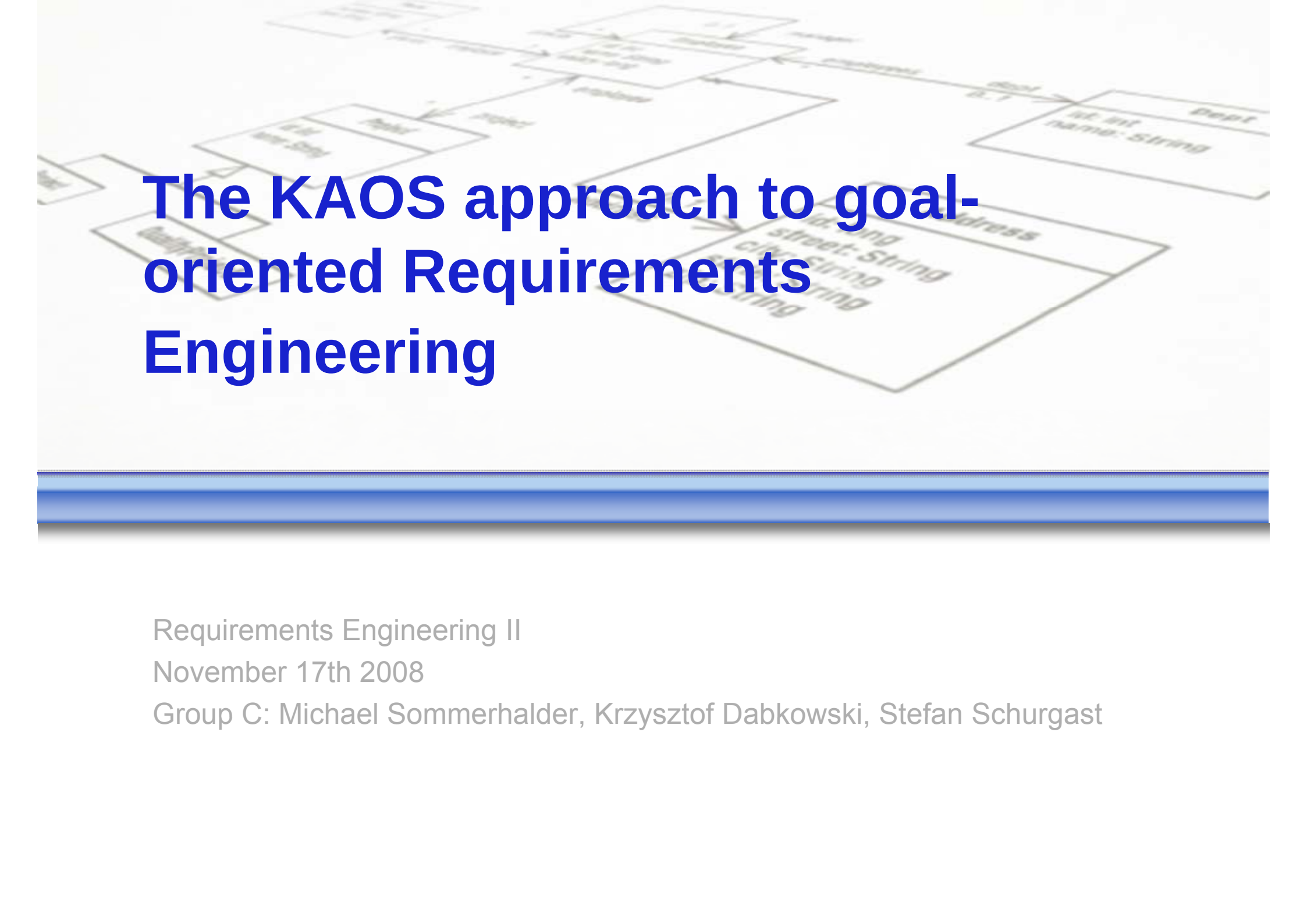




The KAOS approach to goal-oriented Requirements Engineering

Requirements Engineering II

November 17th 2008

Group C: Michael Sommerhalder, Krzysztof Dabkowski, Stefan Schurgast

Table of Contents

- What is KAOS
- The four steps in modelling with KAOS
- A real world example

What is KAOS? - 1

- Knowledge Acquisition in automated Specification
 - ◆ Machine learning technology (learning-by-instruction, learning-by-analogy)
- A goal-oriented approach of capturing requirements combined of three components:
 - ◆ Conceptual model
 - A meta-model of both functional and non-functional reqs
 - Has to be rich enough to capture objectives, constraints, agents (both human and automated), events on entities
 - Represented as a conceptual graph of nodes and links

What is KAOS? - 2

- ◆ Acquisition strategy
 - Composition of steps in order to create a meta-model
- ◆ Acquisition assistant
 - Automated support according to a certain acquisition strategy
 - Build of two repositories:
 - Requirements database – holds the gradually developed requirements model
 - Requirements knowledge base – contains *Domain-level knowledge* (concerns and requirements of application domain) and *meta-level knowledge* (properties of the abstraction of the meta-model, e.g. a constraint can be violated only if...)

Step 1 and 2 – goals and objects - 1

- Elaboration of the goal structure = identifying:
 - ◆ “**SystemGoals**”, category, pattern → Reduction
 - ◆ **Objects** – definitions, features
 - ◆ Possible conflicts – **Conflict metarelationship**
- 1. Define Goals and form a tree-model
- 2. Derive Objects, Relationships and Attributes from Goal model
- 3. Find conflicts and prioritize them

Step 1 and 2 – goals and objects - 2

- The process is not sequential
- Reuse of generic goals
- Consider alternative reduction to minimize costs/number of conflicts

SystemGoal Achieve [ResourceRequestSatisfied]

InstanceOf SatisfactionGoal

Concerns User, Resource, Using, ...

FormalDef ($\forall$ u: User, res: Resource, rep:Repository)

.....

ReducedTo DetailedGoal_1, DetailedGoal_2, ...

Step 3 - Agents

- Identify potential agents in the composite system
 - ◆ Person, system, program
- Identify/define responsibilities in goal hierarchy for each agent
- Define own goals for agents
- At the end of the process consider the relevance of the capabilities of automated agents

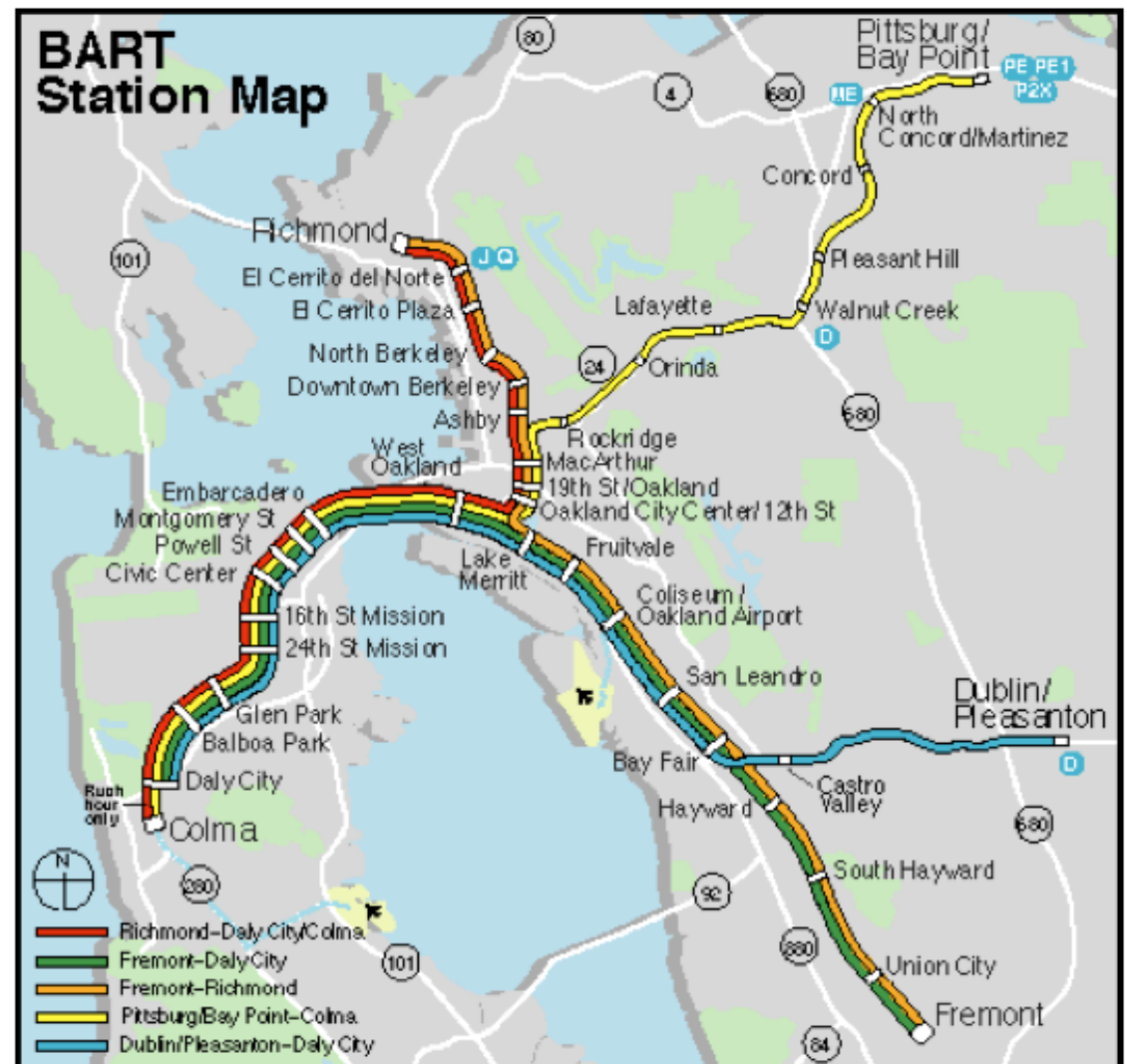
Step 4 – Operation model

- Defining constraints to achieve a goal
- Like goal-modeling – non-trivial task to be optimal
 - ♦ Formal: traceability to goal has to be provided
 - ♦ Goal reduction process for constraints
 - ♦ E.g.: to fulfill the goal “Safety during transportation”:

$\forall$ elevator:
elevator.state='moving'
 $\Rightarrow$ door.state='closed'

A real world example: BART

Bay Area Rapid Transit (BART) wants new software to control their trains



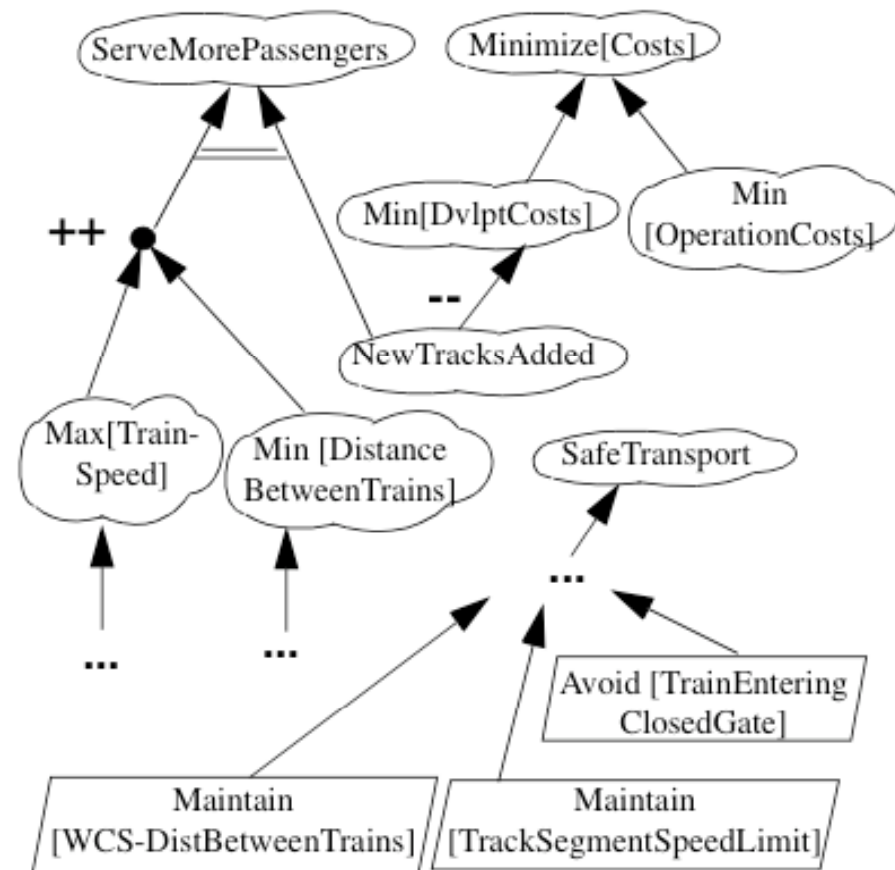
http://faculty.ist.unomaha.edu/winter/publications/2001/Bay_Area_Rapid_Transit_District_Advanced_Automated_Train_Control_System_Case_Study_Description.pdf

Goals

- Serve more passengers
- Minimize costs
- Pay close attention to safety-critical issues
 - ◆ Distance between trains
 - ◆ Entering gates
 - ◆ Speed

Step 1: Goal model - 1

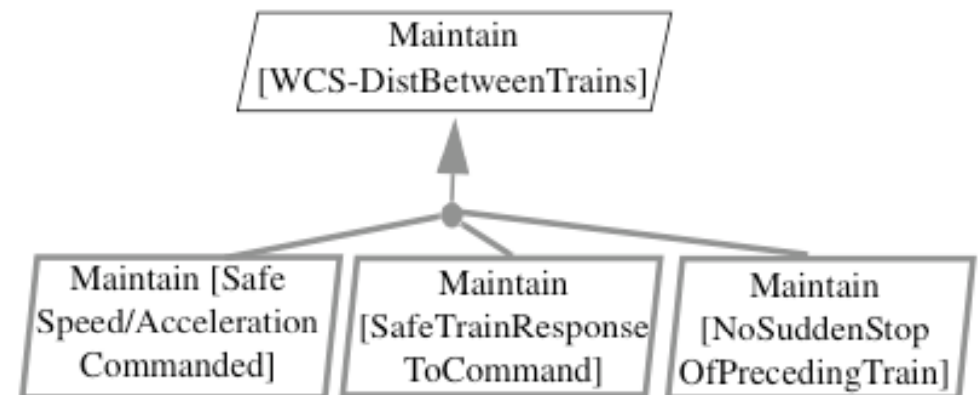
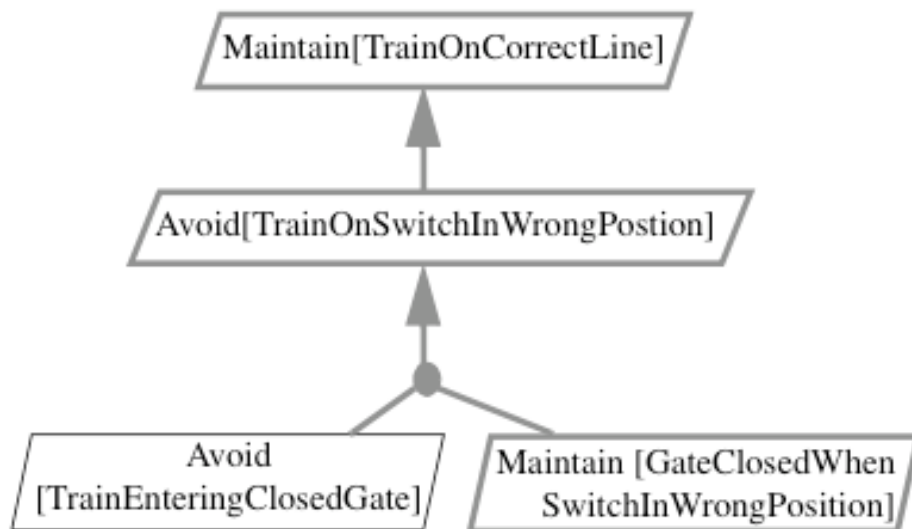
- Collect goals and achieve hierarchies by linking them
- Result: rough overview



All examples: [Lamsweerde 2001]

Step 1: Goal model - 2

- Refine goals by asking
Why? How?



Step 2: Object model - 1

- Formalize

Goal Maintain[TrackSegmentSpeedLimit]

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

FormalDef $\forall \text{tr: Train, s: TrackSegment} : \text{On}(\text{tr}, \text{s}) \Rightarrow \text{tr.Speed} \leq \text{s.SpeedLimit}$

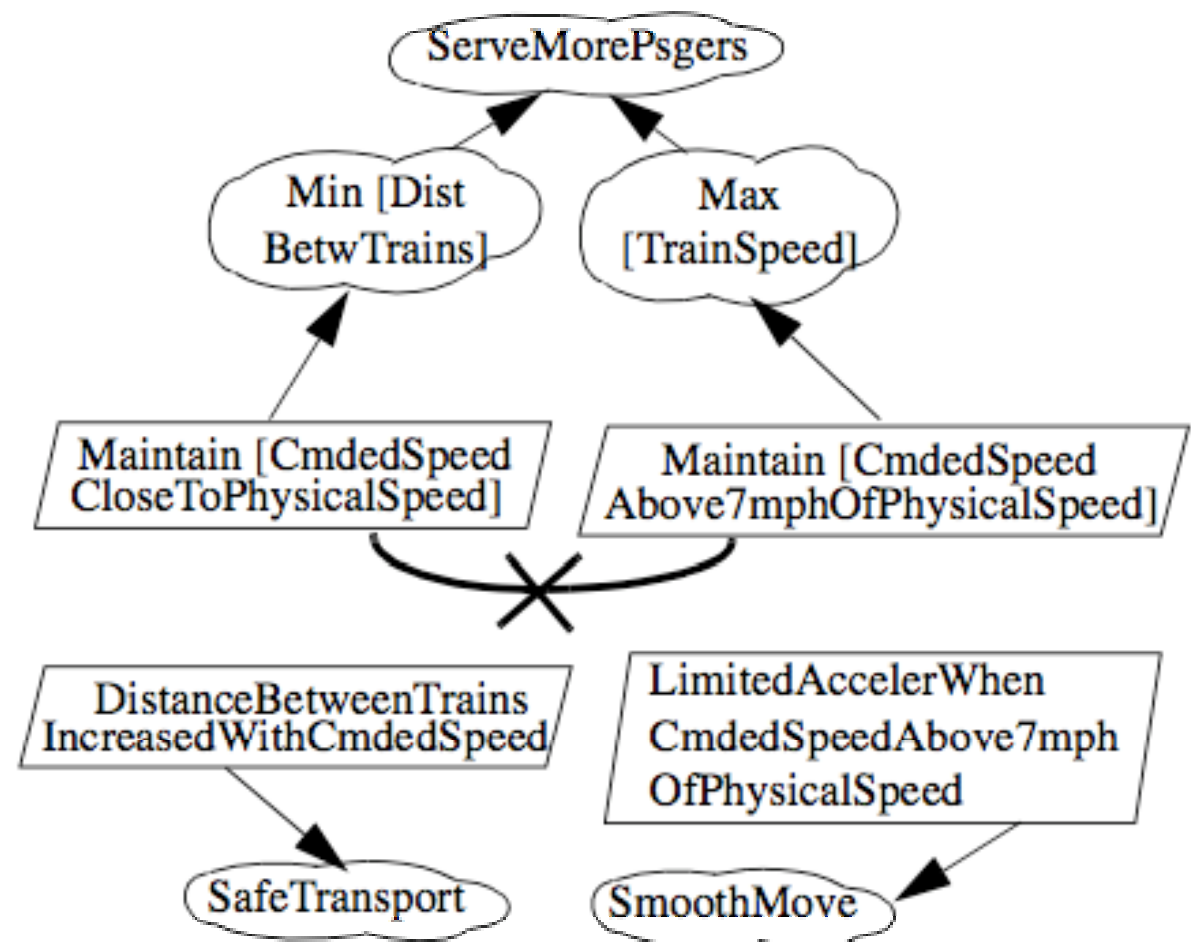
Step 2: Object model - 2

- Derive object model



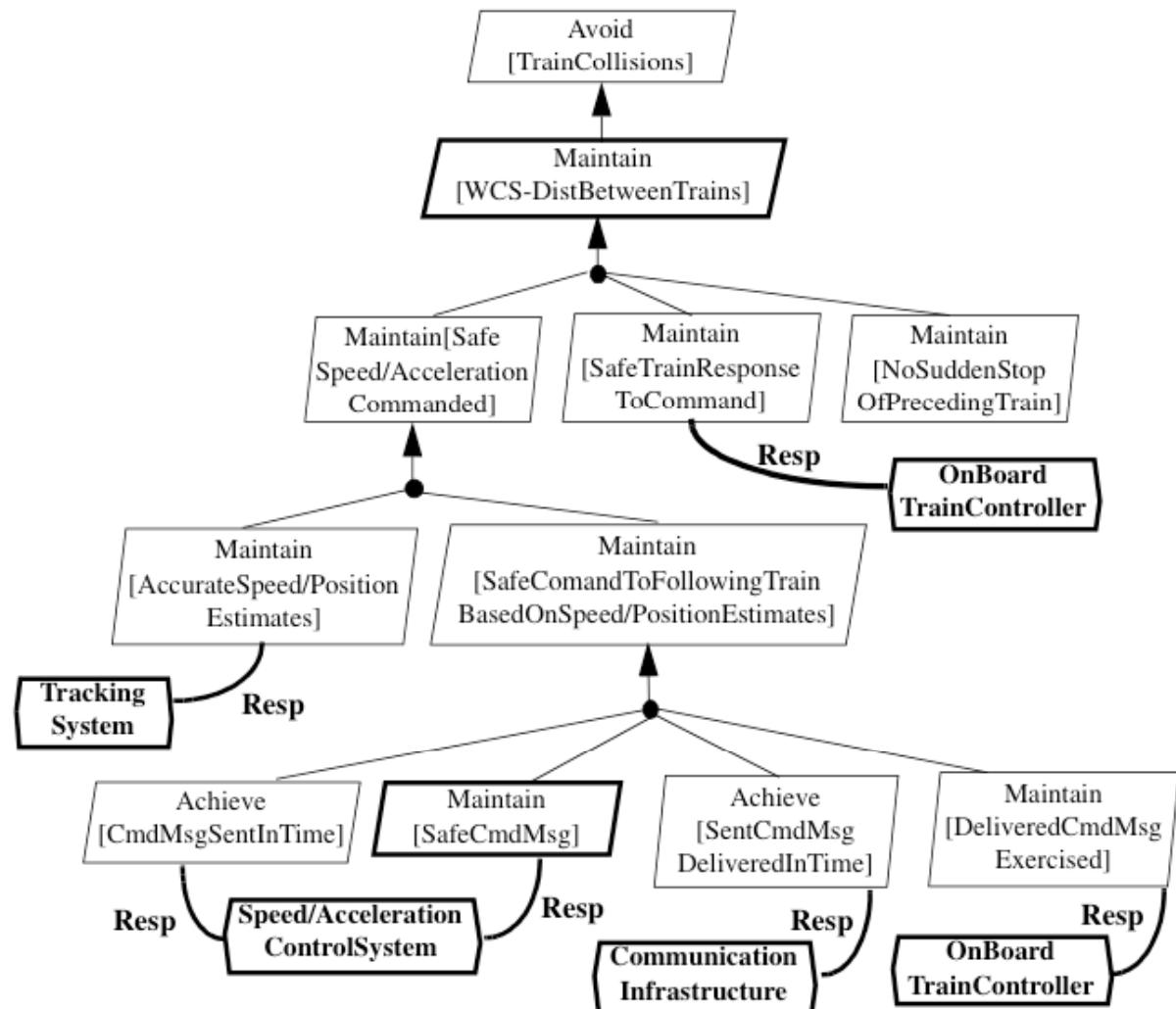
Step 2: Object model - 3

- Resolving conflicts
 - ◆ Identify conflicting goals
 - ◆ Prioritize
 - ◆ Adjust



Step 3: Agent responsibility model

- Identify agents
- Add agents to goals they're responsible for



Step 4: Operation model - 1

- Example: How to operationalize the following goal?

Goal Maintain [DoorsClosedWhileMoving]

FormalDef $\forall$ tr: Train, loc, loc': Location

$\text{At}(\text{tr}, \text{loc}) \wedge \text{o} \text{ At}(\text{tr}, \text{loc}') \wedge \text{loc} \neq \text{loc}'$

$\Rightarrow \text{tr.Door} = \text{'closed'} \wedge \text{o} (\text{tr.Door} = \text{'closed'})$

Step 4: Operation model - 2

Operation Move

Input tr: Train; loc, loc': Location ; **Output** At

DomPre At (tr, loc) $\wedge$ loc $\neq$ loc'

DomPost At (tr, loc')

RequiredPre for DoorsClosedWhileMoving:

tr.Doors = 'closed'

RequiredPost for DoorsClosedWhileMoving:

tr.Doors = 'closed'

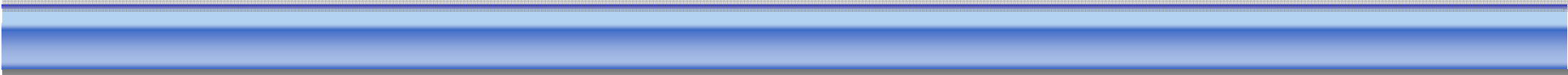
Literature

[Dardene, Lamsweerde, Fickas 1993]: van Lamsweerde, A. (1993).

Goal-directed Requirements Acquisition

[Lamsweerde 2001]: van Lamsweerde, A. (2001).
Goal-Oriented Requirements Engineering: A Guided Tour. Proceedings of the 5th IEEE International Symposium on Requirements Engineering (RE'01), Toronto, Canada. 249-261.

Thanks for your attention!

A decorative horizontal bar with a blue gradient, transitioning from a lighter blue on the left to a darker blue on the right, located at the bottom of the slide.