Towards Self-Explainable Cyber-Physical Systems

"Does your car have any idea why my car pulled it over?" [1]

Mathias Blumreiter, Joel Greenyer, Francisco Javier Chiyah Garcia, **Verena Klös**,
Maike Schwammberger, Christoph Sommer, Andreas Vogelsang and Andreas Wortmann

14th International Workshop on Models@run.time 2019

Motivation


uncertainties
safety-critical
dynamic networks
autonomous decisions
run-time evolution
TRUST ???
ERP

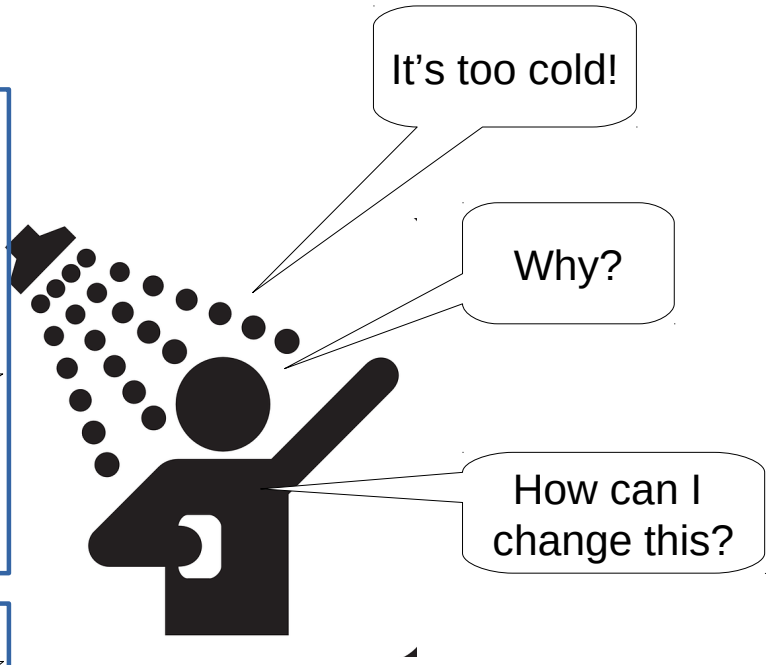
Vision

explanation of past & current behavior

Warm water will be available in 20 minutes.

You normally don't shower before 7 a.m. The water boiler starts heating At 6.30 a.m. to save energy.

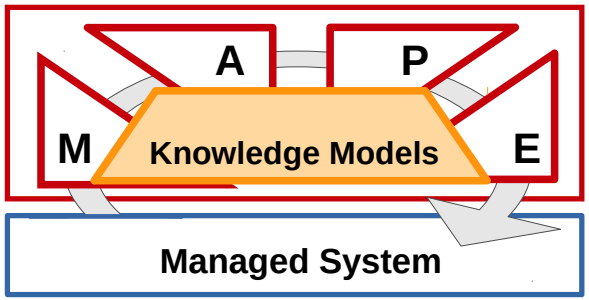
I found three options:
1) switch off energy saving
2) manually enter starting time
3) let me sync with your alarm



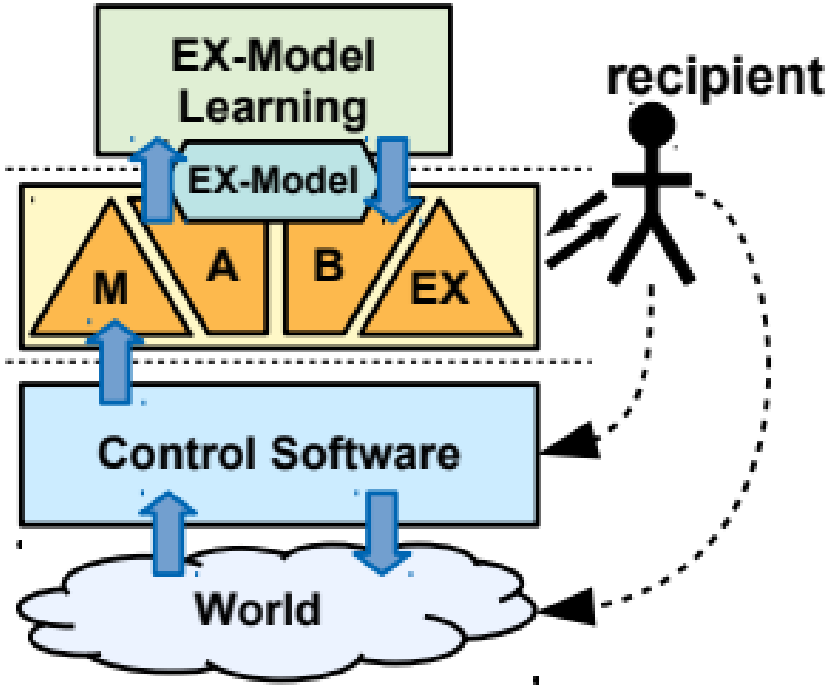
answer questions about the system's future behavior

The MAB-EX Loop for Explainability

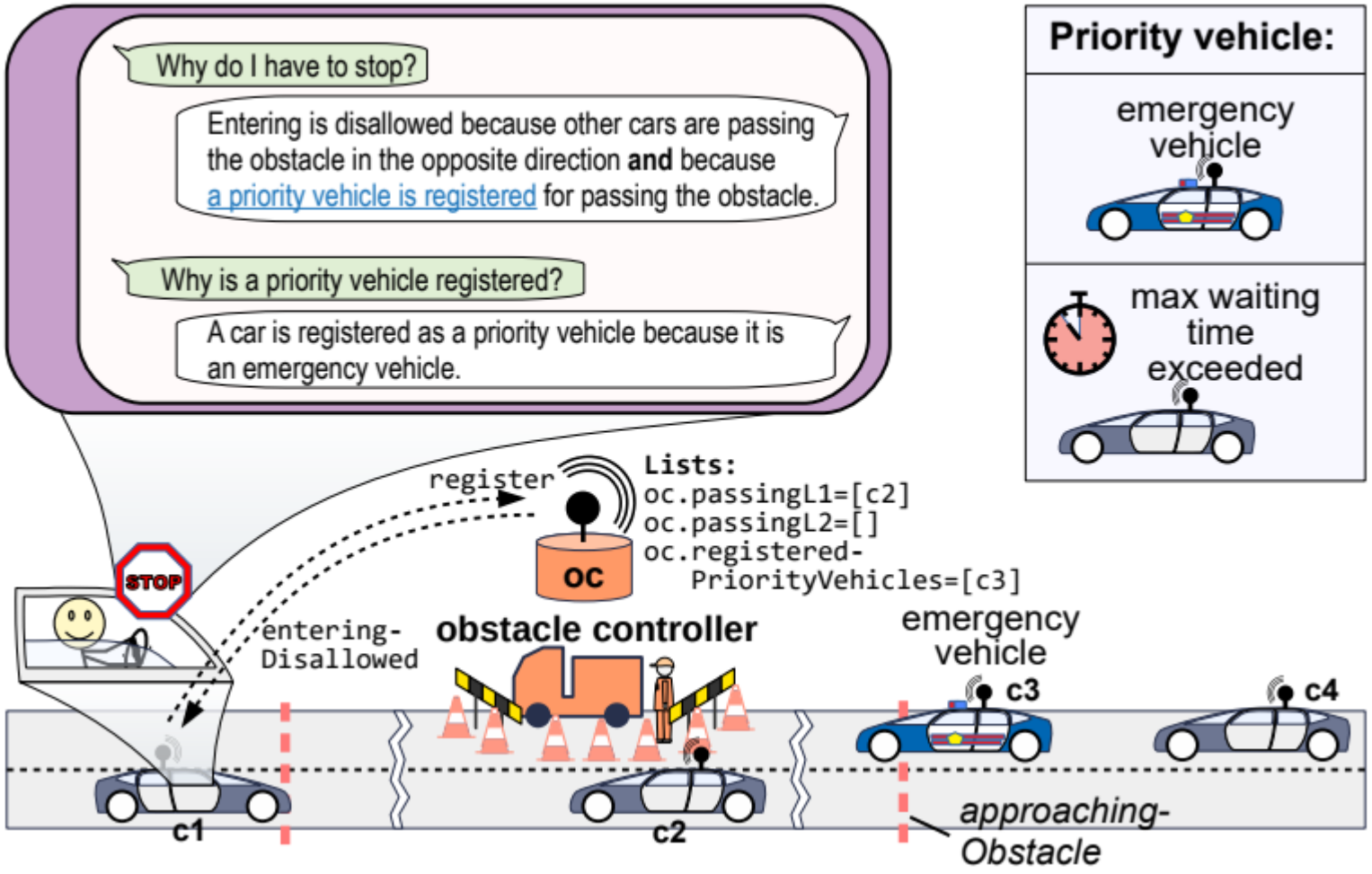
Self-Explaining:
- **autonomously detect** need for explanations
- provide **recipient-specific** explanations
- **learn** from observations & interactions



MAPE-K Loop from IBM [2]



Example: V2X driver assistance system



Monitor

example: position of the car, answer of the controller

relevant sensor data

commands from controller components

user and/or system interactions & former explanations

Analyze

example: car on lane L1 and enteringDisallowed?

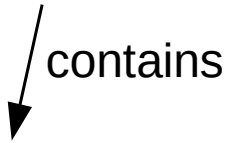
process explanation queries from recipient

detect behavior that requires an explanation

(e.g., irregularities in the monitored sensor data, sudden changes in the user interactions)

Build

evaluate *explanation model* to build explanation

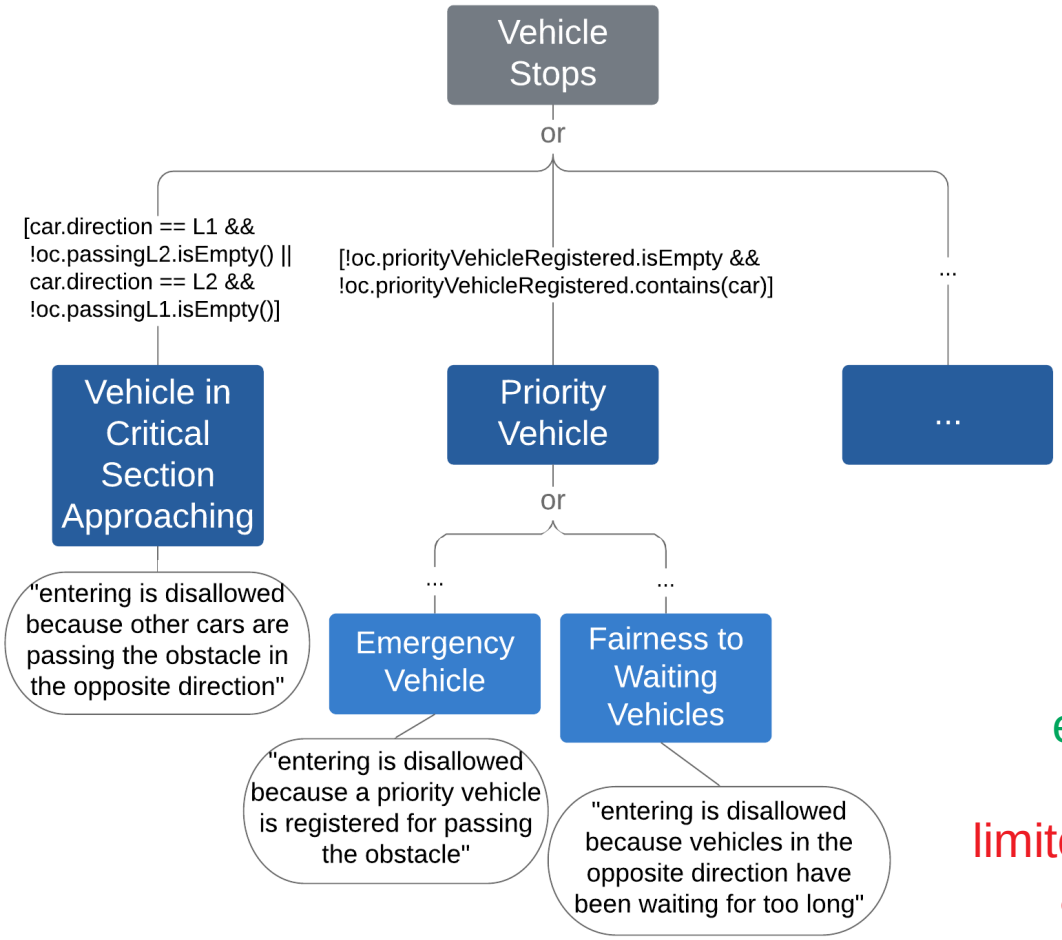


causal relationships between events and system reactions

→ traces of events

→ look-ahead simulation (“What happens if ... ?”, “When will ... be possible again?”)

Example: Models of Causality Approach [3]



easy to model & integrate

limited to anticipated phenomena & past/current situations

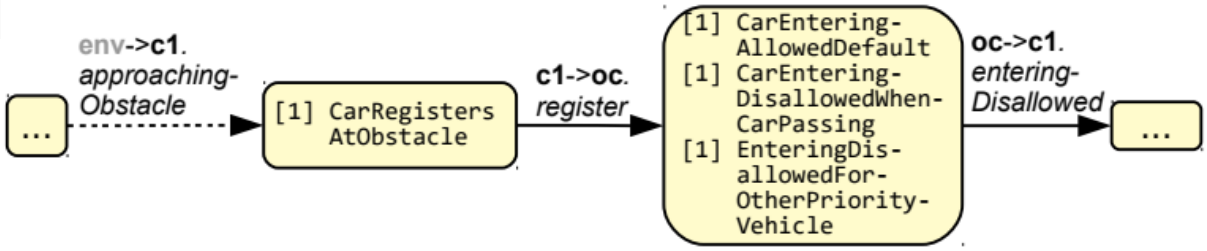
Example: Explanations from Run-Time Models

```
...
guarantee scenario CarRegistersAtObstacle
 bindings [oc = cp.obstacleCtrl] {
 sensor -> car.approachingObstacle()
 //@EX: when approaching an obstacle, the car must register at
       the obstacle control
 strict requested car -> oc.register()
}

guarantee scenario CarEnteringAllowedDefault {
 car -> oc.register()
 // @EX: entering is allowed because there is no indication to
       disallow it.
  requested oc -> car.enteringAllowed()
} constraints [
 interrupt oc -> car.enteringDisallowed()
]

guarantee scenario CarEnteringDisallowedWhenCarPassing {
 car -> oc.register()
 alternative [car.direction == L1 && !oc.passingL2.isEmpty() ||
       car.direction == L2 && !oc.passingL1.isEmpty()] {
  // @EX: entering is disallowed because other cars are passing
        the obstacle in the opposite direction.
  strict requested oc -> car.enteringDisallowed()
 } constraints [
  forbidden oc -> car.enteringAllowed()
 ]
}
```

Scenario Modeling Language [4]

can be queried and executed for look-ahead predictions

higher modeling effort



Explain

example: “Entering is disallowed because other cars are passing the obstacle in the opposite direction and a priority vehicle is registered for passing the obstacle“

understandable explanation for the recipient

based on *recipient model:*

mental model of a human

explanation interface

between different systems

→ explanation format, level of abstraction, points of interest

EX-Model Learning

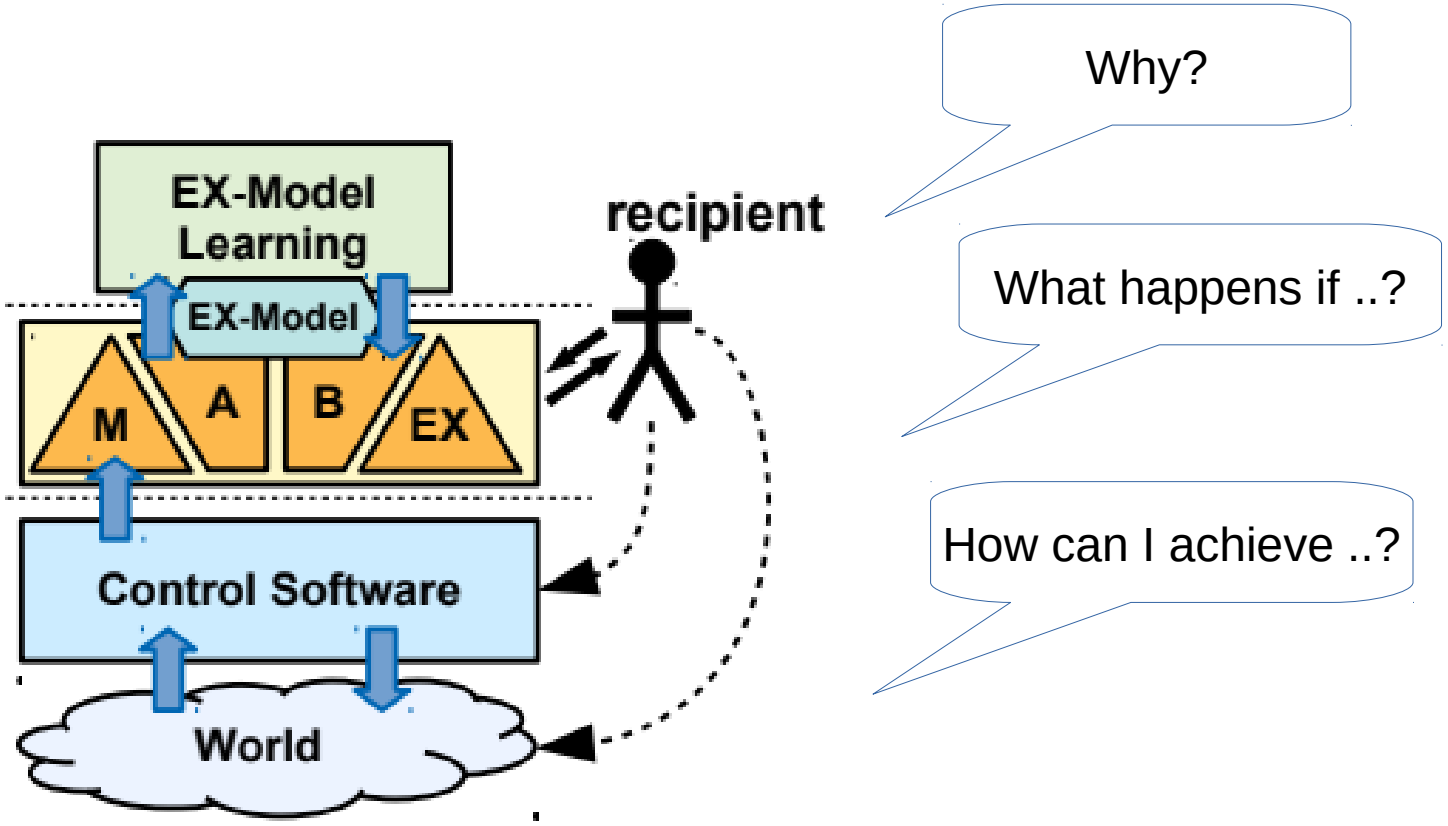
system and recipient may evolve over time

uncertainties at design time (about the system behavior, operational context, and the recipient and its preferences)

→ update *explanation model* and *recipient model*

possible realizations: machine learning algorithms, expert system, learning from user reactions

Summary


EX-Model Learning
EX-Model
M
A
B
EX
Control Software
World
recipient
Why?
What happens if ..?
How can I achieve ..?

References

[1] https://www.newyorker.com/cartoon/a19697?verso=true

[2] “An Architectural Blueprint for Autonomic Computing,” IBM, White Paper, Jun. 2005.

[3] F. J. Chiyah Garcia, D. A. Robb, X. Liu, A. Laskov, P. Patron, and H. Hastie, “Explain Yourself: A Natural Language Interface for Scrutable Autonomous Robots,” in Explainable Robotic Systems Workshop (HRI), 2018.

[4] J. Greenyer, D. Gritzner, T. Gutjahr, F. König, N. Glade, A. Marron, and G. Katz, “ScenarioTools – A tool suite for the scenario-based modeling and analysis of reactive systems,” Elsevier Science of Computer Programming, vol. 149, pp. 15–27, 2017, Special Issue on MODELS’16.