Goal-Oriented Requirements Engineering: the Way to High Assurance Systems

Axel van Lamsweerde

University of Louvain

B-1348 Louvain-la-Neuve (Belgium)

Canadian Software Requirements Symposium '03,
Hamilton, 26/05/03

High Assurance Systems

McLean'95

"Systems where compelling evidence is required that the system delivers its services in a manner that satisfies certain critical properties such as

- safety,
- security,
- fault tolerance,
- survivability"

Outline

- Introduction
  - Problems & challenges with HAS
  - Problems & challenges with RE for HAS
- A goal-oriented RE method in action
  - Goal refinement & abstraction
  - Analysis of obstacles & conflicts
  - Goal operationalization
- Goal-based reasoning for higher assurance
- Conclusion

Problems & challenges with HAS

- The later defects are found,
  the more expensive & dangerous they are ...

  ⇓

  - start caring for high assurance at requirements engineering time
  - preserve high assurance at every transition to downstream products (architecture, code)

Problems & challenges with HAS (2)

- A posteriori detection/fix of defects may endlessly generate other defects …

  ⇓

  – adopt a constructive approach where high assurance is provided by construction

Problems & challenges with HAS (3)

- High assurance requires much stronger level of confidence …

  ⇓

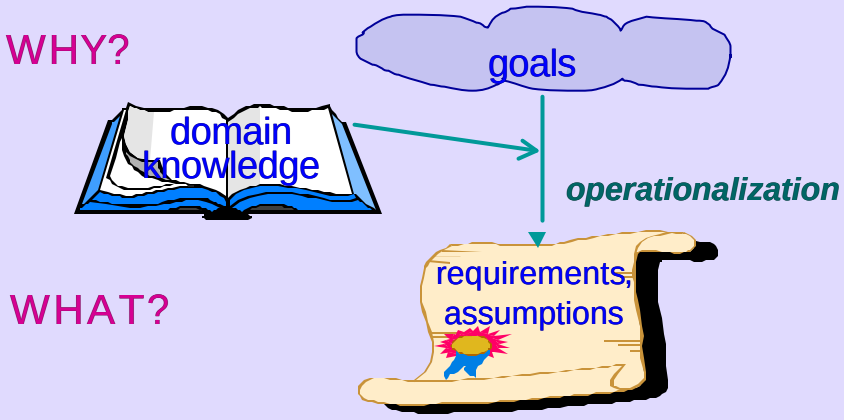
  – *stronger confidence* requires formal elaboration & analysis, supported by tools

  – *usability* at requirements engineering time requires lightweight techniques

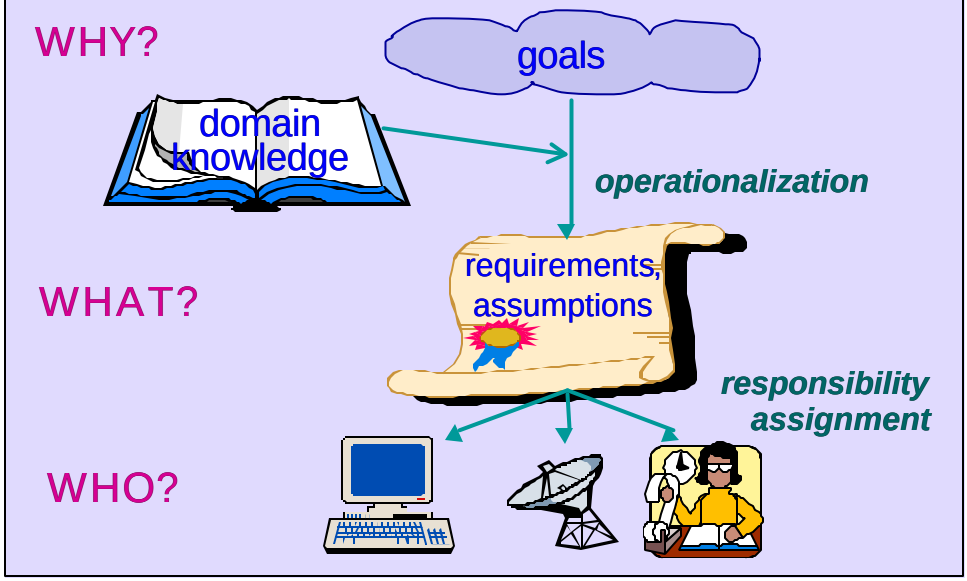
Problems & challenges with RE for HAS

- Requirements are not there,
  you have to elicit them
  & structure them
- RE [1] translation informal requirements into
  ± formal model

What is RE about?



What is RE about?



Problems & challenges with RE for HAS

- Requirements elaboration is hard ...
  - ranges from high-level, strategic objectives to detailed, technical requirements
  - involves software + environment
  - requires evaluation of alternatives
  - raises conflicting concerns
  - requires anticipation of unexpected behaviors (for requirements completeness, system robustness)

HAS: requirements on requirements elaboration process ...

- goal-oriented: to ensure that operational requirements meet safety, security, survivability objectives

HAS: requirements on requirements elaboration process ...

- goal-oriented: to ensure that operational requirements meet safety, security, survivability objectives
- incremental: for early analysis of partial models
- systematic: for analyst guidance & req completeness

HAS: requirements on requirements elaboration process ...

- goal-oriented: to ensure that operational requirements meet safety, security, survivability objectives
- incremental: for early analysis of partial models
- systematic: for analyst guidance & req completeness
- based on multiple models: for capturing multiple facets
  intentional, structural, operational, responsibilities

HAS: requirements on requirements elaboration process ...

- goal-oriented: to ensure that operational requirements meet safety, security, survivability objectives
- incremental: for early analysis of partial models
- systematic: for analyst guidance & req completeness
- based on multiple models: for capturing multiple facets
  intentional, structural, operational, responsibilities
- formal when needed, but lightweight
- flexible, opportunistic Þ top-down, bottom-up

HAS: requirements on requirements elaboration process ...

- goal-oriented: to ensure that operational requirements meet safety, security, survivability objectives
- incremental: for early analysis of partial models
- systematic: for analyst guidance & req completeness
- based on multiple models: for capturing multiple facets
  intentional, structural, operational, responsibilities
- formal when needed, but lightweight
- flexible, opportunistic Þ top-down, bottom-up
- Open to seamless transition to architecture

A few definitions...

- Goal: prescriptive statement of intent
  (functional, non-functional)
- Domain prop: descriptive statement about domain

A few definitions...

- Goal: prescriptive statement of intent
  (functional, non-functional)
- Domain prop: descriptive statement about domain
- Agent: active component, controls behaviors
  software-to-be, existing software, device, human

Goal achievement requires agent cooperation

The more fine-grained a goal is, the less agents are required

A few definitions...

- Goal: prescriptive statement of intent
  (functional, non-functional)
- Domain prop: descriptive statement about domain
- Agent: active component, controls behaviors
  software-to-be, existing software, device, human

Goal achievement requires agent cooperation

The more fine-grained a goal is, the less agents are required

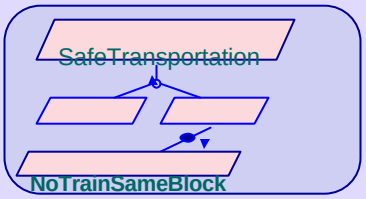
- Requirement: goal assigned to software agent
- Expectation: goal assigned to environment agent

Outline

- Introduction
  - Problems & challenges with HAS
  - Problems & challenges with RE for HAS
- A goal-oriented RE method in action
  - Goal refinement & abstraction
  - Analysis of obstacles & conflicts
  - Goal operationalization
- Goal-based reasoning for higher assurance
- Conclusion

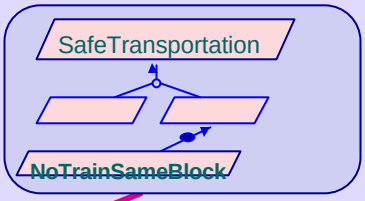
The KAOS goal-oriented RE method

1. Domain analysis: refine/abstract goals

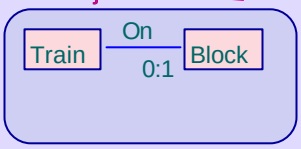


The KAOS goal-oriented RE method

1. Domain analysis: refine/abstract goals

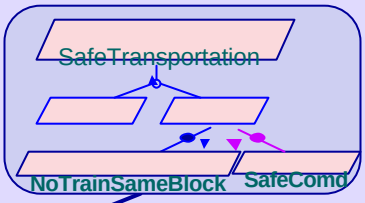


2. Domain analysis: derive/structure objects



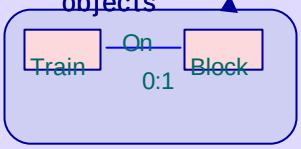
The KAOS goal-oriented RE method

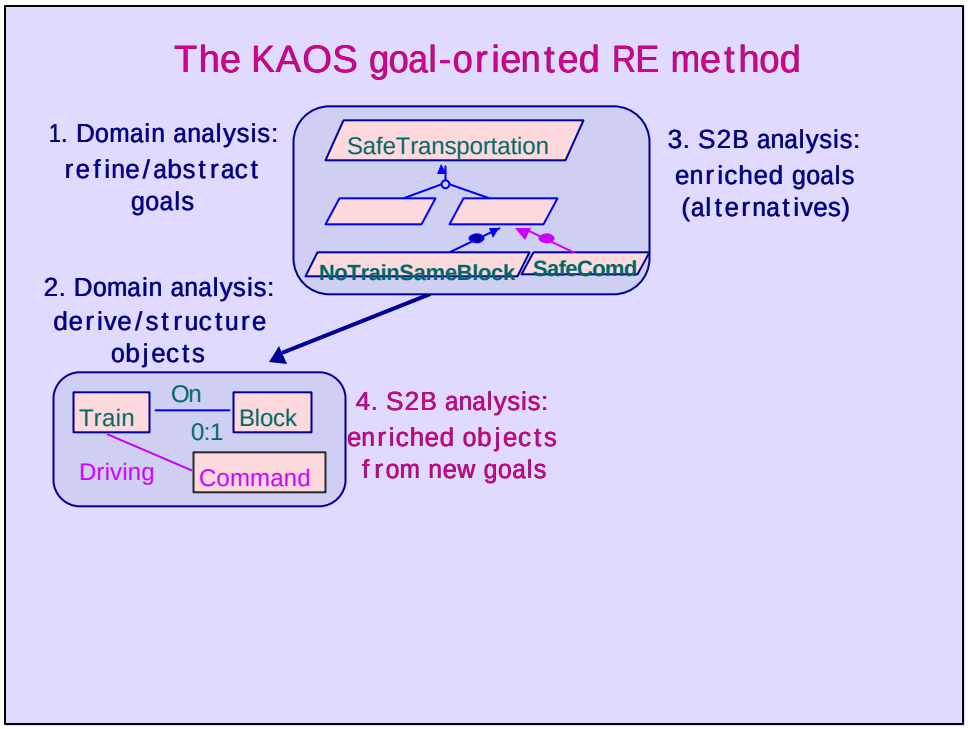
1. Domain analysis: refine/abstract goals

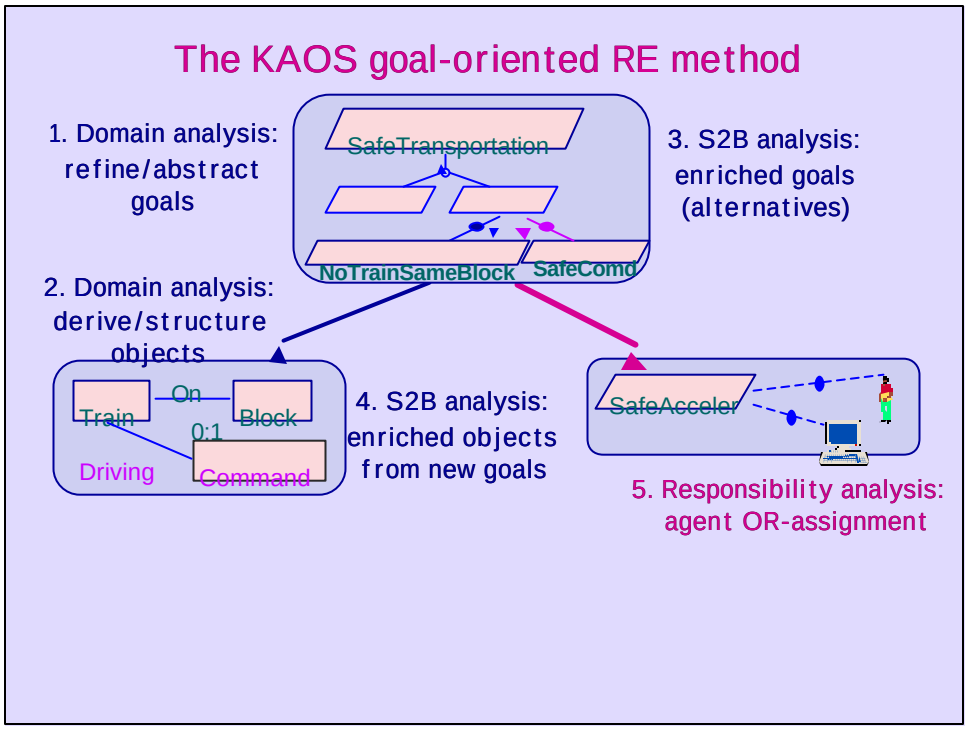


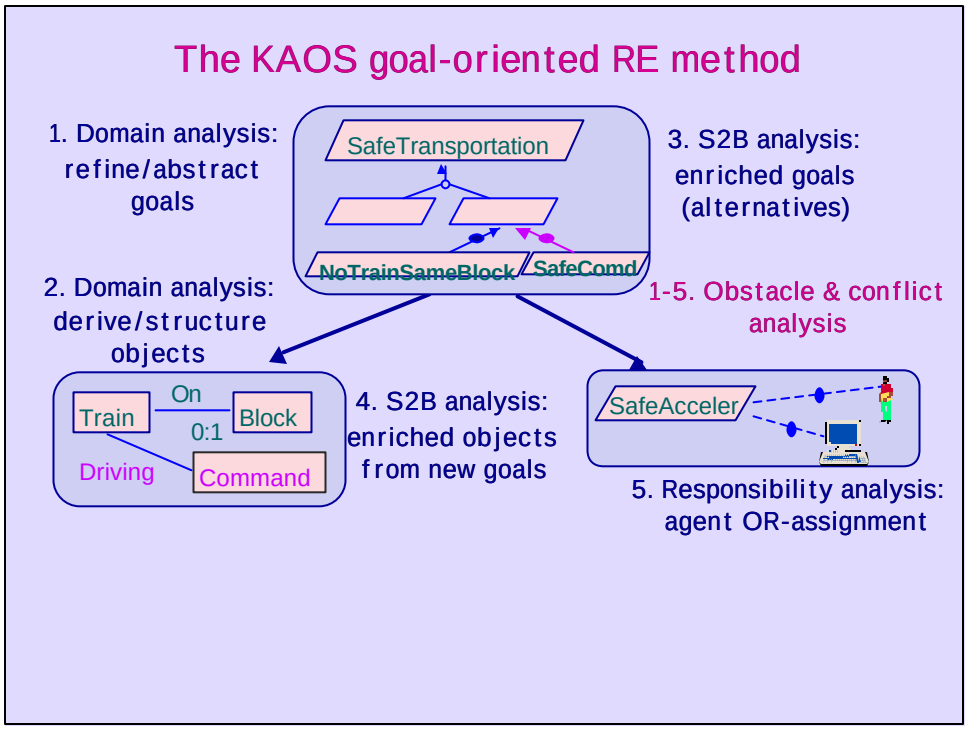
3. S2B analysis: enriched goals (alternatives)

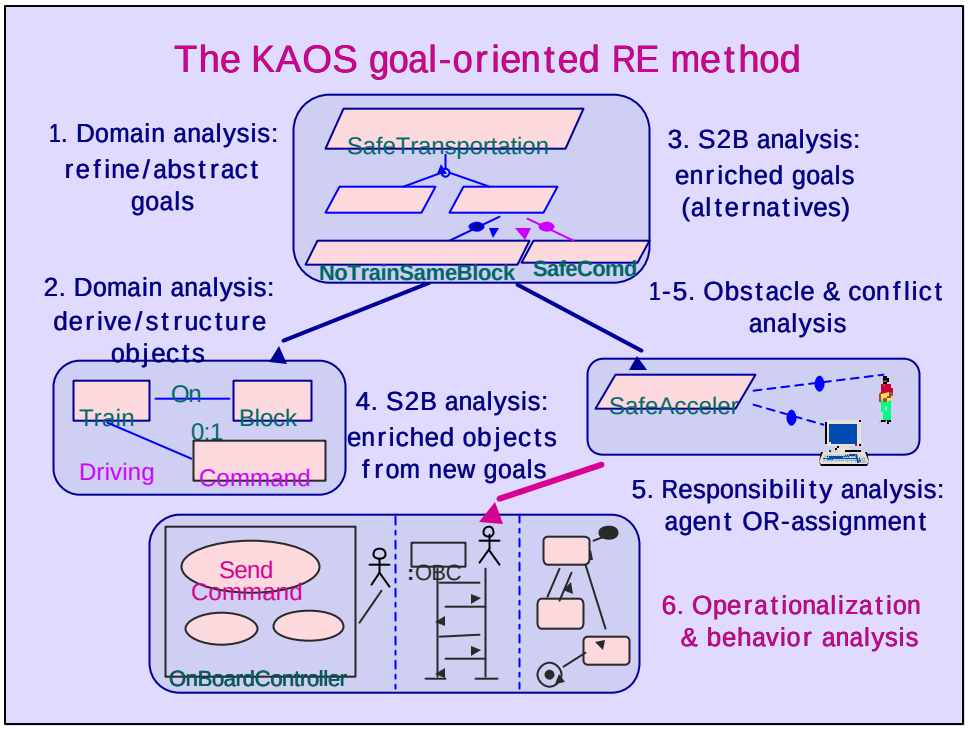
2. Domain analysis: derive/structure objects



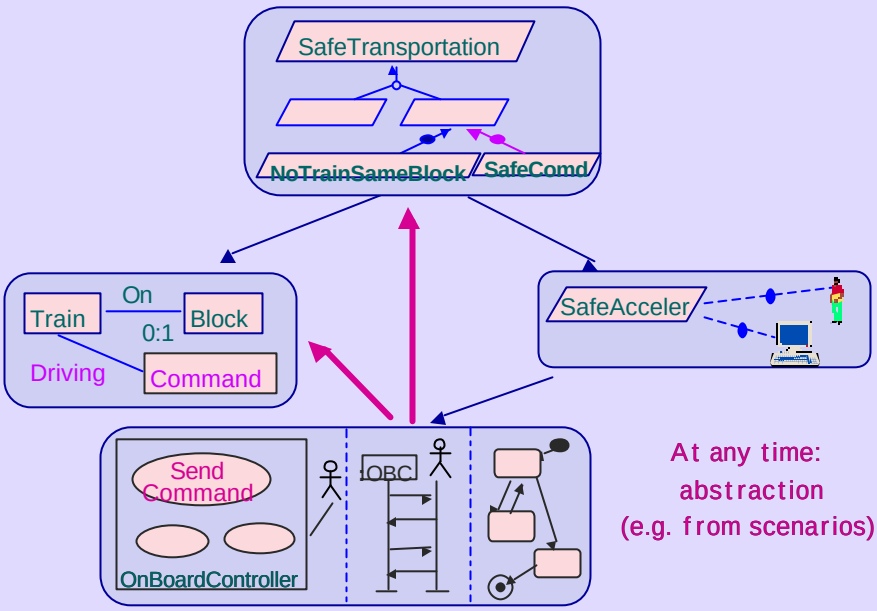

The KAOS goal-oriented RE method
1. Domain analysis: refine/abstract goals
SafeTransportation
NoTrainSameBlock
SafeComd
3. S2B analysis: enriched goals (alternatives)
2. Domain analysis: derive/structure objects
Train
On
Block
0:1
Driving
Command
4. S2B analysis: enriched objects from new goals


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
Train
On
Block
0:1
Driving
Command
4. S2B analysis: enriched objects from new goals
SafeAcceler
5. Responsibility analysis: agent OR-assignment


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


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
:OBC
OnBoardController
6. Operationalization & behavior analysis

The KAOS goal-oriented RE method (2)

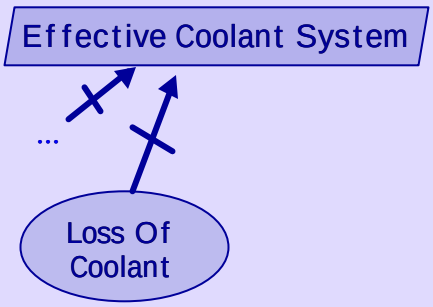


Goal identification from initial material

Effective Coolant System

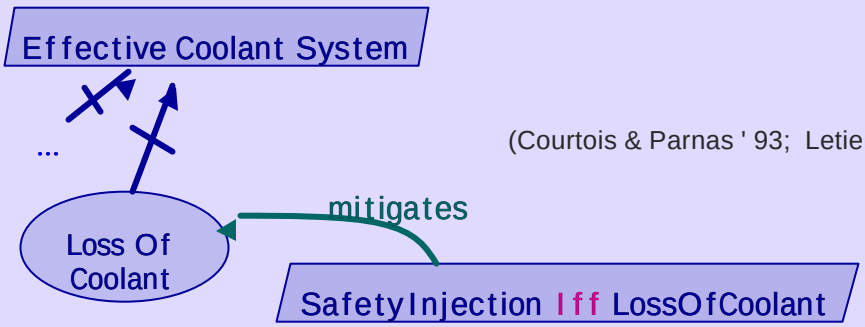
(Courtois & Parnas ' 93; Letier'02)

Goal identification from initial material

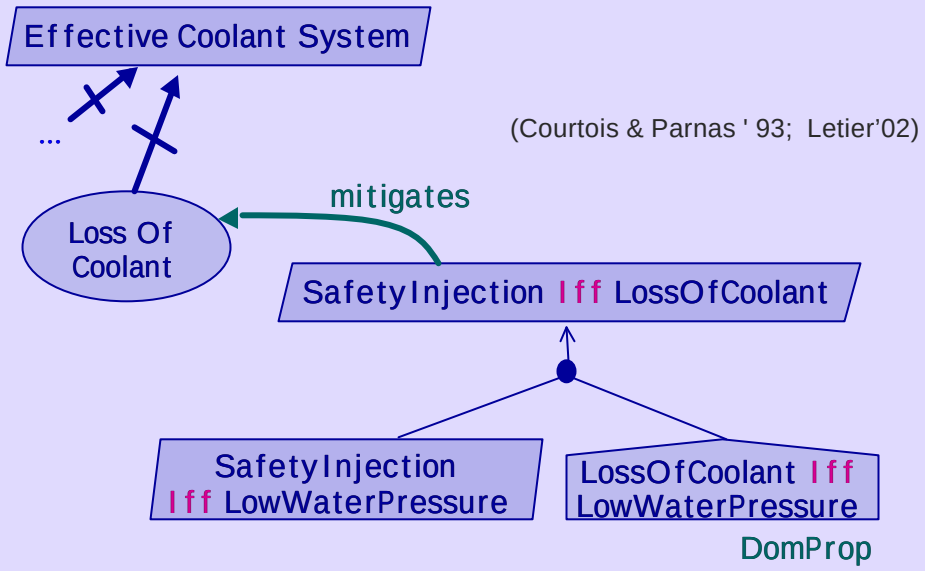


(Courtois & Parnas ' 93; Letier'02)

Goal identification from initial material



Goal identification from initial material



Formalizing goals & deriving objects

Goal *Maintain* [SafetyInjectionIffLowWaterPressure]

Def *The safety injection signal should be 'On' when and only when the water pressure is below the low set point*

FormalSpec

SafetyInjectionSignal = 'On' Û WaterPressure < ' Low '

Formalizing goals & deriving objects

Goal *Maintain* [SafetyInjection Iff LowWaterPressure]

Def *The safety injection signal should be 'On' when and only when the water pressure is below the low set point*

FormalSpec

SafetyInjectionSignal = 'On' Û WaterPressure < ' Low '

⇓

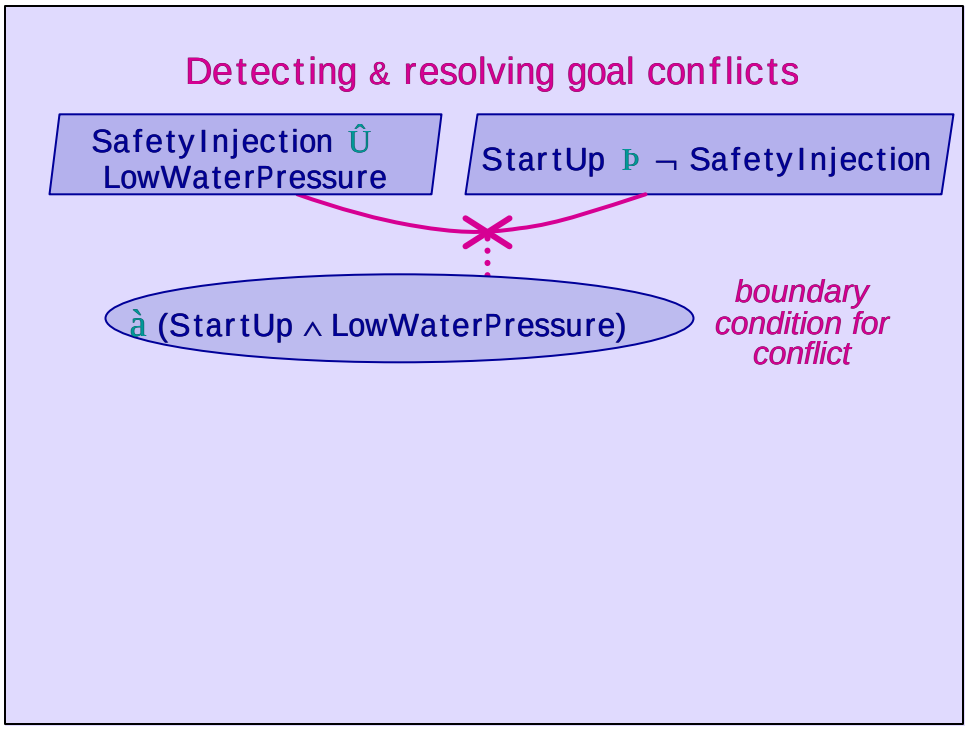
Coolant System	ESFAS
WaterPressure: *PressureUnit*	SafetyInjectionSignal: *{On, Off}*

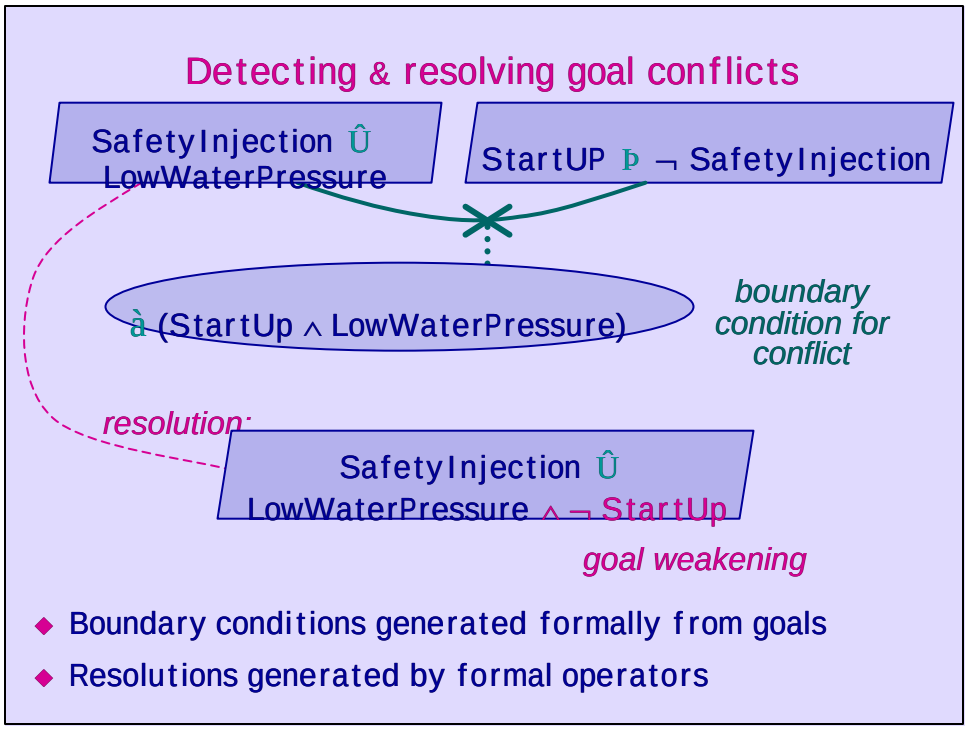
Goals provide a precise criterion for complete, pertinent object model

Detecting & resolving goal conflicts

SafetyInjection Û LowWaterPressure

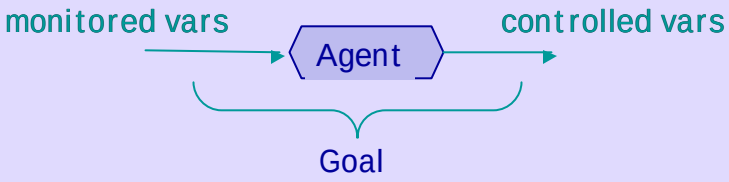
StartUP Þ ¬ SafetyInjection


Detecting & resolving goal conflicts
SafetyInjection Û LowWaterPressure
StartUp Þ ¬ SafetyInjection
à (StartUp ∧ LowWaterPressure)
boundary condition for conflict


Detecting & resolving goal conflicts
SafetyInjection Û LowWaterPressure
StartUP Þ ¬ SafetyInjection
à (StartUp ∧ LowWaterPressure)
boundary condition for conflict
resolution:
SafetyInjection Û LowWaterPressure ∧ ¬ StartUp
goal weakening
Boundary conditions generated formally from goals
Resolutions generated by formal operators

Generating goal refinements & assignments

- Goals need to be refined until assignable to single agents
- A goal is realizable by agent if amounts to relation on variables that are monitorable & controllable by the agent



Generating goal refinements & assignments (2)

- Goal may be unrealizable by agent because...
  - unmonitorable variable
  - uncontrollable variable
  - unachievable monitoring/control of future, ...

  (complete taxonomy of unrealizability problems)

  ß

- Agent-based tactics generate alternative refinements & assignments to resolve unrealizability

(Letier'02)

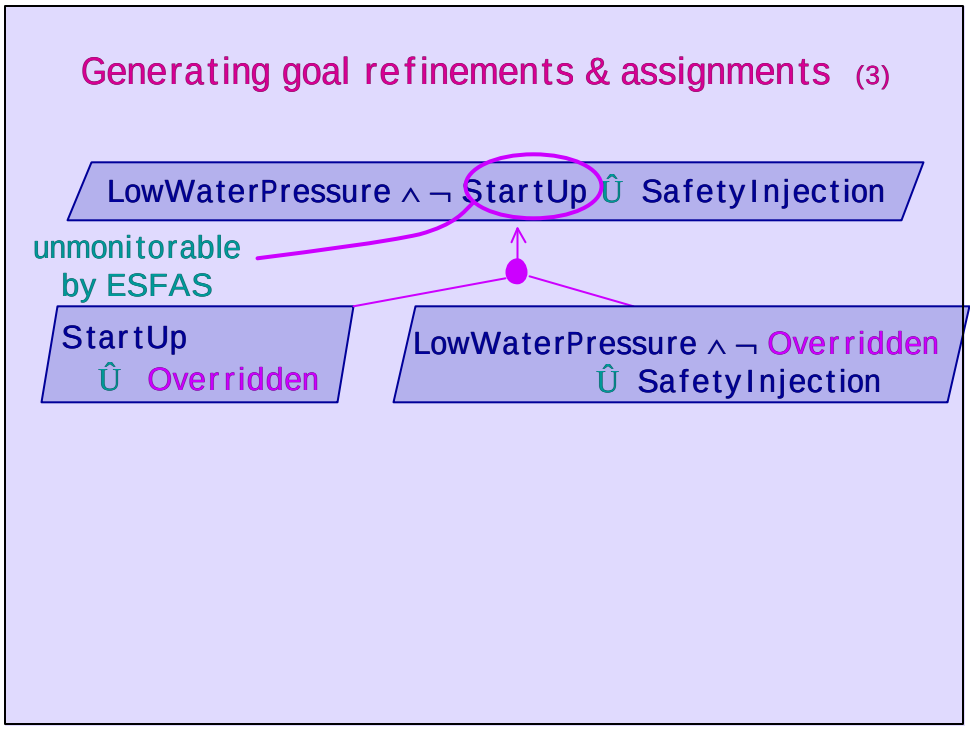
Generating goal refinements & assignments (3)

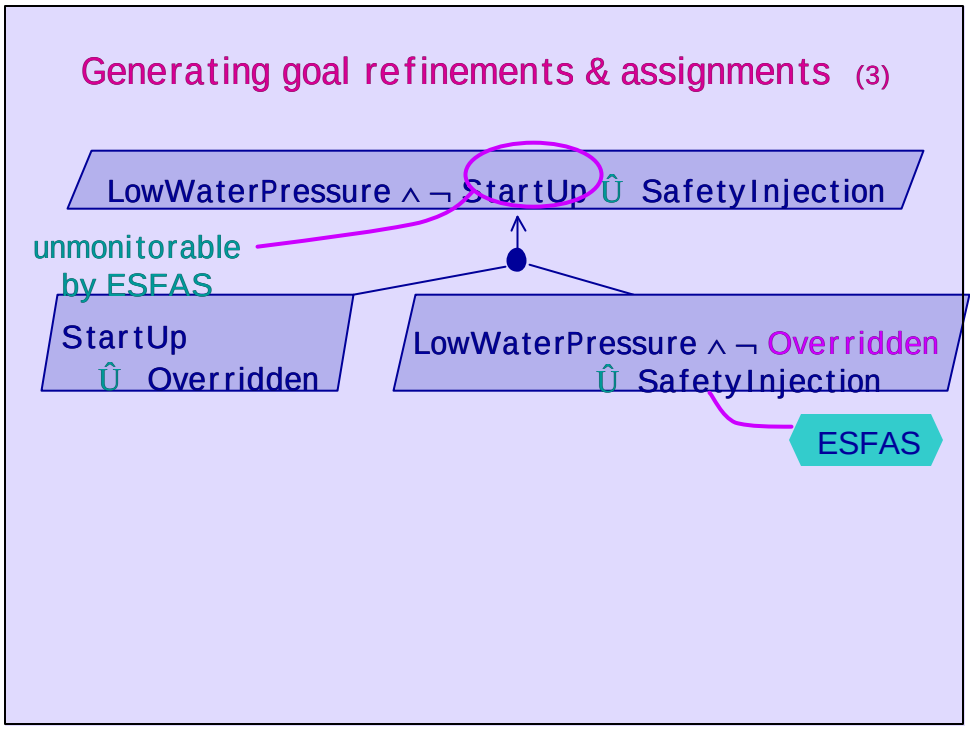
LowWaterPressure ∧ ¬ StartUp Û SafetyInjection

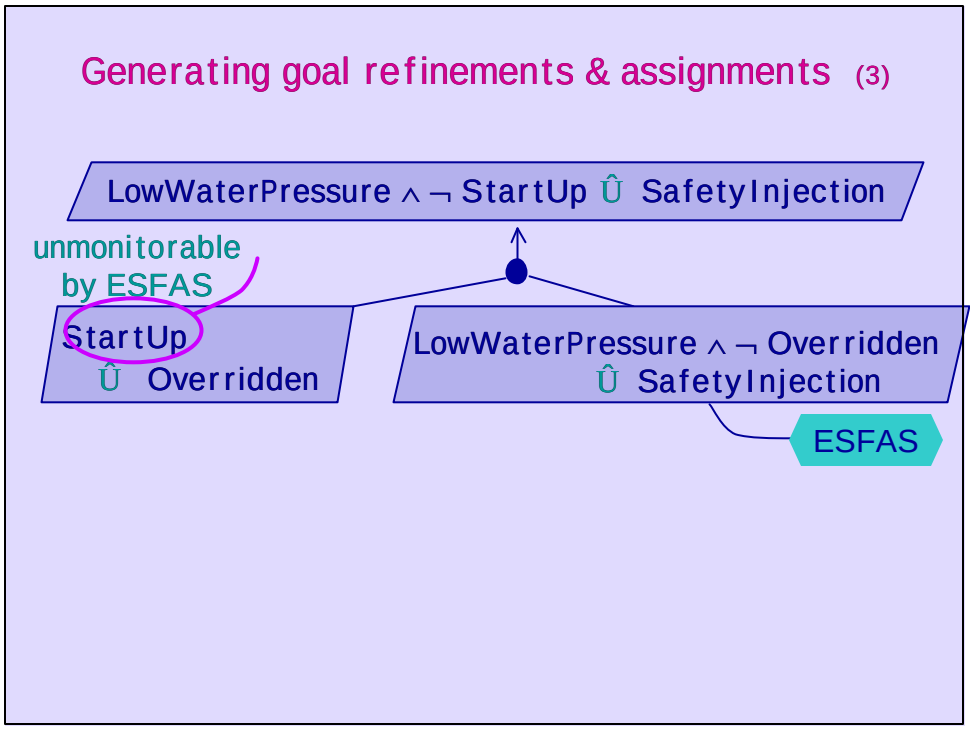
Generating goal refinements & assignments (3)

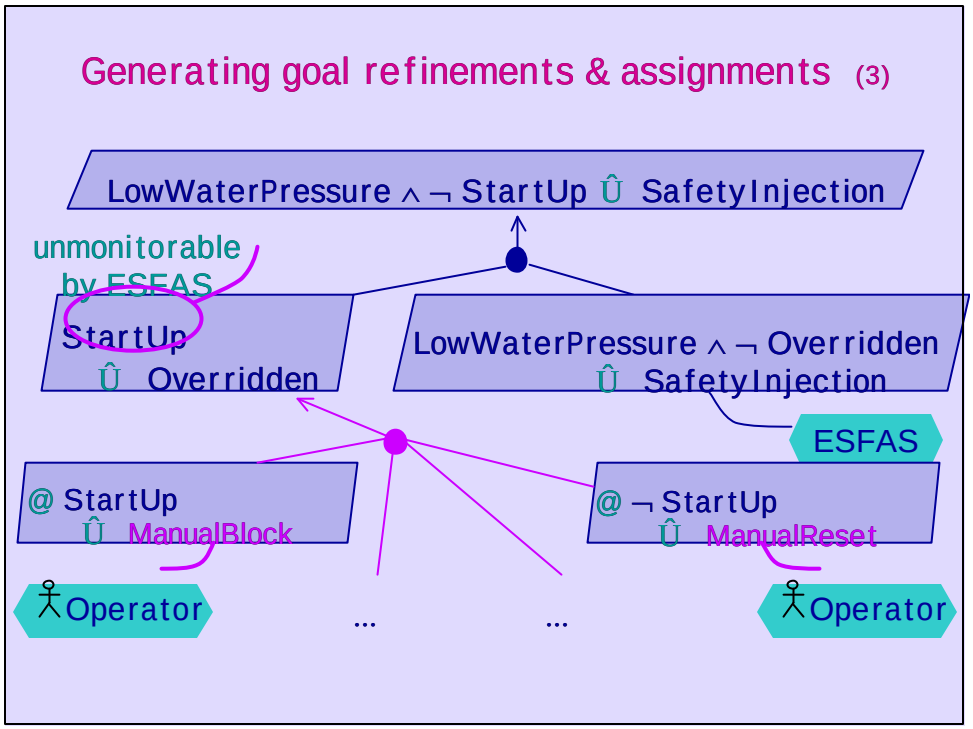
LowWaterPressure ∧ ¬ StartUp Û SafetyInjection

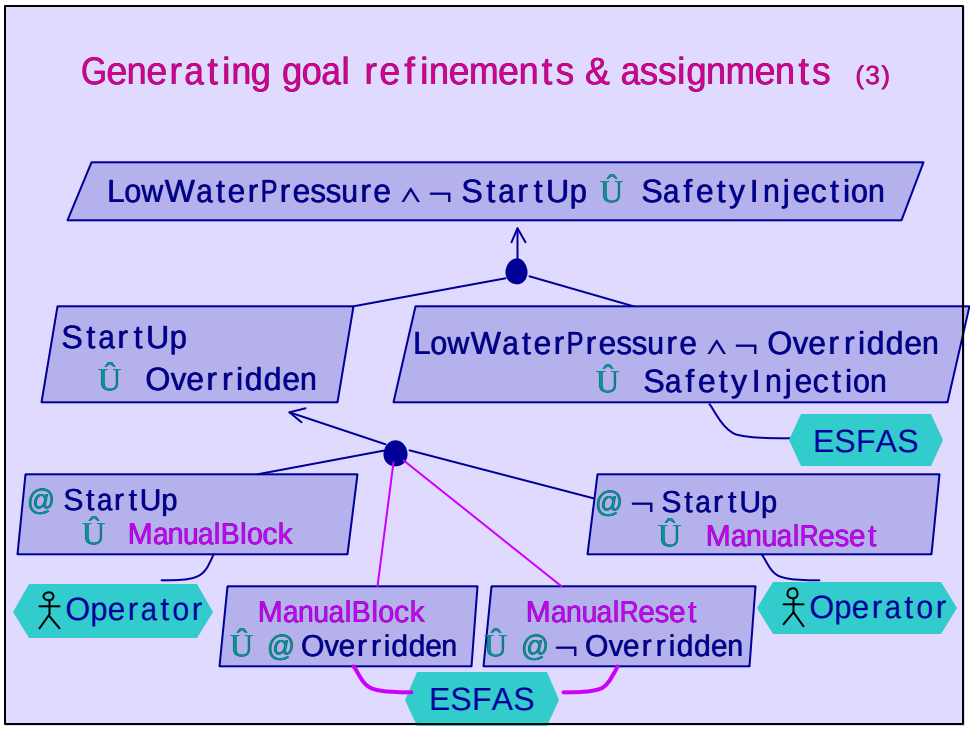
unmonitorable by ESFAS

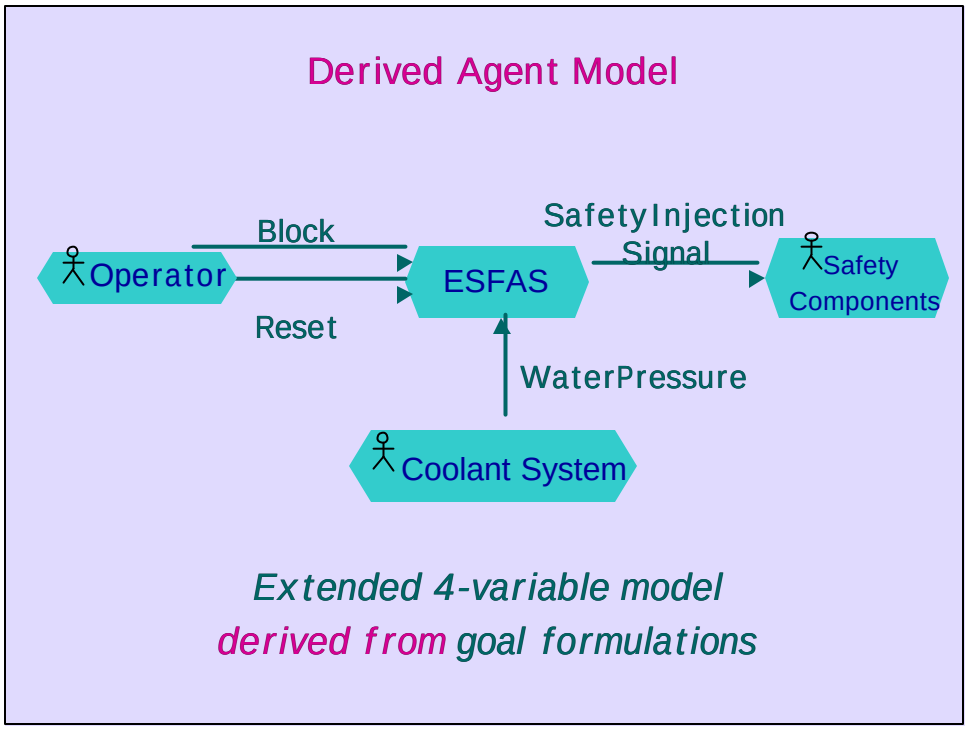

Generating goal refinements & assignments (3)
LowWaterPressure ∧ ¬ StartUp Û SafetyInjection
unmonitorable
by ESFAS
StartUp
Û Overridden
LowWaterPressure ∧ ¬ Overridden
Û SafetyInjection

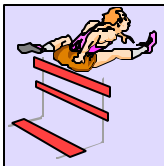

Generating goal refinements & assignments (3)
LowWaterPressure ∧ ¬ StartUp Û SafetyInjection
unmonitorable
by ESFAS
StartUp
Û Overridden
LowWaterPressure ∧ ¬ Overridden
Û SafetyInjection
ESFAS


Generating goal refinements & assignments (3)
LowWaterPressure ∧ ¬ StartUp Û SafetyInjection
unmonitorable by ESFAS
StartUp Û Overridden
LowWaterPressure ∧ ¬ Overridden Û SafetyInjection
ESFAS


Generating goal refinements & assignments (3)
LowWaterPressure ∧ ¬ StartUp Û SafetyInjection
unmonitorable by ESFAS
StartUp Û Overridden
LowWaterPressure ∧ ¬ Overridden Û SafetyInjection
ESFAS
@ StartUp Û ManualBlock
@ ¬ StartUp Û ManualReset
Operator
...
...
Operator


Generating goal refinements & assignments (3)
LowWaterPressure ∧ ¬ StartUp Û SafetyInjection
StartUp Û Overridden
LowWaterPressure ∧ ¬ Overridden Û SafetyInjection
ESFAS
@ StartUp Û ManualBlock
@ ¬ StartUp Û ManualReset
Operator
ManualBlock Û @ Overridden
ManualReset Û @ ¬ Overridden
Operator
ESFAS


Derived Agent Model
Block
SafetyInjection Signal
Operator
ESFAS
Safety Components
Reset
WaterPressure
Coolant System
Extended 4-variable model
derived from goal formulations



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

  $\{O, Dom\} \models \neg G$ obstruction

  $Dom \not\models \neg O$ domain consistency

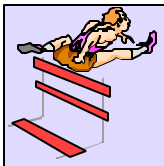
  = anti-goal

  high-level exception, in S2B *or* in environment
- Examples

  for safetyGoals: obstacles = hazards

  for securityGoals: obstacles = threats, attacks

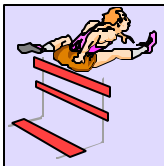
  ...



Obstacle analysis (2)

- Anticipate obstacles ...
  - Þ new, deidealized goals
  - Þ more complete, realistic requirements
  - Þ more robust system

(Potts 1995; van Lamsweerde 1998, 2000)

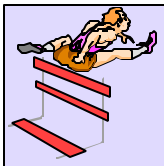


Obstacle analysis (3)

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

  = goal-anchored ...

  hazard analysis for safetyGoals
  threat analysis for securityGoals
  ...

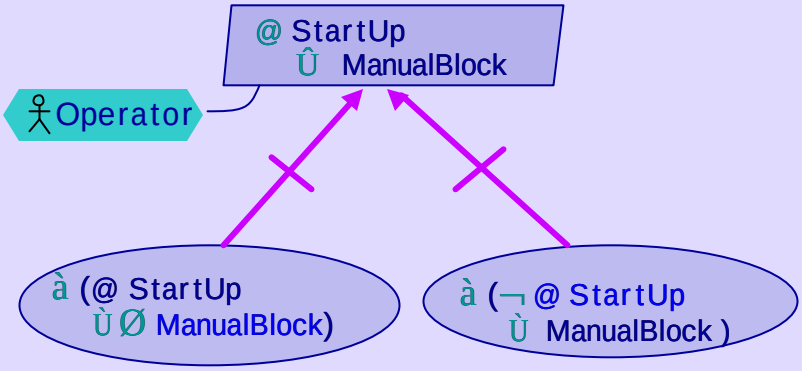


Obstacle analysis (4)

- 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, likely & observable
- If "formal button" pressed:

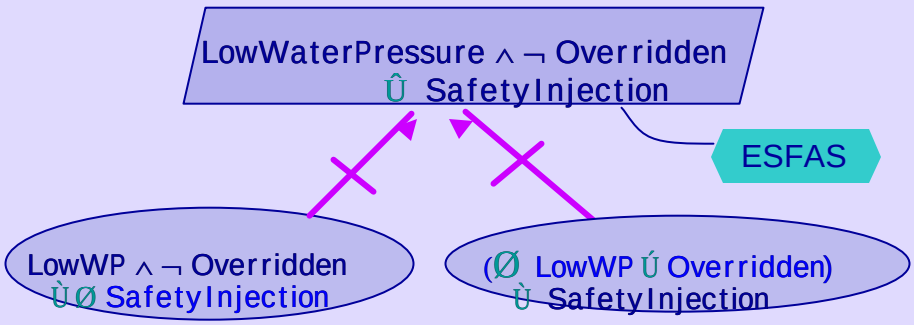
  domain-complete set of obstacles can be generated

Generating Obstacles

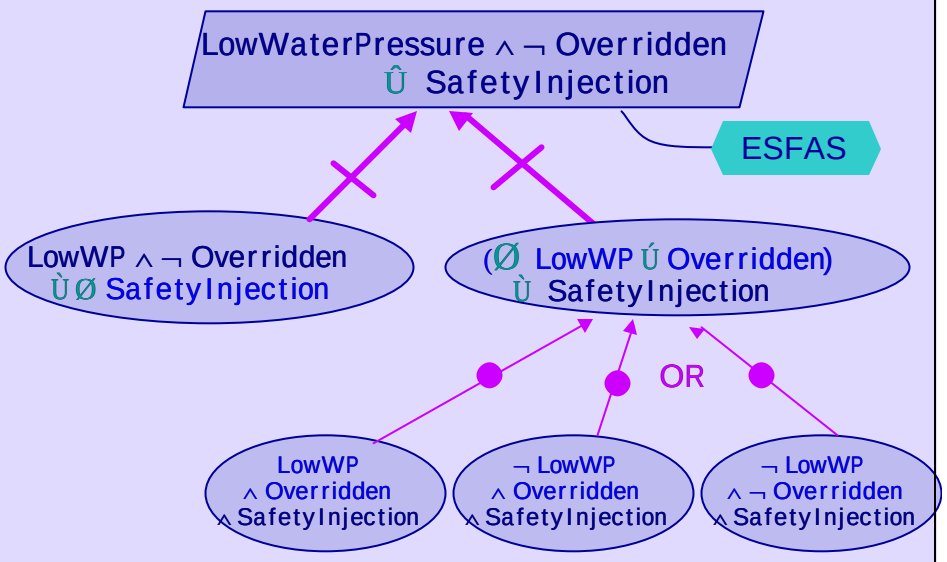


Goal-anchored form of formal fault tree

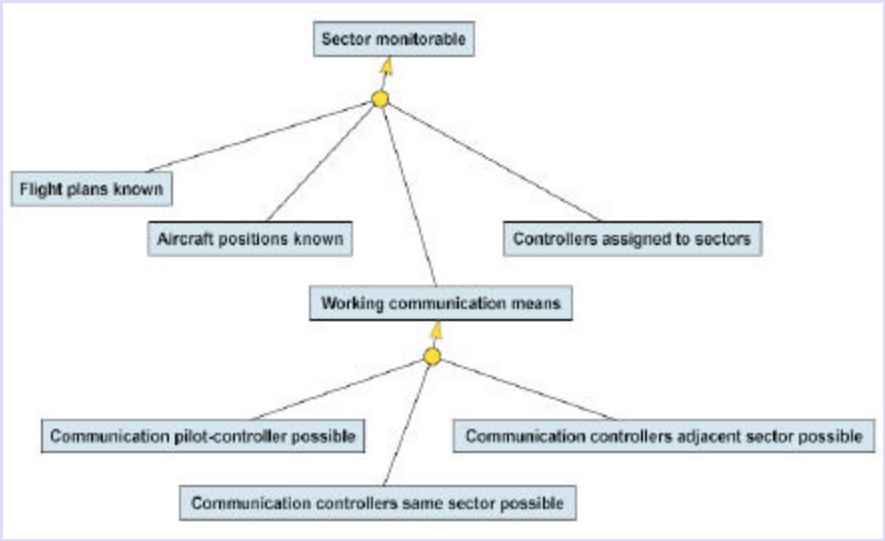
Generating Obstacles (2)



Generating Obstacles (2)

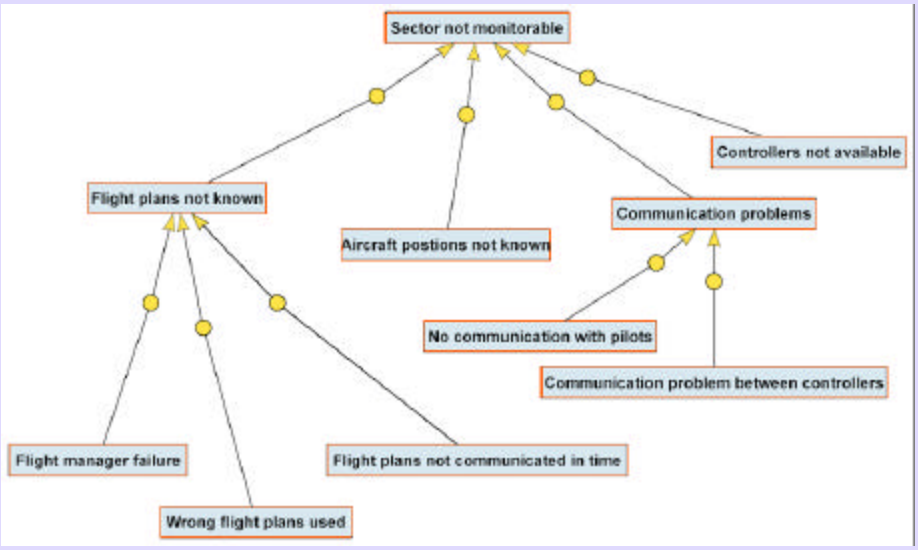


Obstacle analysis : a real (but sad) example



ATC project @ CEDITI, completed March 2001

Obstacle analysis : a real (but sad) example



ATC project @ CEDITI, completed March 2001

Uberlingen mid-air collision, July 2001

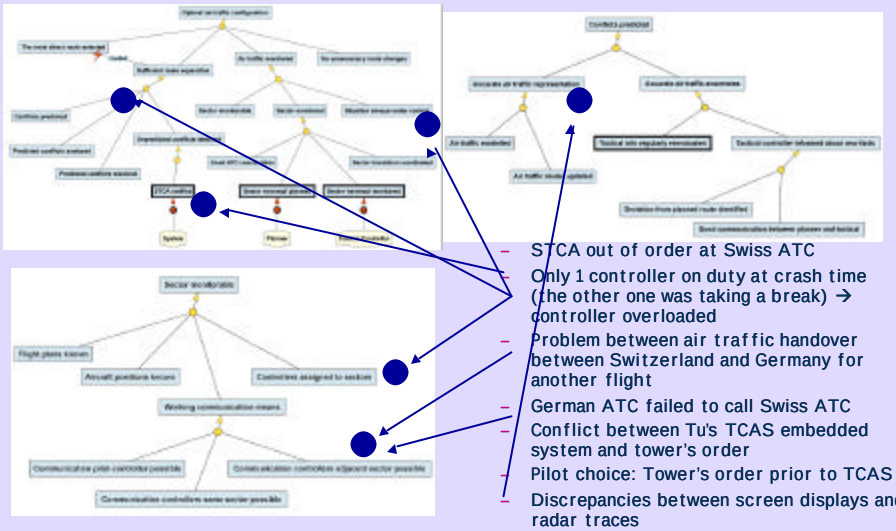
Facts

- July 1st 2002, southern Germany
- DHL Boeing 757 x Russian Tu-154
- 71 people killed, incl. 52 children

Preliminary analysis shows:

- STCA out of order at Swiss ATC
- Only 1 controller on duty at crash time (the other one was taking a break) → controller overloaded
- Problem between air traffic handover between Switzerland and Germany for another flight landing
- German ATC failed to call Swiss ATC
- Conflict between Tu's TCAS embedded system and tower's order
- Pilot choice: Tower's order prior to TCAS
- Discrepancies between screen displays and radar traces

Obstacle analysis : a real (but sad) example



- STCA out of order at Swiss ATC
- Only 1 controller on duty at crash time (the other one was taking a break) → controller overloaded
- Problem between air traffic handover between Switzerland and Germany for another flight
- German ATC failed to call Swiss ATC
- Conflict between Tu's TCAS embedded system and tower's order
- Pilot choice: Tower's order prior to TCAS
- Discrepancies between screen displays and radar traces

Resolving Obstacles

- Assess likelihood & criticality of obstacles
- Generate alternative resolutions

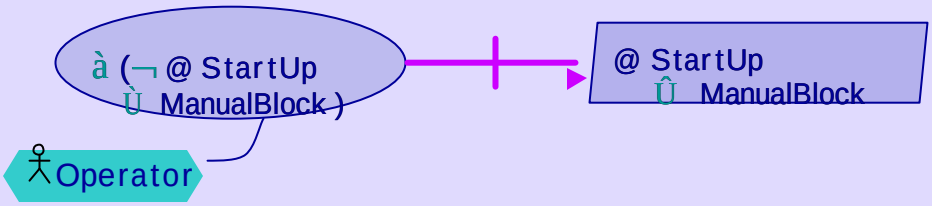
  (based on resolution operators/tactics)
- Evaluate "best" resolutions & select one

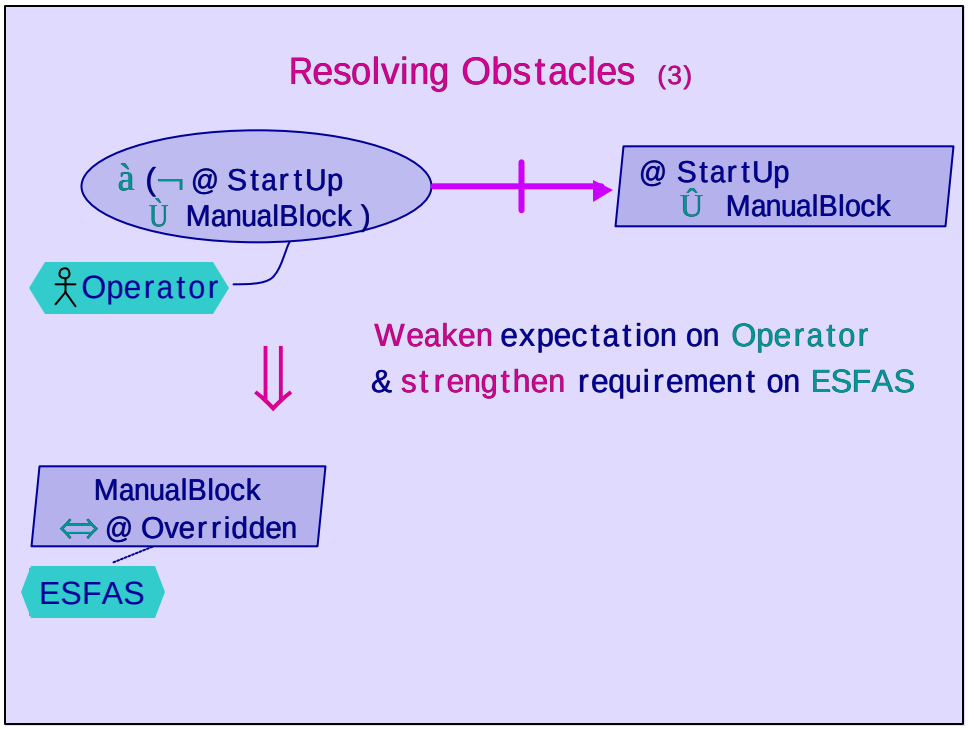
  (based on non-functional/quality goals)

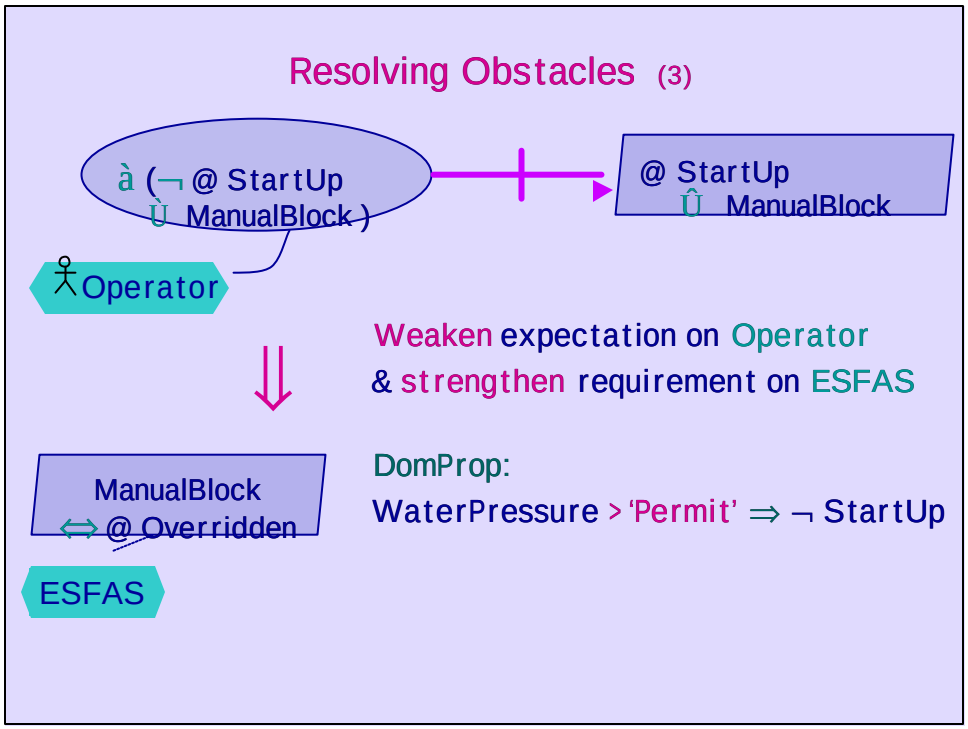
Resolving Obstacles (2)

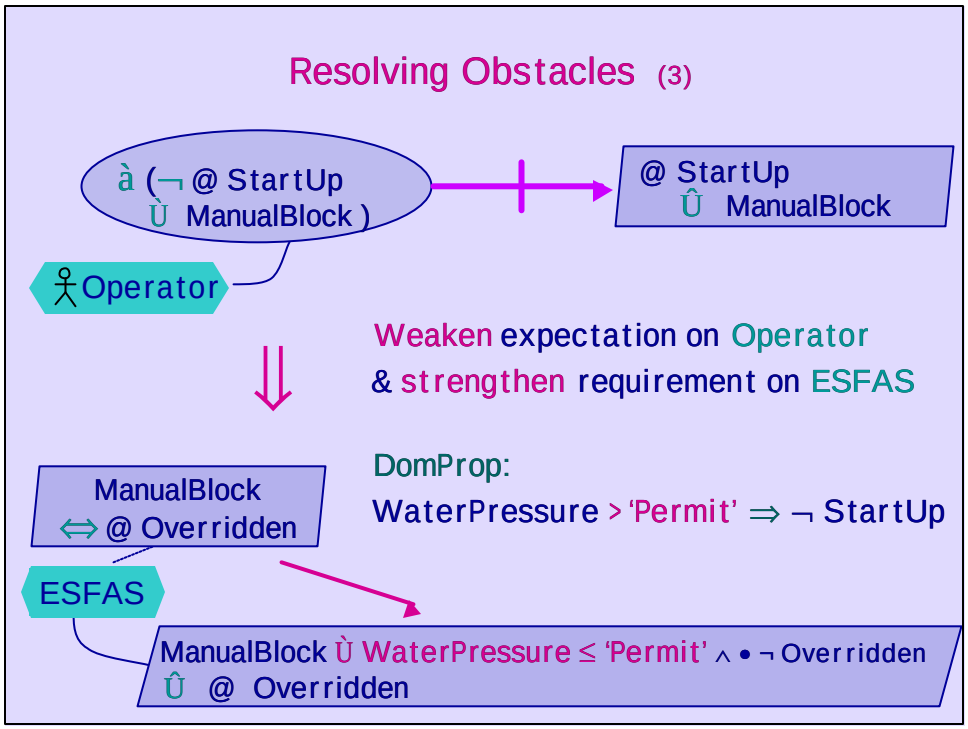
- Resolution operators/tactics
  - eliminate obstacle
    goal substitution, agent substitution,
    goal weakening, obstacle prevention, ...
  - reduce obstacle
  - tolerate obstacle
    goal restoration, obstacle mitigation, ...
- may be applied...
  - at specification time ⇒ spec transformation
  - at run-time ⇒ obstacle monitoring

Resolving Obstacles (3)




Resolving Obstacles (3)
à (¬ @ StartUp Ù ManualBlock)
@ StartUp Û ManualBlock
Operator
Weaken expectation on Operator
& strengthen requirement on ESFAS
ManualBlock ⇔ @ Overridden
ESFAS


Resolving Obstacles (3)
à (¬ @ StartUp Ù ManualBlock)
@ StartUp Û ManualBlock
Operator
Weaken expectation on Operator
& strengthen requirement on ESFAS
DomProp:
WaterPressure > 'Permit' ⇒ ¬ StartUp
ManualBlock ⇔ @ Overridden
ESFAS


Resolving Obstacles (3)
à (¬ @ StartUp Ù ManualBlock)
@ StartUp Û ManualBlock
Operator
Weaken expectation on Operator
& strengthen requirement on ESFAS
DomProp:
WaterPressure > 'Permit' ⇒ ¬ StartUp
ManualBlock ⇔ @ Overridden
ESFAS
ManualBlock Ù WaterPressure ≤ 'Permit' ∧ • ¬ Overridden Û @ Overridden


Operationalizing goals
Goal Maintain [OverriddenWhenManualBlock AndPressureLessThanPermit]
FormalSpec
ManualBlock ∧ WaterPressure ≤ 'Permit' ∧ • ¬ Overridden Û @ Overridden
formal operationalization patterns
(Letier'02)
Operation OverrideSafetyInjection
DomPre ¬ Overridden
DomPost Overridden
ReqPre/Trig for OverriddenWhenManualBlock AndPressureLessThanPermit:
ManualBlock ∧ WaterPressure ≤ 'Permit'
Goal-oriented elaboration process ends where most specification techniques start

Outline

- Introduction
  - Problems & challenges with HAS
  - Problems & challenges with RE for HAS
- A goal-oriented RE method in action
  - Goal refinement & abstraction
  - Analysis of obstacles & conflicts
  - Goal operationalization
- Goal-based reasoning for higher assurance
- Conclusion

Formal goal-based reasoning for higher assurance

- Early analysis, partial models, intertwined with model construction

Formal goal-based reasoning for higher assurance

- Early analysis, partial models, intertwined with model construction
- Wide range of opportunities:
  - checking/deriving goal refinements
  - checking/deriving operationalizations
  - generating obstacles
  - generating boundary conditions for conflict
  - goal mining from scenarios
  - generating state machines from operationalizations
  - reusing goal-based specs by analogy

Checking goal refinements

- Aim: show that refinement is correct & complete

  R, Ass, Dom |-- G

  R: conjunctive set of requirements or subgoals

Checking goal refinements

- Aim: show that refinement is correct & complete

  R, Ass, Dom |-- G

  R: conjunctive set of requirements or subgoals

- Approach 1: use TL theorem prover

  heavyweight, non-constructive

Checking goal refinements

- Aim: show that refinement is correct & complete

  R, Ass, Dom |-- G

  R: conjunctive set of requirements or subgoals

- Approach 1: use TL theorem prover

  heavyweight, non-constructive

- Approach 2: use formal refinement patterns

  lightweight, constructive:

  - to complete partial refinements
  - to explore alternative refinements

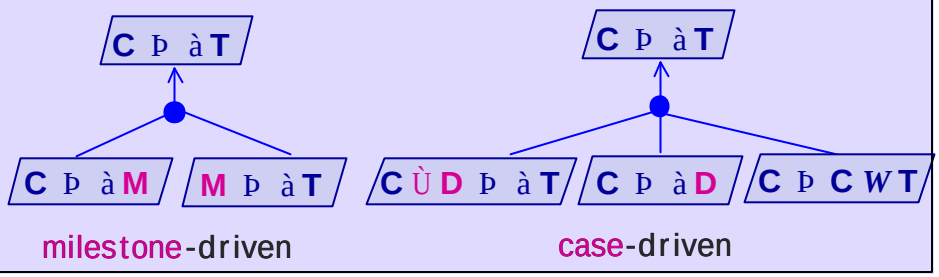
  (Darimont'96)

Checking goal refinements (2)

Idea:

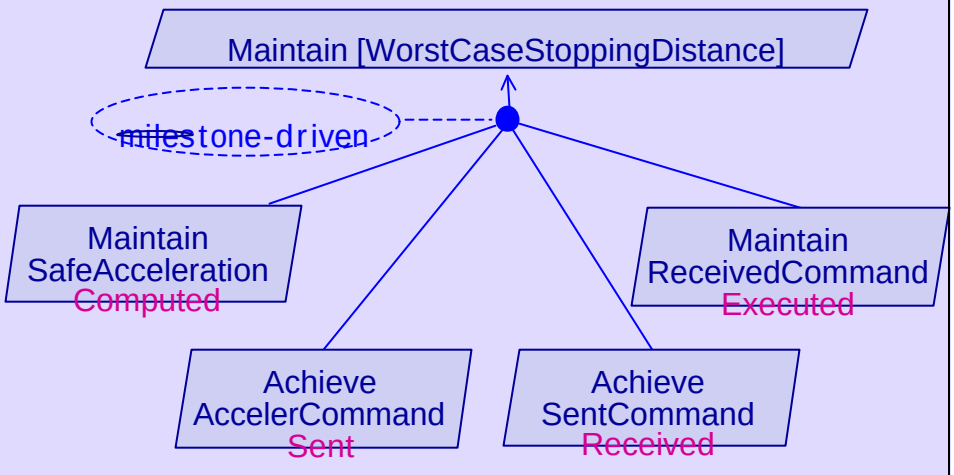
- Buid library of patterns (structured by *tactics*)
- Prove patterns once for all
- Reuse through instantiation, in matching situation

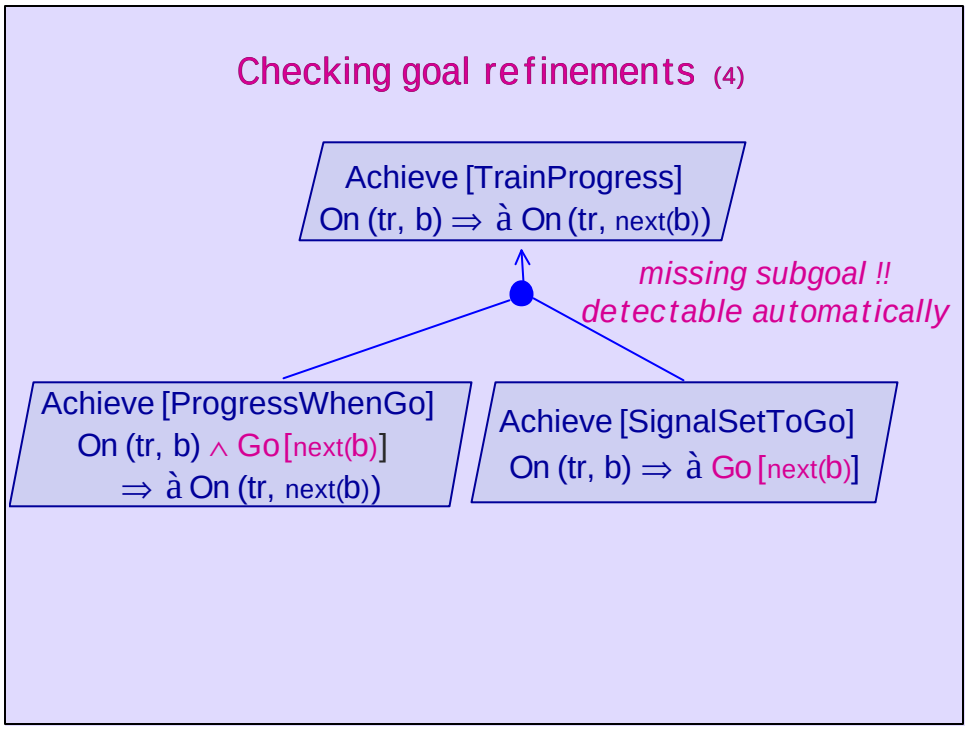
e.g. frequent patterns:

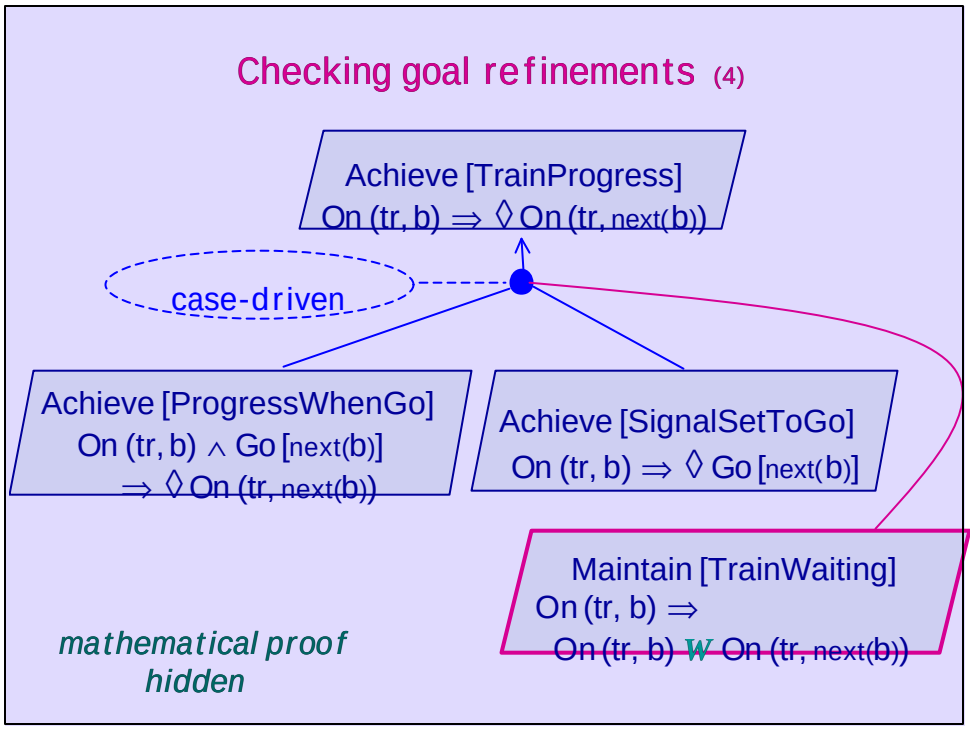


milestone-driven case-driven

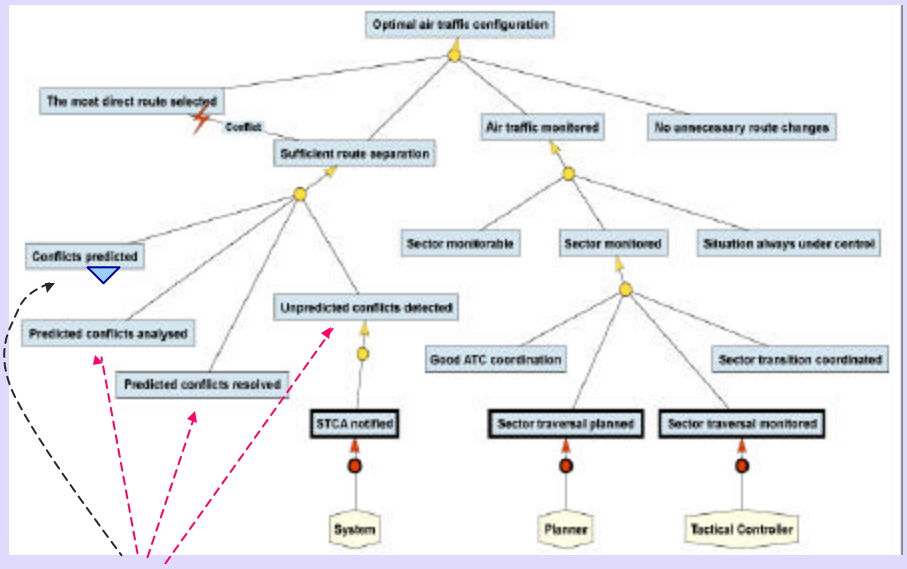
Checking goal refinements (3)




Checking goal refinements (4)
Achieve [TrainProgress]
On (tr, b) ⇒ à On (tr, next(b))
missing subgoal !!
detectable automatically
Achieve [ProgressWhenGo]
On (tr, b) ∧ Go [next(b)]
⇒ à On (tr, next(b))
Achieve [SignalSetToGo]
On (tr, b) ⇒ à Go [next(b)]


Checking goal refinements (4)
Achieve [TrainProgress]
On (tr, b) ⇒ ◊ On (tr, next(b))
case-driven
Achieve [ProgressWhenGo]
On (tr, b) ∧ Go [next(b)]
⇒ ◊ On (tr, next(b))
Achieve [SignalSetToGo]
On (tr, b) ⇒ ◊ Go [next(b)]
Maintain [TrainWaiting]
On (tr, b) ⇒
On (tr, b) W On (tr, next(b))
mathematical proof
hidden

Refinement by case + milestone: a real example



milestone goals

(from project @ CEDITI)

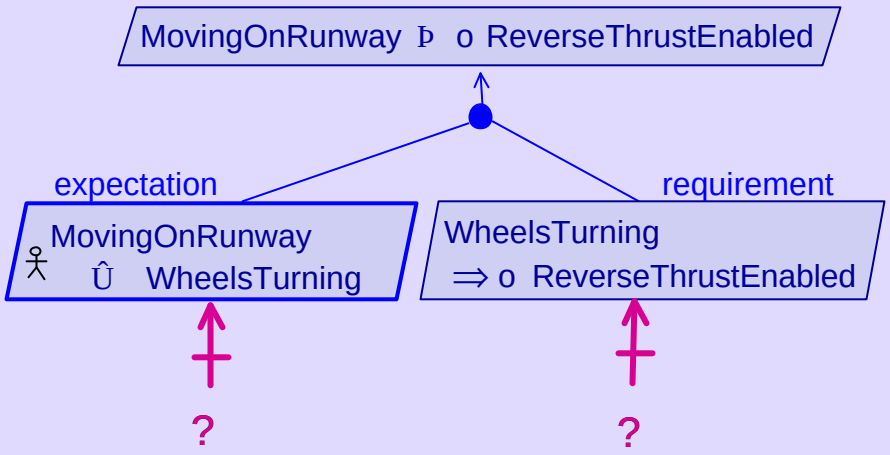
Checking goal refinements (5)

- Approach 3: early bounded model checking
  - checking of goal models
  - partial models
  - incremental checking/debugging
  - on selected object instances *(propositionalization)*
  - ouput:
    - OK
    - KO + counter-example scenario

Roundtrip use of SAT solver, NuSMV, theorem prover

Time for demo...

Generating obstacles



Generating obstacles (2)

- Deriving precondition for obstruction

  MovingOnRunway Þ WheelsTurning

Generating obstacles (2)

- Deriving precondition for obstruction

  MovingOnRunway Þ WheelsTurning

  ® goal negation:

  à MovingOnRunway Ù ¬ WheelsTurning

Generating obstacles (2)

- Deriving precondition for obstruction

  MovingOnRunway Þ WheelsTurning

  ® goal negation:

  à MovingOnRunway Ù ¬ WheelsTurning

  ® regress through Dom:

  ? necessary conditions for wheels turning ?

  WheelsTurning Þ ¬ Aquaplaning

  i.e. Aquaplaning Þ ¬ WheelsTurning

Generating obstacles (2)

- Deriving precondition for obstruction

MovingOnRunway Þ WheelsTurning

® goal negation:

à MovingOnRunway Ù ¬ WheelsTurning

® regress through Dom:

? necessary conditions for wheels turning ?

WheelsTurning Þ ¬ Aquaplaning

i.e. Aquaplaning Þ ¬ WheelsTurning

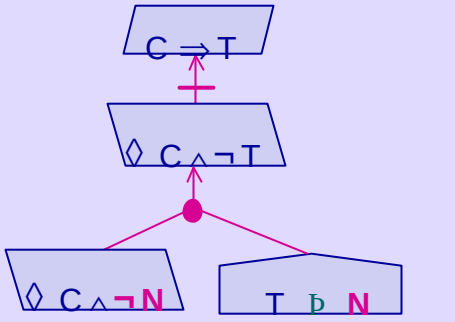
® RHS unifiable:

à MovingOnRunway Ù Aquaplaning *Warsaw obstacle*

Generating obstacles (3)

- Using formal obstruction patterns

in fact we just used a frequent pattern:



Conclusion

- Start thinking about high assurance at RE time
- Be constructive
- Be formal (but lightweight) when needed
- Stay declarative as long as possible

  Goals provide better abstractions for decision makers

Conclusion (2)

- For constructive guarantee of high assurance:
  adopt a systematic elaboration process

  from high-level goals to detailed operational specs

  from detailed operational specs to high-level goals
- Adopt a system engineering perspective

  model software + environment

  (e.g., attacker, attackee)
- Be prepared to explore & evaluate alternatives

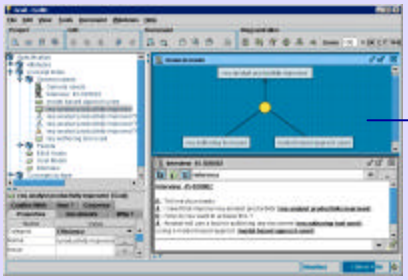
Conclusion (3)

- Build rich models
  - multiple facets: intentional, structural, operational, responsibilities
  - multiple versions: current, to-be, evolutions
- Be pessimistic from beginning
  - ⇒ requirements-level exception handling
- Be prepared to handle conflicts
  - ⇒ conflict detection + resolution tactics

Conclusion (4)

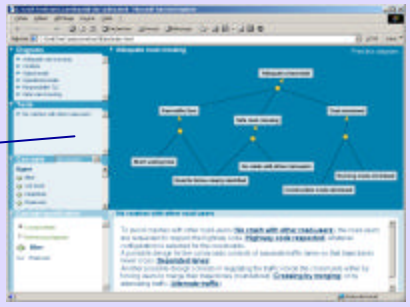
- Benefits of multi-button framework
  - semi-formal:
    for modeling, navigation, traceability
  - formal, à la carte
    for precise, incremental reasoning on model fragments

The GRAIL tool



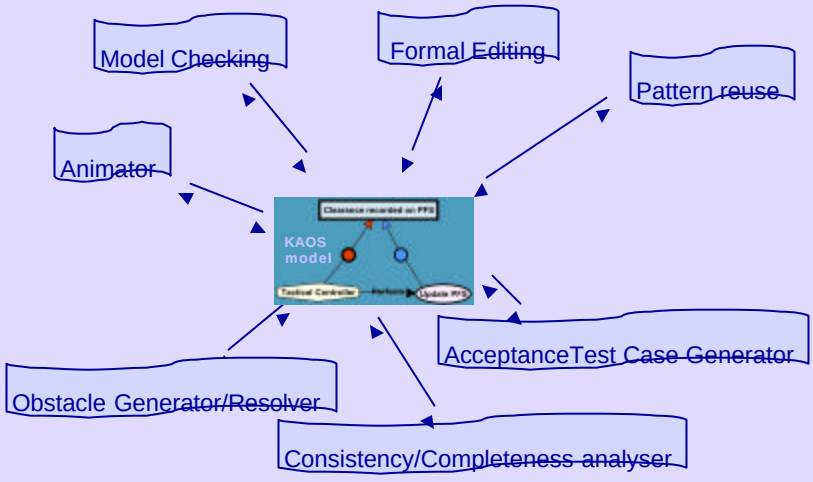
KAOS model editor

Requirements documents generation



model browser

The GRAIL/FAUST analysis environment



Thanks to the KAOS crew

- UCL @LLN: basic research
  - method, techniques

  E. Letier, H. Tran Van, L. Willemet
- CETIC / FAUST: tech transfer center
  - applied research, formal analysis tools

  P. Massonet, J.F. Molderez, C. Ponsard, A. Rifaut
- CEDITI / IGLOO: UCL spin-off
  - tool packaging (semi-formal tools)
  - industrial experience, consulting; *feedback*

  R. Darimont, E. Delor, C. Nève, J.L. Roussel