Goal-Oriented Requirements Engineering (GORE) : KAOS

Agent-Oriented Requirements Engineering (AORE) : i*

How to elicit?

- Goal-Directed Strategy 2: Using more expressive power
  - Expressive Power *Revisited*

    "Propositional and predicate logic provide all the basic concepts needed for a systematic engineering design methodology"

    [C. A. Hoare, Mathematical Models for Computing Science, Oxford Univ. Rpt., Aug. 1994]

    - **modelling all the possible worlds**

      => what are they?

      => what are our conceptualization of them?

      these are philosophical questions!

    - **Ontology**

      on-tol-o-gy n. The branch of philosophy that deals with being

      what exists in reality?

      what are essential things in reality?

      entities, activities, constraints
      goals, agents, roles, rationales

    - **Epistemology**

      e-pis-te-mol-o-gy n., pl. -gies. 1. The division of philosophy that investigates the nature and origin of knowledge.

      2. A theory of the nature of knowledge.

      how do we organize them?

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

[Dardenne, van Lamsweerde and Fickas, "Goal-directed Requirements Acquisition," Science of Computer Programming, 20, pp. 3-50]

- **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: $\Box(P \rightarrow Q)$, Achieve: $P \rightarrow \Diamond Q$
    - Safety – Avoid: $\Box(P \rightarrow \neg Q)$
- **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.

$\Box$: necessary
e.g., $\Box$ Fp
$\Diamond$: possible
e.g., $\Diamond$ Fp

KAOS

- 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.

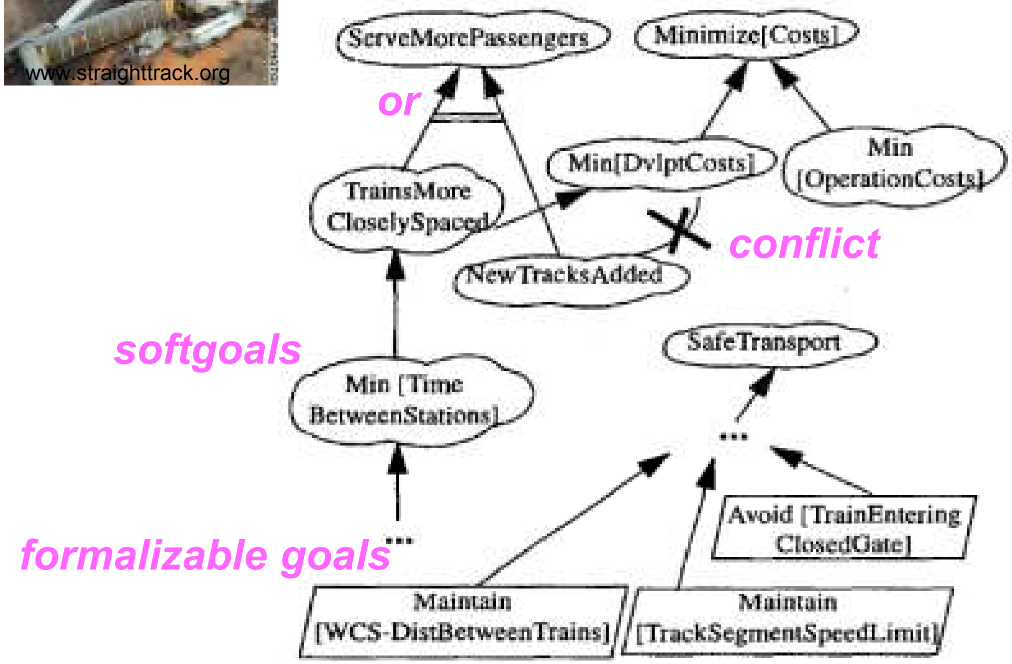
If ***R*** denotes the set of requirements,
As the set of assumptions *(**expectations**)*,
S the set of software specifications,
Ac the set of accuracy goals, and
G the set of goals,
the following satisfaction relations must hold:

$$S, Ac, D \models R \quad \text{with} \quad S, Ac, D \not\models \textbf{false}$$
$$R, As, D \models G \quad \text{with} \quad R, As, D \not\models \textbf{false}$$

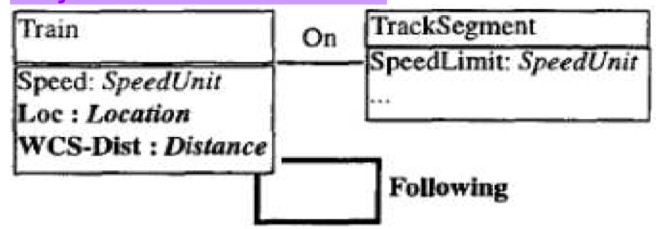
Although both optative, requirements are to be enforced by the software whereas assumptions/expectations can be enforced by agents in the environment only

KAOS: Automated Train Control System Example

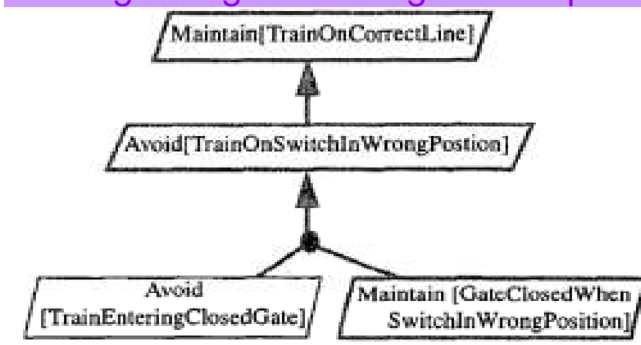
[A. van Lamsweerde, "Requirements engineering in the year 00: a research perspective", *Proc., 22nd ICSE'00*, pp5-19. IEEE Computer Society Press]



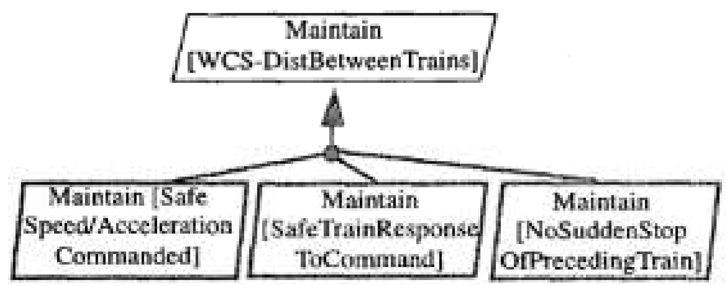
Object model



Eliciting new goals through WHY questions



Eliciting new goals through HOW questions



Goal Maintain[TrackSegmentSpeedLimit]

InformalDef *A train should stay below the maximum speed the track segment can handle.*

FormalDef ∀ tr: Train, s: TrackSegment :
On(tr, s) ⇒ tr.Speed ≤ s.SpeedLimit

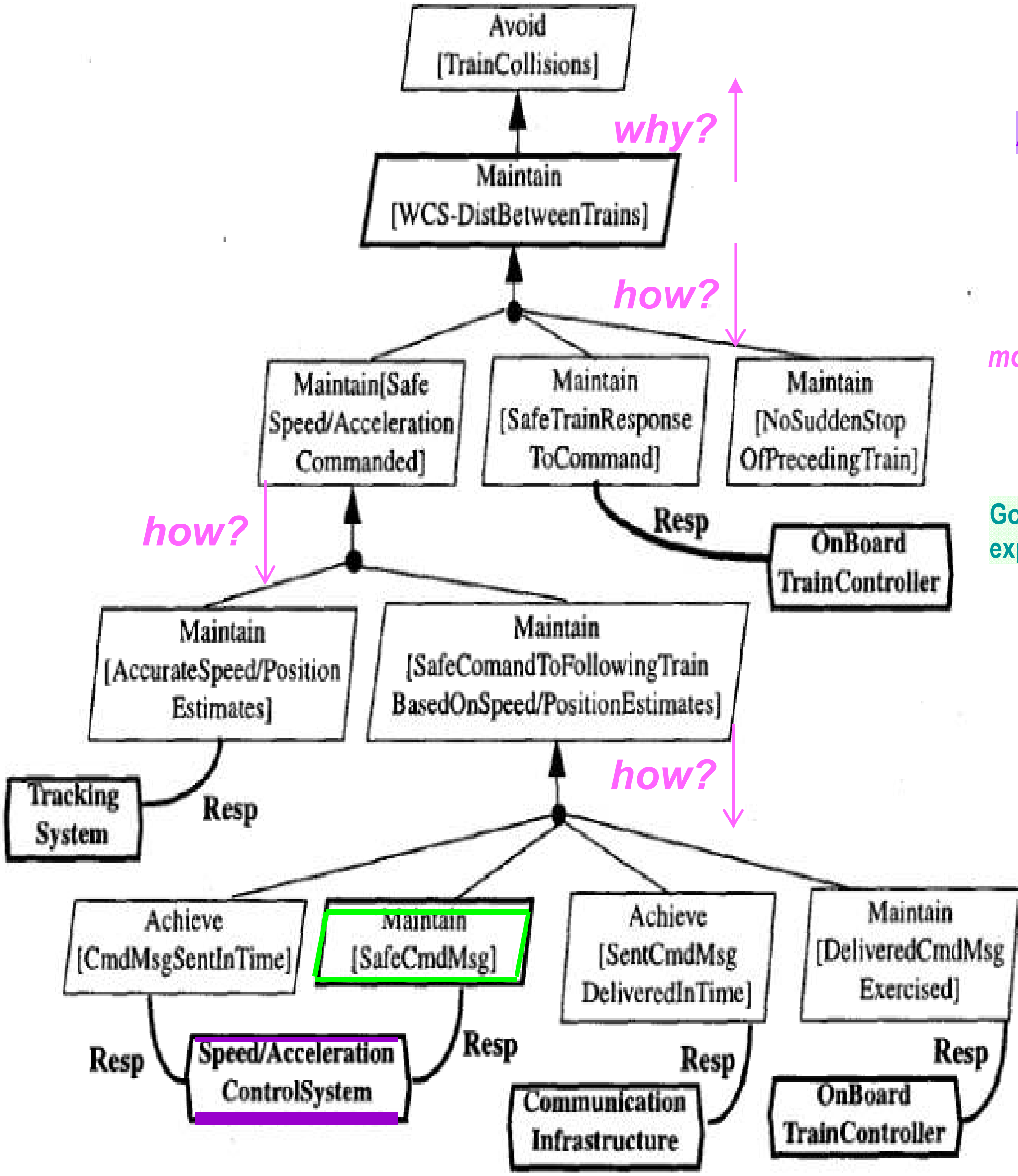
Goal Maintain[WCS-DistBetweenTrains]

InformalDef *A train should never get so close to a train in front so that if the train in front stops suddenly (e.g., derailment) the next train would hit it.*

FormalDef ∀ tr1, tr2: Train :
Following(tr1, tr2) ⇒ tr1.Loc - tr2.Loc > tr1.WCS-Dist

worst case stopping

KAOS: Automated Train Control System Example



Goal Maintain[SafeCmdMsg] *monitored*

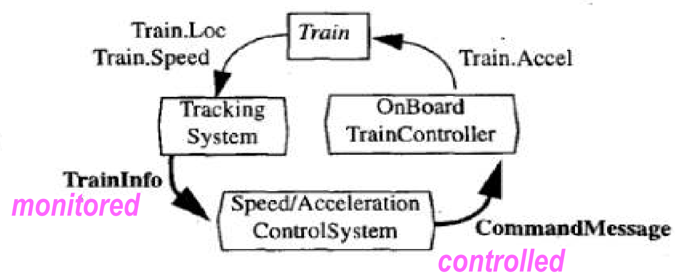
FormalDef $\forall$ cm: *CommandMessage*, ti1, ti2: *TrainInfo*

cm.Sent $\wedge$ cm.TrainID = ti1.TrainID $\wedge$ FollowingInfo (ti1, ti2)

$\Rightarrow$ cm.Accel $\leq$ F (ti1, ti2) $\wedge$ cm.Speed > G (ti1)

controlled

Agent interfaces



Operations, Operationalizations

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

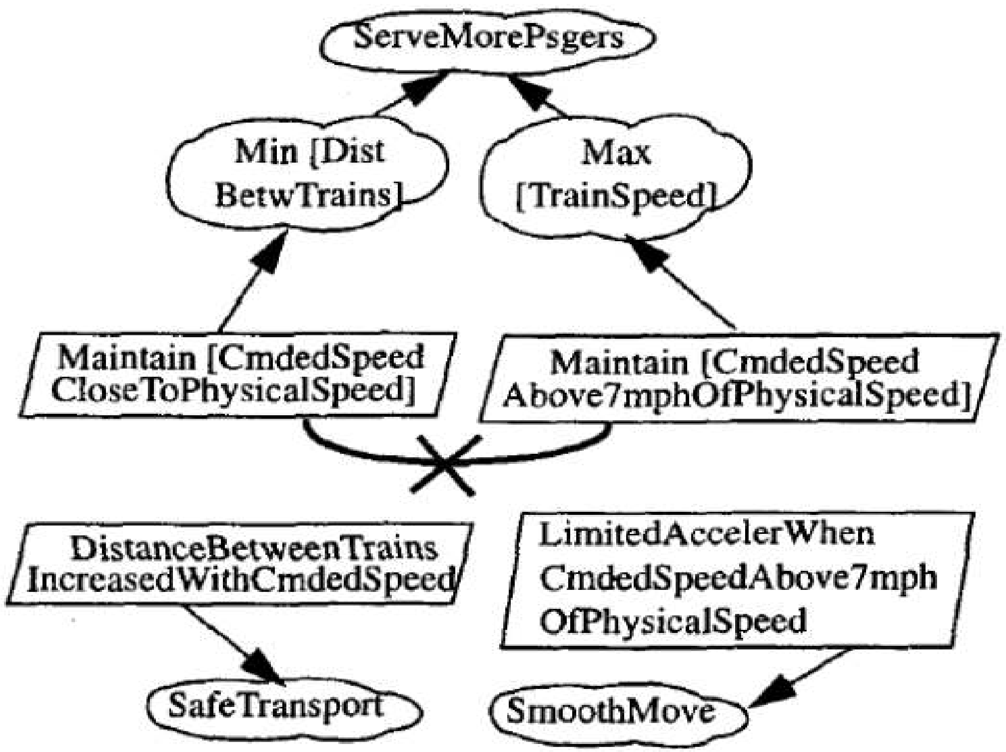
Operation SendCommandMessage
Input Train {**arg** tr}
Output ComandMessage {**res** cm}
DomPre $\neg$ cm.Sent
DomPost cm.Sent $\wedge$ cm.TrainID = tr.ID

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.

Operation SendCommandMessage
Input Train {**arg** tr}, TrainInfo; **Output** ComandMsg {**res** cm}
DomPre ... ; **DomPost** ...
ReqPost for SafeCmdMsg:
Tracking (ti1, tr) $\wedge$ Following (ti1, ti2)
$\rightarrow$ cm.Acc $\leq$ F (ti1, ti2) $\wedge$ cm.Speed > G (ti1)
ReqTrig for CmdMsgSentInTime:
$\blacksquare_{\leq 0.5\ sec}\ \neg\ \exists$ cm2: CommandMessage:
cm2.Sent $\wedge$ cm2.TrainID = tr.ID

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]



Conflicts

Conflicting Goals:

Goal Maintain [CmdedSpeedCloseToPhysicalSpeed]
FormalDef $\forall$ tr: Train
$\text{tr.Acc}_{CM} \geq 0$
$\Rightarrow \text{tr.Speed}_{CM} \leq \text{tr.Speed} + f(\text{dist-to-obstacle})$

Goal Maintain [CmdedSpeedAbove7mphOfPhysicalSpeed]
FormalDef $\forall$ tr: Train
$\text{tr.Acc}_{CM} \geq 0 \Rightarrow \text{tr.Speed}_{CM} > \text{tr.Speed} + 7$

Boundary condition for logical inconsistency

$\Diamond\, (\exists\, \text{tr: Train})\ (\text{tr.Acc}_{CM} \geq 0 \wedge f(\text{dist-to-obstacle}) \leq 7)$

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

Goal Maintain [CmdedSpeedAbove7mphOfPhysicalSpeed]
FormalDef $\forall$ tr: Train
$\text{tr.Acc}_{CM} \geq 0 \Rightarrow \text{tr.Speed}_{CM} > \text{tr.Speed} + 7$
$\vee\ f(\textit{dist-to-obstacle}) \leq 7$

Goal-Oriented Requirements Engineering (GORE) : KAOS

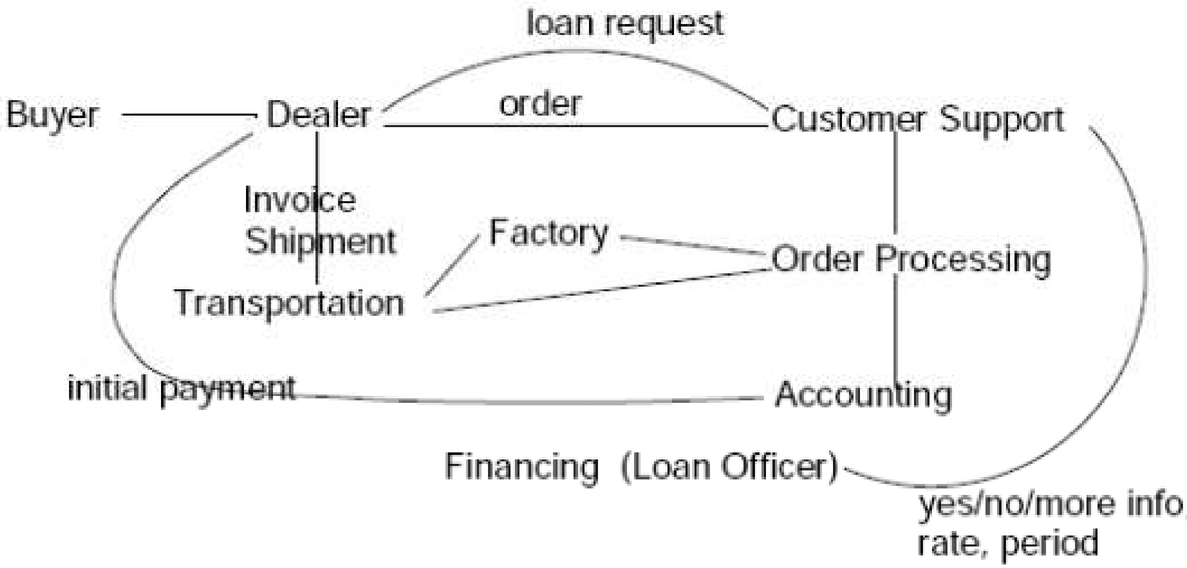
Agent-Oriented Requirements Engineering (AORE) : i*

Reengineering Work: Don't Automate, Obliterate

Why Enterprise Modelling?

M. Hammer, Reengineering Work: Don't Automate, Obliterate, Harvard Business Review, July-August 1990]

Order Processing



Person power: several hundreds
Processing time: long enough to invite complaints
Errors: annoying and significant

Why should a system be created? goals/objectives
Whose work would the system support? agents, activities/operations
How would the info. in the system help run the enterprise? information flow
What's the role of the system for cross-functional activities? workflow

Is the system, to be implemented according to the SRS, going to be helpful or harmful?
How do we come up with the SRS?

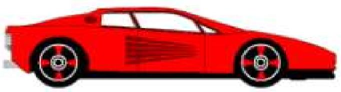
Reengineering Work: Don't Automate, Obliterate

- Also see:

https://www.utdallas.edu/~chung/SYSM6309/SYSM6309-Spring2012-Presentations/taraneh-Presentation1-case study.pptx

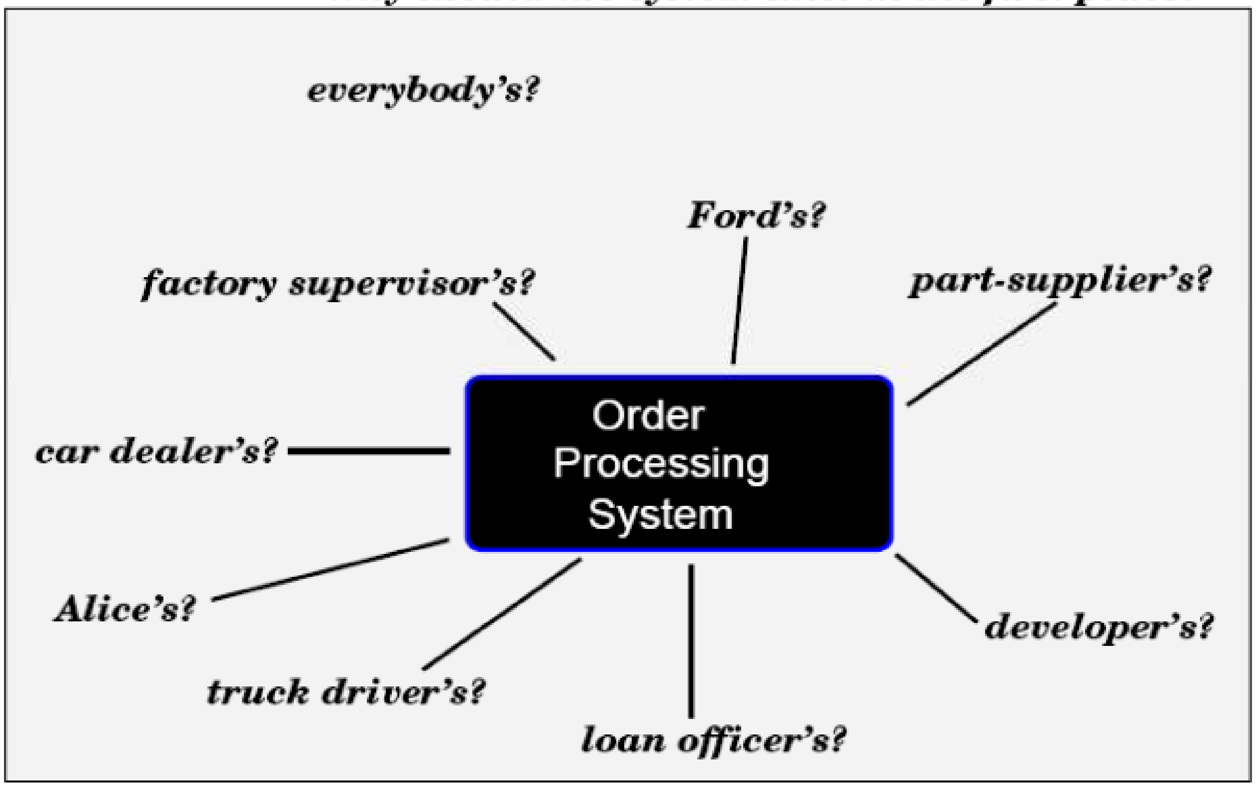
Agent-Oriented Approach to Enterprise Modelling

- Recall: Goal-oriented approach



But, whose goals are they?

Why should the system exist in the first place?



The system exists to help agents achieve their goals?

i*

- **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**

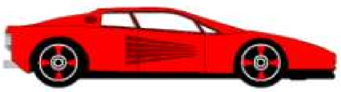
  - Strategic dependency (SD) model - models relationships between the actors
    - **goal/softgoal dependency** - an actor depends on another actor to attain a goal
    - **resource dependency** - an actor needs a resource from another actor
    - **task dependency** - an actor needs another actor to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

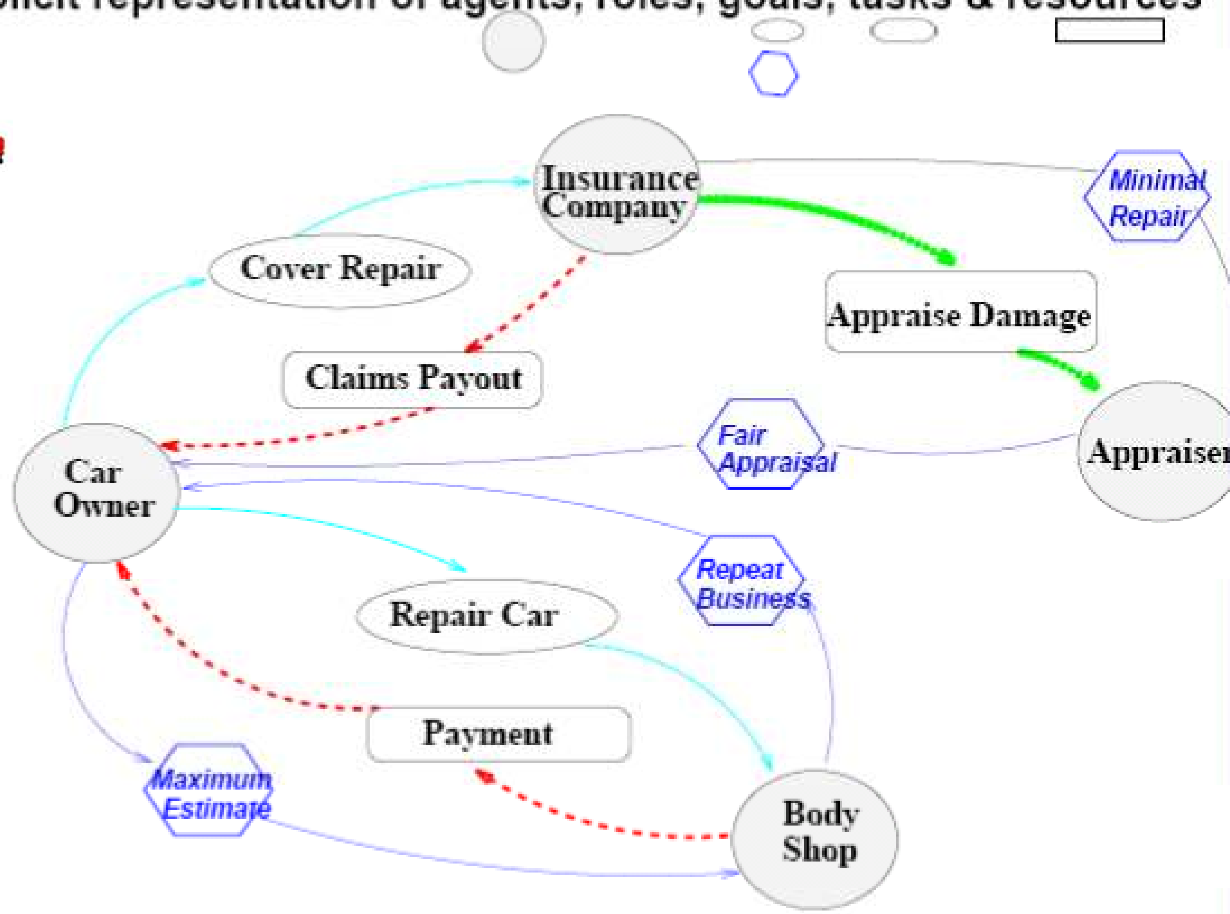
  - Strategic rationale (SR) model - models concerns and interests of the actors
    - Shows task decompositions
    - Shows means-ends links between tasks and goals

From ERD to ARD

[Yu & Mylopoulos, ICSE, '94]

- explicit representation of agents, roles, goals, tasks & resources

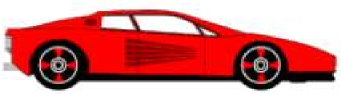


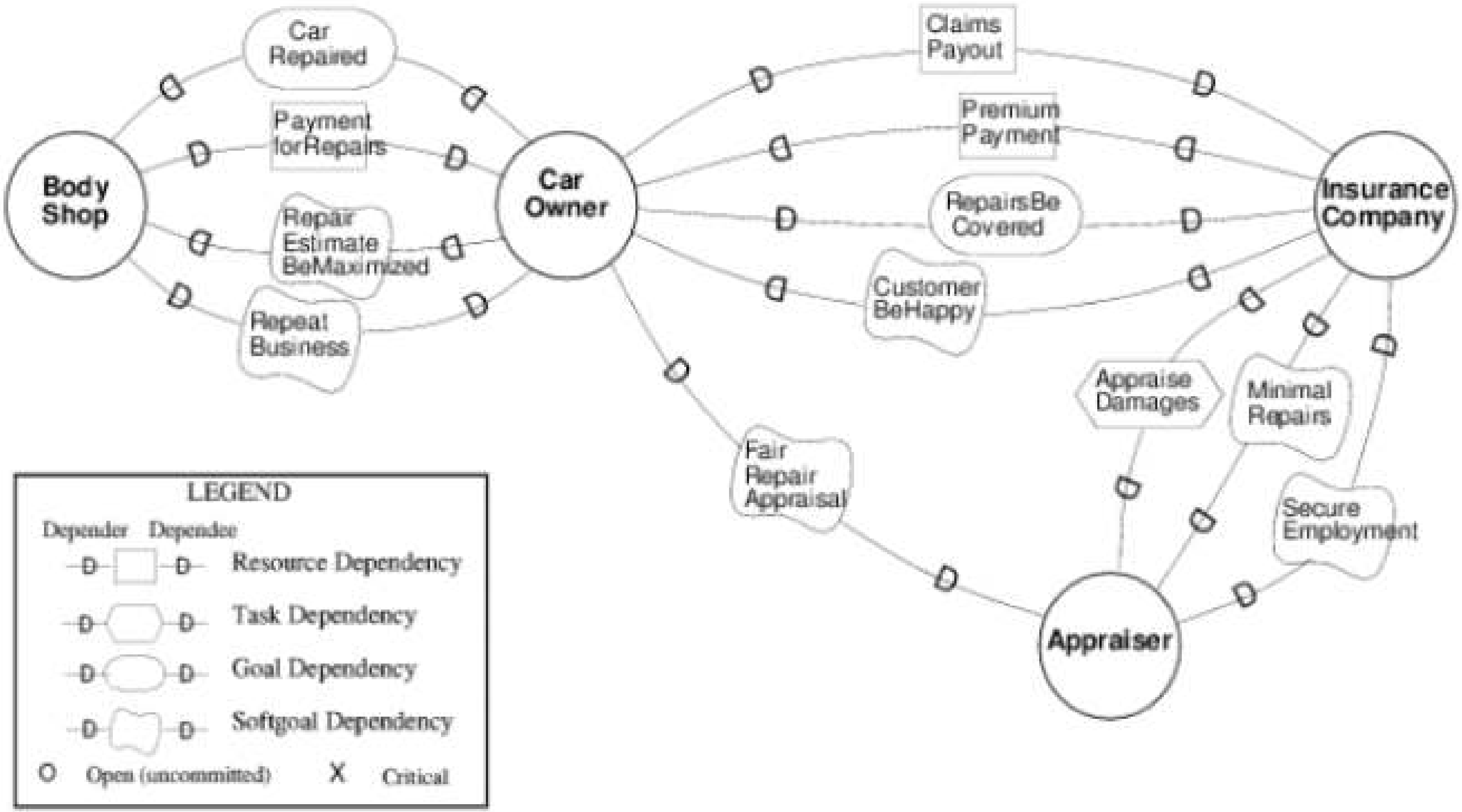


- *Why does it take so long to have a claim settled after an automobile accident?*
- *Why does the company hire appraisers to assess damages?*
- *What should a software agent do and why?*

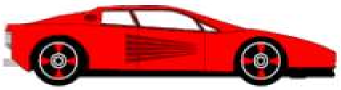
Can you represent this in UML?

Strategic Dependency Model

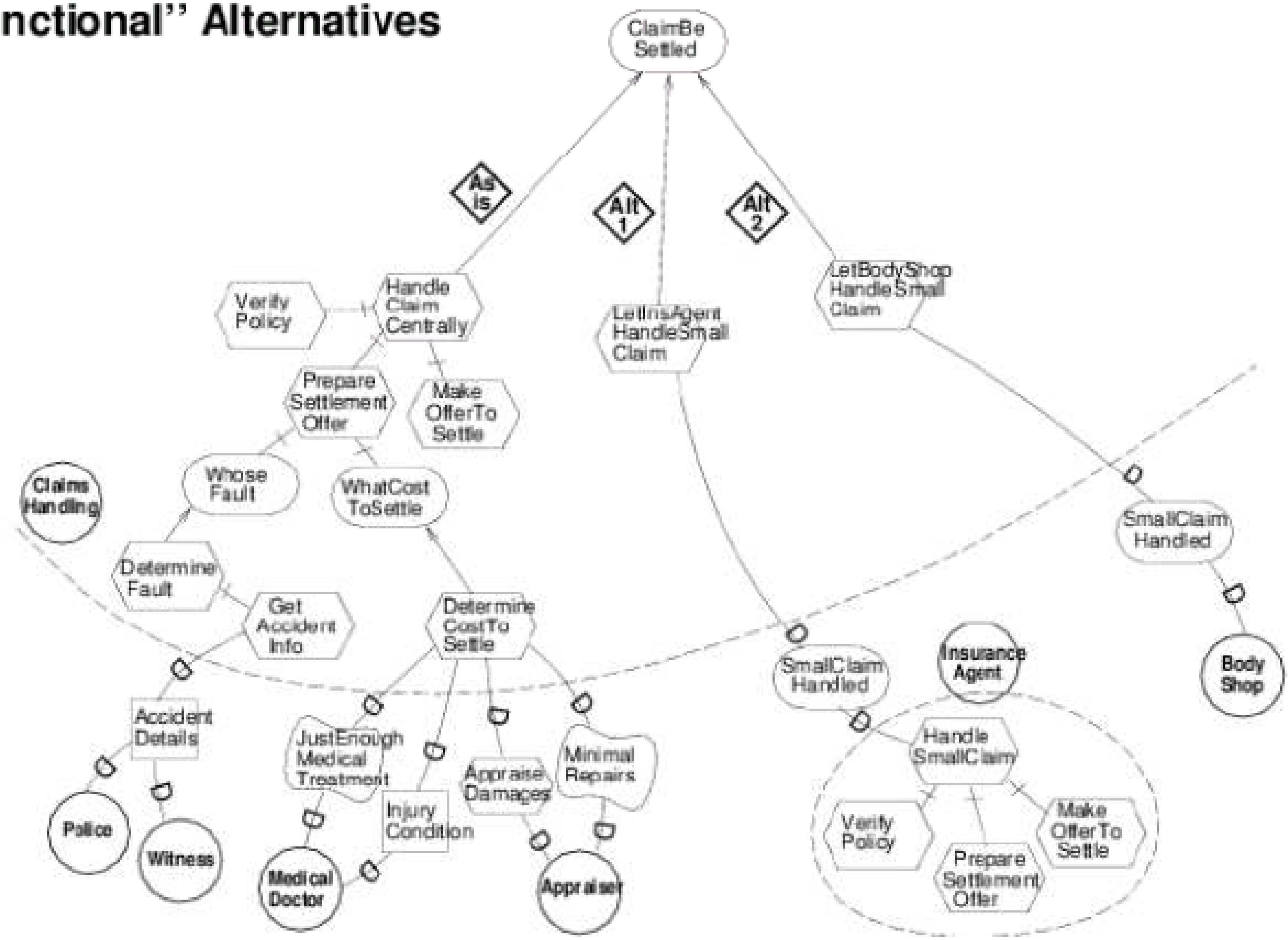




Strategic Rationale Model



"Functional" Alternatives



Appendix I

KAOS: Temporal Logic

Prior's Tense logic

Pp	**$\Diamond t(t<now \wedge p(t))$**
Fp	**$\Diamond t(now<t \wedge p(t))$**
Hp	**$\forall t(t<now \rightarrow p(t))$**
Gp	**$\forall t(now<t \rightarrow p(t))$**

$\Box$: necessary
e.g., $\Box$ Fp
$\Diamond$: possible
e.g., $\Diamond$ Fp

Extensions to Tense Logic

Spq	q has been true since a time when p was true
Upq	q will be true until a time when p is true

more in the module on Model Checking

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)
$P \Rightarrow Q$ $\Box(P \rightarrow Q)$