RBAC-MAS & Agent Coordination Contexts

Andrea Omicini
after Mirko Viroli, Alessandro Ricci
Alma Mater Studiorum – Università di Bologna
MAS Scenario

- An open MAS where agents come in and interact with a (structured) environment
  - the agent workspace
  
in order to achieve their own goals

- The individual view of the agent environment is made of
  - other agents
  - artifacts
    - modelling the agent environment
    - structuring the agent workspace
  - agents *speak* to agents, agents *use* artifacts, artifacts *link* with artifacts
    - [Omicini et al. @ AAMAS 2004 “Coordination Artifacts”],
    - [Viroli et al. @ E4MAS 2005 “Engineering the Environment with Artifacts”]
Reference MAS Architecture

Incoming Agent

Artifacts

Agents

ENVIRONMENT
Roles, RBAC & RBAC-MAS

• What are the consequences of this scenario on the management of a MAS organisation?
  – managing organisation as either a human or an agent

• In the most common and simplest acceptation, this means supporting a notion of role
  – each agent can enter a MAS only if authorised
  – each agent is allowed to play some given roles
  – each role enables / prevents given agent interactions within the MAS
    • hence supporting a notion of security as well

• In particular, which role model for a MAS?
  – we start from the RBAC model
  – and devise out our RBAC-MAS
RBAC

**SSD**

**USERS**

**ROLES**

**SESSIONS**

**OPS**

**OBJJS**

**PERMISSIONS**

- User assignment
- Role Hierarchy
- Role permissions
- User sessions
- Session roles
- DSD
RBAC Features

- Role-Based Access Control (RBAC)
  - [“Proposed NIST standard for role-based access control”, by Ferraiolo et al., ACM Transactions on Information and System Security, 2001]
  - makes it possible to establish relations
    - between roles
    - between permissions and roles
    - between users and roles

- The notion of role
  - is meant to model competency, authority and responsibility
  - is policy-neutral

- Interactions
  - are seen as operations over objects
  - generally model permissions in an abstract way
    - such as credit and debit

- Separation of Duties
  - rules for role inter-dependencies
RBAC-MAS

AGENTS (CLASS) → Agent-role assignment → ROLES → Role policy → ACTIONS

Agent Sessions → Sessions roles → SD

SESSIONS

POLICIES → PERCEPTIONS
RBAC-MAS Features

• RBAC-MAS
  – [“RBAC for Organisation and Security in an Agent Coordination Infrastructure”, by Omicini&Ricci&Viroli, ENTCS 128(5)]

• Moving to MASs
  – *users* become *agent* (classes)
  – *operations* become *actions/perceptions*
  – *objects* become *agents/artifacts*
  – *permissions* become *policies* (protocol templates)

• How can we model sessions?
  – we need a run-time notion to enact this
Agent Coordination Context

AGENT

ACC

ENVIRONMENT

Artifacts

Agents

Agent
ACC Features

• Agent Coordination Contexts
  – available in TuCSoN
    • http://tucson.sourceforge.net

• Represents and enacts
  – an interface for the agent towards the environment
  – the agent from the MAS viewpoint

• One ACC for each individual agent in the MAS
  – it is the only means by which an agent can interact with the MAS
  – the ACC enables only some given actions/perceptions
    • dynamically
    • according to a history-dependent policy
Organisation Infrastructure

• In principle, it can be added on top of any existing MAS infrastructure
  – it adopts the RBAC-MAS model
  – it uses ACCs to enact sessions

• MAS designer duties
  – role specification
    • that is, specify agent classes, roles, policies and their mutual relationships

• Infrastructure responsibilities
  – deliver ACCs to agents
  – maintain their global consistency

• Agent behaviour
  – is both *enabled* and *constrained* accordingly
Infrastructure View

Existing Social MAS

Artifacts

Agents

Incoming agents

Organisation Layer

Role spec.
Phase: Entering a MAS

- The agent is first authenticated & classified
- The agent negotiates successfully a set of policies for interacting in the MAS
- A new ACC is prepared accordingly
- The agent enters the ACC
Phase: Role (De/)Activation

- The agent can ask for a role to be activated
- If this is granted, all the policies for that role are enabled by the ACC
- Eventually, a role can reach a state where it can be deactivated
Phase: Interacting in a MAS

- Given a role and a policy of its, at a given time, the agent is enabled to execute one or more actions
  - this information can be inspected by the agent

- When an action is executed, and the state of the policy state is accordingly changed in a dynamic way
  - hence a policy can specify true protocols
Phase: Managing Organisation

- Some roles may even enable agents to manage the organisation
  - reading/adding/removing/updating roles, policies, ...
Phase: Quitting MAS

- If no roles are currently activated, the agent can ask to leave the ACC, hence the society.
Infrastructure Design

• Formal specification based on process algebras

• Syntax of a whole MAS configuration
  – specified with a BNF grammar

• Dynamic aspects of MAS
  – managed by a layered operational semantics
  – for agents, ACCs, negotiation aspects
## Syntax

\[
S ::= X \parallel O \parallel E \quad \text{MAS configuration}
\]

\[
X ::= 0 \quad \text{agent configuration}
| \langle a, A \rangle \quad \text{inert agent}
| \langle a, (C)A \rangle \quad \text{active agent}
| X \parallel X \quad \text{agents}
\]

\[
O ::= K \parallel V \quad \text{organisation configuration}
\]

\[
K ::= 0 \quad \text{structure configuration}
| \{c(n)\}_C \quad \text{agent class (with cardinality)}
| \{c, r\}_{CR} \quad \text{class role}
| \{r, p\}_{RP} \quad \text{role policy}
| \{p := P\}_P \quad \text{policy definition}
| \{[RS] + r\}_{SD} \quad \text{separation of duty constraint}
| K \parallel K \quad \text{control structures}
\]

\[
RS ::= 0 \mid r \mid RS \parallel RS \text{ roleset}
\]

\[
V ::= 0 \quad \text{activity configuration}
| [a, c]_A \quad \text{active agent class}
| [a, r]_{AR} \quad \text{active agent role}
| V \parallel V \quad \text{activity controls}
\]
Agent Behaviour and Actions

\[
A ::= 0 \mid \alpha.A \mid A + A \quad \text{agent action configuration}
\]

\[
\alpha ::= r : p : \phi \mid \nu \quad \text{agent actions}
\]

\[
\phi ::= \epsilon \mid \omega \quad \text{environment / organisation actions}
\]

\[
\omega ::= +K \quad \text{control structure addition}
\mid -K \quad \text{control structure removal}
\mid K \leftrightarrow K \quad \text{control structure update}
\mid ?K \quad \text{control structure reading}
\]

\[
\nu ::= \quad \text{ACC actions}
\]

\[
\downarrow \quad \text{ACC entry}
\mid \uparrow \quad \text{ACC exit}
\mid +r : R \quad \text{role activation}
\mid -r \quad \text{role deactivation}
\]

\[
\iota ::= a \nu \quad \text{negotiation actions}
\]
**ACC Configuration**

\[
\begin{align*}
C & ::= 0 & \text{ACC configuration} \\
   & | \quad r : R & \text{active role} \\
   & | \quad C \parallel C & \text{ACCs} \\
R & ::= 0 & \text{role} \\
   & | \quad p : P & \text{active policy} \\
   & | \quad R \parallel R & \text{roles} \\
P & ::= 0 & \text{policy} \\
   & | \quad \pi & \text{controlled action} \\
   & | \quad P ; P & \text{sequence} \\
   & | \quad P + P & \text{choice} \\
   & | \quad P \parallel P & \text{interleaving} \\
   & | \quad D & \text{(recursive) definition call} \\
\pi & ::= \phi \mid \diamond & \text{policy actions}
\end{align*}
\]
Interaction Rule

• Agent $A$ in an ACC with role $R$ and policy $P$
• Executes action $\phi$, subject to substitution $\sigma$
• The ACC policy accordingly moves to $P'$
Operational Rules

\[ A \xrightarrow{\perp} A' \]
\[ \langle a, A \rangle \xrightarrow{a} X \langle a, (0)A' \rangle \]

[ENTER]

\[ A \xrightarrow{\perp} A' \]
\[ \langle a, (0)A \rangle \xrightarrow{a\uparrow} X \langle a, A' \rangle \]

[QUIT]

\[ A \xrightarrow{\perp} A' \]
\[ \langle a, (C)A \rangle \xrightarrow{a+r:R} X \langle a, (C\parallel r : R)A' \rangle \]

[ACTIVATE]

\[ \langle a, (C\parallel r : R)A \rangle \xrightarrow{a-r} X \langle a, (C)A' \rangle \]
\[ (P \xrightarrow{\sigma \bowtie \diamond} P' \text{ or } P \not\xrightarrow{\rho} P) \]

[DEACT-REC]

\[ \langle a, (C\parallel r : (R\parallel p : P))A \rangle \xrightarrow{a-r} X \langle a, (C)A' \rangle \]

[DEACT-FIX]

\[ A \xrightarrow{-r} A' \]
\[ \langle a, (C\parallel r : 0)A \rangle \xrightarrow{a-r} X \langle a, (C)A' \rangle \]