KAOS in Action: The BART System

Emmanuel Letier and Axel van Lamsweerde

Dept. Ingénierie Informatique, Univ. Louvain
B-1348 Louvain-la-Neuve (Belgium)
{eletier, avl}@info.ucl.ac.be
Outline

Introduction
Goal-Oriented RE with KAOS

The BART Case Study
Identifying Goals from Initial Document
Formalizing Goals and Identifying Objects
Elaborating the Goal Structure
Identifying Agents and Responsibilities
Deriving Monitored and Controlled Quantities
Exploring Alternative Responsibility Assignments
Identifying Operations
Operationalizing Goals through Strengthened Operations

Conflict Analysis: an Example

Obstacle Analysis
   Obstacle Generation
   Obstacle Resolution

Conclusion
Goal-Oriented Requirement Engineering with KAOS

Goal Model

Object Model

Responsibility Model

Agent Interface Model

Operation Model

AND/OR goal refinement
Goals, Requirements, Assumptions
+ DomProps

2-level language:
semantic net level
formal assertions

software agents
+ environment agents

Train
On
TrackSegment

0:1

Goal: NoTrainCollision

Operation: SendComand

DomPre → Sent(m, tr)

DomPost Sent(m, tr)

ReqPostFor SafeAcceleration

m.Acceleration ≤ F(tr.Loc, tr.Speed, ...)

KAOS in Action: the BART System
Goal Identification From Initial Document

ServeMorePassengers

Minimize[Costs]

Min[DevlptCosts]

Min[OperationalCosts]

Maintain[WCSDistBetweenTrains]

Maintain[TrackSegmentSpeedLimit]

Avoid[TrainEnteringClosedGate]

SafeTransport

Min[TimeBetweenStations]

NewTracksAdded

TrainsMoreClosely Spaced

...
Goal Identification From Initial Statement

ServeMorePassenger

Minimize[Costs]

Min[DevelopCosts]

Min[OperationalCosts]

Minimize[PowerUsage]

Minimize[StressOnEquipment]

SafeTransport

Maintain[WCSDistBetweenTrains]

Maintain[TrackSegmentSpeedLimit]

Avoid[TrainEnteringClosedGate]

PsgerComfort

ServeMorePassenger

TrainsMoreCloselySpaced

NewTracksAdded

Min[TimeBetweenStation]

Minimize[Costs]

Further Goals identified by asking WHY and HOW questions
Formalizing Goals and Identifying Objects (1)

Goal: Maintain[TrackSegmentSpeedLimit]

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

**Formal Def**
\[ \forall \ tr: \text{Train}, s: \text{TrackSegment} : \]
\[ \text{On}(tr, s) \Rightarrow tr.\text{Speed} \leq \]
Formalizing Goals and Identifying Objects (2)

Goal: Maintain[WCSDistBetweenTrains]

Definition: A train should never get so close to a train in front so that if the train in front stops suddenly (e.g., derailment) the next train would hit it.

Formal Def: \( \forall \ tr1, tr2: Train \ :
Following(tr1, tr2) \Rightarrow tr1.Loc - tr2.Loc > tr1.WCSDist \)

Following

Train

Speed: SpeedUnit
Loc: Location
WCSDist: Distance

On

TrackSegment

SpeedLimit: SpeedUnit
...
**Goal**  Avoid[TrainEnteringClosedGate]

**Definition** A train should not enter a closed gate if it can, (i.e. if it is possible for the train to stop before the gate)

**FormalDef**  \( \forall tr : \text{Train}, g : \text{Gate}, s : \text{TrackSegment} : \)
\[ g.\text{status} = \text{'closed'} \quad \text{Since} \quad tr.\text{Loc} - g.\text{Loc} > tr.\text{WCSDist} \land \text{HasGate}(s, g) \]
\[ \Rightarrow \]
\[ \neg \@ \text{On}(tr, s) \]

"Since":
"Until" in the past
Eliciting New Goals: WHY Questions (1)

**Goal** Avoid[TrainCollisions]
**Definition** Trains should never collide
**FormalDef** $\forall \ tr1, tr2: \text{Train}: \Box \neg \text{Collision}(tr1, tr2)$

**Goal** Avoid[TrainDerailment]
**Definition** Trains should never derail
**FormalDef** $\forall \ tr: \text{Train}: \Box \neg \text{Derailment}(tr)$
Eliciting New Goals: WHY Questions (2)

**Goal** Avoid\([\text{TrainOnSwitchInWrongPostion}]\)

**Definition** When a train is on a switch, the switch should be in the direction of travel of the train

**FormalDef**  \( \forall \, \text{tr}: \text{Train}, \, \text{sw}: \text{Switch}: \)
\[
\text{On}(\text{tr}, \, \text{sw}) \Rightarrow \text{sw}.\text{Position} = \text{tr}.\text{Direction}
\]
Eliciting New Goals: HOW Questions (1)

WCSDist: the physical Worst-Case Stopping Distance based on the *physical* speed of the train
Following(tr1, tr2)
⇒
tr1.Loc - tr2.Loc > tr1.WCSDist

\[ \text{DomProp:} \]
\[ \text{tr.Loc}' = \text{tr.Loc} + \delta \text{tr.Speed} \]
\[ \text{tr.Speed}' = \text{tr.Speed} + \delta \text{tr.acc} \]

Following(tr1, tr2)
⇒
\[ \text{tr1.Acc}_{CM}' \leq F(\text{tr1.Loc}, \text{tr2.Loc}, \text{tr1.Speed}, \text{tr2.Speed}) \]
\[ \land \]
\[ \text{tr1.Speed}_{CM}' > \text{tr1.Speed} \quad (!) \]

∀ tr: Train
\[ \text{tr.Acc}_{CM} \geq 0 \Rightarrow \text{tr.Acc}' \leq \text{tr.Acc}_{CM} \]
\[ \land \]
\[ \blacksquare_{\text{MCdelay}} \text{tr.Acc}_{CM} < 0 \Rightarrow \text{tr.Acc}' \leq 0 \]
\[ \land \]
\[ \text{tr.Speed} \leq \text{tr.Speed} \]
⇒ \[ \text{tr.Speed}' \leq \text{tr.Speed}_{CM} \]

∀ tr2: Train
\[ \phantom{\text{tr.Acc}_{CM}} \]
\[ \square \]
\[ \text{tr2.Speed}' \geq \text{tr2.Speed} - \delta \text{MaxBrakeRate} \]
Eliciting New Goals: HOW Questions (1)

Avoid
[TrainsCollisions]

Maintain
[WCSDistBetweenTrains]

Maintain
[SafeSpeed/AccelerationCommanded]

Maintain
[SafeTrainResponse ToCommand]

Maintain
[NoSuddenStop OfPrecedingTrain]

Following\(\text{tr}_1, \text{tr}_2\)
\[\Rightarrow\]
\[\text{tr}_1.\text{Loc} - \text{tr}_2.\text{Loc} > \text{tr}_1.\text{WCSDist}\]

\[\forall \text{tr}: \text{Train}\]
\[\text{tr}.\text{Acc}_{\text{CM}} \geq 0 \Rightarrow \text{tr}.\text{Acc}' \leq \text{tr}.\text{Acc}_{\text{CM}}\]
\[\Rightarrow \text{tr}.\text{Speed}' \leq \text{tr}.\text{Speed}_{\text{CM}}\]

\[\forall \text{tr}_2: \text{Train}\]
\[\neg\]
\[\text{tr}_2.\text{Speed}' \geq \text{tr}_2.\text{Speed} - \delta \text{MaxBrakeRate}\]

DomProp:
\[\text{tr}.\text{Loc}' = \text{tr}.\text{Loc} + \delta \text{tr}.\text{Speed}\]
\[\text{tr}.\text{Speed}' = \text{tr}.\text{Speed} + \delta \text{tr}.\text{Acc}\]

WCSDist : the physical Worst-Case Stopping Distance based on the \textit{physical} speed of the train
Eliciting New Goals: HOW Questions (2)

- Avoid [TrainCollisions]
- Maintain [WCSDistBetweenTrains]
  - Maintain [SafeSpeed/AccelerationCommanded]
  - Maintain [AccurateSpeed/PositionEstimates]
  - Maintain [SafeCommandToFollowingTrain BasedOnSpeed/PositionEstimates]
  - Maintain [SafeTrainResponse ToCommand]
  - Maintain [NoSuddenStop OfPrecedingTrain]
∀ tr: Train, ∃ ti: TrainInfo: Tracking(ti, tr)

∀ tr: Train, ti: TrainInfo:
Tracking(ti, tr) ⇒ ❑

FollowingInfo(ti1, ti2) ∧ Tracking(ti1, tr1) ∧ Tracking(ti2, tr2)
⇒

tr1.Acc_{CM}' ≤ F (ti1.Loc + ti1.LDev, ti2.Loc - ti2.Ldev,
∧

tr1.Speed_{CM}' > ti1.Speed + ti1.Sdev

∀ tr: Train, ∃! ti: TrainInfo: Tracking(ti, tr)
Eliciting New Goals: HOW Questions (2)

- Avoid [TrainsCollisions]
- Maintain [WCSDistBetweenTrains]
- Maintain [SafeSpeed/AccelerationCommanded]
- Maintain [SafeTrainResponse ToCommand]
- Maintain [NoSuddenStop OfPreceedingTrain]
- Maintain [AccurateSpeed/PositionEstimates]
- Maintain [SafeComandToFollowingTrain BasedOnSpeed/PositionEstimates]

∀ tr: Train, ∃! ti: TrainInfo: Tracking(ti, tr)

∀ tr: Train, ti: TrainInfo:
Tracking(ti, tr) ⇒ □

\[ \text{FollowingInfo}(ti1, ti2) \land \text{Tracking}(ti1, tr1) \land \text{Tracking}(ti2, tr2) \]
⇒

\[ tr1.acc_{CM}' \leq F(t1.Loc + t1.LDev, t2.Loc - t2.Ldev, t1.speed + t1.Sdev, t2.Speed - t2.Sdev) \]

\[ \land \]

\[ tr1.Speed_{CM}' > t1.Speed + t1.Sdev \]
Eliciting New Goals: HOW Questions (3)

- **Avoid**
  - [TrainCollisions]

- **Maintain**
  - [WCSDistBetweenTrains]
    - **Maintain**
      - [SafeSpeed/AccelerationCommanded]
    - **Maintain**
      - [SafeTrainResponse ToCommand]
    - **Maintain**
      - [NoSuddenStop OfPrecedingTrain]
      - **Maintain**
        - [AccurateSpeed/PositionEstimates]
      - **Maintain**
        - [SafeComandToFollowingTrain BasedOnSpeed/PositionEstimates]

- **Achieve**
  - [CmdMsgSentInTime]
- **Maintain**
  - [SafeCmdMsg]
- **Achieve**
  - [SentCmdMsg DeliveredInTime]
- **Maintain**
  - [DeliveredCmdMsg Exercised]
∀ cm: CommandMessage, ti1, ti2: TrainInfo
  cm.Sent ∧ cm.TrainID = ti1.TrainID
  ∧ FollowingInfo(ti1, ti2)
  ⇒ cm.Acc ≤ F (ti1.Loc + ti1.LDev, ti2.Loc - ti2.Ldev,
                  ti1.Speed + ti.Sdev, ti2.Speed - ti2.Sdev)
  ∧ cm.Speed > ti1.Speed + ti1.Sdev
Eliciting New Goals: HOW Questions (3)

∀ cm: CommandMessage, ti1, ti2: TrainInfo
  cm.Sent ∧ cm.TrainID = ti1.TrainID
  ∧ FollowingInfo(ti1, ti2)
  ⇒
  cm.Acc ≤ F (ti1.Loc+ ti1.LDev, ti2.Loc - ti2.Ldev,
  ∧ cm.Speed > ti1.Speed+ ti1.Sdev
Identifying Potential Responsibility Assignments

Avoid
[TrainCollisions]

Maintain
[WCSDistBetweenTrains]

Maintain
[SafeSpeed/AccelerationCommanded]

Maintain
[SafeTrainResponse ToCommand]

Maintain
[NoSuddenStop OfPrecedingTrain]

Maintain
[SafeComandToFollowingTrain BasedOnSpeed/PositionEstimates]

Achieve
[CmdMsgSentInTime]

Maintain
[SafeCmdMsg]

Achieve
[SentCmdMsg DeliveredInTime]

Maintain
[DeliveredCmdMsg Exercised]

OnBoard TrainController

Tracking System

OnBoard TrainController

Speed/Acceleration ControlSystem

Communication Infrastructure
Goal Maintain[SafeCmdMsg]

Formal Def \( \forall cm: Command\Message, ti_1, ti_2: Train\Info \)
\( \quad cm.\text{Sent} \land cm.\text{TrainID} = ti_1.\text{TrainID} \)
\( \quad \land \text{FollowingInfo}(ti_1, ti_2) \)
\( \quad \Rightarrow \)
\( \quad cm.\text{Acc} \leq F (ti_1.\text{Loc} + ti_1.\text{LDev}, ti_2.\text{Loc} - ti_2.\text{Ldev}, \)
\( \quad ti_1.\text{Speed} + ti.\text{Sdev}, ti_2.\text{Speed} - ti_2.\text{Sdev}) \)
\( \quad \land cm.\text{Speed} > ti_1.\text{Speed} + ti_1.\text{Sdev} \)
Alternative Goal Refinements and Responsibility Assignments

==> different design: fully distributed system

Maintain [WCSDistBetweenTrains]

Or

Maintain [PreceedingTrainSpeed/Position KnownToFollowingTrain]

Maintain [SafeAccelerationBasedOn PreceedingTrainSpeed/Position]

Maintain [NoSuddenStop OfPreceedingTrain]

Maintain [AccurateSpeed/PositionEstimates]

Achieve [PreceedingTrainSpeed/Position CommunicatedToFollowingTrain]

Resp

Tracking System

OnBoard TrainController

Communication Infrastructure

Resp

Resp

Resp
Identifying Operations and DomPre/Post

= Identify state transitions relevant to goals

Goal Maintain[SafeCmdMsg]
  FormalDef \( \forall cm: \text{CommandMessage}, \, ti1, \, ti2: \text{TrainInfo} \)
  \( cm.\text{Sent} \land cm.\text{TrainID} = ti1.\text{TrainID} \)
  \( \land \text{FollowingInfo}(ti1, \, ti2) \)
  \( \Rightarrow \)
  \( cm.\text{Acc} \leq F(ti1.\text{Loc}+ ti1.\text{LDev}, \, ti2.\text{Loc} - ti2.\text{Ldev}, \)
  \( ti1.\text{Speed} + ti.\text{Sdev}, \, ti2.\text{Speed} - ti2.\text{Sdev}) \)
  \( \land cm.\text{Speed} > ti1.\text{Speed} + ti1.\text{Sdev} \)

==>

Operation SendCommandMessage
  Input Train \{arg\} \{tr\}
  Output CommandMessage \{res\} \{cm\}
  DomPre \neg cm.\text{Sent}
  DomPost cm.\text{Sent} \land cm.\text{TrainID} = \text{tr.ID}
Operationalizing Goals

Goal Maintain[SafeCmdMsg]

FormalDef \( \forall cm: \text{CommandMessage}, ti1, ti2: \text{TrainInfo} \)
\( cm.\text{Sent} \land cm.\text{TrainID} = ti1.\text{TrainID} \)
\( \land \text{FollowingInfo}(ti1, ti2) \)
\( \Rightarrow \)
\( cm.\text{Acc} \leq F (ti1.\text{Loc} + ti1.\text{LDev}, ti2.\text{Loc} - ti2.\text{Ldev}, \)
\( ti1.\text{Speed} + ti.\text{Sdev}, ti2.\text{Speed} - ti2.\text{Sdev}) \)
\( \land cm.\text{Speed} > ti1.\text{Speed} + ti1.\text{Sdev} \)

implies

Operation SendCommandMessage

Input Train {arg tr}

TrainInfo

Output CommandMsg {res cm}

DomPre \( \neg cm.\text{Sent} \)

DomPost cm.\text{Sent} \land cm.\text{TrainID} = tr.\text{ID} 

ReqPostFor [SafeCmdMsg]

Tracking(ti1, tr) \land Following(ti1, ti2)

\( \Rightarrow \)

\( cm.\text{Acc} \leq F (ti1.\text{Loc} + ti1.\text{LDev}, ti2.\text{Loc} - ti2.\text{Ldev}, ti1.\text{Speed} + ti.\text{Sdev}, ti2.\text{Speed} - ti2.\text{Sdev}) \)
\( \land cm.\text{Speed} > ti1.\text{Speed} + ti1.\text{Sdev} \)

ReqTrigFor [CmdMsgSentInTime]

\( \leq 1/2 \text{ sec} \rightarrow \exists cm2: \text{CommandMessage}: cm2.\text{Sent} \land cm2.\text{TrainID} = tr.\text{ID} \)
This is not the end of the story ...
Conflict Analysis: An Example

ServeMorePsgers

Min [DistBetweenTrains]

Max [TrainSpeed]

SafeTransport

DistanceBetweenTrains IncreasesWithCmdedSpeed

Maintain [CmdedSpeedCloseToPhysicalSpeed]

\[ \text{tr.Acc}_{CM} \geq 0 \]
\[ \Rightarrow \]
\[ \text{tr.Speed}_{CM} \leq \text{tr.Speed} + \text{fn}(\text{dist_obstacle}) \]

Maintain [CmdedSpeedAbove7mphOfPhysicalSpeed]

\[ \text{tr.acc}_{CM} \geq 0 \]
\[ \Rightarrow \]
\[ \text{tr.Speed}_{CM} > \text{tr.Speed} + 7 \]

\( \diamond (\exists \text{tr: Train}): \)
\[ \text{tr.Acc}_{CM} \geq 0 \]
\[ \land \]
\[ \text{fn}(\text{dist_obstacle}) \leq 7 \]

boundary condition for conflict

ServeMorePsgers

Min [DistBetweenTrains]

Max [TrainSpeed]

ServeMorePsgers

Min [DistBetweenTrains]

Max [TrainSpeed]

SmoothMove

LimitedAccelerationWhen CmdedSpeedAbove7mph OfPhysicalSpeed

Maintain [CmdedSpeedCloseToPhysicalSpeed]

\[ \text{tr.Acc}_{CM} \geq 0 \]
\[ \Rightarrow \]
\[ \text{tr.Speed}_{CM} \leq \text{tr.Speed} + \text{fn}(\text{dist_obstacle}) \]

Maintain [CmdedSpeedAbove7mphOfPhysicalSpeed]

\[ \text{tr.acc}_{CM} \geq 0 \]
\[ \Rightarrow \]
\[ \text{tr.Speed}_{CM} > \text{tr.Speed} + 7 \]

\( \diamond (\exists \text{tr: Train}): \)
\[ \text{tr.Acc}_{CM} \geq 0 \]
\[ \land \]
\[ \text{fn}(\text{dist_obstacle}) \leq 7 \]

boundary condition for conflict
Conflict Resolution

Note: $fn(dist\_obst)$ increases with $dist(obst)$

$$tr.\text{Acc}_{CM} \geq 0 \Rightarrow tr.\text{Speed}_{CM} > tr.\text{Speed} + 7 \lor fn(dist\_obst) \leq 7$$

Rationale: if boundary condition is true, priority is to avoid going into deceleration mode
Obstacle Analysis

• Obstacle = high-level exception

Obstacle $O$ obstructs goal $G$ iff

1. $\{O, \text{Dom} \} \models \neg G$ (Obstruction)
2. $\text{Dom} \models \neg O$ (Domain Consistency)

• Handle obstacles at RE time

===> identification of new requirements

===> more robust system

Handling obstacles during goal-oriented requirements elaboration

1. Identify obstacles
   -> formal techniques for generating obstacles from goal formulations
   -> heuristics as lightweight rules of thumb
2. Generate alternative obstacle resolutions
   -> resolution operators ==⇒ new goals/requirements
3. Alternative evaluation and selection
• Goal-anchored form of Fault-Tree construction

• Formal techniques to generate obstacles from goal formulations

Generating Alternative Obstacle Resolutions

Goal Substitution
= choose alternative goal

\[ \text{CmdMsgSentLate Obstructs Achieve[CmdMsgSentInTime] UnderResponsibilityOf Speed/AccelerationControlSystem} \]

\[ \Rightarrow \text{alternative design: acceleration calculated by on-board train controller} \]

Agent Substitution
= change responsibility assignment for obstructed goal

\[ \text{UnsafeAccelerationInCmdMsg Obstructs SafeAccelerationInCmdMsg UnderResponsibilityOf Speed/AccelerationControlSystem} \]

\[ \Rightarrow \text{UnderResponsibilityOf VitalStationComputer} \]

Obstacle Prevention
= add new goal: \( \neg O \)

\[ \text{ImpossibleChangeInTrainSpeed/PositionEstimates Obstructs AccurateSpeed/PositionEstimates} \]

\[ \Rightarrow \text{New Goal: Avoid[ImpossibleTrainInfoChange]} \]

( to be assigned as responsibility of TrackingSystem OR StationComputer )

Goal Deidealization
= weakening goal to make obstruction disappear
Obstacle Mitigation
= add new goal that tolerates obstacle but mitigates its consequences e.g.

⇒ derivation of new requirements

Message Origination Time Tag attribute
Conclusions

• Systematic derivation of requirements from goals
  (required pre/post/trigger conditions, monitored/controlled objects)

• Goal formalization
  ==> refinement correctness proof
  ==> conflict identification/resolution
  ==> obstacle generation/resolution

• Goal-oriented explanation of requirements

• Goal structure provides structure for requirements document

• Separation of concerns: requirements vs. assumptions vs. domain properties

• Exploration of alternative system proposals