SOFTWARE ENGINEERING

Goal-Oriented Requirements Engineering

The KAOS approach

What are *Goals*?

- *Goals* are desired system properties that have been expressed by some stakeholders

- **Goal** = prescriptive statement of intent the system should satisfy through cooperation of its agents

- Goals can be specified in different levels of abstraction, covering at higher level strategic concerns and at lower level technical issues

The central role of goals in the RE process (5)

- Support for evolution management
  - *Higher-level goals → more stable concerns*
  - *Roots for conflict detection & resolution*
- Anchors for risk management

The granularity of goals

- Goals can be stated at different levels of abstraction
  - ***Higher-level*** *goals: strategic, coarse-grained*
    - ✓ "50% increase of transportation capacity"
    - ✓ ”Effective access to state of the art"
  - ***Lower-level*** *goals: technical, fine-grained*
    - ✓ ”Acceleration command sent every 3 secs"
    - ✓ ”Reminder issued by end of loan period if no return"
- The **finer**-grained a goal, the **fewer** agents required for its satisfaction




Identifying and Refining Goals

- Discover system goals through interviews, technical documents, etc.
- Each goal in the model (except the roots) is justified by at least another goal explaining **why** the goal was introduced in the model
- Each goal in the model (except the leaves, bottom goals) is refined by a collection of subgoals describing **how** the refined goal can be reached
- The identification is both top-down and bottom-up
  - *In summary, refining and abstracting goals (WHY & HOW)*

Functional and NF goals

- They can be:
  - *Functional* – related to the services provided
  - *Non-functional* – related to **quality** of services (e.g. Security, performance, availability) and development
  - Examples of goals (elevator system):
    - *"Each time a passenger calls an elevator from floor f1 to go to floor f2, the elevator system eventually takes him to f2"*
    - *"Safe elevator system"*

About KAOS

- It originates from the cooperation between University of Oregon and University of Louvain in 1990
- CEDITI, a spin-off company of the University of Louvain developed the tool **Objectiver**

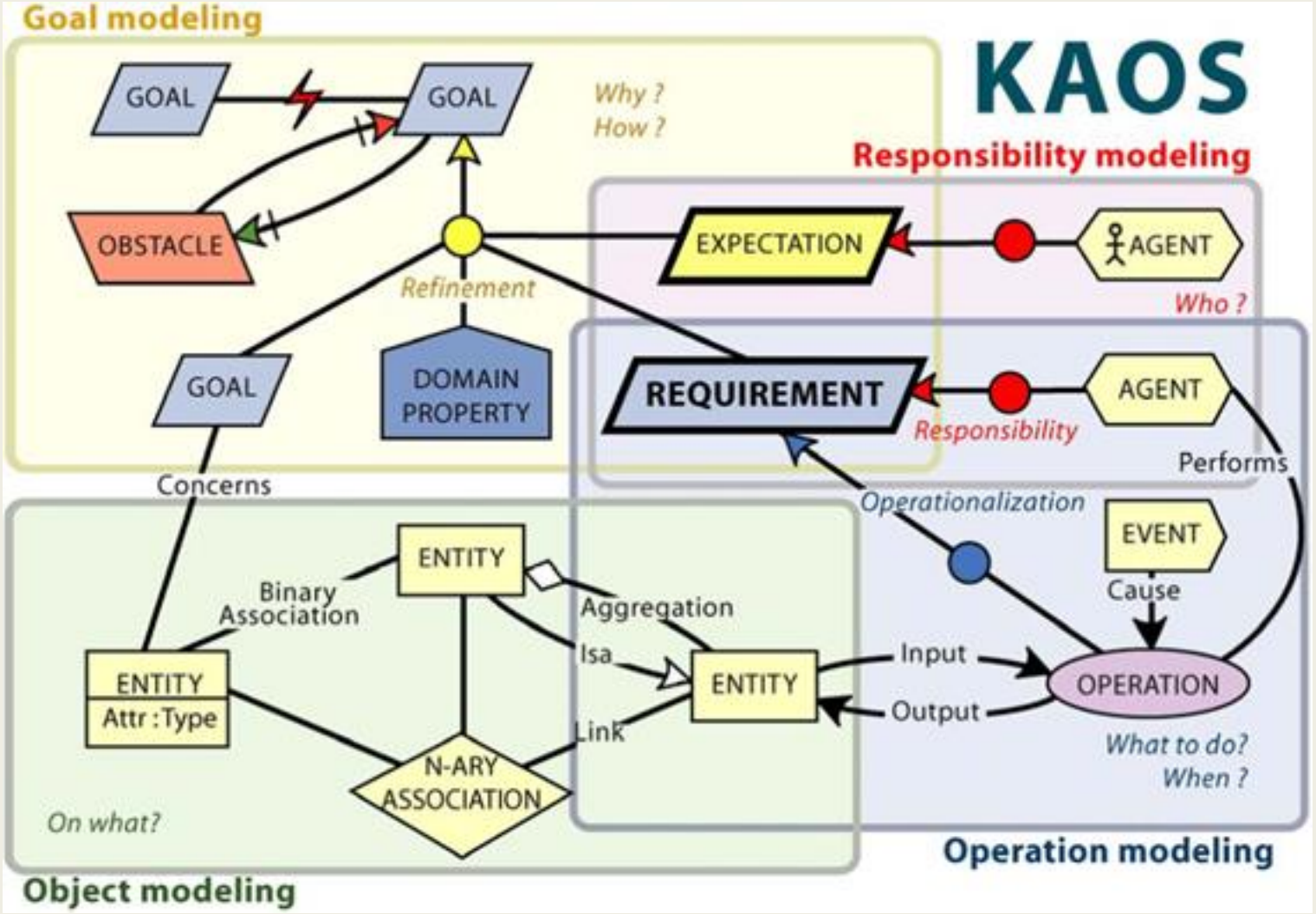
KAOS (cont.)

- It is a systematic approach to discover and structure requirements while:
  - *Avoiding ambiguous or irrelevant requirements*
  - *Allowing efficient and easy communication between stakeholders*
  - *Clarifying stakeholders responsibilities*
- It also provides mechanisms to:
  - *Choose between different alternatives*
  - *Manage conflicts*
  - *Refine goals to structure complex requirements*

KAOS models

- Goal model
- Responsibility model
- Object model
- Operation model

KAOS main model elements



Goal features are specified in model annotations

DoorsClosedWhileMoving

goal

annotation

Goal *Maintain* [DoorsClosedWhileMoving]

Def *All train doors shall be kept closed at any time when the train is moving*

precise definition

[FormalSpec ... in temporal logic for analysis.]

[Category Safety]

[Priority Highest]

[Source From interview with railway engineer X ...]

features

Specification of goals

- Informal (mainly text)
- Semi-formal (mainly diagrams)
- Formal (use of a formal language – expressed in temporal logic formulas)

Goal

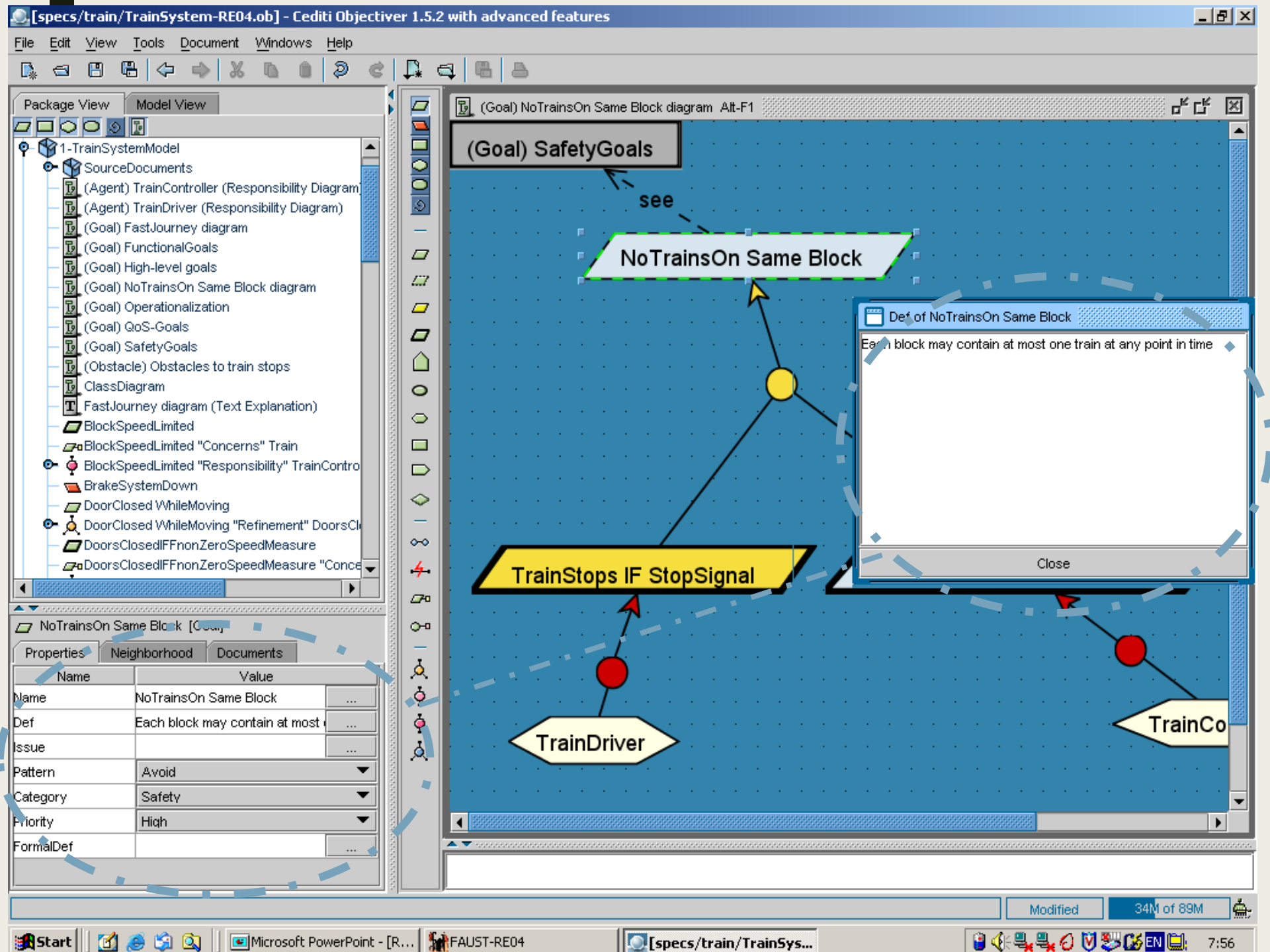
Achieve[AmbulanceIntervention]

InformalDef

For every urgent call reporting an incident, there should be an ambulance at the scene of the incident within 14 mins

FormalDef

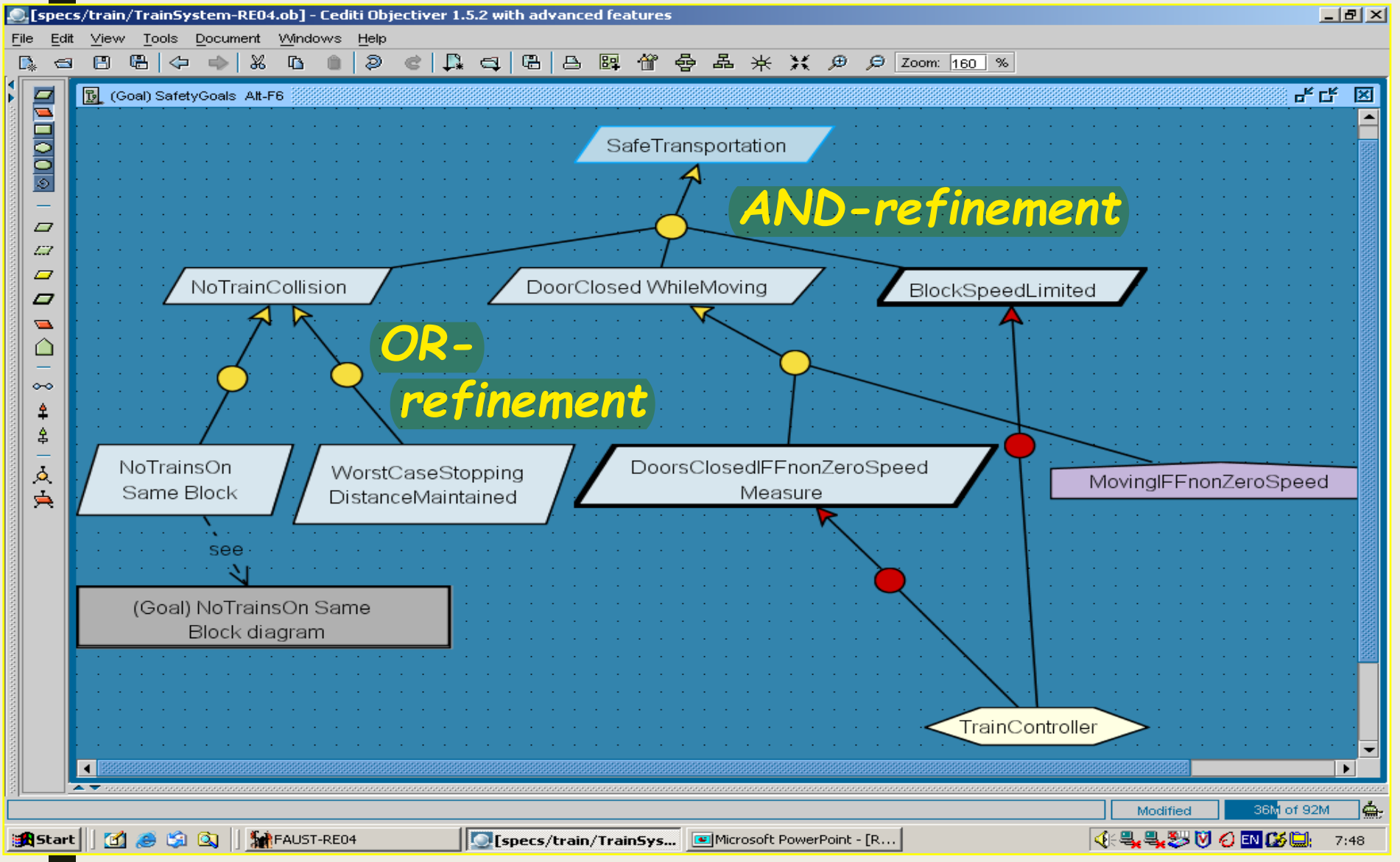
$\forall\, c$: UrgentCall, inc : Incident (@ Reporting(c, inc) $\Rightarrow$
$\Diamond_{\leq 14 \text{ mins}}\ \exists\, amb$: Ambulance (Intervention(amb, inc)))


[specs/train/TrainSystem-RE04.ob] - Cediti Objectiver 1.5.2 with advanced features
File Edit View Tools Document Windows Help
Package View
Model View
1-TrainSystemModel
SourceDocuments
(Agent) TrainController (Responsibility Diagram)
(Agent) TrainDriver (Responsibility Diagram)
(Goal) FastJourney diagram
(Goal) FunctionalGoals
(Goal) High-level goals
(Goal) NoTrainsOn Same Block diagram
(Goal) Operationalization
(Goal) QoS-Goals
(Goal) SafetyGoals
(Obstacle) Obstacles to train stops
ClassDiagram
FastJourney diagram (Text Explanation)
BlockSpeedLimited
BlockSpeedLimited "Concerns" Train
BlockSpeedLimited "Responsibility" TrainContro
BrakeSystemDown
DoorClosed WhileMoving
DoorClosed WhileMoving "Refinement" DoorsCl
DoorsClosedIFFnonZeroSpeedMeasure
DoorsClosedIFFnonZeroSpeedMeasure "Conce
NoTrainsOn Same Bloc
Properties
Neighborhood
Documents
Name
Value
Name
NoTrainsOn Same Block
Def
Each block may contain at most
Issue
Pattern
Avoid
Category
Safety
Priority
High
FormalDef
(Goal) NoTrainsOn Same Block diagram Alt-F1
(Goal) SafetyGoals
see
NoTrainsOn Same Block
Def of NoTrainsOn Same Block
Each block may contain at most one train at any point in time
Close
TrainStops IF StopSignal
TrainDriver
TrainCo
Modified
34M of 89M
Start
Microsoft PowerPoint - [R...
FAUST-RE04
[specs/train/TrainSys...
7:56

Goals relationships

- AND Refinements
- OR Refinements
- Conflicts
- Obstruction and resolution links
- Responsibilities links

A goal model shows contribution links

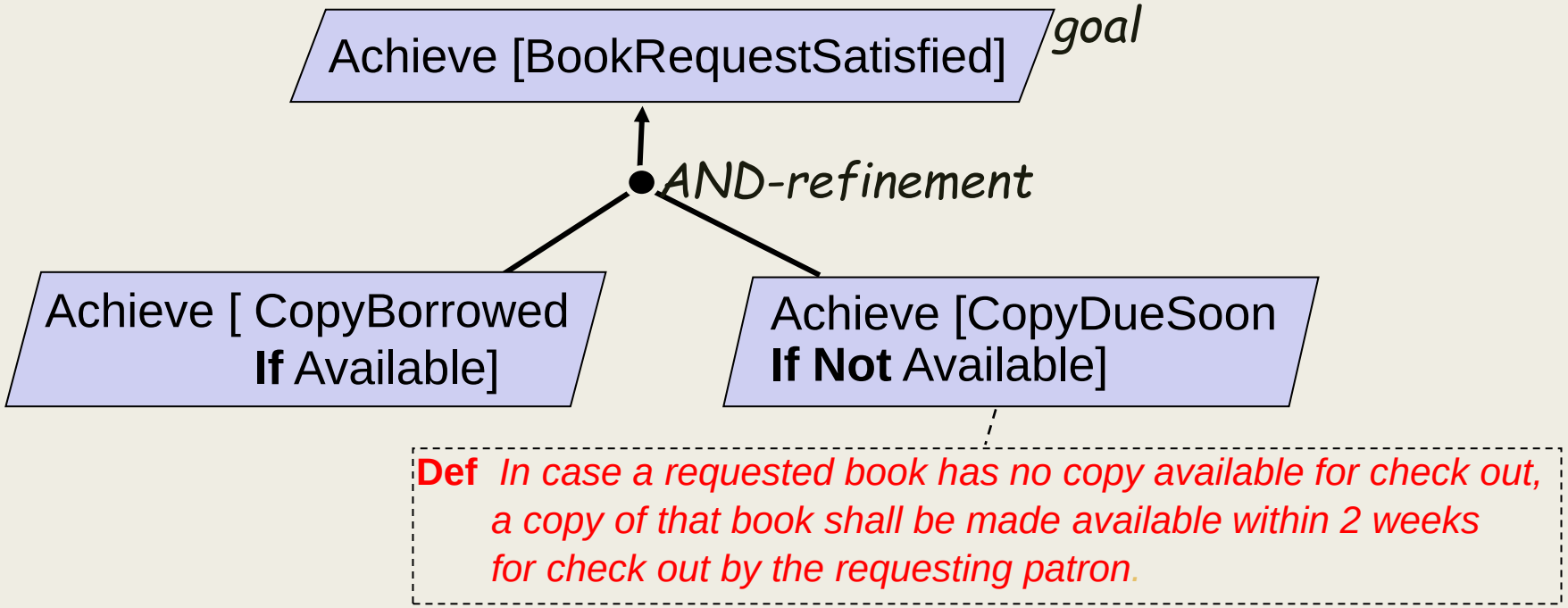


Goal refinement

- An **AND-refinement** of goal *G* into subgoals G_1, ..., G_n states that *G* can be satisfied by satisfying G_1, ..., G_n

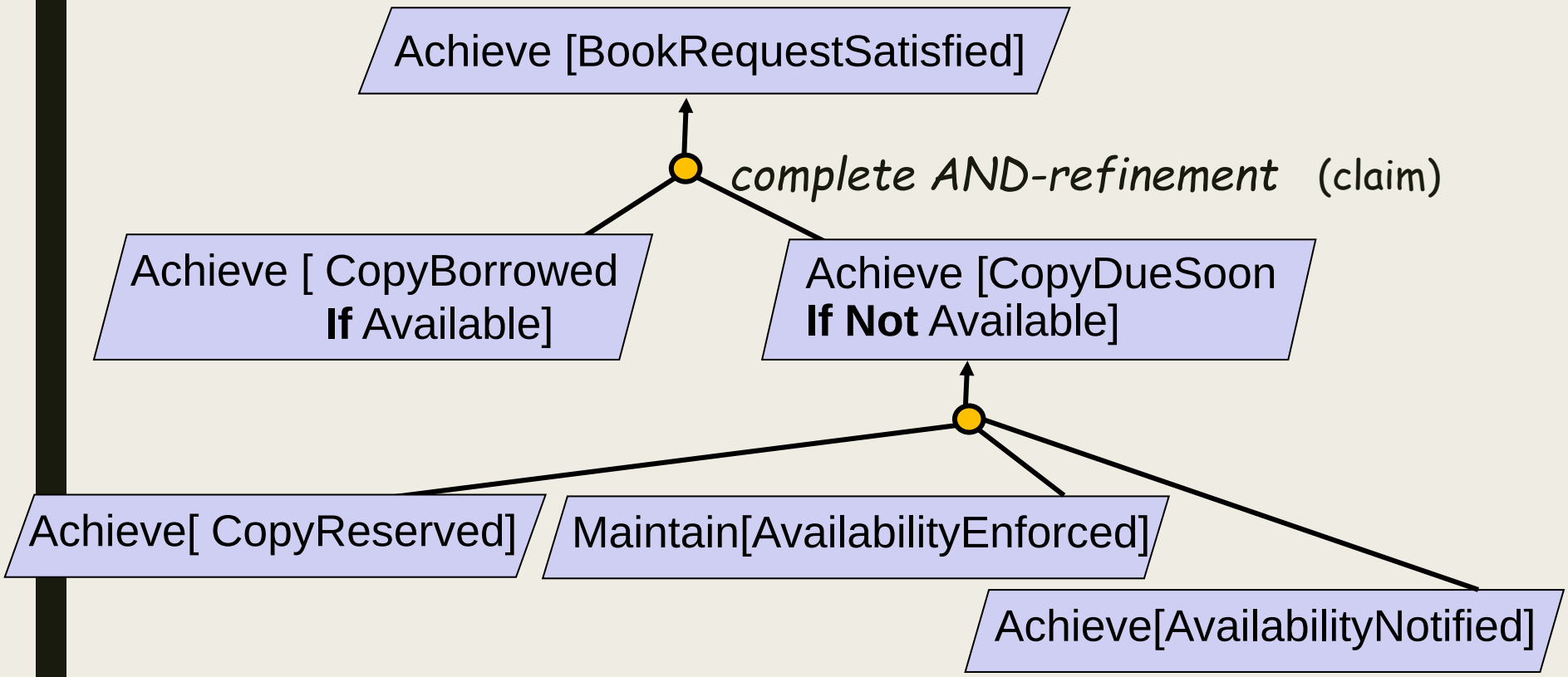
  The set {G_1, ..., G_n} is called **refinement** of *G*

  *Subgoal G_i is said to **contribute positively** to G*

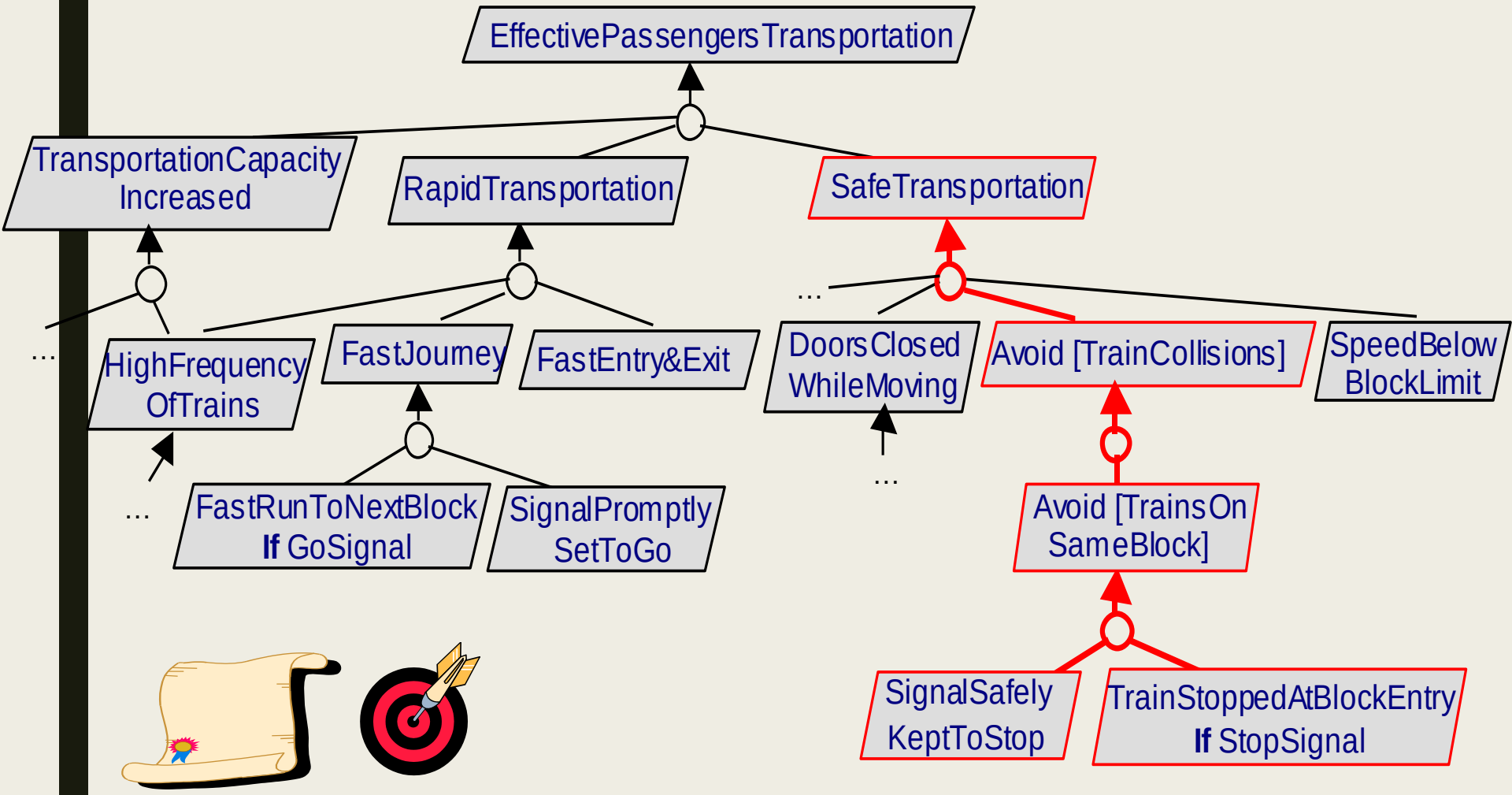


AND-refinements should be complete

- $\{G_1, ..., G_n\}$ is a **complete AND-refinement** of G iff satisfying $G_1, ..., G_n$ is *sufficient* for satisfying G in view of known domain properties
  - $\{G_1, ..., G_n, \text{Dom}\} \models G$



Refinement trees visualize satisfaction arguments

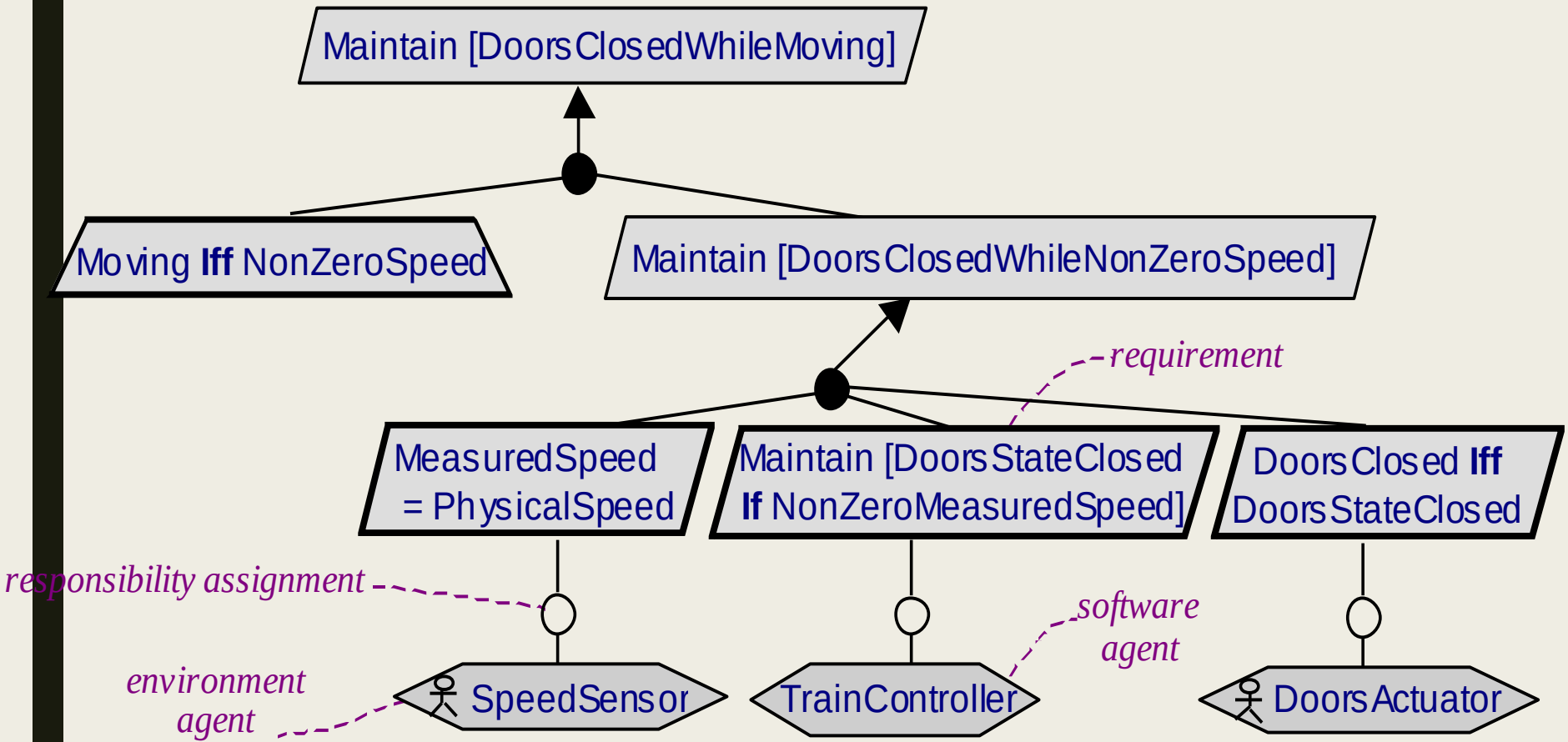


Goal satisfaction requires agent cooperation

- Maintain [SafeTransportation] ↔ 
  - *on-board train controller* + *tracking system* + *station computer* + *passenger* + *train driver* + ...
- Achieve [BookCopyReturnedToShelves] ↔ patron + staff + library SW 
- **Agent**: active system component
  - ✓ *responsible for goal satisfaction*
- Agent = role, rather than individual
  - ✓ *must restrict its behavior to meet its assigned goals*
  - ✓ *must be able to monitor/control phenomena involved in assigned goals*
- Agent types
  - ✓ *software (software-to-be, legacy software, foreign software)*
  - ✓ *device (sensor, actuator, ...)*
  - ✓ *human*

Refinement trees

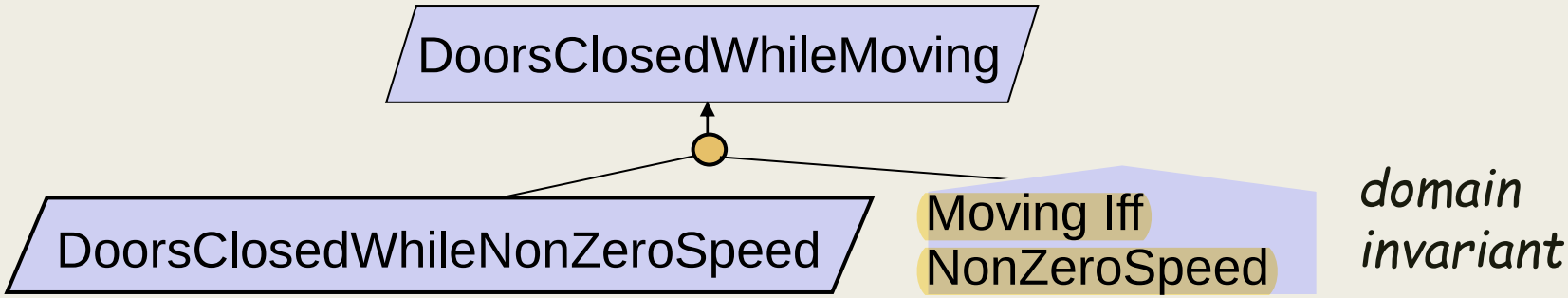
- Goals are recursively refinable
- Leaf nodes = goals assignable to single system agents



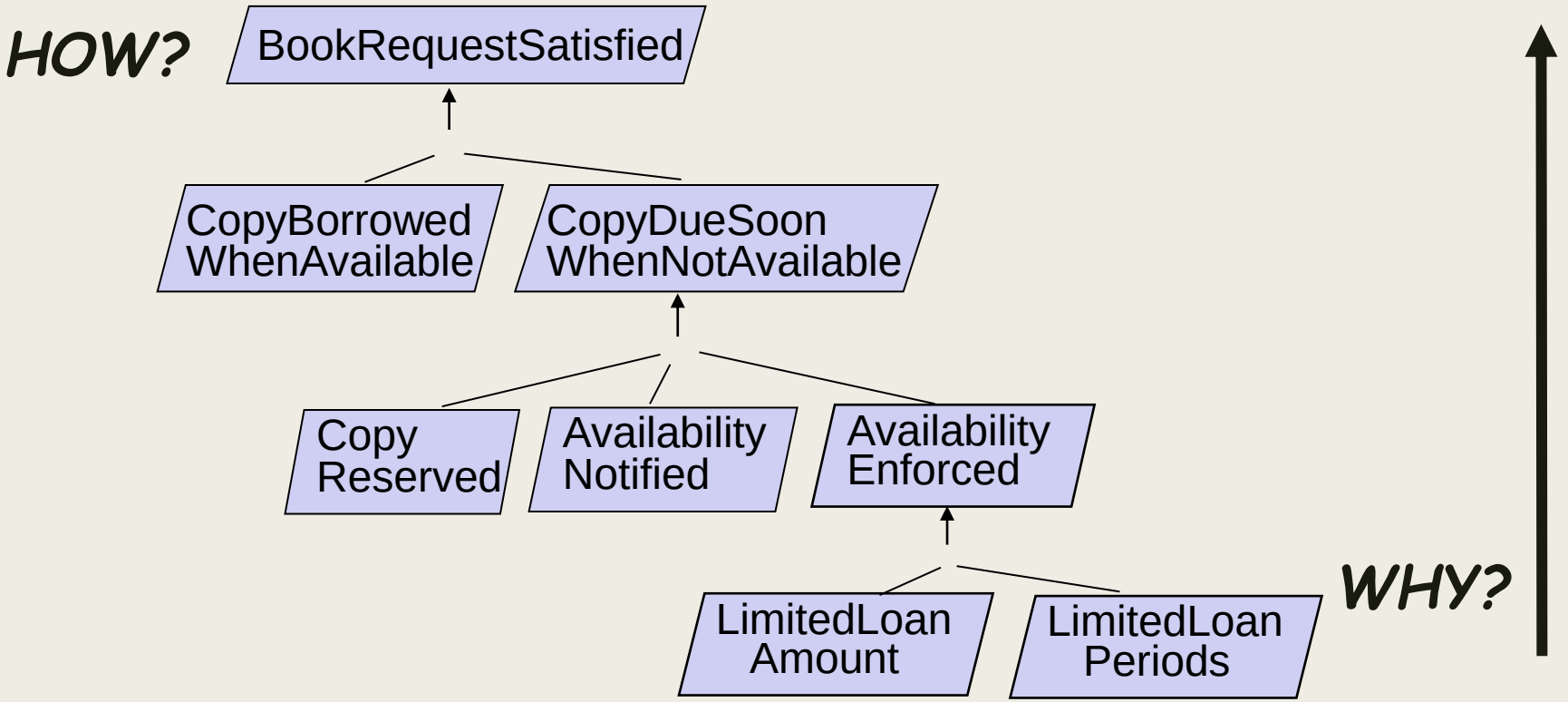
Goals vs. domain properties

- **Domain property** = descriptive statement about environment
  - ✓ *indicative mood: "is", "are", etc --not prescriptive*
  - ✓ *e.g. "A borrowed book is not available for other patrons"*
- The distinction between goals & domain properties is essential for RE …
  - ✓ *goals can be negotiated, weakened, prioritized*
  - ✓ *domain properties cannot*
  - ✓ *both required in requirements documentation*

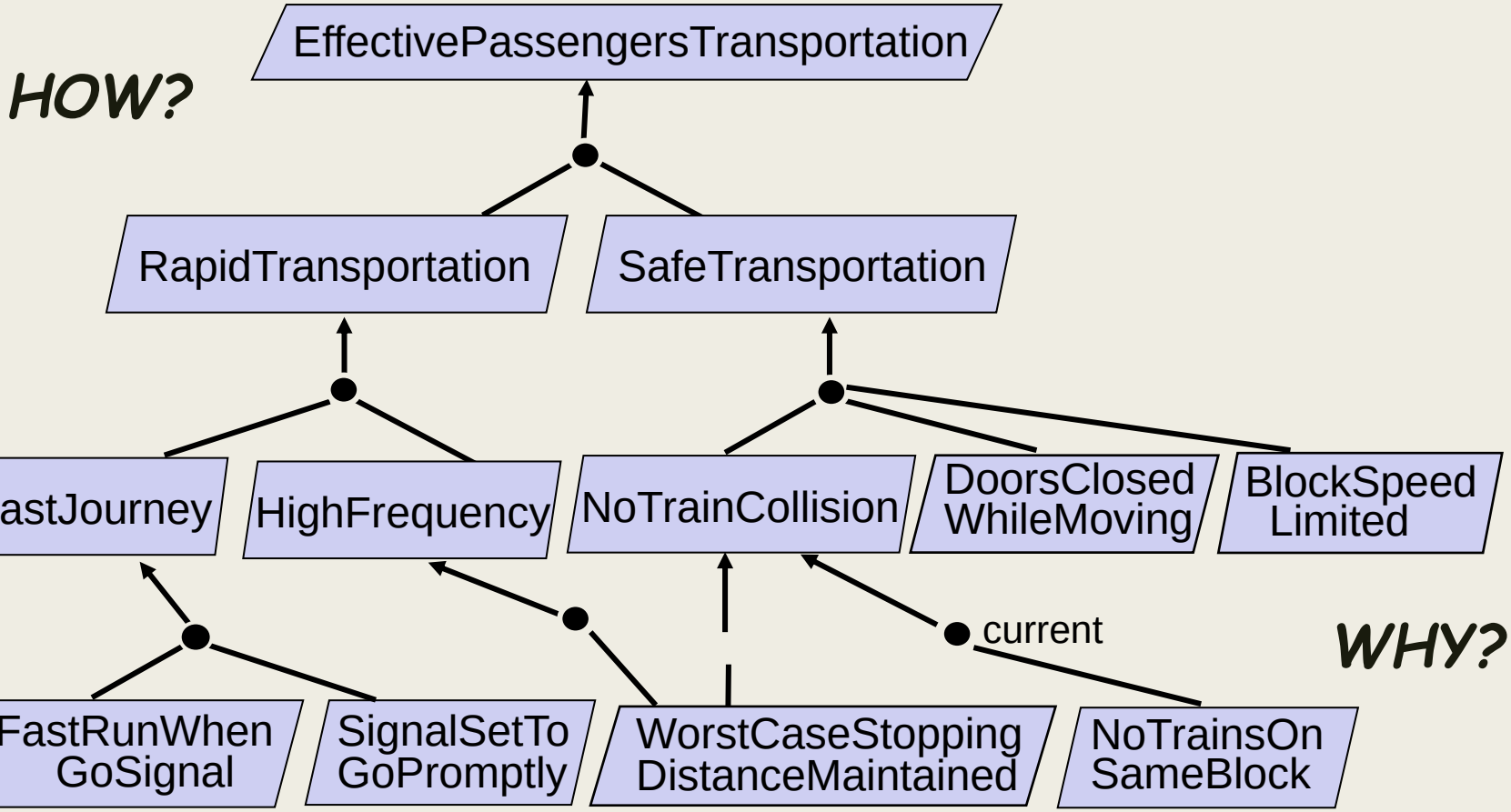
Domain properties in AND-refinements



Building goal models: How and why questions



Building goal models: HOW and WHY questions



Responsibility model

- **Agents** are humans or automated components that are responsible for achieving requirements expectations
- **Expectations** are requirements on agents interacting with the system
  - *They are introduced to show how the SW system and its environment have to cooperate to achieve the goals*
  - *It is type of goal to be achieved by an agent part of the environment of the system*
- A **requirement** is a low level type of goal to be achieved by a software agent
  - *The software agent is responsible for it*

Goals, requirements & expectations

- **Requirement** = goal assigned to single agent in software-to-be

  "doorState = ‘closed’ while measuredSpeed ≠ 0" ↔ *TrainController*

  ”Acceleration command sent every 3 secs” ↔ *StationComputer*

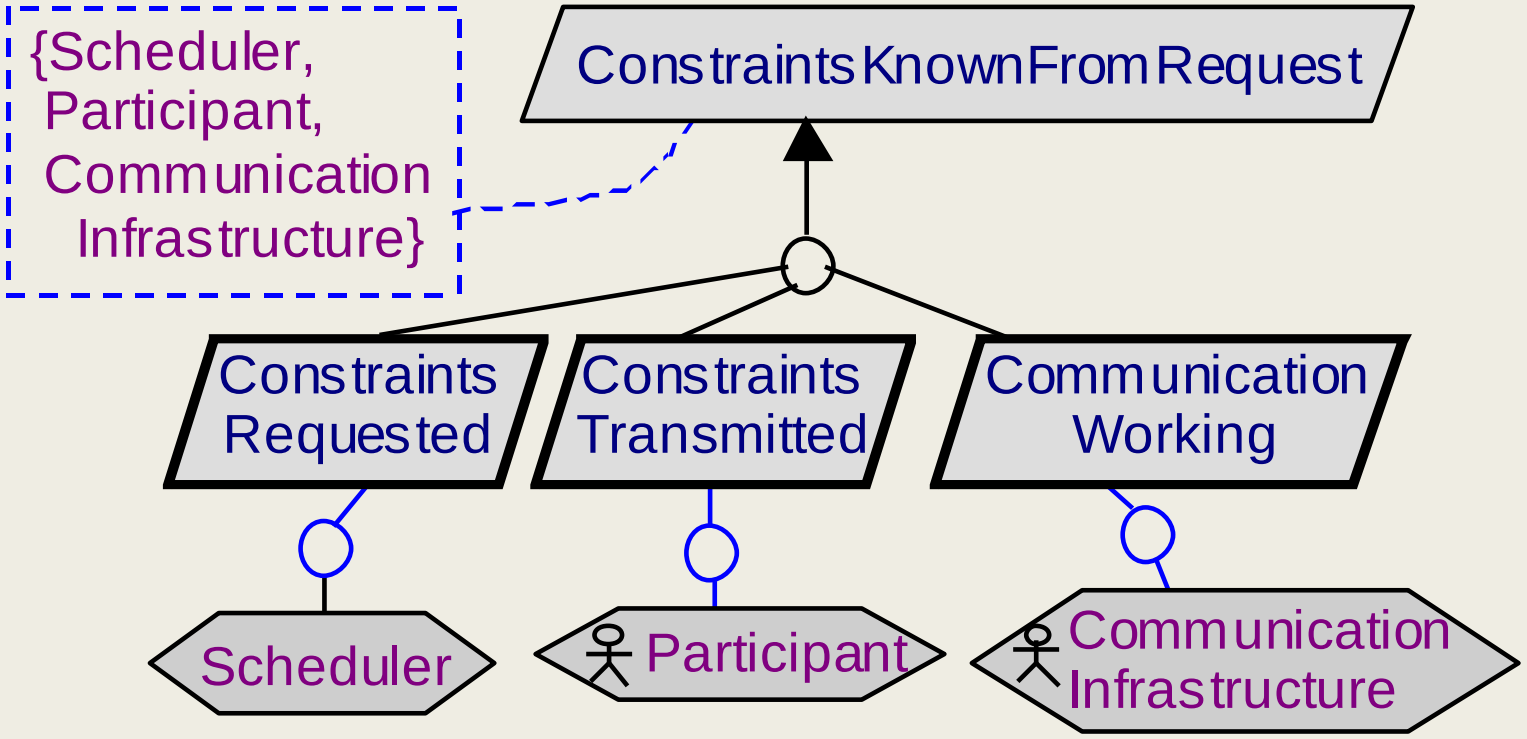
- **Expectation** = goal assigned to single agent in environment
  - *prescriptive assumption on environment*
  - *cannot be enforced by software-to-be (unlike requirements)*

  ”Passenger enters when doors open at destination” ↔ *Passenger*

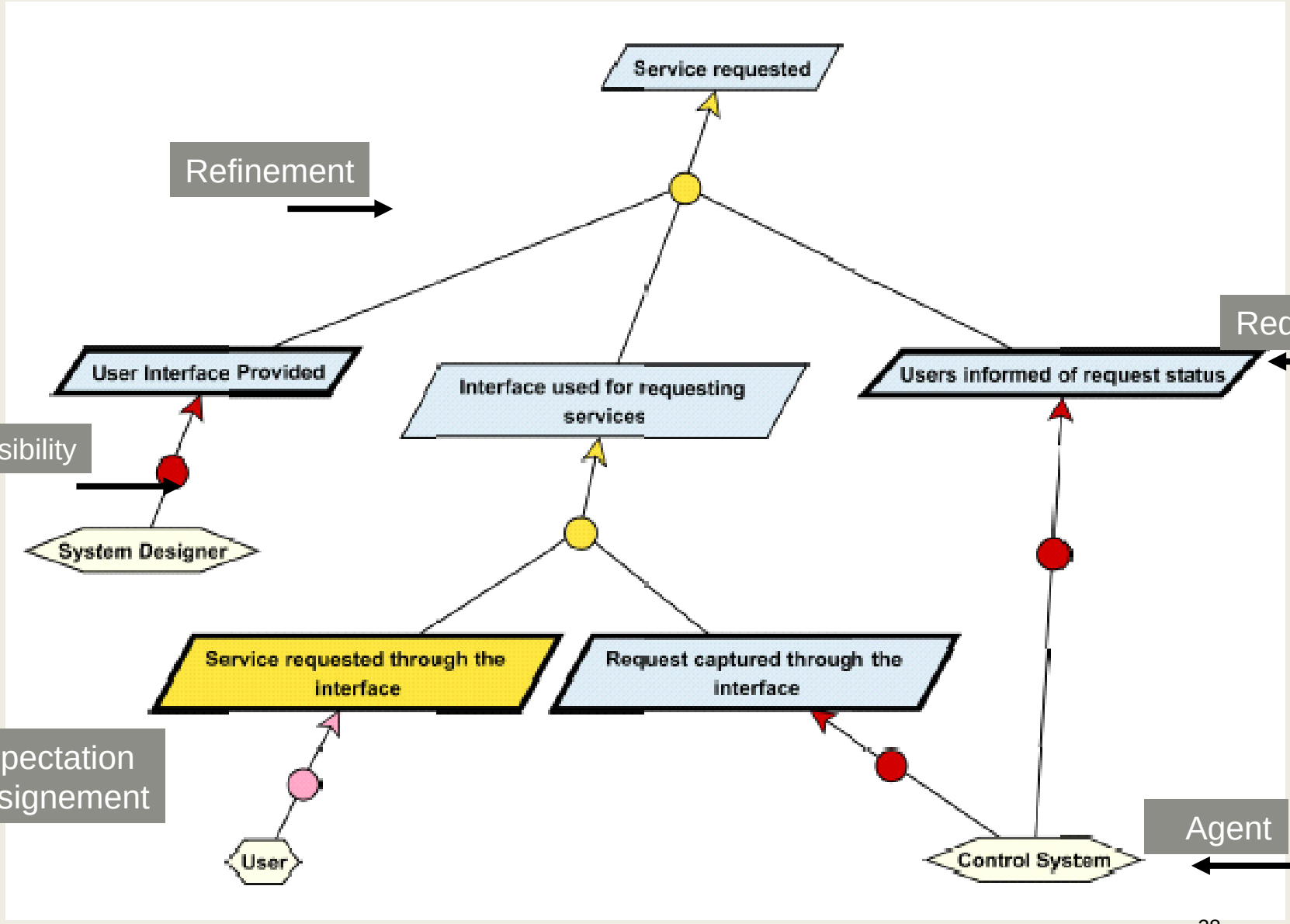
Split responsibilities among agents

- *to get subgoals involving fewer agents*

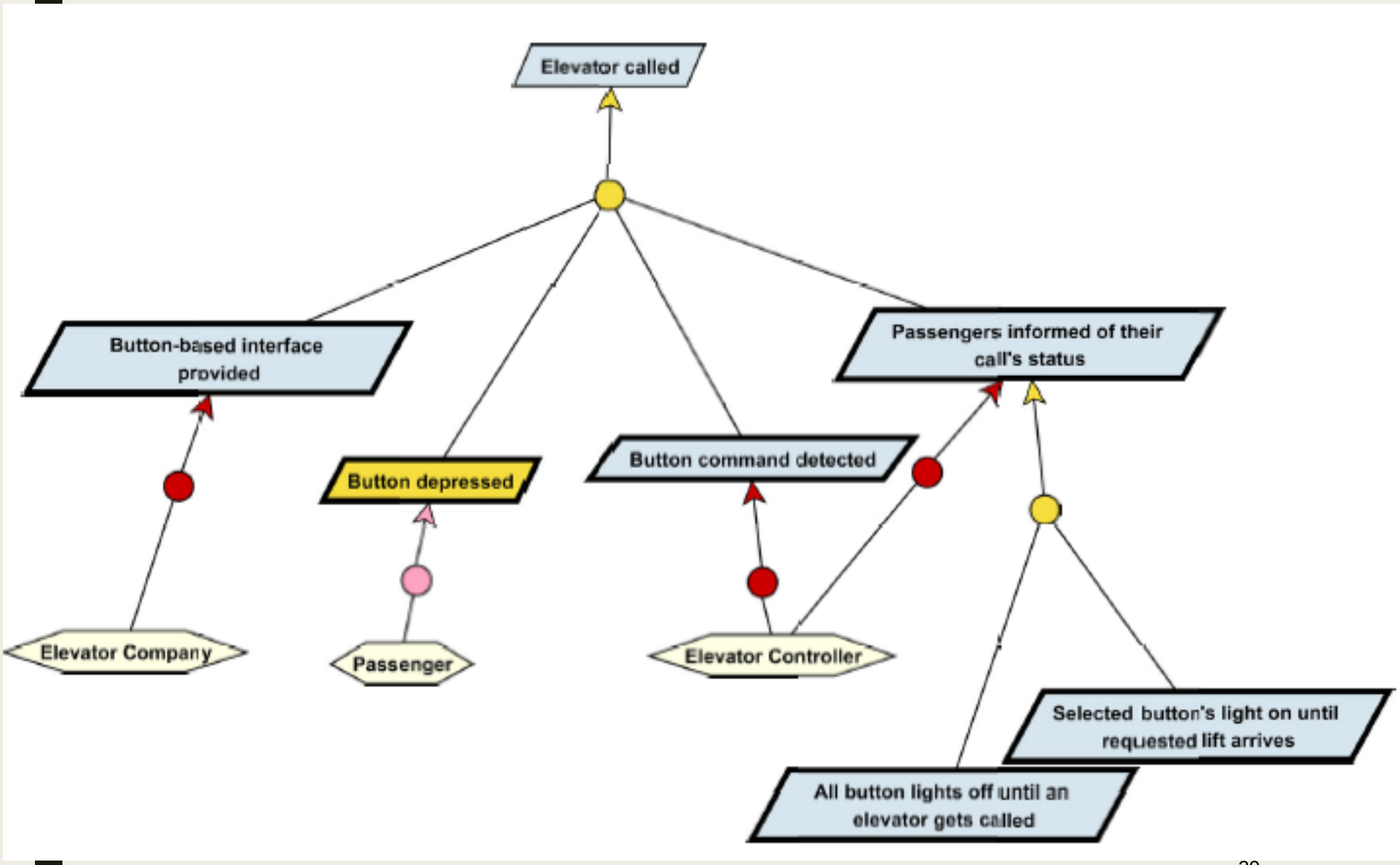
  and move towards requirements and expectations



Patterns can be defined: Generic Goal "Requested service"



Applying the generic pattern to the elevator system

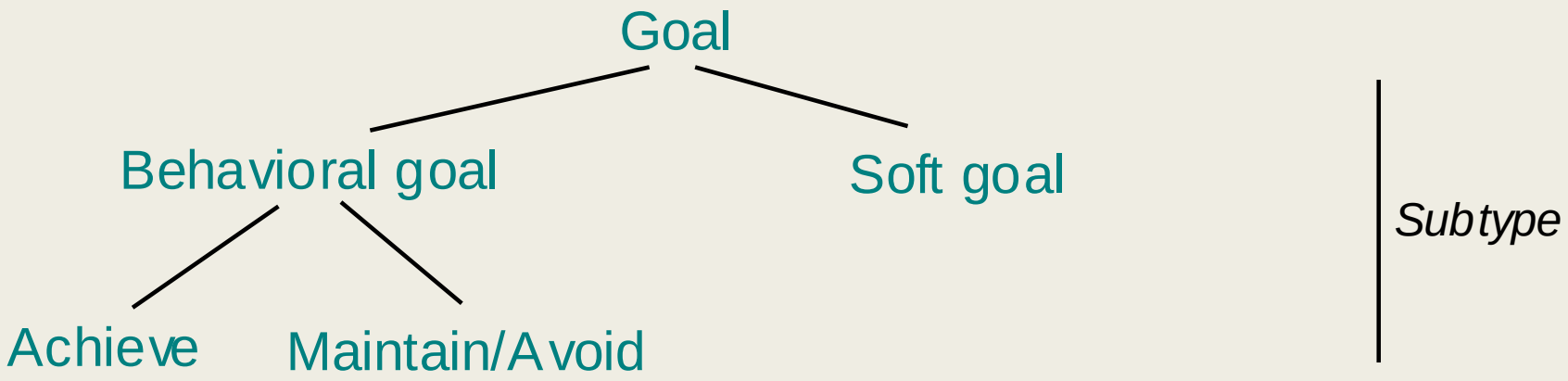


Goal types

Behavioral goals: prescribe behaviors

vs.

Soft goals: state preferences among alternative behaviors



Goal types: behavioral goals

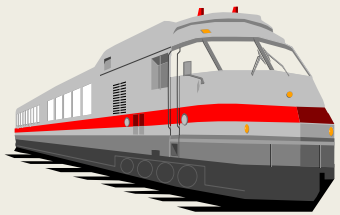
- Prescribe intended system behaviors declaratively
  - *implicitly define admissible agent behaviors*
- Can be satisfied in a clear-cut sense: YES *or* NO
- Used for building operation models to meet them

”Worst-case stopping distance maintained”

“Reminder sent if book not returned on time”

Behavior goals prescribe desired behaviors

DoorsClosed WhileMoving



Behavioral goals: subtypes and specification patterns

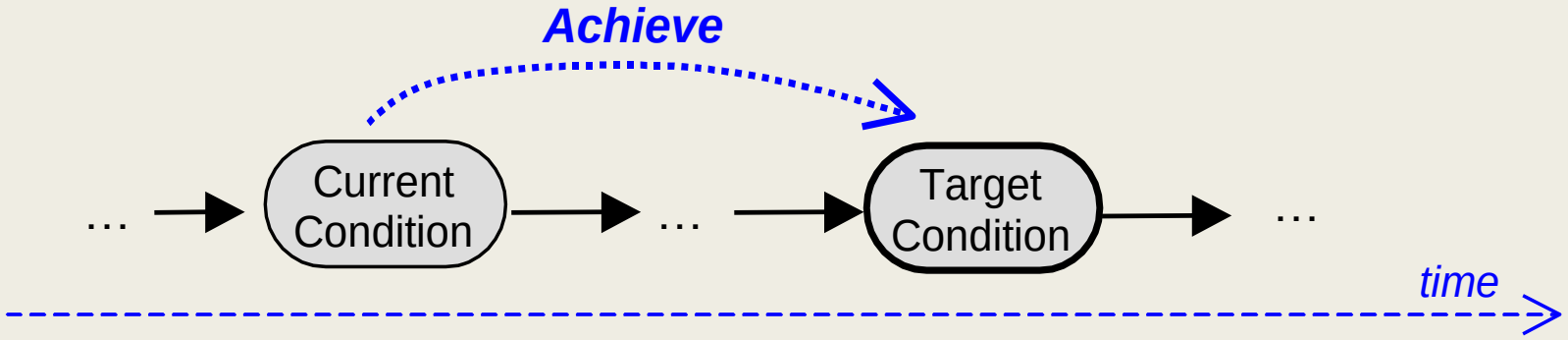
- ***Achieve*** [TargetCondition]:
  - **[if** CurrentCondition **then]** **sooner-or-later** TargetCondition

Achieve *[BookRequestSatisfied]:*

if a book is requested ***then sooner-or-later***
a copy of the book is borrowed by the requesting patron

Achieve *[FastJourney]:*

if train is at some platform ***then within 5 minutes*** *it is at next platform*



Behavioral goals: subtypes and specification patterns (

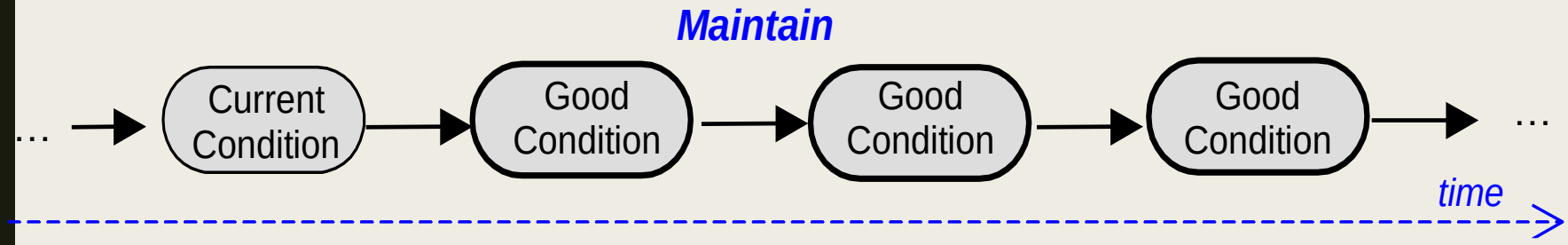
- ***Maintain*** [GoodCondition]:
  - **[if** CurrentCondition **then]** **always** GoodCondition
  - **always** (**if** CurrentCondition **then** GoodCondition)

Maintain *[DoorsClosedWhileMoving]:*

- **always** (**if** a train is moving **then** its doors are closed)

Maintain *[WorstCaseStoppingDistance]:*

- **always** (**if** a train follows another **then**
- its distance is sufficient to allow the other to stop suddenly)



Behavioral goals: subtypes and specification patterns (3

- Accuracy goals are usually of type *Maintain*

  Maintain *[**Accurate**BookClassification]:*

  - **if** a book is registered in the library directory **then**
  - **always** its keyword-based classification reflects its covered topics

- ***Avoid*** [BadCondition]: dual of *Maintain* ...
  - **[if** CurrentCondition **then]** **never** BadCondition

  Avoid *[BorrowerLoansDisclosed]:*

  never *patron loans disclosed to other patrons*

  Many security goals are Avoid goals

Goal types: soft goals

- Capture preferences among alternative behaviors
- Cannot be satisfied in clear-cut sense:

  more *satisfied in one option,* ***less*** *satisfied in another*

  - *goal satisficing, qualitative analysis*
- Used for comparing options to select preferred
- Often take the form

  Maximize / Minimize, Increase / Reduce, Improve, …

  "Stress conditions of air traffic controllers shall be reduced"

  "The workload of library staff shall be reduced"

  "The bibliographical search engine shall be usable by non-CS students"

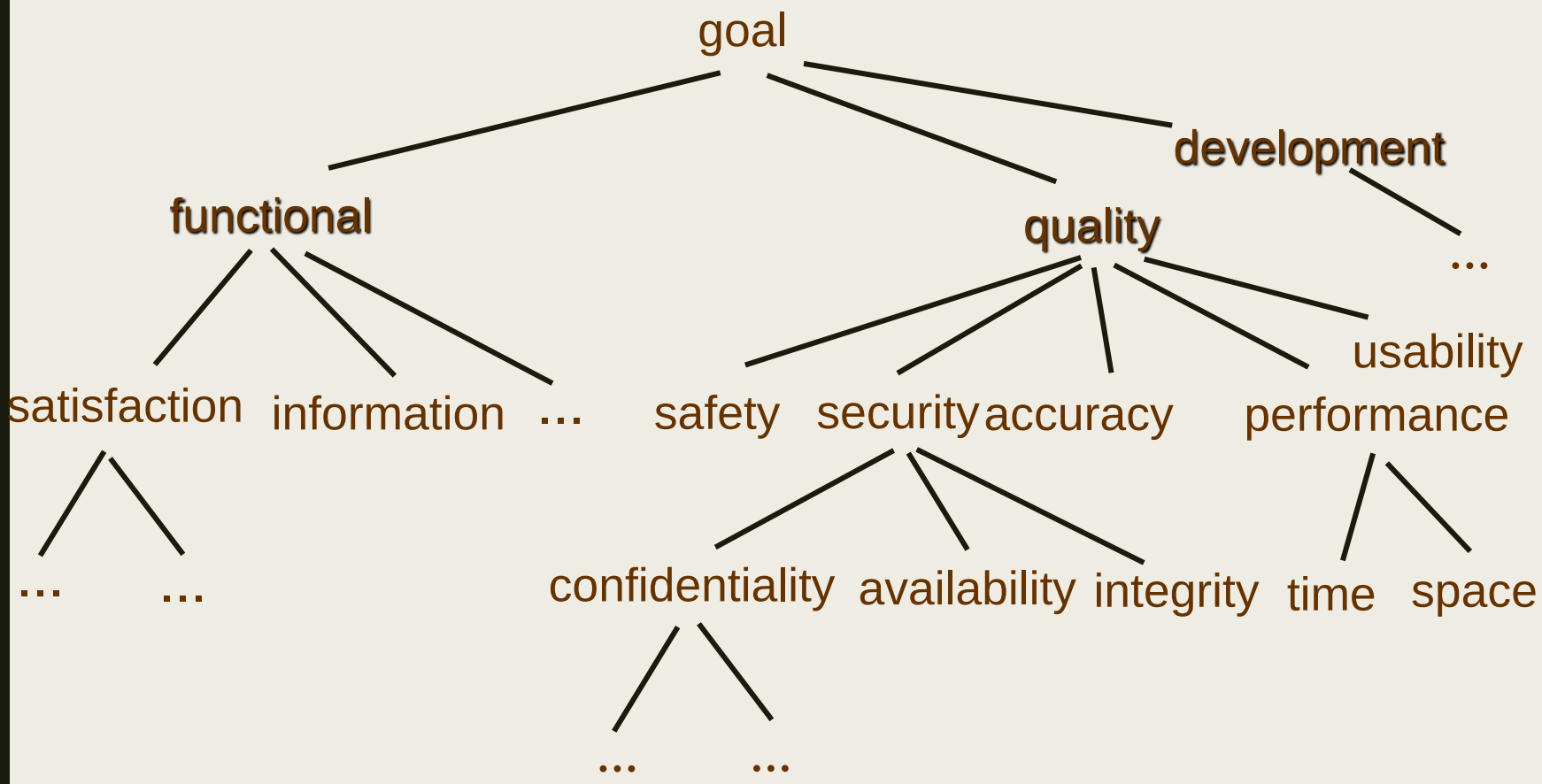
Goal categories

- Classification into **functional, quality, development** goals
- Categories may overlap; boundary not always clear
  - *unlike goal types*
- **Functional goals**
  - *prescribe intended services to be provided by the system*
  - *used for building operational models of such services*
    - use cases, state machines (see later)
    - e.g. “Passengers transported to their destination”
      “Train acceleration computed”
      ”Book request satisfied"

Goal categories: non-functional goals

- **Quality** goals (not to be confused with soft goals,)
  - *about quality of service …*
    - security "info about other patrons kept confidential"
    - safety "worst-case stopping distance maintained"
    - accuracy ”measured speed = physical speed”
      - ”book displayed as available **iff** there is a copy in shelves"
    - performance ”acceleration command sent every 3 seconds”
    - usability
    - interoperability, …
- **Development** goals
  - *about quality of development …*
    - cost, deadline, variability, maintainability, reusability, etc.

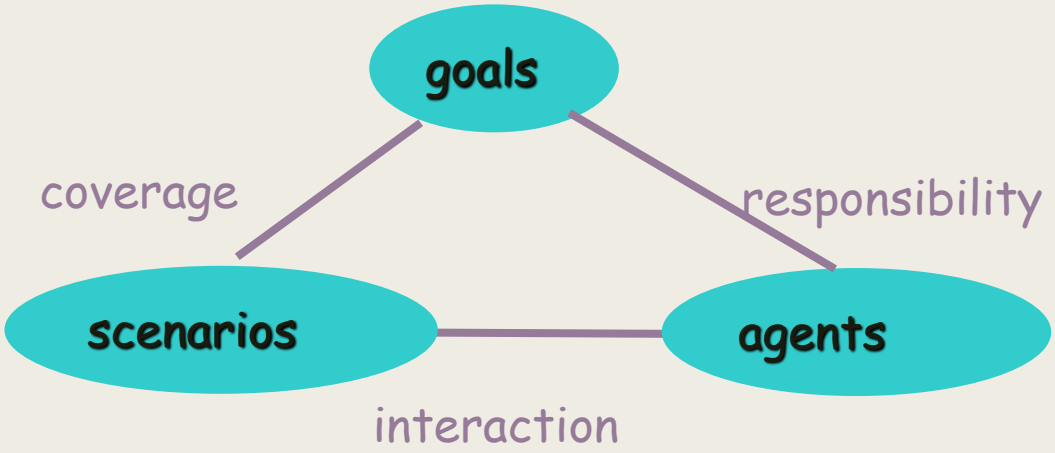
Goal categories (2)

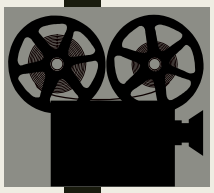


Avoid frequent misconceptions

- Goal-oriented ≠ top-down
  - *bottom-up elaboration as well (goal abstraction)*
- Goal-oriented ⇒ agent-oriented,
  scenario-oriented

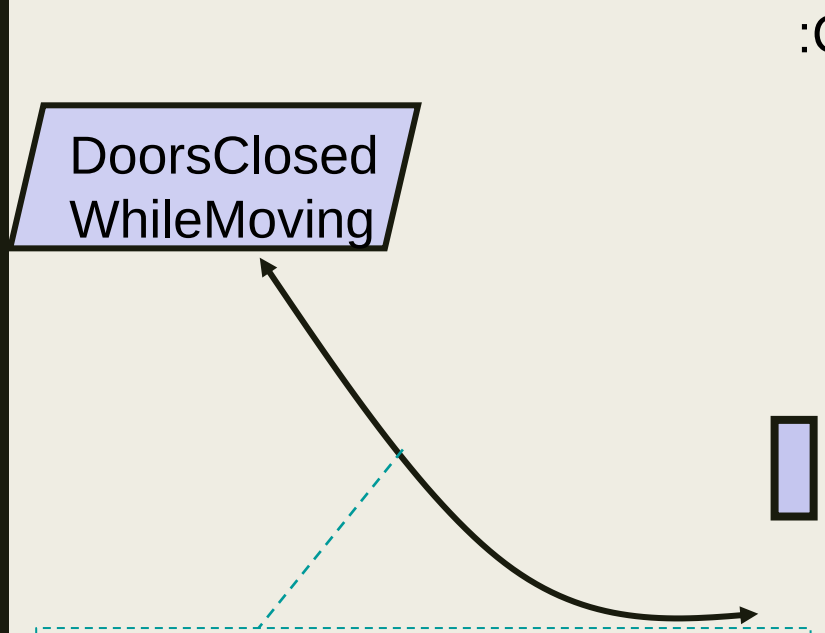
the magic RE triangle:





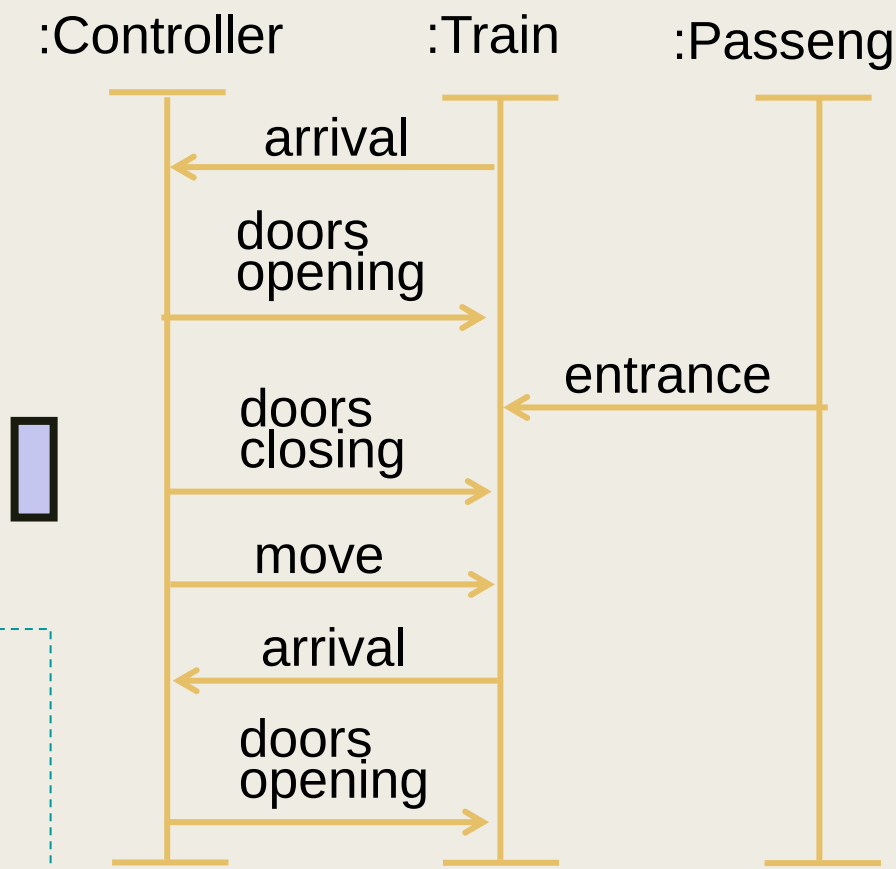
Scenarios as concrete vehicles for goal elicitation / validation

easy to get from or validate with stakeholders



G **covers** *Sc*:

Sc is subhistory in set of behaviors prescribed by *G*



Completeness criteria

- Criterion 1:
  - *A goal model is said to be complete with respect to the refinement relationship if and only if every leaf goal is either an expectation, a domain property or a requirement*
- Criterion 2:
  - *A goal model is said to be complete with respect to the responsibility relationship if and only if every requirement or expectation is placed under the responsibility of one and only one agent or implicitly if the requirement refines another one which has been placed under the responsibilty of some agent*