An Abstract Component-based Model for Constructing Operational Models (of Agent Behaviors) for Multi-Agent-based Simulations

Jean-Pierre Briot
Thomas Meurisse

1 Laboratoire d'Informatique de Paris 6 (LIP6)
Université Paris 6 - CNRS

Outline

• Methodological context
• From a design model to an operational model
• Scheduling / Activity tracking
• The MALEVA model of agent components
• Examples
• Directions
• Conclusion
Sketch of a methodological framework for MABS [Drogoul et al. MABS’2002]

Conceptual gaps between thematician and computer scientist [Fishwick 95]

Translating design models into operational models is critical and not trivial [Drogoul et al. 2002] [Axtell 2000]

Example: microsimulation of population evolution

Simplification of demography micro-simulation model Destinie [INSEE 1999]

GetMarried

NewBaby

Divorce

3 behaviors/components (probabilistic state change):

GetMarried  NewBaby  Divorce

Issue for the designer of the model/simulation:
(Note: often not an expert programmer)

In what order should we activate these behaviors?
Possible bias

(a) Mate ; Separate ; Reproduce
(b) Mate ; Reproduce ; Separate
(c) Mate || Reproduce || Separate

Specification of intra-agent temporal dependencies

[Meurisse 04]

(a) Mate ; Separate ; Reproduce  (b) Mate ; Reproduce ; Separate  (c) Mate || Reproduce || Separate

- Via control flow connexions
- Without any change to the components/behaviors
- GUI tool (CGraphGen)
Reengineering of existing behavioral code [Meurisse 2004]

- a Java class (name)
- a method (name)
- method signature
e.g., position Follow(position p)

CGraphGen tool

Follow

- typed ports
- one FIFO for each data input port

MALEVA: An Abstract (Component-based) Model for Constructing/Implementing Operational Models

- Domain: multi-agent-based simulation (*but in fact more general model*)
  - Ex: traffic simulation, eco-systems, population micro-simulation...

- Unit of decomposition: agent behavior
  - Ex: random move, gradient follow, reproduce...

- Assembling behaviors into more complex behaviors
  - Concept of composite component (behavior)

- "Componentification" of control flow:
  - control ports, control connexions, control components
  - Distinction between control flow and data flow

- Helps at a fine-grained control of intra-agent scheduling
  *Activity tracking*

- Supports behavior dynamic change
  - Ex: from an egg, to a larva, to a worker ant
A first example: behavior of a Prey

- if the Prey detects a Predator, it flees away, otherwise, it moves randomly

Reuse of a Prey: Predator

- if the Predator detects a prey, it follows the prey, otherwise, it acts as a Prey (cannibalism among Predators)

Prey is reused as a black box, (notion of composite component)
2nd example: Behavior of an ant

Advantages of explicit control flow

- decoupling activation logic from functionality
- more genericity
- fine grain control of intra-agent scheduling (specification of temporal dependencies)

Activity tracking/control
Directions

- Behavior components library
  - concrete components
    » for multi-agent simulation
    » for other technical (ex: interaction) and application (ex: e-commerce) domains
  - abstract components (design patterns)
  - control components

- Behavior dynamicity
  - (see next slide)

- Control flow vs coordination language term
  - (see next next slide)

Dynamicity (dynamic change of behavior)

- Motivations
  - Internal change - Model change of behavior -
    » Ex: from an egg, to a larva, to a worker ant or to a queen
  - External change - Context-awareness
    » Ex: adapt to change of resources

- Currently implemented through a specific behavior/component [Guillemet and Haïk 1998]
  - named meta-component
  - set up future behavior
  - check what components to keep, to add, to remove
  - install connexions

- Ongoing work:
  - Use a higher-level general realiassemblage model/mechanism
  - MadCar [Grondin et al. 2004]
  - Based on concepts of configuration, roles and policies
From control flow graph to process algebra term

Even with the hierarchy of components (composite components), which helps at encapsulate some complexity of the control flow graph, specifying it is precise but low level.

An alternative direction could then be in using a formalism (coordination language), to specify control coordination language (a very fine grained one).

Process algebra, e.g., CCS

Pi-calculus to handle dynamicity

a compact term

\[ \text{isPrey.Follow} \parallel \text{isPredator.Flee} \parallel (\text{isNoPrey.RandomMove} + \text{isNoPredator.RandomMove}) \]

Conclusion

- Components can be useful to help at decomposing/recomposing agent architectures
- Reification of control
- Data flow and control flow
  - for decoupling activation from functionality (more genericity)
  - for fine-grained control of intra-agents scheduling, Activity tracking
  - Thus helps at control possible bias
- MALEVA model is fine grained (behaviors)
  - but optimizations are possible
- Notion of composite component
- Dynamic change of behaviors
References


