A Requirements-Driven Methodology for MAS
An E-Commerce Example

Manuel Kolp
Department of Computer Science
University of Toronto
November 13, 2000

At a Glimpse

- Agents
- Mismatch and Requirements-Driven
- **Developing a B2C System with TROPOS**
  - Requirements Analysis (organizational modeling)
    - Early Requirements
    - Late Requirements
  - System Design focuses (organizational architectures)
    - Architectural Design
    - Detailed Design
  - Implementation (agent organizations)
What is an Agent?

- Entity who can act
  - Autonomous, re-, pro- active, communicative, adaptative, with/in its environment
    ➠ Intelligence

- Software Agent
  - Implemented with/in software technologies
  - Environment :humans, machines, and other software agents in various platforms.

- Multi-agent systems: organizations of agents to achieve particular, possible and common goals.

Inside the BDI Model

Human ➔ Belief, Desire, Intentions Agent

Beliefs - perceived understanding of the world
Goals or desires

Execution Engine

Intents - currently executing plans

Pre-compiled plans
Mismatch and Requirements-Driven

- Operational environment understood in terms of actors, responsibilities, objectives, tasks and resources, roles, needs...
- BUT information systems conceived as a collection of (software) modules, data structures and interfaces.
- Poor quality, frequent failure
- Development techniques have been implementation-driven.
- Aligning analysis, design and implementation makes sense.
- Why not requirements-driven? (Analysis: most important step)
- TROPOS: I* concepts not only for Early but also Late Requirements, Design and even (Agent) Implementation

Who is taking Requirements into account?

<table>
<thead>
<tr>
<th>TROPOS</th>
<th>KAOS</th>
<th>GAIA</th>
<th>Z</th>
<th>AUML</th>
<th>UML &amp; co.</th>
<th>The GAP!!</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Early requirements | Late requirements | Architectural design | Detailed design | Agent Implementation
A User 2 On-line Buying Case Study

- Media taxonomy
  - on-line catalog
  - DBMS
- E-Shopping Cart
  - Check In
  - Buying
  - Check Out
- Search Engine
  - catalog browser
  - Keywords
  - full-text
- Secure
  - $ transactions
  - orders
- Multimedia
  - description
  - samples

Early Requirements Analysis with TROPOS

- Understanding the problem by studying an existing organizational setting;
- Output: Organizational model with relevant actors and respective goals.
Means-Ends Analysis & Functional Alternatives

Specifications describing functional and non-functional requirements for the system-to-be within its environment.
Leaving goal dependencies makes sense whenever there is a foreseeable need for flexibility, reusability, modularity.

- We have left 3 goals: Availability, Security and Adaptability.
- Ex.: Security Strategies supported and let the system decide.
Architectural Design

- Global system architecture defined in terms of subsystems, interconnected through data, control and other dependencies.

- 3 Steps
  - 1 Macro level: Architectural Styles (Organizational Styles): “From Late Requirements to Architectural Design”
    - Vertical Integration, Pyramid, Joint Venture, Structure in 5, Bidding, Hierarchical Contracting, Co-optation, Takeover
  - 2 Micro level: Patterns (Agent Community)
    - Broker, Matchmaker, Contract-Net, Mediator, Monitor, Embassy, Wrapper, Master-Slave, ...
  - 3 Assigning Actors to Agents, Positions, Roles: “From Actors to Agents”

Organizational Architectures (Macro Level)

- Joint Venture
- Structure in 5
Organizational Architectures (Macro Level)

- Vertical Integration
- Bidding

Organizational Architectures (Macro Level)

- Takeover
- Arm’s Length
Organizational Architectures (Macro Level)

Co-optation

Hierarchical Contracting

Quality Attributes for Agent Systems

<table>
<thead>
<tr>
<th>Correlation Catalogue</th>
<th>Predictability</th>
<th>Security</th>
<th>Adaptability</th>
<th>Cooperativity</th>
<th>Competitiveness</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structure-in-5</td>
<td>HELP</td>
<td>HELP</td>
<td>HELP</td>
<td>HELP</td>
<td>HURT</td>
<td>HELP</td>
</tr>
<tr>
<td>Pyramid</td>
<td>MAKE</td>
<td>MAKE</td>
<td>MAKE</td>
<td>MAKE</td>
<td>BREAK</td>
<td>MAKE</td>
</tr>
<tr>
<td>Join Venture</td>
<td>HELP</td>
<td>HELP</td>
<td>MAKE</td>
<td>HELP</td>
<td>HURT</td>
<td>HELP</td>
</tr>
<tr>
<td>Bidding</td>
<td>BREAK</td>
<td>BREAK</td>
<td>MAKE</td>
<td>HURT</td>
<td>MAKE</td>
<td>HURT</td>
</tr>
<tr>
<td>Takeover</td>
<td>MAKE</td>
<td>MAKE</td>
<td>HURT</td>
<td>MAKE</td>
<td>BREAK</td>
<td>HELP</td>
</tr>
<tr>
<td>Arm’s-Length</td>
<td>HURT</td>
<td>BREAK</td>
<td>HELP</td>
<td>HURT</td>
<td>MAKE</td>
<td>BREAK</td>
</tr>
<tr>
<td>Hierarchical Contracts</td>
<td>HELP</td>
<td>HELP</td>
<td>HURT</td>
<td>HELP</td>
<td>HELP</td>
<td>HELP</td>
</tr>
<tr>
<td>Vertical Integration</td>
<td>HELP</td>
<td>HELP</td>
<td>HURT</td>
<td>HELP</td>
<td>HURT</td>
<td>HELP</td>
</tr>
<tr>
<td>Co-optation</td>
<td>HURT</td>
<td>HURT</td>
<td>MAKE</td>
<td>MAKE</td>
<td>HELP</td>
<td>HURT</td>
</tr>
</tbody>
</table>
Architectural components defined in details in terms of inputs, outputs, control, and other relevant information.
Agent Interaction Protocol with AUML

The Checkout Dialogue

Plan Diagram for checking out
From TROPOS to JACK Java Agents

Partial JACK Implementation for checking out