RIO: Model, Language and Tools for Agent-Oriented Software Engineering

BAALA Oumaya  BAKHOUYA Mohamed  BERDAI Abdelhamid  CAMINADA Alexandre  CREPUT Jean-Charles GABER Jaafar  GALLAND Stéphane  GARROUTY Gilles GRUER Pablo  HAJJAM-EL-HASSANI Amir  HARITI Mohamed  HAYARI Noureddine  HILAIRE Vincent KOUKAM Abderrafiaa  LAMOTTE Olivier  LAMROUS Sid Ahmed  LISSAJOUX Thomas  MABED Hakim  RODRIGUEZ Sebastian  RUICHEK Yassine  SIMONIN Olivier

UTBM
Belfort Technopôle 90000 Belfort - France

Agent Link III TFG-AOSE
Outline

Introduction

The RIO meta-model

The OZS notation

Tools: prototyping and verification

Conclusion
Outline

Introduction

The RIO meta-model

The OZS notation

Tools: prototyping and verification

Conclusion
Statements

- Many models and architectures for agent and multiagent systems
- Heterogeneous engineering approaches
- Understanding model is difficult
- Comparison and reuse are difficults
Comparison criteria

Model
- Concepts underlying the model
- Architecture dependant

Language
- Formal
- Semi-Formal

Tools
- Validation
- Verification
- Code generation
Model

- Organizational
- Componential [Brazier et al., 1997]
- Architecture dependant
- Application range
Language

Semi-Formal

- Mainly extensions of Object-Oriented notation: AUML [Bergenti and Poggi, 2000], MASE [DeLoach, 1999], ADELFE [Bernon et al., 2002]

Formal

- Formal Method [Luck and D’Inverno, 1995],
- Modal and Temporal Logic GAIA [Wooldridge et al., 1999]
- Knowledge Representation DESIRE [Herlea et al., 1999]
Advantages of formal approach

- Non ambiguous semantics.
- Precise specifications.
- Proof, simulation, prototyping.
Drawbacks of formal approach

- Difficult to produce an implementation.
- Limited expressive power (reactive aspects, transformational aspects, ...).
Tools

- Model checking [Herlea et al., 1999, Gruer et al., 2000]
- Proofs
- Prototyping [Hilaire et al., 2000a, Hilaire et al., 2000b]
Needed for a methodology

Objectives

- Concepts suited for description of a wide range of MAS.
- Covering phases from formal specification to implementation.
- Validation and Verification.
Outline

Introduction

The RIO meta-model

The OZS notation

Tools: prototyping and verification

Conclusion
Definitions

**Definition**

A role is an abstraction of a behaviour or a status in an organisation.

**Definition**

An organization is a set of roles such as every role interacts with at least one role in the organization.
Meta-model overview

Actor
attributes
actions
stimulus

Role
behaviour
+

RoleContainer
playing
0..1
*

Agent
acquaintances
position
Organizational meta-model

- The meta-model uses concepts such as organization and role as first class citizen.
- Each concept (role, interaction, organization) is specified by a class with the OZS notation.
- Specification consists in specialization of framework classes.
Organization example

![Organization Example Diagram]
Outline

Introduction

The RIO meta-model

The OZS notation

Tools: prototyping and verification

Conclusion
Multiformalisms notation

We have chosen two formalisms: Object-Z [Duke et al., 1991] and statecharts [Harel, 1987].

- Great expressive power.
- Specification simplified.
- Operational semantics.
Integration principles

[Gruer et al., 2004]

Syntaxic integration

Object-Z classes integrates a statechart which specifies their behaviours
Integration principles

[Integrating Object-Z classes with a state-chart into transition systems (Gruer et al., 2004)]

Semantic integration
Object-Z classes with a state-chart are given an operational semantics with transition systems.
Individual

Role

\[\text{current} : \text{Action}\]
\[\text{initialWeight}, \text{weight} : \text{Action} \to [0, 1]\]
\[\text{progressionReward} : \text{Action} \to \text{BMValue}\]
\[\text{s} : \text{DiscreteSensor}\]
\[\text{satisfaction, I} : [-P_{\text{max}}, P_{\text{max}}]\]
\[\alpha : [0, 1]\]

\[\text{current} \subseteq \text{actions}\]
\[\text{obtainCondition} = \{|\text{I}_{\text{ext}}()| \leq P() \land |\text{I}_{\text{ext}}()| \leq I\}\]
\[\text{leaveCondition} = \{|\text{I}_{\text{ext}}()| \geq P() \land |\text{I}_{\text{ext}}()| \geq I\}\]
\[ l_{\text{ext}} \]
\[ \text{ext!} : \mathbb{R} \]
\[ \text{ext!} = s.\text{getMax}() \]
behavior

Individual

/actionSelection

act

individualAction

Quiet

[s.perceiveHinderer]

Repulsion

Emission

Attraction

Emission

[s.decideAttract]
Altruist

Role

\[ \text{progressionReward} : \text{Action} \rightarrow \text{BMValue} \]
\[ s : \text{DiscreteSensor} \]
\[ \text{satisfaction, } I : [-P_{max}, P_{max}] \]

\[ \text{current } \subseteq \text{actions} \]
\[ \text{obtainCondition} = \{|l_{ext}()| \geq P() \land |l_{ext}()| \geq I\} \]
\[ \text{leaveCondition} = \{|l_{ext}()| \leq P() \land |l_{ext}()| \leq I\} \]

\[ I_{ext} \]
\[ \text{ext!} : \mathbb{R} \]
\[ \text{ext!} = s.\text{getMax()} \]
behavior

Altruist

Altruism

altruismReaction

Quiet

Emission

[propagationCondition]
Outline

Introduction

The RIO meta-model

The OZS notation

Tools: prototyping and verification

Conclusion
Prototyping

![Diagram](image-url)
Verification

- Model-checking
- Verification
Outline

Introduction

The RIO meta-model

The OZS notation

Tools: prototyping and verification

Conclusion
Conclusion

- Formal specification linked to implementation.
- Modularity and abstraction,
- Simulation and animation.
- Proofs.
Perspectives

- CASE Tool currently in development.
- Compositional verification techniques.
- Refinement process.
- Integration with existing semi-formal notations.


Designing a multi-agent solution for a bookstore with the passi methodology.

**DeLoach, S. (1999).**
Multiagent systems engineering: a methodology and language for designing agent systems.
*In Agent Oriented Information Systems ’99.*

The Object-Z specification language.
Technical report, Software Verification Research Center, Departement of Computer Science, University of Queensland, AUSTRALIA.

**Ferber, J. and Gutknecht, O. (1998).**
A meta-model for the analysis and design of organizations in multi-agent systems.


