Goal-Oriented Requirements Engineering:

from System Objectives

to UML Models

to Precise Software Specifications

Axel van Lamsweerde

University of Louvain

B-1348 Louvain-la-Neuve (Belgium)

avl@info.ucl.ac.be

May 2003

The requirements problem: the good old time...

- Poor requirements are ubiquitous ...

  "requirements need to be *engineered*
  and have continuing review & revision"

  (Bell & Thayer, empirical study, 1976)

- Prohibitive cost of late correction ...

  "up to 200 x cost of early correction"

  (Boehm, 1981)

- RE is hard & critical ...

  "hardest, most important function of SE is the
  iterative *extraction* & *refinement* of requirements"

  (Brooks, 1987)

2

The requirements problem: more recently ...

- Survey of 350 US companies, 8000 projects
  - success: 16 %
  - failure: 33 %
  - so so: 51 %
    (partial functionalities, excessive costs, big delays)

major source of failure:
poor requirements engineering @ 50% responses

(Standish Group, 1995)

3

The requirements problem: more recently ...

Major source of failure:
poor requirements engineering @ 50% responses:

- lack of user involvement 13%
- incomplete requirements 13%
- changing requirements 9%
- unrealistic expectations 10%
- unclear goals 5%

www.standishgroup.com/chaos.html

4

The requirements problem: more recently ...

- Survey of 3800 EUR organizations, 17 countries

  main software problems are in...
  - requirements specification

    > 50% responses
  - requirements management

    50% responses

  (European Software Institute, 1996)

5

Outline

- Requirements engineering
- Goal-oriented requirements engineering
- Building rich system models for RE
  - Modeling & specification techniques

    The goal model

    The object model

    The agent model

    The operation model
  - A goal-oriented RE method in action
- From requirements to software specs
- Conclusion

6

Requirements Engineering... *what?*

Ross'77

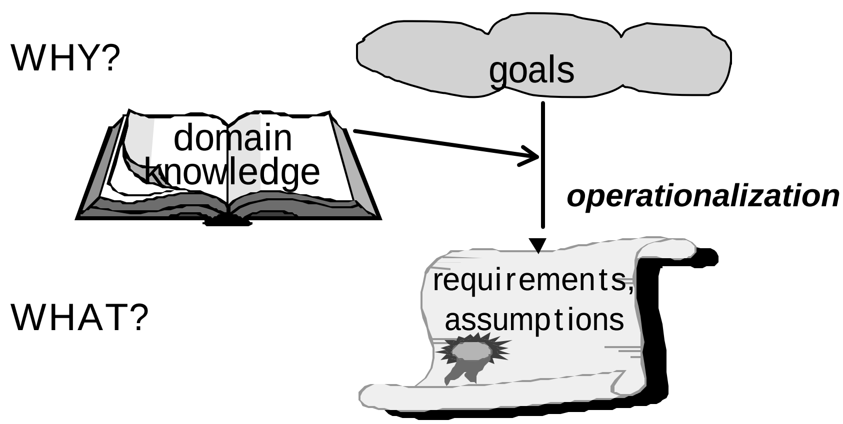
- "Requirements definition must say
  - **why** a system is needed, based on current or foreseen conditions,
  - what system features will satisfy this context,
  - how the system is to be constructed"

Zave'97

- "RE is concerned with the real-world goals for, functions of, constraints on software systems; and with their
  - link to *precise specs* of sw behavior,
  - *evolution* over time & families"

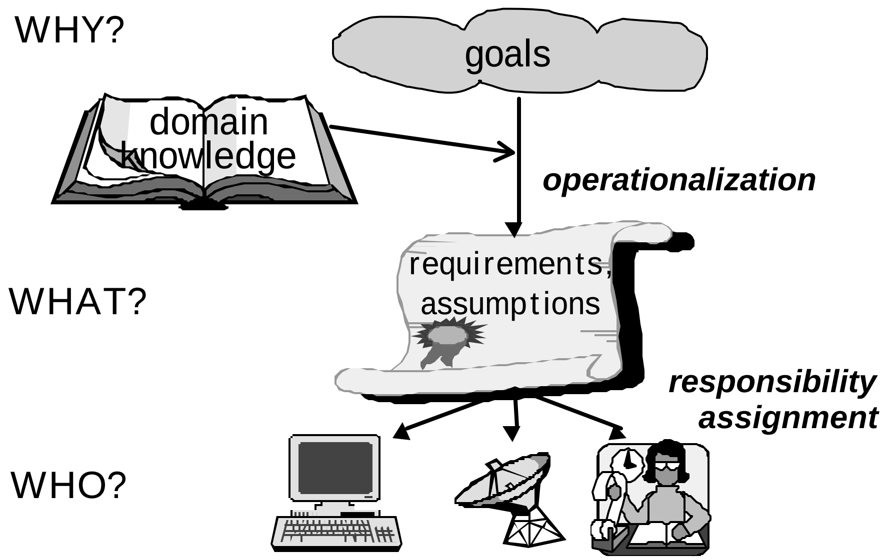
7

Requirements Engineering... *what?* (2)



8

Requirements Engineering... *what?* (2)



9

Requirements Engineering... *what?* (3)

- Broad scope
  - 2 systems: current, to-be
  - system-to-be = software + environment
  - hybrid environments:
    human organizations, policies
    devices, physical laws
- Multiple concerns
  - functional, quality, development → *conflicts*
- Multiple abstraction levels
  - high-level goals, operational details
- Multiple expression means
  - informal, graphical, formal

10

Requirements Engineering... *what?* (4)

- Multiple products
  - report on current system: domain concepts, current procedures, problems & deficiencies, opportunities
  - alternative proposals for system-to-be
  - development contract
  - requirements on software-to-be *in vocabulary of the domain/clients*
  - software specifications *in vocabulary of developers*
- Multiple parties involved, different background
  - organization stakeholders, domain experts, clients, subcontractors, analysts, developers, ...

    ® *conflicting viewpoints*

11

Requirements Engineering... *what?* (5)

- Multiple processes intertwined
  - domain analysis
  - elicitation of objectives, constraints, alternative system proposals, risks
  - negotiation, agreement
  - specification
  - spec analysis
  - documentation
  - evolution management

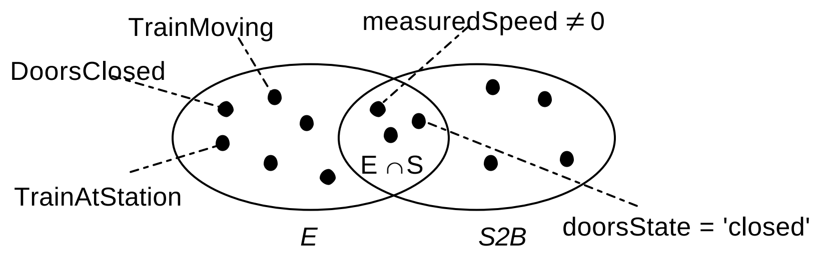
rich model = best interface between all processes

$\Rightarrow$ *system modeling is a core business*

12

Requirements Engineering... *what?* (6)

- Requirements vs. software specifications:
  the software-to-be (S2B) & its environment (E) ...
  - share some common phenomena
  - other phenomena are owned by E
  - other phenomena are owned by S2B



13

Requirements Engineering... *what?* (7)

- Requirements are *prescriptive* assertions formulated in terms of *environment* phenomena *(not necessarily shared)*

  TrainMoving $\Rightarrow$ DoorsClosed

- Software specifications are *prescriptive* assertions formulated in terms of *shared* phenomena

  measuredSpeed $\neq 0 \Rightarrow$ doorsState = 'closed'

- Domain properties are *descriptive* assertions assumed to hold in the domain

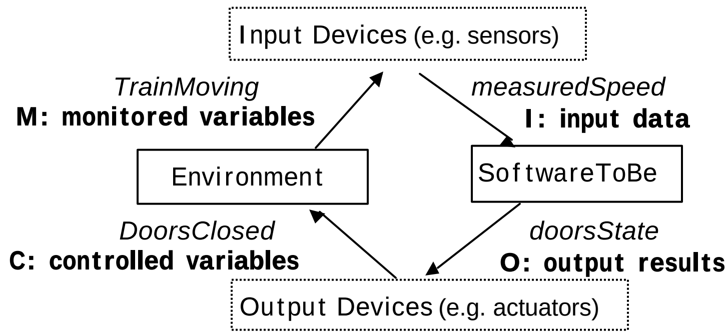
  TrainMoving $\Leftrightarrow$ measuredSpeed $\neq 0$

(Jackson, 1995 & Parnas, 1995)

14

Requirements Engineering... *what?* (8)

- Requirements vs. software specifications: 4-variable model (Parnas, 1995)



$Req \subseteq M \times C$

$Spec \subseteq I \times O$

Spec = *Translation* (Req) such that

{Spec, Dom} |= Req

15

Requirements Engineering... *what?* (9)

- Target qualities for RE process
  - completeness of reqs, specs, dom assumptions
  - consistency of reqs, specs, dom props
  - adequacy of reqs, specs, dom assumptions
  - precision of reqs, specs, dom assumptions
  - relevance of reqs
  - understandability by consumers of reqs, specs
  - good structuring of requirements document
  - modifiability of reqs, specs, dom assumptions
  - traceability of reqs, specs, dom assumptions
  - measurability of reqs, specs, dom assumptions

16

Requirements Engineering... *what?* (10)

Þ wide variety of deficiencies

- incompleteness (critical error!)
- inconsistency (critical error!)
- inadequacy (critical error!)
- ambiguity (critical error!)
- unintelligibility
- wishful thinking
- poor structure
- overspec
- noise

requirements errors are
numerous, persistent, expensive, dangerous

17

Requirements Engineering... *why?*

◆ Critical impact, multiple stakes ...

- legal: contractual commitment client-provider
- economic: cf. cost of requirements errors
- social: *from* user satisfaction *to* degradation of working conditions *to* system rejection
- ethical: wrt safety, health & welfare (IEEE code of ethics)
- certification: mastered RE process required by many quality standards & certification authorities (CMM, ISO, SPICE, ...)

18

Requirements Engineering... *why?* (2)

- Critical impact, multiple stakes (cont'd) ...
  - technical: requirements provide the basis for...
    - acceptance test data generation
    - architectural design
    - software documentation
    - software evolution
    - project management

requirements document = main interface between multiple parties

19

Requirements Engineering: to sum up & move on...

- A few confusions to get rid of:
  - requirements are *not* domain properties
  - requirements are *not* software specifications
  - requirements are problem formulations, *not* solution formulations (i.e. design specs)
  - RE is *not* translation of pre-existing problem formulations
  - composition is *not* necessarily conjunction
  - "precise" does *not* necessarily mean "formal"
  - a set of notations is *not* sufficient for a "method"

20

Outline

- Requirements engineering
- Goal-oriented requirements engineering
- Building rich system models for RE
  - Modeling & specification techniques
    - The goal model
    - The object model
    - The agent model
    - The operation model
  - A goal-oriented RE method in action
- From requirements to software specs
- Conclusion

21

Goal orientation...

- found in traditional methodologies for system engineering ("context analysis", "definition study", "participative analysis", ...)
- addressed by IEEE-Std-830
- ignored by UML, "but needed" say UMLers (Fowler, Cockburn)
- increasingly considered in RE research

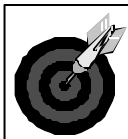
22

Goal-oriented RE ... *what?*

= *use of goals for requirements...*

- elicitation
- elaboration
- structuring
- specification
- analysis
- negotiation
- documentation
- evolution

23



WHAT are goals ?

- **Objectives** to be achieved by the system ...
  - prescriptive statements of intent
    (unlike dom props)
  - "system":
    software + environment
    current system, system-to-be

24

WHAT are goals ? (2)

- Different levels of abstraction ...
  - high-level goals

    strategic, coarse-grained, organization-wide

    "more passengers served" (train control)

    "effective access to state of the art" (library system)
  - low-level goals

    technical, fine-grained, design-specific

    "acceleration command sent every 3 secs"

    "reminder issued by end of loan period if no return"

25

WHAT are goals ? (3)

- Different types of concern ...
  - functional goals: about expected services

    "train acceleration computed"

    "book request satisfied"
  - non-functional goals: about...

    quality of service: security, safety, accuracy, performance, cost, usability, ...

    "worst-case stopping distance maintained"

    "access to info about other borrowers denied"

    quality of development: adaptability, interoperability, reusability, ...

26

WHAT are goals ? (4)

- Achieving goals requires agent cooperation
  - agent = role (rather than individual) :
    responsible for goal achievement
    software (existing, S2B), device, human

    "safe transportation" «
    on-board train controller, tracking system, station computer, passengers, train driver, ...

    "book copy returned on shelves" « borrower, staff, library software
  - the more fine-grained a goal is, the less agents are required for its achievement

    "acceleration command sent every 3 secs" « station computer

    "reminder issued by end of loan period" « library software

27

WHAT are goals ? (5)

- Agent responsible for goal Þ
  must restrict behaviors (Feather'87)
  goal must be *realizable* (Letier'01)
- Goal assigned to single agent in *software-to-be*
  = requirement

  "maintain doors closed while non-zero speed"

  "loan coupon issued when book copy available"
- Goal assigned to single agent in *environment*
  = expectation (cannot be enforced by software)

  "get in when doors open at station"

  "book copy provided when loan coupon issued"

28

WHAT are goals ? (6)

- Goals may be owned by stakeholders (at process level)

"train frequency increased" (passengers)

"number of passengers increased" (train company)

® potential for conflicting viewpoints

"book copy returned within 2 weeks" (staff)

"book copy kept as long as needed" (borrowers)

(Robinson'89, Dardenne'93, Nuseibeh'94, Boehm'95, van Lamsw'98)

29

WHY are goals needed?

- Criterion for requirements completeness

REQ is complete if for all G:

{REQ, EXPECT, Dom} |= G

- Criterion for requirements relevance

r in REQ is pertinent if for some G:

r is used in {REQ, EXPECT, Dom} |– G

(Yue'87)

30

WHY are goals needed? (2)

- Goals drive the elaboration of requirements to support them

  (Ross'77, Dardenne'91, Rubin'92, Anton'98, Kaindl'00, ...)

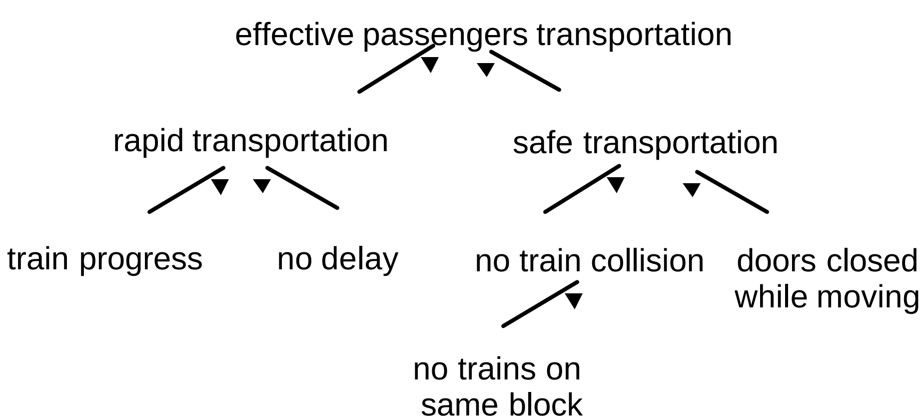
- Goals provide rich structuring mechanism:

  AND/OR refinement, abstraction

  (Dardenne'91, Mylopoulos'92)

31

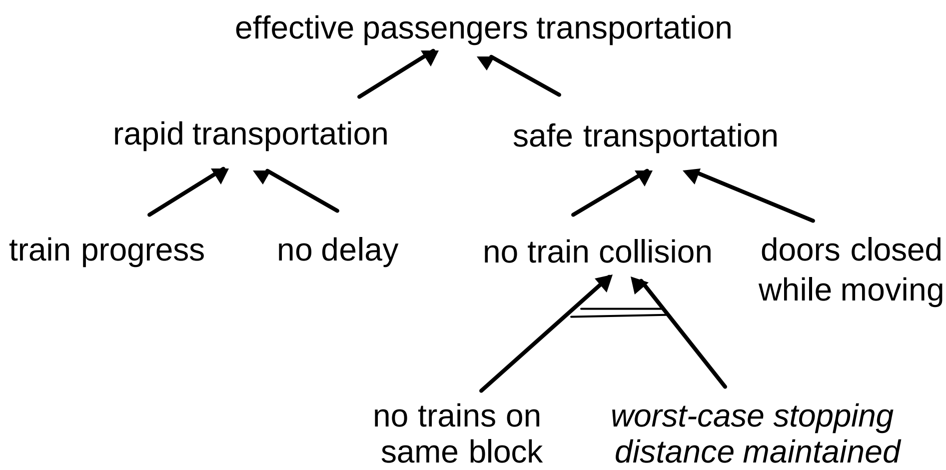
WHY are goals needed? (3)



AND-refinement

32

WHY are goals needed? (4)



OR-refinement

33

WHY are goals needed? (5)

- Goal AND *abstraction* Þ
  requirements rationale
- Goal AND *refinement* Þ
  - user-oriented structuring of documentation
  - traceability
    strategic objectives ® technical requirements

34

WHY are goals needed? (6)

- Goal OR *refinement* Þ

  identification, validation, negotiation of
  alternative requirements

  (Dardenne'91, Mylopoulos'92, Chung'00)

- Goal OR *assignment* Þ

  identification, validation, negotiation of
  alternative system boundaries/proposals

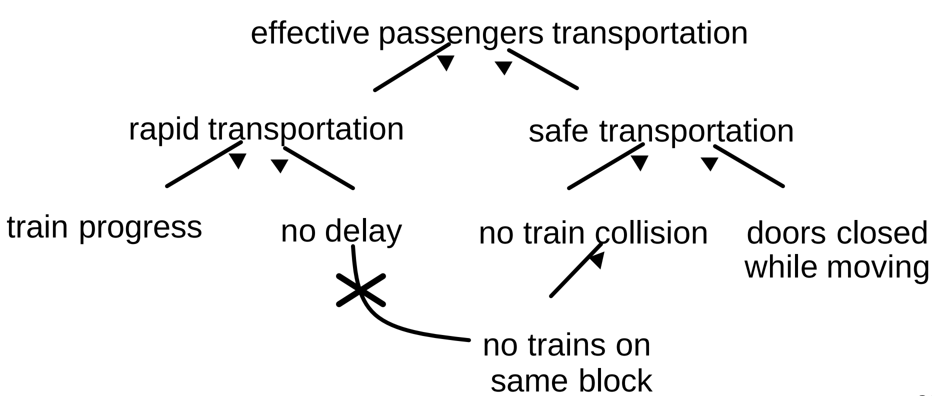
  (Dardenne'93)

35

WHY are goals needed? (7)

- Roots for conflict detection & resolution

  (Robinson'89, Boehm'95, van Lamsweerde'98)



36

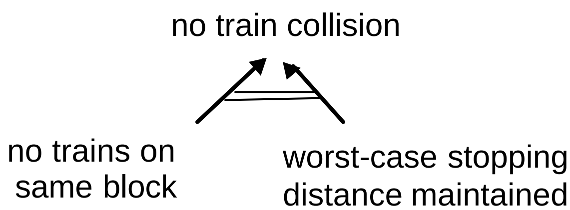
WHY are goals needed? (8)

- Support for evolution management

  higher-level goals Þ more stable concerns

  (Anton'94, ...)

  Þ multiple system versions within single model:
  common parent goals, different OR-branches



37

WHY are goals needed? (9)

In short:
goals provide the right abstractions for RE processes of ...

- domain analysis
- elicitation, elaboration
- negotiation
- specification
- analysis
- documentation
- evolution

38

Outline

- Requirements engineering
- Goal-oriented requirements engineering
- Building rich system models for RE
  - Modeling & specification techniques
    - The goal model
    - The object model
    - The agent model
    - The operation model
  - A goal-oriented RE method in action
- From requirements to software specs
- Conclusion

39

Model-based RE

- Good model = best interface between multiple RE processes & actors
- Good model should support...
  - all processes: domain analysis, requirement elicitation, negotiation, specification, analysis, evolution
  - structured documentation
  - traceability of decisions
  - multiple levels of abstraction & precision
  - coverage of all facets within broad scope of RE

⇒ need to integrate multiple views of current system & system-to-be

40

Building rich system models: the KAOS approach

- Multiple views along the WHY, WHAT, WHO axes
  - intentional: modeling functional & non-functional goals by AND/OR goal diagrams
  - structural: modeling domain objects by UML class diagrams
  - responsibility: modeling system agents by context diagrams
  - functional: modeling S2B services by operationalization diagrams (& UML use cases)
  - behavioral: modeling system dynamics by scenarios & state machines (UML sequence and state diagrams)
- View integration: derivation links, consistency rules

KAOS = Keep All Objectives Satisfied

41

Modeling goals

- Intentional view of the system being modeled
- Goals are modeled by ...
  - types: Maintain/Avoid, Achieve/Cease, SoftGoal
  - taxonomic categories: Satisfaction, Information, Accuracy, Security, Safety, Usability, ...
  - attributes: Name, Definition, Priority, Owner, ...
  - links

    intra-model: refinement, obstruction, conflict

    inter-model: reference, operationalization, responsibility, ...

42

Modeling goals: types

- Types define classes of behavior prescribed or preferred
  - Achieve / Cease goals: *generate* behaviors

    CurrentCondition Þ eventually TargetCondition

    e.g. Achieve [TrainProgress]
  - Maintain / Avoid goals: *restrict* behaviors

    CurrentCondition Þ TargetCondition

    CurrentCondition Þ always TargetCondition unless NewSituation

    e.g. Maintain [DoorsClosedWhileMoving]

    CurrentCondition Þ not TargetCondition

    e.g. Avoid [TrainsOnSameBlock]
  - SoftGoals goals: *prefer* behaviors when alternatives to be selected

43

Modeling goals: types (2)

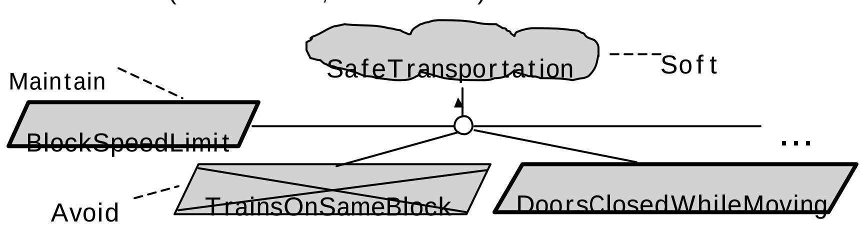
- SoftGoals vs. Achieve/Maintain:
  - SoftGoal achievement cannot be established in clear-cut sense

    ® goal satisficing, qualitative reasoning

    (Mylopoulos'92, Chung'00)
  - Achieve/Maintain goal achievement can be verified

    ® goal satisfaction, formal reasoning

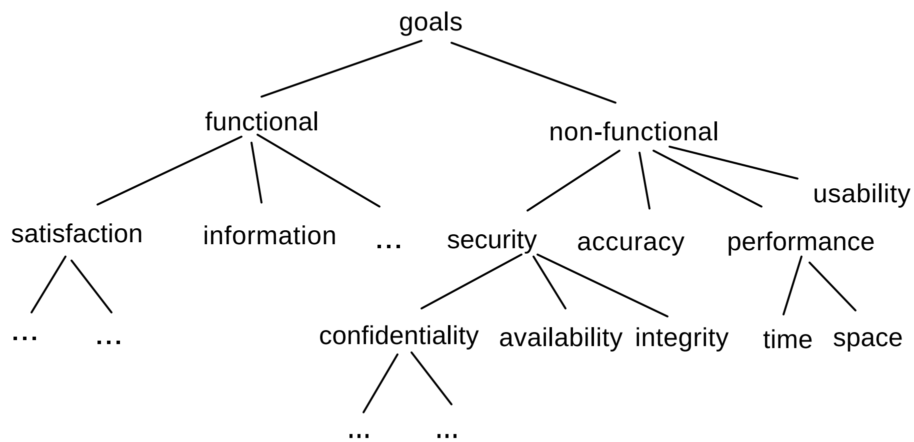
    (Dardenne'93, Darimont'96)



44

Modeling goals: categories

Goals may be classified in
functional/non-functional categories



45

Modeling goals: types & categories

Goal types & categories are used...

- for lightweight specification (Dardenne'93, Dwyer et al'99)
- in heuristic rules for elicitation, validation, reuse, conflict management, ...
  (Dardenne'93, Sutcliffe'93, Anton'98, Chung'00, ...)

"Is there any conflict between ***Information*** goals and ***Confidentiality*** goals?"

"***Confidentiality*** goals are ***Avoid*** goals (on ***Knows*** predicates)"

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

more specific types & categories
Þ *more specific heuristics*

46

Modeling goals: goal attributes

- Attributes capture intrinsic goal features
  - Name — BlockSpeedLimit
  - Definition — A train should stay below the max speed the block can handle
  - [Priority] — highest, high, …, lowest
  - [Owner] — which process-level actor required that goal
- Used for specification & reasoning (e.g. conflict management)

  (Robinson'89, Dardenne'93, van Lamsweerde'98, Chung'00)

47

Modeling goals: goal links

- Basis for goal-based model building & reasoning
- Links relate goals to ...
  - other goals
    - AND/OR refinement ⇒ goal contributions
    - obstruction ⇒ deidealized, more robust model
    - conflict ⇒ resolutions
  - other submodels ⇒ traceability
    - reference ® *objects*
    - responsibility ® *agents*
    - operationalization ® *operations*
    - coverage ® *scenarios*

48

Modeling goals: AND/OR refinement

- Goal G is AND-refined into subgoals $G_1, \ldots, G_n$ iff achieving $G_1, \ldots, G_n$ contributes to achieving G

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

  G_i is said to contribute positively to G

- The set $\{G_1, \ldots, G_n\}$ is a complete AND-refinement of G iff $G_1, \ldots, G_n$ are sufficient for achieving G in view of known domain properties

$$\{G_1, \ldots, G_n, \text{Dom}\} \models G$$

- Goal G is OR-refined into refinements $R_1, \ldots, G_m$ iff achieving the subgoals of R_i is one alternative to achieving G $(1 \leq i \leq m)$

  R_i is called alternative for G

49

Modeling goals: AND/OR refinement (2)

- Getting complete AND-refinements of non-soft goals is essential for requirements completeness
- Domain properties used for arguing about complete AND-refinements are ...
  - descriptive assertions attached to domain objects in the object model
  - classified as ...

    domain invariants - known to hold in every state

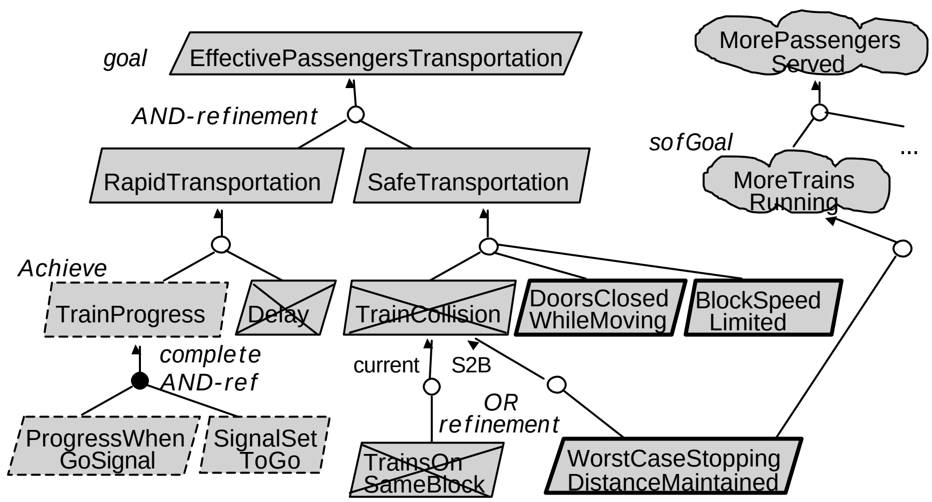
    "train doors are either open or closed"

    domain hypotheses - assumed to hold in specific states

    "railway tracks are in good conditions ..."

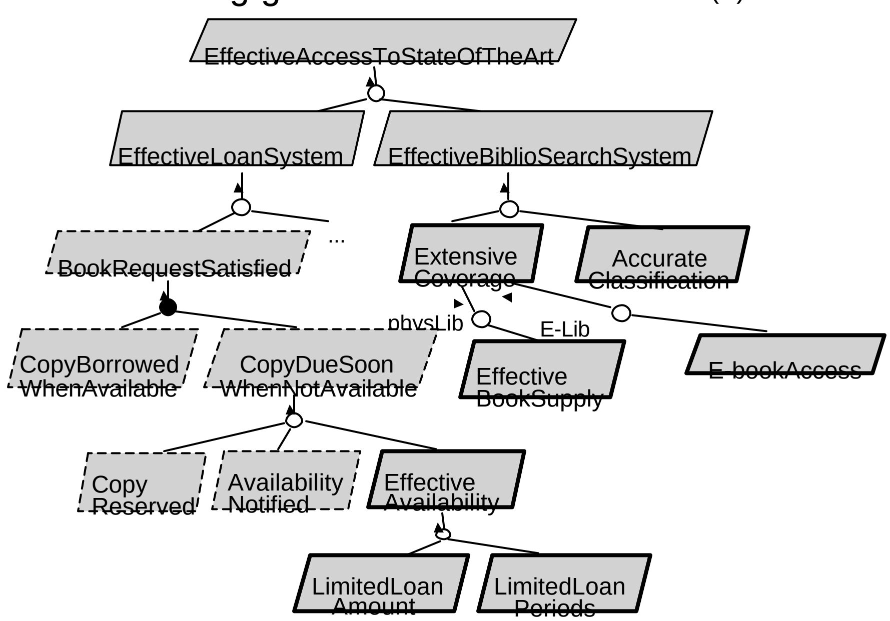
50

Modeling goals: AND/OR refinement (3)



51

Modeling goals: AND/OR refinement (4)



52

Modeling goals: tips & heuristics

- How do you elicit goals ?
  - analysis of current system
    - Þ problems, deficiencies
    - Þ goals of S2B: avoid / reduce them
  - search for intentional keywords in preliminary material (documents available, interview transcripts)
  - later elicitation by refinement & abstraction ...

53

Modeling goals: tips & heuristics (2)

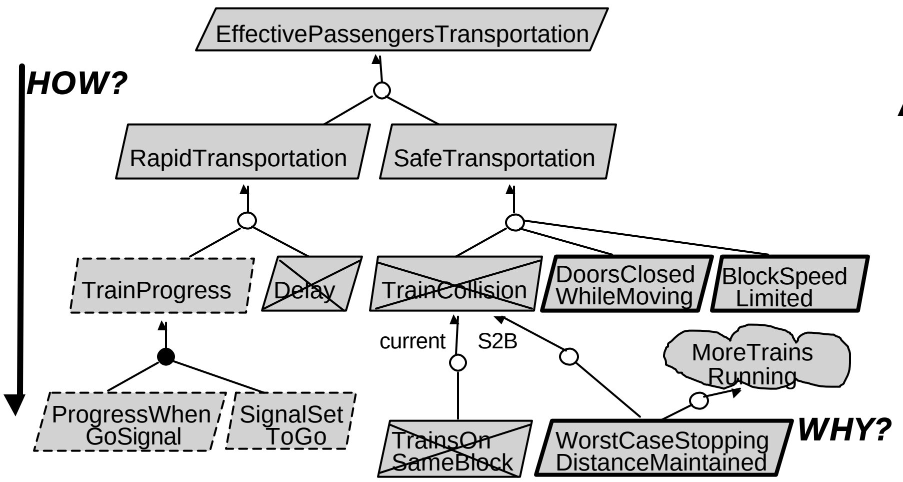
- Later goal elicitation ...
  - by refinement (top-down):
    asking HOW? questions about goals available
  - by abstraction (bottom-up):
    asking WHY? questions about...
    lower-level goals
    scenario episodes (cf. scenario modeling)
    other operational material available

    goal-oriented ^{1} top-down !!
  - by resolution of obstacles, conflicts (cf. below)

(van Lamsweerde'95, '98, '00)

54

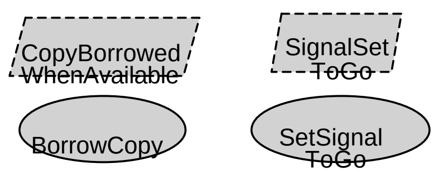
Modeling goals: tips & heuristics (3)



55

Modeling goals: tips & heuristics (4)

- Do not confuse ...
  - goal ...
  - operation ...



Goal ≠ service from functional model (e.g. use case)

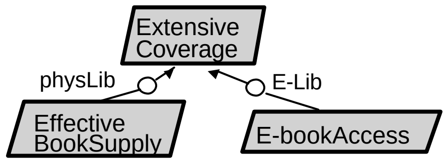
- services operationalize functional, leaf goals in refinement graph
- non-functional goals are often not operationalized in functional model but used to select among alternatives

Tip: past participle for goal name (state to be reached/maintained)
infinitive for operation name (action to reach/maintain that state)

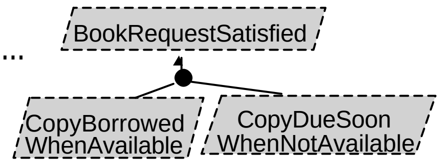
56

Modeling goals: tips & heuristics (5)

- Do not confuse ...
  - OR-refinement ...



  - AND-refinement by case ...



cf. case analysis:

(Case1 **or** Case2) $\Rightarrow$ X **equiv** (Case1 $\Rightarrow$ X) **and** (Case2 $\Rightarrow$ X)

  - OR-refinement introduces alternative systems to reach parent goal
  - AND-refinement by case introduces complementary, conjoined subgoals within same system

57

Modeling goals: tips & heuristics (6)

- To avoid ambiguity in goal interpretation:
  - a precise, complete, consistent goal definition is essential
  - definition must be grounded on domain (cf. "designation", Zave&Jackson'97)
  - must be agreed upon by all stakeholders & actors of requirements process

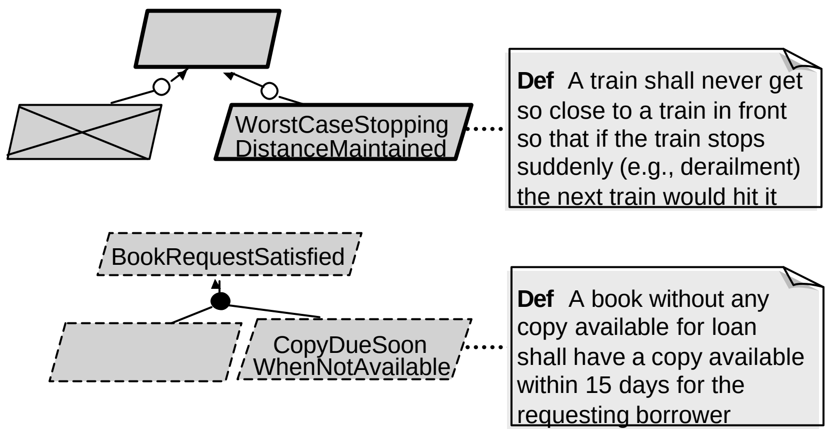
e.g. DoorsClosedWhileMoving ...

= ... during moves only ?
... between stations ?

58

Modeling goals: tips & heuristics (7)

Goal Def attribute =
placeholder for such precise definition



59

Modeling goals: tips & heuristics (8)

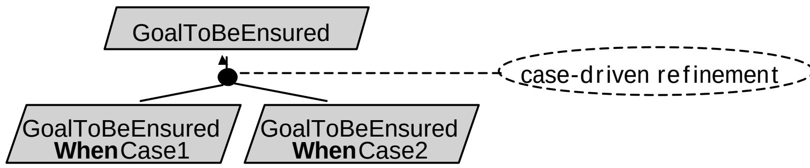
- Reuse complete AND-refinement patterns wherever appropriate
  - encode refinement tactics
  - codify experience
  - guide modeling process by suggesting refinements to be instantiated
  - aid in model documentation & understanding
  - verification of refinement correctness for free: (formal) proof of completeness hidden
    $\Rightarrow$ lightweight analysis provided

(Darimont'96, Letier'02)

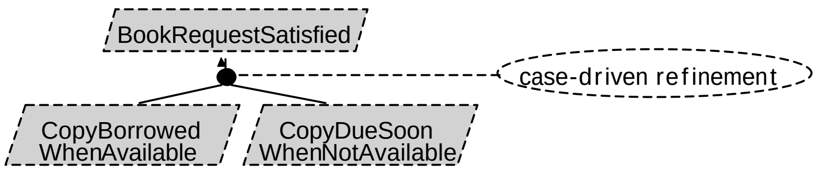
60

Modeling goals: AND-refinement patterns

- Refinement by case
  - applicable when goal achievement space can be *partitioned into cases*



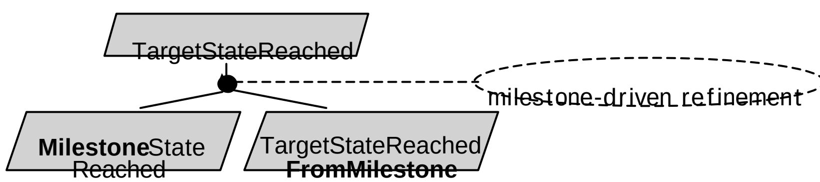
  - example of use



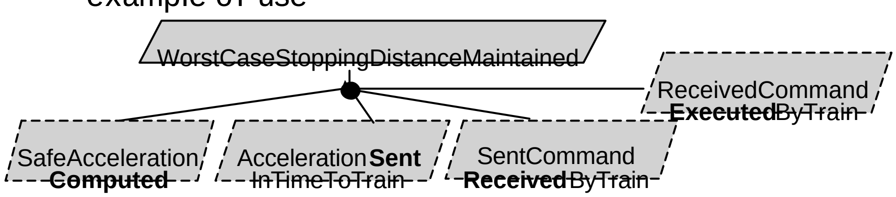
61

Modeling goals: AND-refinement patterns (2)

- Refinement by milestone
  - applicable when *milestone states* can be identified on the way to the goal's target condition



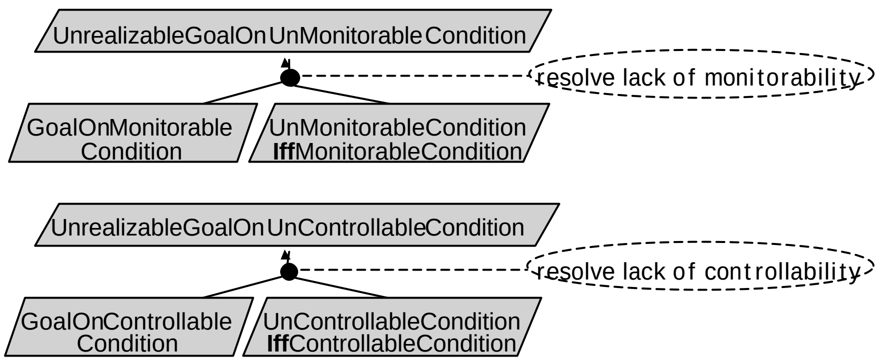
  - example of use



62

Modeling goals: AND-refinement patterns (3)

- Refinement towards goal realizability
  - applicable when goal refers to quantities not monitorable/controllable by candidate agent (Letier'02)

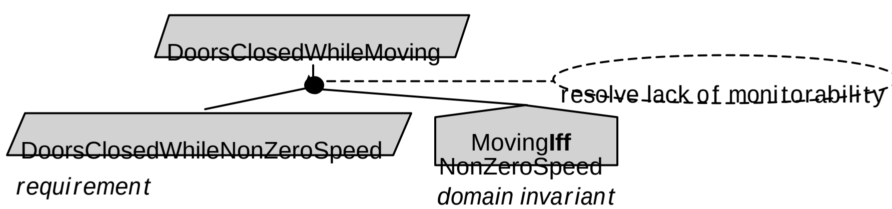


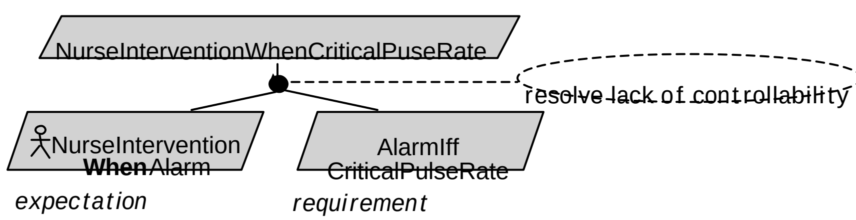
child node may be goal (incl. requirement, expectation)
or domain property (invariant/hypothesis)

63

Modeling goals: AND-refinement patterns (4)

Example of use





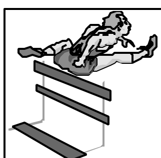
64

Modeling goals... where do we stand?

- Goals are modeled by ...
  - types: Maintain/Avoid, Achieve/Cease, SoftGoal
  - taxonomic categories: Satisfaction, Information, Accuracy, Security, Safety, Usability, ...
  - attributes: Name, Definition, Priority, Owner, ...
  - links

    ® intra-model: refinement, obstruction, conflict
    inter-model: reference, operationalization, responsibility, ...

65



Eliciting goals for more robust system: obstacle analysis

- Problem: goals are often *too ideal*, will be violated because of unexpected agent behavior
- Obstacle = condition on system for goal obstruction

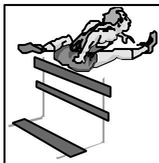
  {O, Dom} |= ¬ G   *obstruction*

  Dom |¹ ¬ O   *domain consistency*

  (high-level exception)
- Anticipate obstacles ...

  Þ new, deidealized goals
  more complete, realistic requirements
  more robust system

66



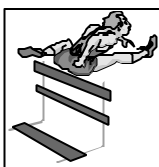
Modeling goals: obstacle analysis

- For every leaf goal in refinement graph (requirement or expectation):
  - identify as many obstacles as possible
  - retain those feasible & plausible ones
  - resolve them according to their criticality

= goal-anchored ...
  hazard analysis for safetyGoals
  threat analysis for securityGoals
  ...

(Potts 1995; van Lamsweerde 1998, 2000)

67



Modeling goals: obstacle analysis (2)

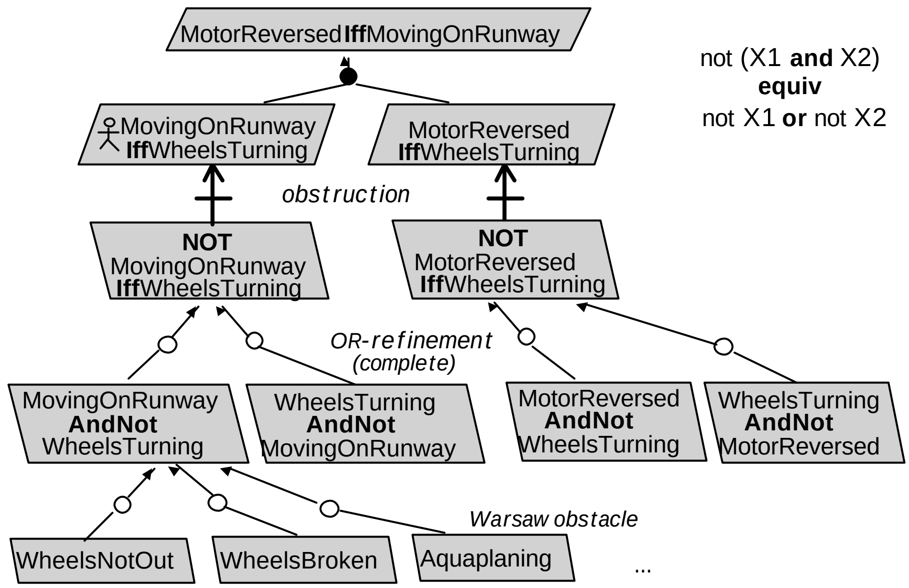
- To identify obstacles to goal G:
  - negate G;
  - find as many AND/OR refinements of ¬G as possible in view of domain properties (known or to be elicited) ...
  - ... until reaching obstruction preconditions that are feasible, plausible and observable

= goal-anchored fault-tree construction

- When "formal button" pressed, formal techniques are available for systematic generation of domain-complete obstacle set (cf. below)

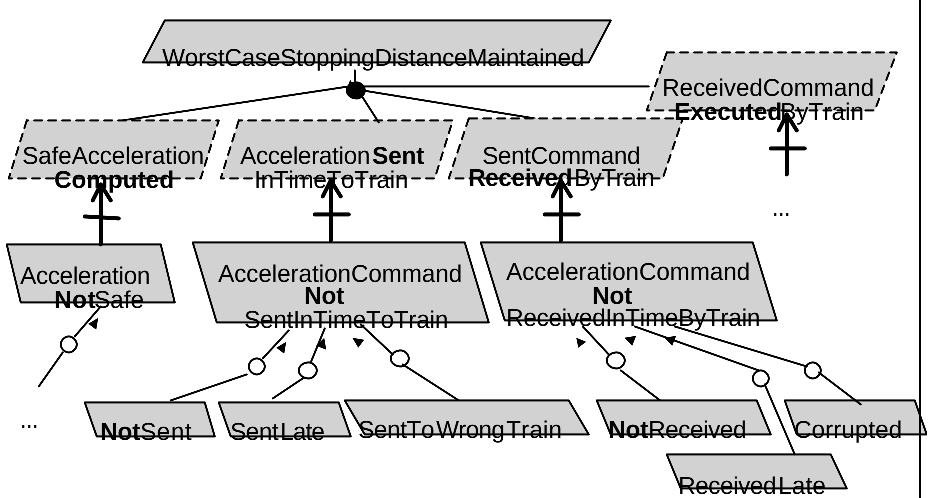
68

Modeling goals: obstacle analysis (3)



69

Modeling goals: obstacle analysis (4)



70

Modeling goals: obstacle analysis (5)

- To resolve obstacles identified:
  - at RE time: explore alternative resolution tactics and select one based on criticality & plausibility of obstacle

    goal substitution,

    agent substitution,

    goal weakening,

    goal restoration,

    obstacle prevention, mitigation ...
  - at run-time:

    obstacle monitoring (Feather 1995, Feather et al 1998)

71

Modeling goals: obstacle analysis (6)

- Alternative resolution operators (tactics):
  - goal substitution: consider alternative refinement of parent goal to avoid obstruction of child goal

    MovingOnRunway **Iff** WheelsTurning

    $\rightarrow$ MovingOnRunway **Iff** *PlaneWeightSensed*
  - agent substitution: consider alternative responsibilities

    WheelSensor $\rightarrow$ *WeightSensor*

    OnBoardTrainController $\rightarrow$ *VitalStationComputer*
  - goal weakening:

    MovingOnRunway **Iff** WheelsTurning

    $\rightarrow$ MovingOnRunway **Iff** *(WheelsTurning **or** ...)*

72

Modeling goals: obstacle analysis (7)

- Alternative resolution operators (cont'd):
  - goal restoration: enforce target condition after goal violation

    BookCopyNotReturnedInTime → *ReminderSent*
    WheelsNotOut → *WheelsAlarmGenerated*
  - obstacle prevention: new *Avoid* goal

    AccelerationCommandCorrupted
    → *Avoid [*AccelerationCommandCorrupted*]*
  - obstacle mitigation: tolerate obstacle but mitigate its effects

    OutdatedSpeed/PositionEstimates
    → Avoid [TrainCollision*WhenOutDatedTrainInfo*]

(van Lamsweerde 2000)

73

Modeling goals... where do we stand?

- Goals are modeled by ...
  - types: Maintain/Avoid, Achieve/Cease, SoftGoal
  - taxonomic categories: Satisfaction, Information, Accuracy, Security, Safety, Usability, ...
  - attributes: Name, Definition, Priority, Owner, ...
  - links

    ® intra-model: refinement, obstruction, conflict
    inter-model: reference, operationalization, responsibility, ...

74

Modeling goals: conflict analysis

- Goals $G_1, \ldots, G_n$ are divergent iff
  there exists a boundary condition B :

  $\{B, \forall i\ G_i, Dom\} \models$ **false** inconsistency

  $\{B, \forall j \neq i\ G_j, Dom\} \not\models$ **false** minimality

- Example

  G_1: DoorsClosedBetweenStations

  G_2: DoorsOpenWhenAlarm

  B: AlarmRaisedBetweenStations

(van Lamsweerde'98)

75

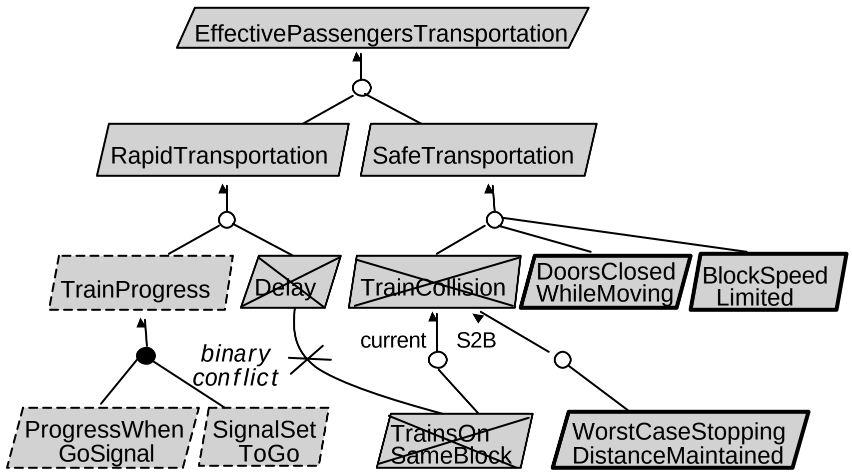
Modeling goals: conflict analysis (2)

- Frequent type of inconsistency (esp. from multiple viewpoints), must be managed !
- Systematic detection:
  - cf. formal technique below
- Resolution:
  - cf. obstacle resolution tactics
    but applied to boundary condition
- Particular case:
  - binary conflict, with B = **true**:

    $\{G_1, Dom\} \models \neg G_2$

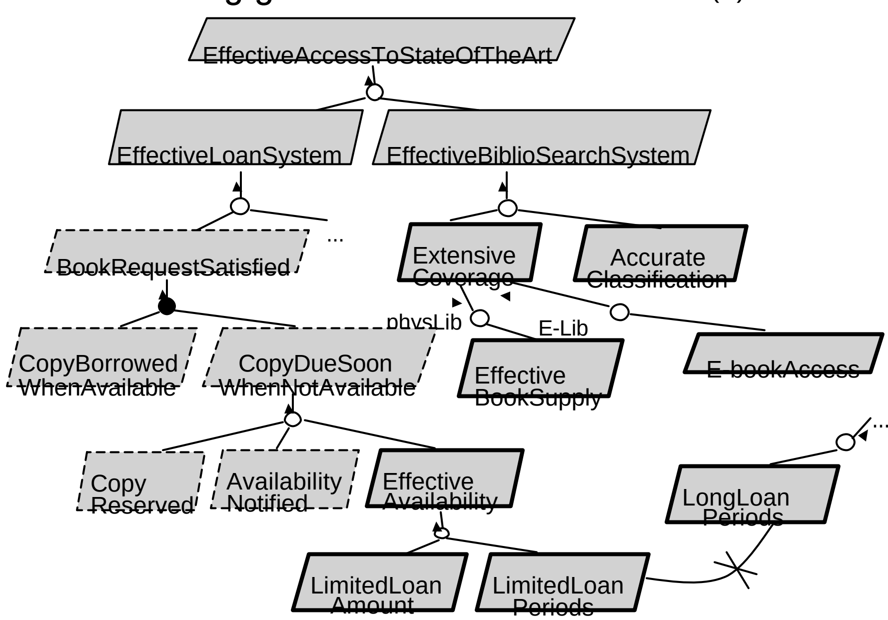
76

Modeling goals: conflict detection



77

Modeling goals: conflict detection (2)



78

Modeling goals... where do we stand?

- Goals are modeled by ...
  - types: Maintain/Avoid, Achieve/Cease, SoftGoal
  - taxonomic categories: Satisfaction, Information, Accuracy, Security, Safety, Usability, ...
  - attributes: Name, Definition, Priority, Owner, ...
  - links

    intra-model: refinement, obstruction, conflict

    ® inter-model:

    reference, operationalization, responsibility

⇓

goal-based model derivation

traceability

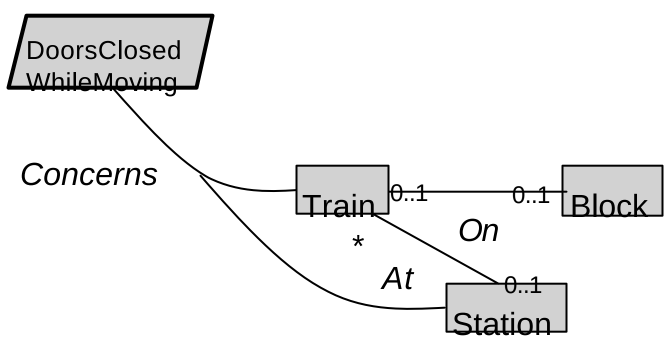
79

Modeling goals: inter-model links

- Object reference:

  G concerns Ob iff G's *Def* refers to Ob

  (generally to its attributes)



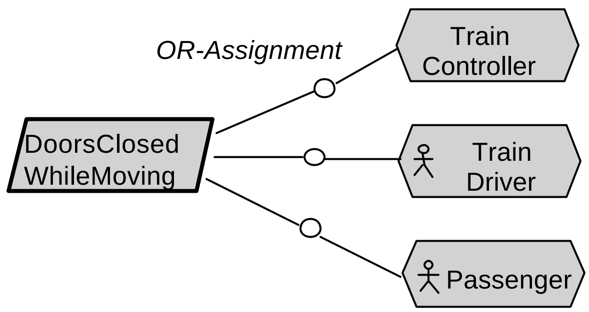
(Dardenne'91, Mylopoulos'92, Rolland'98, ...)

80

Modeling goals: inter-model links (2)

- Goal responsibility:

  G is assignable to Ag iff G can be realized by Ag



(Dardenne'93, Letier'02)

81

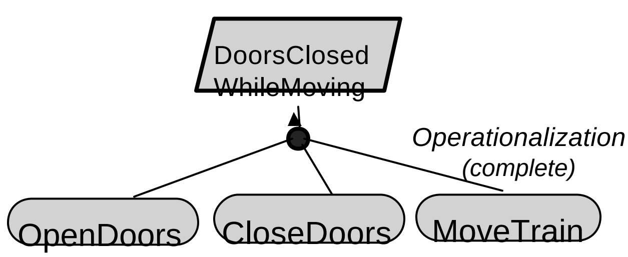
Modeling goals: inter-model links (3)

- Goal operationalization:

  G is correctly operationalized by $Op_1, \ldots, Op_n$ iff the specs of $Op_1, \ldots, Op_n$ are necessary & sufficient for ensuring G

  $\{Spec(Op_1), \ldots, Spec(Op_n)\} \models G$ completeness

  $G \models \{Spec(Op_1), \ldots, Spec(Op_n)\}$ minimality

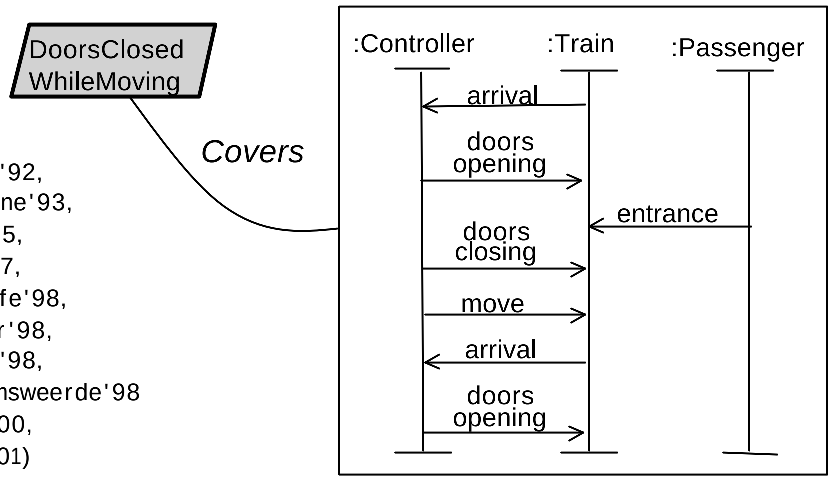


(Mylopoulos'92, Dardenne'93, Letier'02)

82

Modeling goals: inter-model links (4)

- Scenario coverage: G covers Sc iff Sc is a sub-history in the set of behaviors prescribed by G



(Fickas'92,
Dardenne'93,
Potts'95,
Leite'97,
Sutcliffe'98,
Haumer'98,
Rolland'98,
van Lamsweerde'98
Kaindl'00,
Anton'01)

83

Specifying goals in KAOS

- to document elements of goal model
- to enable some form of reasoning
- 2-button specification:
  - semi-formal: graphical: goal diagrams
    + textual: template for type, attributes, links
  - formal: real-time temporal logic
    optional button
    to enable more accurate analysis
    can be pressed locally & incrementally

84

Specifying goals in KAOS (2)

- Textual template [ ] = optional

Goal *Maintain* [DoorsClosedWhileMoving]

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

Concerns Train/DoorsState

[Category Safety]

[Priority Highest]

[AndRefines SafeTransportation]

[RefinedTo DoorsClosedWhileNonZeroSpeed
MovingIffNonZeroSpeed]

[OperationalizedBy OpenDoors, CloseDoors, MoveTrain]

[UnderResponsibilityOf TrainController]

[FormalSpec $\forall$ tr: Train
Moving (tr) $\Rightarrow$ tr.DoorsState = "closed"]

85

Outline

- Requirements engineering
- Goal-oriented requirements engineering
- Building rich system models for RE
  - Modeling & specification techniques
    - The goal model
    - The object model
    - The agent model
    - The operation model
  - A goal-oriented RE method in action
- From requirements to software specs
- Conclusion

86

Modeling objects in KAOS/UML

- Structural view of the system being modeled
- Object = thing of interest in the system whose instances ...
  - share similar features
  - can be distinctly identified
  - have specific behavior from state to state
- Object specializations (at meta level):
  - entity: autonomous object
  - association: object dependent on other objects it links
  - event: instantaneous object
  - agent: active object, controls behaviors

87

Modeling objects in KAOS/UML (2)

- An object is modeled & specified by its features ...
  - Domain-independent attributes ...

    Name, Def: must precisely define conditions for an individual to be among the object's instances in some state (object semantics)

    A borrower is any person who has registered to the corresponding library for the corresponding period of time

    Alive: true in some state iff corresponding instance has appeared in system but not disappeared yet
  - Domain-specific attributes & associations
  - Domain invariants
  - Domain initializations in state where arbitrary instance appears

88

Modeling objects in KAOS/UML (3)

- Association Ass
  - instance = tuple of object instances linked, each playing corresponding role
  - $Ass[O_1, ..., O_n].Alive$ noted by predicate $Ass\ (O_1, ..., O_n)$
  - multiplicities:
    for source instance, min/max number of target instances
    may encode some ...
    domain properties
    "A train may be at one station at most at a time"
    requirements
    "A block may not accommodate more than one train at any time"

89

Modeling objects in KAOS/UML (4)

- Association Ass (cont'd)
  - multiplicities may encode some domain properties & requirements

  BUT ...
  - mix prescriptive & descriptive assertions
  - most assertions are not expressible by multiplicities

    ⇒ need for ...
    domain invariants
    "Non-zero train acceleration implies non-zero speed"
    goals/requirements
    "Acceleration commands shall be sent every 3 secs to the train"

90

Modeling objects in KAOS/UML (5)

- Attribute Att of object Ob

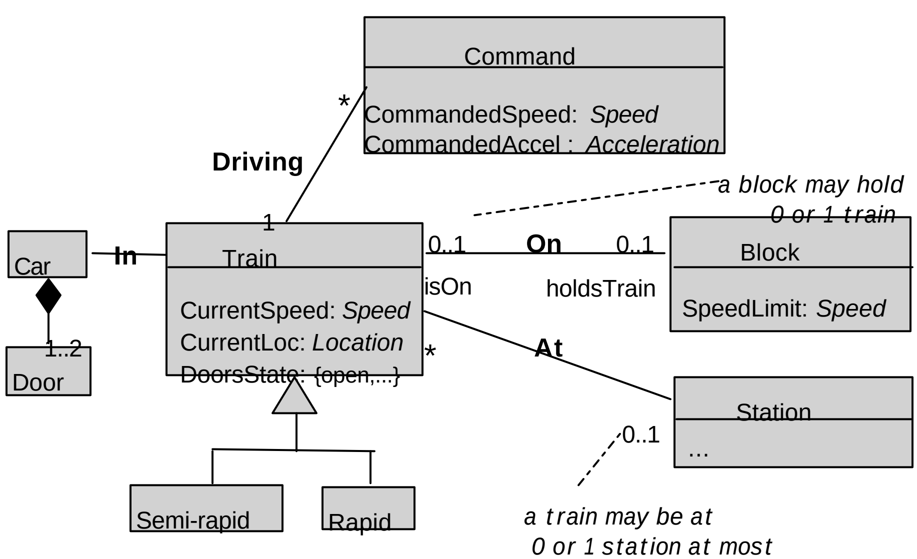
  function Att: Ob → Range (elementary or structured ranges)

  can be attached to entity, association, event, agent (like association)

- Built-in associations
  - specialization (**IsA** sub-classing with multiple inheritance)
  - aggregation/composition (**PartOf** tupling)

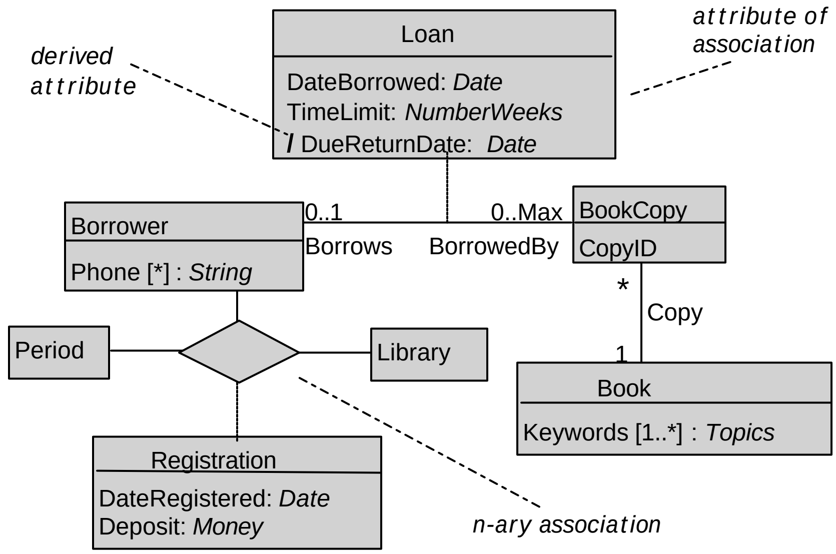
91

Modeling objects in KAOS/UML (6)



92

Modeling objects in KAOS/UML (7)



93

Specifying objects

- Textual template

Entity Train

Def *Any train circulating under control of the system-to-be. The current location of a train is determined by the position of its first car.* ... etc ...

Has DoorsState: {open, closed}
Speed: *SpeedRange*
Accel: *AccelerationRange*
CurrentLoc: *PositionRange*
Capacity [0..1]: # *Natural* % optional & rigid attribute %

DomInvar *Non-zero train acceleration implies non-zero speed. ...*
[FormalSpec ...]

DomInit *A train appearing in the S2B has doors closed, zero speed and position XX.*
[FormalSpec ...]

94

Modeling objects: tips & heuristics

- Elements of object model should be derived incrementally as they are referenced in the goal model
  - Þ completeness & pertinence of object model
  - Þ systematic method, no "hocus pocus" as confessed by UML gurus
- For X a structural element appearing in goal formulations, should X be...
  - an entity?
  - an association?
  - an attribute?
  - an event?
  - an agent?

95

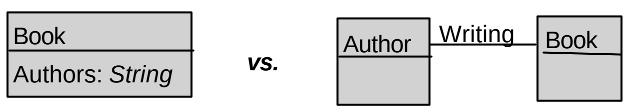
Modeling objects: tips & heuristics (2)

- For X a structural element appearing in goal formulations...
  - X is defined in one single state
    - Þ event class
  - X is active (controls pertinent behaviors)
    - Þ agent class
  - X is passive, *autonomous* (with distinguishable instances)
    - Þ entity class
  - X is passive, *contingent* upon other concepts (with distinguishable instances)
    - Þ association class

96

Modeling objects: tips & heuristics (3)

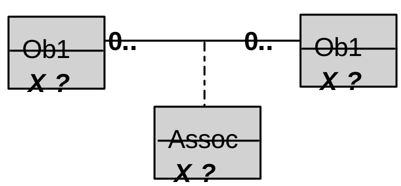
- For X a structural element appearing in goal formulations...
  - Make X an attribute when...
    - instances of X are non-distinguishable
    - X is a function (yielding one single value when applied to some conceptual instance)
    - the range is not a concept you want to attach attributes/associations to
    - you do not want to attach attributes/associations to X



97

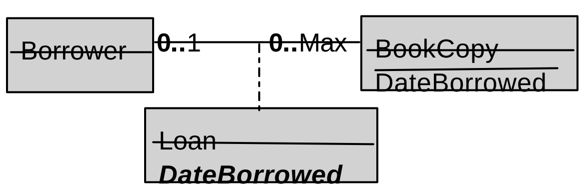
Modeling objects: tips & heuristics (4)

- Should I attach an attribute X to an object in association or to the association?



to association if object instance can be unassociated

to avoid losing info e.g. who is borrower at date d



98

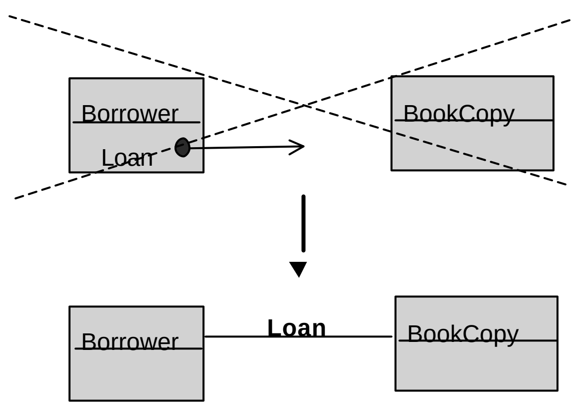
Modeling objects: tips & heuristics (5)

- For conceptual link X between "component" & "composite" objects, should X be... an aggregation? an association?
  - X has a domain-specific semantics *Def*
    - Þ association
  - X has a domain-independent semantics
    - Þ aggregation
  - Component & composite objects seem independent
    - Þ association
  - Component object seems subordinate to composite
    - Þ aggregation/composition

99

Modeling objects: tips & heuristics (6)

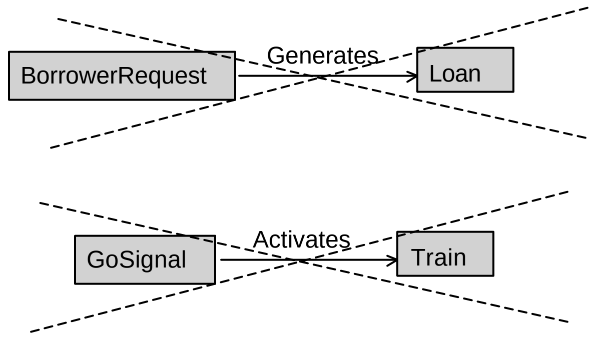
- Avoid frequent flaws in conceptual modeling ...
  - object attribute as "pointer" to another object



100

Modeling objects: tips & heuristics (7)

- Avoid frequent flaws in conceptual modeling ...
  - dynamic information that should be modeled in behavior model (scenarios, state machines), not in structural model



101

Outline

- Requirements engineering
- Goal-oriented requirements engineering
- Building rich system models for RE
  - Modeling & specification techniques
    - The goal model
    - The object model
    - The agent model
    - The operation model
  - A goal-oriented RE method in action
- From requirements to software specs
- Conclusion

102

Modeling agents in KAOS

- Responsibility view of the system being modeled: who's doing what & why
- Agent (recall) :
  - software (existing, S2B), device, human
  - active object responsible for goal achievement (role)
    S2B agent → goal = requirement
    environment agent → goal = requirement
  - alternative agent assignments
    Þ alternative system boundaries/proposals
  - agent controls behaviors by performing operations that operationalize the goals assigned to it

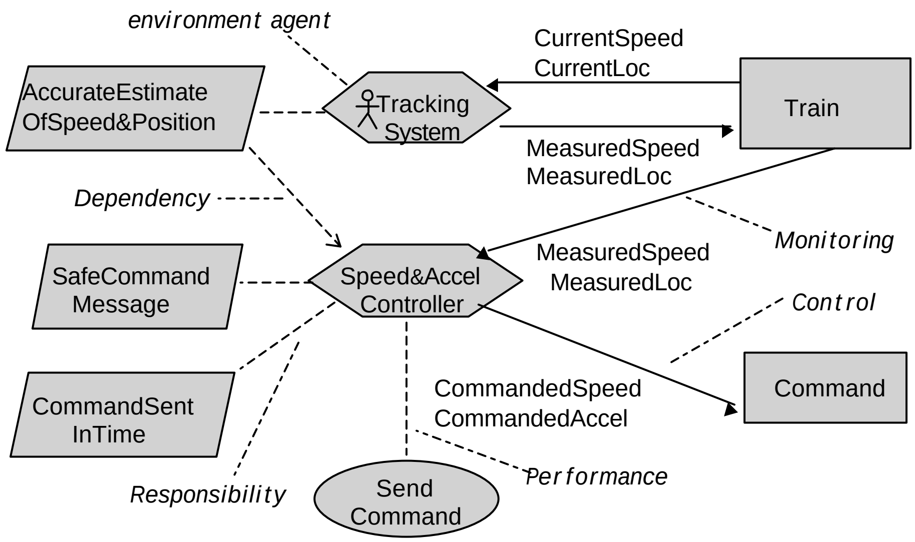
103

Modeling agents in KAOS (2)

- An agent is modeled & specified by its features ...
  - Name, Def
  - (features as an *object*: dom-specific attributes & associations, DomInvar/Init → *in object model*)
  - type: software or environment agent
  - links to goal model: OR-assignment, Responsibility
  - links to object model: Monitoring, Control (cf. 4-var model)
  - links to operation model: Performs
  - Dependency links to other agents for goal achievement or successful operation performance
    (cf. i* [Yu'93], Parnas USE relation)

104

Modeling agents in KAOS (3)



105

Specifying agents

- Textual template

Agent Speed&AccelController

Def *A S2B component that controls the speed and acceleration of all trains in the system...*

Has ...

DomInvar ...

Monitors Train/MeasuredSpeed, Train/MeasuredLoc

Controls Command/CommandedSpeed, Command/CommandedAccel

Responsiblefor SafeCommandMessage, CommandSentInTime

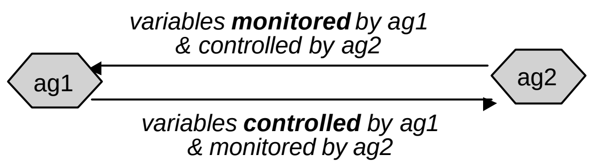
DependsOn TrackingSystem For AccurateEstimateOfSpeed&Position

Performs SendCommand

106

A useful, derivable diagram: context diagram

- A context diagram links agents through their interfaces
  - interface = monitored/controlled variables (attributes)
  - generation of link (ag_1, ag_2) from agent model:
    variable controlled by ag_1 & monitored by ag_2
    variable monitored by ag_1 & controlled by ag_2

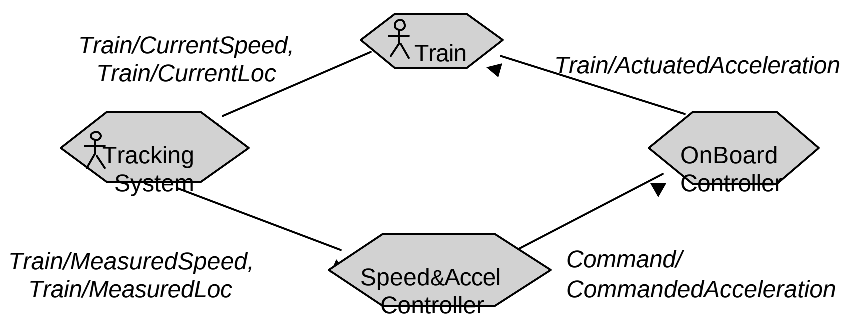


(Jackson 1995, Letier 2001)

107

A useful, derivable diagram: context diagram (2)

- Example:



108

Modeling agents: tips & heuristics

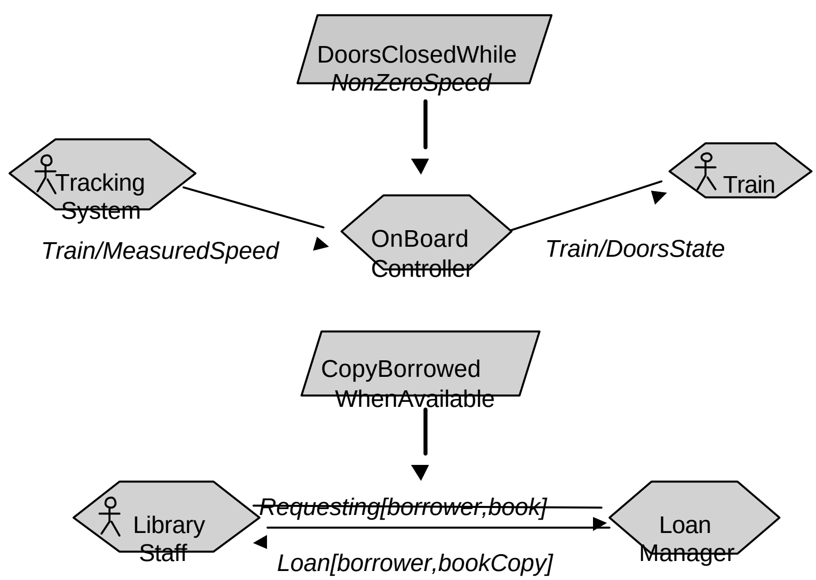
- Model *structural* elements of an agent (domain-specific attributes & associations, specializations/aggregations) ...
  - in the object model,
  - not in the agent model
- An agent's Monitoring/Control links in agent/context diagrams should be derived from precise formulation of the goals assigned to it

G: CurrentCondition (monitoredVariables)

Þ eventually/always TargetCondition (controlledVariables)

109

Modeling agents: tips & heuristics (2)



110

Outline

- Requirements engineering
- Goal-oriented requirements engineering
- Building rich system models for RE
  - Modeling & specification techniques
    - The goal model
    - The object model
    - The agent model
    - The operation model
  - A goal-oriented RE method in action
- From requirements to software specs
- Conclusion

111

Modeling operations in KAOS

- Functional view of the system being modeled: what operations are to be provided? (statics)
- Behavioral view: according to what behavior? (dynamics)
- Operation Op:
  - relation $Op \subseteq InputState \times OutputState$
  - Op applications define state transitions
  - Op operationalizes ...
    - goal(s) assigned to S2B agent $\rightarrow$ software operation
    - goal(s) assigned to environment agent $\rightarrow$ manual/device operation

112

Modeling operations: functional view

- An ***operation*** is modeled & specified by ...
  - Name, Def
  - attributes DomPre, DomPost

    DomPre: condition characterizing the class of input states in the domain

    DomPost: condition characterizing the class of output states in the domain
  - links to goal model: Operationalization
  - links to object model: Input, Output (operation's signature)
  - links to agent model: Performance

113

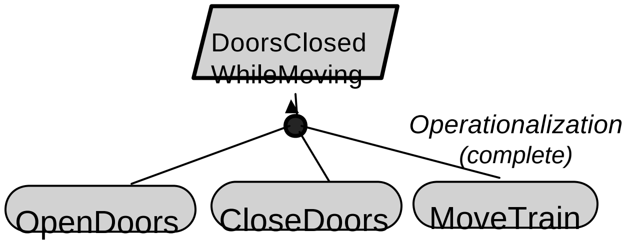
Operations & goals (recall)

- Goal operationalization:

  G is correctly operationalized by $Op_1, \ldots, Op_n$ iff the specs of $Op_1, \ldots, Op_n$ are necessary & sufficient for ensuring G

  $\{Spec(Op_1), \ldots, Spec(Op_n)\} \models G$ completeness

  $G \models \{Spec(Op_1), \ldots, Spec(Op_n)\}$ minimality



(Mylopoulos'92, Dardenne'93, Letier'02)

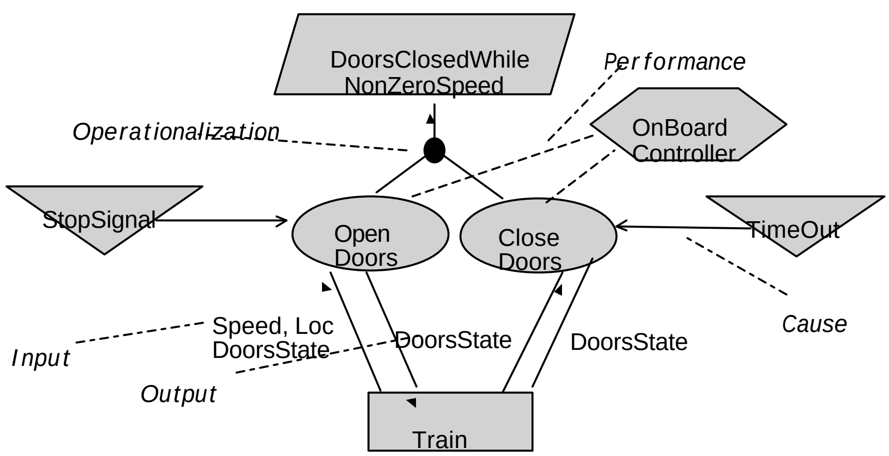
114

Modeling operations: functional view (2)

- An ***operationalization*** of G in Op is modeled & specified by attributes ReqPre, ReqPre, ReqPost
  - ReqPre: *necessary* condition on Op's input states for ensuring G (permission)
  - Reqtrig: *sufficient* condition on Op's input states for ensuring G :
    requires immediate application of Op
    provided DomPre holds (obligation)
  - ReqPost: condition on Op's output states for ensuring G

115

Modeling operations: functional view (4)



116

Specifying operations

- Textual template

Operation OpenDoors

Def *The operation to control the opening of all train doors at once*

Input Train, **Output** Train/DoorsState

DomPre *The train doors are closed*

DomPost *The train doors are open*

ReqPre For DoorsClosedWhileNonzeroSpeed
The train speed is 0

ReqPre For SafeEntry&Exit
The train is at some station

ReqTrig For NoDelayToPassengers
The train has just stopped

[CausedBy StopSignal]

PerformedBy OnBoardController

117

A useful, derivable diagram: use case diagram

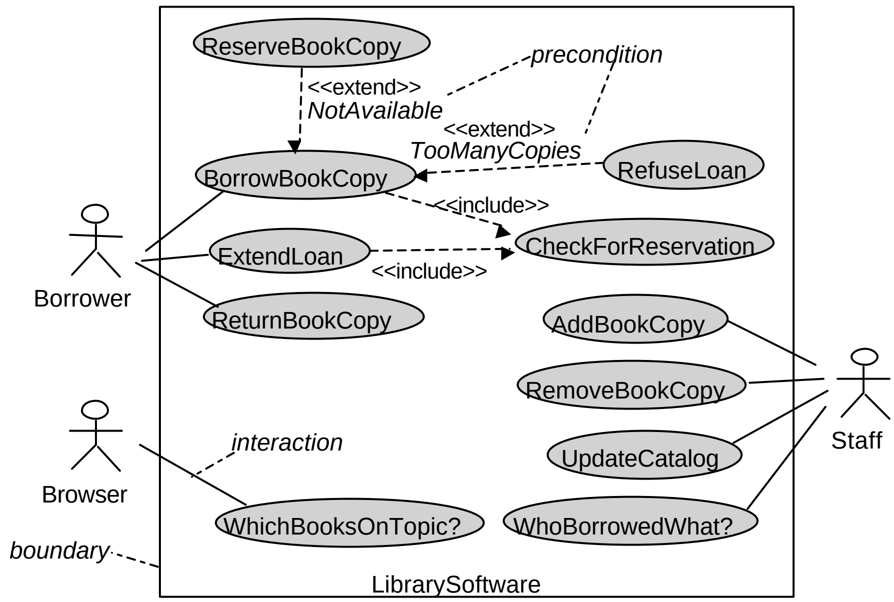
- A use case outlines the operations an agent has to perform
  + interactions with ...
    - the agents controlling operation inputs
    - the agents monitoring operation outputs
  + optional (ill-defined) links:
    to exceptions with preconditions ("extend")
    to sub-operations ("include")

$\Rightarrow$ A use case operationalizes the goals ensured by the operations in it

- Generation of use cases from the operation & agent models is straightforward

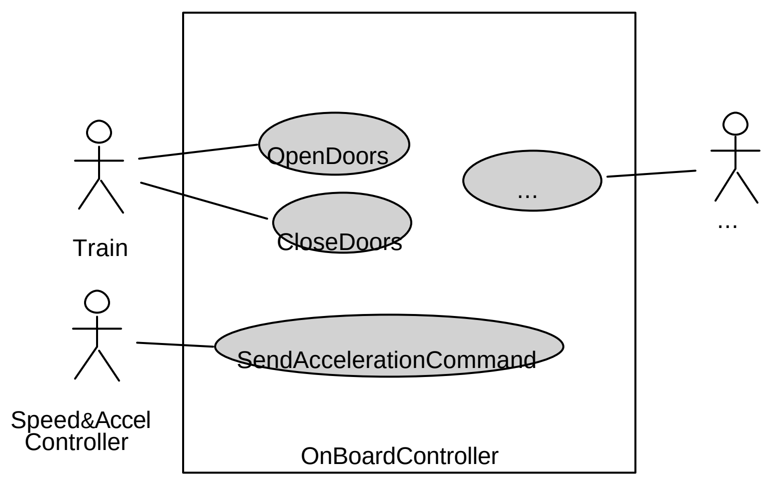
118

A useful, derivable diagram: use case diagram (2)



119

A useful, derivable diagram: use case diagram (3)

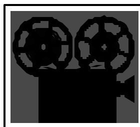


120

Modeling operations: behavioral view

- Behaviors in the current system or the system-to-be are modeled at ...
  - instance level: scenarios of interactions among agent instances
  - class level: state machine capturing set of behaviors of agent, entity or association (including interactions)

121

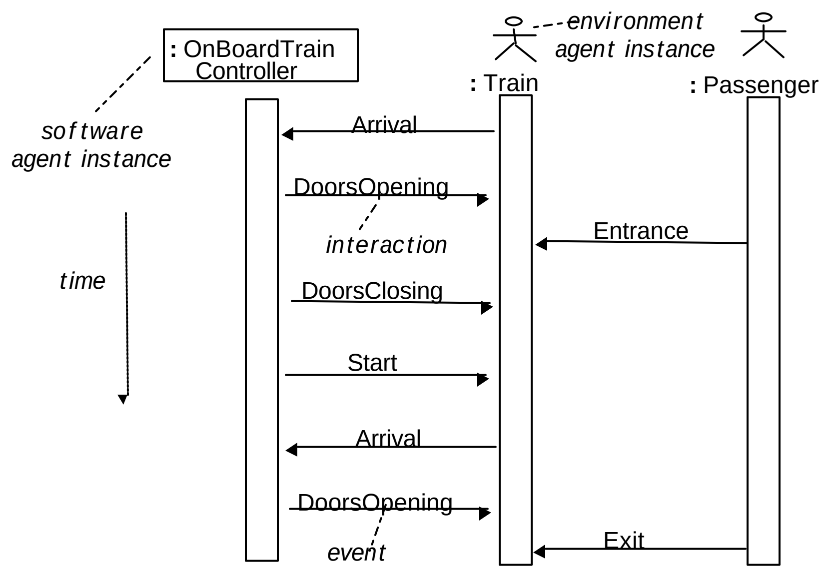


Modeling behaviors: scenarios

- Scenario Sc =
  - historical sequence of interaction *events* among agent instances
  - to illustrate some way of achieving some goal G: Sc is a sub-history in the set of behaviors prescribed by G
  - possibly composed of episodes (sub-scenarios)
  - interaction corresponds to application of operation by source agent, notified to target agent
- Specializations:
  - Positive scenario: desired behavior
  - Negative scenario: undesired behavior

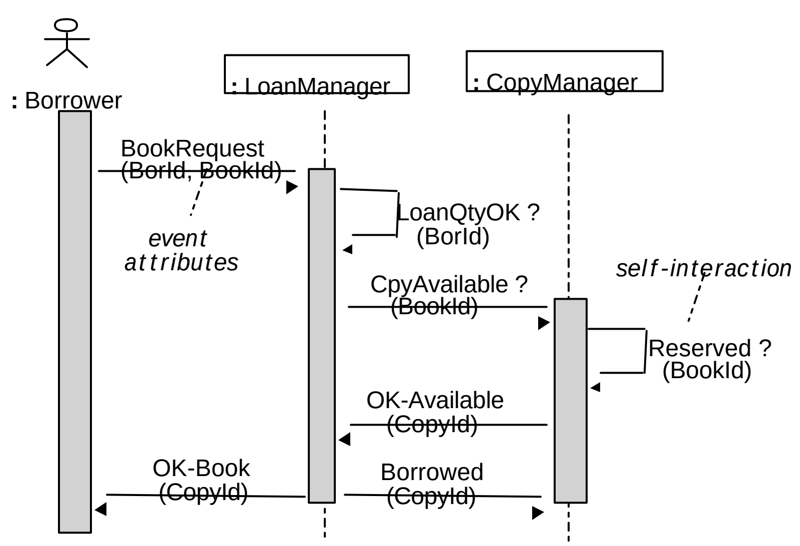
122

Modeling behaviors: scenarios (2)



123

Modeling behaviors: scenarios (3)



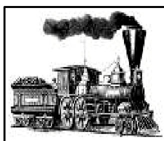
124

Modeling behaviors: scenarios (4)

- Scenarios vs. goals: complementary benefits
  - pros of scenarios:
    concrete
    narrative
    acceptance test data
  - cons of scenarios:
    partial (cf. test coverage problem)
    combinatorial explosion (cf. program traces)
    procedural (unnecessary sequencing)
    premature choice of system boundaries
    requirements kept implicit

Þ use of ... scenarios for elicitation, validation
goals for reasoning

125



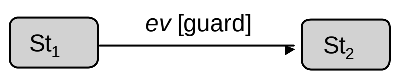
Modeling behaviors: state machines

- State machine SM: State $\times$ Event $\rightarrow$ State
- State of an entity/association/agent instance:
  set of pairs (feature, value)
  $\{att_1 \mapsto val_1, ..., att_n \mapsto val_n, \; assoc_1 \mapsto link_1, ..., assoc_p \mapsto link_p\}$
  e.g.
  {CurrSpeed |→ 0, CurrLoc |→ X, DoorsState |→ 'closed', At |→ (tr, st)}
- State of an entity/association/agent's SM:
  set of states sharing same value for some behavioral attribute/association e.g.
  {CurrSpeed |→ 0, CurrLoc |→ X, DoorsState |→ 'closed' , At |→ (tr, st)}
  {CurrSpeed |→ 5, CurrLoc |→ Y, DoorsState |→ 'closed', At |→ nil}
  belong to state "doorsClosed" of Train SM

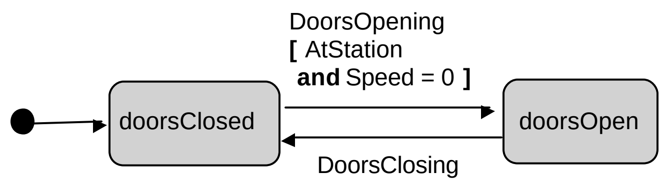
126

Modeling behaviors: state machines (2)

- Guarded state transition:



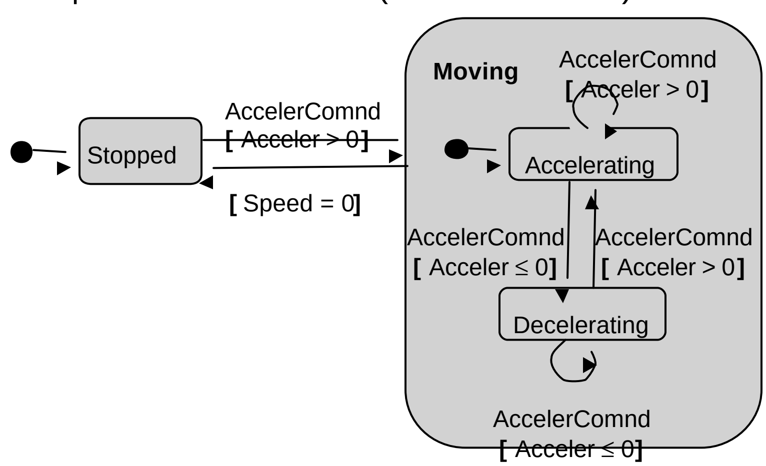
the object gets to state St2 ...
if it is in St1 and *ev* occurs
and **only if** the guard is true



127

Modeling behaviors: state machines (3)

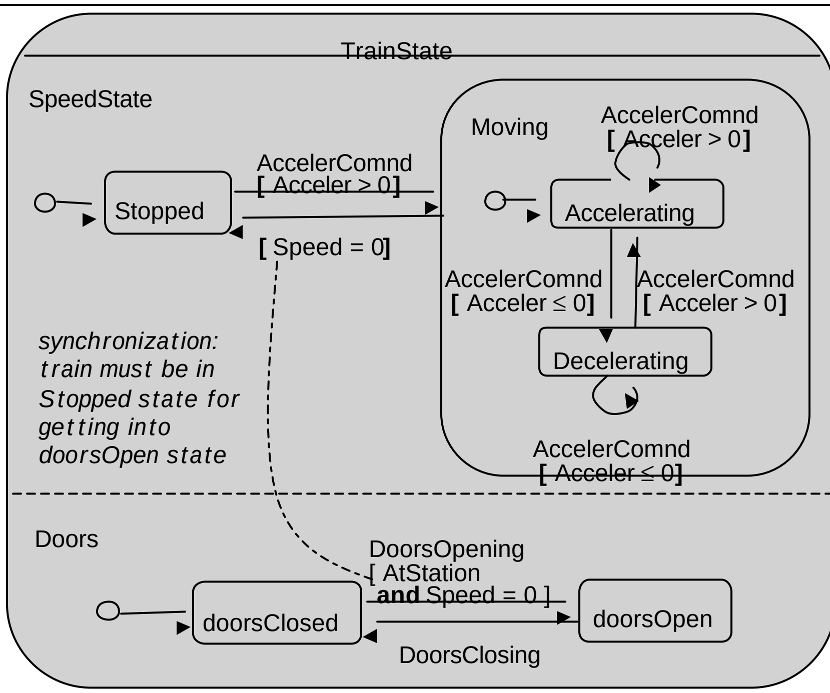
- Nested states ...
  – sequential: super state is diagram composed of sequential sub-states (cf. Statecharts)



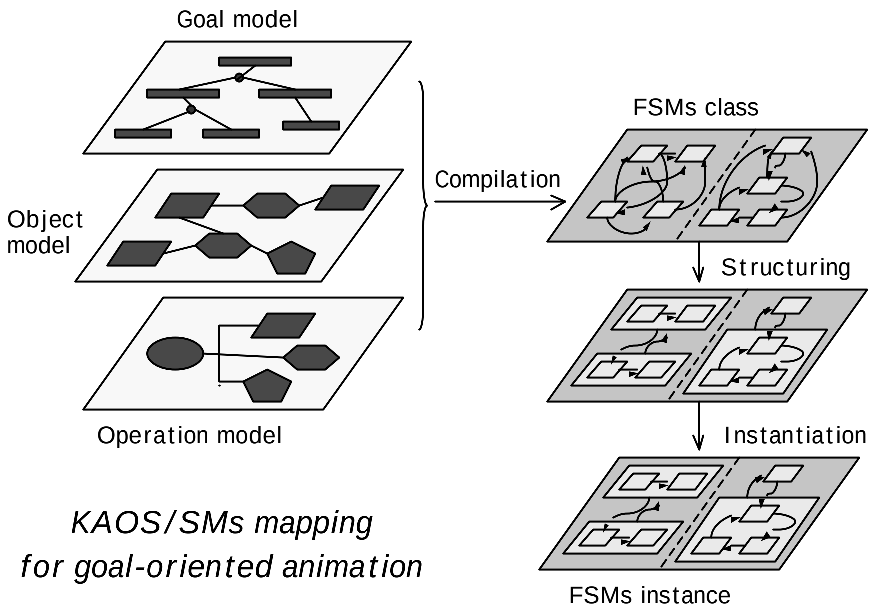
128

Modeling behaviors: state machines (3)

- Nested states ...
  - concurrent: super state is diagram composed of concurrent sub-states (cf. Statecharts)
  - Example: see next slide



State machines are derivable from KAOS models



KAOS/SMs mapping for goal-oriented animation

131

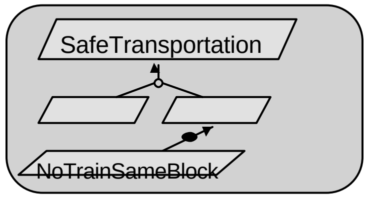
Outline

- Requirements engineering
- Goal-oriented requirements engineering
- Building rich system models for RE
  - Modeling & specification techniques
    - The goal model
    - The object model
    - The agent model
    - The operation model
  - A goal-oriented RE method in action
- From requirements to software specs
- Conclusion

132

The KAOS goal-oriented RE method

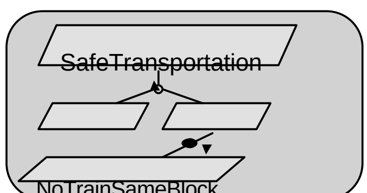
1. Domain analysis: refine/abstract goals



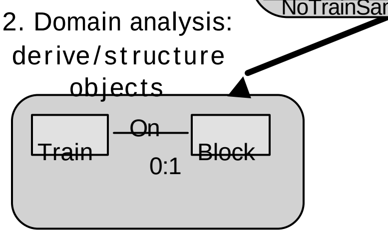
133

The KAOS goal-oriented RE method

1. Domain analysis: refine/abstract goals



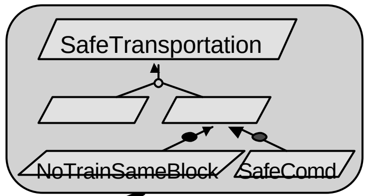
2. Domain analysis: derive/structure objects



134

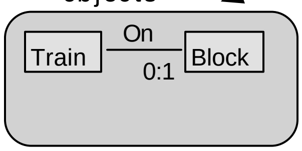
The KAOS goal-oriented RE method

1. Domain analysis: refine/abstract goals



3. S2B analysis: enriched goals (alternatives)

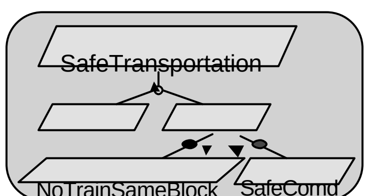
2. Domain analysis: derive/structure objects



135

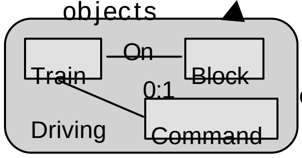
The KAOS goal-oriented RE method

1. Domain analysis: refine/abstract goals



3. S2B analysis: enriched goals (alternatives)

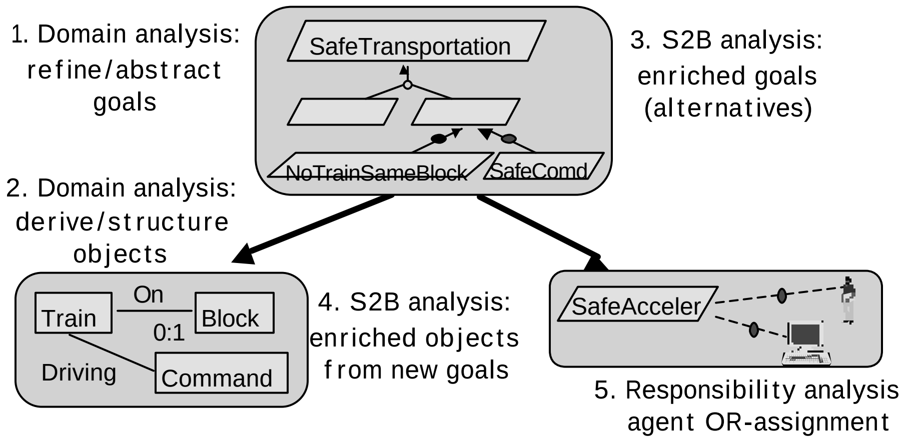
2. Domain analysis: derive/structure objects



4. S2B analysis: enriched objects from new goals

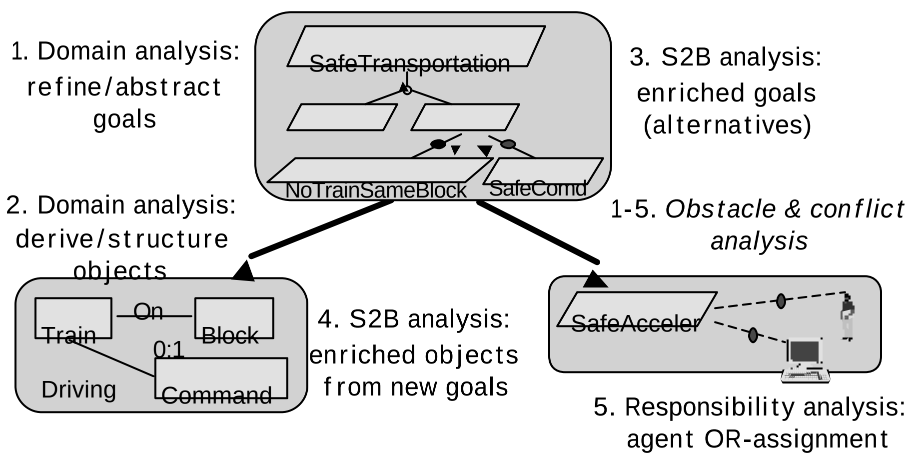
136

The KAOS goal-oriented RE method

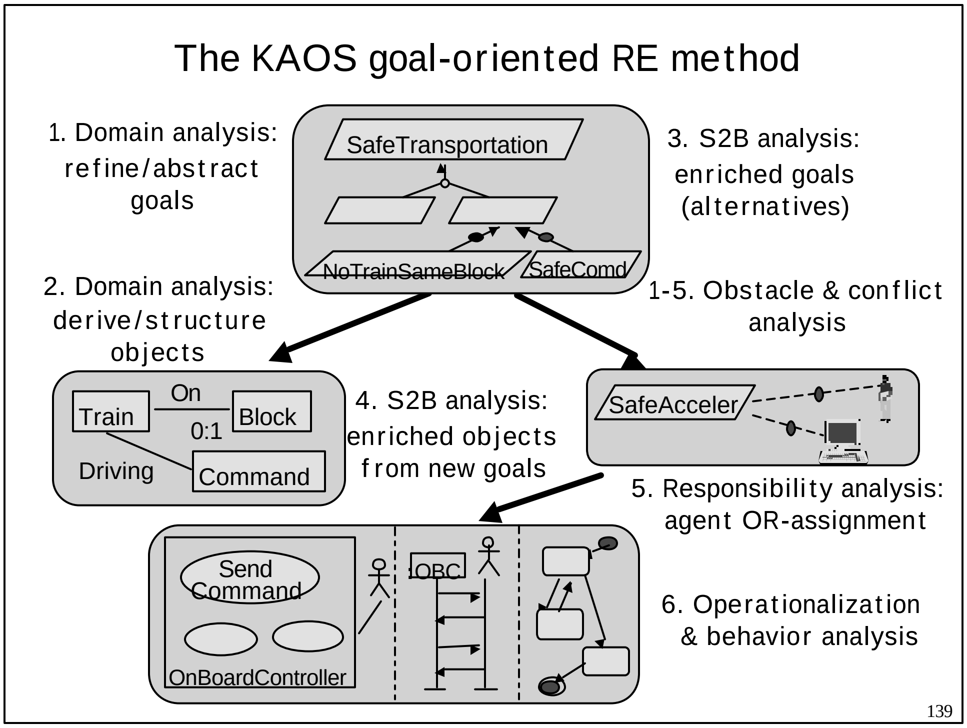


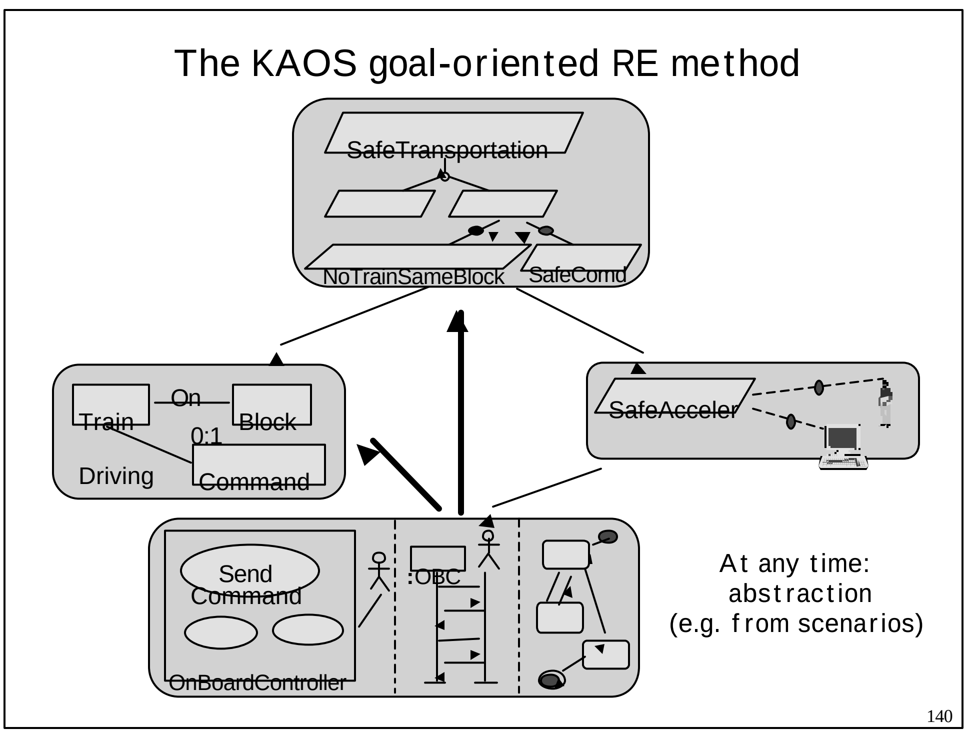
137

The KAOS goal-oriented RE method



138


The KAOS goal-oriented RE method
1. Domain analysis: refine/abstract goals
SafeTransportation
NoTrainSameBlock
SafeComd
3. S2B analysis: enriched goals (alternatives)
2. Domain analysis: derive/structure objects
1-5. Obstacle & conflict analysis
Train
On
Block
0:1
Driving
Command
4. S2B analysis: enriched objects from new goals
SafeAcceler
5. Responsibility analysis: agent OR-assignment
Send Command
OnBoardController
:OBC
6. Operationalization & behavior analysis
139


The KAOS goal-oriented RE method
SafeTransportation
NoTrainSameBlock
SafeComd
Train
On
Block
0:1
Driving
Command
SafeAcceler
Send Command
OnBoardController
:OBC
At any time: abstraction (e.g. from scenarios)
140

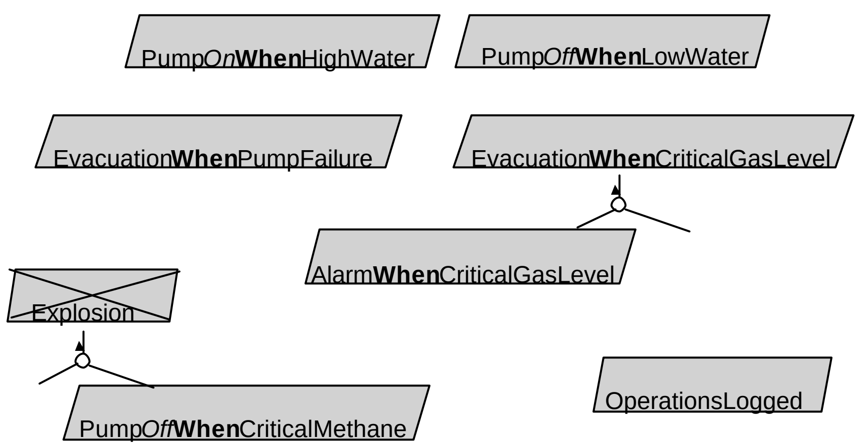
The goal-oriented RE method in action

- "Water percolating into a mine has to be collected in a sump to be pumped out of the mine. Water level sensors shall detect when water is above a *high* and below a *low* level, respectively. A software pump controller shall switch the pump *on* when the water reaches the high water level and *off* when the water reaches the low water level. If, due to a failure of the pump, the water cannot be pumped out, the mine must be evacuated within one hour.
- The mine shall have other sensors to monitor the carbon monoxide, methane and airflow levels. An alarm shall be raised and the operator informed within one second when any of these levels reach a critical treshold so that the mine can be evacuated within one hour. In order to avoid the risk of explosion, the pump shall be operated only when the methane level is below a critical level.
- Human operators can also control the operation of the pump, but within limits. An operator can swith the pump *on* or *off* if the water is between the *low* and *high* water levels. A special operator, the supervisor, can switch the pump *on* or *off* without this restriction. In all cases, the methane level must be below its critical level if the pump is to be operated.
- Readings from all sensors, and a record of the operation of the pump, shall be logged for further analysis." (Joseph, 1996)

141

The goal-oriented RE method in action (2)

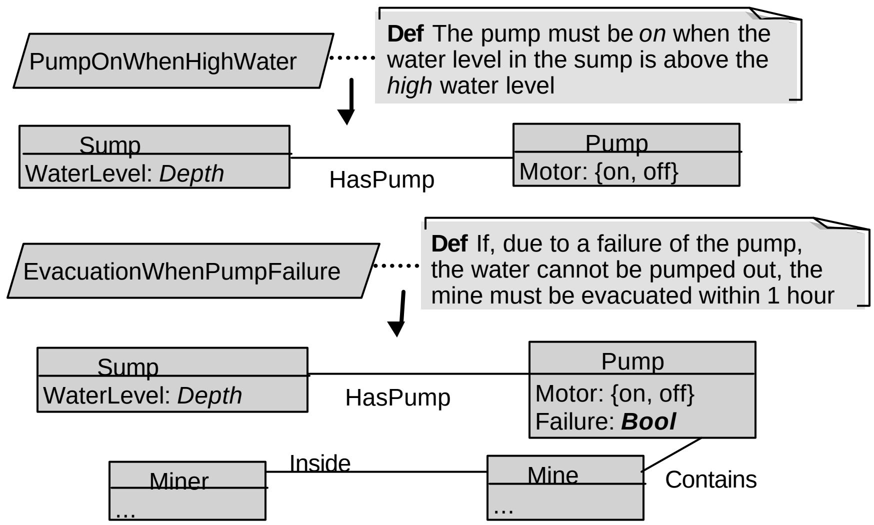
- 1-3: First sketch of goal model (fragments)
  - from intentional keywords: "to", "in order to", shall", etc. in preliminary material, interviews, ... (pre-canned here !)



142

The goal-oriented RE method in action (3)

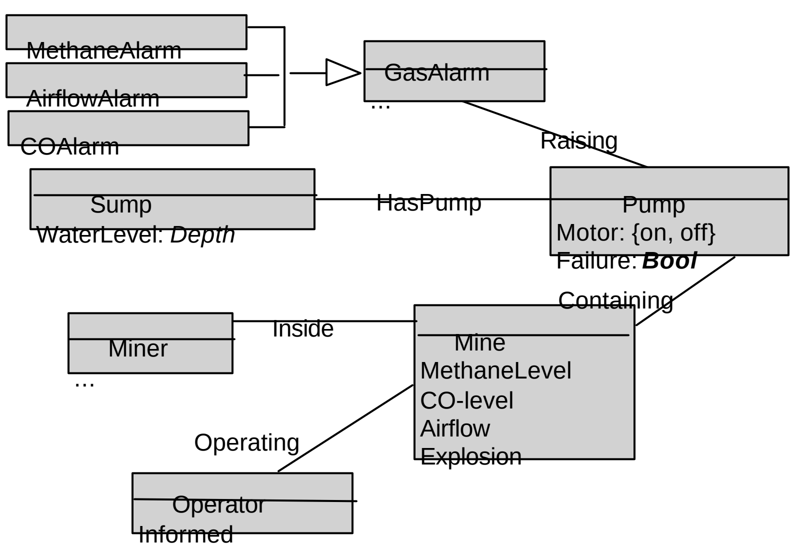
- 2-4: First sketch of object model (fragments)



143

The goal-oriented RE method in action (4)

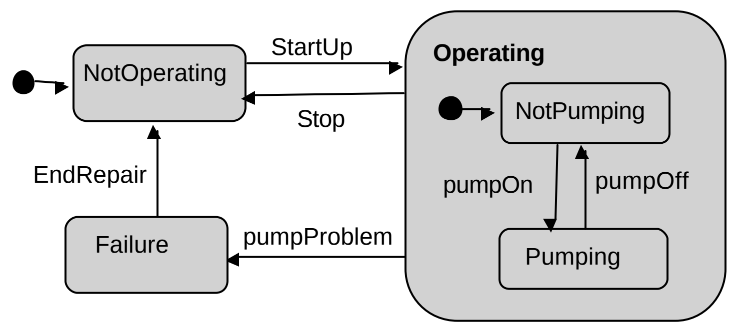
- First object model after Def of every perliminary goal:



144

The goal-oriented RE method in action (5)

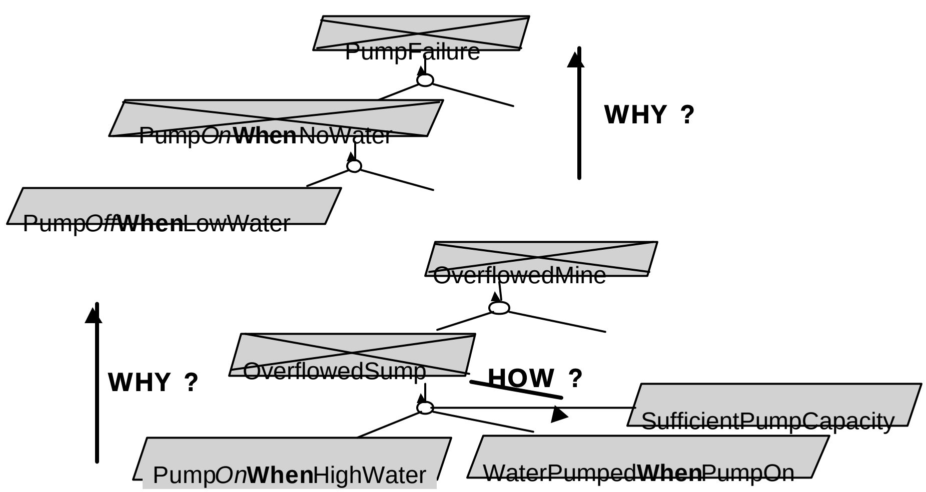
- Modeling behavior of "interesting" domain objects
  - e.g. Pump entity



145

The goal-oriented RE method in action (6)

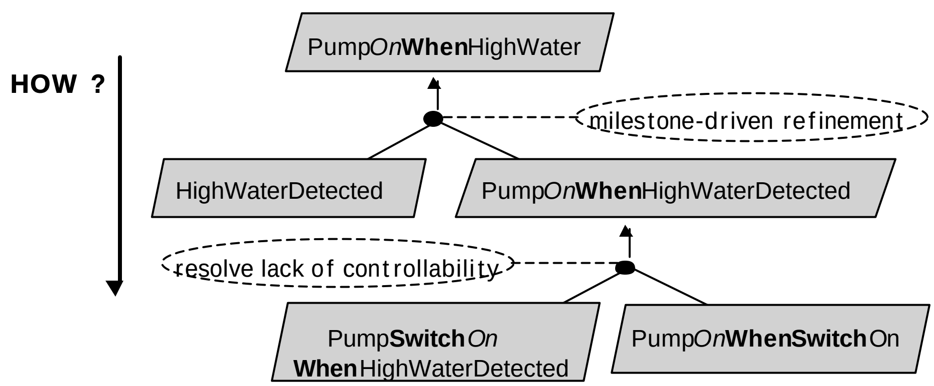
- 1-3: Enrich first goal model
  - asking WHY? & HOW? questions



146

The goal-oriented RE method in action (7)

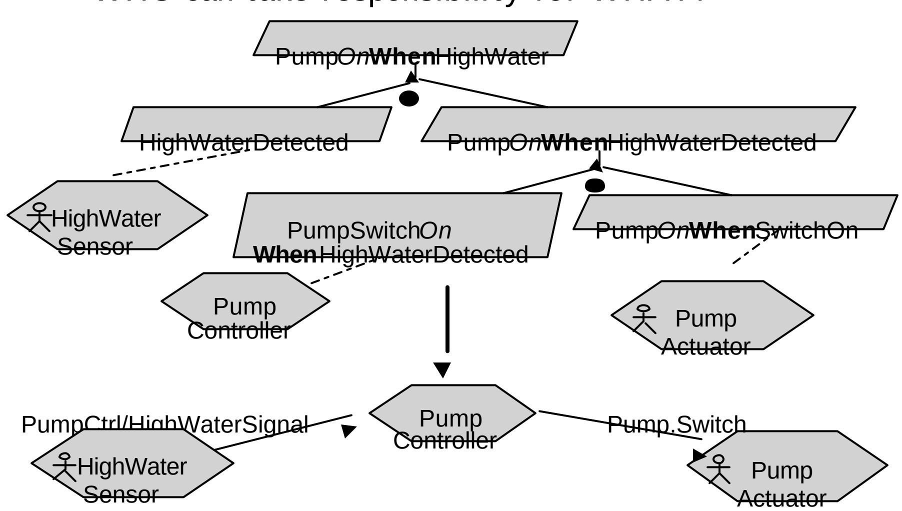
- 1-3: Enrich first goal model
  - asking HOW? questions: using refinement patterns



147

The goal-oriented RE method in action (8)

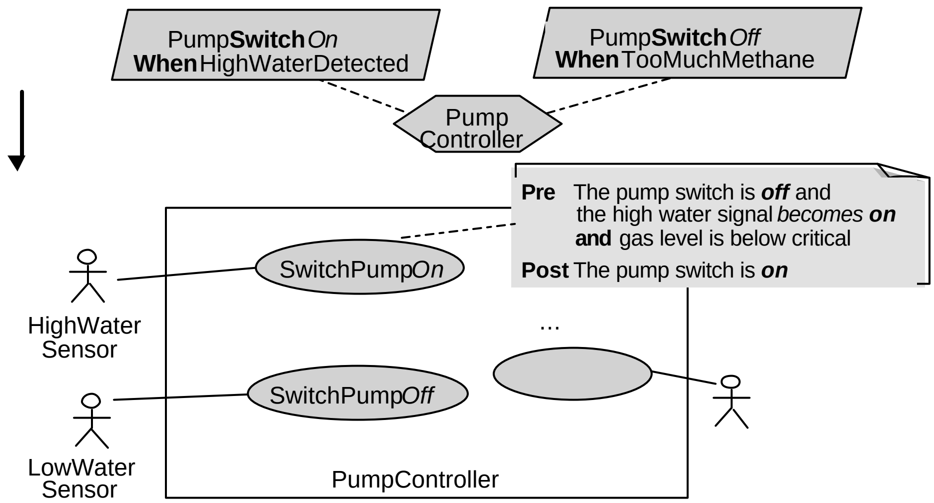
- 5: Elaborate agent model
  - WHO can take responsibility for WHAT?



148

The goal-oriented RE method in action (9)

- 6: Operationalization of functional goals into use cases + operation specs



149

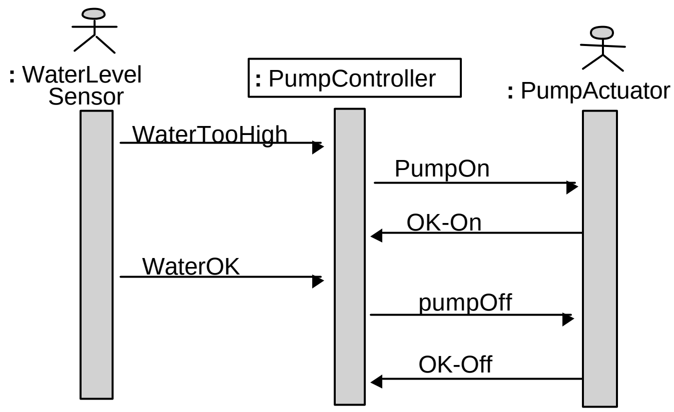
The goal-oriented RE method in action (10)

- 1-5: Obstacle & conflict analysis:
  - cfr. below: use of formal techniques for more accurate analysis
- 6: Scenario modeling
  - for further elicitation (goal mining), validation, and behavior prescription for operations within use case

150

The goal-oriented RE method in action (10)

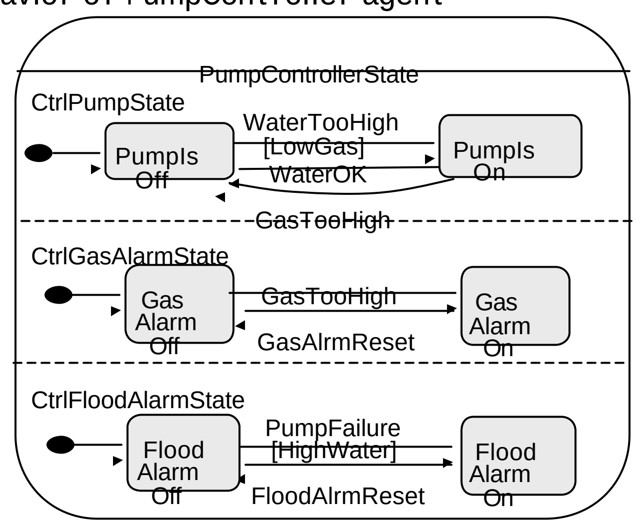
- 6: Scenario modeling (cont'd)



151

The goal-oriented RE method in action (11)

- 6: Inductive synthesis of state machines from scenarios:
  - behavior of PumpController agent



(Whittle 2000)

152

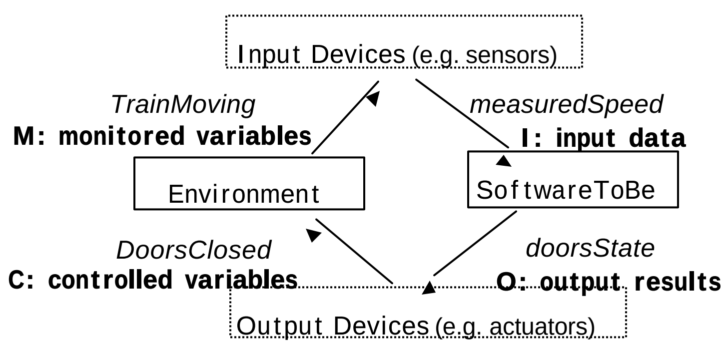
Outline

- Requirements engineering
- Goal-oriented requirements engineering
- Building rich system models for RE
  - Modeling & specification techniques
    - The goal model
    - The object model
    - The agent model
    - The operation model
  - A goal-oriented RE method in action
- From requirements to software specs
- Conclusion

153

From requirements to software specs

- Requirements vs. software specifications: recall



$Req \subseteq M \times C$

$Spec \subseteq I \times O$

Spec = *Translation* (Req) such that {Spec, Dom} |= Req

154

From requirements to software specs (2)

- To map Reqs to Specs:
  - translate goals assigned to software agents in vocabulary of software-to-be: input-output variables (if needed)
  - map (domain) object model elements to their images in the software's object model (if needed)
  - introduce (non-functional) accuracyGoals requiring the consistency between monitored/controlled variables in the environment & their software image (input/output vatiables, database elements)
  - introduce input/output agents to be responsible for such accuracy goals (sensor, actuator & other input/output devices)

155

From requirements to software specs (3)

- Example:
  - Req:

    MotorReversed ⇔ MovingOnRunway
  - TargetSpec:

    Reverse = 'enabled' ⇔ WheelPulses = 'on'
  - accuracyGoals:

    MovingOnRunway ⇔ WheelPulses = 'on'

    expectation on wheelSensor

    MotorReversed ⇔ Reverse = 'enabled'

    expectation on motorActuator

156

Conclusion

- Goal-based reasoning is central to RE for…
  - elaboration of requirements
  - exploration of alternatives
  - conflict management
  - requirements-level exception handling
  - architecture derivation

157

Conclusion (2)

- Goals provide better abstractions for decision makers

  from strategic/business goals
  to technical requirements

- Uniform framework integrating ..
  - current system, system-to-be
    alternative subtrees in goal AND/OR graph
  - different sub-models for different views

158

Conclusion (3)

- Benefits of combining 2 levels
  - semi-formal: for modeling, navigation
  - formal (optional): for precise reasoning
- Benefits of constructive, formal reasoning at goal level

159

Other developments

- Formal & qualitative reasoning about goals:
  - goal refinement, goal mining from scenarios, obstacle analysis, conflict management, requirements reuse
- Goal-oriented requirements animation
- Early model checking:
  - partial models
  - incremental
- Run-time monitoring & resolution of goal violations
- Goal-oriented security management

160

Thanks to the KAOS crew

- UCL, Louvain-la-Neuve: basic research
  - method, techniques

  E. Letier, H. Tran Van, L. Willemet
- CEDITI / IGLOO:
  - tool packaging (semi-formal tools)
  - industrial experience, feedback

  R. Darimont, E. Delor, C. Nève, J.L. Roussel
- CETIC / FAUST:
  - formal analysis tools

  P. Massonet, J.F. Molderez, C. Ponsard, A. Rifaut

161