Lero THE IRISH SOFTWARE RESEARCH CENTRE

Autonomy Requirements for Smart Vehicles

Emil Vassev

April 24, 2017

University of Limerick Ollscoil Luimnigh
Trinity College Dublin
DCU
UCD Dublin
Ollscoil na Gaillimhe NUI Galway
Dundalk Institute of Technology
UCC University College Cork, Ireland
Maynooth University National University of Ireland Maynooth

Outline

- **Autonomy**
- **Autonomous Vehicles and Safety**
- **Autonomy Requirements Engineering (ARE)**
- **KnowLang – The ARE Formal Language**
- **Verification of Autonomy Requirements**
- **eMobility Case Study**
- **Simulation and Testing**
- **Conclusion**

Autonomy?

Capability to *autonomously*:

- ✓ exhibit knowledge (awareness)
- ✓ interact with the operational environment
- ✓ perceive important structural and dynamic aspects of the operational environment

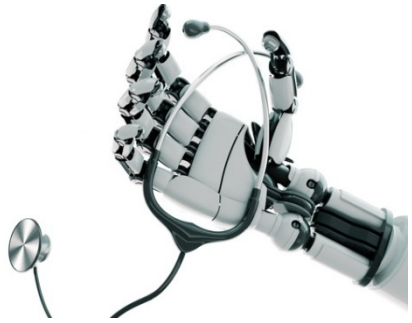
Integration and promotion of autonomy - extremely challenging task:

- ✓ autonomy requirements – elicitation and expression
  - determine what autonomic features are to be developed for a particular self-adaptive system

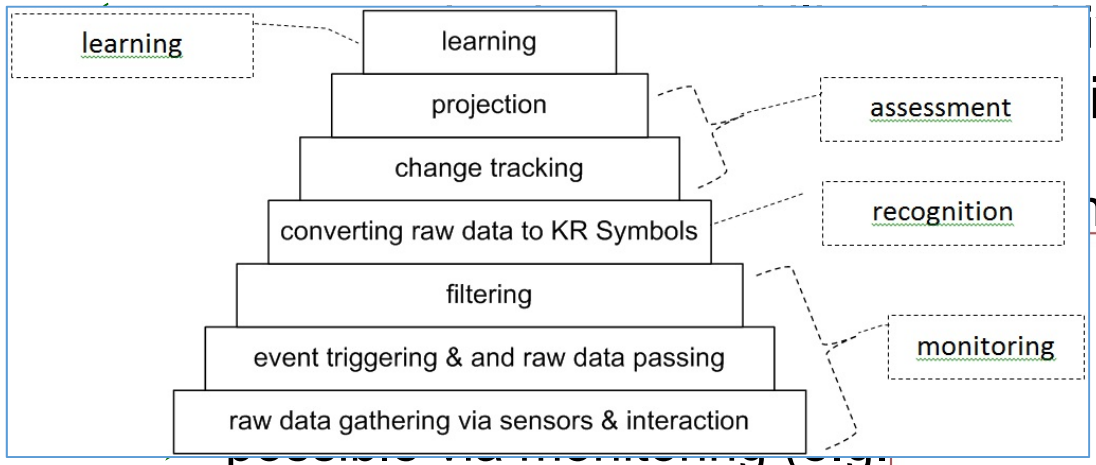
Autonomy vs Automation

- ✓ processes can be executed without human intervention
- ✓ automation:

- ✓ complete autonomy may not be desirable or possible for some systems - adjustable and mixed autonomy:
  - *adjustable autonomy* - the level of autonomy of the system (e.g., spacecraft) can vary depending on the circumstances or the needed interaction and control
  - autonomy can be adjusted to be either *complete*, *partial* or *no autonomy*

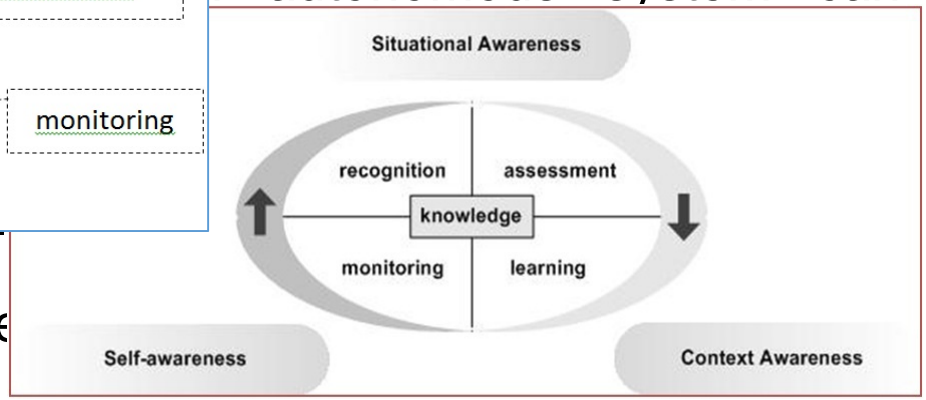


Autonomy: Awareness and Adaptation



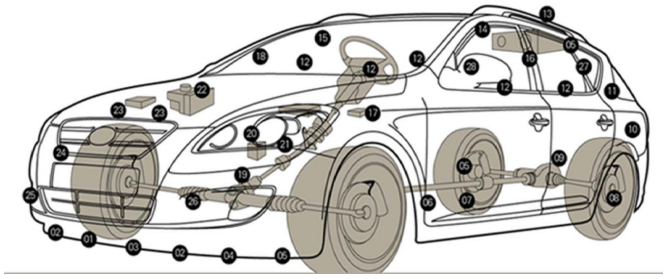
ves the autonomy, i.e., it
itions:

n autonomous system can



- learning – improve with expe

✓ adaptation – product of automation and awareness to cope with changes (self-adaptation – autonomous adaptation)



Autonomous Vehicles and Safety

Transportation assistance:

- ✓ massive investments of money and effort into self-driving cars;
- ✓ autonomous cars share the road with other cars, motorists, bicyclists, and pedestrians;

- ✓ first severe accidents prove that they are not very secure;
- ✓ social, legal and warranty issues – laws, regulations.

Autonomous Vehicles and Safety

Establishing trust in autonomous vehicles (twofold objective):

- ✓ *establish boundaries (range) of adaptation* — certain properties (e.g., safety) should be unavoidably held, and thus unforeseen adaptations that may mitigate such properties, should not be allowed without change (human control) in the established adaptation range;
- ✓ *pursue autonomy in a stepwise manner* — autonomy can be gradually introduced: no autonomy, partial autonomy, controlled autonomy, and full autonomy: in earlier stages, autonomy should be used in less risky domains.

Autonomy and Adaptation Cannot Be 100% Safe

Formal methods shall assist us in the construction of autonomous vehicles with autonomy features:

- ✓ provide means for comprehensive analysis of requirements
- ✓ near-to-complete exploration of system behavior

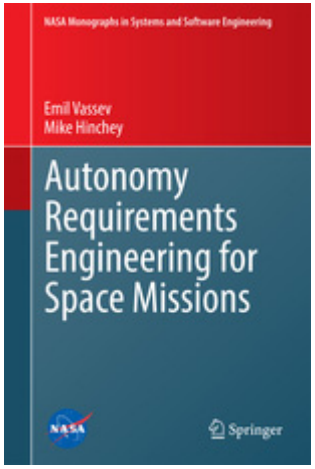
Quantitative measure of safety gained with formal methods:

- ✓ formal verification and validation allow for early detection of safety flaws, i.e., before implementation
- ✓ high quality of safety requirements improves the design and implementation of these requirements
- ✓ formally specified safety requirements assist in the derivation and generation of efficient test cases

ARE – Autonomy Requirements Engineering

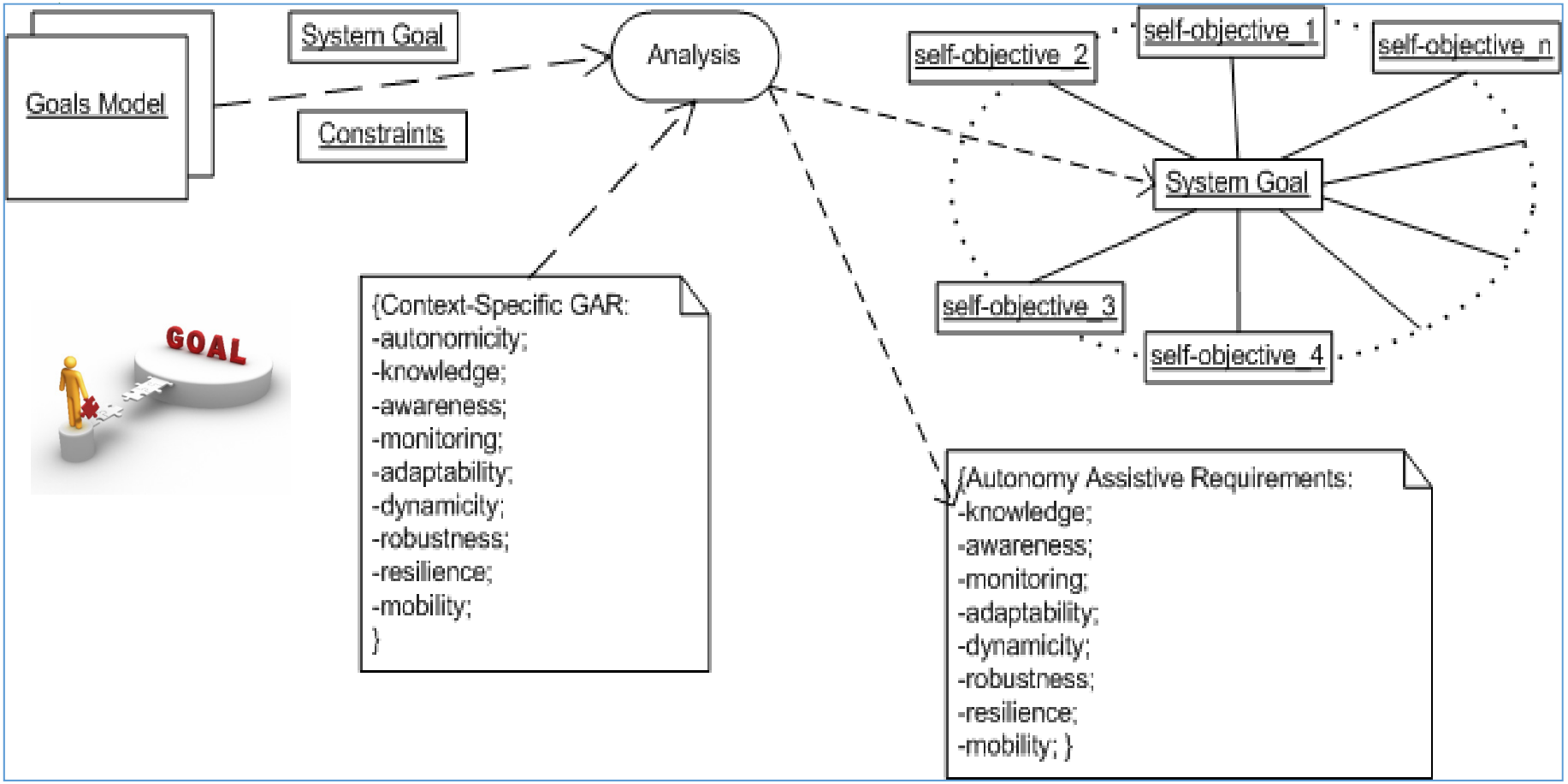
Developed by Lero (joint project with ESA).

Tackles the system's autonomic and self-adaptive nature:



- ✓ able to reason and autonomously pursue goals
- ✓ monitor environment and subsystem
- ✓ eventually modify its behavior and/or structure according to changes in the operational environment or goals

ARE – Autonomy Requirements Engineering



GORE + GAR = self-* objectives + autonomy assistive requirements

ARE – Autonomy Requirements Engineering

GORE for ARE

- ✓ extended upstream the software development process by adding a new phase - early requirements analysis
- ✓ fundamental concepts used to drive the goal-oriented form of analysis are those of goal and actor
- ✓ helps to analyze the space of alternatives, which makes the process more systematic - an explicitly represented space of alternatives
- ✓ produces goals models that represent system objectives and their inter-relationships

ARE – Autonomy Requirements Engineering

GAR – Generic Autonomy Requirements

- ✓ Autonomicity (self-* objectives):
  - aims at freeing human operators from complex tasks (a lot of decision making)
  - provides autonomy behavior (e.g., self-protecting)
  - refines decisions concerning the priority and quality of service

The simplest form of autonomicity:

- a rule-based engine obeying a predefined set of conditional statements (e.g., if-then-else) put in an endless loop
- not sufficient - the rule-based engine should enforce learning capabilities (e.g., reinforcement learning)

ARE – Autonomy Requirements Engineering

GAR – Generic Autonomy Requirements

- ✓ Knowledge – structured knowledge:
  - ➢ implies knowledge structuring and processing (reasoning)
  - ➢ domain knowledge, system knowledge, control knowledge
- ✓ Awareness – product of monitoring and knowledge reasoning
- ✓ Monitoring – obtaining raw data through a collection of sensors
- ✓ Adaptability – an ability to achieve change in observable behavior and/or structure:
  - ➢ changes in functionality, algorithms, system parameters, or structure
  - ➢ amplified by self-adaptation

ARE – Autonomy Requirements Engineering

GAR – Generic Autonomy Requirements

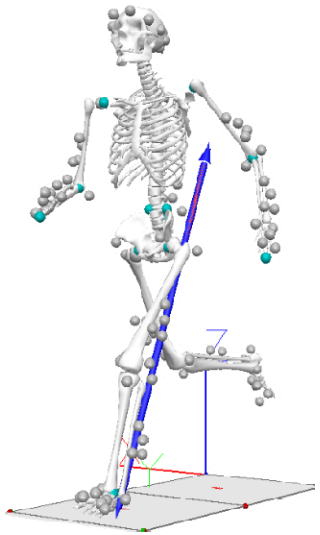
- ✓ Dynamicity – technical ability to change at runtime:
  - ➢ ability to remove, add or exchange services and components
  - ➢ starting, stopping and running functions
- ✓ Robustness – cope with errors during execution:
  - ➢ facilitate the design of system parts that deal with self-healing and self-protecting
  - ➢ error-avoidance and error-propagation-stopping strategies: error avoidance, error prevention, and fault tolerance

ARE – Autonomy Requirements Engineering

GAR – Generic Autonomy Requirements

- ✓ Resilience – a quality attribute prerequisite for resilience and system agility:
  - ➢ closely related to safety - enables systems to bounce back from unanticipated disruptions

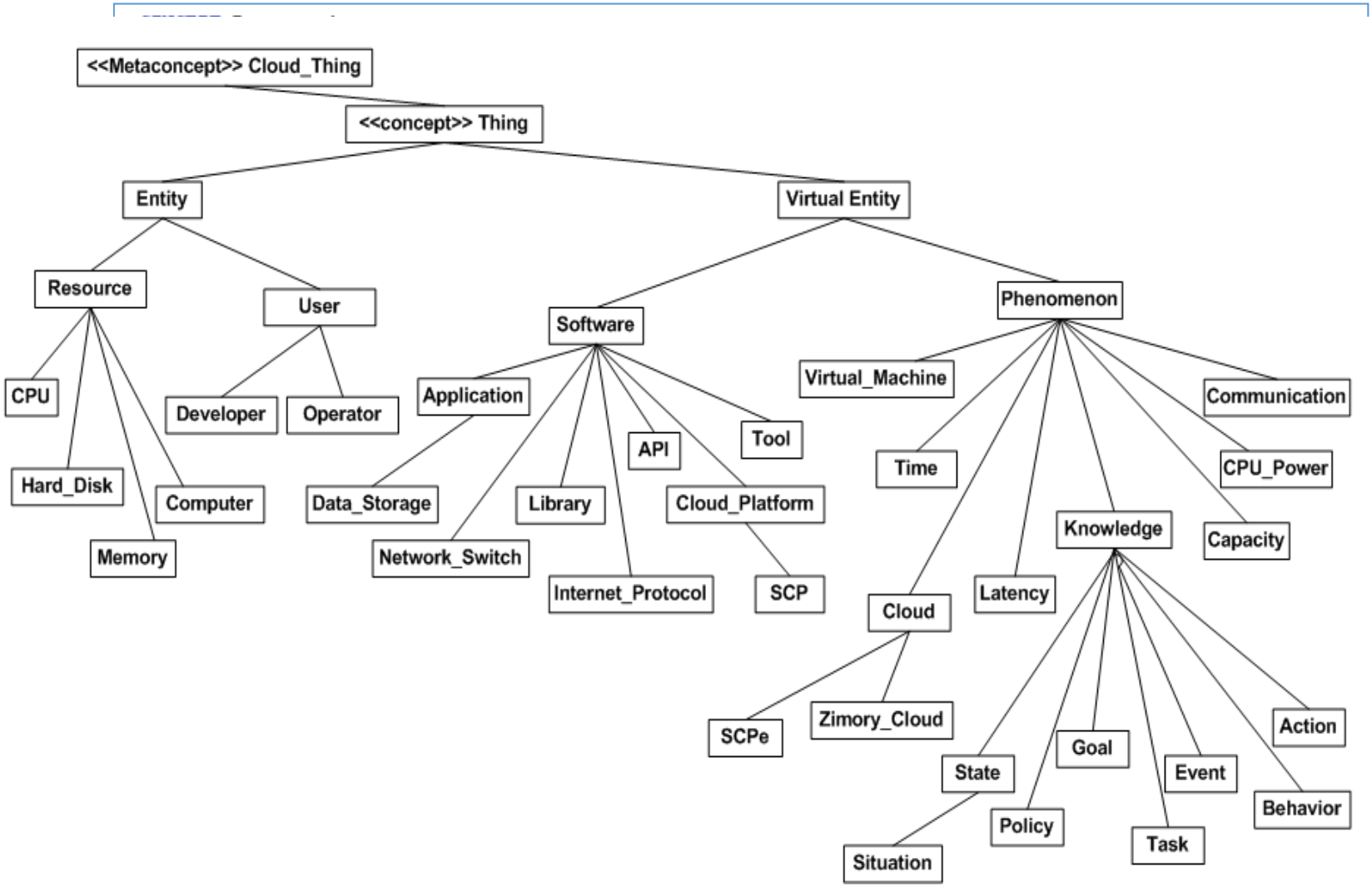
- ✓ Mobility – what moves in the system (design and runtime):
  - ➢ code mobility, service mobility, component mobility, resource mobility, etc.
  - ➢ enables dynamic discovery and usage of new resources, recovery, etc.



KnowLang – The ARE Formal Language

Frame
self-ac

- ✓ kno
  Knc
  bas
  kno
  (KR
- ✓ forr
  mul
  spe
  allo
  ont
  rule
  net



KnowLang – The ARE Formal Language

Knowledge structures and a reasoning mechanism:

- *policies*, *events*, *actions*, *situations*, and *relations* between policies and situations;
- a policy π has a *goal* (g), *policy situations* (Si_π), *policy-situation relations* (R_π), and *policy conditions* (N_π) *mapped* to *policy actions* (A_π);
- *policy situations* (Si_π) may trigger (or imply) a policy , in compliance with the *policy-situation relations R;*
- probabilistic beliefs (Z) justify the probability of policy execution;
- relations (*R)* must be specified to connect policies with situations over an optional probability distribution (Z);

$$\Pi := \{\pi_1, \pi_2, ..., \pi_n\}, n \geq 0 \quad \textit{(Policies)}$$

$$\pi :=< g, Si_\pi, [R_\pi], N_\pi, A_\pi, map(N_\pi, A_\pi, [Z]) >$$

$$A_\pi \subset A, N_\pi \overset{[Z]}{\rightarrow} A_\pi \quad \textit{(A_π - Policy Actions)}$$

$$Si_\pi \subset Si, Si_\pi := \{si_{\pi_1}, si_{\pi_2}, ..., si_{\pi_n}\}, n \geq 0$$

$$R_\pi \subset R, R_\pi := \{r_{\pi_1}, r_{\pi_2}, ..., r_{\pi_n}\}, n \geq 0$$

$$\forall r_\pi \in R_\pi \bullet (r_\pi :=< si_\pi, [rn], [Z], \pi >), si_\pi \in Si_\pi$$

$$Si_\pi \overset{[R_\pi]}{\rightarrow} \pi \rightarrow N_\pi$$

$$N_\pi := \{n_1, n_2, ..., n_k\}, k \geq 0 \quad \textit{(Conditions)}$$

$$n := be(O) \quad \textit{(Condition - Boolean Expression)}$$

$$g := \langle \Rightarrow s' \rangle | \langle s \Rightarrow s' \rangle \quad \textit{(Goal)}$$

$$s := be(O) \quad \textit{(State)}$$

$$Si := \{si_1, si_2, ..., si_n\}, n \geq 0 \quad \textit{(Situations)}$$

$$si :=< s, A \overset{\leftarrow}{si}, [E \overset{\leftarrow}{si}], A_{si} > \quad \textit{(Situation)}$$

$$A \overset{\leftarrow}{si} \subset A \quad \textit{($A \overset{\leftarrow}{si}$ - Executed Actions)}$$

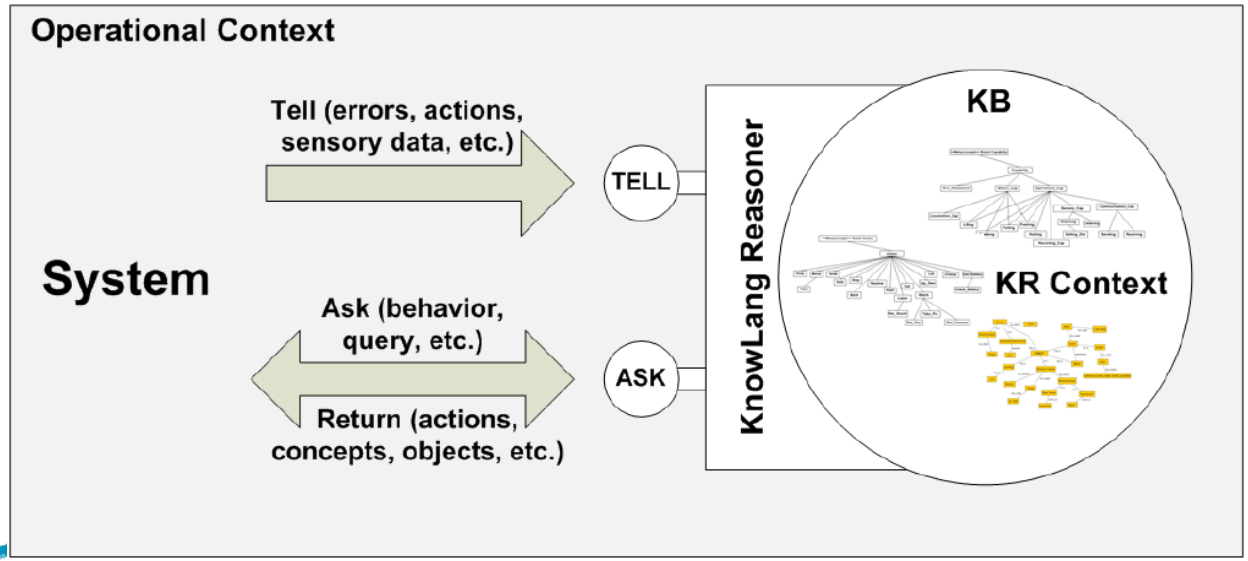
$$A_{si} \subset A \quad \textit{(A_{si} - Possible Actions)}$$

$$E \overset{\leftarrow}{si} \subset E \quad \textit{($E \overset{\leftarrow}{si}$ - Situation Events)}$$

KnowLang – The ARE Formal Language

KnowLang Reasoner:

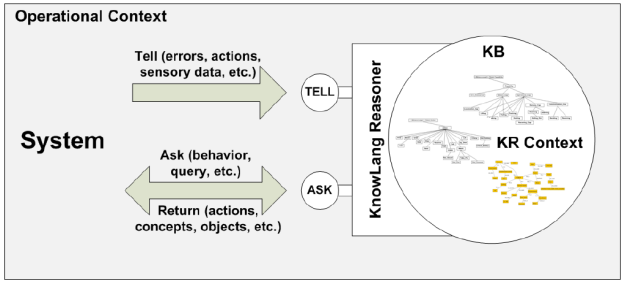
- ✓ supplied as a component hosted by a self-adaptive system
- ✓ runs in the system's Operational Context and operates in the KR Context
- ✓ ASK and TELL operators provide knowledge queries and updates
- ✓ builds up and returns a self-adaptive behavior - a chain of actions to be realized in the environment or in the system



Verification of Autonomy Requirements

ARE-Based Test Bed

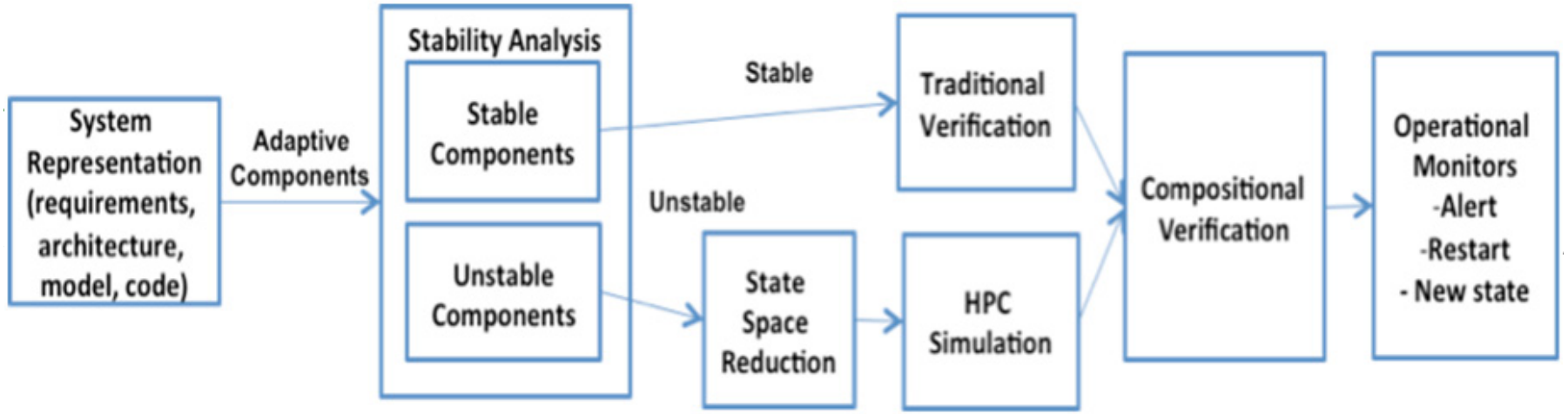
- ✓ allows testing of autonomy properties under simulated conditions
- ✓ aims at testing autonomous behavior by validating self-* objectives through evaluation of the system's ability to perceive both the internal and external environments and react to changes
- ✓ an application that embeds the ARE-adapted KnowLang Reasoner and allows direct communication with the reasoner via predefined operators using the reasoner's ASK and TELL interface:
  - ➢ formal KnowLang models can be loaded and executed on the reasoner
  - ➢ simple simulation conditions can be enforced
  - ➢ test results can be obtained through direct communication with the reasoner



Verification of Autonomy Requirements

Adaptive Behavior Verification (ABV)

1) a stability analysis (linear system stability analysis - Routh-Hurwitz, Root Locus) capability that identifies instabilities given a system model and



4) a compositional verification capability that integrates individual component verifications

5) operational monitors to detect and take action to correct undesired unstable behavior of the system during operation

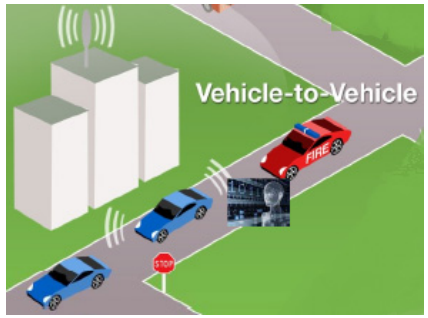
eMobility Case Study

Smart mobility transportation concept:

- ✓ based on a network of electrical vehicles
- ✓ e-vehicles are competing for infrastructure resources of the traffic environment:
  - ➢ roads, parking lots, and charging stations have a limited capacity
  - ➢ the availability of infrastructure resources may not match the demand
- ✓ exhibits self-adaptation:
  - ➢ an important paradigm making eMobility capable of modifying system behavior and/or structure in response to increasing workload demands and service failures
  - ➢ required to ensure that services will be provided in a fail-safe manner and under consideration of system goals

eMobility Case Study

- ✓ e-vehicles move according to a schedule defined by a driver
- ✓ e-vehicles automatically determine:
  - ➢ optimal routes
  - ➢ time constraints imposed by the driver's schedule
  - ➢ required space at particular POIs
- ✓ e-vehicles are equipped with a Route Planner that plans travels
- ✓ routes are composed of multiple driving locations, e. g., POIs
- ✓ self-adaption is required:
  - ➢ availability of resources does not match the demand;
  - ➢ environment constraints (e.g., speed limit, or delay due to high traffic)



eMobility Case Study

ARE for eMobility

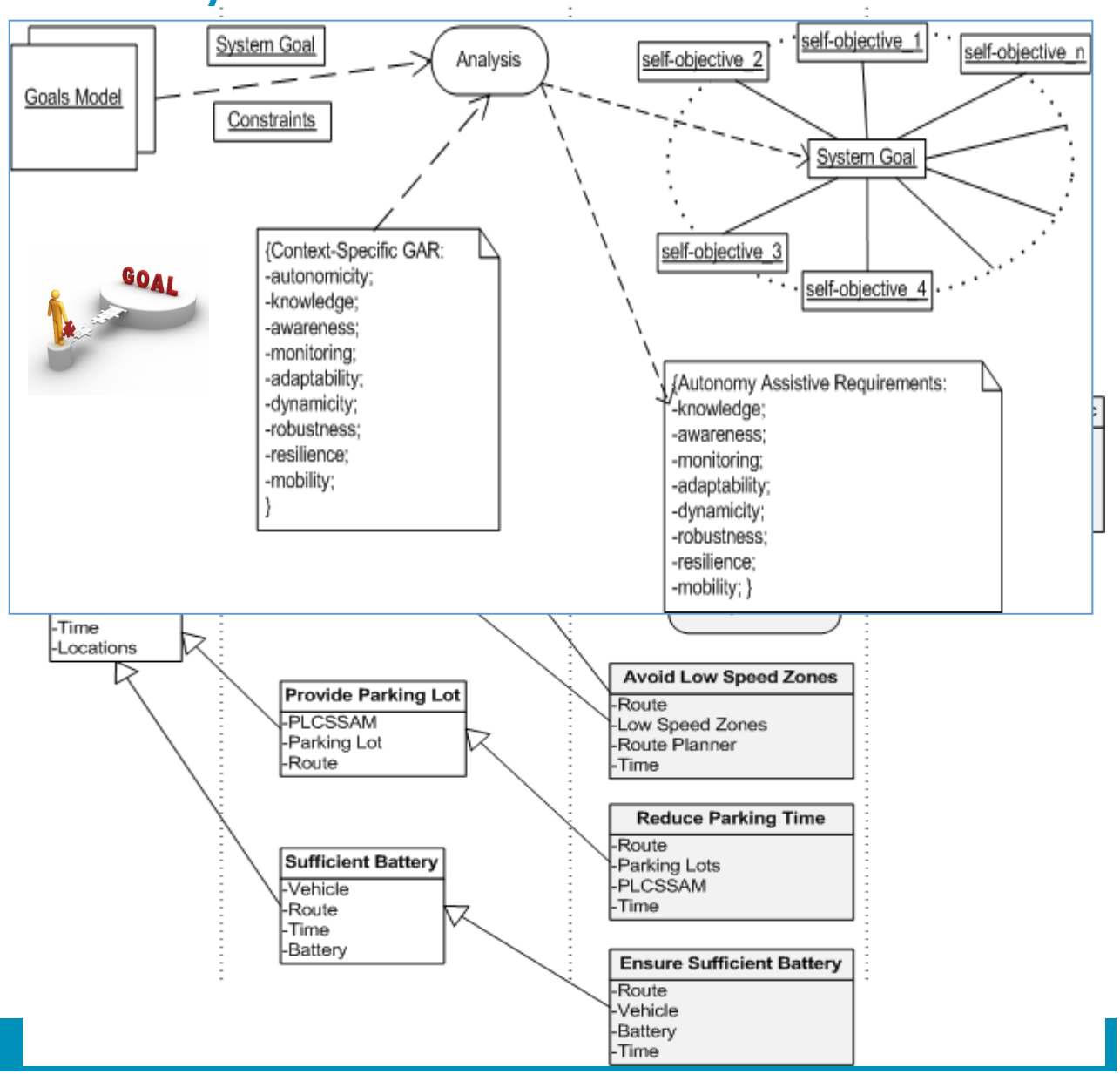
- ✓ ARE applied to capture the autonomy requirements for eMobility:
  - ➢ helped to determine the right level of abstraction for the knowledge representation model
- ✓ GORE used to build goals models for eMobility:
  - ➢ the objectives of the system
  - ➢ restrictions that exist in this environment

  - ➢ immediate targets supporting the system objectives
  - ➢ constraints the system needs to address
  - ➢ system actors: Vehicles, Drivers, Charging Stations, etc

eMobility Case Study

ARE for eMobility

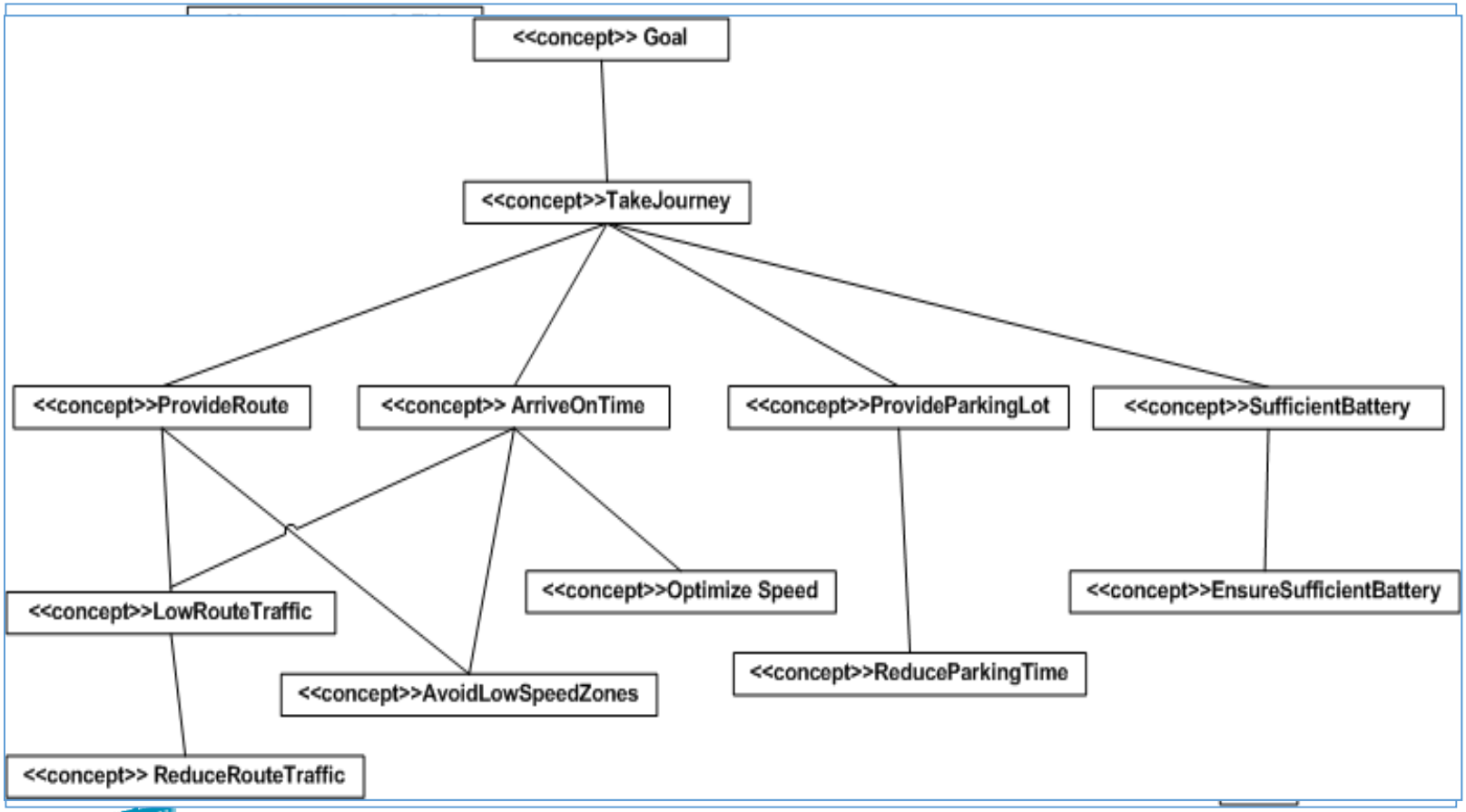
- ✓ GORE + GAR
- ✓ goals model with self-* objectives assisting system goals



eMobility Case Study

KnowLang model for eMobility

- ✓ eMobility Knowledge Base: eMobility ontology



eMobility Case Study

KnowLang model for eMobility

- ✓ eMobility Knowledge Base: concepts (properties, functions, states)

```
CONCEPT Route {
   PARENTS {eMobility.eCars.CONCEPT_TREES.Phenomenon}
   CHILDREN {}
   PROPS {
      PROP locationA {TYPE {eMobility.eCars.CONCEPT_TREES.Location} CARDINALITY {1}}
      PROP locationB {TYPE {eMobility.eCars.CONCEPT_TREES.Location} CARDINALITY {1}}
      PROP intermediateStops {TYPE {eMobility.eCars.CONCEPT_TREES.Location} CARDINALITY {*}}
      PROP currentRoad {TYPE {eMobility.eCars.CONCEPT_TREES.Road} CARDINALITY {1}}
      PROP alternativeRoads {TYPE {eMobility.eCars.CONCEPT_TREES.Road} CARDINALITY {*}}
   }
   FUNCS {
      FUNC getCurrentLocation {TYPE {eMobility.eCars.CONCEPT_TREES.GetCurrentLocation}}
      FUNC takeAlternativeRoad {TYPE {eMobility.eCars.CONCEPT_TREES.TakeAlternativeRoad}}
      FUNC recomputeRoads {TYPE {eMobility.eCars.CONCEPT_TREES.RecomputeRoads}}
   }
   STATES {
      STATE AtBeginning {eMobility.eCars.CONCEPT_TREES.Route.FUNCS.getCurrentLocation =
                         eMobility.eCars.CONCEPT_TREES.Route.PROPS.locationA}
      STATE AtEnd {eMobility.eCars.CONCEPT_TREES.Route.FUNCS.getCurrentLocation =
                         eMobility.eCars.CONCEPT_TREES.Route.PROPS.locationB}
      STATE OnRoute { NOT eMobility.eCars.CONCEPT_TREES.Route.STATES.AtBeginning AND
                      NOT eMobility.eCars.CONCEPT_TREES.Route.STATES.AtEnd}
      STATE InHighTraffic {eMobility.eCars.CONCEPT_TREES.Route.PROPS.currentRoad.STATES.InHighTraffic}
      STATE InLowTraffic {eMobility.eCars.CONCEPT_TREES.Route.PROPS.currentRoad.STATES.InFluentTraffic}
   }
}
```

eMobility Case Study

KnowLang model for eMobility

- ✓ eMobility self-adaptive behavior: goals, situations, policies, relations

```
CONCEPT_POLICY ReduceRouteTraffic {
   CHILDREN {}
   PARENTS {eMobility.eCars.CONCEPT_TREES.Policy}
   SPEC {
      POLICY_GOAL {eMobility.eCars.CONCEPT_TREES.LowRouteTraffic}
      POLICY_SITUATIONS {eMobility.eCars.CONCEPT_TREES.RouteTrafficIncreased}
      POLICY_RELATIONS {eMobility.eCars.RELATIONS.Situation_Policy_1}
      POLICY_ACTIONS {eMobility.eCars.CONCEPT_TREES.TakeAlternativeRoad,
                      eMobility.eCars.CONCEPT_TREES.RecomputeRoads}
      POLICY_MAPPINGS {
         MAPPING {
            CONDITIONS {eMobility.eCars.CONCEPT_TREES.Route.STATES.OnRoute}
            DO_ACTIONS {eMobility.eCars.CONCEPT_TREES.Route.FUNCS.takeAlternativeRoad}
            PROBABILITY {0.7}
         }
         MAPPING {
            CONDITIONS { eMobility.eCars.CONCEPT_TREES.Route.STATES.OnRoute}
            DO_ACTIONS { eMobility.eCars.CONCEPT_TREES.Route.FUNCS.recomputeRoads,
                         eMobility.eCars.CONCEPT_TREES.Route.FUNCS.takeAlternativeRoad}
            PROBABILITY {0.3}
         }
         MAPPING {
            CONDITIONS { eMobility.eCars.CONCEPT_TREES.Route.STATES.AtBeginning}
            DO_ACTIONS { eMobility.eCars.CONCEPT_TREES.Route.FUNCS.recomputeRoads,
                         eMobility.eCars.CONCEPT_TREES.Route.FUNCS.takeAlternativeRoad}
         }
      }
   }
}
```

eMobility Case Study

KnowLang model for eMobility

- ✓ eMobility monitoring: metrics (abstraction of sensors)

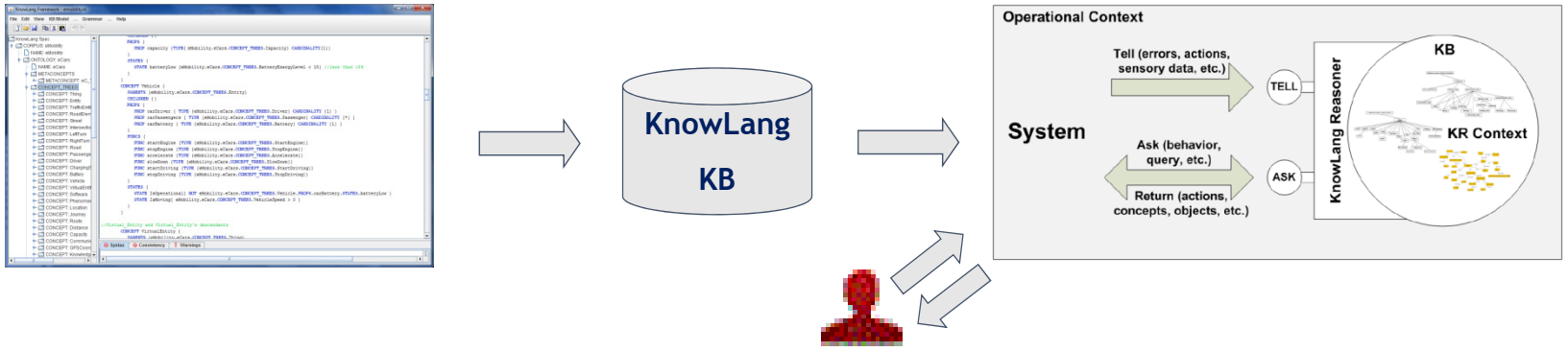
```
CONCEPT_METRIC RoadTrafficLevel {
   CHILDREN {}
   PARENTS {eMobility.eCars.CONCEPT_TREES.Metric}
   SPEC {
      METRIC_TYPE { ENVIRONMENT }
      METRIC_SOURCE { "ECarClass.GetRoadTrafficLevel" }
      DATA_TYPE { NUMBER }
   }
}
CONCEPT_METRIC BatteryEnergyLevel {
   CHILDREN {}
   PARENTS {eMobility.eCars.CONCEPT_TREES.Metric}
   SPEC {
      METRIC_TYPE { RESOURCE }
      METRIC_SOURCE { "ECarClass.GetBatteryEnergyLevel" }
      DATA_TYPE { NUMBER }
   }
}
CONCEPT_METRIC JourneyBatterySufficiency {
   CHILDREN {}
   PARENTS {eMobility.eCars.CONCEPT_TREES.Metric}
   SPEC {
      METRIC_TYPE { RESOURCE }
      METRIC_SOURCE { "ECarClass.GetJourneyBatterySufficiency" }
      DATA_TYPE { BOOLEAN }
   }
}
```

```
CONCEPT_METRIC VehicleSpeed {
   CHILDREN {}
   PARENTS {eMobility.eCars.CONCEPT_TREES.Metric}
   SPEC {
      METRIC_TYPE { RESOURCE }
      METRIC_SOURCE { "ECarClass.GetVehicleSpeed" }
      DATA_TYPE { NUMBER }
   }
}
CONCEPT_METRIC JourneyTime {
   CHILDREN {}
   PARENTS {eMobility.eCars.CONCEPT_TREES.Metric}
   SPEC {
      METRIC_TYPE { RESOURCE }
      METRIC_SOURCE { "ECarClass.GetJourneyTime" }
      DATA_TYPE { DATETIME }
   }
}
```

Simulation and Testing

Simulation and Testing with the ARE-Based Test Bed

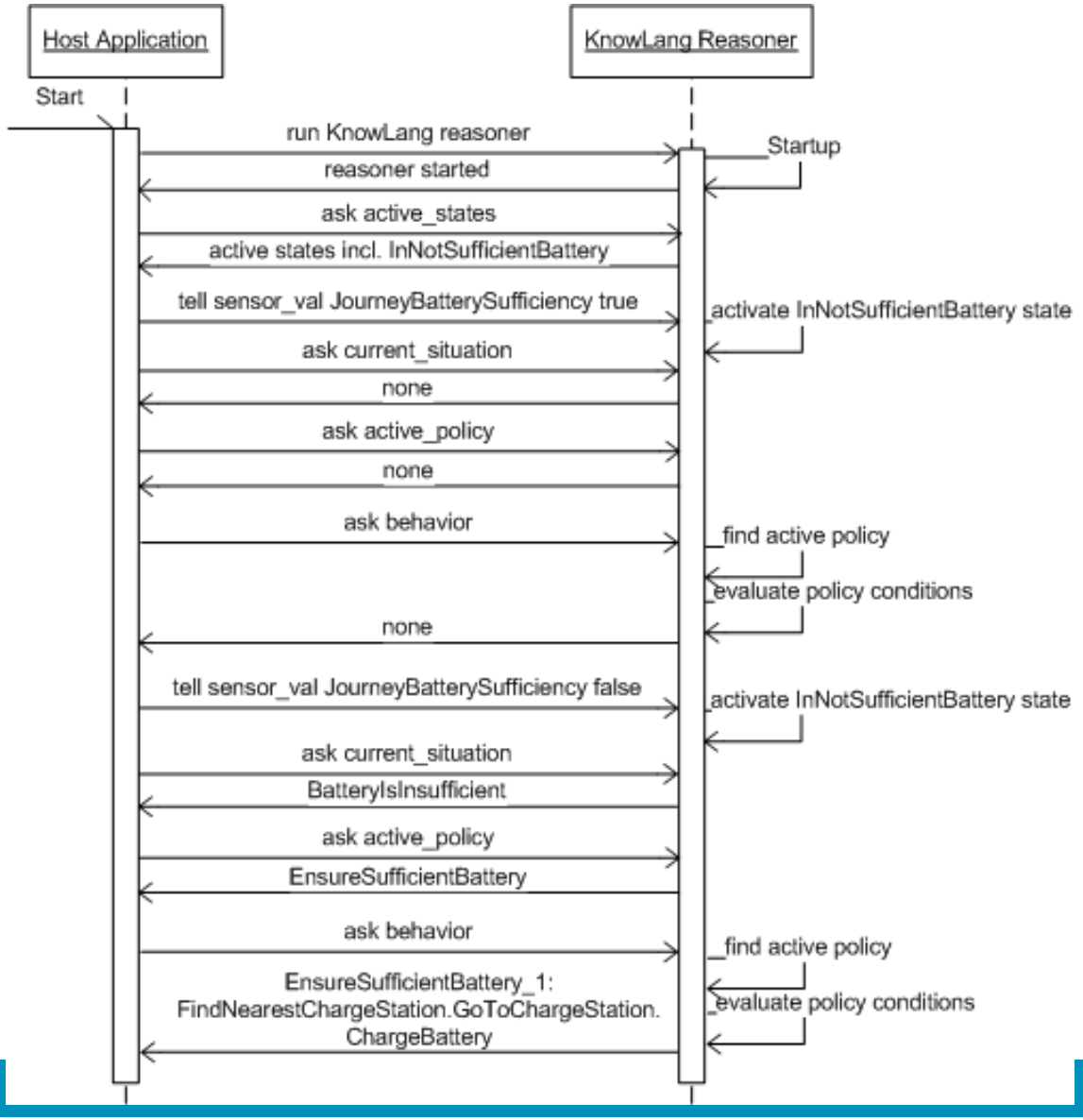
- ✓ simulation exercise:
  - ➢ host application running the KnowLang Reasoner
  - ➢ the KnowLang Reasoner operates over the eMobility KB
  - ➢ users enter ASK and TELL commands via command line



Simulation and Testing

Simulation:

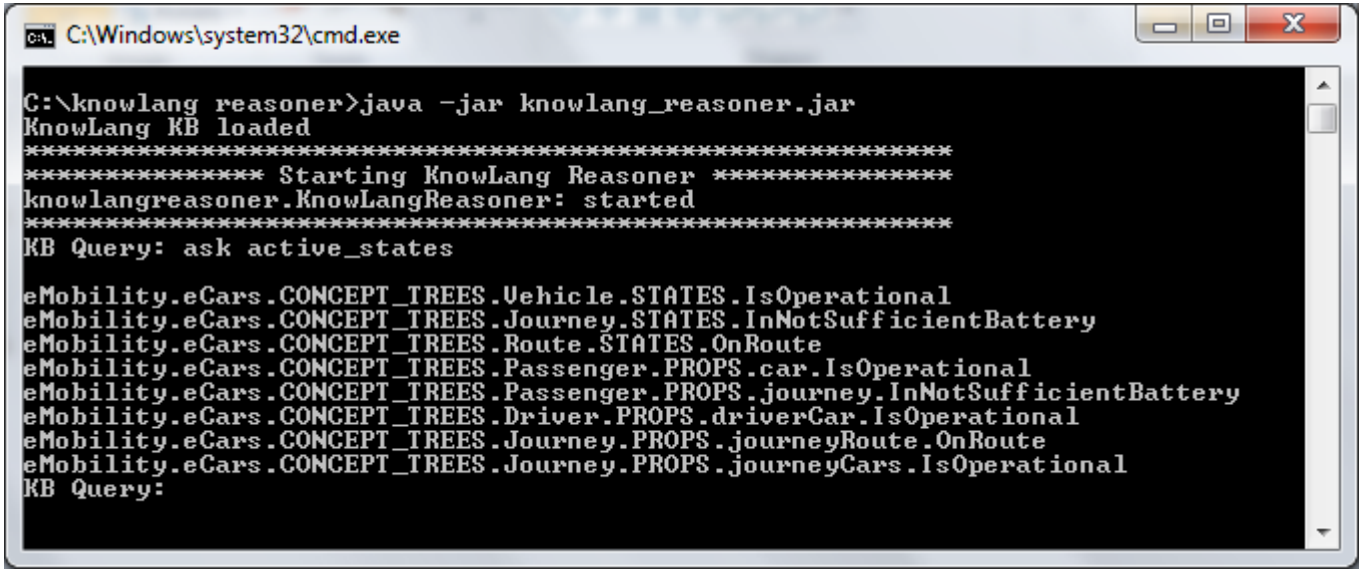
- ✓ simulated sensory inputs raise an ”insufficient battery” situation
- ✓ simulated awareness:
  - ➢ active states
  - ➢ current situations
  - ➢ active policies
- ✓ self-adaptive behavior



Simulation and Testing

Demonstration:

- ✓ KnowLang eMobility.kl
- ✓ ARE-based test bed



Conclusion

Smart vehicles – loaded with AI and operate in nondeterministic environment:

- ✓ reliable smart vehicles - we need to plan for uncertainty by capturing the autonomy requirements
- ✓ ARE and KnowLang – a unique way to capture autonomy requirements
  - ➢ ARE elicits autonomy requirements
  - ➢ KnowLang formalizes autonomy requirements
  - ➢ eMobility case study
- ✓ simulation & testing:
  - ➢ host application running the KnowLang Reasoner
  - ➢ the KnowLang Reasoner operates over the eMobility KB
  - ➢ users enter ASK and TELL commands via command line

SFI Science Foundation Ireland

Ireland's European Structural and Investment Funds Programmes 2014-2020
Co-funded by the Irish Government and the European Union

European Union
European Regional Development Fund

SOUTHERN Regional Assembly
Promoting our Region