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)
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)

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
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)

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
Requirements Engineering... what?

- "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"

- "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"

---

Requirements Engineering... what? (2)

WHY?

domain knowledge

goals

operationalization

WHAT?

requirements, assumptions
Requirements Engineering... what? (2)

**WHY?**
- Goals
- Domain knowledge
- Operationalization

**WHAT?**
- Requirements
- Assumptions

**WHO?**
- Responsibility
- Assignment

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
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

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
⇒ system modeling is a core business
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

![Diagram showing requirements and software specifications]

Requirements Engineering... what? (7)

- Requirements are prescriptive assertions formulated in terms of environment phenomena (not necessarily shared)
  \[
  \text{TrainMoving} \Rightarrow \text{DoorsClosed}
  \]

- Software specifications are prescriptive assertions formulated in terms of shared phenomena
  \[
  \text{measuredSpeed} \neq 0 \Rightarrow \text{doorsState} = 'closed'
  \]

- Domain properties are descriptive assertions assumed to hold in the domain
  \[
  \text{TrainMoving} \Leftrightarrow \text{measuredSpeed} \neq 0
  \]

(Jackson, 1995 & Parnas, 1995)
**Requirements Engineering... what? (8)**

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

  ![Diagram showing the 4-variable model]

  \[
  \begin{align*}
  M: \text{monitored variables} & \quad \text{measuredSpeed} \\
  C: \text{controlled variables} & \quad \text{doorsState} \\
  I: \text{input data} & \quad \text{TrainMoving} \\
  O: \text{output results} & \quad \text{doorsState} \\
  \end{align*}
  \]

  \[
  \text{Req} \subseteq M \times C \\
  \text{Spec} \subseteq I \times O
  \]

  \[
  \text{Spec} = \text{Translation} (\text{Req}) \quad \text{such that} \quad \{\text{Spec}, \text{Dom}\} \models \text{Req}
  \]

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

⇒ 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

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, ...)

© A. van Lamsweerde
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

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"
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
Goal-oriented RE ... what?

= use of goals for requirements...
  - elicitation
  - elaboration
  - structuring
  - specification
  - analysis
  - negotiation
  - documentation
  - evolution

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
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"

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, ...

© A. van Lamsweerde
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" $\leftrightarrow$
      on-board train controller, tracking system, station computer,
      passengers, train driver, ...
    "book copy returned on shelves" $\leftrightarrow$ 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" $\leftrightarrow$ station computer
    "reminder issued by end of loan period" $\leftrightarrow$ library software

WHAT are goals?  (5)

- Agent responsible for goal $\Rightarrow$
  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"
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)

WHY are goals needed?

- Criterion for requirements completeness

  REQ is complete if for all G:
  
  \{REQ, EXPECT, Dom\} \models G

- Criterion for requirements relevance

  r in REQ is pertinent if for some G:
  
  r is used in \{REQ, EXPECT, Dom\} \models G

(Yue’87)
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)

WHY are goals needed? (3)

effective passengers transportation

  rapid transportation

    train progress

    no delay

  safe transportation

    no train collision

    doors closed while moving

    no trains on same block

  AND-refinement
WHY are goals needed? (4)

- effective passengers transportation
  - rapid transportation  
  - safe transportation
  - train progress  
  - no delay
  - no train collision
  - doors closed while moving
  - no trains on same block
  - worst-case stopping distance maintained

OR-refinement

WHY are goals needed? (5)

- Goal AND abstraction ⇒ requirements rationale
- Goal AND refinement ⇒
  - user-oriented structuring of documentation
  - traceability
    - strategic objectives ⇒ technical requirements
WHY are goals needed? (6)

- **Goal OR refinement** \( \Rightarrow \)
  identification, validation, negotiation of alternative requirements
  (Dardenne’91, Mylopoulos’92, Chung’00)

- **Goal OR assignment** \( \Rightarrow \)
  identification, validation, negotiation of alternative system boundaries/proposals
  (Dardenne’93)

WHY are goals needed? (7)

- **Roots for conflict detection & resolution**
  (Robinson’89, Boehm’95, van Lamsweerde’98)

  - effective passengers transportation
  - rapid transportation
    - train progress
      - no delay
    - safe transportation
      - no train collision
      - doors closed while moving
  - no trains on same block
WHY are goals needed? (8)

- Support for evolution management
  higher-level goals ⇒ more stable concerns
  \(\text{(Anton'94, ...)}\)
  ⇒ multiple system versions within single model:
  common parent goals, different OR-branches

no train collision

no trains on same block  worst-case stopping distance maintained

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
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

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
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

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, ...
Modeling goals: types

- **Types** define classes of behavior prescribed or preferred
  - **Achieve / Cease goals**: generate behaviors
    \[
    \text{CurrentCondition} \Rightarrow \text{eventually TargetCondition}
    \]
    e.g. Achieve [TrainProgress]
  - **Maintain / Avoid goals**: restrict behaviors
    \[
    \text{CurrentCondition} \Rightarrow \text{TargetCondition}
    \]
    \[
    \text{CurrentCondition} \Rightarrow \text{always TargetCondition unless NewSituation}
    \]
    e.g. Maintain [DoorsClosedWhileMoving]
    \[
    \text{CurrentCondition} \Rightarrow \lnot \text{TargetCondition}
    \]
    e.g. Avoid [TrainsOnSameBlock]
  - **SoftGoals goals**: prefer behaviors when alternatives to be selected

Modeling goals: types (2)

- **SoftGoals vs. Achieve/Maintain**:
  - SoftGoal achievement cannot be established in clear-cut sense
    \[
    \Rightarrow \text{goal satisficing, qualitative reasoning}
    \]
    (Mylopoulos ‘92, Chung ‘00)
  - Achieve/Maintain goal achievement can be verified
    \[
    \Rightarrow \text{goal satisfaction, formal reasoning}
    \]
    (Dardenne ‘93, Darimont ‘96)
**Modeling goals: categories**

Goals may be classified in functional/non-functional categories

```
  goals
  /   \
 functional  non-functional
  |     /   \     /
 satisfaction  information  ...  security  accuracy  performance
  |     |     |     |     |         \
 confidentiality availability integrity time space
```

**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 highest priority in conflict resolution"

more specific types & categories \(\Rightarrow\) more specific heuristics
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)

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
Modeling goals: AND/OR refinement

- **Goal** \( G \) is **AND**-refined into subgoals \( G_1, ..., G_n \) iff achieving \( G_1, ..., G_n \) contributes to achieving \( G \)
  
  the set \( \{G_1, ..., G_n\} \) is called refinement of \( G \)
  
  \( G_i \) is said to contribute positively to \( G \)

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

  \[
  \{G_1, ..., G_n, \text{Dom}\} \models G
  \]

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

  \( R_i \) is called alternative for \( G \)

---

**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 ..."
Modeling goals: \textit{AND/OR} refinement (3)

- \textit{Goal Oriented Requirements Engineering}
- \textit{May 2003}
- © A. van Lamsweerde

Modeling goals: \textit{AND/OR} refinement (4)

- EffectiveAccessToStateOfTheArt
- EffectiveLoanSystem
- EffectiveBiblioSearchSystem

Modeling goals: tips & heuristics

How do you elicit goals?

- analysis of current system
  ⇒ problems, deficiencies
  ⇒ goals of S2B: avoid / reduce them

- search for intentional keywords in preliminary material
  (documents available, interview transcripts)

- later elicitation by refinement & abstraction ...

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 ≠ top-down !!

- by resolution of obstacles, conflicts (cf. below)
  (van Lamsweerde'95, '98, '00)
Goal-Oriented Requirements Engineering

Modeling goals: tips & heuristics (3)

HOW?

EffectivePassengersTransportation

RapidTransportation

SafeTransportation

TrainProgress

Delay

TrainCollision

DoorsClosedWhileMoving

BlockSpeedLimited

MoreTrainsRunning

WHY?

ProgressWhenGoSignal

SignalSetToGo

TrainsOnSameBlock

WorstCaseStoppingDistanceMaintained

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)
Modeling goals: tips & heuristics (5)

- Do not confuse ...
  - OR-refinement ...
  - AND-refinement by case ...

  cf. case analysis:
  \[(\text{Case1} \lor \text{Case2}) \Rightarrow X \equiv (\text{Case1} \Rightarrow X) \land (\text{Case2} \Rightarrow X)\]

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

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 ?
**Modeling goals: tips & heuristics** (7)

Goal Def attribute =
placeholder for such precise definition

- **WorstCaseStopping**
  - Def: A train shall never get so close to a train in front so that if the train stops suddenly (e.g., derailment) the next train would hit it.

- **DistanceMaintained**

- **BookRequestSatisfied**

- **CopyDueSoon**
  - Def: A book without any copy available for loan shall have a copy available within 15 days for the requesting borrower.

- **WhenNotAvailable**

**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
    - ⇒ lightweight analysis provided

  (Darimont’96, Letier’02)
Modeling goals: AND-refinement patterns

- Refinement by case
  - applicable when goal achievement space can be partitioned into cases
    - example of use
      - GoalToBeEnsured
        - WhenCase1
        - WhenCase2

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
      - TargetStateReached
        - MilestoneState Reached
        - FromMilestone

© A. van Lamsweerde
Modeling goals: AND-refinement patterns (3)

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

\[
\text{UnrealizableGoalOnUnMonitorableCondition} \quad \text{UnrealizableGoalOnUnControllableCondition}
\]

\[
\text{GoalOnMonitorableCondition} \quad \text{UnMonitorableCondition} \quad \text{Iff MonitorableCondition}
\]

\[
\text{GoalOnControllableCondition} \quad \text{UnControllableCondition} \quad \text{Iff ControllableCondition}
\]

Resolve lack of monitorability
Resolve lack of controllability

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

Modeling goals: AND-refinement patterns (4)

Example of use

- \text{DoorsClosedWhileMoving}
- \text{DoorsClosedWhileNonZeroSpeed} \quad \text{NonZeroSpeed requirement}
- \text{MovingIff NonZeroSpeed} \quad \text{domain invariant}
- \text{NurseInterventionWhenCriticalPulseRate}
- \text{NurseInterventionWhenAlarm} \quad \text{expectation}
- \text{AlarmIff CriticalPulseRate} \quad \text{requirement}

Resolve lack of monitorability
Resolve lack of controllability
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, ...

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
  \[
  \{ O, \text{Dom} \} \models \neg G \quad \text{obstruction}
  \]
  \[
  \text{Dom } \not\models \neg O \quad \text{domain consistency}
  \]
  (high-level exception)
- **Anticipate obstacles** ...
  \[\Rightarrow\] new, deidealized goals
  - more complete, realistic requirements
  - more robust system
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)

Modeling goals: obstacle analysis (2)

- To identify obstacles to goal \( G \):
  - negate \( G \);
  - find as many AND/OR refinements of \( \neg G \) 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)
Modeling goals: obstacle analysis (3)

MotorReversed Iff MovingOnRunway

MovingOnRunway Iff WheelsTurning

MotorReversed Iff WheelsTurning

NOT MovingOnRunway Iff WheelsTurning

MotorReversed Iff WheelsTurning

MovingOnRunway AndNot WheelsTurning

WheelsTurning AndNot MovingOnRunway

MotorReversed AndNot WheelsTurning

WheelsTurning AndNot MotorReversed

WheelsNotOut

WheelsBroken

Aquaplaning

obstruction

NOT (X1 and X2) equiv not X1 or not X2

OR-refinement (complete)

Warsaw obstacle

Modeling goals: obstacle analysis (4)

WorstCaseStoppingDistanceMaintained

SentCommand ReceivedByTrain

ReceivedCommand ExecutedByTrain

SafeAcceleration Computed

AccelerationSent InTimeToTrain

SentCommand ReceivedByTrain

AccelerationCommand Not SentInTimeToTrain

AccelerationCommand Not ReceivedInTimeByTrain

NotSent SentLate SentToWrongTrain NotReceived Corrupted

ReceivedLate

© A. van Lamsweerde
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)

Modeling goals: obstacle analysis (6)

- Alternative resolution operators (tactics):
  - goal substitution: consider alternative refinement of parent goal to avoid obstruction of child goal
    \[
    \text{MovingOnRunway} \iff \text{WheelsTurning} \rightarrow \text{MovingOnRunway} \iff \text{PlaneWeightSensed}
    \]
  - agent substitution: consider alternative responsibilities
    \[
    \text{WheelSensor} \rightarrow \text{WeightSensor} \\
    \text{OnBoardTrainController} \rightarrow \text{VitalStationComputer}
    \]
  - goal weakening:
    \[
    \text{MovingOnRunway} \iff \text{WheelsTurning} \\
    \rightarrow \text{MovingOnRunway} \iff (\text{WheelsTurning or ...})
    \]
Modeling goals: obstacle analysis (7)

- Alternative resolution operators (cont’d):
  - goal restoration: enforce target condition after goal violation
    BookCopyNotReturnedInTheTime $\rightarrow$ ReminderSent
    WheelsNotOut $\rightarrow$ WheelsAlarmGenerated
  - obstacle prevention: new Avoid goal
    AccelerationCommandCorrupted $\rightarrow$ Avoid [AccelerationCommandCorrupted]
  - obstacle mitigation: tolerate obstacle but mitigate its effects
    OutdatedSpeed/PositionEstimates $\rightarrow$ Avoid [TrainCollision WhenOutdatedTrainInfo]

(van Lamsweerde 2000)

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
    $\rightarrow$ intra-model: refinement, obstruction, conflict
    inter-model: reference, operationalization, responsibility, ...

© A. van Lamsweerde
Modeling goals: conflict analysis

- Goals G1, ..., Gn are divergent iff there exists a boundary condition B:
  
  \[ \{ B, \forall i \ G_i, \text{Dom} \} \models \text{false} \quad \text{inconsistency} \]
  
  \[ \{ B, \forall j \neq i \ G_j, \text{Dom} \} \not\models \text{false} \quad \text{minimality} \]

- Example
  
  \(G_1\): DoorsClosedBetweenStations
  
  \(G_2\): DoorsOpenWhenAlarm
  
  \(B\): AlarmRaisedBetweenStations

(van Lamsweerde'98)

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 \(B = \text{true}\):
    
    \[ \{ G_1, \text{Dom} \} \models \neg G_2 \]
Modeling goals: conflict detection

- EffectivePassengersTransportation
  - RapidTransportation
  - SafeTransportation

- TrainProgress
  - Delay
  - TrainCollision
  - DoorsClosed
  - BlockSpeed

- ProgressWhenGoSignal
  - SignalSetToGo

- TrainsOnSameBlock
- WorstCaseStoppingDistanceMaintained

Modeling goals: conflict detection (2)

- EffectiveAccessToStateOfTheArt
  - EffectiveLoanSystem
  - EffectiveBiblioSearchSystem

- BookRequestSatisfied
  - CopyBorrowed
  - CopyReserve

- CopyDueSoon
  - Availability

- CopyAvailable
  - EffectiveAvailability

- LimitedLoanAmount
  - LimitedLoanPeriods

- E-bookAccess
  - E-lib
  - LimitedLoanPeriods
  - OnlineLoanPeriods

© A. van Lamsweerde
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

Modeling goals: inter-model links

- Object reference:
  
  \[ G \text{ concerns } Ob \iff G's \text{ Def refers to } Ob \]
  
  (generally to its attributes)

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

© A. van Lamsweerde
Modeling goals: inter-model links (2)

- **Goal responsibility:**
  
  \( G \) is assignable to \( Ag \) iff \( G \) can be realized by \( Ag \)

- (Dardenne’93, Letier’02)

Modeling goals: inter-model links (3)

- **Goal operationalization:**
  
  \( G \) is correctly operationalized by \( Op_1, ..., Op_n \) iff the specs of \( Op_1, ..., Op_n \) are necessary & sufficient for ensuring \( G \)

  \[
  \{ \text{Spec}(Op_1), ..., \text{Spec}(Op_n) \} \models G \quad \text{completeness}
  \]

  \[
  G \models \{ \text{Spec}(Op_1), ..., \text{Spec}(Op_n) \} \quad \text{minimality}
  \]

- (Mylopoulos’92, Dardenne’93, Letier’02)
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)

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
Specifying goals in KAOS (2)

- **Textual template**

  
  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 Moving\(\text{iff}\)NonZeroSpeed ]
  
  [ OperationalizedBy OpenDoors, CloseDoors, MoveTrain ]
  
  [ UnderResponsibilityOf TrainController ]
  
  [ FormalSpec  \(\forall\) tr: Train
  
  Moving (tr) \(\Rightarrow\) tr.DoorsState = "closed" ]

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
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

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
Modeling objects in KAOS/UML  (3)

- Association Ass
  - instance = tuple of object instances linked, each playing corresponding role
  - Ass\([O_1, \ldots, O_n]\).Alive noted by predicate \( Ass (O_1, \ldots, O_n) \)
  - 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"

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"
Modeling objects in KAOS/UML (5)

- Attribute \( \text{Att} \) of object \( \text{Ob} \)
  
  function \( \text{Att} : \text{Ob} \rightarrow \text{Range} \) (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)

Modeling objects in KAOS/UML (6)

- A block may hold 0 or 1 train
- A train may be at 0 or 1 station at most
Modeling objects in KAOS/UML

- **Loan**
  - **DateBorrowed**: Date
  - **TimeLimit**: NumberWeeks
  - **DueReturnDate**: Date

- **Borrower**
  - **0..1**
  - **Borrows**
  - **BorrowedBy**
  - **Phone**: String

- **Copy**
  - **0..Max**
  - **BookCopy**
  - **CopyID**

- **Library**
  - **1**
  - **Book**
    - **Keywords**: Topics

- **Period**
- **Registration**
  - **DateRegistered**: Date
  - **Deposit**: Money

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 ... ]**

© A. van Lamsweerde
Modeling objects: tips & heuristics

- Elements of object model should be derived incrementally as they are referenced in the goal model
  ⇒ completeness & pertinence of object model
  ⇒ 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?

Modeling objects: tips & heuristics (2)

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

![Diagram showing examples of modeling objects]

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

![Diagram showing examples of association and attributes association]
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
    ⇒ association
  - X has a domain-independent semantics
    ⇒ aggregation
  - Component & composite objects seem independent
    ⇒ association
  - Component object seems subordinate to composite
    ⇒ aggregation/composition

Modeling objects: tips & heuristics (6)

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

```
BorrowerRequest Generates Loan

GoSignal Activates Train
```

**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
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 $\rightarrow$ goal = requirement
    - environment agent $\rightarrow$ goal = requirement
  - alternative agent assignments
    $\Rightarrow$ alternative system boundaries/proposals
  - agent controls behaviors by performing operations that operationalize the goals assigned to it

Modeling agents in KAOS (2)

- An agent is modeled & specified by its features ...
  - Name, Def
  - (features as an object: dom-specific attributes & associations, $\text{DomInvar/Init} \rightarrow \text{in object model}$)
  - 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)
Modeling agents in KAOS (3)

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
  
  Responsible for SafeCommandMessage, CommandSentInTime
  
  DependsOn TrackingSystem For AccurateEstimateOfSpeed&Position
  
  Performs SendCommand
A useful, derivable diagram: context diagram

- A context diagram links agents through their interfaces
  - interface = monitored/controlled variables (attributes)
  - generation of link (ag1, ag2) from agent model:
    - variable controlled by ag1 & monitored by ag2
    - variable monitored by ag1 & controlled by ag2

(Jackson 1995, Letier 2001)

A useful, derivable diagram: context diagram (2)

- Example:

Train/CurrentSpeed, Train/CurrentLoc

Train/ActuatedAcceleration

Tracking System

Train/MeasuredSpeed, Train/MeasuredLoc

Speed&Accel Controller

Command/CommandedAcceleration

OnBoard Controller
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

\[ G: \text{CurrentCondition (monitoredVariables)} \rightarrow \text{eventually/always TargetCondition (controlledVariables)} \]
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

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  \( \text{Op} \subseteq \text{InputState} \times \text{OutputState} \)
  - 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
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

Operations & goals (recall)

- Goal operationalization:
  $G$ is correctly operationalized by $Op_1, ..., Op_n$ iff the specs of $Op_1, ..., Op_n$ are necessary & sufficient for ensuring $G$
  \[
  \{\text{Spec}(Op_1), ..., \text{Spec}(Op_n)\} = G \quad \text{completeness}
  \]
  \[
  G = \{\text{Spec}(Op_1), ..., \text{Spec}(Op_n)\} \quad \text{minimality}
  \]

(Mylopoulos’92, Dardenne’93, Letier’02)
Modeling operations: functional view

- 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 $G$
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

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")

  ⇒ A use case operationalizes the goals ensured by the operations in it

- Generation of use cases from the operation & agent models is straightforward
A useful, derivable diagram: use case diagram (2)

- **BorrowBookCopy**
  - **Precondition**: NotAvailable, TooManyCopies
  - **Include**: ExtendLoan
  - **Boundary**: interaction

- **ExtendLoan**
  - **Include**: CheckForReservation

- **ReturnBookCopy**
  - **Include**: UpdateCatalog

- **AddBookCopy**

- **RemoveBookCopy**

- **CheckForReservation**

- **WhoBorrowedWhat?**

- **WhichBooksOnTopic?**

- **ReserveBookCopy**
  - **Include**: NotAvailable

- **RefuseLoan**

**Staff**

**Borrower**

**Browser**

**LibrarySoftware**

---

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

- **OpenDoors**
- **CloseDoors**

- **Train**
  - **Speed&Accel Controller**
    - **SendAccelerationCommand**

- **OnBoardController**
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)

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
Modeling behaviors: scenarios (2)

- OnBoardTrain Controller
  - Arrival
  - DoorsOpening
  - DoorsClosing
  - Start
  - Arrival
  - DoorsOpening

- Train
  - Entrance
  - Exit

- Passenger

Modeling behaviors: scenarios (3)

- Borrower
  - BookRequest (BorId, BookId)
  - event attributes

- LoanManager
  - LoanQtyOK? (BorId)
  - CpyAvailable? (BookId)
  - OK-Available (CopyId)
  - Reserved? (BookId)
  - self-interaction

- CopyManager
  - OK-Book (CopyId)
  - Borrowed (CopyId)
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

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

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 \rightarrow val_1, ..., att_n \rightarrow val_n, assoc_1 \rightarrow link_1, ..., assoc_p \rightarrow link_p\}
  e.g.
  \{CurrSpeed \rightarrow 0, CurrLoc \rightarrow X, DoorsState \rightarrow 'closed', At \rightarrow (tr, st)\}
- State of an entity/association/agent's SM:
  set of states sharing same value for some behavioral attribute/association
  e.g.
  \{CurrSpeed \rightarrow 0, CurrLoc \rightarrow X, DoorsState \rightarrow 'closed', At \rightarrow (tr, st)\}
  \{CurrSpeed \rightarrow 5, CurrLoc \rightarrow Y, DoorsState \rightarrow 'closed', At \rightarrow nil\}
  belong to state "doorsClosed" of Train SM
Modeling behaviors: state machines (2)

- Guarded state transition:

```
St₁ \[ ev [guard] \] \rightarrow \[ \text{the object gets to state } St₂ \]
```

- if it is in \( St₁ \) and \( ev \) occurs
- and only if the guard is true

Modeling behaviors: state machines (3)

- Nested states...
  - sequential: super state is diagram composed of sequential sub-states (cf. Statecharts)
Modeling behaviors: state machines (3)

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

![State machine diagram for train states: Synchronized model for doors and speed states.](image-url)

**TrainState**

- **SpeedState**
  - Stopped: AccelerComnd \([\text{Acceler} > 0]\) \(\rightarrow\) Stopped \([\text{Speed} = 0]\)
  - Moving: AccelerComnd \([\text{Acceler} > 0]\), AccelerComnd \([\text{Acceler} \leq 0]\)
    - Accelerating
    - Decelerating

**Doors**

- doorsClosed \(\rightarrow\) DoorsOpening \([\text{AtStation and Speed} = 0]\) \(\rightarrow\) doorsOpen
  - DoorsClosing
State machines are derivable from KAOS models

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
The KAOS goal-oriented RE method

1. Domain analysis: refine/abstract goals
   - SafeTransportation
   - NoTrainSameBlock

2. Domain analysis: derive/structure objects
   - Train
   - Block
   - On
   - 0:1
The KAOS goal-oriented RE method

1. Domain analysis: refine/abstract goals
   - SafeTransportation

2. Domain analysis: derive/structure objects
   - On Train 0:1 Block

3. S2B analysis: enriched goals (alternatives)
   - NoTrainSameBlock SafeCmd

4. S2B analysis: enriched objects from new goals
   - Driving Command

© A. van Lamsweerde
The KAOS goal-oriented RE method

1. Domain analysis: refine/abstract goals
   - SafeTransportation
   - NoTrainSameBlock
   - SafeCmd

2. Domain analysis: derive/structure objects
   - On
   - 0:1 Block
   - Driving
   - Command

3. S2B analysis: enriched goals (alternatives)
   - SafeAccel

4. S2B analysis: enriched objects from new goals
   - SafeComd
   - NoTrainSameBlock

5. Responsibility analysis: agent OR-assignment

1-5. Obstacle & conflict analysis
The KAOS goal-oriented RE method

1. Domain analysis: refine/abstract goals
   - SafeTransportation
   - NoTrainSameBlock
   - SafeComd

2. Domain analysis: derive/structure objects
   - Train
   - Block
   - Command
   - OnBoardController
   - SafeAccel

3. S2B analysis: enriched goals (alternatives)
   - SafeComd
   - OnBoardController

4. S2B analysis: enriched objects from new goals
   - Command
   - At any time: abstraction (e.g. from scenarios)

5. Responsibility analysis: agent OR-assignment
   - Driving
   - Send Command
   - OBC
   - NoTrainSameBlock

6. Operationalization & behavior analysis
   - SafeComd
   - NoTrainSameBlock
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 threshold 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 switch 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)
The goal-oriented RE method in action (3)

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

**PumpOnWhenHighWater**
- **Def**: The pump must be on when the water level in the sump is above the high water level

**EvacuationWhenPumpFailure**
- **Def**: If, due to a failure of the pump, the water cannot be pumped out, the mine must be evacuated within 1 hour

The goal-oriented RE method in action (4)

- First object model after Def of every preliminary goal:

**MethaneAlarm**
- **AirflowAlarm**
- **COAlarm**

**Sump**
- **WaterLevel: Depth**
- **HasPump**

**Pump**
- **Motor**: {on, off}
- **Failure**: Bool

**Mine**
- **Contains**

**Miner**
- **Inside**

**Operator**
- **Informed**

**GasAlarm**
- **Raising**

**Operating**

© A. van Lamsweerde
The goal-oriented RE method in action (5)

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

The goal-oriented RE method in action (6)

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

© A. van Lamsweerde
The goal-oriented RE method in action (7)

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

```
PumpOnWhenHighWater
HighWaterDetected
PumpOnWhenHighWaterDetected
```

HOW?

```
PumpOnWhenSwitchOn
WhenHighWaterDetected
```

milestone-driven refinement
resolve lack of controllability

The goal-oriented RE method in action (8)

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

```
PumpOnWhenHighWater
HighWaterDetected
PumpOnWhenHighWaterDetected
```

```
PumpSwitchOn
WhenHighWaterDetected
```

```
Sensor
Controller
Actuator
```

```
PumpCtrl/HighWaterSignal
```

© A. van Lamsweerde
The goal-oriented RE method in action (9)

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

  ![Diagram of operationalization process](image)

  **PumpSwitch On**
  - When HighWaterDetected

  **PumpSwitch Off**
  - When TooMuchMethane

  **SwitchPumpOn**
  **SwitchPumpOff**

  **Pre**: The pump switch is off and the high water signal becomes on and gas level is below critical
  **Post**: The pump switch is on

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
The goal-oriented RE method in action (10)

6: Scenario modeling (cont’d)

: WaterLevelSensor
  WaterTooHigh → PumpController
  WaterOK → PumpActuator

The goal-oriented RE method in action (11)

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

151

© A. van Lamsweerde
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

From requirements to software specs

- Requirements vs. software specifications: recall

\[
\text{Input Devices (e.g. sensors)} \\
\text{TrainMoving} \quad \text{measuredSpeed} \\
\text{I: input data} \\
\text{M: monitored variables} \\
\text{Environment} \\
\text{DoorsClosed} \quad \text{doorsState} \\
\text{C: controlled variables} \\
\text{SoftwareToBe} \\
\text{O: output results} \\
\text{Output Devices (e.g. actuators)}
\]

\[
\text{Req} \subseteq M \times C \\
\text{Spec} \subseteq I \times O \\
\text{Spec} = \text{Translation (Req)} \quad \text{such that} \quad \{\text{Spec, Dom}\} \models \text{Req}
\]
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 variables, database elements)
  - introduce input/output agents to be responsible for such accuracy goals (sensor, actuator & other input/output devices)

From requirements to software specs (3)

- **Example:**
  - **Req:**
    \[ \text{MotorReversed} \iff \text{MovingOnRunway} \]
  - **TargetSpec:**
    \[ \text{Reverse} = \text{'enabled'} \iff \text{WheelPulses} = \text{'on'} \]
  - **accuracyGoals:**
    \[ \text{MovingOnRunway} \iff \text{WheelPulses} = \text{'on'} \]
    \[ \text{expectation on wheelSensor} \]
    \[ \text{MotorReversed} \iff \text{Reverse} = \text{'enabled'} \]
    \[ \text{expectation on motorActuator} \]
Conclusion

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

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
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

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
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. Massonnet, J.F. Molderez, C. Ponsard, A. Rifaut