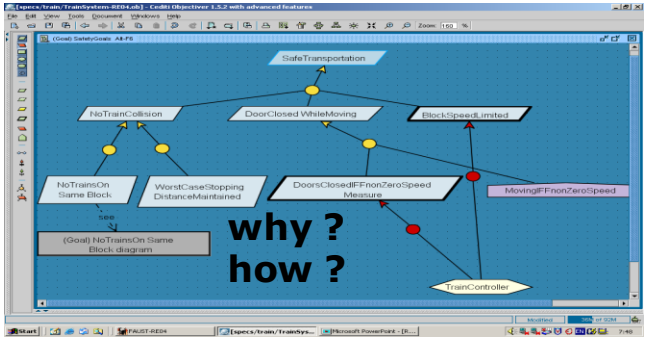
Modeling System Agents and Responsibilities

Mariano Ceccato

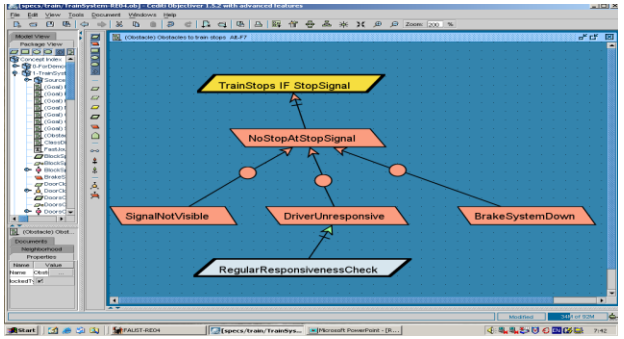
mariano.ceccato@univr.it

Building models for RE

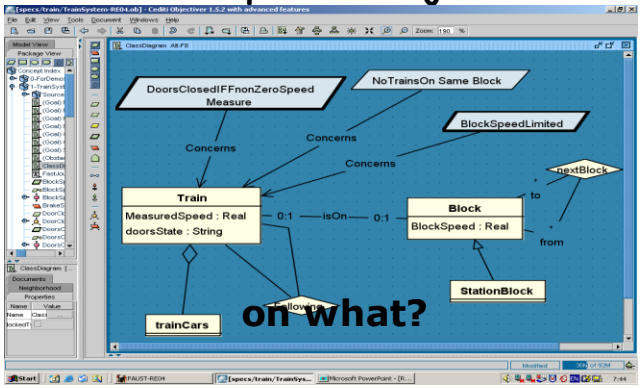
Goals



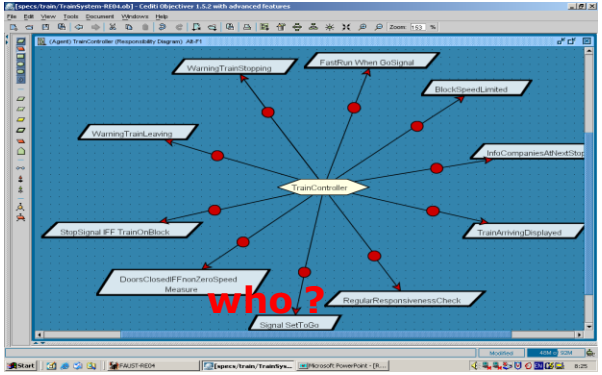
Risks



Conceptual objects



Agents



The agent model

- **Responsibility** view of the system being modeled
  - **who** is doing what, and why
- Different perspectives, different diagrams
  - agent capabilities, responsibilities, interfaces
  - dependencies among agents
- Multiple uses
  - showing distribution of responsibilities within system
  - load analysis
  - system scope & configuration, boundary software/environment
  - heuristics for responsibility assignment
  - vulnerability analysis
  - input to architectural design

Modeling system agents: outline

- What we know about agents so far
- Characterizing system agents
  - capabilities
  - responsibilities
  - operation performers
  - wishes & beliefs
  - dependencies
- Representing agent models
  - agent diagram, context diagram, dependency diagram
- Refinement of abstract agents
- Building agent models: heuristics & derivation rules

What we know about agents so far

- Active objects: control behaviors in system *as-is* or *to-be*
  - "processors" of operations
- Responsible for goal satisfaction
  - role rather than individual
  - assigned to leaf goals (requirements, expectations)
  - must restrict system behaviors accordingly
- May run concurrently with others
- Different categories
  - software-to-be
  - environment: people, devices, legacy/foreign software

Characterizing system agents

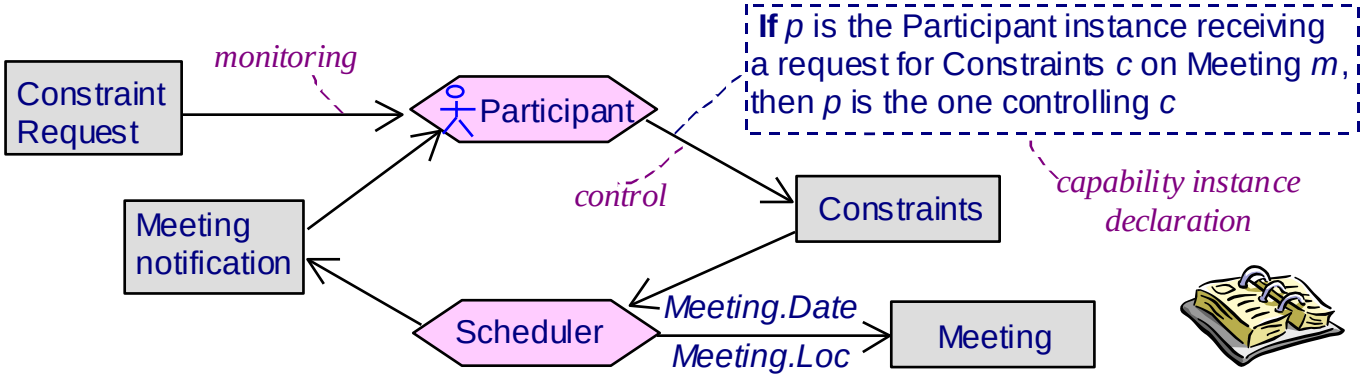
- **Def**: condition for individual to be currently instance of this agent
- Attributes/associations, Dom Invariants: **in** object model
- **Category**: software or environment agent 
- **Capabilities**: what the agent can monitor and control
  - monitoring/control links to object model, cf next slides
- **Responsibility**: links to goal model
- **Performance**: links to operation model
- **Dependency** links to other agents for goal satisfaction
- **Wishes** (for responsibility assignment heuristics)
- **Knowledge** and **beliefs** (for obstacle analysis, security analysis)

Agent capabilities

- Ability to monitor or control items declared in object model
  - attributes/associations get instantiated as **state variables** monitorable/controllable by agent instances (cf. 4-var model)
  - which agent instance monitors/controls attribute/association of which object instance: specified in instance declaration annotating link

- An agent **monitors** (resp. **controls**) an object attribute if its instances can get (resp. set) values of this attribute
  - it **monitors** (resp. **controls**) an association if its instances can get (resp. create or delete) association instances
  - it **monitors** (resp. **controls**) an object if it monitors (resp. controls) all object's attributes & associations

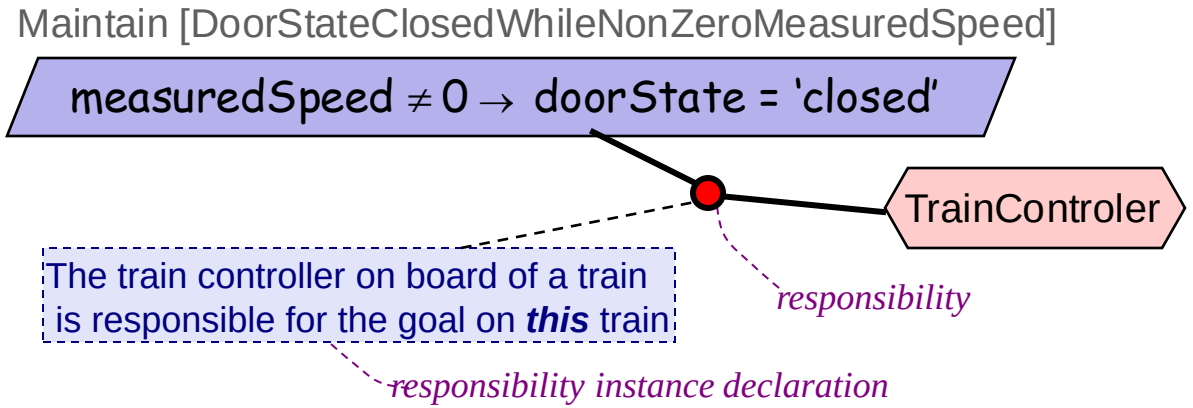
Agent capabilities



- Capabilities define agent interfaces
  - an agent monitors a state variable controlled by another
- Higher-level capabilities sometimes convenient
  - an agent **monitors** (resp. **controls**) a condition if its instances can evaluate it (resp. make it true/false)
- A variable may be controlled by at most one agent
  - to avoid interferences among concurrent agents

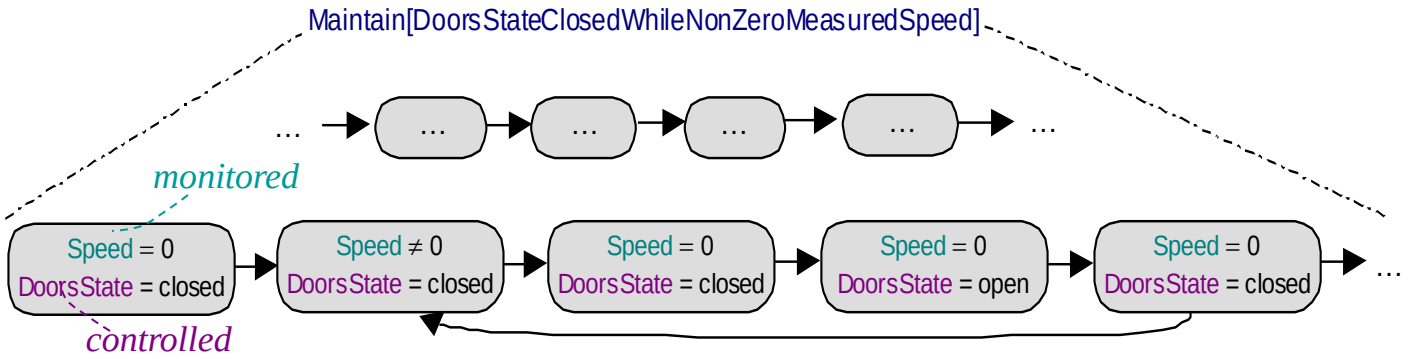
Agent responsibilities

- An agent is **responsible** for a goal if its instances are the only ones required to restrict behaviors to satisfy the goal
  - through setting of their controlled variables
  - which agent instance is responsible for the goal on which object instance: specified in **instance declaration** annotating link



Agent capabilities & goal realizability

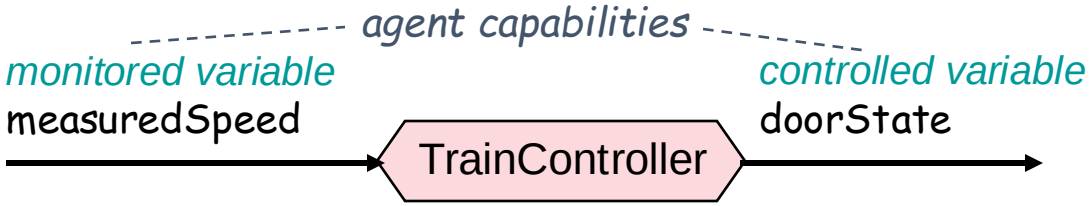
- Responsibility assignment is subject to agent capabilities
  - the goal must be realizable by the agent in view of what the agent can monitor and control
  - roughly: we can define a set of sequences of state transitions on the agent's monitored/controlled variables that coincides with the set of behaviors prescribed by the goal



Causes of goal unrealizability by agents

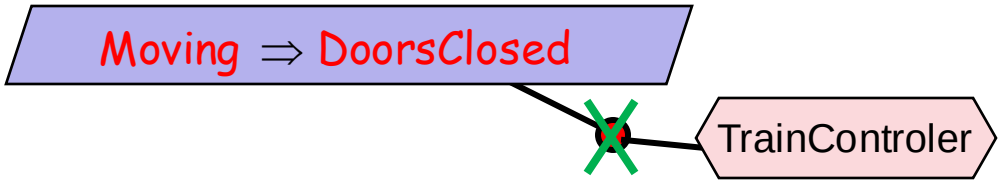
- **Lack of monitorability** of state variables to be evaluated in assigned goals
- **Lack of controllability** of state variables to be constrained in assigned goals
- State variables to be evaluated in future states
- Goal unsatisfiability under certain conditions
- Unbounded achievement of assigned *Achieve* goals
  - target can be indefinitely postponed

Agent capabilities & goal realizability: examples



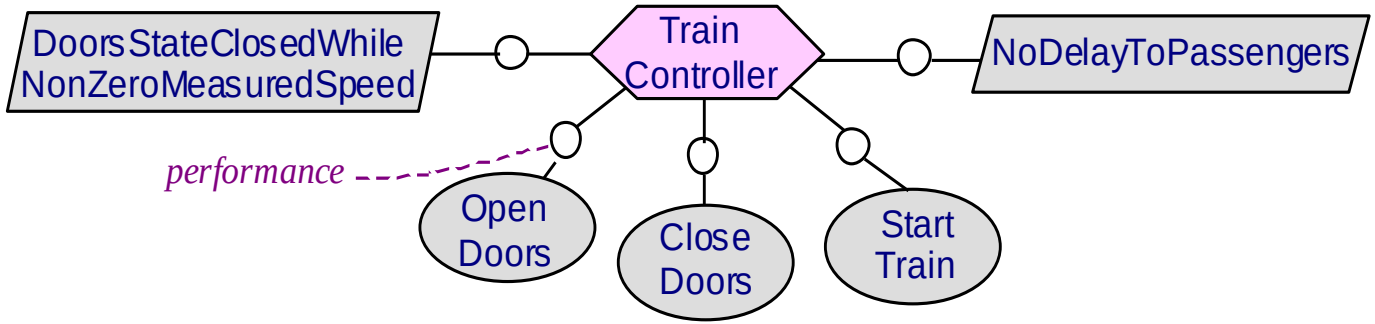
Ex 1: Realizable by TrainController

Ex 2: **Not** realizable by TrainController



Agents as operation performers

- An agent **performs** an operation if the applications of this operation are activated by instances of this agent
  - means for getting/setting the agent's monitored/controlled variables
  - under restricted conditions so as to satisfy assigned goals: permissions, obligations specified in operation model (cf. next lecture)
  - which agent instance activates which operation application: specified in **instance declaration** annotating *Performance link*



Agent wishes

- A human agent **wishes** a goal if its instances would like the goal to be satisfied
  - e.g. *Wish* link between Patron and LongLoanPeriods
    Participant and MinimumInteraction

- Optional agent feature used for
  - Goal elicitation: *goals wished by this human agent?*
  - Responsibility assignment:
    - Avoid assignments of goals conflicting with wished goals
      e.g. no assignment of ReturnEncoded to Patron

    - Favor assignments of security goals to trustworthy agents: wishing them

Agent belief and knowledge

- Agents may be equipped with a **local memory** maintaining facts about their environment
  - domain properties should state how facts get in and out
- An agent **believes** a fact *F* if *F* is in *ag* local memory
- An agent **knows** a fact *F* if *ag* believes *F* and *F* actually holds
- Optional agent feature used for
  - obstacle analysis: **wrong belief** obstacles are common

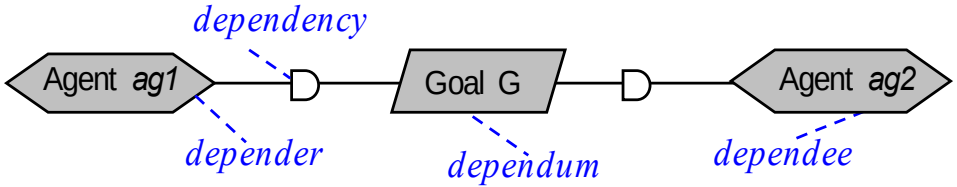
    ag believes *F* and *F* does not hold

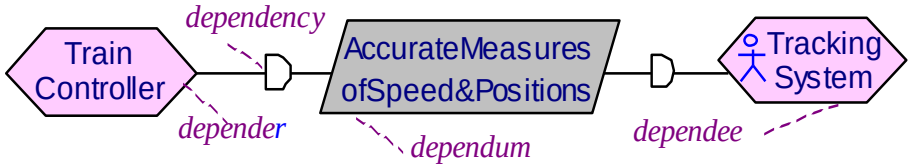
    e.g. $\text{Belief}_{\text{Participant}}$ (m.Date = d) **and** m.Date ≠ d for some meeting m

    
  - security analysis: goals on what agents may **not** know
    - no knowledge of sensitive facts

Agent dependencies

- An agent ag1 **depends on** another agent ag2 **for** a goal G under responsibility of ag2 **if** ag2's failure to get G satisfied can result in ag1's failure to get one of its assigned goals satisfied
  - dependee ag2 is not responsible for ag1's goals & their failure
  - goal failure propagates
    - up in refinement trees
    - backwards through dependency chains
- Optional agent feature used for
  - vulnerability analysis along dependency chains
    => agent model restructuring, countermeasures
  - capturing strategic dependencies among organizational agents

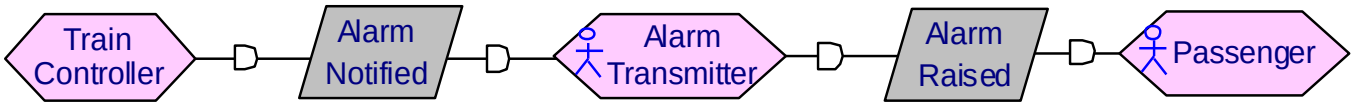




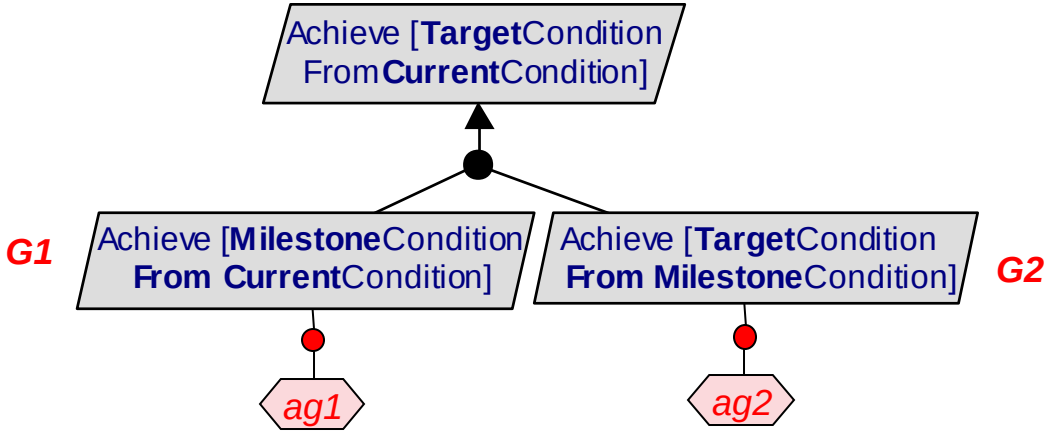
Dependencies may propagate along chains

- If ag1 depends on ag2 for G2,
  ag2 depends on ag3 for G3,
  G2 is among ag2's failing goals when G3 fails;
  then ag1 depends on ag3 for G3

- Critical dependency chains should be detected and broken
  - alternative goal refinements or assignments with fewer, less critical dependencies
  - dependency mitigation goals
  - basis for *vulnerability analysis*



A common dependency pattern: milestone-based dependency

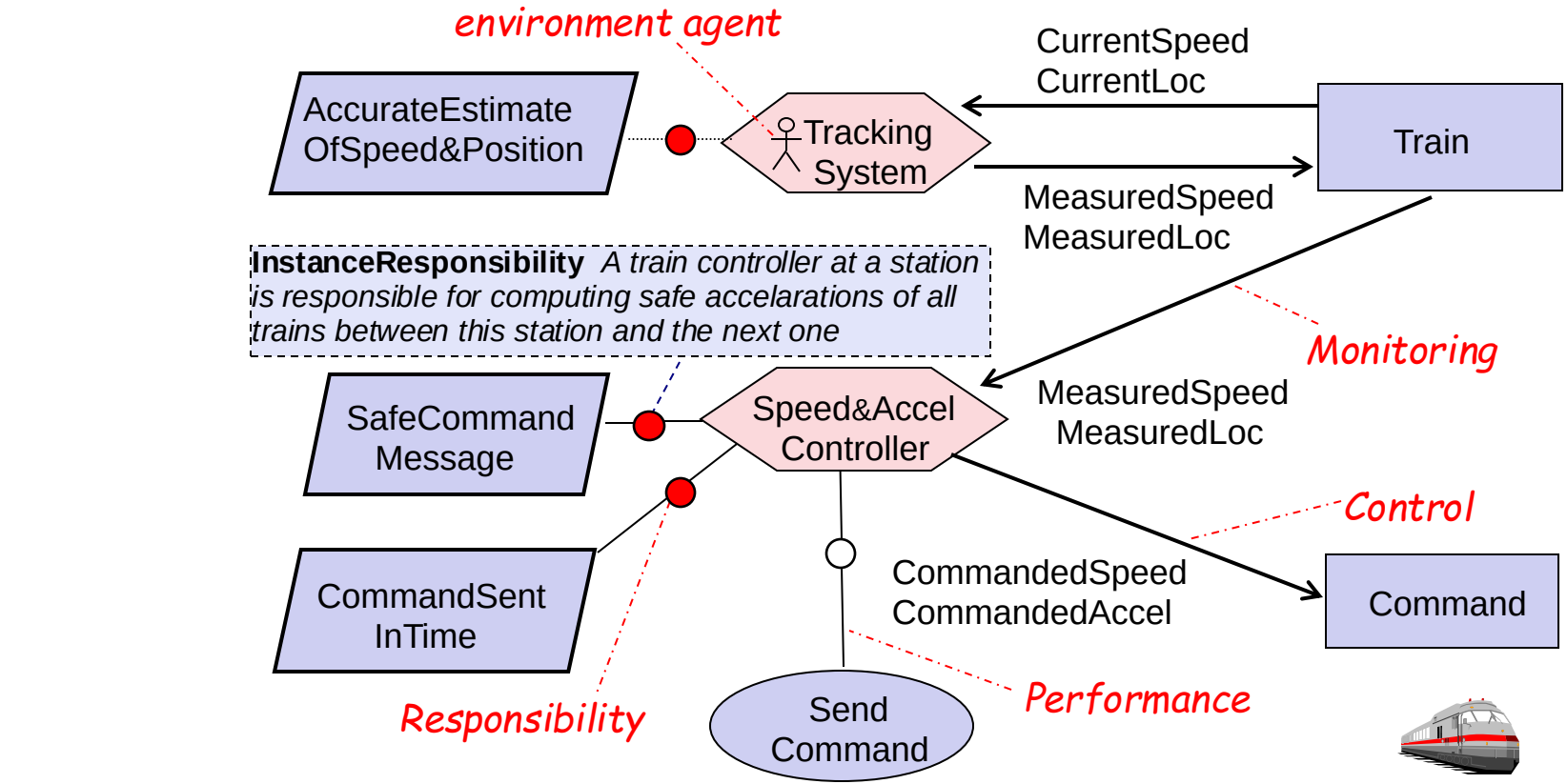


- **If** ag2 can fail to establish **Target**Condition
  when ag1 fails to establish **Milestone**Condition
- **then** ag2 depends on ag1 for G1

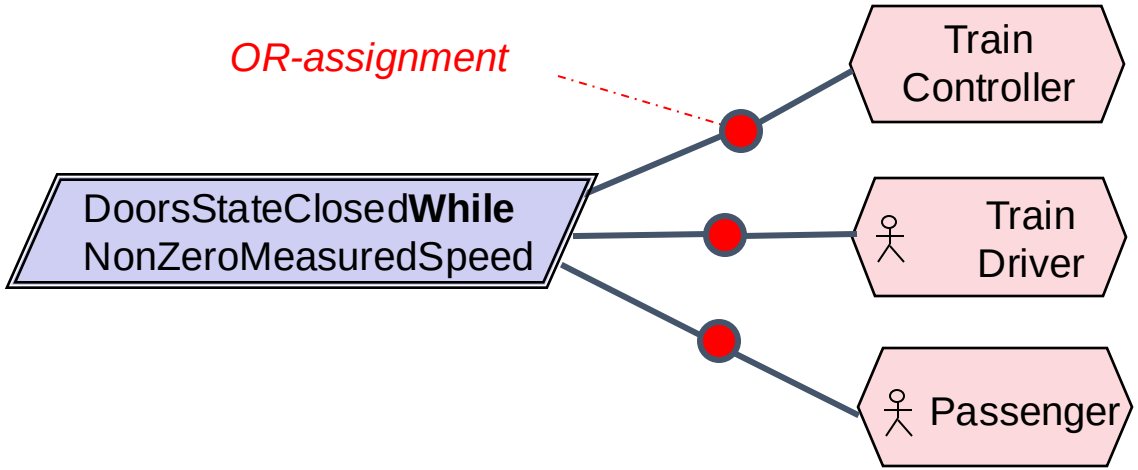
Modeling system agents: outline

- What we know about agents so far
- Characterizing system agents
  - capabilities
  - responsibilities
  - operation performers
  - wishes & beliefs
  - dependencies
- Representing agent models
  - agent diagram, context diagram, dependency diagram
- Refinement of abstract agents
- Building agent models: heuristics and derivation rules

An agent diagram shows agents with their capabilities, responsibilities & operations

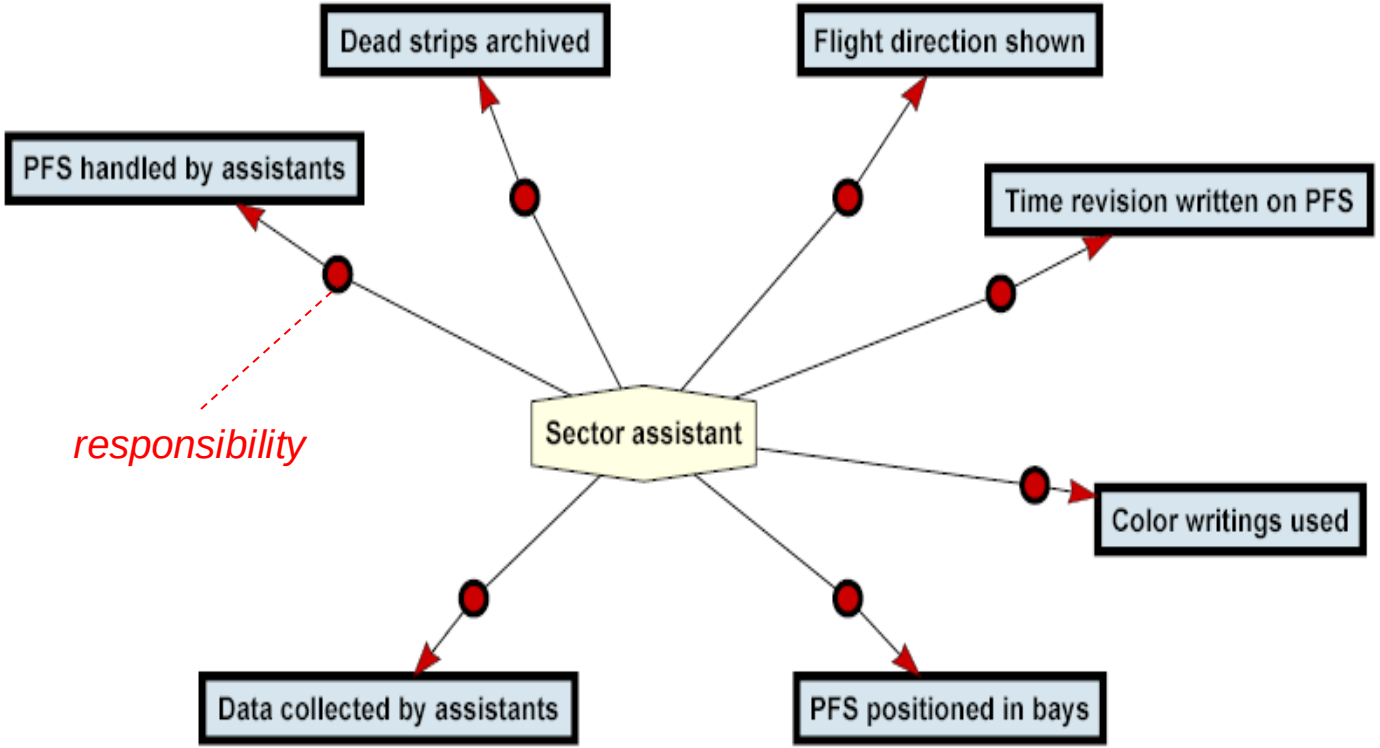


Alternative agent assignments define alternative software-environment boundaries



- OR-assignment => alternative options => alternative system proposals
  - more or less automation
- Captured in goal model; selected assignment shown in agent model

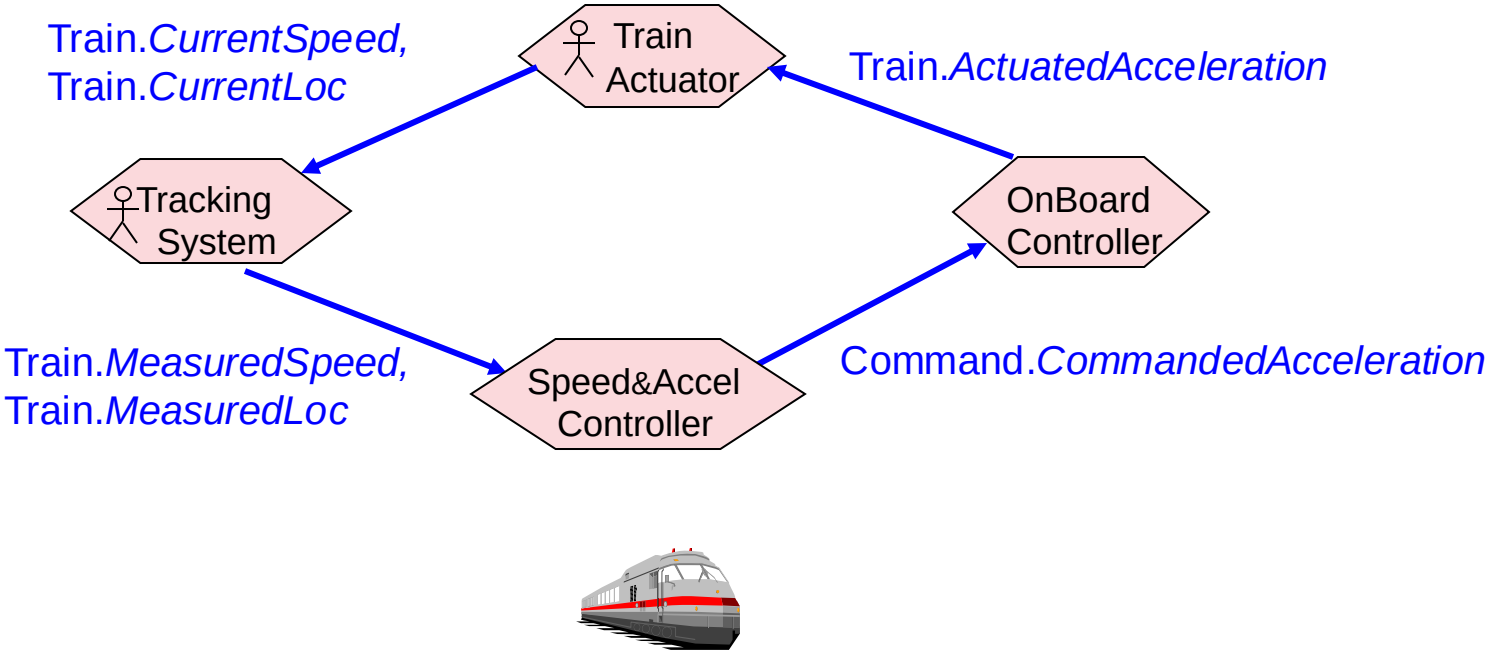
Load analysis from query on agent model for air traffic control



A context diagram shows agents and their interfaces

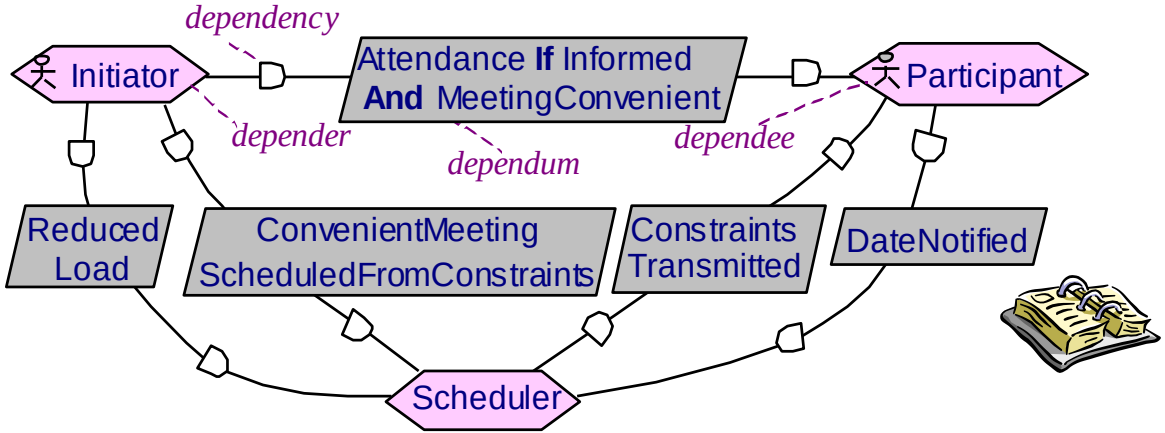
- Partial view: focus on capabilities & interfaces
  - interface = **monitored/controlled** state variables
    (attrib/assoc from object model)
  - **link (ag1, ag2)** with label **var** generated from agent diagram iff
    var is controlled by ag1, monitored by ag2

Context diagram: example



A dependency diagram shows agents and their dependencies

- Dependencies among agent pairs for goals to be satisfied
  - including dependency chains
- Complementary view to agent/context diagrams
  - for vulnerability analysis: goal failure propagation
  - for modeling organizational components of the system
- Cf. i* diagrams

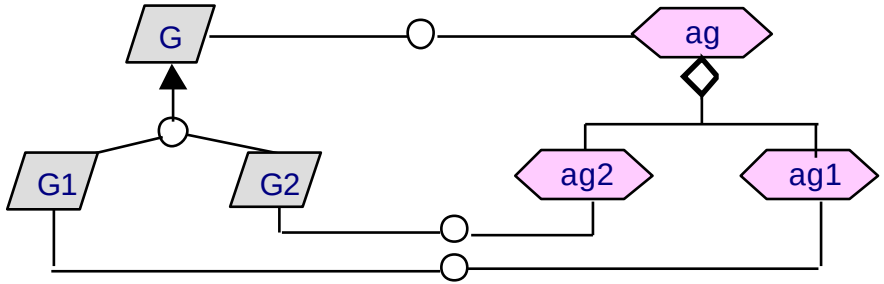


Modeling system agents: outline

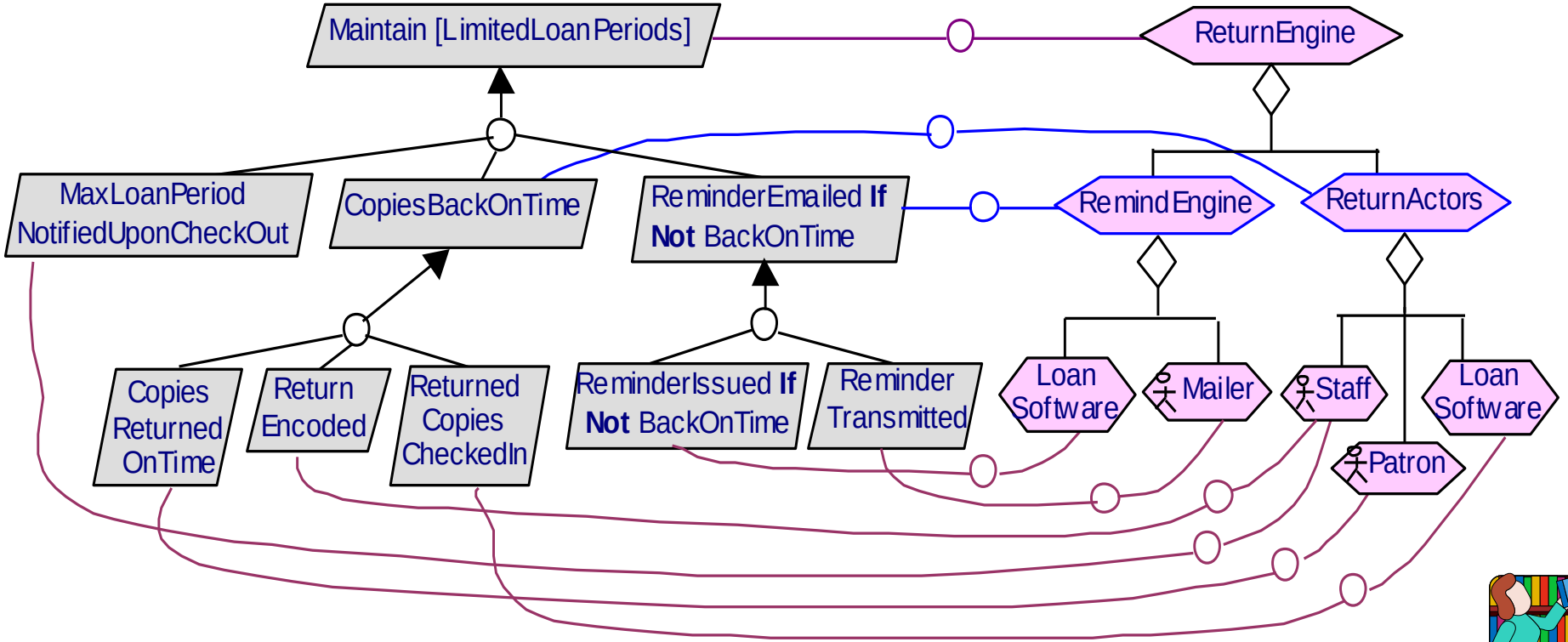
- What we know about agents so far
- Characterizing system agents
  - capabilities
  - responsibilities
  - operation performers
  - wishes & beliefs
  - dependencies
- Representing agent models
  - agent diagram, context diagram, dependency diagram
- Refinement of abstract agents
- Building agent models: heuristics and derivation rules

Agent refinement

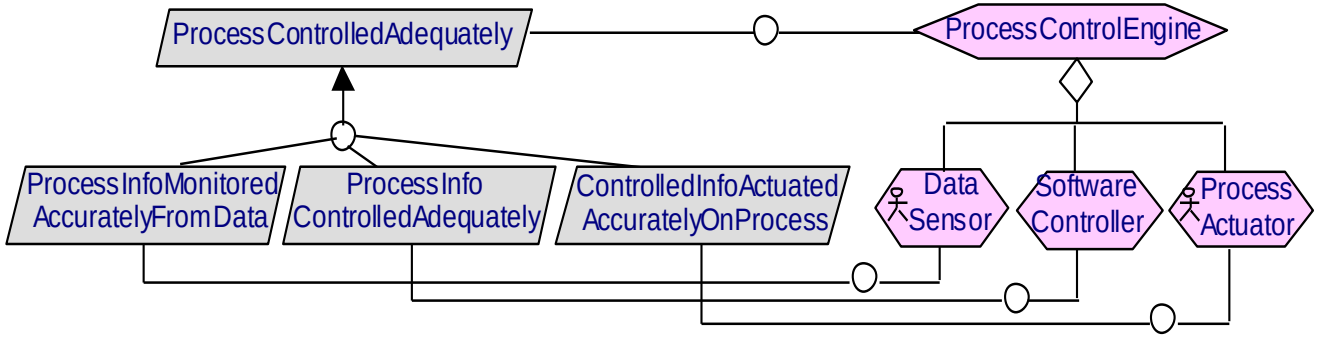
- Agents may be defined as aggregations of finer-grained agents
  - like any object in object model, cf. last lecture
- Supports incremental refinement of responsibilities
  - coarse-grained goal assigned to coarse-grained agent, then subgoals assigned to finer-grained agents
- Coarse-grained agent may be
  - environment agent e.g. organizational department -> units -> operators
  - hybrid: environment agent + software-to-be
  - for software-to-be agents: deferred to architectural design

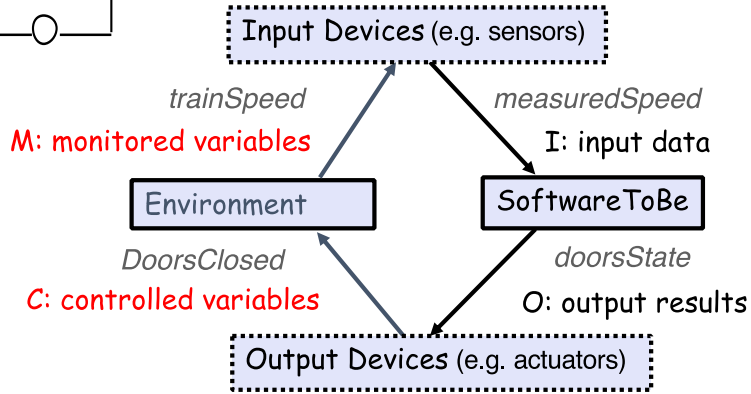


Goal-agent co-refinement: example



A goal-agent co-refinement pattern in process control





- Cf. 4-variable model (Intro lecture), problem frame for control systems (specification lecture)

Modeling system agents: outline

- What we know about agents so far
- Characterizing system agents
  - capabilities
  - responsibilities
  - operation performers
  - wishes & beliefs
  - dependencies
- Representing agent models
  - agent diagram, context diagram, dependency diagram
- Refinement of abstract agents
- Building agent models: heuristics and derivation rules

Heuristics for building agent diagrams

- For agent identification
  - active objects **Concerned by** this goal?
    their monitoring & control capabilities in object model?
    e.g. Achieve [ResourceRequestSatisfied] => ~~Book~~ ~~Requesting~~ Patron
  - possible enforcers of this goal? their capabilities?
    e.g. Avoid [CopiesStolen] => Staff or AntiTheftDevice
  - human system agents **Wishing** this goal? their capabilities?
    e.g. Maintain [AccurateBookClassification] => ResearchStaff
  - possible source (resp. target) of this **Monitoring** (resp. **Control**) link in this context diagram? why?
    e.g. Scheduler Controls Meeting.RequiredEquipment
    => LocalOrganizer as monitoring agent

☠ Don't confuse product-level agents & process-level stakeholders

Heuristics for building agent diagrams

- For goal <u>responsibility assignment</u>
  - Consider agents whose monitoring/control capabilities match quantities to be evaluated/constrained in the goal spec
  - Consider software assignment as alternative to human assignment
    + pros/cons as soft goals

    e.g. AccurateBookClassification => Staff vs. AutoClassifier ?
  - Identify finer-grained assignments by goal-agent co-refinement
  - Select assignments that best contribute to high-priority soft goals
  - Favor human assignments to agents wishing the goal or a parent goal

    e.g. AccurateBookClassification to ResearchStaff
    rather than AdministrativeStaff

☠ Avoid assignments resulting in critical agent dependencies

e.g. BiblioSearchEngine depending on AdministrativeStaff
for AccurateBookClassification

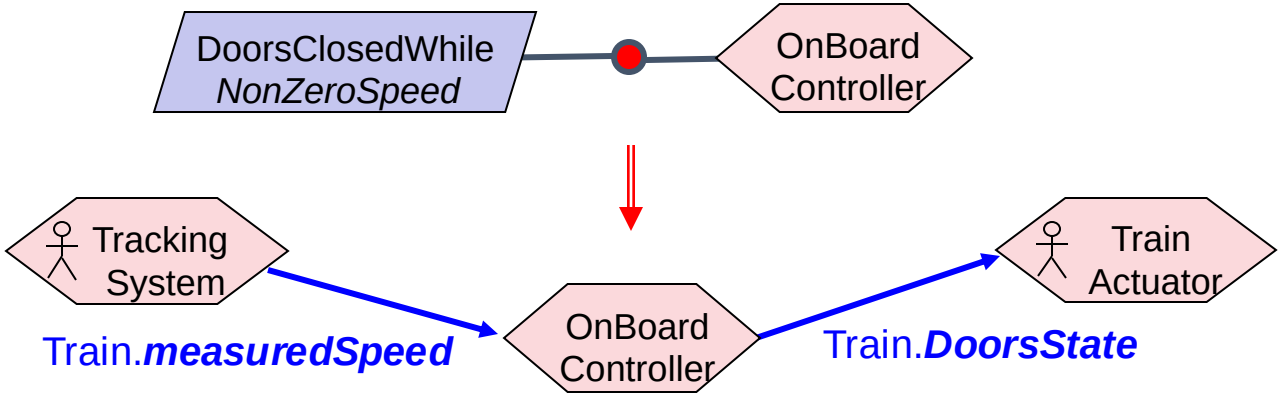
Deriving context diagrams from goals

- Behavioral goal specs are of form:

  G: CurrentCondition [monitoredVariables]
  $\Rightarrow$ [sooner-or-later/always] TargetCondition [controlledVariables]

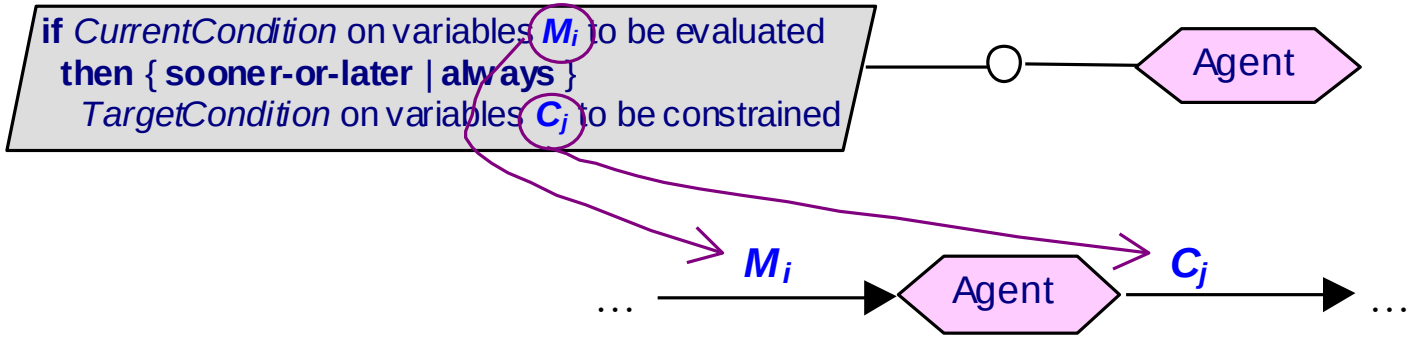
- Cf. goal-capability matching for goal realizability

  tr.measuredSpeed $\neq 0 \Rightarrow$ tr.DoorsState = 'closed'



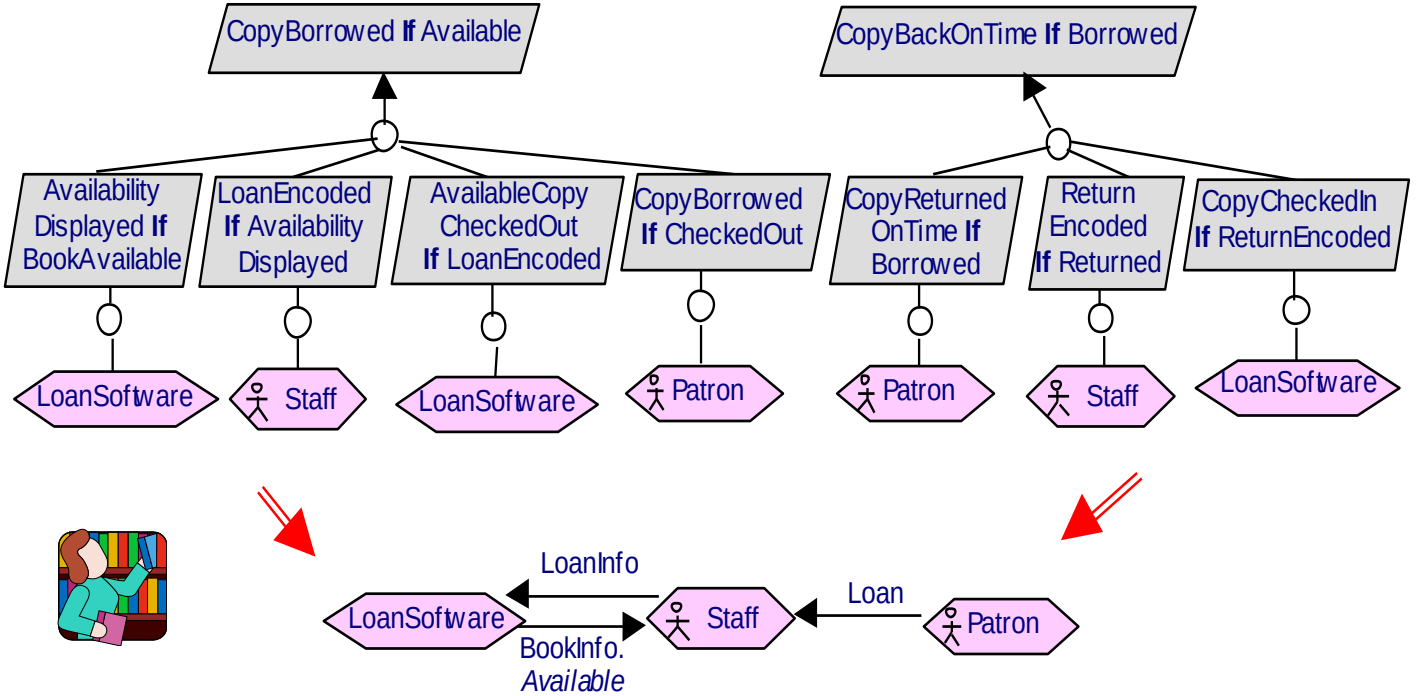
Deriving context diagrams from goals, more generally

- Agent interfaces are derived from goal specs



- Context diagram is derived piecewise by iteration on leaf goals
  - agent with outgoing arrow labelled var is connected to all agents with incoming arrow labelled var

Deriving context diagrams from goals: another example



Modeling system agents: summary

- What we know about agents so far
- Characterizing system agents
  - capabilities
  - responsibilities
  - operation performers
  - wishes & beliefs
  - dependencies
- Representing agent models
  - agent diagram, context diagram, dependency diagram
- Refinement of abstract agents
- Building agent models: heuristics & derivation rules