Goal-Oriented Requirements Engineering (GORE) : KAOS

Agent-Oriented Requirements Engineering (AORE) : i*
In philosophy, ontology is the study of being or existence. It seeks to describe or posit the basic categories and relationships of being or existence to define entities and types of entities within its framework. Ontology can be said to study conceptions of reality.

http://en.wikipedia.org/wiki/Ontology
KAOS


Background
- Developed in the early 90’s
- first major teleological requirements modeling language
- full tool support available
- has been applied to a number of industrial case studies

Two parts:
- Semi-formal goal structuring model
- Formal definitions for each entity in (linear) temporal logic
  - Liveness – Maintain: □(P→Q), Achieve: P→◊Q
  - Safety – Avoid: □(P→¬Q)

Approach
- Method focuses on goal elaboration:
  - define initial set of high level goals & objects they refer to
  - define initial set of agents and actions they are capable of

Then iteratively:
- refine goals using AND/OR decomposition
- identify obstacles to goals, and goal conflicts
- operationalize goals into constraints (or sw requirements) that can be assigned to individual agents
- refine & formalize definitions of objects & actions
- Goal refinement ends when every subgoal is realizable by some individual agent assigned to it, that is, expressible in terms of conditions that are monitorable and controllable by the agent.

©Lawrence Chung
A goal is a prescriptive statement of intent about some system (existing or to-be) whose satisfaction in general requires the cooperation of some of the agents forming that system (= software and environment).

Domain properties are descriptive statements about the environment, e.g., physical laws, organizational norms, etc.

A requirement is a realizable goal under responsibility of an agent in the software-to-be (expressed in terms of monitored and controlled objects);

An expectation is a realizable goal under responsibility of an agent in the environment (unlike requirements, expectations cannot be enforced by the software-to-be).

An operation is an input-output relation over objects.

The state of the system (software or environment) is defined by aggregation of the states of its objects. An object model is represented by a UML class diagram.

Although both optative, requirements are to be enforced by the software whereas assumptions/expectations can be enforced by agents in the environment only.
KAOS: Automated Train Control System Example


Eliciting new goals through WHY questions

Eliciting new goals through HOW questions

©Lawrence Chung

worst case stopping
KAOS: Automated Train Control System Example

Agent interfaces

Operations, Operationalizations

Goals refer to specific state transitions; Operations causing them are expressed by domain pre- and postconditions. For Maintain\([\text{SafeCmdMsg}]\):

Strengthen domain conditions so that the various goals linked to the operation are ensured. For goals assigned to software agents, this step produces requirements on the operations for the corresponding goals to be achieved.
KAOS: Automated Train Control System Example

[A. van Lamsweerde, "Requirements engineering in the year 00: a research perspective", Proc., 22nd ICSE'00, pp5-19. IEEE Computer Society Press]

Conflicting Goals:

Goal Maintain [CmdedSpeedCloseToPhysicalSpeed]
FormalDef ∀ tr: Train
  tr.Acc_CM ≥ 0
  ⇒ tr.Speed_CM ≤ tr.Speed + f(dist-to-obstacle)

Goal Maintain [CmdedSpeedAbove7mphOfPhysicalSpeed]
FormalDef ∀ tr: Train
  tr.Acc_CM ≥ 0 ⇒ tr.Speed_CM > tr.Speed + 7

Boundary condition for logical inconsistency

(∃ tr: Train) (tr.Acc_CM ≥ 0 ∧ f(dist-to-obstacle) ≤ 7)

Conflict resolution:
e.g., keep the safety goal and weaken the other

Goal Maintain [CmdedSpeedAbove7mphOfPhysicalSpeed]
FormalDef ∀ tr: Train
  tr.Acc_CM ≥ 0 ⇒ tr.Speed_CM > tr.Speed + 7
  ∨ f(dist-to-obstacle) ≤ 7

©Lawrence Chung
Goal-Oriented Requirements Engineering (GORE) : KAOS

Agent-Oriented Requirements Engineering (AORE) : i*
Is the system, to be implemented according to the SRS, going to be helpful or harmful?

How do we come up with the SRS?
Reengineering Work: Don't Automate, Obliterate

• Also see:
  https://www.utdallas.edu/~chung/SYSM6309/SYSM6309-Spring2012-Presentations/taraneh-Presentation1-case_study.pptx
Recall: Goal-oriented approach

But, whose goals are they?

Why should the system exist in the first place?

everybody’s?

factory supervisor’s?

Ford’s?

part-supplier’s?

car dealer’s?

Alice’s?

truck driver’s?

Order Processing System

developer’s?

loan officer’s?

The system exists to help agents achieve their goals?

Lawrence Chung
Background
- developed in the early 90’s
- provides a structure for asking ‘why’ questions in RE
- models the organizational context for information systems
- based on the notion of an “intentional actor”

Two parts to the model

Strategic dependency (SD) model - models relationships between the actors
- goal/softgoal dependency - an actor depends on another actor to attain a goal
- resource dependency - an actor needs a resource from another actor
- task dependency - an actor needs another actor to carry out a task

Strategic rationale (SR) model - models concerns and interests of the actors
- Shows task decompositions
- Shows means-ends links between tasks and goals
Can you represent this in UML?
Strategic Dependency Model
Strategic Rationale Model

“Functional” Alternatives
Appendix I
KAOS: Temporal Logic

**Prior’s Tense logic**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pp</strong></td>
<td>◇t(t&lt;now ∧ p(t))</td>
</tr>
<tr>
<td><strong>Fp</strong></td>
<td>◇t(now&lt;t ∧ p(t))</td>
</tr>
<tr>
<td><strong>Hp</strong></td>
<td>∀t(t&lt;now → p(t))</td>
</tr>
<tr>
<td><strong>Gp</strong></td>
<td>∀t(now&lt;t → p(t))</td>
</tr>
</tbody>
</table>

\(\Box: \text{necessary e.g., } \Box Fp\)
\(\Diamond: \text{possible e.g., } \Diamond Fp\)

Maintain, Avoid: “always” goals \(\Box (P \rightarrow Q)\), \(\Box (P \rightarrow \neg Q)\).
Achieve: “eventually” \(P \Rightarrow \Diamond Q\).

“⇒”: logical implication (the two below are the same)
\(P \Rightarrow Q\), \(\Box (P \rightarrow Q)\).

Extensions to Tense Logic

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Spq</strong></td>
<td>q has been true since a time when p was true</td>
</tr>
<tr>
<td><strong>Upq</strong></td>
<td>q will be true until a time when p is true</td>
</tr>
</tbody>
</table>

more in the module on Model Checking

©Lawrence Chung