ARKnowD: Agent-oriented Recipe for Knowledge Management System Development

Renata S. S. Guizzardi
Once upon a time...

Uniform:
White blouse and long jeans skirt
5 units, size M
# Knowledge Management Pitfalls

- “There is no time for filling in the system with new knowledge.” **Effort vs. Knowledge availability**

- “Oh, it’s too much effort to fill in the system, and then I can never find something useful in it when I need it.” **Detachment from daily working practices**

- “What if someone does something wrong with the knowledge I give away?” **Lack of trust**

- “Why should I share my knowledge if knowledge is power?” **Lack of motivation**
Developing Effective KM Solutions

overall organizational goals

Methodology

negotiating and reconciling these goals

stakeholders goals
communities of knowledge management practice and the knowledge spiral in autonomic physical meaningful artifacts and perturbation. This leads to the sharing of non-hierarchical knowledge in a social context. A social-historical perspective on constructivism is influenced by a social-historical perspective on knowledge management. This framework is anchored by the distributed knowledge management context and the dialogue and context in learning.
**ARKnowD: Agent-oriented Recipe for KM Systems Development**

- Agents as metaphors
  - Cognitive and social characteristics.
- Combining Existing work
  - Tropos for *Analysis* & AORML for *Design*
- MDA-inspired Transformation from Tropos to AORML
- Agent Ontology
  - Clarifying definition of applied concepts.
  - Assisting the transformation from Tropos to AORML.
  - Evaluating and adjusting ARKnowD’s notation.
Application Scenarios
## Development Activities

<table>
<thead>
<tr>
<th>Early Requirements</th>
<th>Late Requirements</th>
<th>Architectural Design</th>
<th>Detailed Design</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Early Requirements Diagram" /></td>
<td><img src="image2" alt="Late Requirements Diagram" /></td>
<td><img src="image3" alt="Architectural Design Diagram" /></td>
<td><img src="image4" alt="Detailed Design Diagram" /></td>
</tr>
</tbody>
</table>

**Transformation**

- **Artifact Model**: Artifacts modeled to create a document or artifact.
- **Provide Document**: Document is provided to artifacts.
- **Provide Explanation**: Explanation is provided to artifacts.
- **Question, Answer**: No available artifact.
Case Study:
Developing the KARe System
Goal dependencies between two domain agents
Delegating goals to artificial agent

- CoP
  - letting users keep control of assets while sharing
  - allowing members to ask and answer questions
  - knowing who knows what
  - providing members with personalized help

- KARe
KARe: Knowledgeable Agent For Recommendations

User Model:
- Expertise
- Reliability
- Trust
- Role
- Collaborative Level
- Availability
KARe's Architecture (1)
KARe's Architecture (2)
Transformation to AOR AD

Legend
- generalization/specialization relation
- composition relation
- ternary relation
- "/>" cardinality
- stereotype indicating agent type
- attribute
- derived attribute
- communication relation
AOR Interaction Pattern Diagram

<table>
<thead>
<tr>
<th>Event</th>
<th>Condition</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>ceive searchAnswer (?keyQuest, ?vectorC) FROM ?PeerAssistant</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IF</td>
<td>SimilarArtifact (?keyQuest, ?vectorC, ArtifactModel(?Document))</td>
<td></td>
</tr>
<tr>
<td>THEN</td>
<td>send provideDocument(?Document) TO ?PeerAssistant</td>
<td></td>
</tr>
<tr>
<td>ELSE IF</td>
<td>Similar(?keyQuest, ?vectorC, ArtifactModel(?Question))</td>
<td></td>
</tr>
<tr>
<td>THEN</td>
<td>send provideExplanation(?Question?Answer) TO ?PeerAssistant</td>
<td></td>
</tr>
<tr>
<td>ELSE</td>
<td>send noAvailableArtifact TO ?PeerAssistant</td>
<td></td>
</tr>
</tbody>
</table>
The condition part of the rule becomes code

```java
procedure answer(concVectA, peerQuest, questioner)
{
    //step 1: search the best matching concept for
    //the scope reduction
    projConceptVectorA := intersect(concVectA, indexB)
    for each (concept on the user B context) {
        s := similarity(currentConceptVectorB, projConceptVectorA)
        if (s > maxSimilarity) {
            bestConcept := currentConceptB
            maxSimilarity := s
        }
    }

    //step 2: search among the documents in the
    //bestConcept
    queryVector := createQueryVector(peerQuest, indexB)
    for each (document in bestConcept) {
        documentList.add(document, similarity(queryVector, documentVector))
    }
    documentList.sortBySimilarity()

    //step 3: send the answer back to
    //the questioner
    sendAnswer(documentList, questioner)
}
```
Determining the Interface between two Agents

Peer Assistant

- searchAnswer
- keyQuestion, vectConc

- provideDocument
  - docs

- provideExplanation
  - question, answer
  - noAvailableArtifact

- storeArtifact
  - concept, question, answer, eval

Artifact Manager
Conclusion

- We presented and exemplified the ARKnowD methodology.
- ARKnowD starts with an abstract model of the domain and then slowly adds design details
  - Smoothly transition between analysis and design
  - Materializing agent concepts into concrete elements of a system.
- Applies existing AOSE approaches (Tropos+AOR), combining them through an MDA-inspired transformation, guided by the use of the UFO ontology.
- Future work remains on
  - Testing ARKnowD in real scenarios
  - Refining UFO concepts and thus, the constructs of the methodology.