ActivFORMS: Active Formal Models for Self-Adaptive Systems

NII Shonan Meeting
Engineering Adaptive Software Systems (EASSy)

M. Usman Iftikhar – usman.iftikhar@lnu.se
Danny Weyns – danny.weyns@lnu.se
Kostiantyn Kucher – kostiantyn.kucher@lnu.se

Promise of formal models for self-adaptive systems*

Providing evidence that the system requirements are satisfied during operation

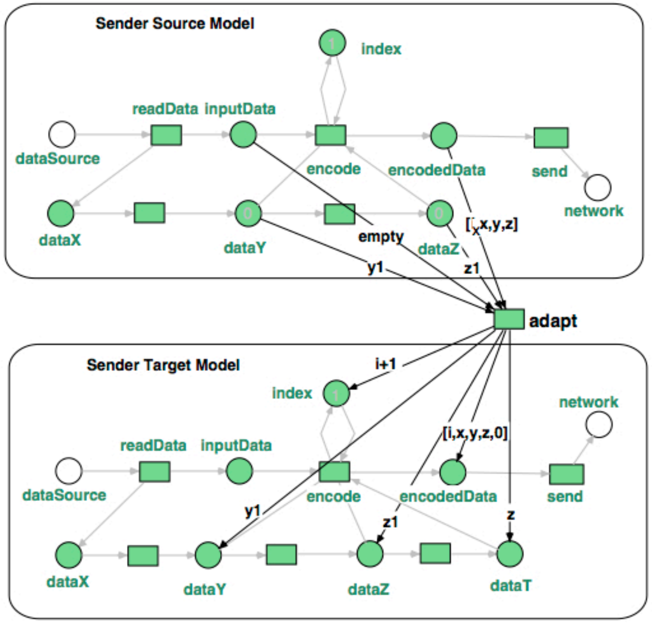
regarding the uncertainty of changes that may affect the system, its environment or its goals

*Software Engineering for Self-Adaptive Systems: Assurances (www.dagstuhl.de/de/programm/kalender/semhp/?semnr=13511)

Overview

- State of the art (some key contributions)
- Our proposal
- Approach
- Realization
- Contributions & Tradeoffs
- Future research

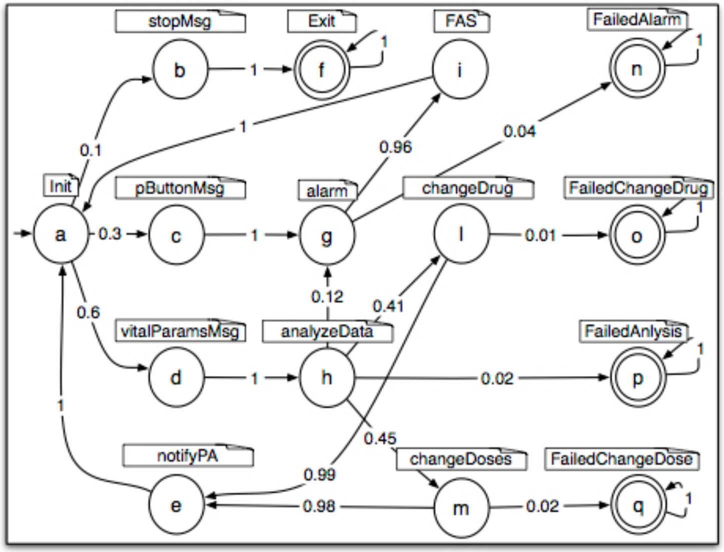
Model-based development of dynamically adaptive software [Zhang & Cheng, ICSE 2006]



- Process to create and verify formal models and automatically generate programs from them (Petri nets and LTL)

- Assuring properties of self-adaptive systems: need for formal underpinning
- Need for clear separation between domain logic and adaptation logic

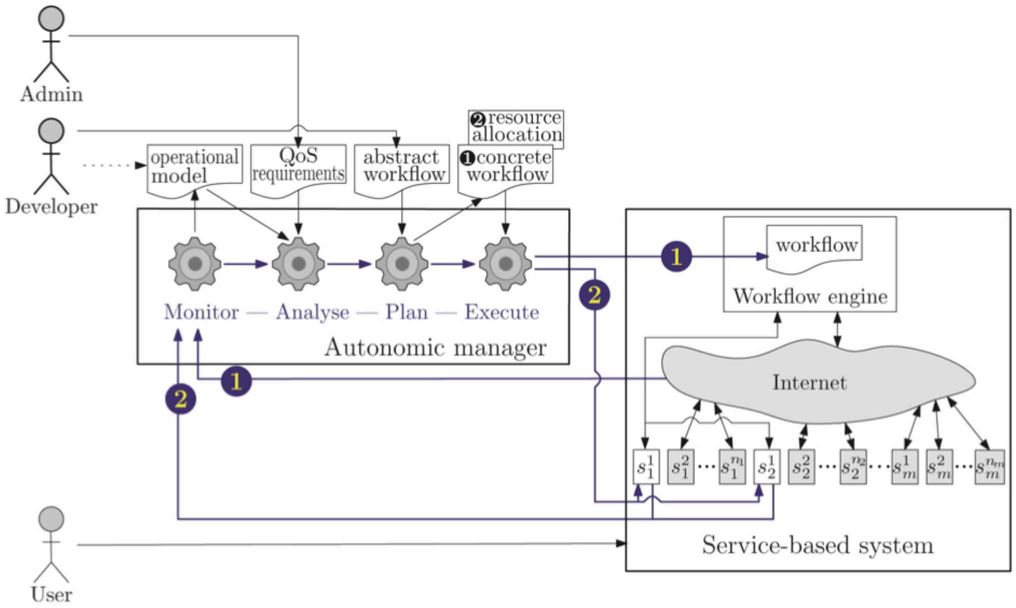
Model evolution by runtime parameter adaptation [Epifani et al., ICSE 2009]

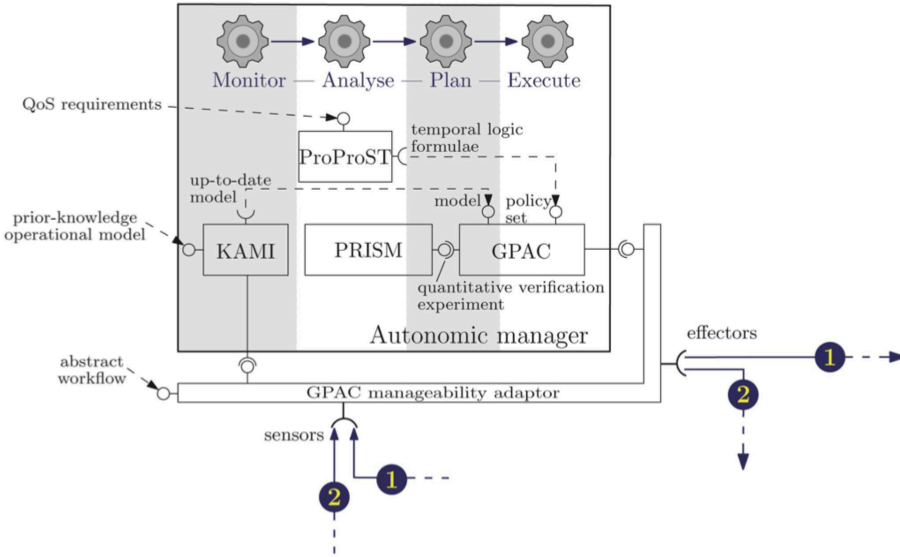


- Probabilistic model represents execution flows of the system
- Probabilities are dynamically updated based on observations

- Formal model of system behavior at runtime: focus on K of MAPE-K

Dynamic QoS management and optimization in service based systems [Calinescu et al. TSE 2011]

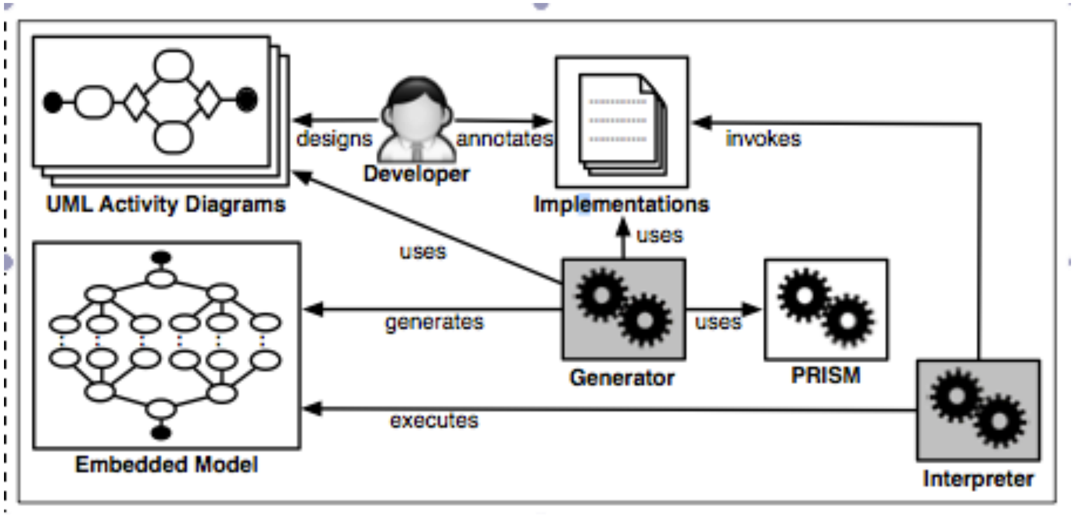




- MAPE-K manager monitors service-based system and adapts workflow engine (service selection + resources)
- Online verification of reliability and performance properties

- Formal model covers the system abstraction + goals (K)
- Adaptation logic consists of set of tools that are glued together

Managing non-functional uncertainty via model-driven adaptivity [Ghezzi et al. ICSE 2013]



- Model with probability distribution of different execution paths of the system
- Interpreter guides the execution of the system using the model
- To guarantee highest utility for set of quality properties

- No clear separation of concerns (domain logic and adaptation logic)
- Adaptation logic is encoded in ad-hoc interpreter

Summary SOTA

- Increasing attention for formal models at runtime to provide guarantees of adaptation
- Quantitative approaches dominate
- Focus on formal models of system, environment and goals (K of MAPE-K)
- No systematic formalization and verification of of adaptation functions (MAPE of MAPE-K)
- Limited support for unpredicted changes

What is needed?

- Formalize adaptation functions to provide guarantees about adaptation
- Support unanticipated changes
  - Require support for adaptations of adaptation functions
- Scalability of runtime verification

Overview

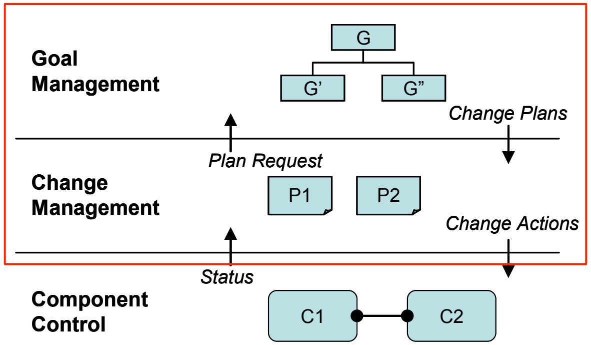
- State of the art (some key contributions)
- Our proposal
- Approach
- Realization
- Contributions & Tradeoffs
- Future research

Our proposal

Active formal models of the complete adaptation loop (MAPE-K)

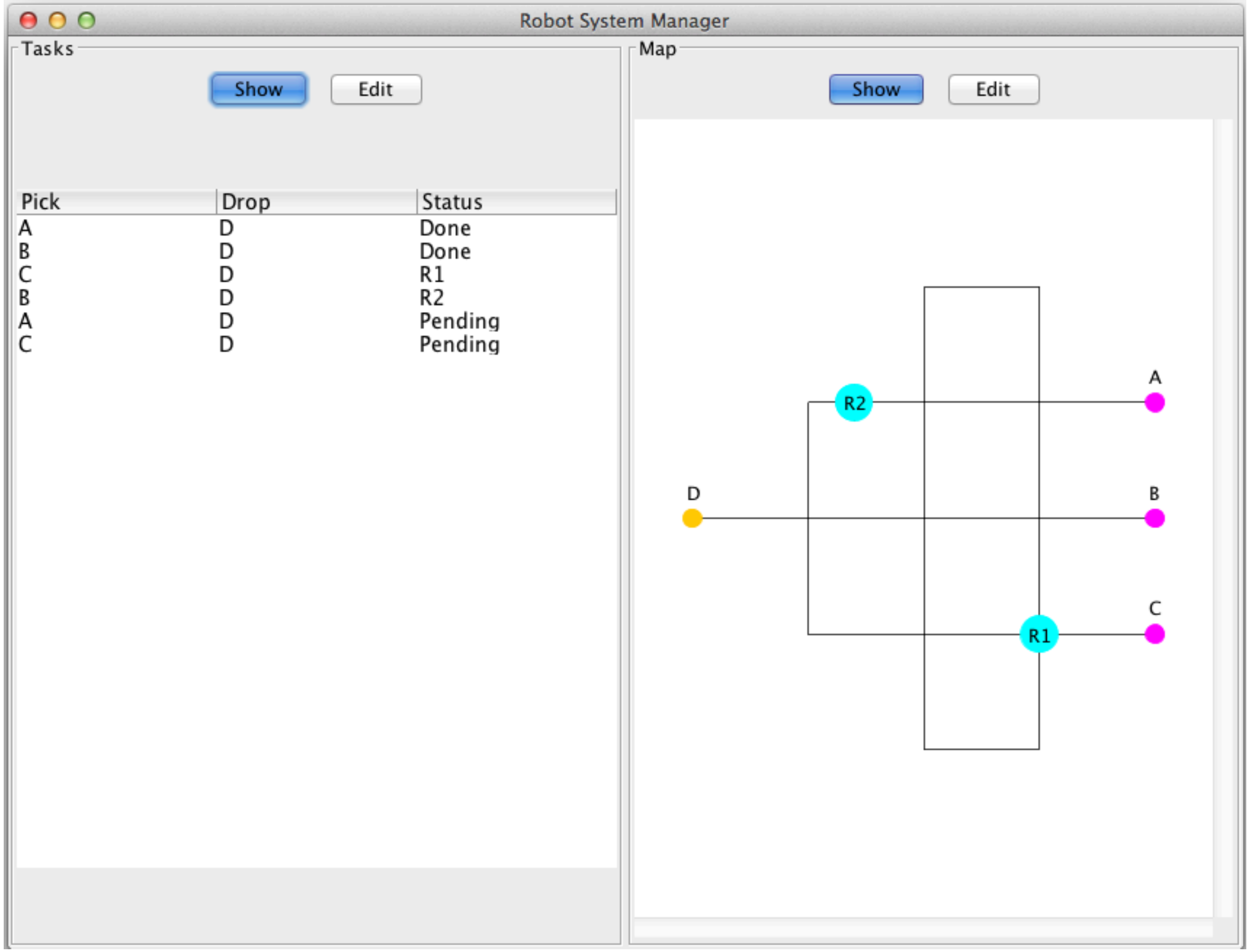
- Formal model is directly executed to adapt the managed system
- Runtime updates of formal model to support unanticipated changes

Focus



- 3 layered model of Kramer & Magee
  - Component control (layer 1), change management (2), goal management (3)
- Focus on layer 2 and 3
  - Assumption: managed system is equipped with required sensors and effectors
  - Instrumentation of managed system is research subject in its own right
- Case study: logistic multi-robot system

Case study

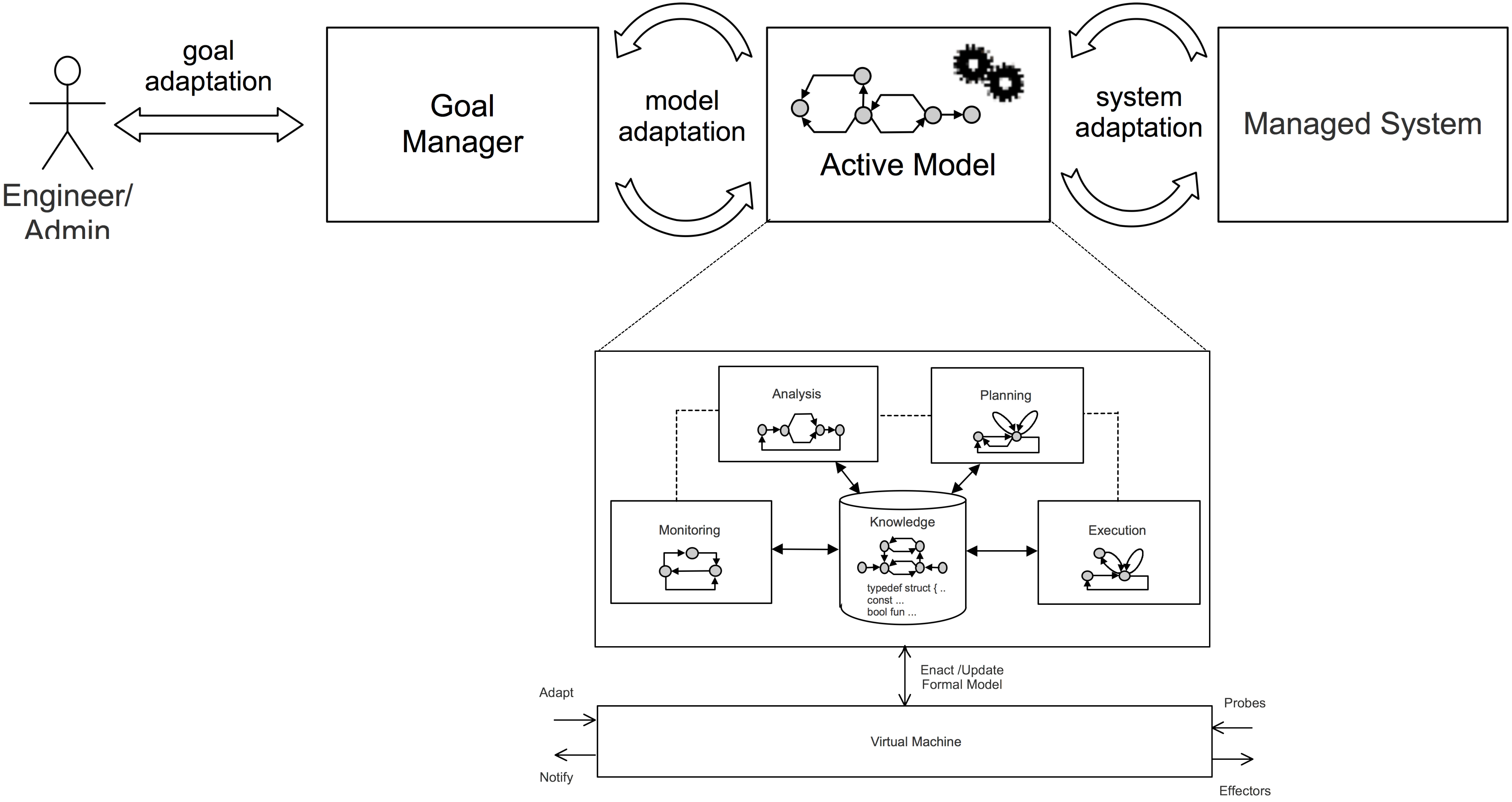


Pick	Drop	Status
A	D	Done
B	D	Done
C	D	R1
B	D	R2
A	D	Pending
C	D	Pending

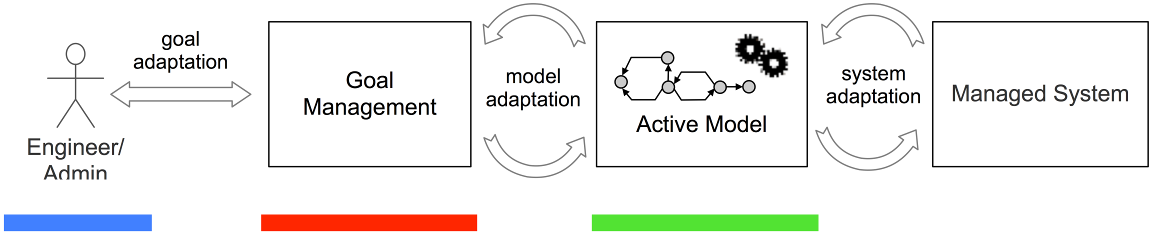
Overview

- State of the art (some key contributions)
- Our proposal
- Approach
- Realization
- Contributions & Tradeoffs
- Future research

Approach


goal
adaptation
Engineer/
Admin
Goal
Manager
model
adaptation
Active Model
system
adaptation
Managed System
Analysis
Planning
Monitoring
Knowledge
typedef struct { ..
const ...
bool fun ...
Execution
Enact /Update
Formal Model
Adapt
Notify
Virtual Machine
Probes
Effectors

Approach



- Active model
  - Is a formally verified model
  - Realizes a MAPE-K loop
  - To adapt the managed system
- Goal management
  - Monitors the active model
  - Can adapt the active model (e.g., to improve it or deal with a particular adaptation problem)
- Engineer/Admin
  - Can monitor goal satisfaction
  - Can change the active model, verify and deploy it, to manage (new) goals using goal management

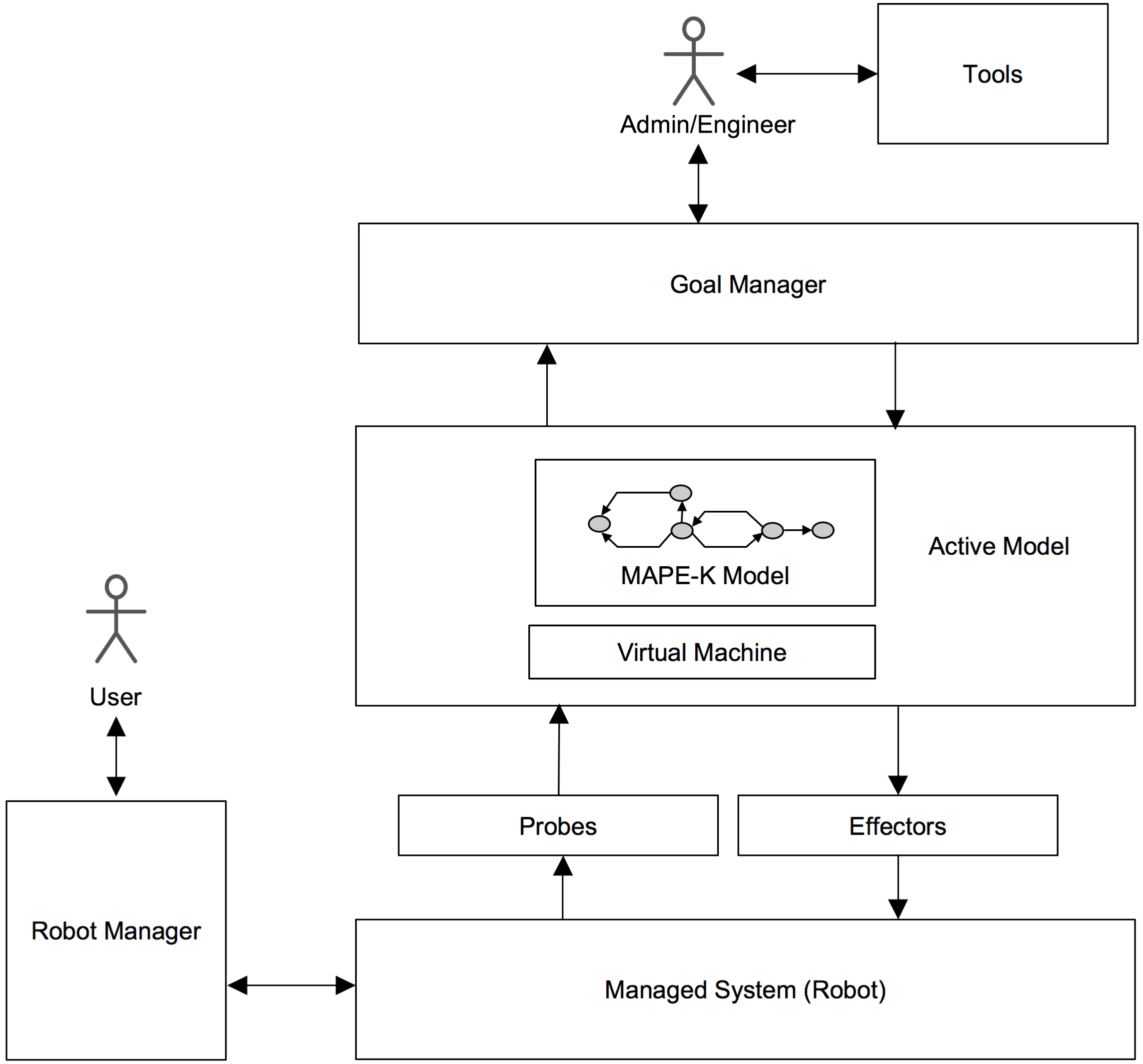
Levels of adaptation

- Level 1: active model adapts the managed system
  - Close temporally a lane in the warehouse for maintenance
- Level 2: adapt the active model (adapt MAPE)
  - Add a new drop location in the warehouse

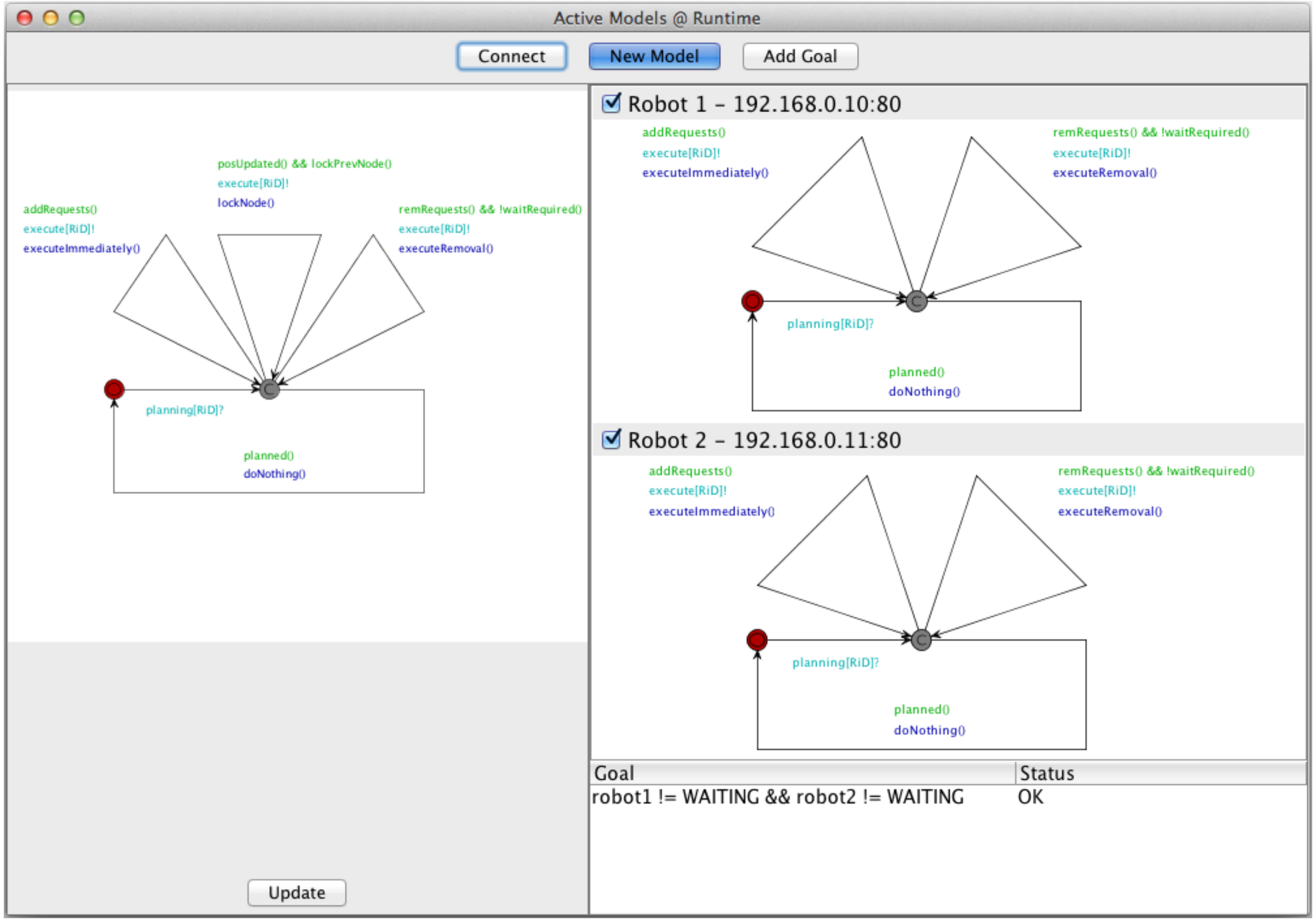
Overview

- State of the art (some key contributions)
- Our proposal
- Approach
- Realization
- Contributions & Tradeoffs
- Future research

Realization



Goal Management Interface



☑ Robot 1 – 192.168.0.10:80

☑ Robot 2 – 192.168.0.11:80

Goal	Status
robot1 != WAITING && robot2 != WAITING	OK

Virtual machine

- Transforms a formal model (network of timed automata) into a graph representation
- Executes that model
- Can adapt the current model at runtime
- Can detect and notify goal violations

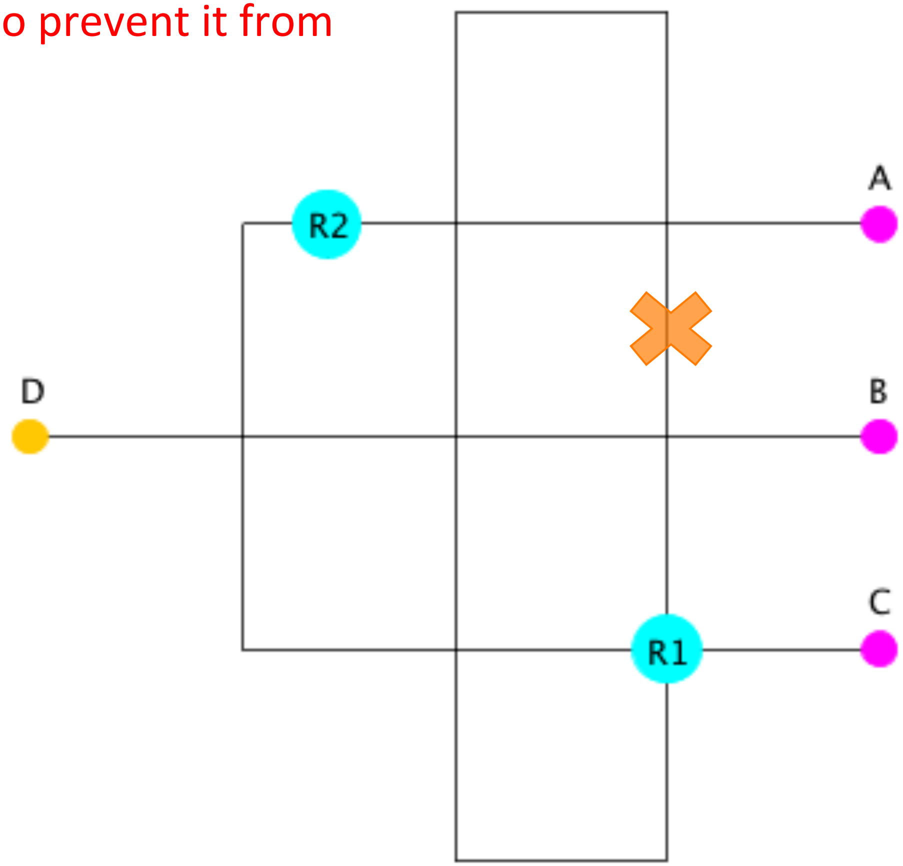
Level of adaptions

- Level 1: active model adapts managed system
- Level 2: adapt the active model

Level 1 adaptations

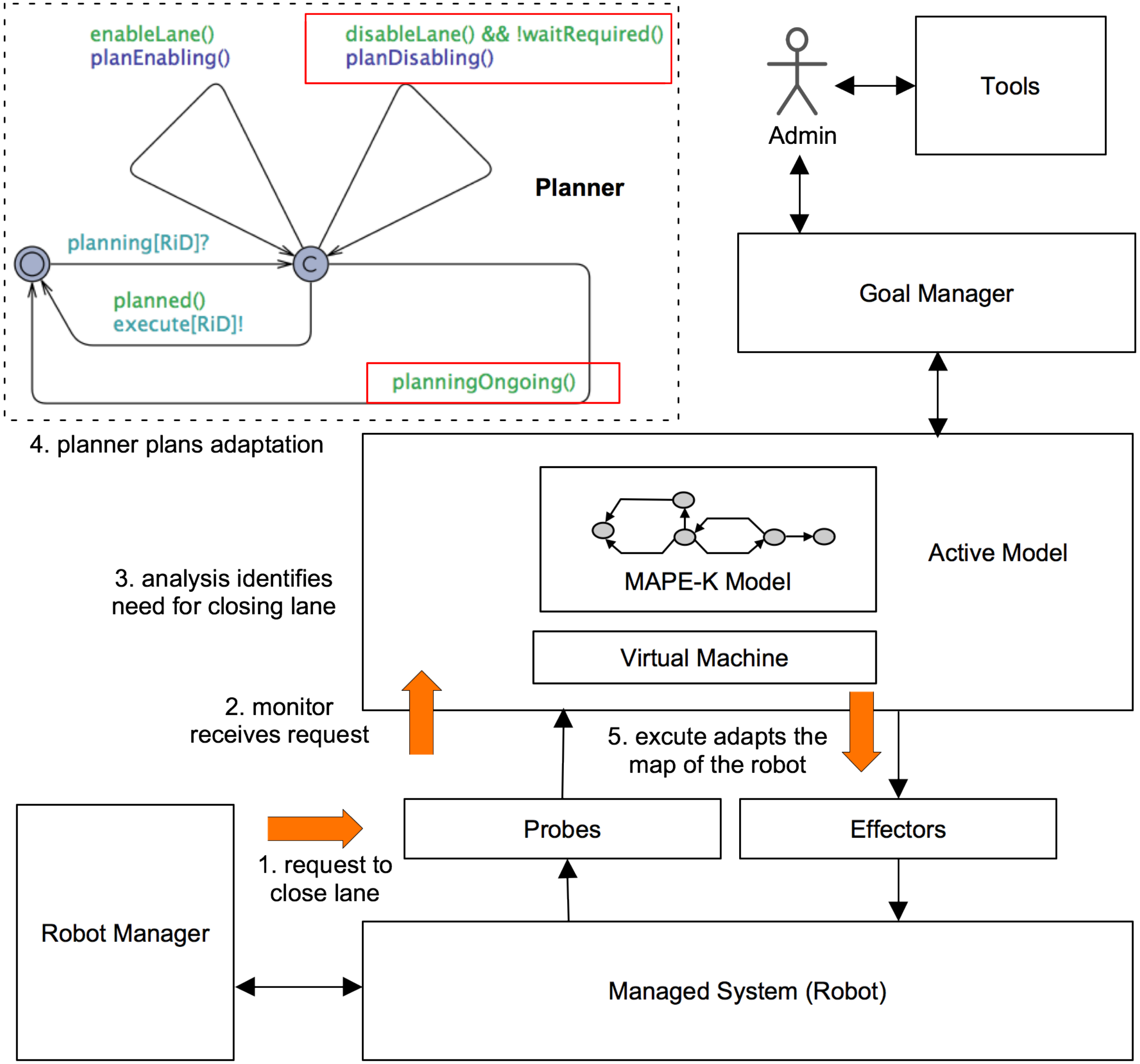
Close temporally a lane in the warehouse for maintenance

- Adapt the robot to prevent it from using a closed lane



Level 1 adaptations

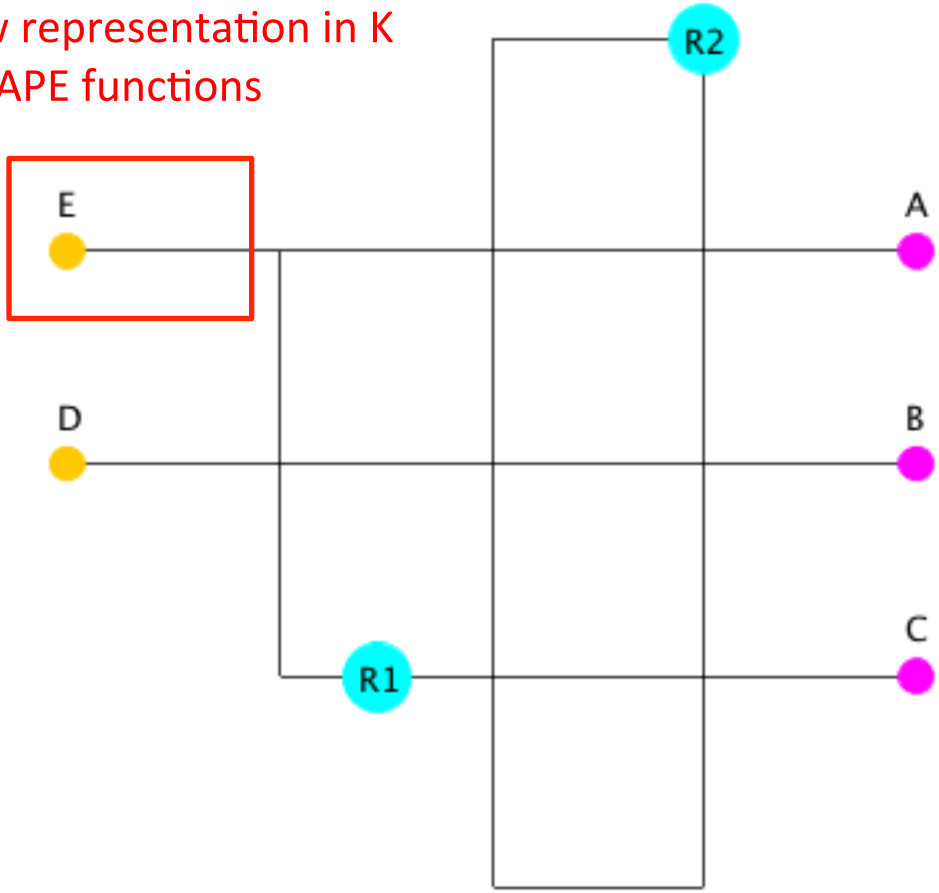
Close temporally a lane in the warehouse for maintenance



Level 2 adaptations

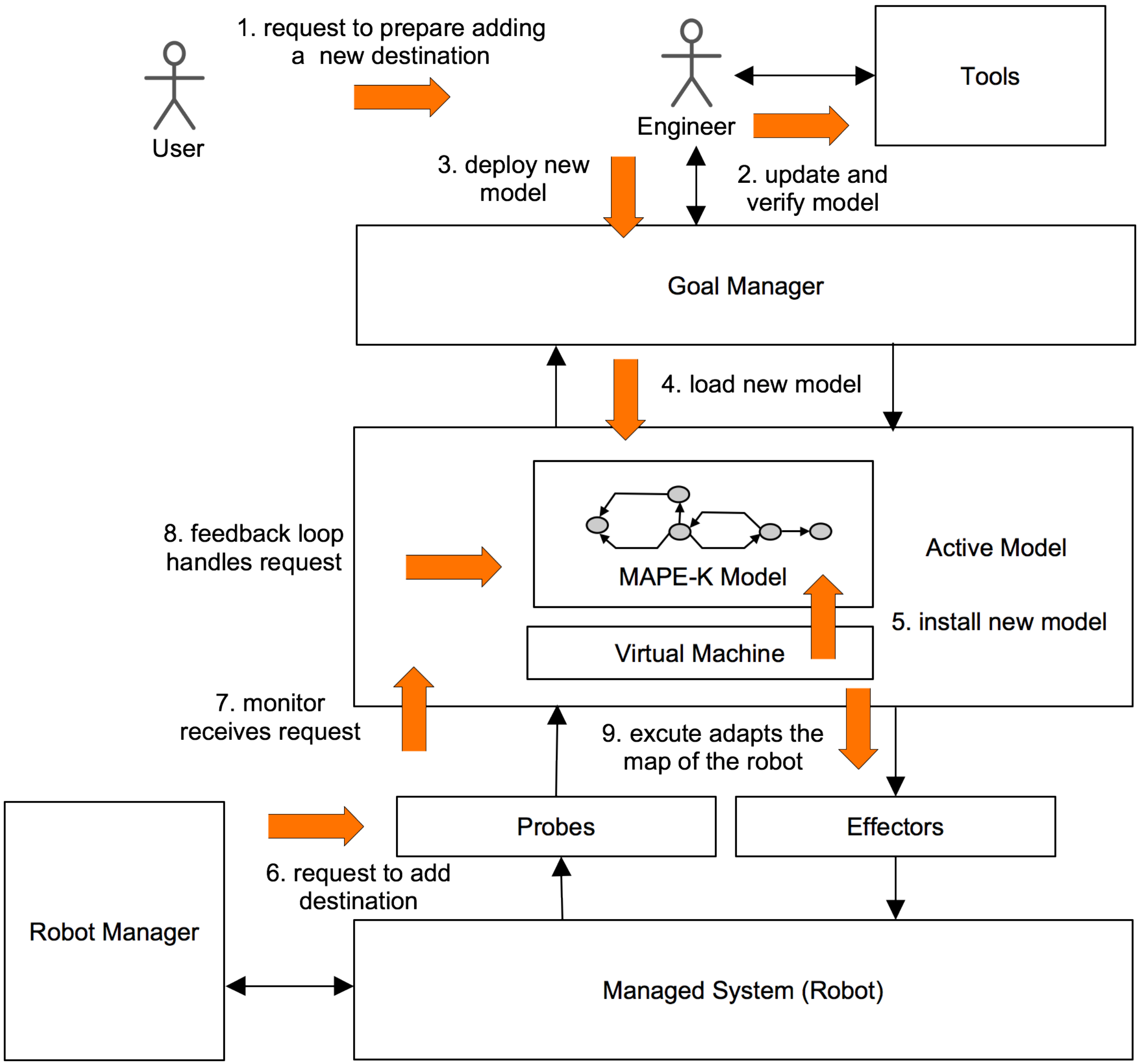
Add a new drop location in the warehouse

- Add new part of the map for the robot
- Creates new deadlock situations when certain lanes are disabled
- Requires adding new representation in K and adaptations of MAPE functions



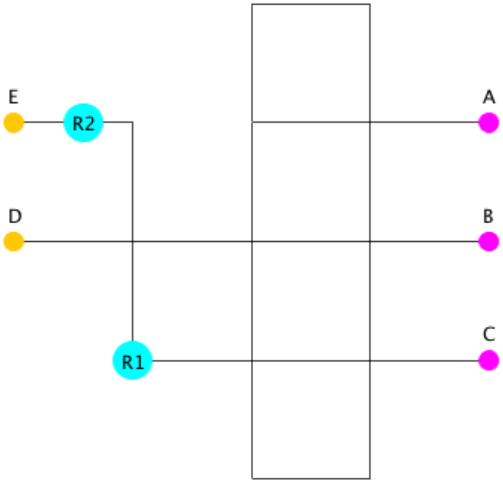
Level 2 adaptations

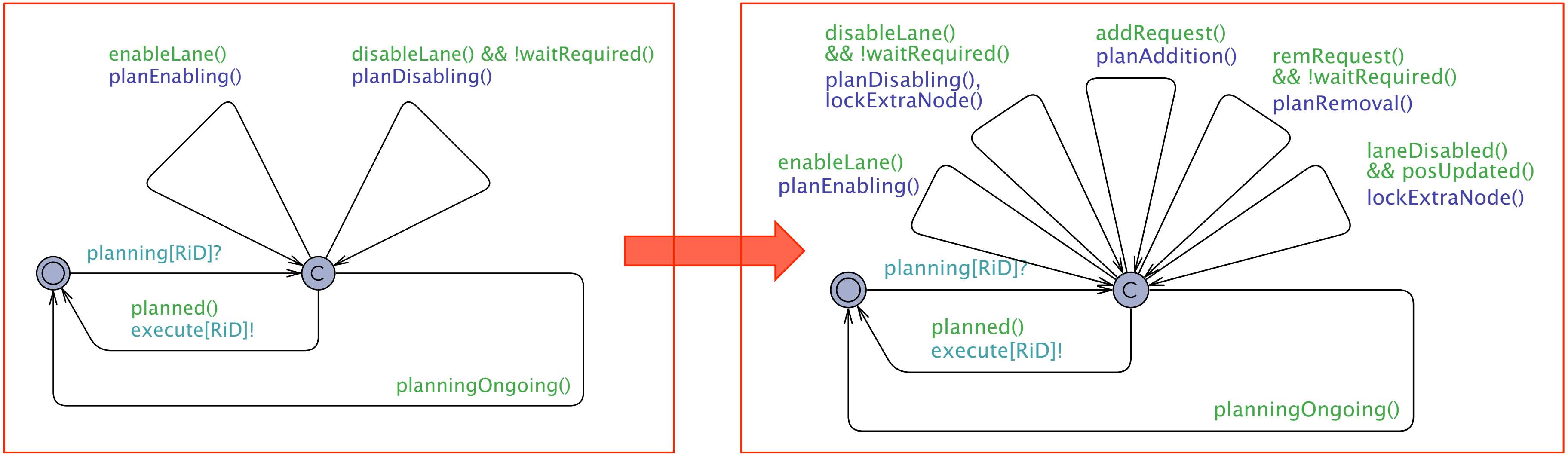
Add a new drop location in the warehouse


1. request to prepare adding a new destination
User
Engineer
Tools
2. update and verify model
3. deploy new model
Goal Manager
4. load new model
Active Model
MAPE-K Model
Virtual Machine
5. install new model
8. feedback loop handles request
7. monitor receives request
9. excute adapts the map of the robot
Probes
Effectors
6. request to add destination
Robot Manager
Managed System (Robot)

Level 2 adaptations

Deal with new deadlock threat (close additional lane): e.g., update planner





Overview

- State of the art (some key contributions)
- Our proposal
- Approach
- Realization
- Contributions & Tradeoffs
- Future research

Contributions

- Formal active model guarantees verified properties of the adaption process
- Active model directly executes the adaptation: no coding, no model transformations
- Adaptation of adaptation functions: lightweight process to add new goals
- Online detection of goal violations

Tradeoffs

- Expert knowledge to design and change the formal models
- We can only express what the modeling language supports
- Language might not be appropriate to model adaption logic for particular types of systems
- Possible performance overhead

Overview

- State of the art (some key contributions)
- Our proposal
- Approach
- Realization
- Contributions & Tradeoffs
- Future research

Paves the way for future research

- Domain specific design primitives to support the designer (Didac Gil de la Iglesia)
- Different modeling languages (e.g. probabilistic automata)
- Scalable runtime verification
- Coordination between Active Models in decentralized setting
- Automation goal management by learning

Thank you! -- The floor is open for questions & critical comments