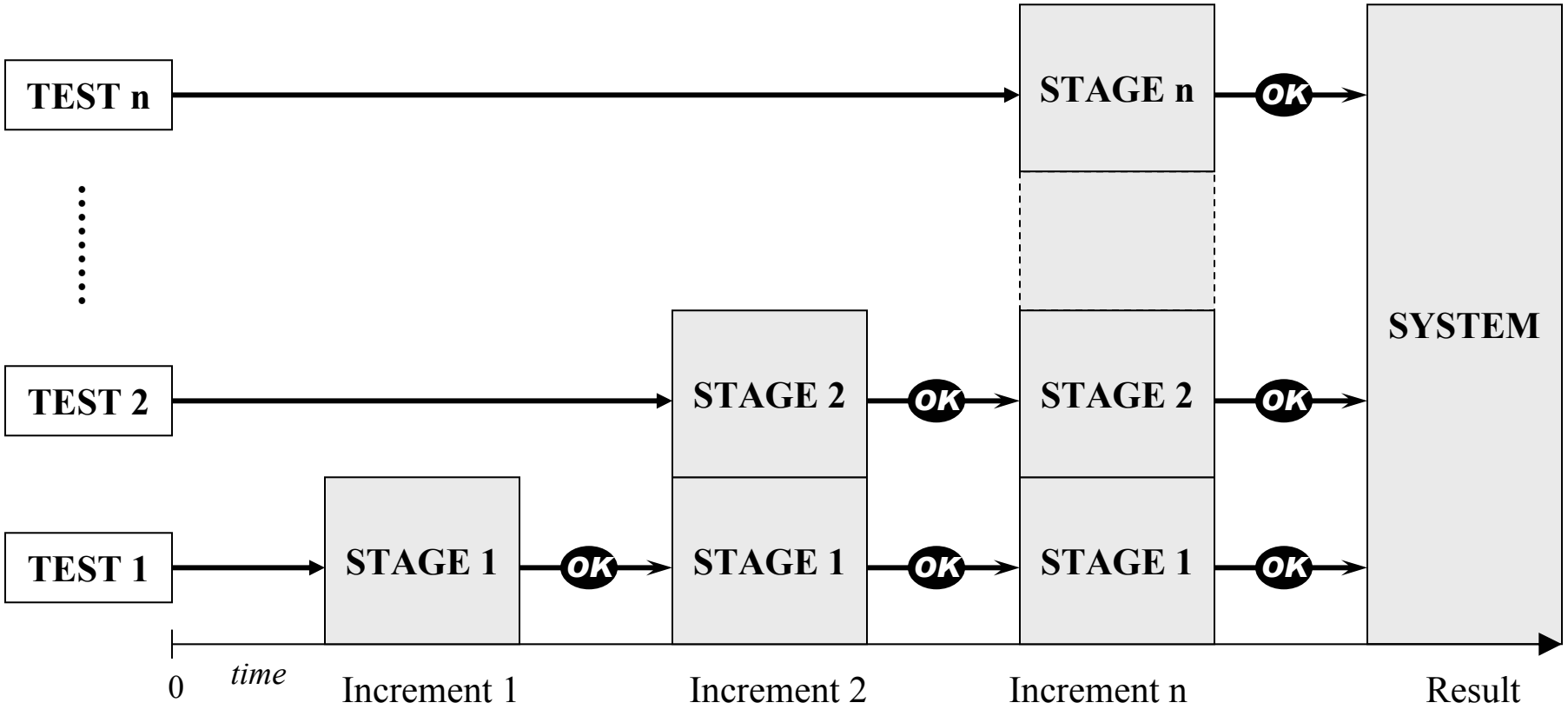
- Naopak to neplatí
- pre agent-space je prirodzená konkurencia, (pre subsumpčnú architektúru priority)
- Referencia bloku menom, ktoré je uložené ako hodnota v inom bloku

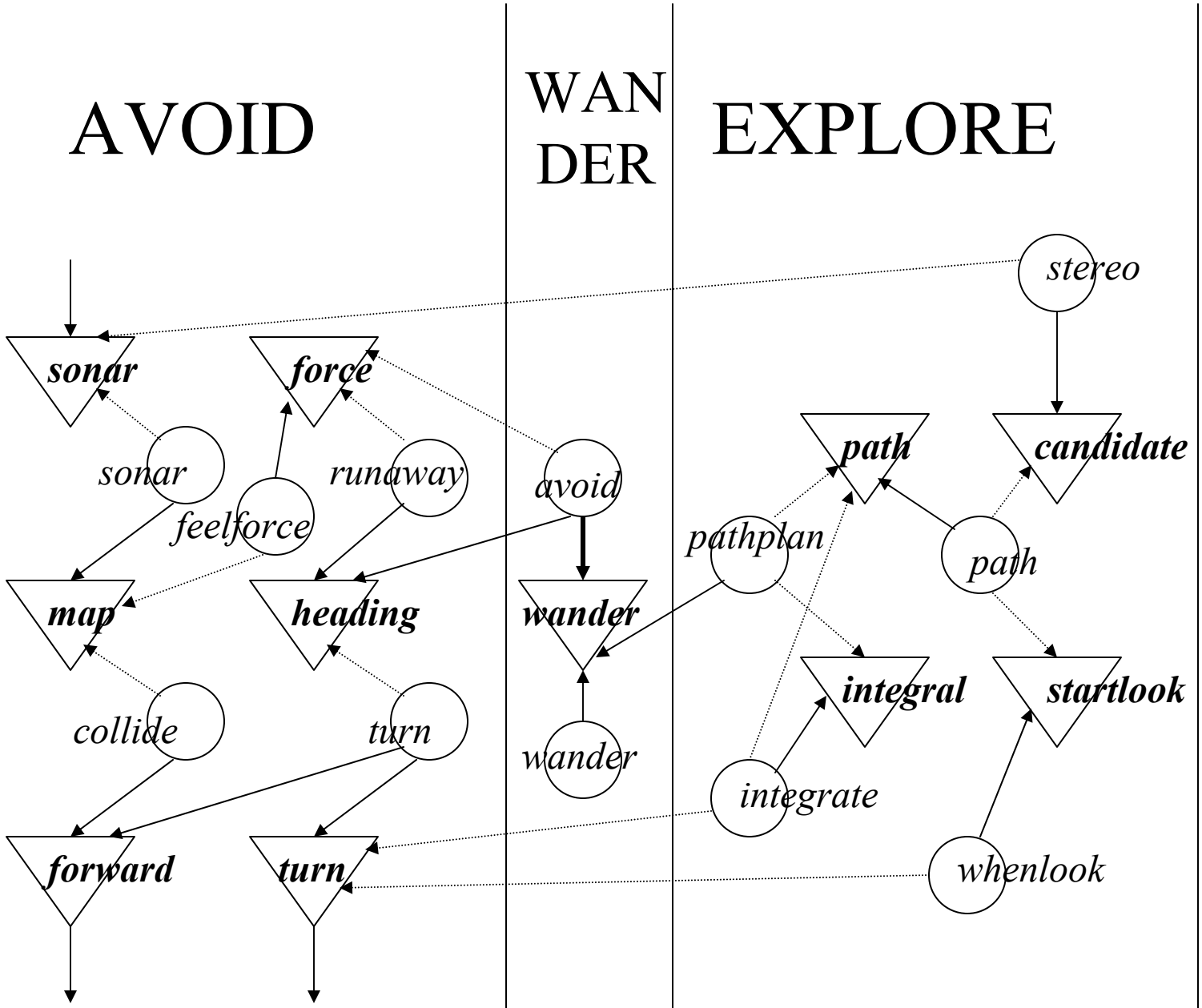
Vývoj

Dôležité je, že systém vyjadrený v agent-space môžeme vyvíjať rovnakou metódou ako v rámci subsumpčnej architektúry, t.j.:

- Inkrementálne
- Zdola nahor
- Dekomponujúc podľa aktivity
- Používajúc vtieranie vyšších vrstiev do činnosti nižších

Vývoj


TEST n
TEST 2
TEST 1
STAGE n
STAGE 2
STAGE 1
STAGE 2
STAGE 1
STAGE 1
OK
OK
OK
OK
OK
OK
SYSTEM
0
time
Increment 1
Increment 2
Increment n
Result


AVOID
WAN
DER
EXPLORE
stereo
sonar
force
path
candidate
sonar
runaway
avoid
feelforce
pathplan
path
map
heading
wander
integral
startlook
collide
turn
wander
integrate
forward
turn
whenlook

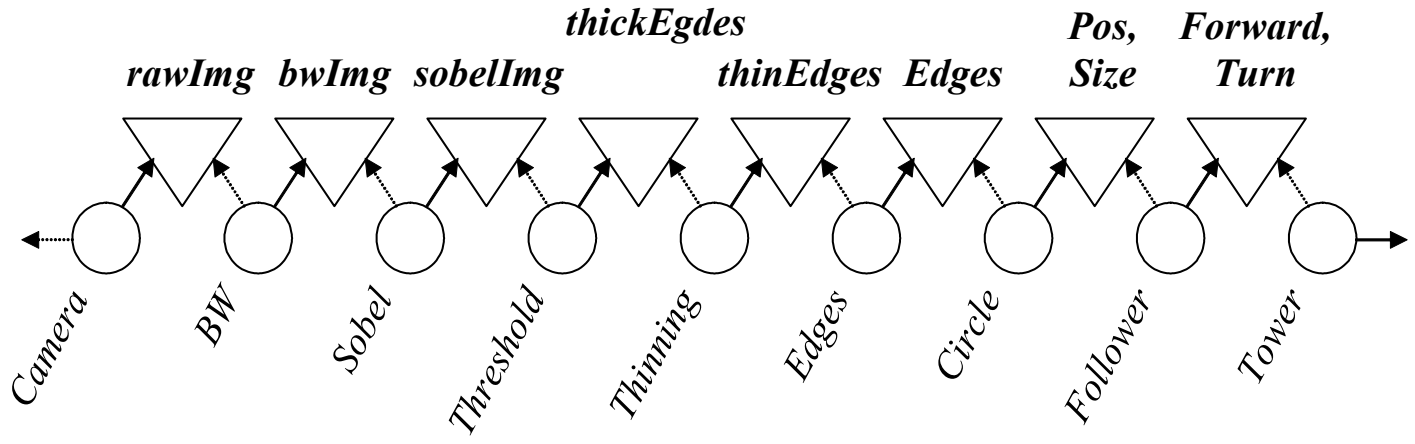
Výhody Agent-Space oproti subsumpčnej architektúre

- atraktívnejší spôsob vyjadrovania
- modelovanie konkurencie (porušenie úrovní kompetencie)
- preferovanie čistej reaktivity (odolnosť voči problému neprístupnosti vnútorného stavu)
- premena hierarchie na enkapsuláciu
- univerzálnosť aplikačnej oblasti

Príklad

Mobilný robot naháňajúci pingpongovú loptičku

Tradičné riešenie: pipeline

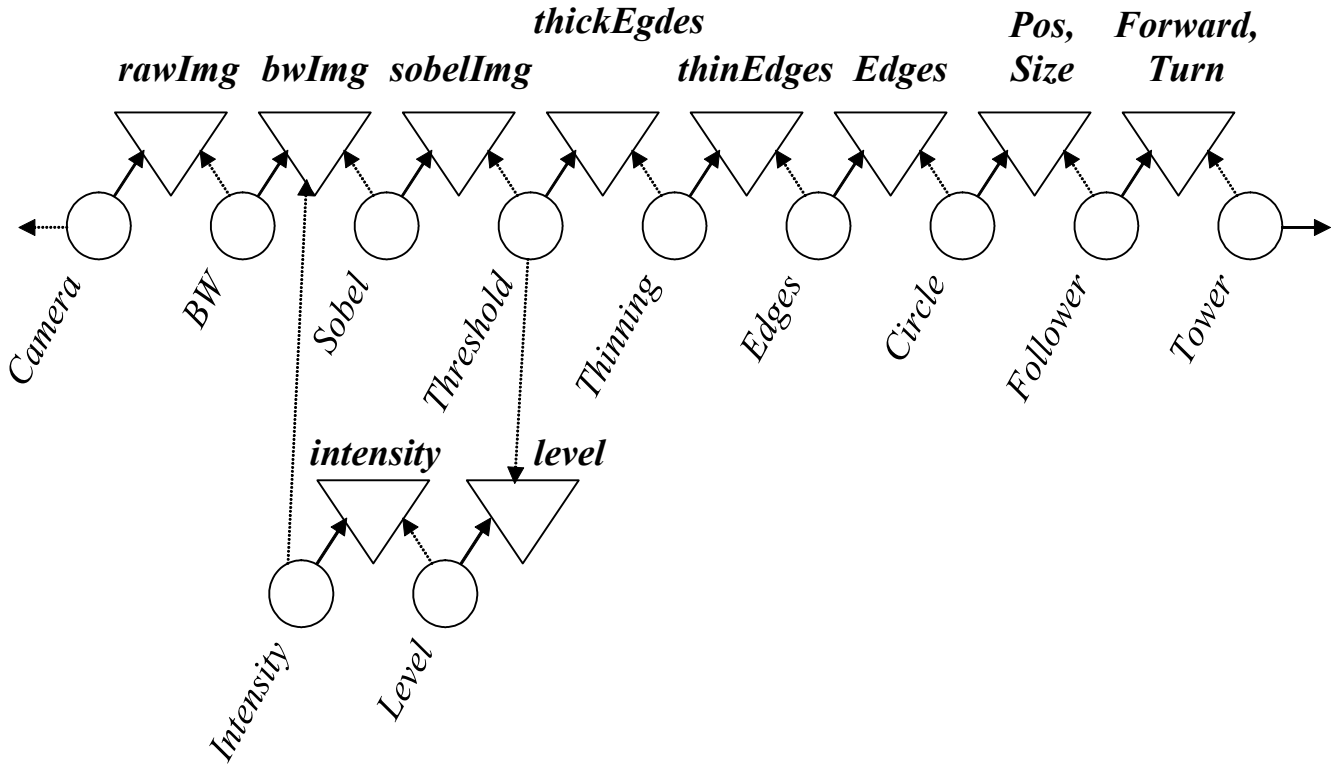


- farebný obraz prijímaný z bezdrôtovej kamery (rawImg) je skonvertovaný na čiernobiely (BwImg)
- na ten je aplikovaný Sobelov operátor, čo zvýrazní hrany (sobelImg).
- potom na základe určitého prahu začiernime v obraze všetko čo netvorí hrany (thickEdges)
- tie premeníme odbúravaním na čiary (thinEdges)
- z takto upraveného obrazu vyberieme zoznam relevantných úsečiek (Edges).
- z týchto sa snažíme zistiť bod, ktorý by mohol byť stredom loptičky (robíme priesečník kolmíc na ich stred) a ten otestujeme či sa okolo neho nachádza významná časť kružnice. Ak niečo nájdeme, vieme polohu stredu loptičky v obraze ako aj veľkosť loptičky na obraze.
- ak je loptička moc vpravo, vydávame príkaz na otáčanie sa doprava, ak moc vľavo, doľava. Ak je loptička moc veľká, cúvame, ak moc malá, ideme dopredu.
- tieto príkazy vyšleme do podvozku robota.

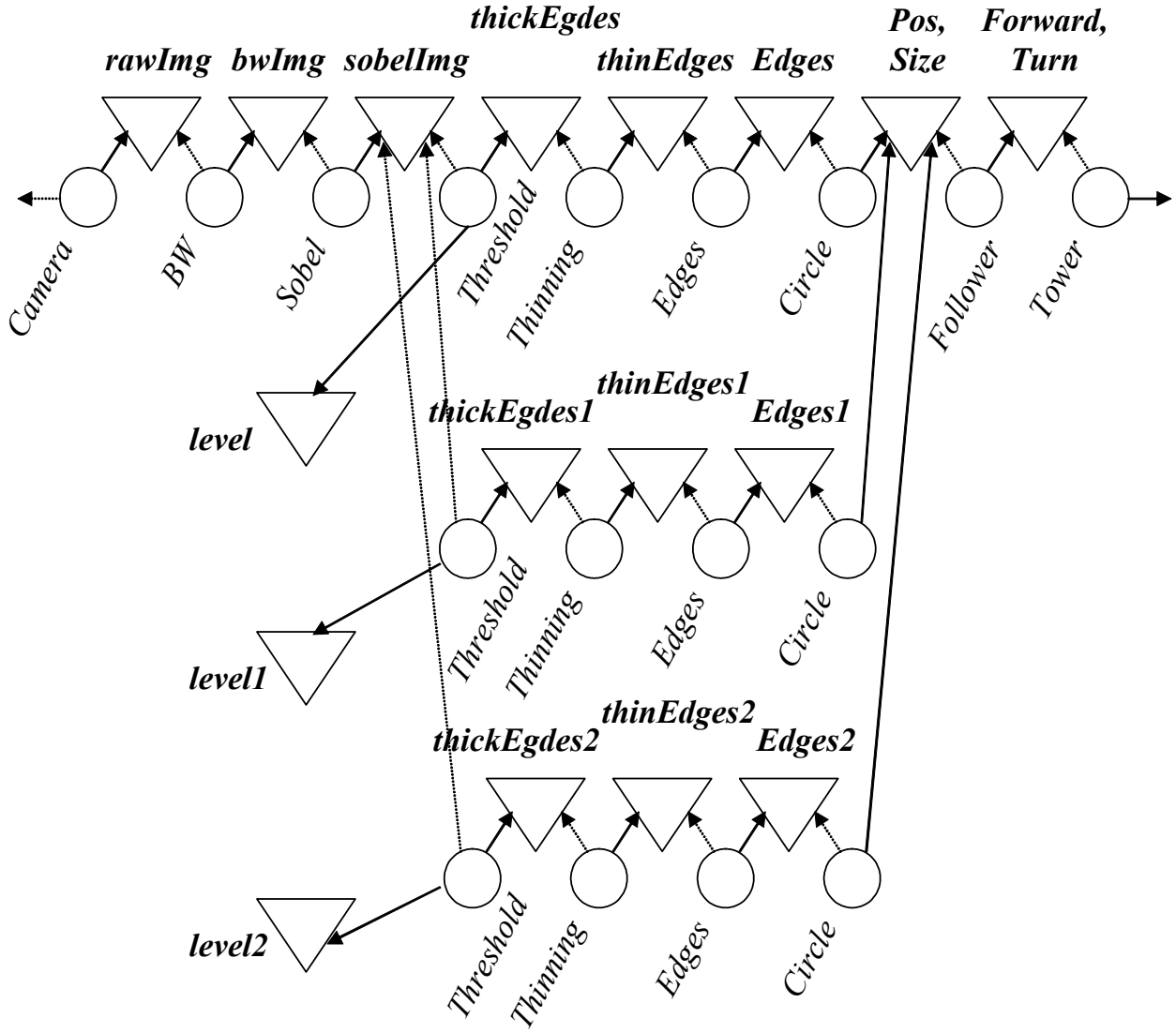
Ukážka kódu agenta

```
public class AgentBW extends Agent {

  public String colorImage;
  private String bwImage;

  AgentBW (String colorImage, String bwImage) {
    this.colorImage = colorImage;
    this.bwImage = bwImage;
  }

  void init () {
    setTrigger(bwImage);
  }

  void sense_select_act () {
    int[] pix = (int[]) read(colorImage);
    if (pix != null) {
      int [] pix2 = Vision.RGB2BW(pix);
      write(bwImage, pix2);
    }
  }

}
```

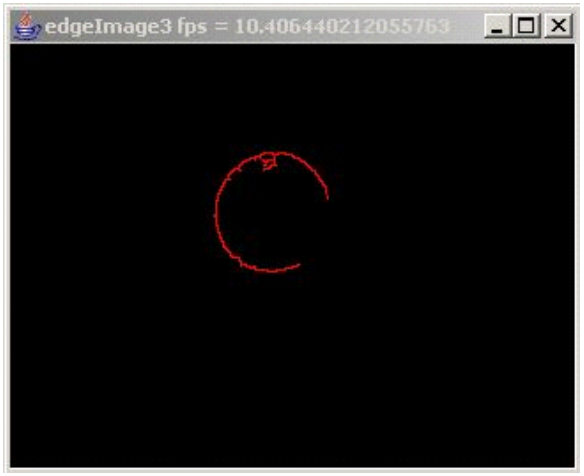
Riešenie s variabilným prahom

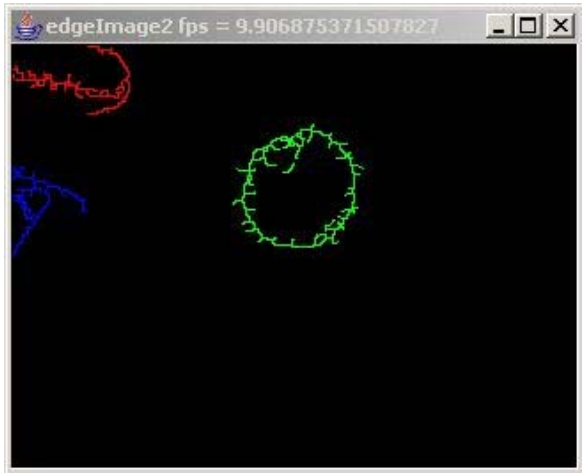


Konkurencia medzi rôznymi prahmi

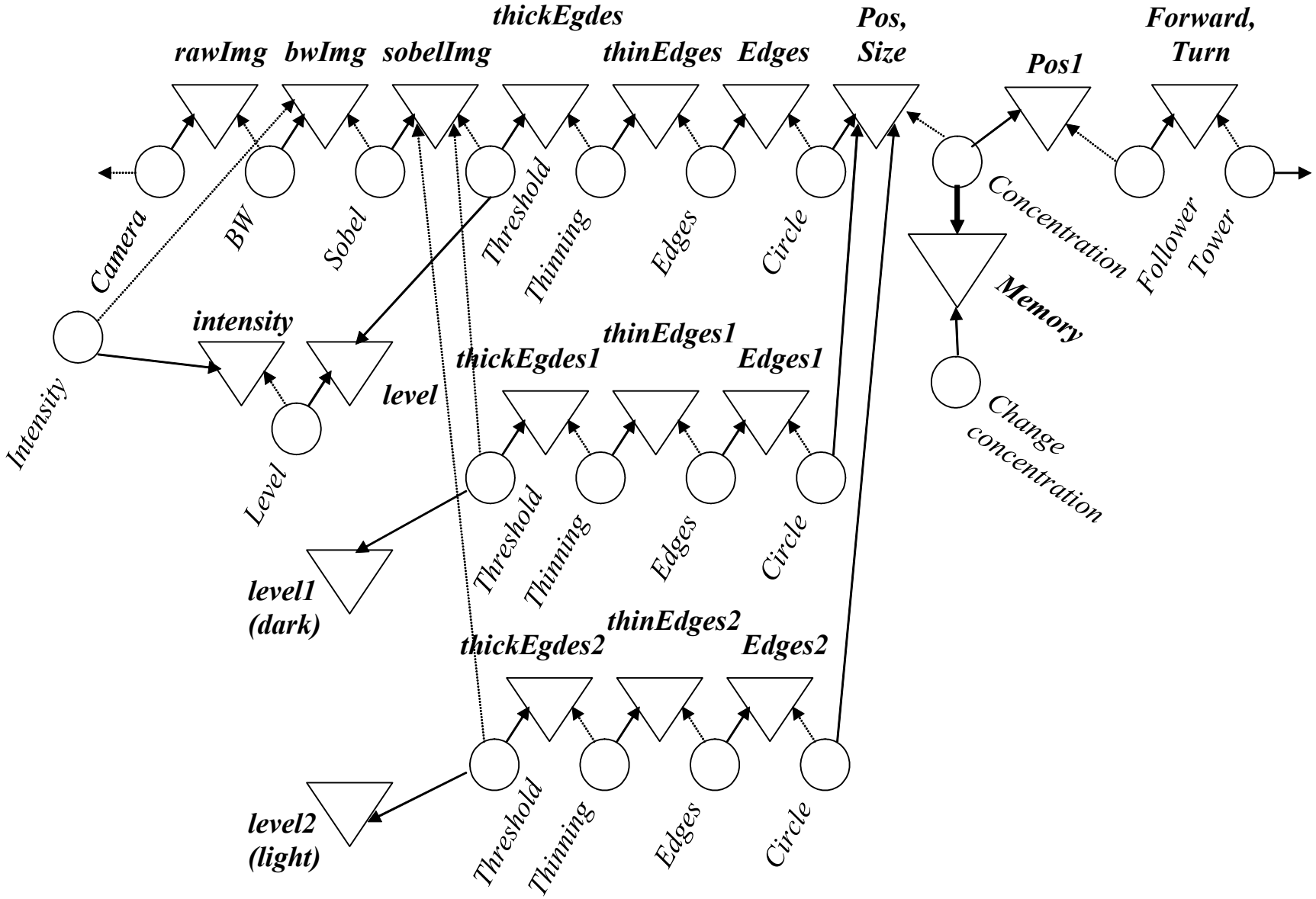


Konkurencia medzi rôznymi prahmi





Sústredenie sa na jednu z loptičiek



Ďakujem za pozornosť !

Andrej Lúčny

MicroStep-MIS & KAI FMFI UK Bratislava, SLOVAKIA

andy@microstep-mis.com

www.microstep-mis.com/~andy