Software Engineering

Goal-Oriented Requirements Engineering

The KAOS approach

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**

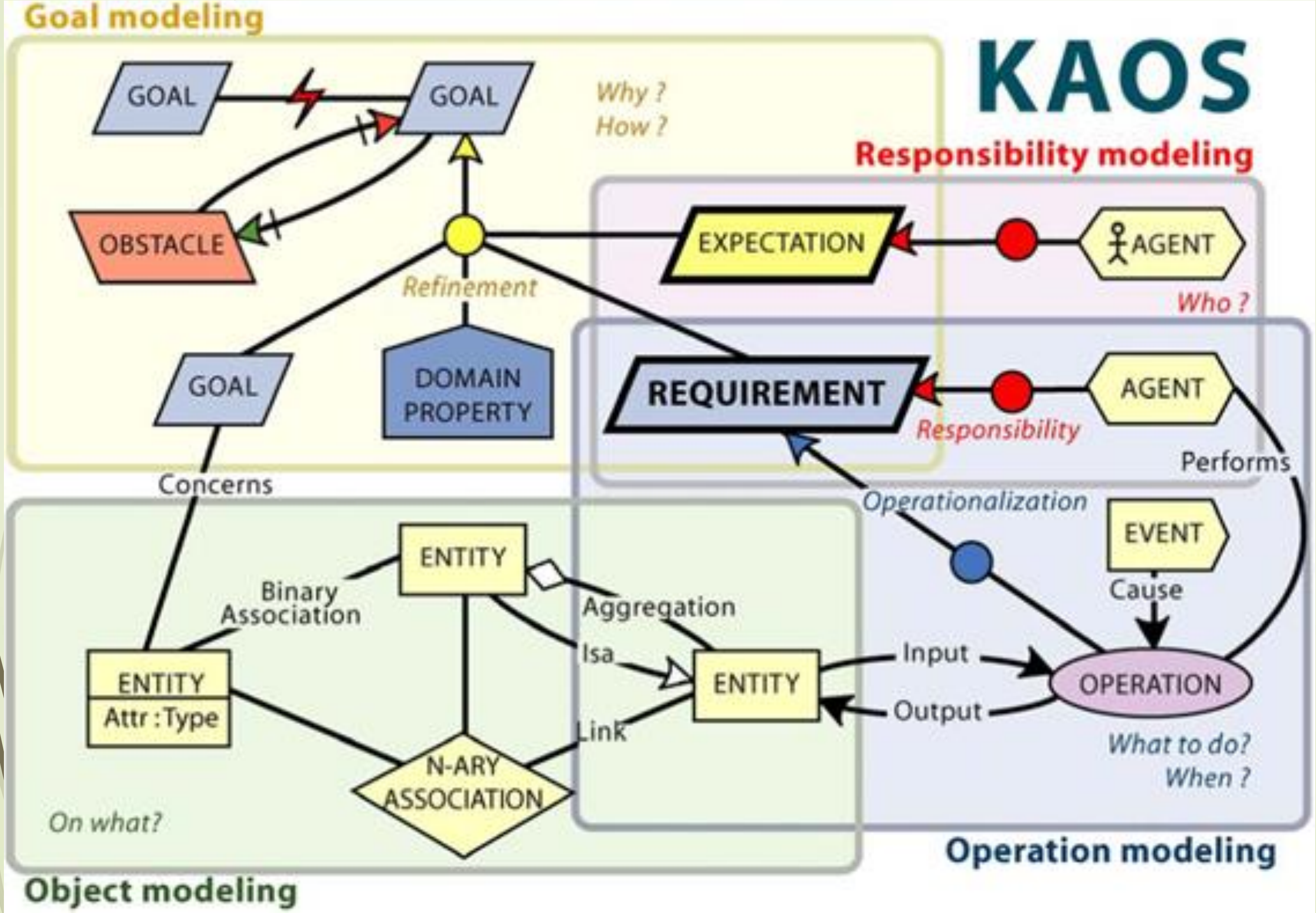
KAOS (cont.)

- 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



What are *Goals*?

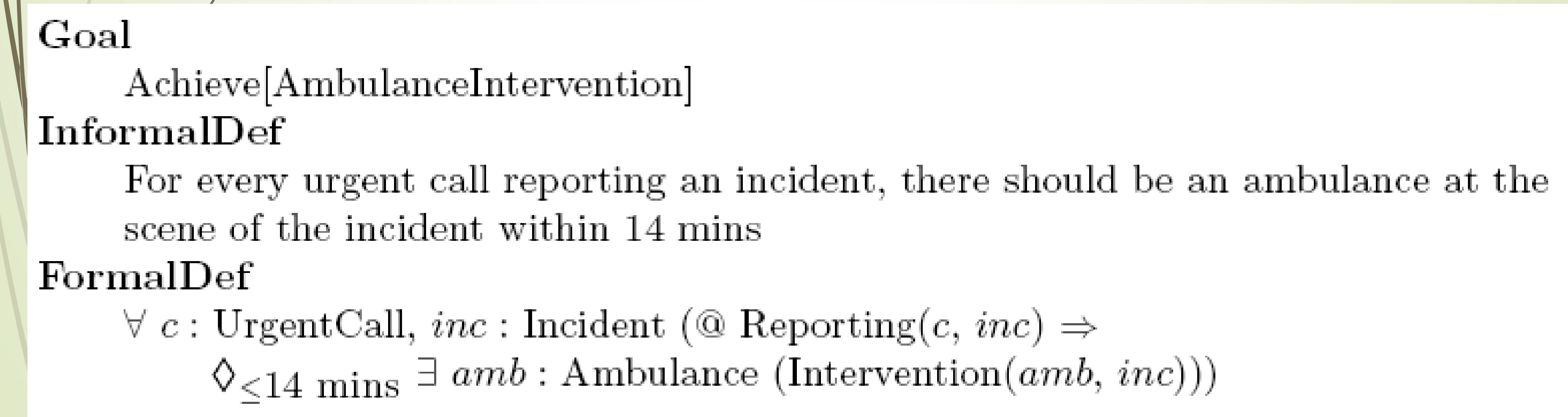
- *Goals* are desired system properties that have been expressed by some stakeholders
- **Goal** = prescriptive statement of intent the system should satisfy through cooperation of its agents
- Goals can be specified in different levels of abstraction, covering at a higher level strategic concerns and at a lower level technical issues
- 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"

Identifying 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)

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

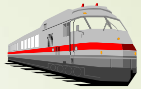
For every urgent call reporting an incident, there should be an ambulance at the scene of the incident within 14 mins

FormalDef

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

Goal satisfaction requires agent cooperation

Maintain [SafeTransportation] ↔



- on-board train controller + tracking system + station computer + passenger + train driver + ...

Achieve [BookCopyReturnedToShelves] ↔

patron + staff + library software

- **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

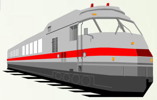
Goals vs. domain properties

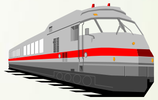
- **Domain property** = descriptive statement
  about environment
  - indicative mood: “is", “are", etc --not prescriptive
  - e.g. “A borrowed book is not available for other patrons”
- The distinction between goals & domain properties is essential for RE ...
  - goals can be negotiated, weakened, prioritized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

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





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)

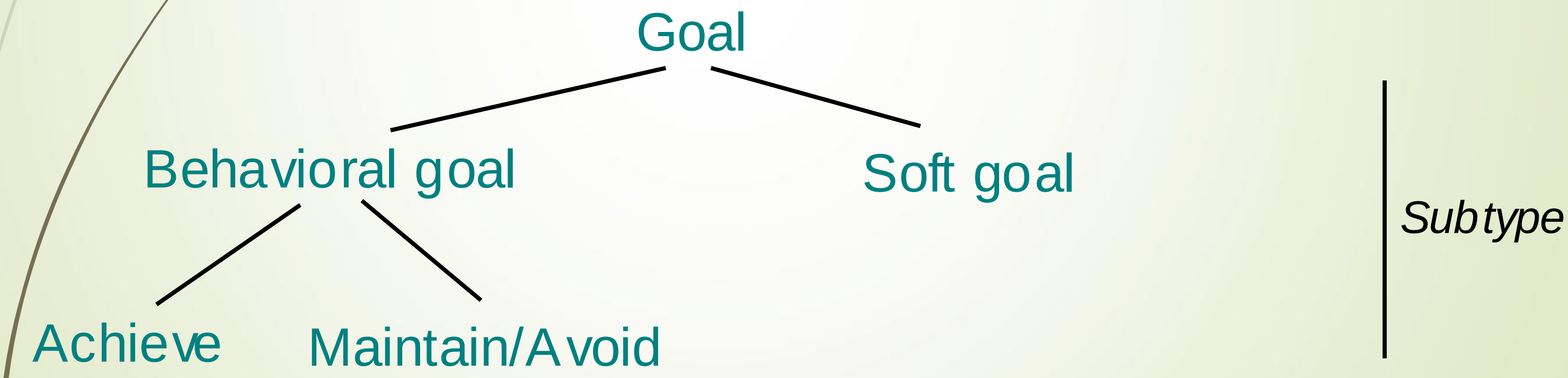
  "Train left when doors open at destination" ↔ *Passenger*

Goal types

Behavioral goals: prescribe behaviors

vs.

Soft goals: state preferences among alternative behaviors



Goal

Behavioral goal

Soft goal

Achieve

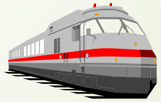
Maintain/Avoid

Subtype

Goal types: behavioral goals

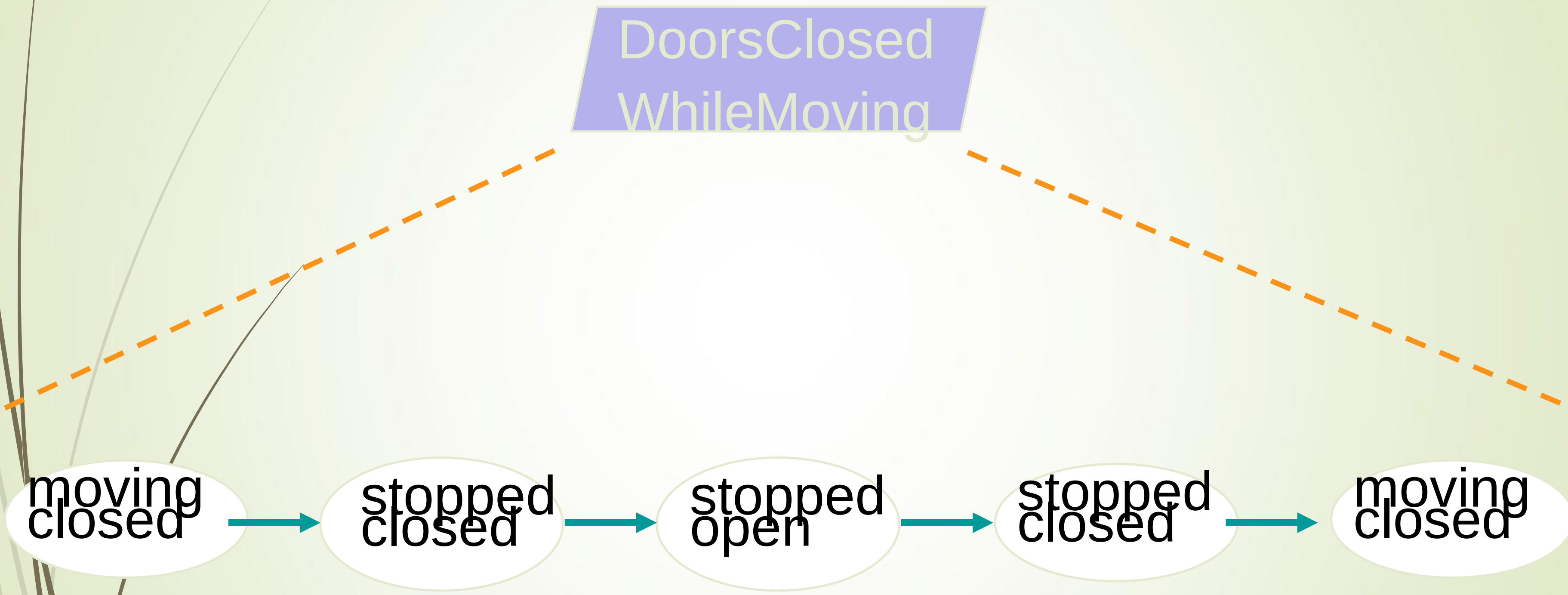
- Prescribe intended system behaviors declaratively
  - implicitly define maximal sets of admissible agent behaviors
- Can be satisfied in a clear-cut sense: YES *or* NO
  - goal satisfaction, formal analysis
- Used for building operation models to meet them

"Worst-case stopping distance maintained"



"Reminder sent if book not returned on time"

Behavior goals prescribe sets of desired behaviors



DoorsClosed WhileMoving

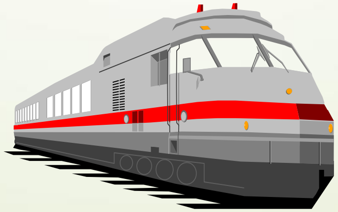
moving closed

stopped closed

stopped open

stopped closed

moving closed



Behavioral goals: subtypes and specification patterns

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

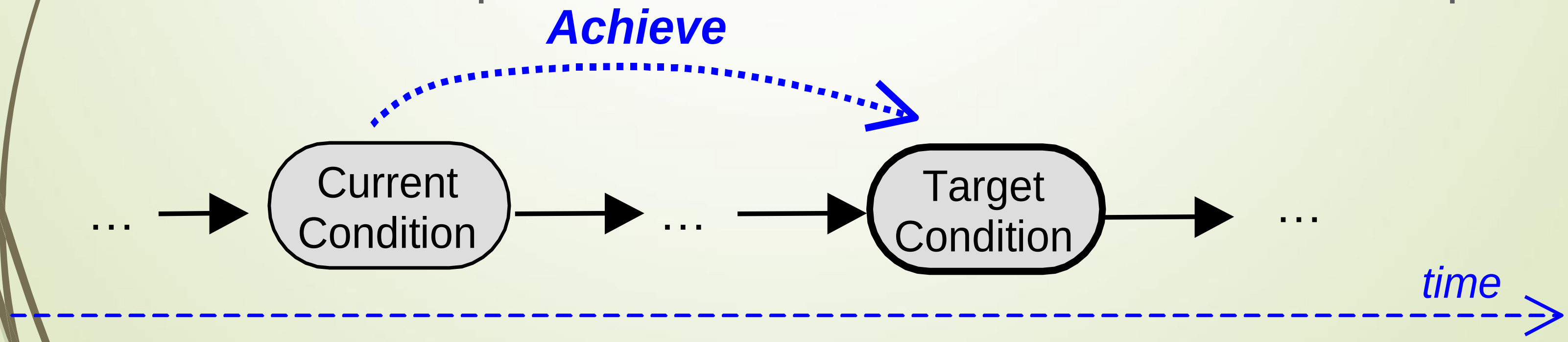
Achieve [BookRequestSatisfied]:

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 (2)

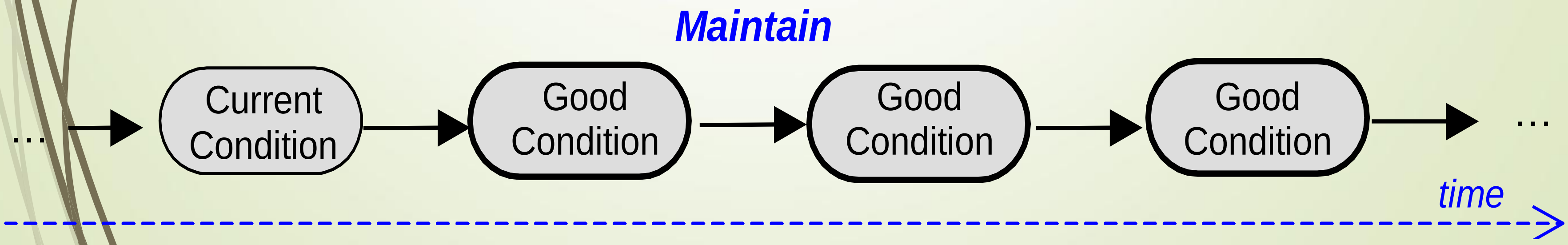
- ***Maintain*** [GoodCondition]:
  - [**if** CurrentCondition **then**] **always** GoodCondition
  - **always** (**if** CurrentCondition **then** GoodCondition)

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”

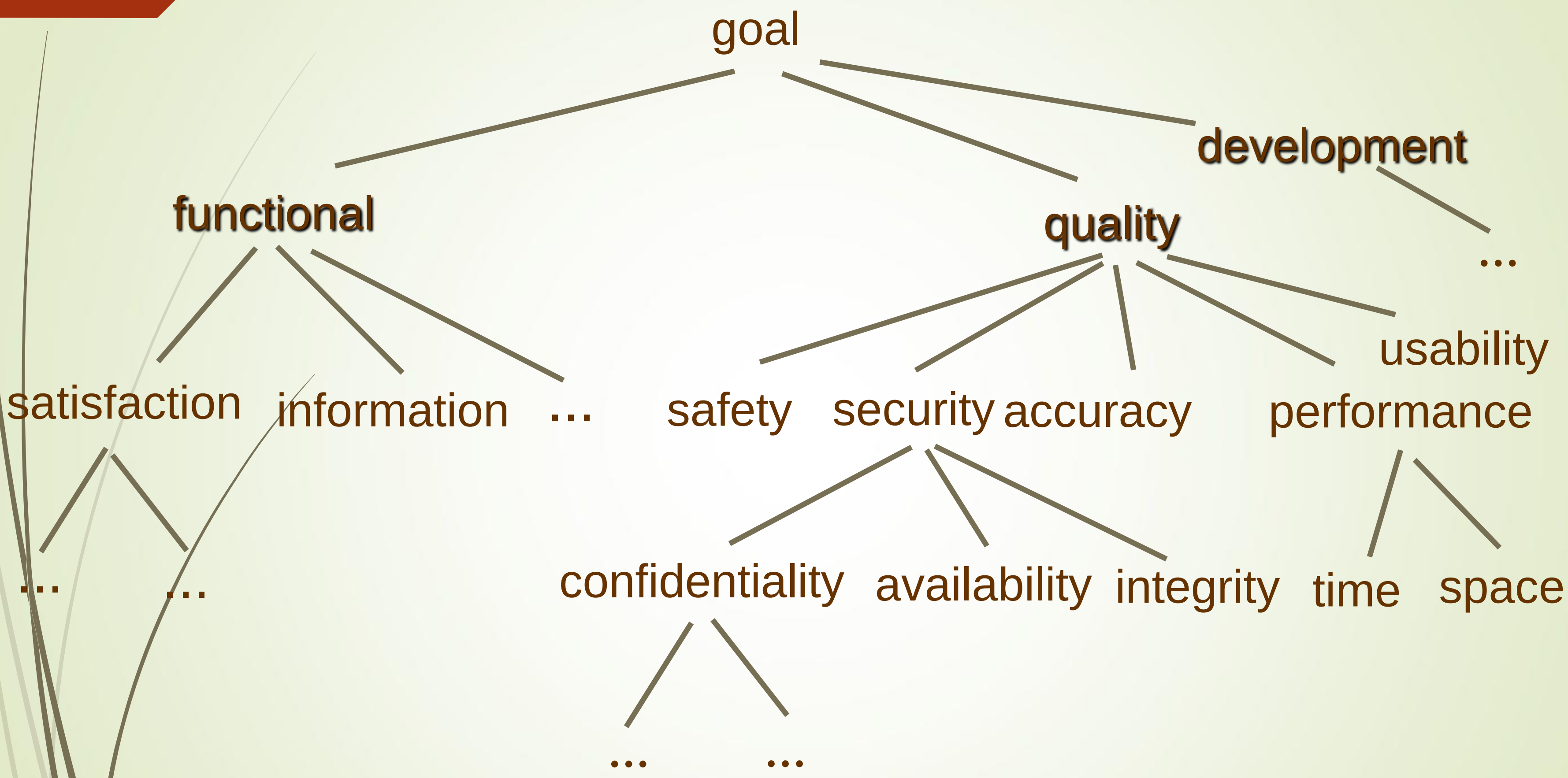
      “Train acceleration computed”

      ”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)



goal

functional

quality

development

satisfaction

information

…

safety

security

accuracy

usability

performance

…

…

…

confidentiality

availability

integrity

time

space

…

…

Using goal types & categories

- Lightweight specification patterns
- Heuristic rules for elicitation, validation, reuse, conflict management, ...
  - "Is there any conflict between ***Information*** goals and ***Confidentiality*** goals?"

"***Confidentiality*** goals are ***Avoid*** goals on sensitive info"

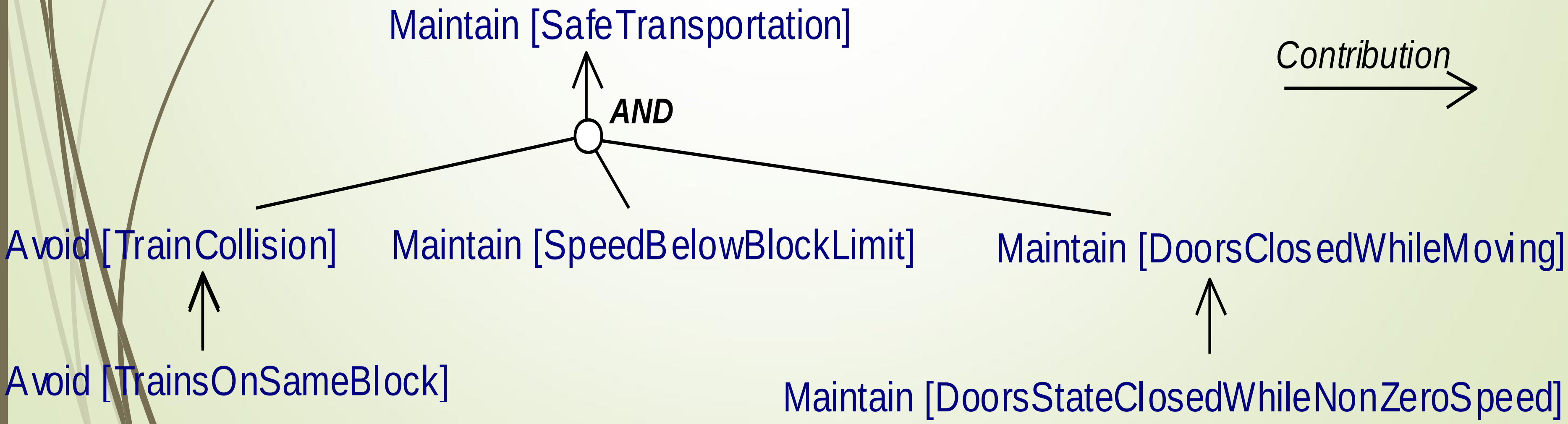
"***Safety*** goals have **highest priority** in conflict resolution"

More specific types & categories

⇒ more specific heuristics

The central role of goals in the RE process

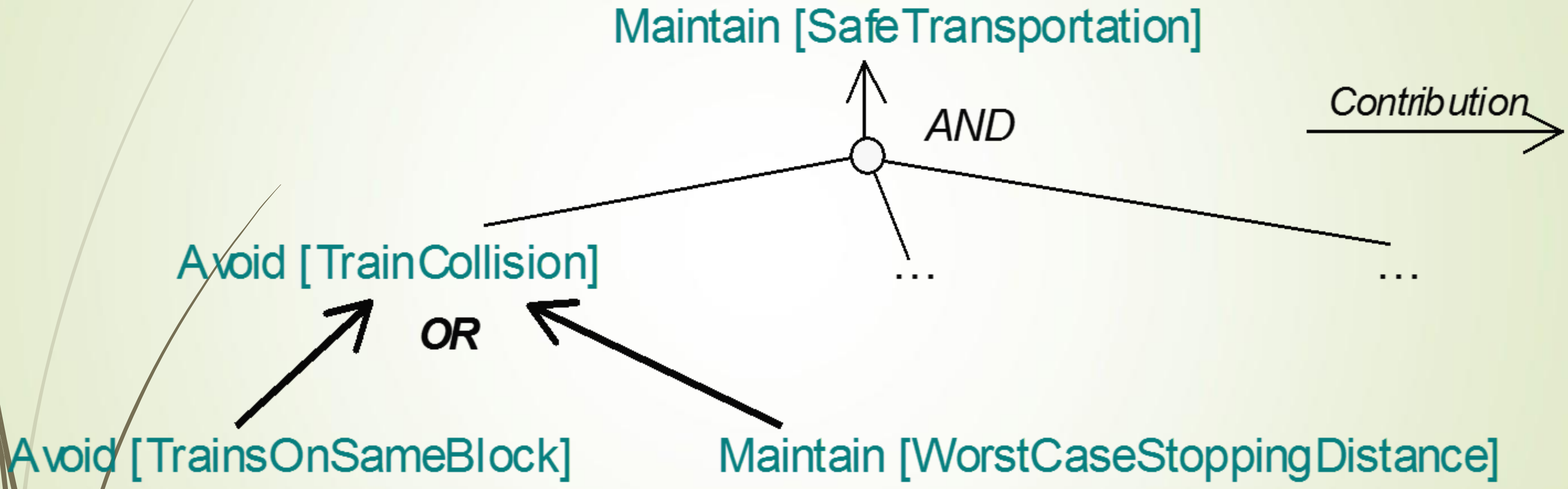
- Goal refinement/abstraction as structuring mechanism
  - shows contribution links among goals
  - drives elaboration of reqs (subgoals)
  - provides rationale for reqs (parent goals)
  - rich traceability: strategic objectives → technical requirements
  - can be used to structure reqs document



Maintain [SafeTransportation]
Contribution
AND
Avoid [TrainCollision]
Maintain [SpeedBelowBlockLimit]
Maintain [DoorsClosedWhileMoving]
Avoid [TrainsOnSameBlock]
Maintain [DoorsStateClosedWhileNonZeroSpeed]

The central role of goals in the RE process

- Goal **OR**-refinement → capture of alternative options


Maintain [SafeTransportation]
AND
Contribution
Avoid [TrainCollision]
...
...
OR
Avoid [TrainsOnSameBlock]
Maintain [WorstCaseStoppingDistance]

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

- Support for evolution management

  higher-level goals → more stable concerns

  ⇒ multiple system versions within single model ...

  - common parent goals
  - different OR-branches

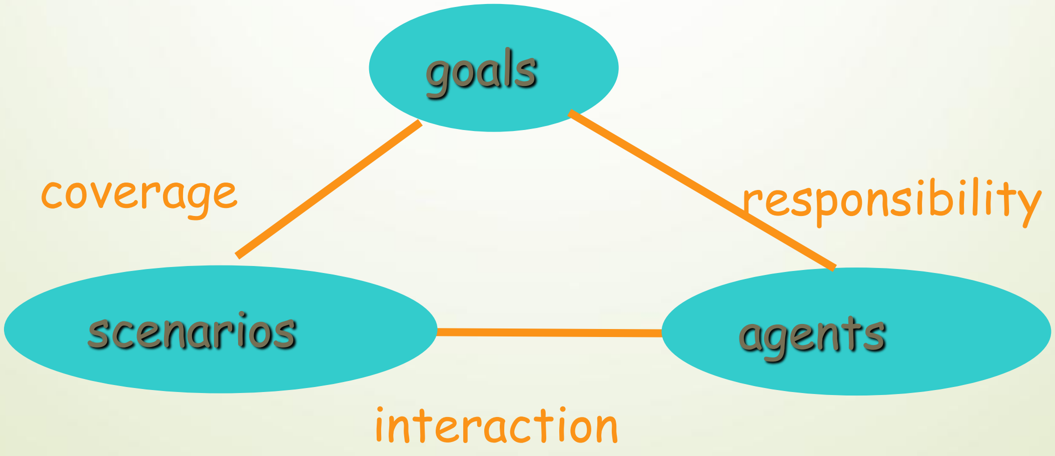
- Roots for conflict detection & resolution

- Anchors for risk management

Avoid frequent misconceptions

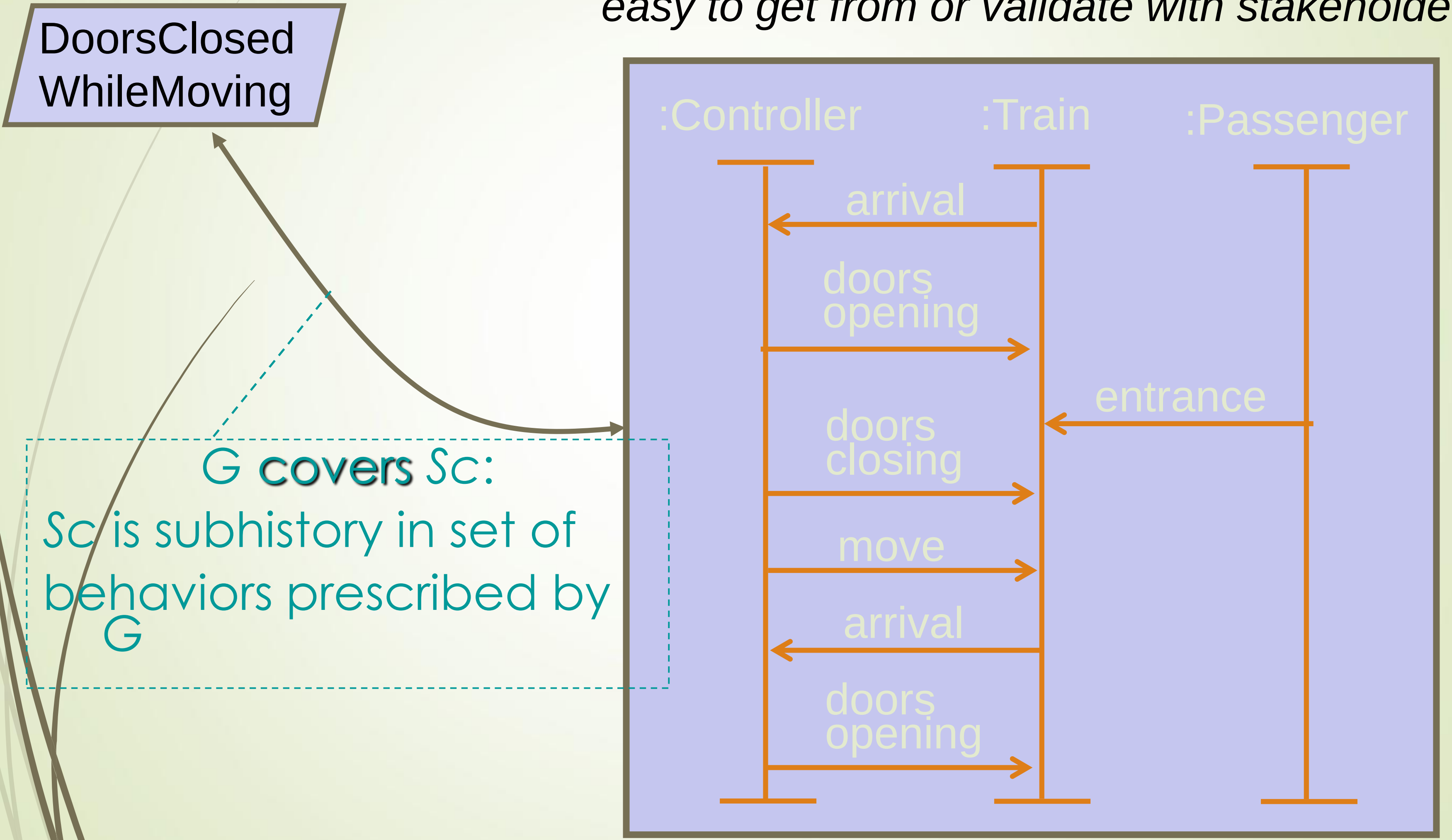
- Goal-oriented ≠ top-down
  - bottom-up elaboration as well (goal abstraction)
- Goal-oriented ⇒ agent-oriented,
  scenario-oriented

the magic RE triangle:



Scenarios as concrete vehicles for goal elicitation/validation

easy to get from or validate with stakeholders



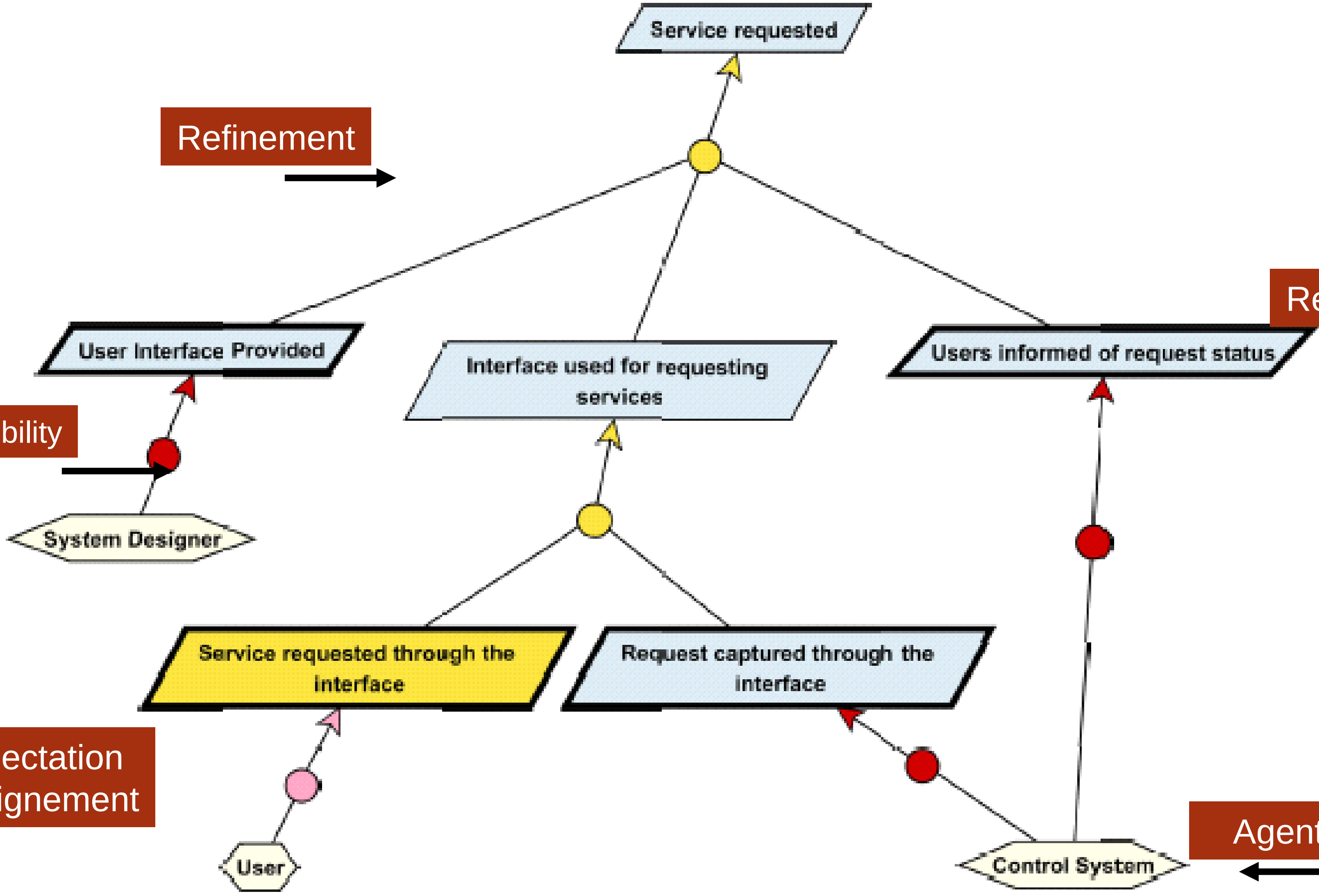
G covers *Sc*:

Sc is subhistory in set of behaviors prescribed by G

Goals relationships

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

Generic Goal “Requested service”


Service requested
User Interface Provided
Interface used for requesting services
Users informed of request status
System Designer
Service requested through the interface
Request captured through the interface
User
Control System

Refinement

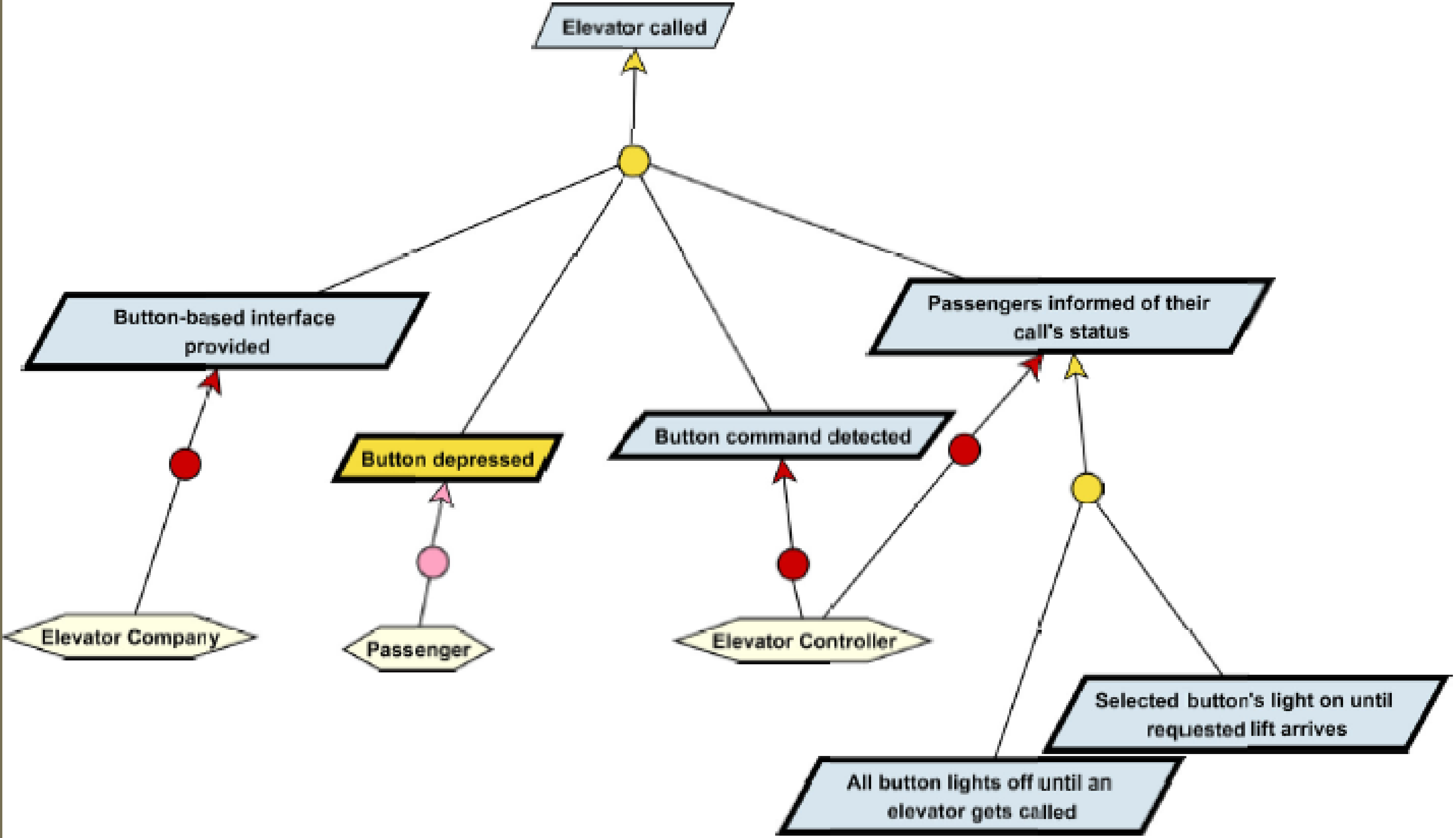
Req

Responsibility

Expectation assignement

Agent

Applying the generic pattern to the elevator system


Elevator called
Button-based interface provided
Passengers informed of their call's status
Button depressed
Button command detected
Elevator Company
Passenger
Elevator Controller
Selected button's light on until requested lift arrives
All button lights off until an elevator gets called

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

Completeness criteria

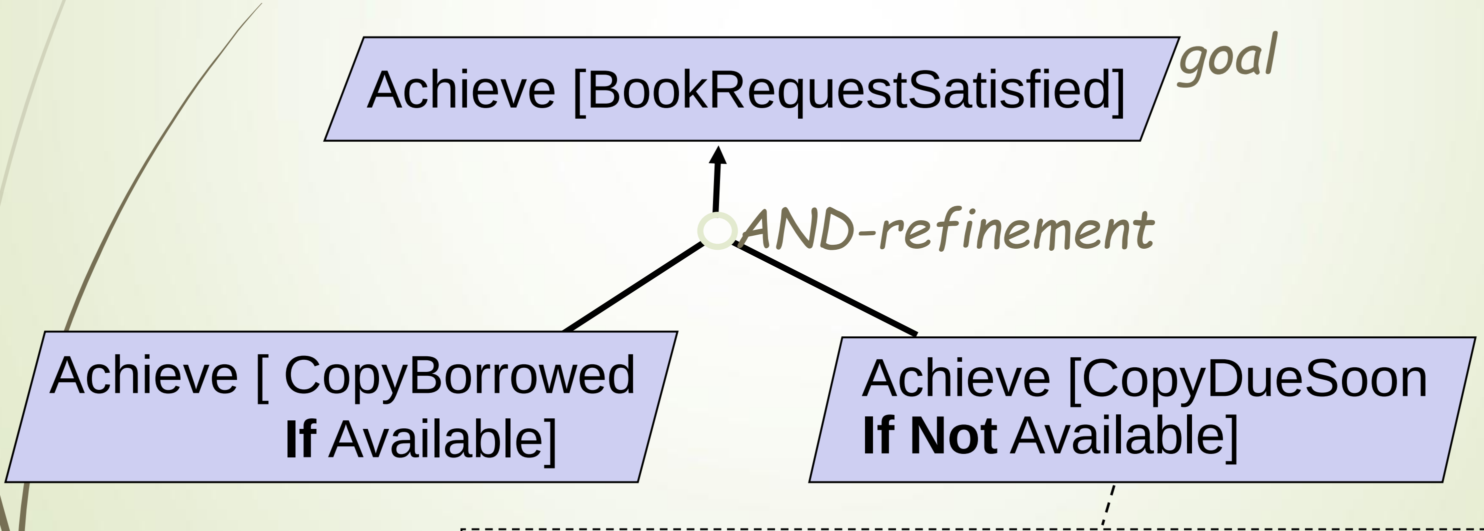
- Criterion 1:
  - A goal model is said to be complete with respect to the refinement relationship if and only if every leaf goal is either an expectation, a domain property or a requirement
- Criterion 2:
  - A goal model is said to be complete with respect to the responsibility relationship if and only if every requirement 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

Goal refinement

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

  The set $\{G_1, \ldots, G_n\}$ is called **refinement** of G

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



Achieve [BookRequestSatisfied]

goal

AND-refinement

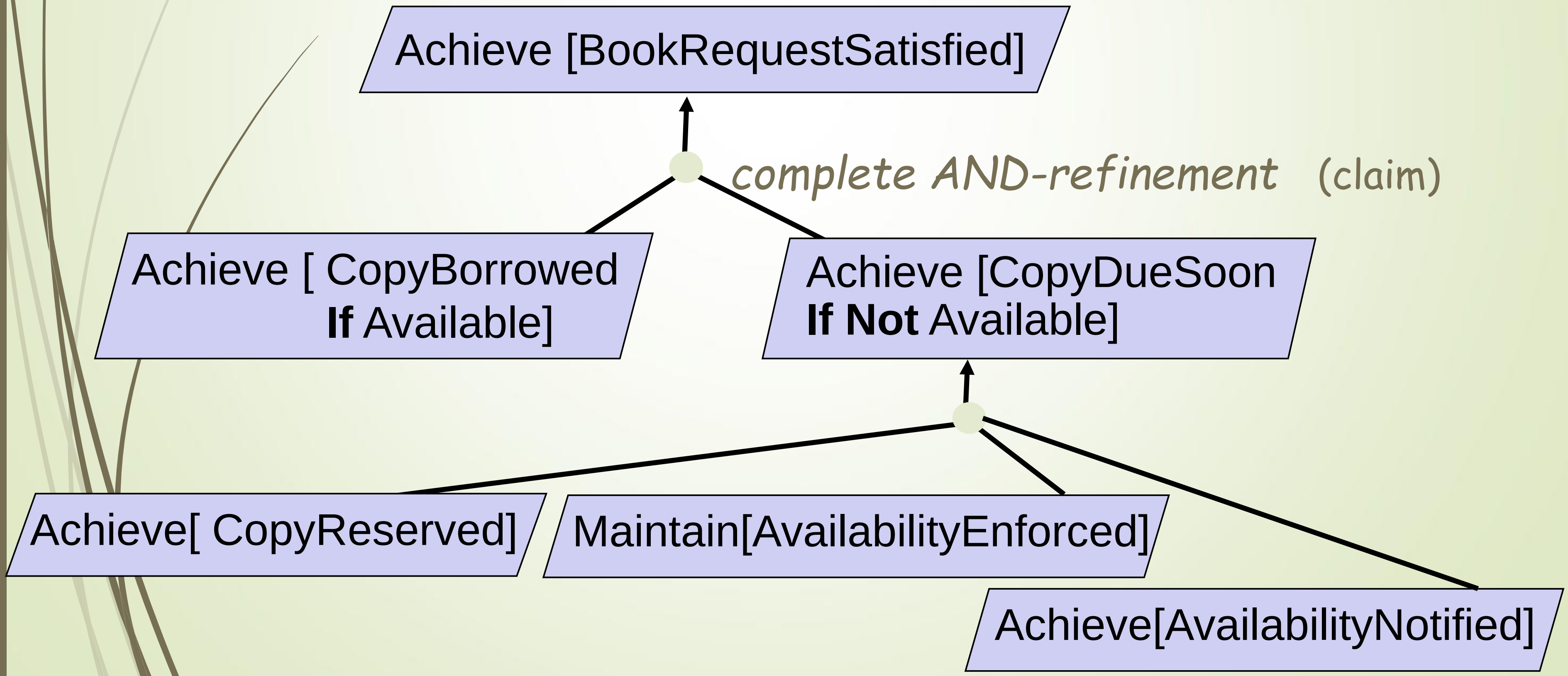
Achieve [CopyBorrowed **If** Available]

Achieve [CopyDueSoon **If Not** Available]

Def *In case a requested book has no copy available for check out, a copy of that book shall be made available within 2 weeks for check out by the requesting patron.*

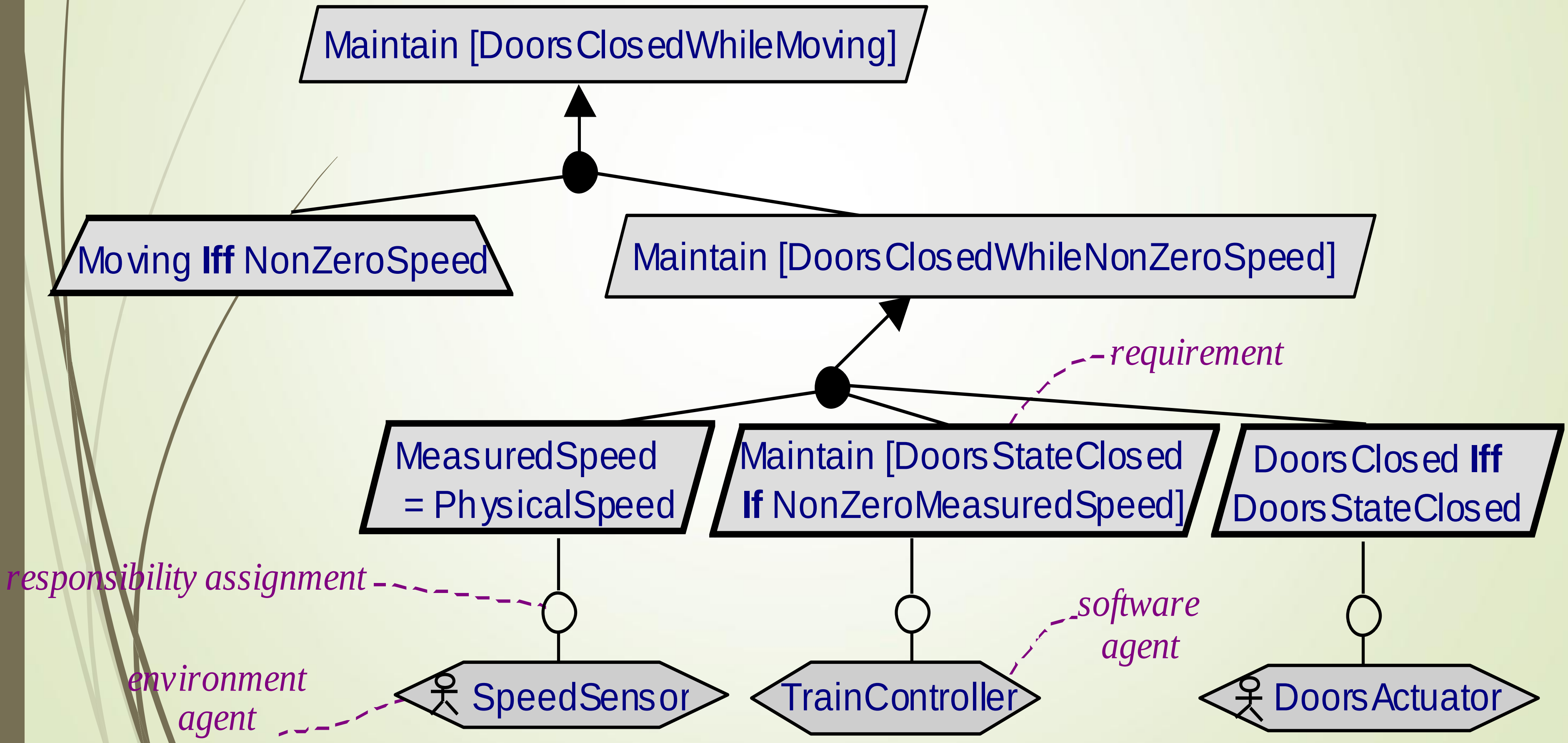
AND-refinements should be complete

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


Achieve [BookRequestSatisfied]
complete AND-refinement (claim)
Achieve [CopyBorrowed **If** Available]
Achieve [CopyDueSoon **If Not** Available]
Achieve[CopyReserved]
Maintain[AvailabilityEnforced]
Achieve[AvailabilityNotified]

Refinement trees

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


Maintain [DoorsClosedWhileMoving]
Moving **Iff** NonZeroSpeed
Maintain [DoorsClosedWhileNonZeroSpeed]
requirement
MeasuredSpeed = PhysicalSpeed
Maintain [DoorsStateClosed **If** NonZeroMeasuredSpeed]
DoorsClosed **Iff** DoorsStateClosed
responsibility assignment
software agent
environment agent
SpeedSensor
TrainController
DoorsActuator