Trustworthy Software Using Self-Adaptation

Danny Weyns

20th International Summer School on Trustworthy Software
9-12 July 2024, Shanghai, China

danny.weyns@gmail.com

https://people.cs.kuleuven.be/danny.weyns/

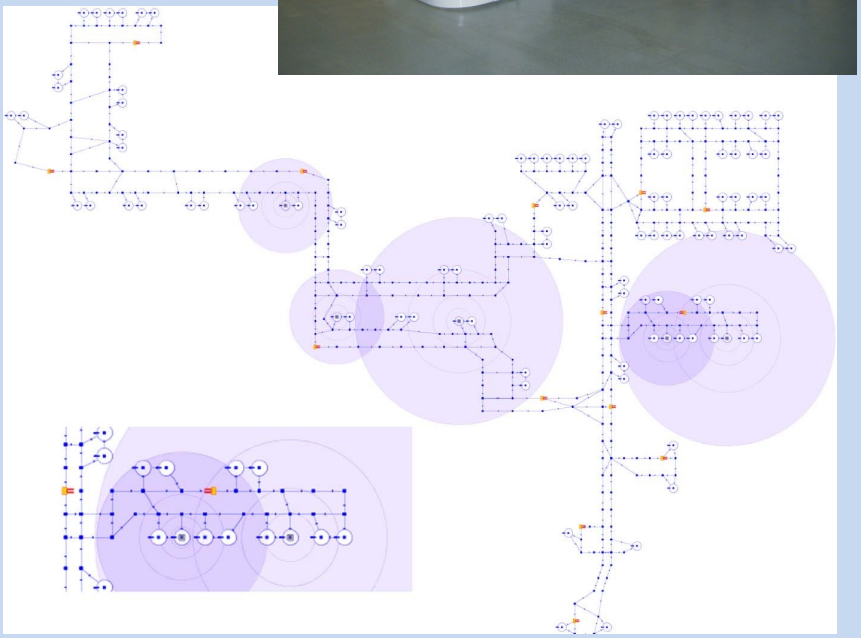
Your tutor for this course

Research context

Research context

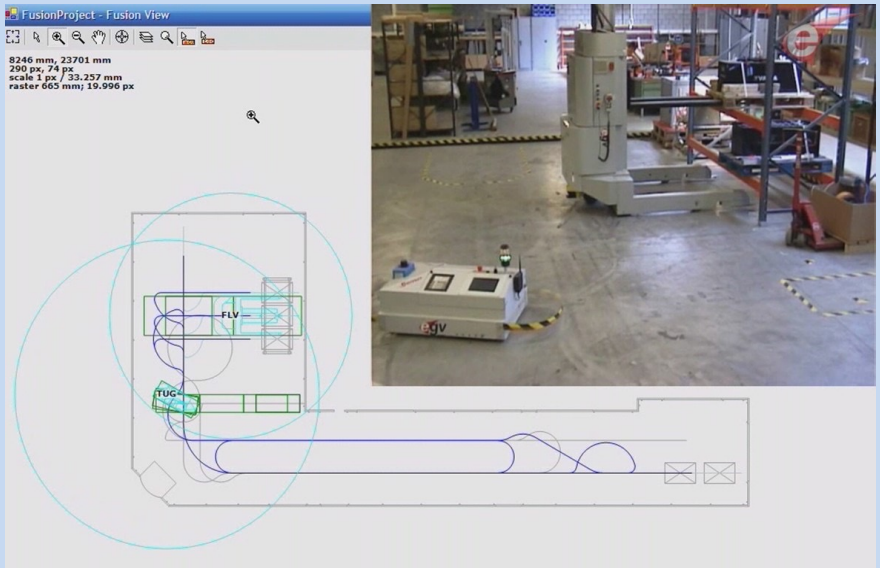
- Department of Computer Science, KU Leuven Belgium
    - Distributed and Secure Software research group: principles of self-adaptation, AI for self-adaptation
- Department of Computer Science and Media Technology, Linnaeus University, Sweden
    - AdaptWise research group: architectures for self-adaptation
- NORCE Research Center, Norway
    - DARWIN research group: ML and self-evolution

Research focus over time (by example)

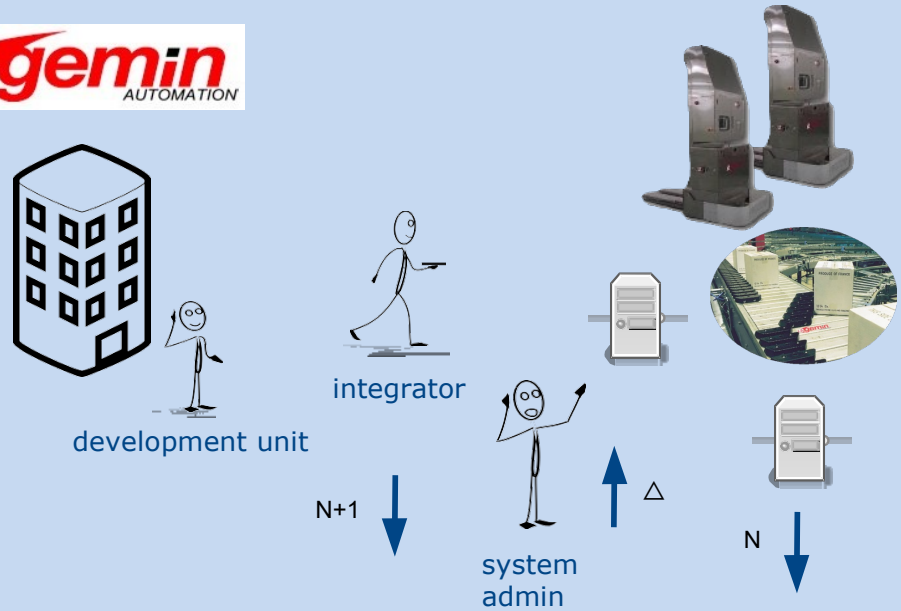


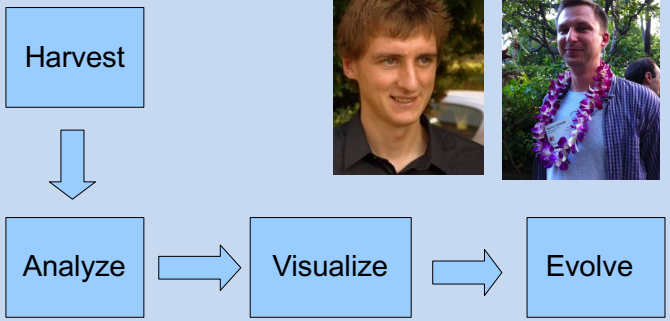
2002-2006

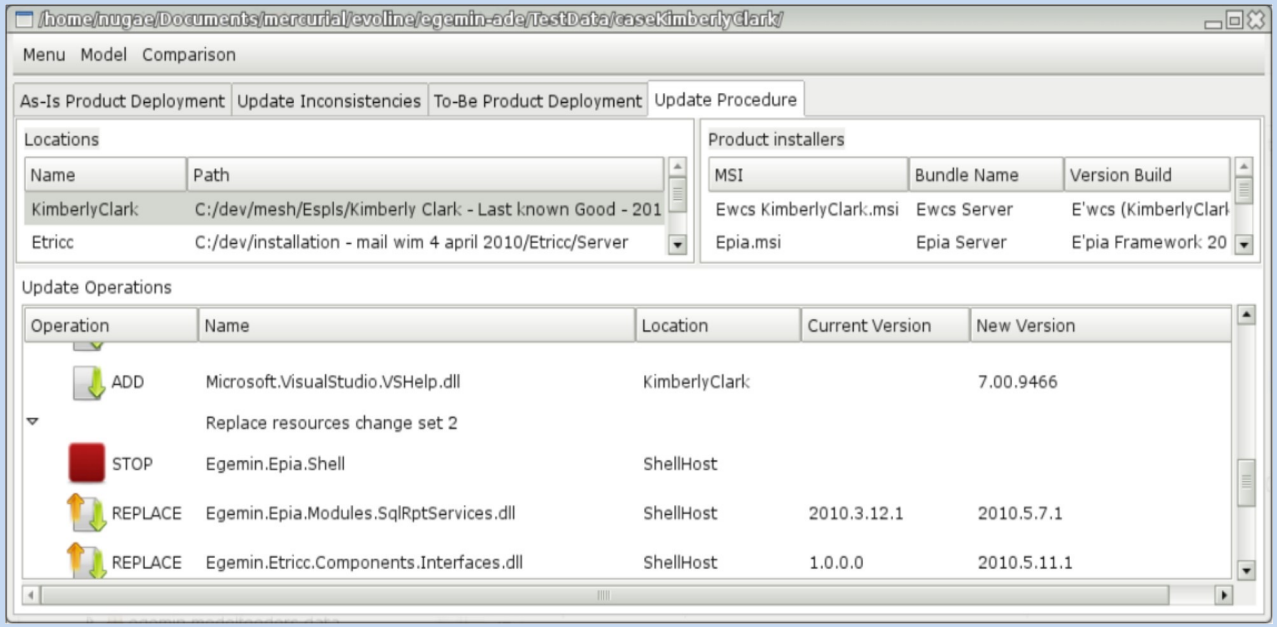
PhD KU Leuven



2006-2011
Postdoc KU Leuven

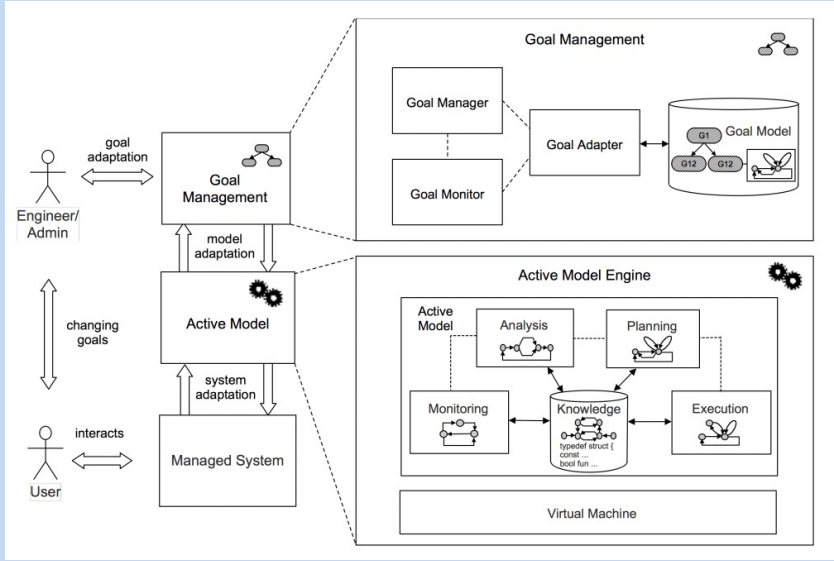


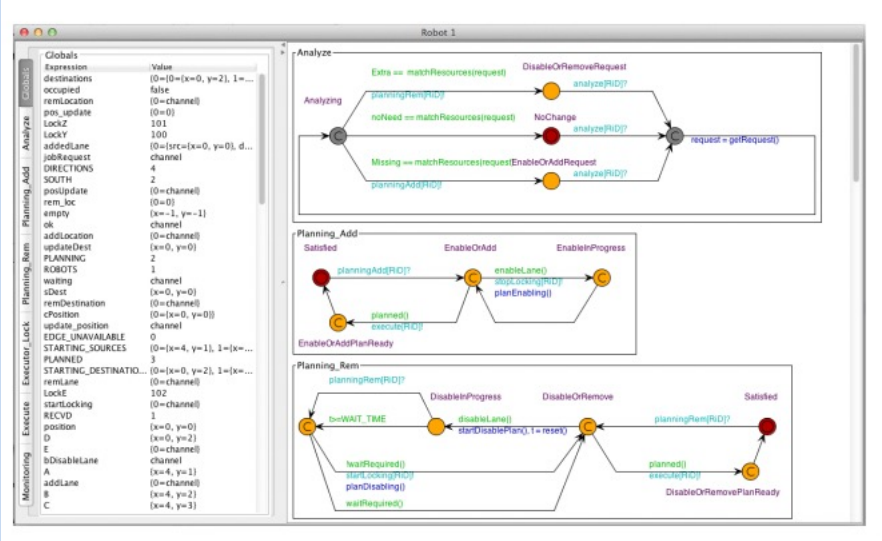


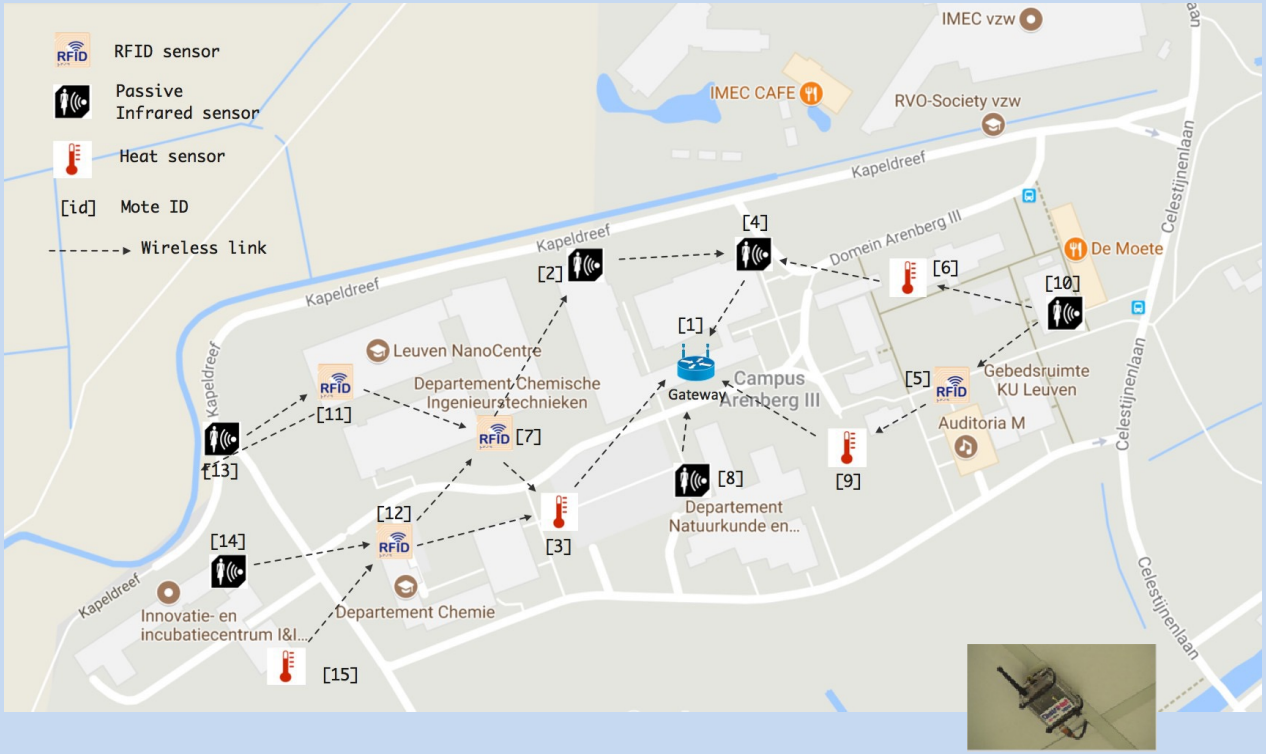


2011-2015

Linnaeus University Sweden

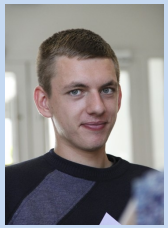
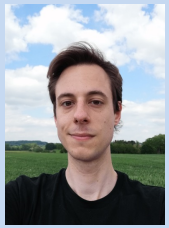


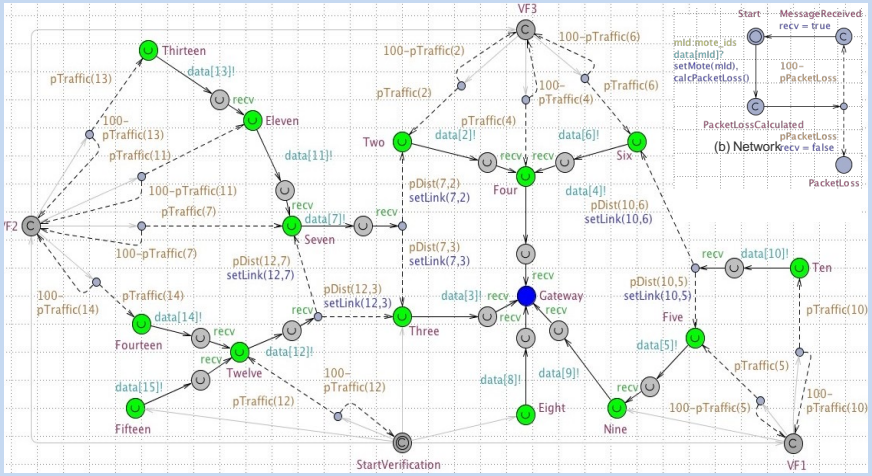


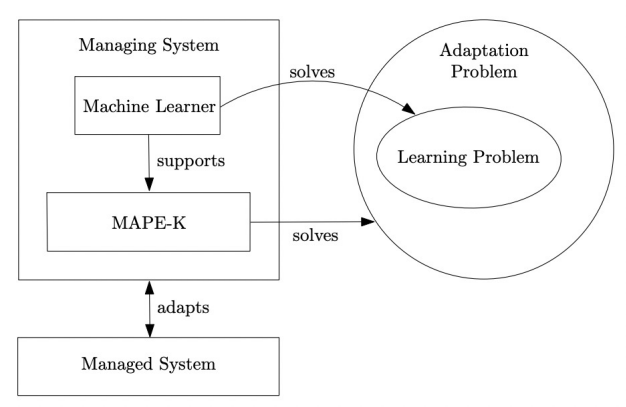


Since 2015

KU Leuven, NORCE and Linnaeus University





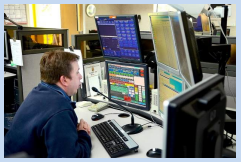
Goals of this course

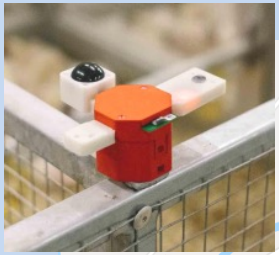
- Illustrate the need for self-adaptation
- Explain the basic principles of self-adaptation
- Understand the use of self-adaptation in industry
- Understand the waves of research on self-adaptation
- Explain how to guarantee that a self-adaptive system complies with its goals at runtime
- Understand some of the open challenges in the field of self-adaptation

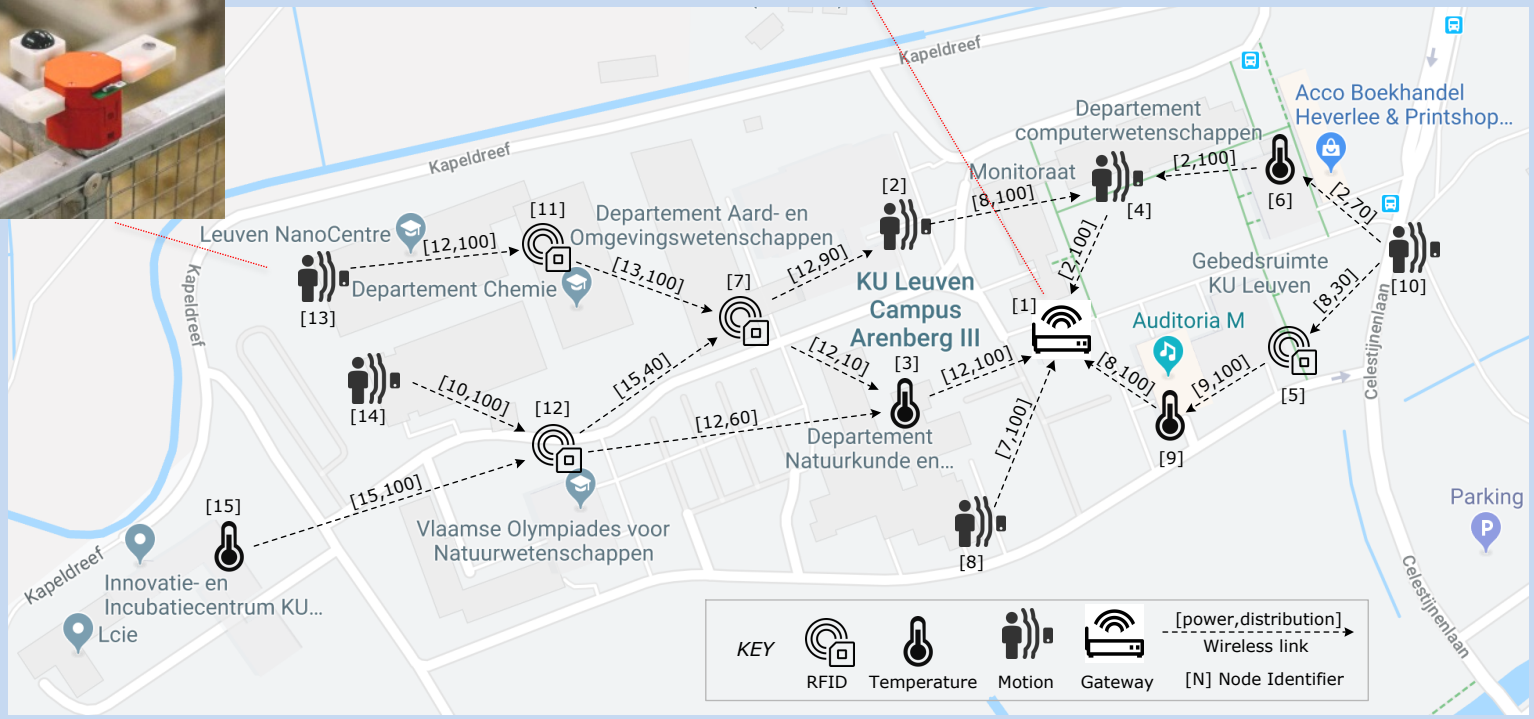
Overview

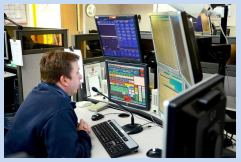
- Why self-adaptation?
- What is self-adaptation?
- Self-adaptation in industry
- How to engineer self-adaptive systems?
- Decision-making with formal techniques at runtime
- Why is “just self-adaptation” not enough?
- Conclusions

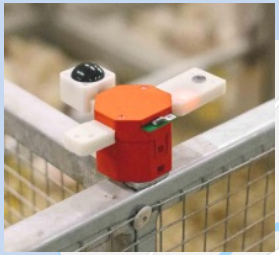
Why self-adaptation?

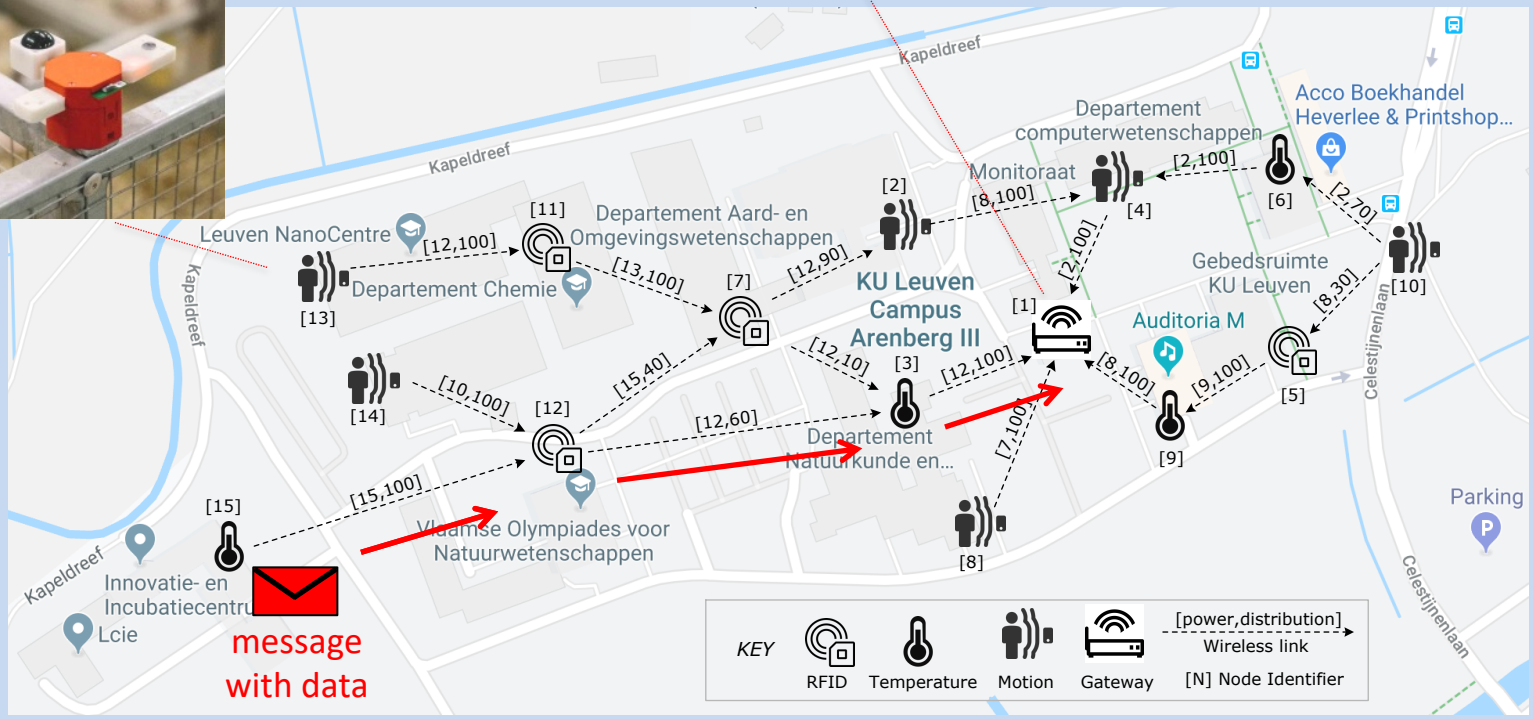


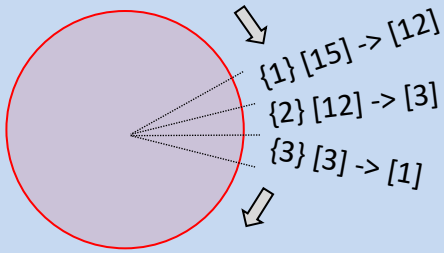


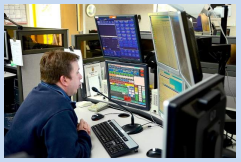

Kapeldreef
Kapeldreef
Kapeldreef
Kapeldreef
Leuven NanoCentre
Departement Chemie
Departement Aard- en Omgevingswetenschappen
KU Leuven Campus Arenberg III
Departement computerwetenschappen
Monitoraat
Acco Boekhandel Heverlee & Printshop...
Gebedsruimte KU Leuven
Auditoria M
Departement Natuurkunde en...
Vlaamse Olympiades voor Natuurwetenschappen
Innovatie- en Incubatiecentrum KU...
Lcie
Parking
Celestijnenlaan
Celestijnenlaan
[1]
[2]
[3]
[4]
[5]
[6]
[7]
[8]
[9]
[10]
[11]
[12]
[13]
[14]
[15]
[12,100]
[13,100]
[12,90]
[8,100]
[2,100]
[2,70]
[2,100]
[8,30]
[12,10]
[12,100]
[8,100]
[9,100]
[7,100]
[10,100]
[15,40]
[12,60]
[15,100]
KEY
RFID
Temperature
Motion
Gateway
[power,distribution]
Wireless link
[N] Node Identifier

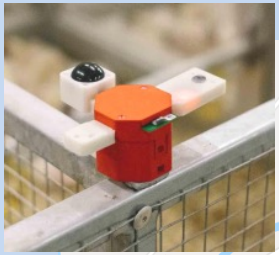


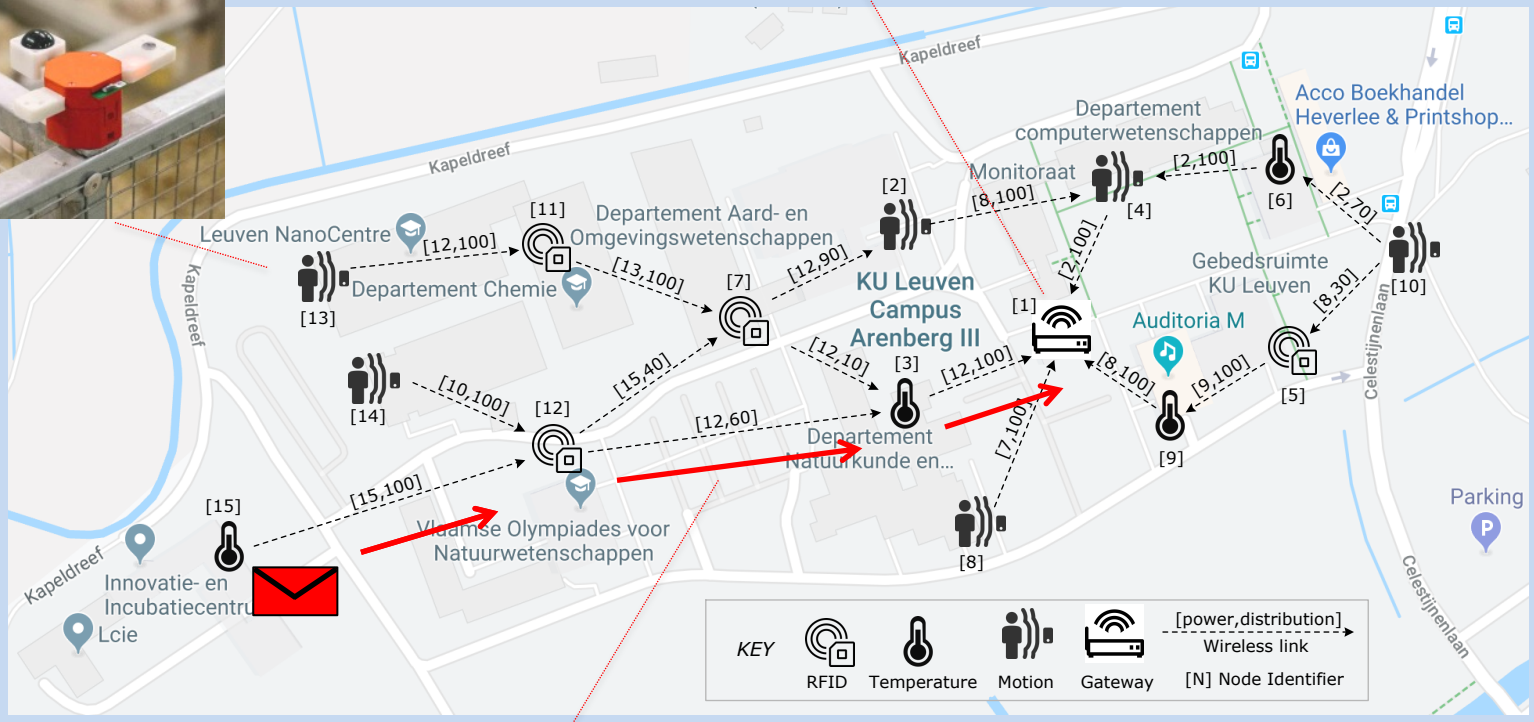


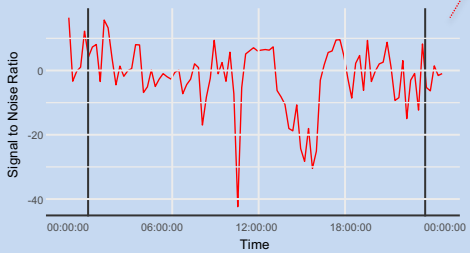

Kapeldreef
Kapeldreef
Kapeldreef
Kapeldreef
Leuven NanoCentre
Departement Chemie
Departement Aard- en Omgevingswetenschappen
KU Leuven Campus Arenberg III
Departement computerwetenschappen
Monitoraat
Acco Boekhandel Heverlee & Printshop...
Gebedsruimte KU Leuven
Auditoria M
Departement Natuurkunde en...
Vlaamse Olympiades voor Natuurwetenschappen
Innovatie- en Incubatiecentru
Lcie
Celestijnenlaan
Celestijnenlaan
Parking
[1]
[2]
[3]
[4]
[5]
[6]
[7]
[8]
[9]
[10]
[11]
[12]
[13]
[14]
[15]
[12,100]
[13,100]
[12,90]
[8,100]
[2,100]
[2,100]
[2,70]
[8,30]
[9,100]
[8,100]
[12,100]
[12,10]
[7,100]
[12,60]
[15,40]
[10,100]
[15,100]
message with data
KEY
RFID
Temperature
Motion
Gateway
[power,distribution]
Wireless link
[N] Node Identifier

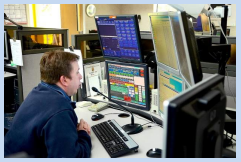

{1} [15] -> [12]
{2} [12] -> [3]
{3} [3] -> [1]

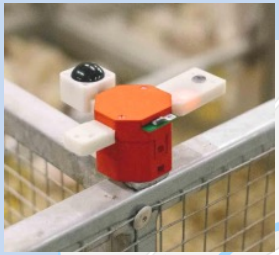


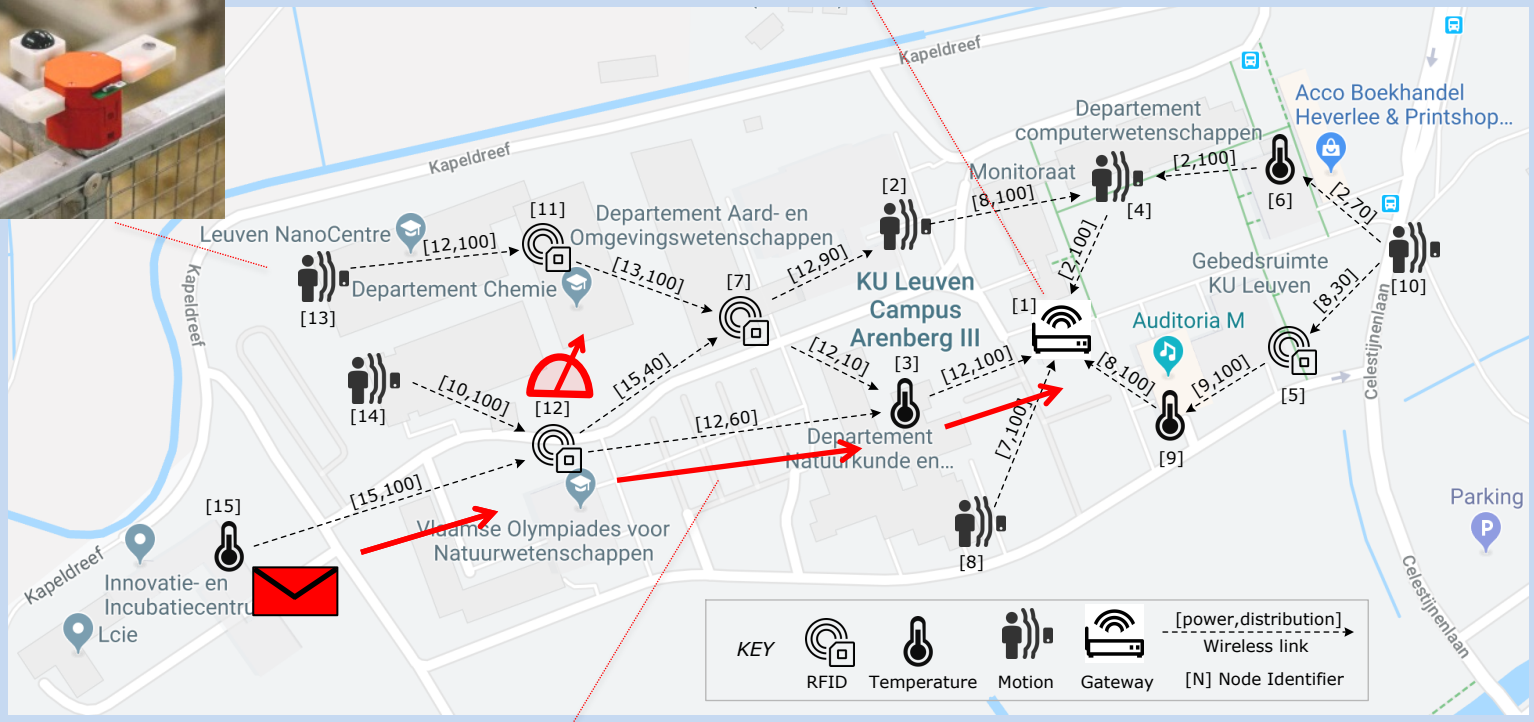


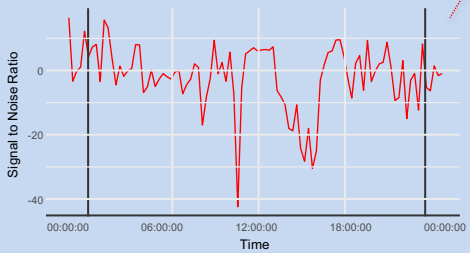

Kapeldreef
Kapeldreef
Kapeldreef
Kapeldreef
Leuven NanoCentre
Departement Chemie
Departement Aard- en Omgevingswetenschappen
KU Leuven Campus Arenberg III
Departement computerwetenschappen
Monitoraat
Acco Boekhandel Heverlee & Printshop...
Gebedsruimte KU Leuven
Auditoria M
Departement Natuurkunde en...
Vlaamse Olympiades voor Natuurwetenschappen
Innovatie- en Incubatiecentru
Lcie
Parking
Celestijnenlaan
Celestijnenlaan
[13]
[11]
[12,100]
[13,100]
[7]
[12,90]
[2]
[8,100]
[4]
[2,100]
[6]
[2,70]
[10]
[8,30]
[5]
[9,100]
[9]
[8,100]
[1]
[2,100]
[12,100]
[3]
[12,10]
[7,100]
[8]
[14]
[10,100]
[12]
[15,40]
[12,60]
[15]
[15,100]
KEY
RFID
Temperature
Motion
Gateway
[power,distribution]
Wireless link
[N] Node Identifier

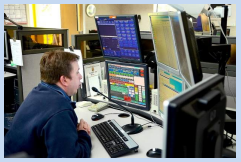

Signal to Noise Ratio
0
-20
-40
00:00:00
06:00:00
12:00:00
18:00:00
00:00:00
Time

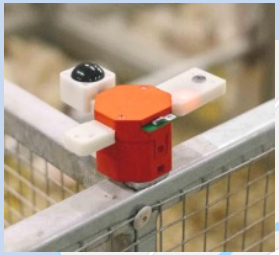


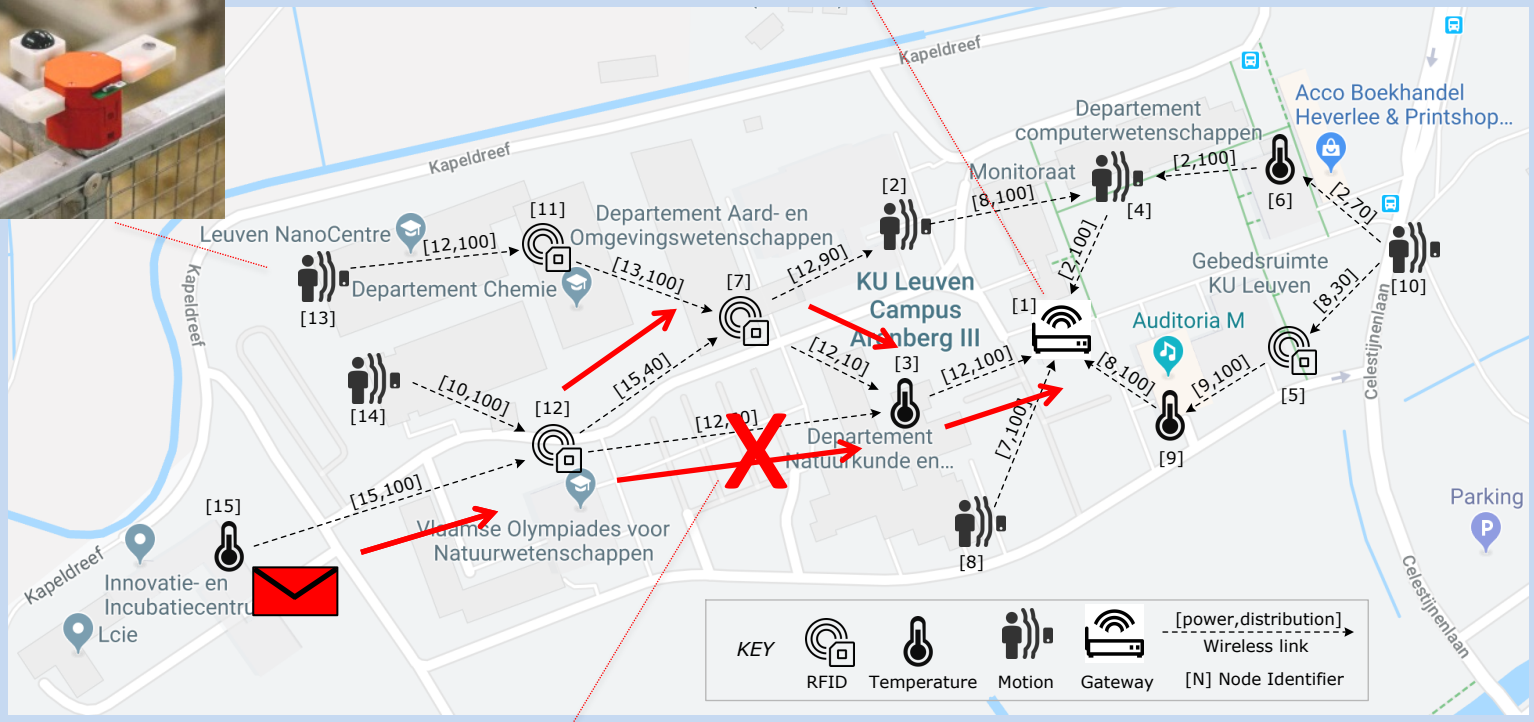


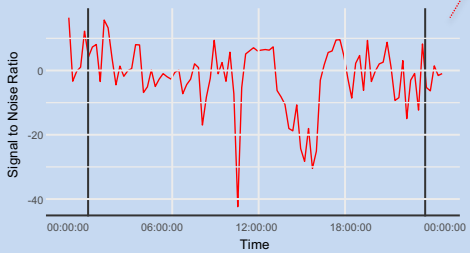

Kapeldreef
Kapeldreef
Kapeldreef
Kapeldreef
Leuven NanoCentre
Departement Chemie
Departement Aard- en Omgevingswetenschappen
KU Leuven Campus Arenberg III
Departement computerwetenschappen
Monitoraat
Acco Boekhandel Heverlee & Printshop...
Gebedsruimte KU Leuven
Auditoria M
Departement Natuurkunde en...
Vlaamse Olympiades voor Natuurwetenschappen
Innovatie- en Incubatiecentru
Lcie
Parking
Celestijnenlaan
Celestijnenlaan
[1]
[2]
[3]
[4]
[5]
[6]
[7]
[8]
[9]
[10]
[11]
[12]
[13]
[14]
[15]
[12,100]
[13,100]
[12,90]
[8,100]
[2,100]
[2,70]
[2,100]
[8,30]
[12,10]
[12,100]
[8,100]
[9,100]
[7,100]
[12,60]
[15,40]
[10,100]
[15,100]
KEY
RFID
Temperature
Motion
Gateway
[power,distribution]
Wireless link
[N] Node Identifier

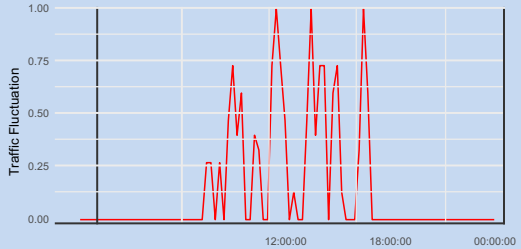

Signal to Noise Ratio
0
-20
-40
00:00:00
06:00:00
12:00:00
18:00:00
00:00:00
Time

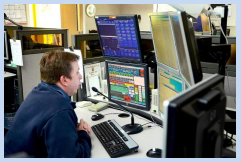


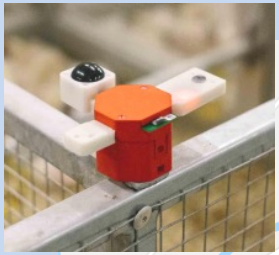


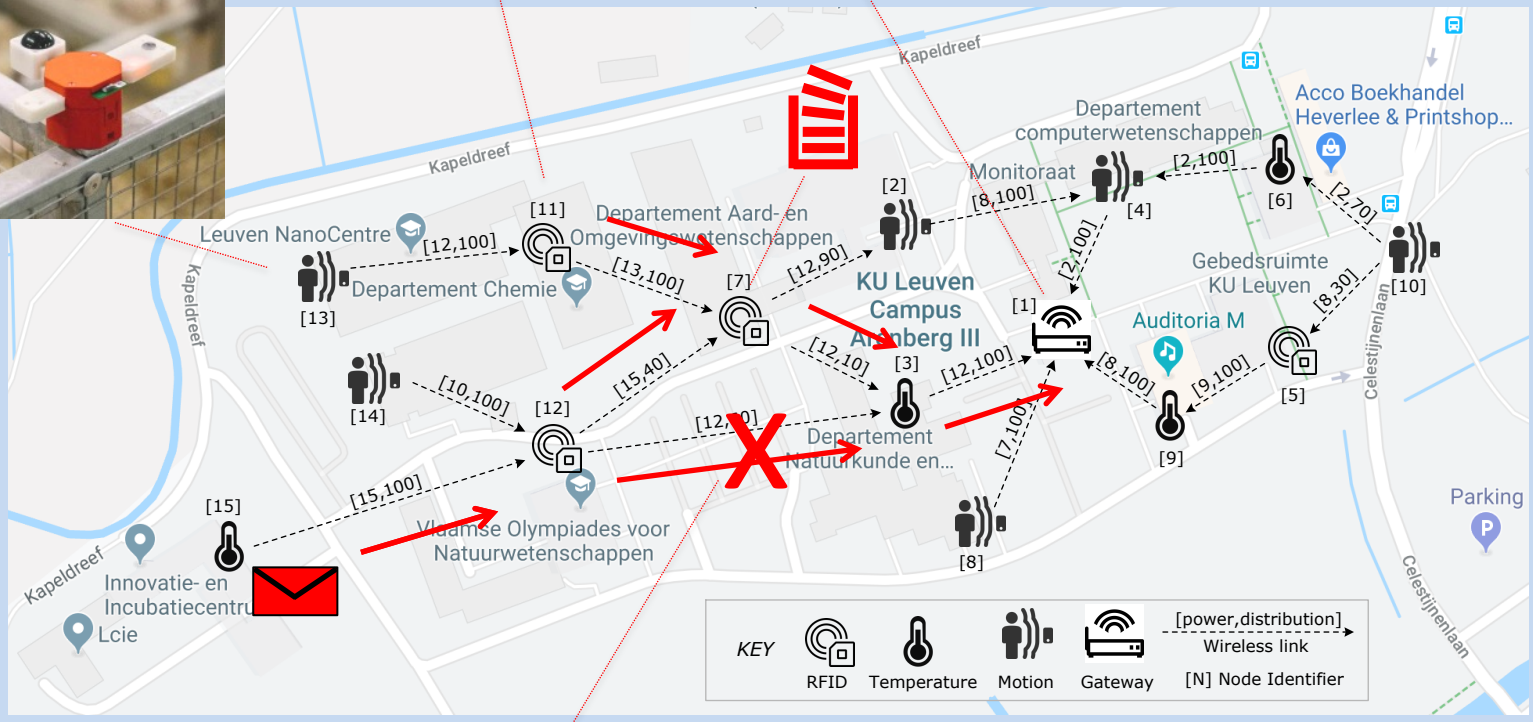

Kapeldreef
Kapeldreef
Kapeldreef
Kapeldreef
Leuven NanoCentre
Departement Chemie
Departement Aard- en Omgevingswetenschappen
KU Leuven Campus Arenberg III
Departement computerwetenschappen
Monitoraat
Acco Boekhandel Heverlee & Printshop...
Gebedsruimte KU Leuven
Auditoria M
Departement Natuurkunde en...
Vlaamse Olympiades voor Natuurwetenschappen
Innovatie- en Incubatiecentru
Lcie
Parking
Celestijnenlaan
Celestijnenlaan
[1]
[2]
[3]
[4]
[5]
[6]
[7]
[8]
[9]
[10]
[11]
[12]
[13]
[14]
[15]
[12,100]
[13,100]
[12,90]
[8,100]
[2,100]
[2,70]
[8,30]
[9,100]
[8,100]
[12,100]
[12,10]
[12,90]
[15,40]
[10,100]
[15,100]
[7,100]
KEY
RFID
Temperature
Motion
Gateway
[power,distribution]
Wireless link
[N] Node Identifier

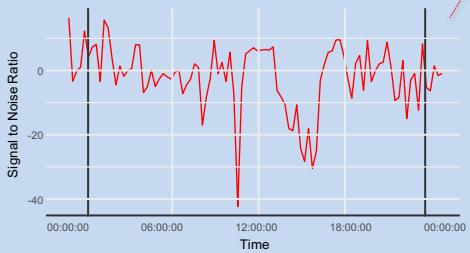

Signal to Noise Ratio
0
-20
-40
00:00:00
06:00:00
12:00:00
18:00:00
00:00:00
Time

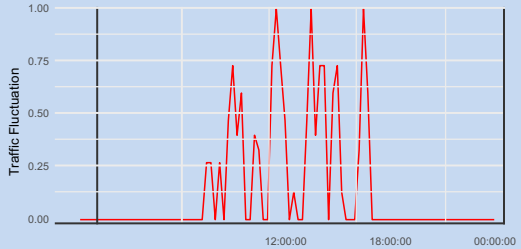

Traffic Fluctuation
1.00
0.75
0.50
0.25
0.00
12:00:00
18:00:00
00:00:00

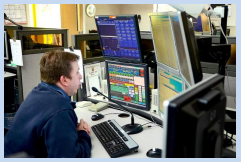


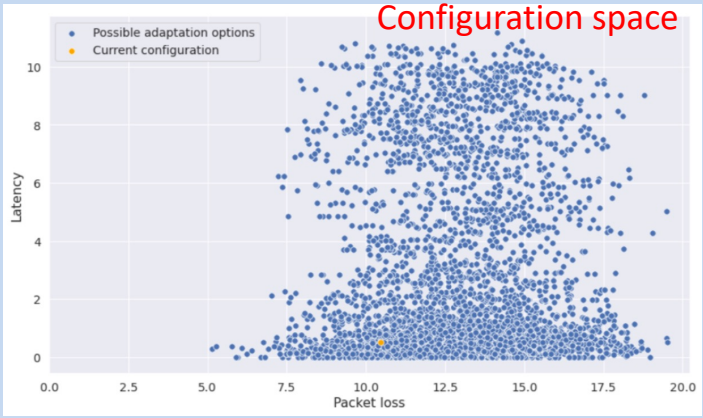


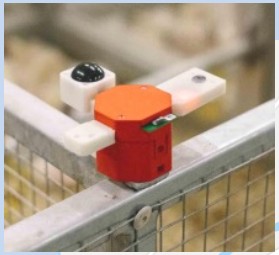

Kapeldreef
Leuven NanoCentre
Departement Chemie
Departement Aard- en Omgevingswetenschappen
KU Leuven Campus Arenberg III
Departement computerwetenschappen
Monitoraat
Acco Boekhandel Heverlee & Printshop...
Gebedsruimte KU Leuven
Auditoria M
Departement Natuurkunde en...
Vlaamse Olympiades voor Natuurwetenschappen
Innovatie- en Incubatiecentru
Lcie
Celestijnenlaan
Parking
[1]
[2]
[3]
[4]
[5]
[6]
[7]
[8]
[9]
[10]
[11]
[12]
[13]
[14]
[15]
[12,100]
[13,100]
[12,90]
[8,100]
[2,100]
[2,70]
[8,30]
[9,100]
[12,100]
[12,10]
[15,40]
[10,100]
[15,100]
[7,100]
KEY
RFID
Temperature
Motion
Gateway
[power,distribution]
Wireless link
[N] Node Identifier

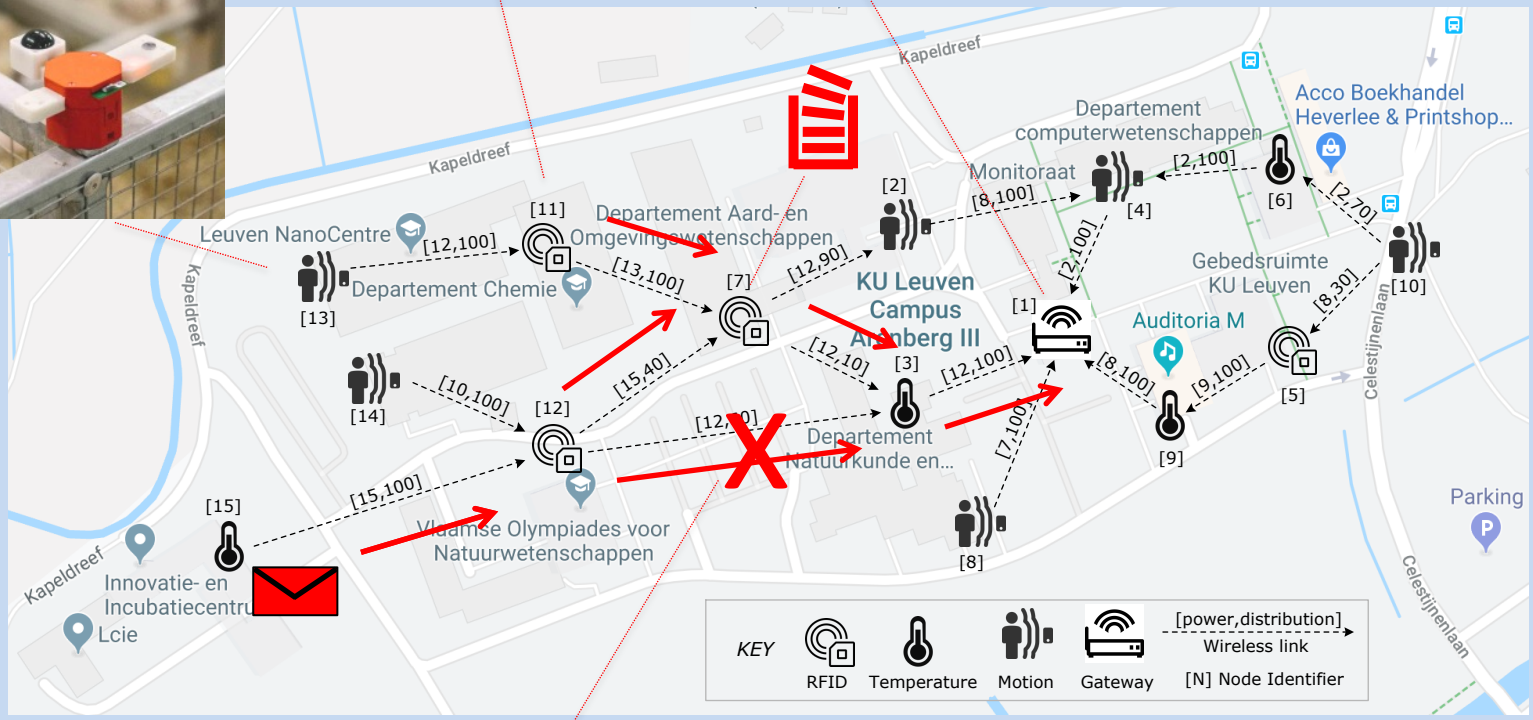

Signal to Noise Ratio
0
-20
-40
00:00:00
06:00:00
12:00:00
18:00:00
00:00:00
Time

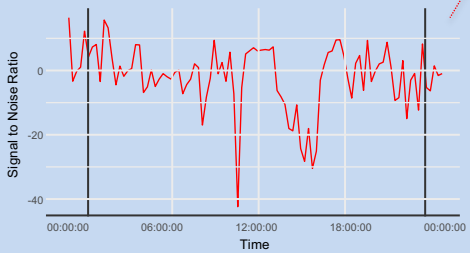

Traffic Fluctuation
1.00
0.75
0.50
0.25
0.00
12:00:00
18:00:00
00:00:00

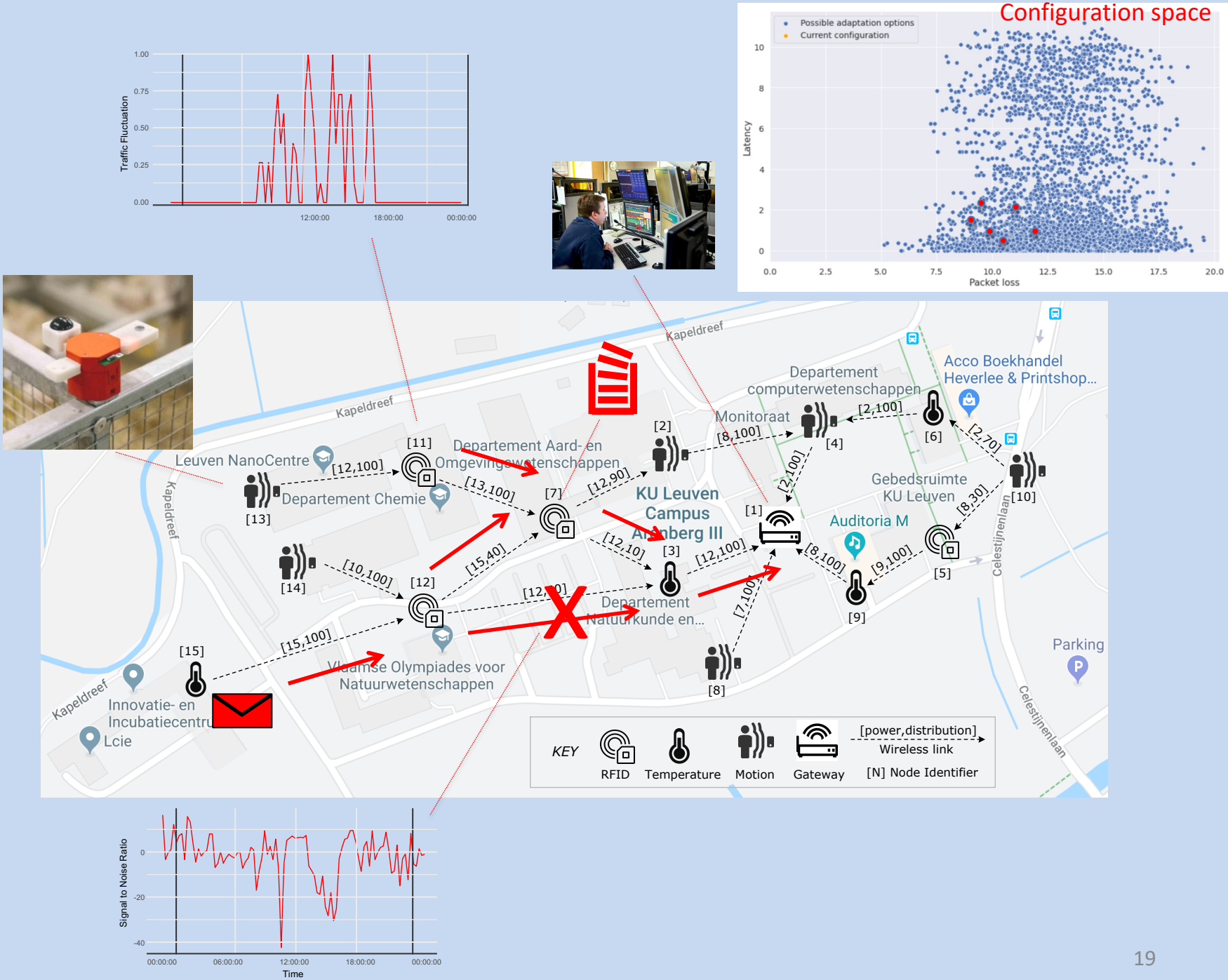


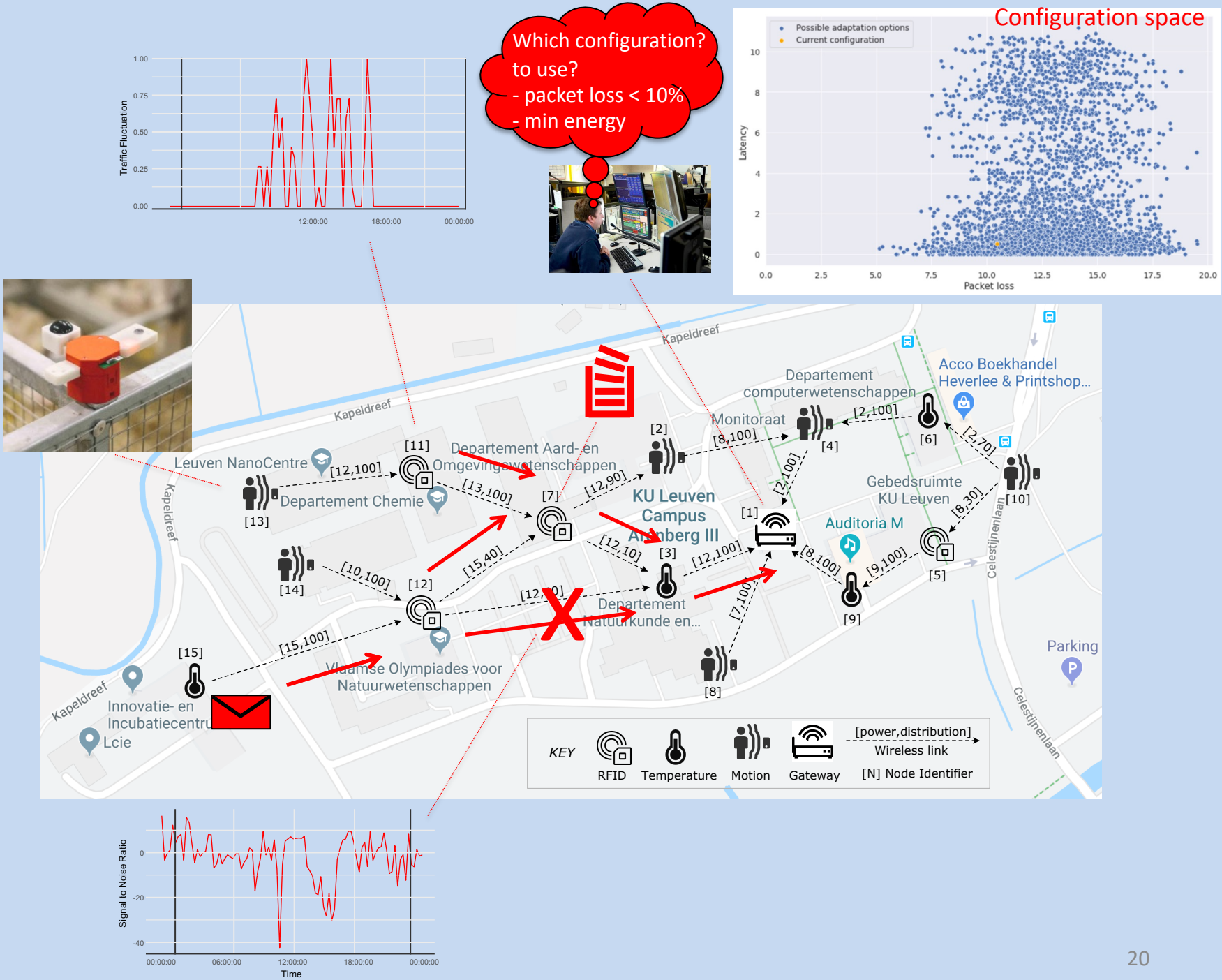

Configuration space
Possible adaptation options
Current configuration
Latency
Packet loss
0
2
4
6
8
10
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0



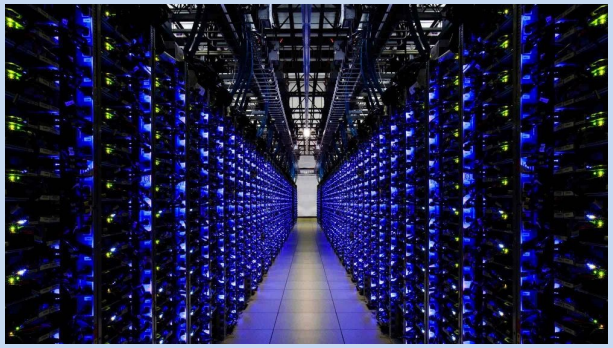

Kapeldreef
Leuven NanoCentre
Departement Chemie
Departement Aard- en Omgevingswetenschappen
KU Leuven Campus Arenberg III
Departement computerwetenschappen
Monitoraat
Acco Boekhandel Heverlee & Printshop...
Gebedsruimte KU Leuven
Auditoria M
Departement Natuurkunde en...
Vlaamse Olympiades voor Natuurwetenschappen
Innovatie- en Incubatiecentru
Lcie
Celestijnenlaan
Parking
[1]
[2]
[3]
[4]
[5]
[6]
[7]
[8]
[9]
[10]
[11]
[12]
[13]
[14]
[15]
[12,100]
[13,100]
[12,90]
[8,100]
[2,100]
[2,70]
[8,30]
[9,100]
[12,100]
[12,10]
[15,40]
[10,100]
[15,100]
[7,100]
KEY
RFID
Temperature
Motion
Gateway
[power,distribution]
Wireless link
[N] Node Identifier


Signal to Noise Ratio
0
-20
-40
00:00:00
06:00:00
12:00:00
18:00:00
00:00:00
Time


Configuration space
Possible adaptation options
Current configuration
Latency
Packet loss
Traffic Fluctuation
Signal to Noise Ratio
Time
Kapeldreef
Leuven NanoCentre
Departement Chemie
Departement Aard- en Omgevingswetenschappen
KU Leuven Campus Arenberg III
Departement Natuurkunde en...
Vlaamse Olympiades voor Natuurwetenschappen
Innovatie- en Incubatiecentrum Lcie
Departement computerwetenschappen
Monitoraat
Acco Boekhandel Heverlee & Printshop...
Gebedsruimte KU Leuven
Auditoria M
Celestijnenlaan
Parking
KEY
RFID
Temperature
Motion
Gateway
[power,distribution]
Wireless link
[N] Node Identifier


Configuration space
Possible adaptation options
Current configuration
Latency
Packet loss
Traffic Fluctuation
Time
Which configuration?
to use?
- packet loss < 10%
- min energy
Signal to Noise Ratio
Time
Kapeldreef
Leuven NanoCentre
Departement Chemie
Departement Aard- en Omgevingswetenschappen
KU Leuven Campus Arenberg III
Departement Natuurkunde en...
Vlaamse Olympiades voor Natuurwetenschappen
Innovatie- en Incubatiecentrum Lcie
Departement computerwetenschappen
Monitoraat
Acco Boekhandel Heverlee & Printshop...
Gebedsruimte KU Leuven
Auditoria M
Celestijnenlaan
Parking
KEY
RFID
Temperature
Motion
Gateway
[power,distribution]
Wireless link
[N] Node Identifier

Management of Computing Systems



Challenges

Complexity of systems

Uncertain operating conditions, 24/7

Implications

System qualities may degrade or worse

Manual approach: costly, prone to error, not timely

Problem

How to engineer systems such that they achieve their qualities and are trustworthy despite the uncertainties they face?

Overview

- Why self-adaptation?
- What is self-adaptation?
- Self-adaptation in industry
- How to engineer self-adaptive systems?
- Decision-making with formal techniques at runtime
- Why is “just self-adaptation” not enough?
- Conclusions

What is self-adaptation?

Self-adaptation is one prominent approach to deal with uncertainty and business continuity

Key idea: let system gather new knowledge at runtime to resolve uncertainties, reason about itself, its context and goals, and adapt to realize goals

Two principles of self-adaptation

Two principles of self-adaptation

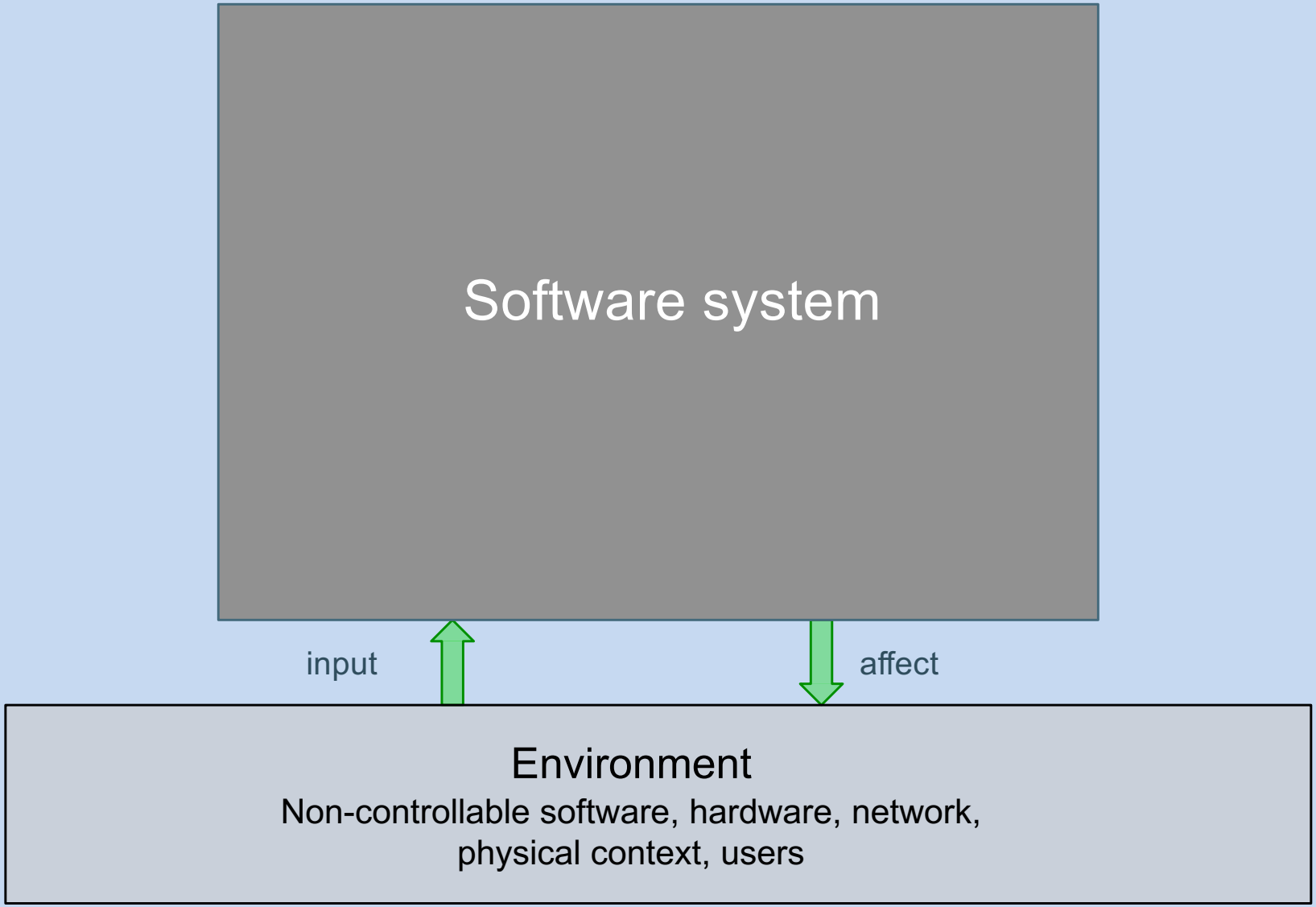
- External principle:
  - A self-adaptive system is a system that can handle uncertainty in its environment, itself and its goals autonomously (or with minimal human interference)

Two principles of self-adaptation

- External principle:
  - A self-adaptive system is a system that can handle uncertainty in its environment, itself and its goals autonomously (or with minimal or required human intervention)

- Internal principle
  - A self-adaptive system consits of two distinct parts:
    - First part interacts with the environment and has domain concerns;
    - Second part interacts with the first part and has adaptation concerns, i.e., concerns about the domain concerns

Conceptual model self-adaptive system



Conceptual model self-adaptive system

operate

Software system

uncertainties

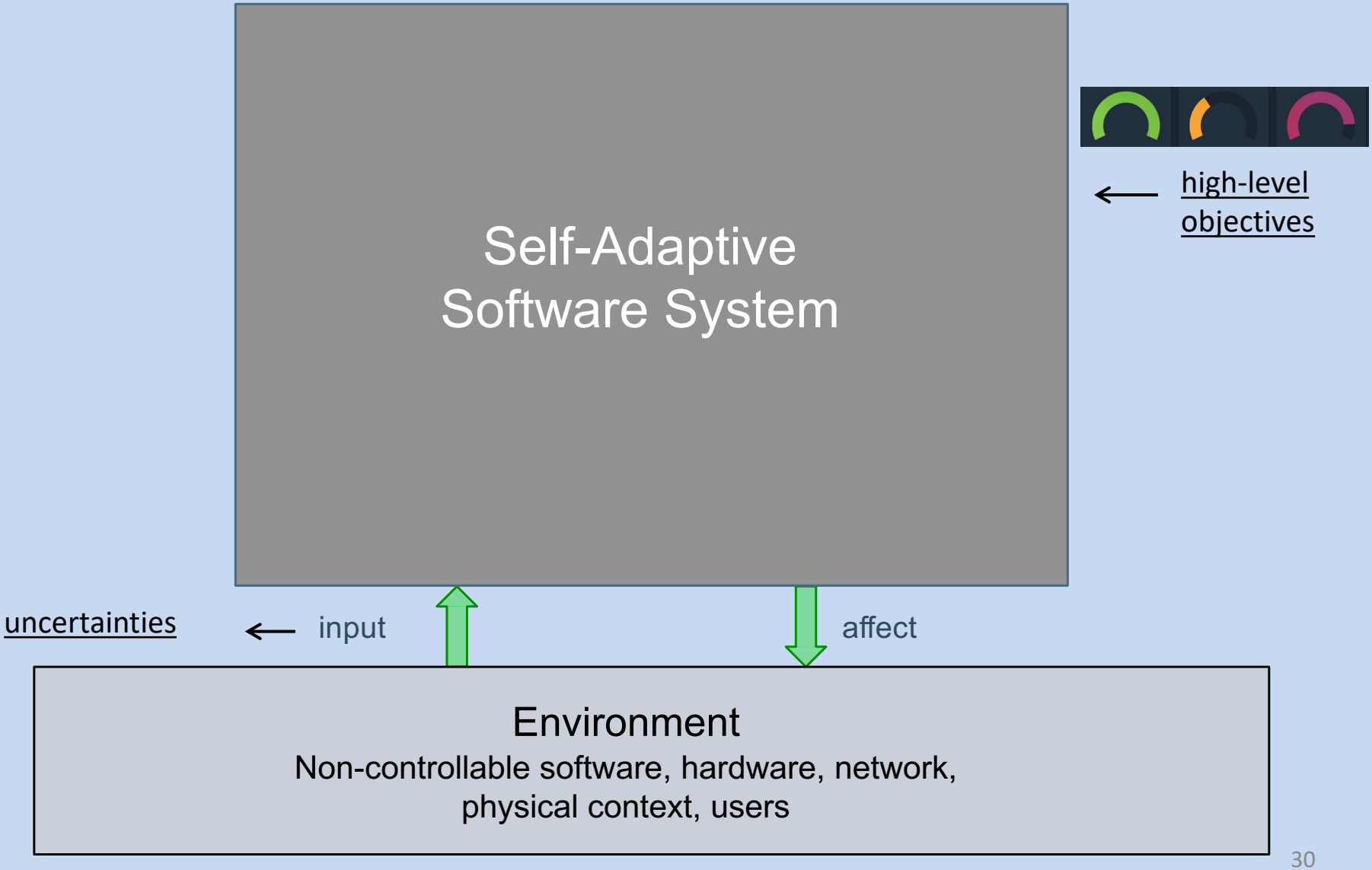
input

affect

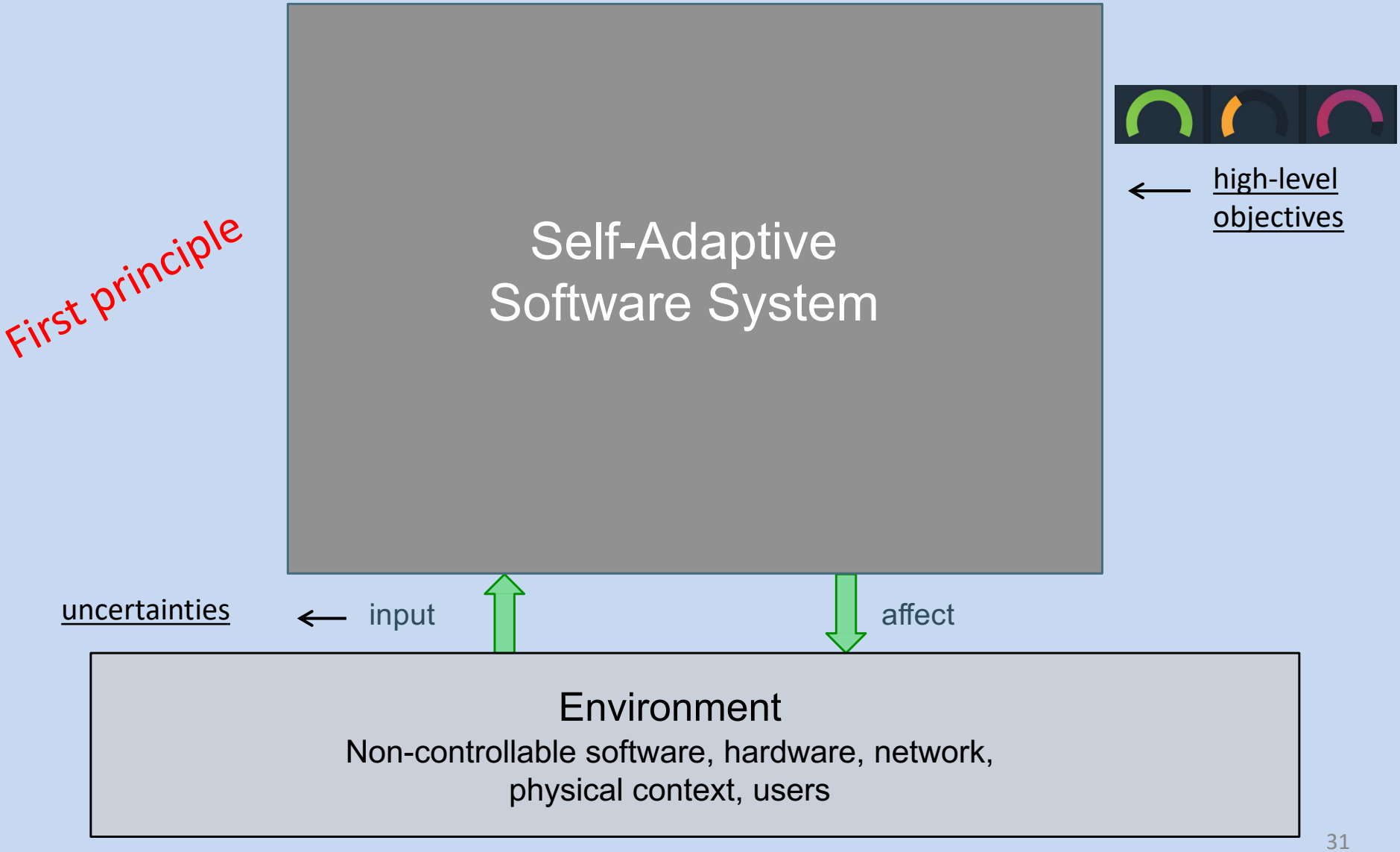
Environment

Non-controllable software, hardware, network, physical context, users

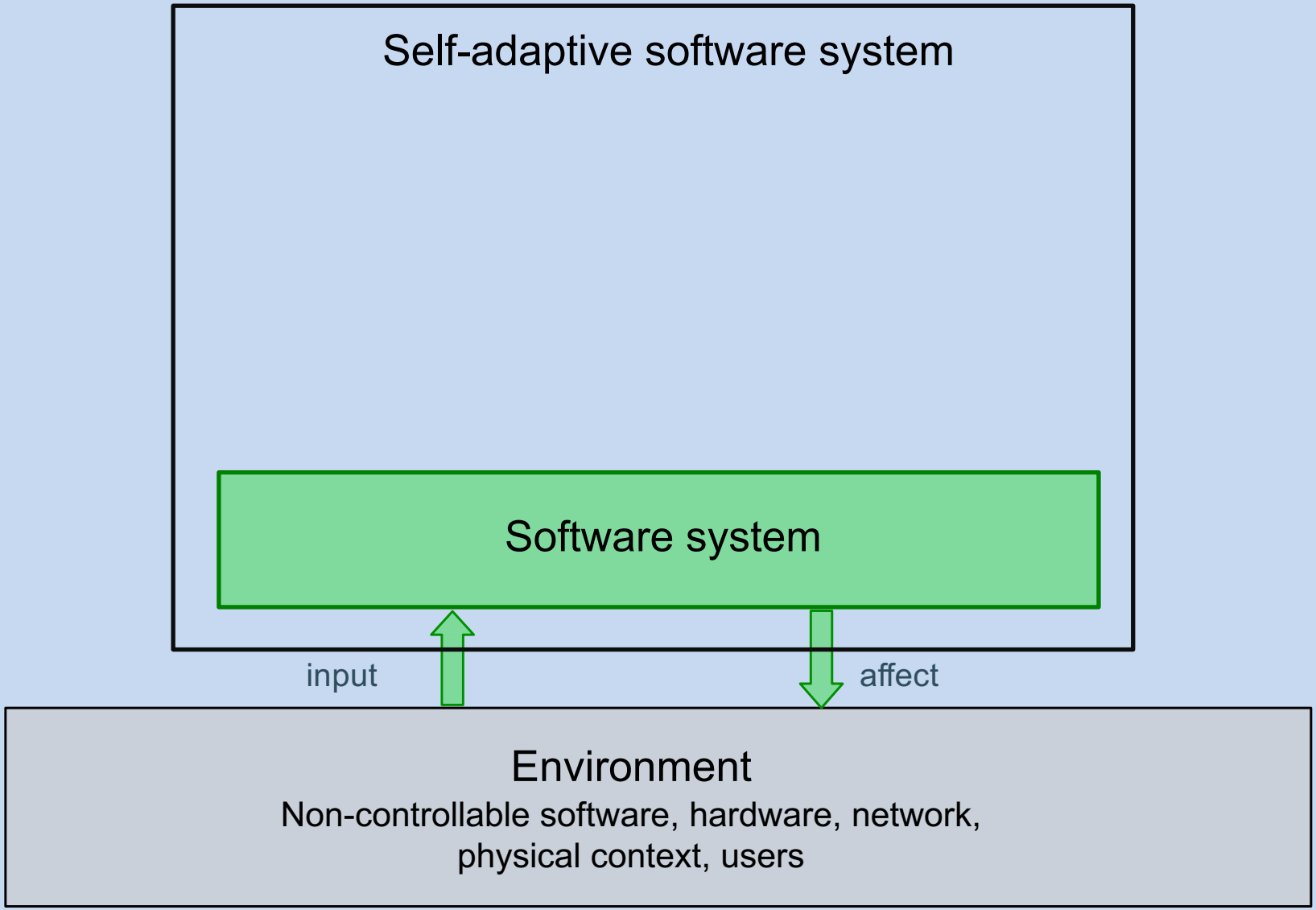
Conceptual model self-adaptive system



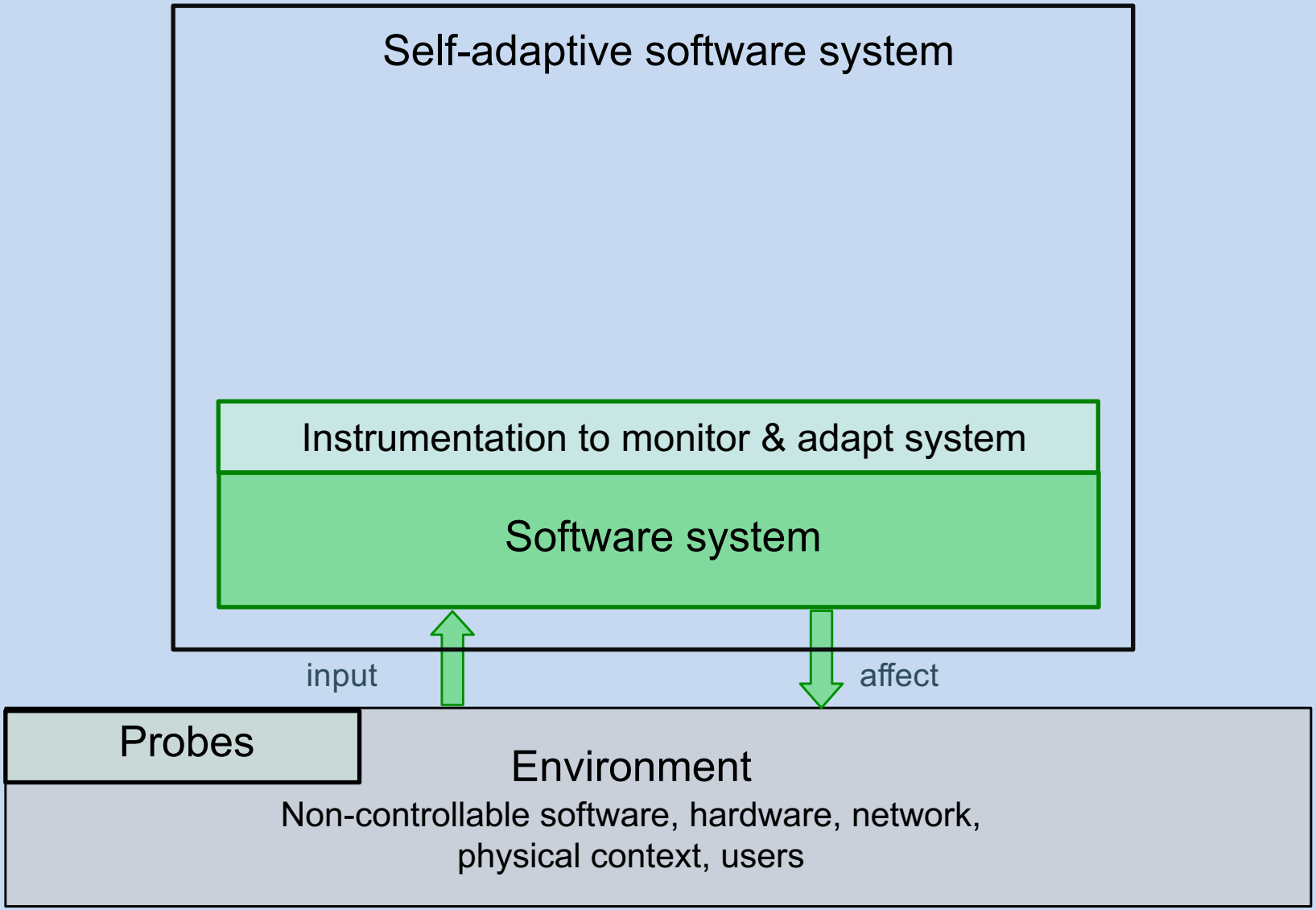
Conceptual model self-adaptive system



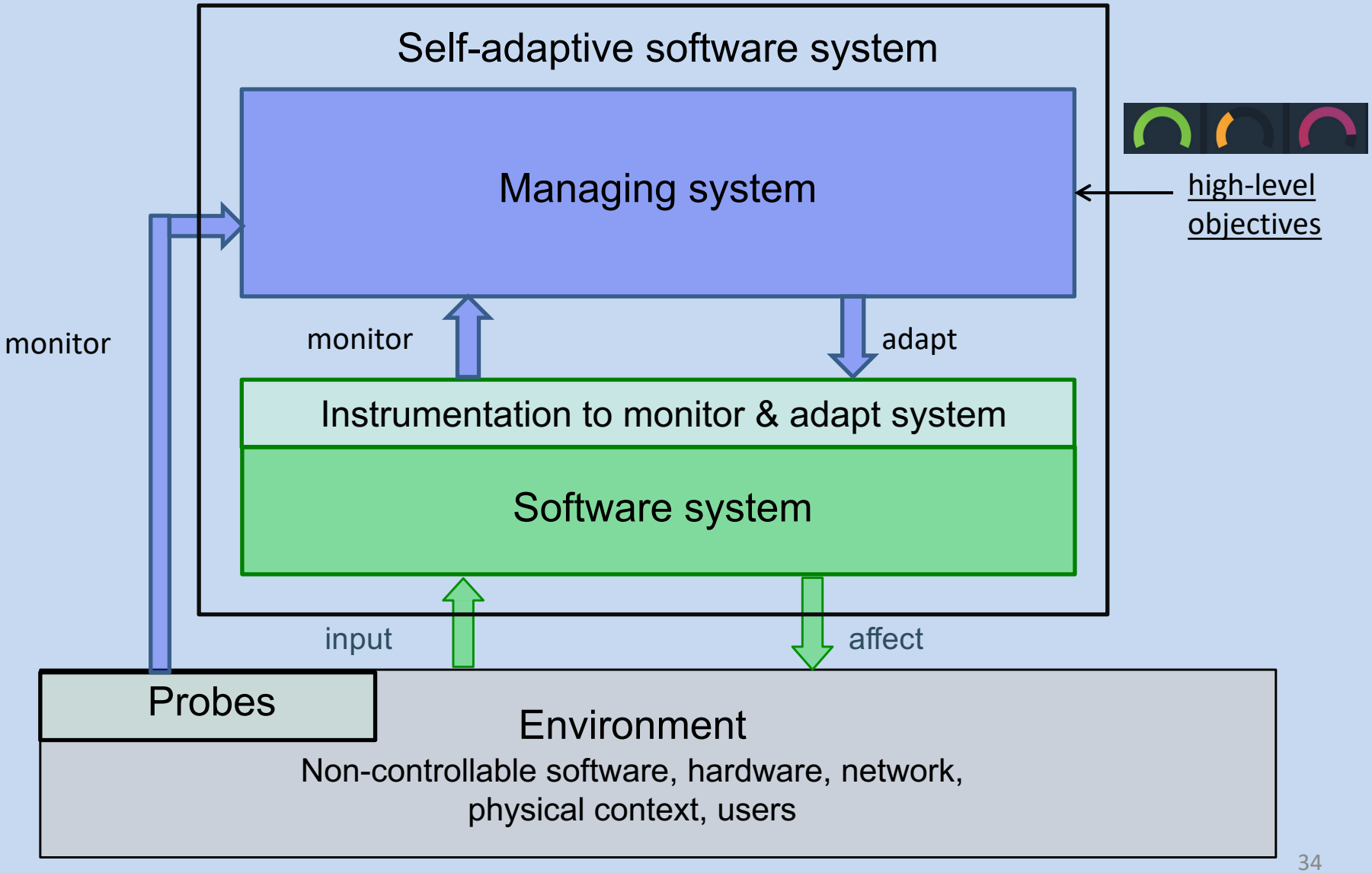
Conceptual model self-adaptive system



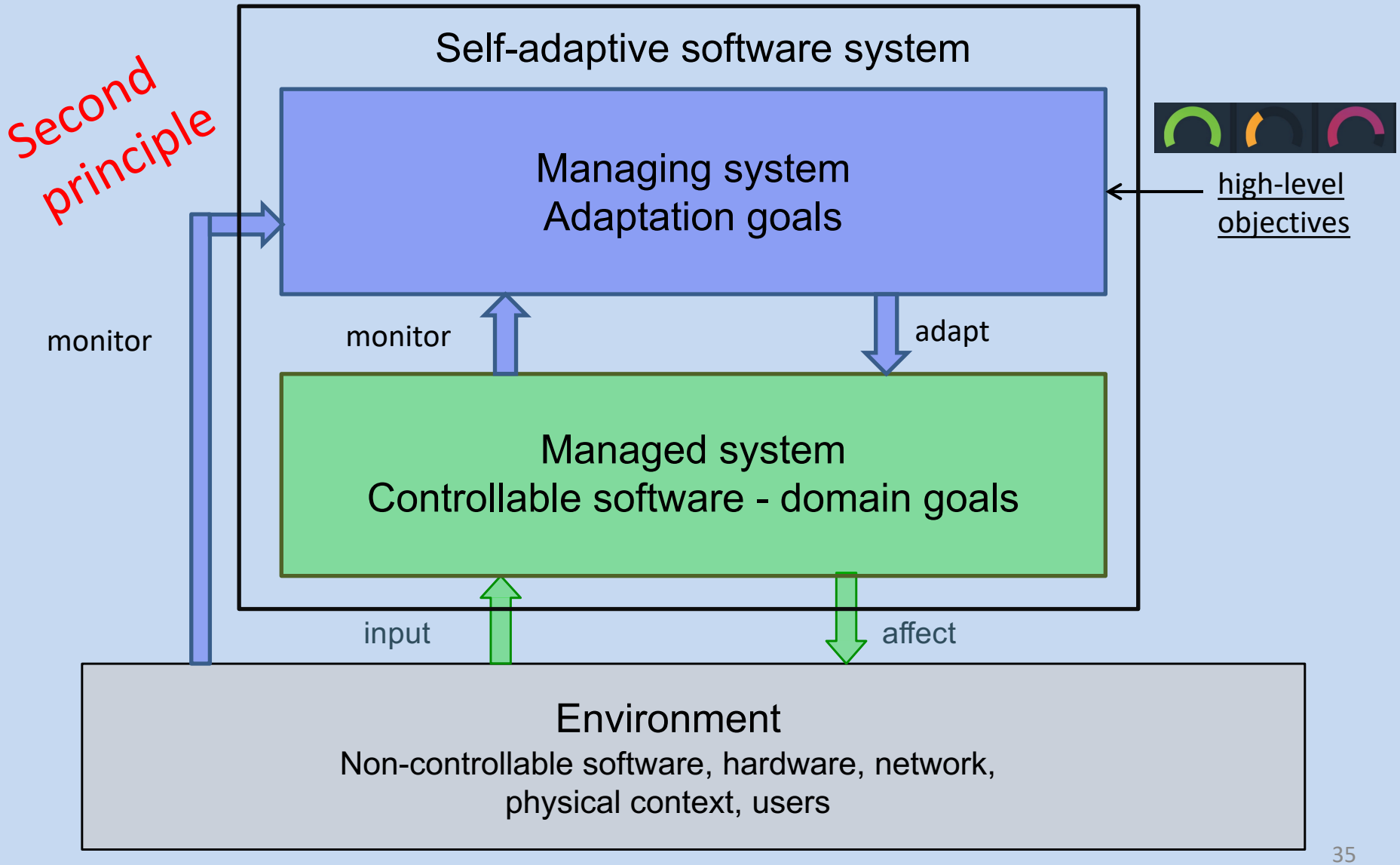
Conceptual model self-adaptive system



Conceptual model self-adaptive system



Conceptual model self-adaptive system

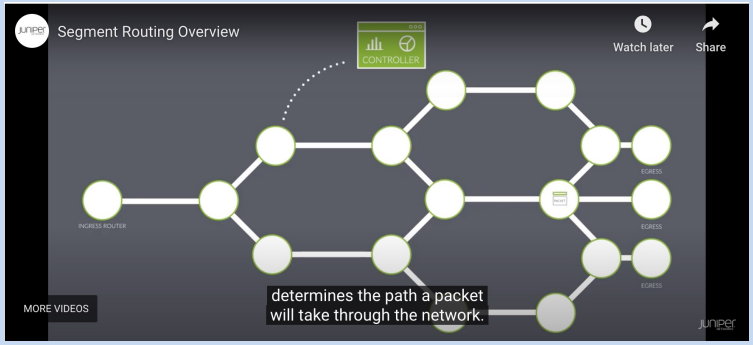


Overview

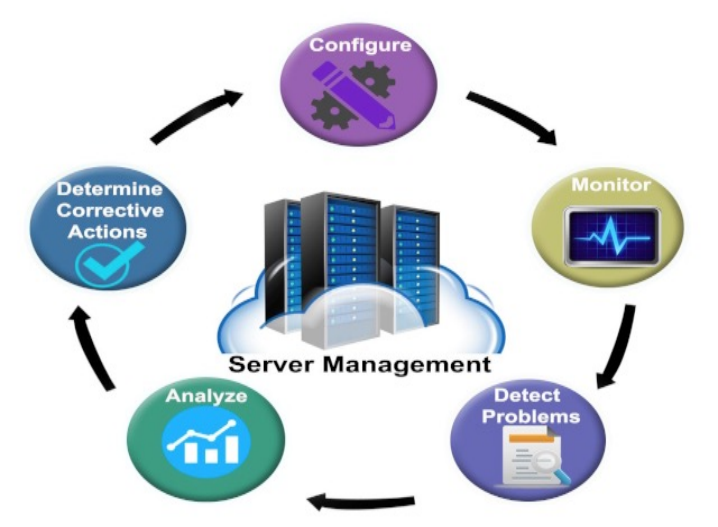
- Why self-adaptation?
- What is self-adaptation?
- Self-adaptation in industry
- How to engineer self-adaptive systems?
- Decision-making with formal techniques at runtime
- Why is “just self-adaptation” not enough?
- Conclusions

Self-adaptation in industry

“Self-adaptation seems to be everywhere”

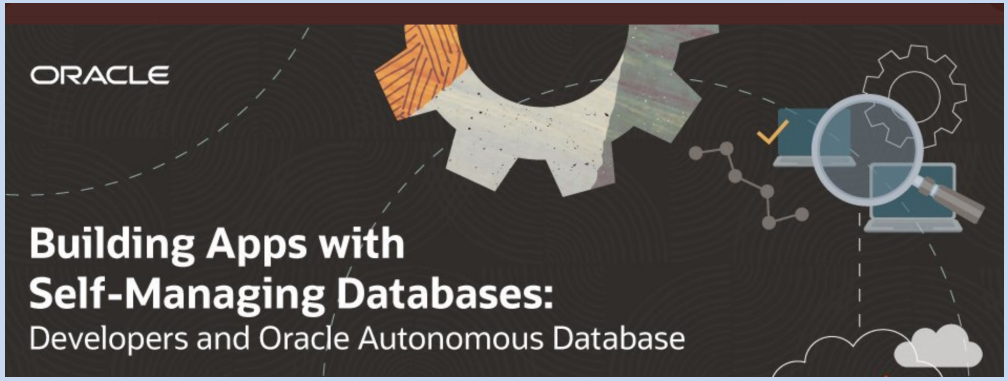


From https://www.juniper.net/



From http://www.jayatech.in/

From https://awsadvices.com/



From https://twitter.com/OracleDevs/

Is self-adaptation used in industry, and if so, how is it used and what issues does industry face with self-adaptation?

Use of self-adaptation in industry

- Large scale survey to obtain insight in state of the practice in self-adaptation
- Problems tackled, mechanisms to solve, problems and risks, opportunities
- 334 invitations (across the world)

Use of self-adaptation in industry

- 184 valid responses
- 100 apply self-adaptation (54,3%)
- Half of the companies with > 100 software engineers
- Main roles: programmer, designer/architect, maintainer, project lead, specific expert
- 70% participants with > +9 years of expertise

Use of self-adaptation in industry

Top types of systems built by organisations

Type of system	# (all)	# (self-adaptation)
Web/mobile systems	35 (19.3%)	15
Embedded/CPS/IoT	24 (13.3%)	13
Data management	21 (11.6%)	10
Cloud	15 (8.3%)	11
Transportation	13 (7.2%)	9
Communication and network	11 (6.1%)	7
IT infrastructure	11 (6.1%)	7
Manufacturing	10 (5.5%)	8

Use of self-adaptation in industry

Top problems to apply self-adaptation

Problem	#
To optimise performance	78
To automate tasks	61
To deal with changes in the environment	60
To (re-)configure a system	51
To detect and resolve errors	46
To detect and protect a system against threats	33
To deal with changing business goals	15

Use of self-adaptation in industry

Top types of self-adaptation applied

Type of adaptation	#
Auto-scaling	33
Auto-tuning	28
Monitor/analysis	22
Automated (re)-configuration	11

Use of self-adaptation in industry

Top subject of self-adaptation

Type of adaptation	#
System	28
Module	22
Platform layer	13
Application layer	11

Use of self-adaptation in industry

Key benefits of self-adaptation

Category / code	**#**
Improved utility	**61**
Robustness	21
Performance	16
Savings	**38**
Costs	25
Resources	13
Improved human interaction	**37**
User experience	19
Burden engineers	18
Handle dynamics timely	**22**
Load changes	12
Context changes	10

Use of self-adaptation in industry

Main difficulties with self-adaptation

Category / code	**#**
Design issues	**43**
Reliable/optimal design	26
Design complexity	17
Lifecycle issues	**42**
Tuning/debugging	19
Limitations tools/methods	13
System/environment evolution	10
Runtime challenges	**30**
Runtime uncertainty	17
Data collection/evaluation	7
People and processes	**19**
Skills/experience	14
Process and management	9

Use of self-adaptation in industry

Main risks with self-adaptation

Risks	#
Faults (correctness, misconfiguration)	20
Impact on qualities (performance, availability…)	16
Difficulties design/development	16
Impact on business (cost, trust…)	14

Use of self-adaptation in industry

Main risks with self-adaptation

Risks	#
Faults (correctness, misconfiguration)	20
Impact on qualities (performance, availability…)	16
Difficulties design/development	16
Impact on business (cost, trust…)	14

Main risks mitigation mechanisms

Risks	#
Extensive testing	15
Rogorous design/development	8
Runtime monitoring and analysis	6
Human supervision	4

Overview

- Why self-adaptation?
- What is self-adaptation?
- Self-adaptation in industry
- How to engineer self-adaptive systems?
- Decision-making with formal techniques at runtime
- Why is “just self-adaptation” not enough?
- Conclusions

How to engineer self-adaptive systems?

Focus on different engineering aspects (vs different approaches)

Seven Waves Beach - Santa Marta

I. Automating tasks

II. Architecture-based adaptation

III. Runtime models

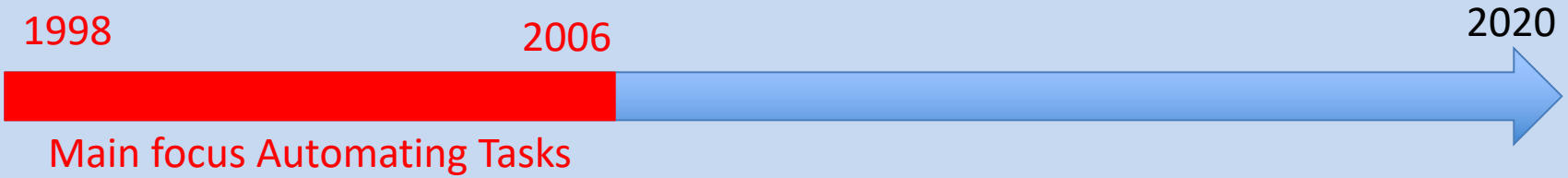
IV. Requirements-driven adaptation

V. Guarantees under uncertainty

VI. Control-based adaptation

VII. Learn from experiences

Wave I
Automating Tasks

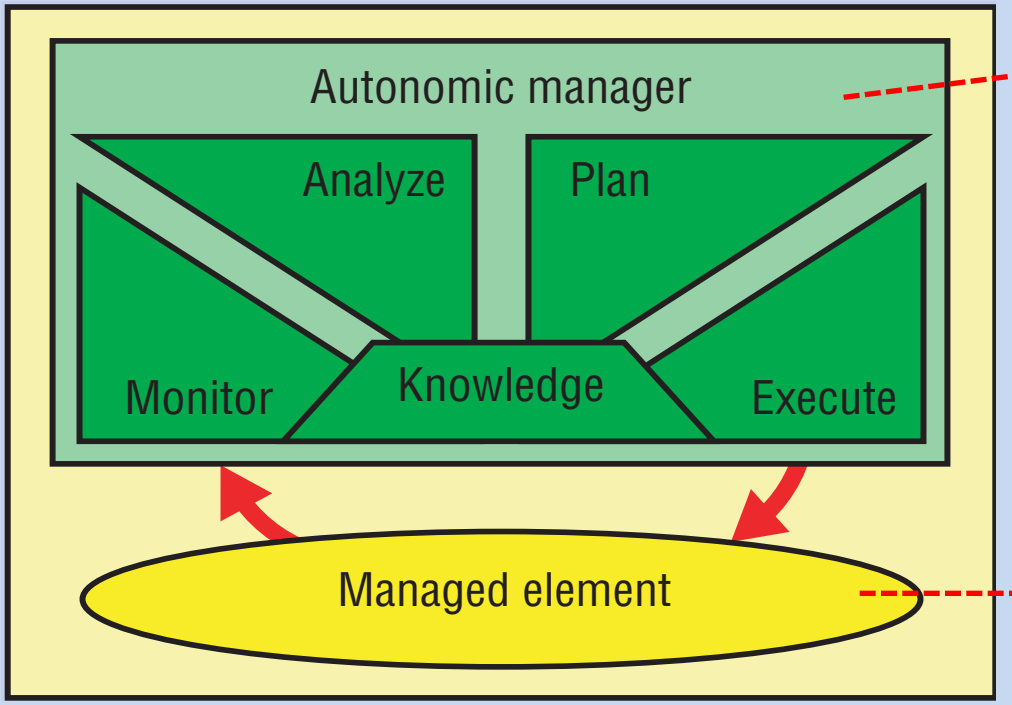


Driver: System management done by human operators is complex and error prone

Topic: Automation of management tasks

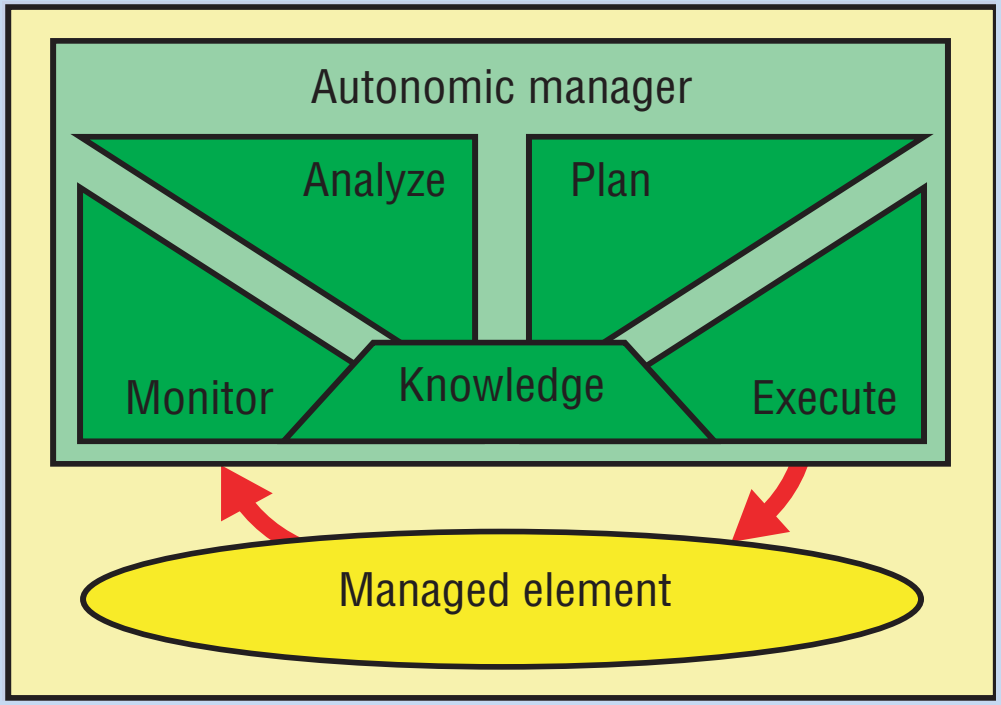
Enabled: System manages itself autonomously based on high-level objectives

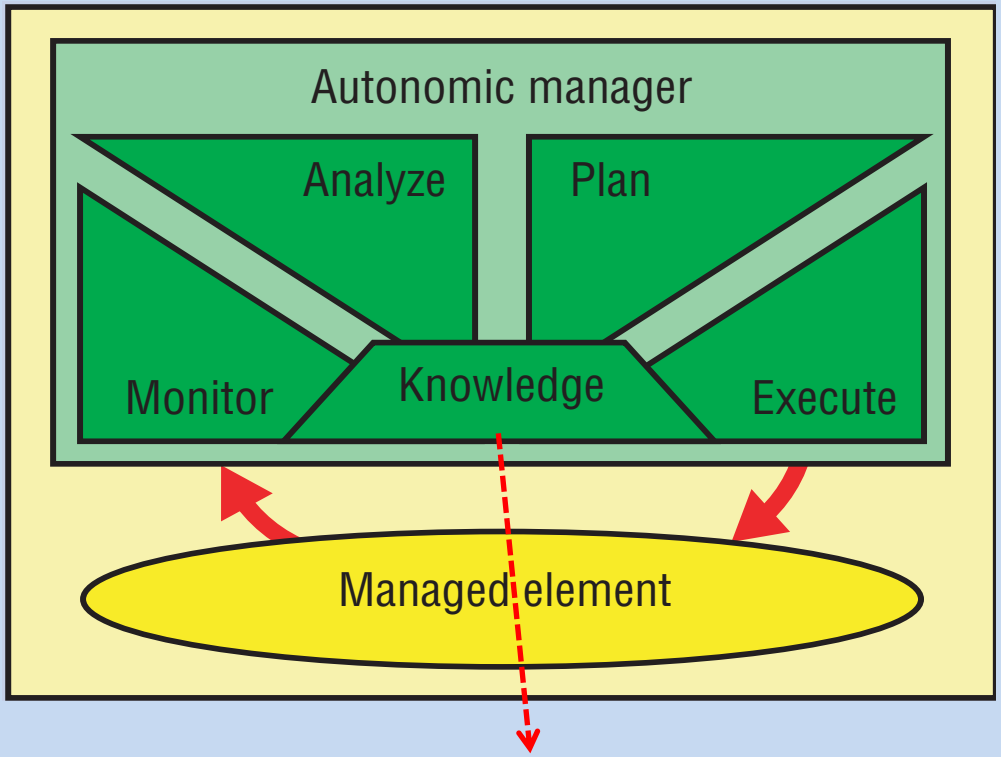
IBM's MAPE-K reference model



Relieves admins of responsibility to manage managed element

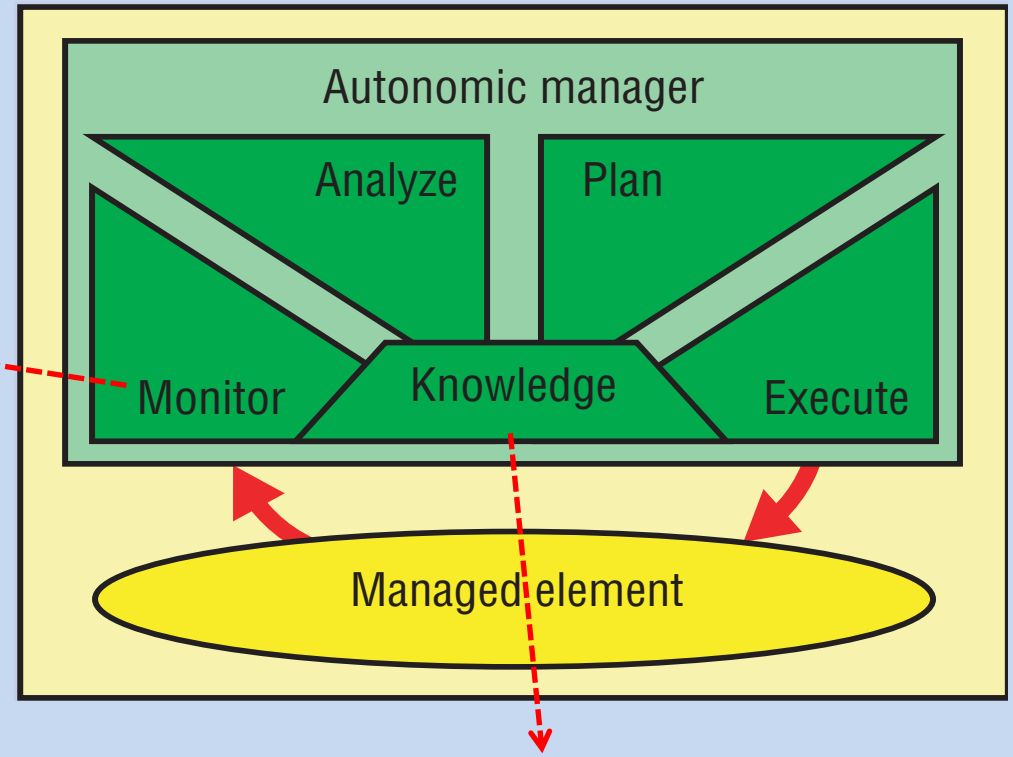
Element to be monitored and controlled to realize admins' goals


Autonomic manager
Analyze
Plan
Monitor
Knowledge
Execute
Managed element



Abstraction of relevant aspects of the managed element, environment, and the administrator's goals

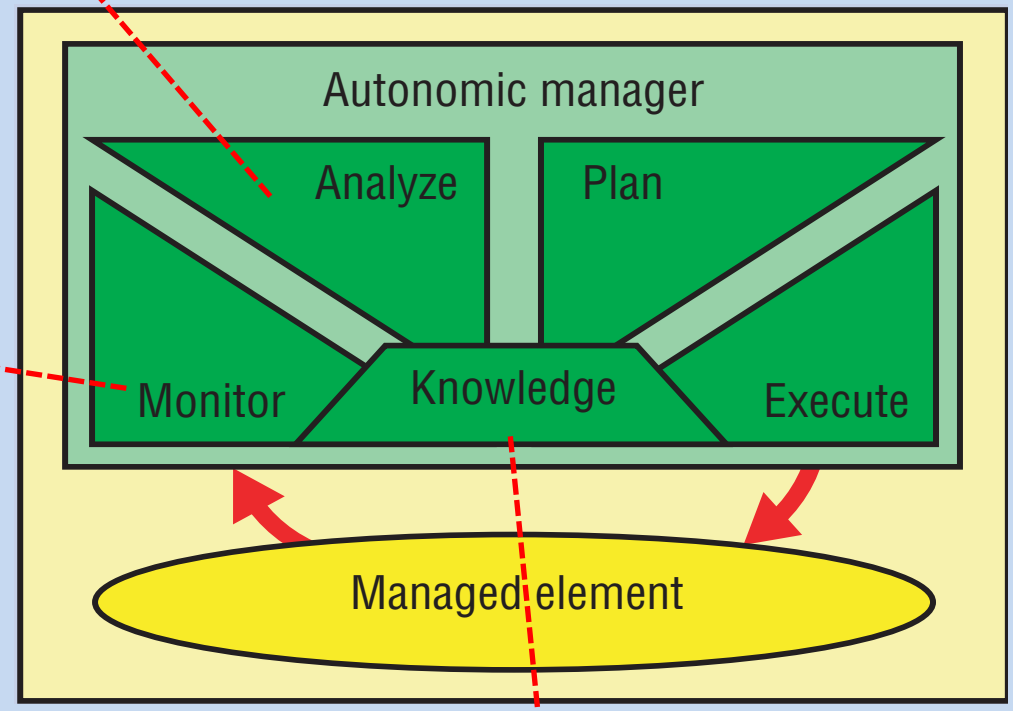
Collects data from managed element and execution context



Abstraction of relevant aspects of the managed element, environment, and the administrator's goals

Determines need for adaptation and if so analyses the options

Collects data from managed element and execution context

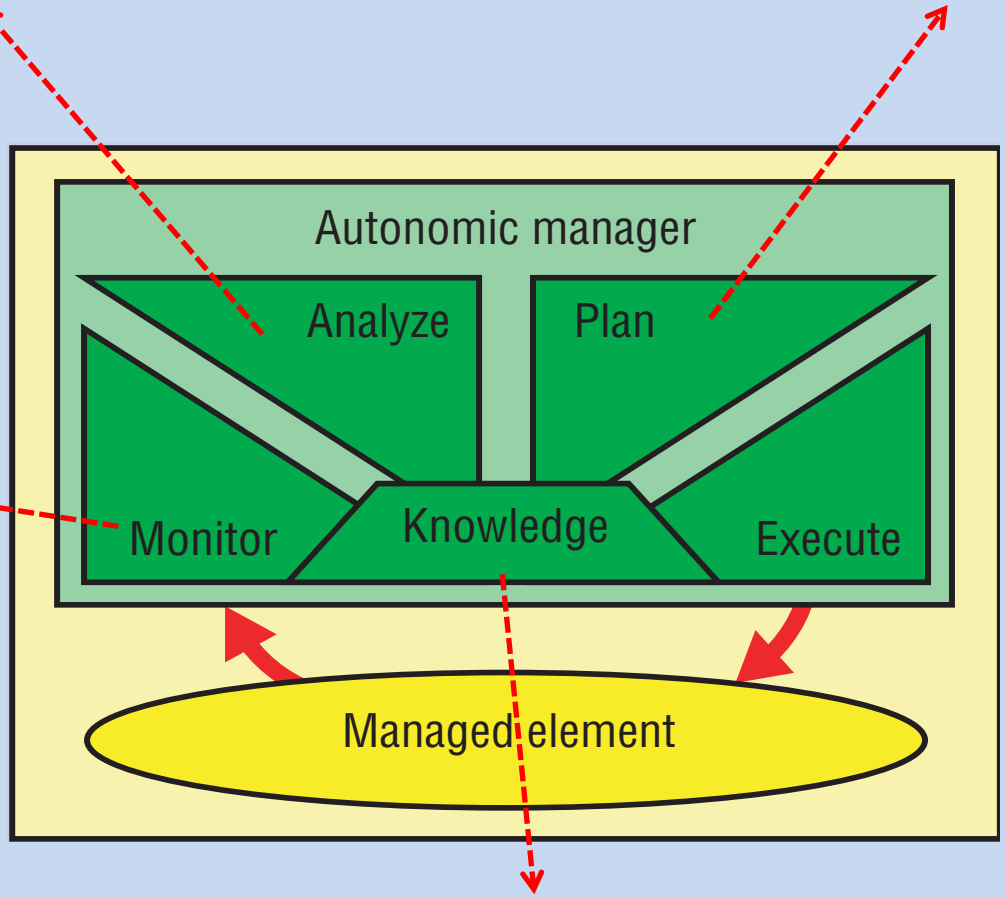


Abstraction of relevant aspects of the managed element, environment, and the administrator's goals

Determines need for adaptation and if so analyses the options

Plans mitigation actions to adapt the managed element when needed

Collects data from managed element and execution context

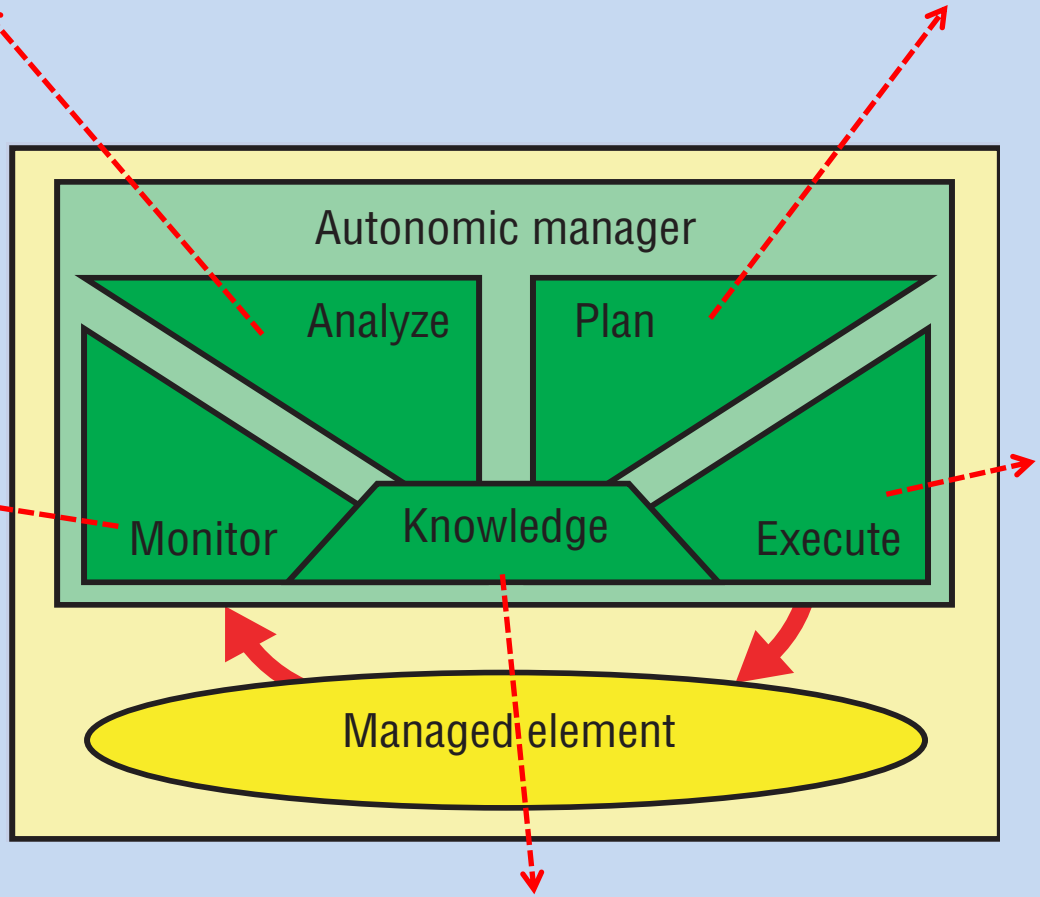


Abstraction of relevant aspects of the managed element, environment, and the administrator's goals

Determines need for adaptation and if so analyses the options

Plans mitigation actions to adapt the managed element when needed

Collects data from managed element and execution context



Enacts the plan adapting the managed element

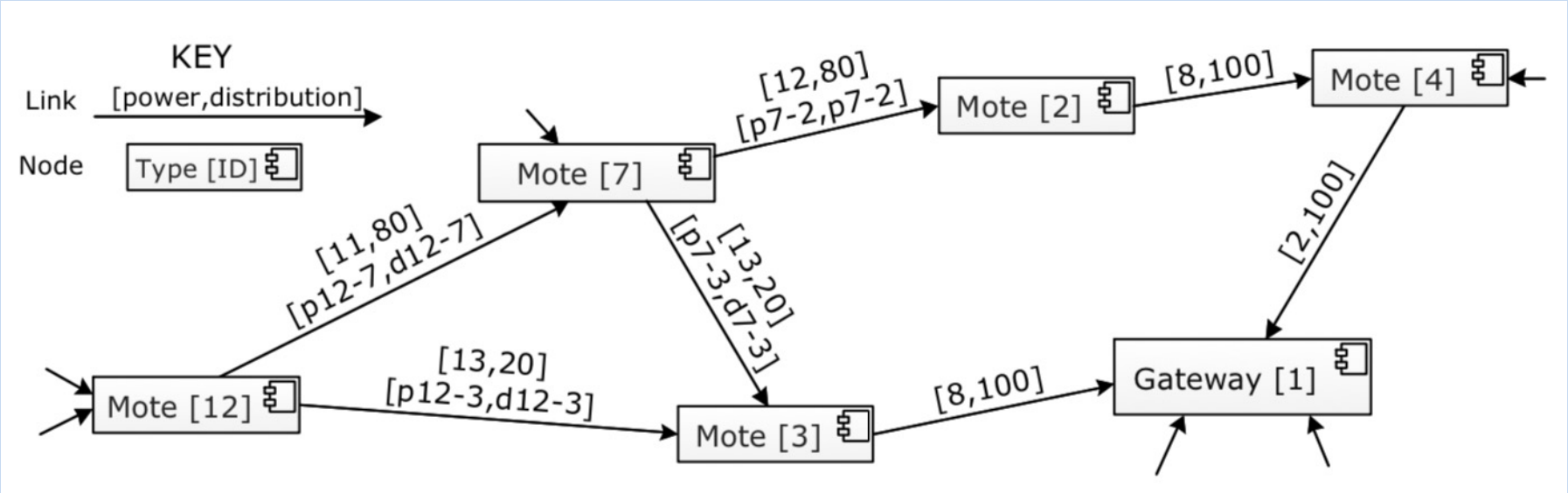
Abstraction of relevant aspects of the managed element, environment, and the administrator's goals

Essential maintenance tasks for automation

Essential maintenance tasks for automation

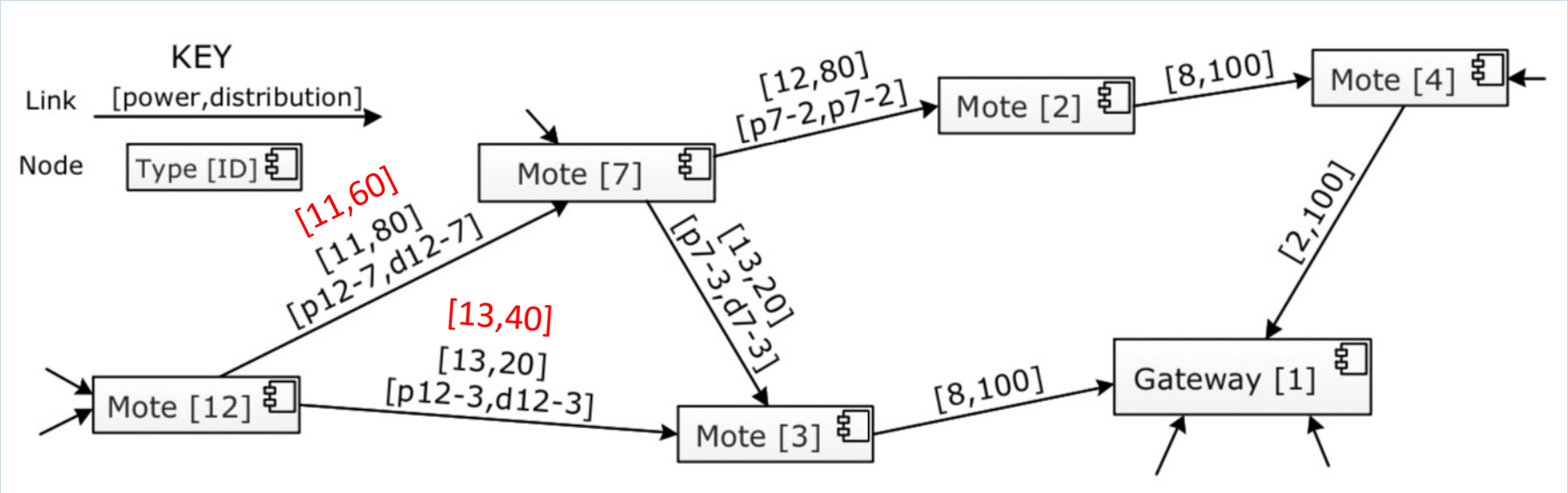
Self-optimization is the capability of a system to seek opportunities to optimize its resources usage, while providing the required quality goals.

Self-optimization



Current configuration: packet loss 9% < 10% -- latency 4% < 5% -- energy consumption 24 C

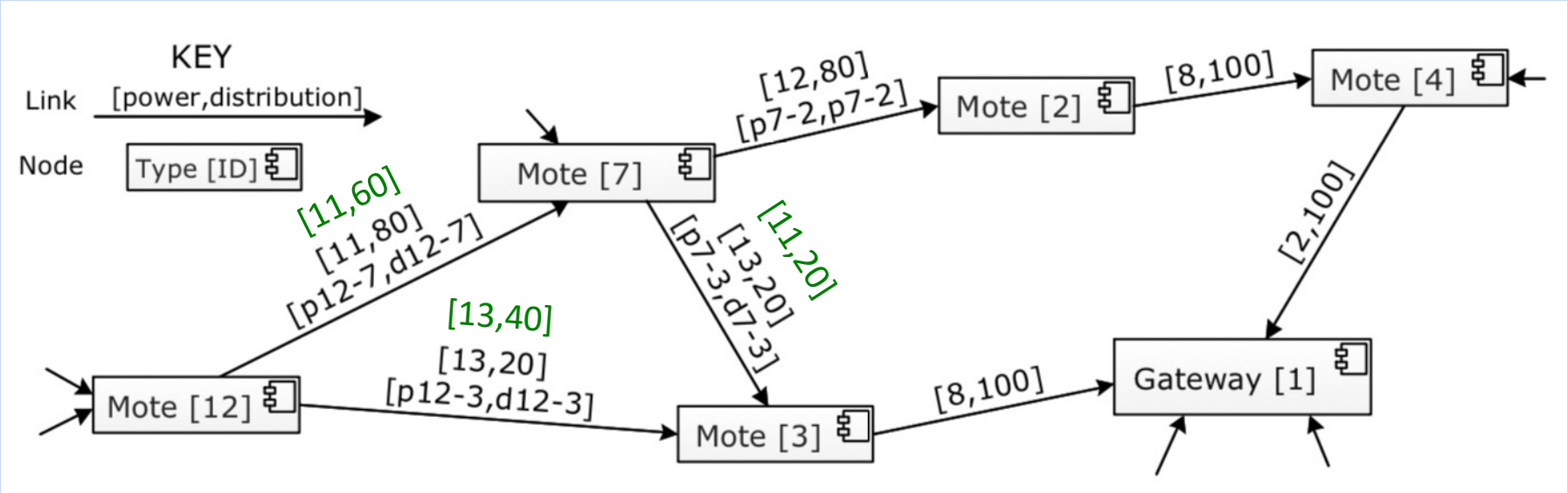
Self-optimization



Current configuration: packet loss 9% < 10% -- latency 4% < 5% -- energy consumption 24 C

Alternative 1: packet loss 9% < 10% -- latency 2% < 5% -- energy consumption 22 C

Self-optimization

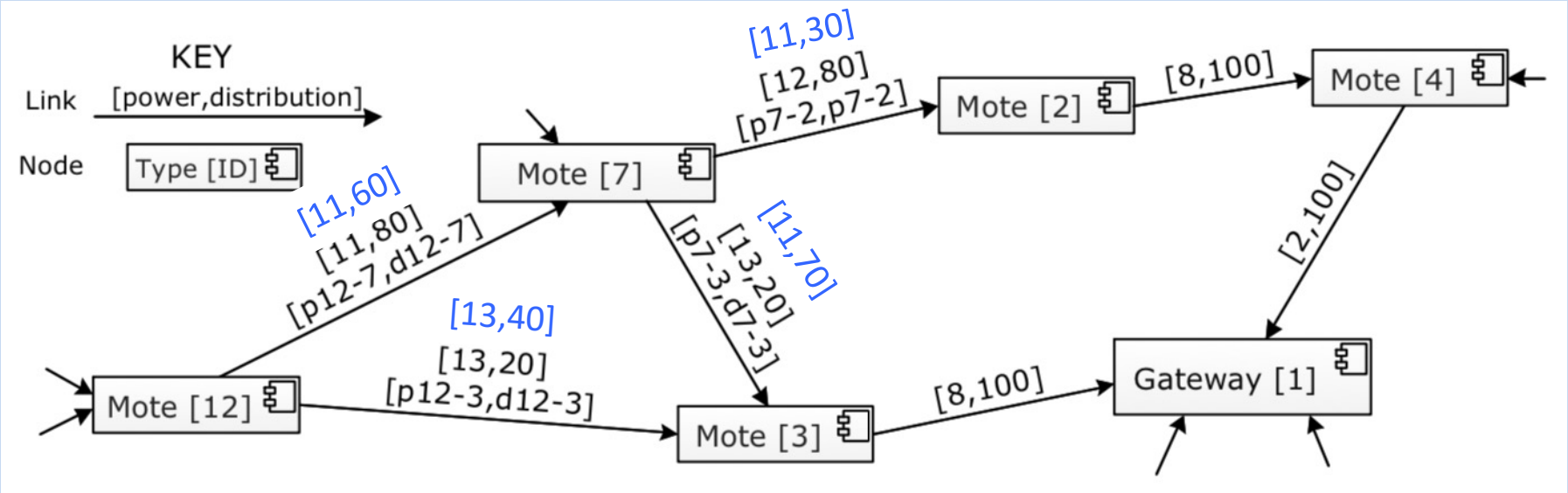


Current configuration: packet loss 9% < 10% -- latency 4% < 5% -- energy consumption 24 C

Alternative 1: packet loss 9% < 10% -- latency 2% < 5% -- energy consumption 22 C

Alternative 2: packet loss 8% < 10% -- latency 11% > 5% -- energy consumption 12 C

Self-optimization



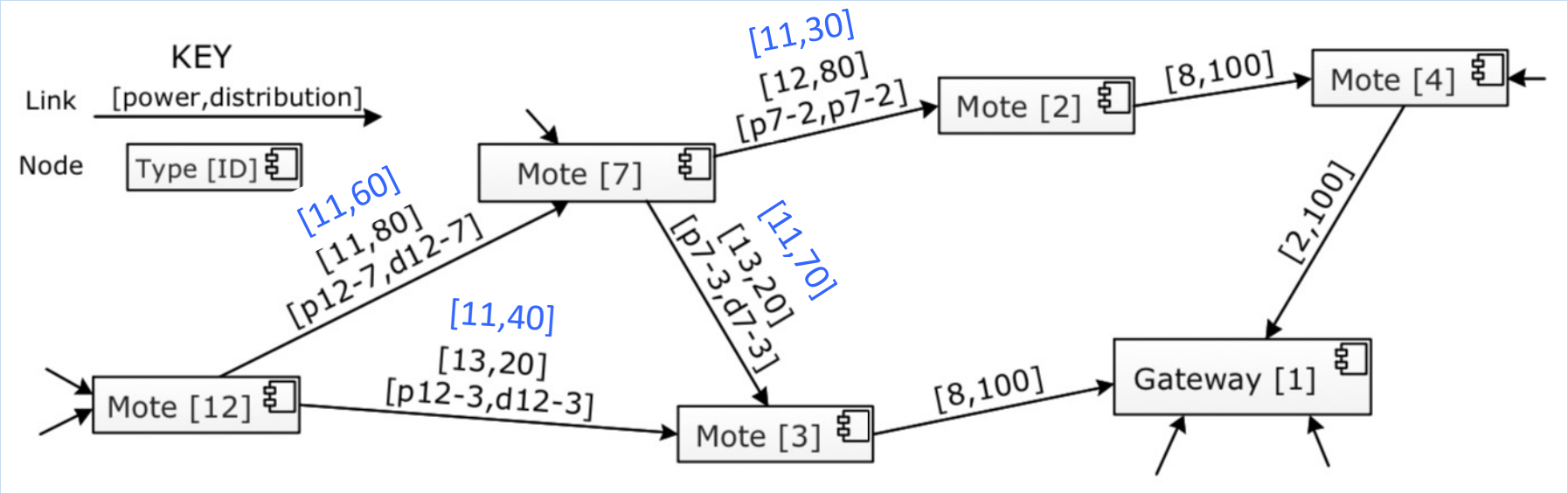
Current configuration: packet loss 9% < 10% -- latency 4% < 5% -- energy consumption 24 C

Alternative 1: packet loss 9% < 10% -- latency 2% < 5% -- energy consumption 22 C

Alternative 2: packet loss 8% < 10% -- latency 11% > 5% -- energy consumption 12 C

Alternative 3: packet loss 9% < 10% -- latency 3% < 5% -- energy consumption 19 C

Self-optimization



Current configuration: packet loss 9% < 10% -- latency 4% < 5% -- energy consumption 24 C

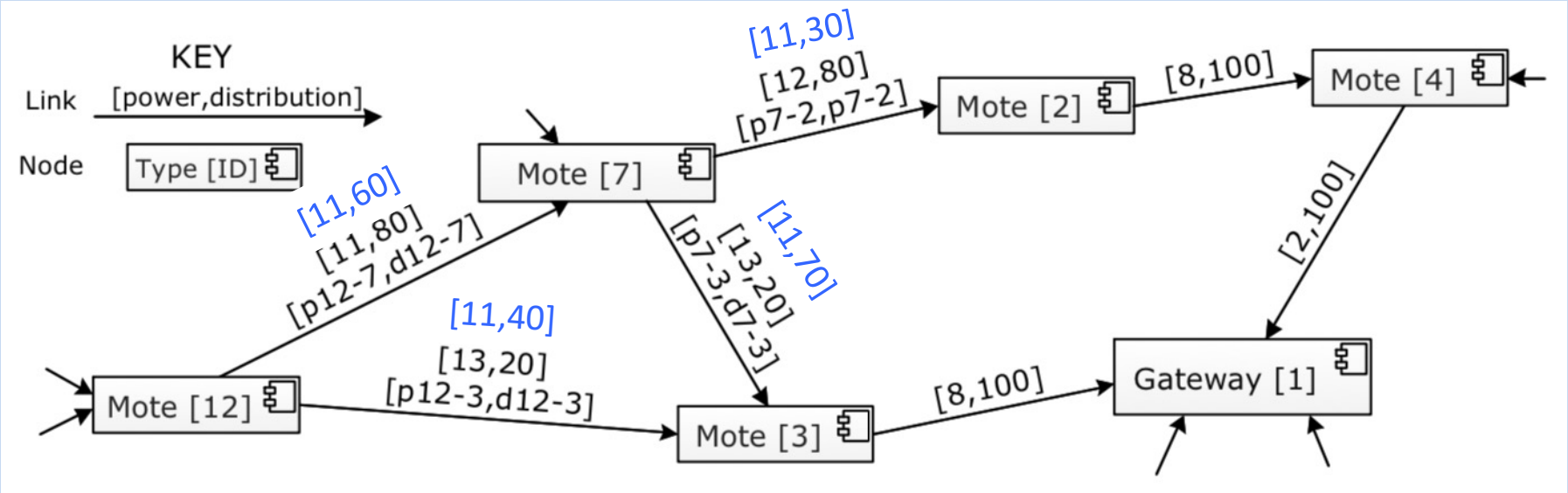
Alternative 1: packet loss 9% < 10% -- latency 2% < 5% -- energy consumption 22 C

Alternative 2: packet loss 8% < 10% -- latency 11% > 5% -- energy consumption 12 C

Alternative 3: packet loss 9% < 10% -- latency 3% < 5% -- energy consumption 19 C

=> switch to ?

Self-optimization



Current configuration: packet loss 9% < 10% -- latency 4% < 5% -- energy consumption 24 C

Alternative 1: packet loss 9% < 10% -- latency 2% < 5% -- energy consumption 22 C

Alternative 2: packet loss 8% < 10% -- latency 11% > 5% -- energy consumption 12 C

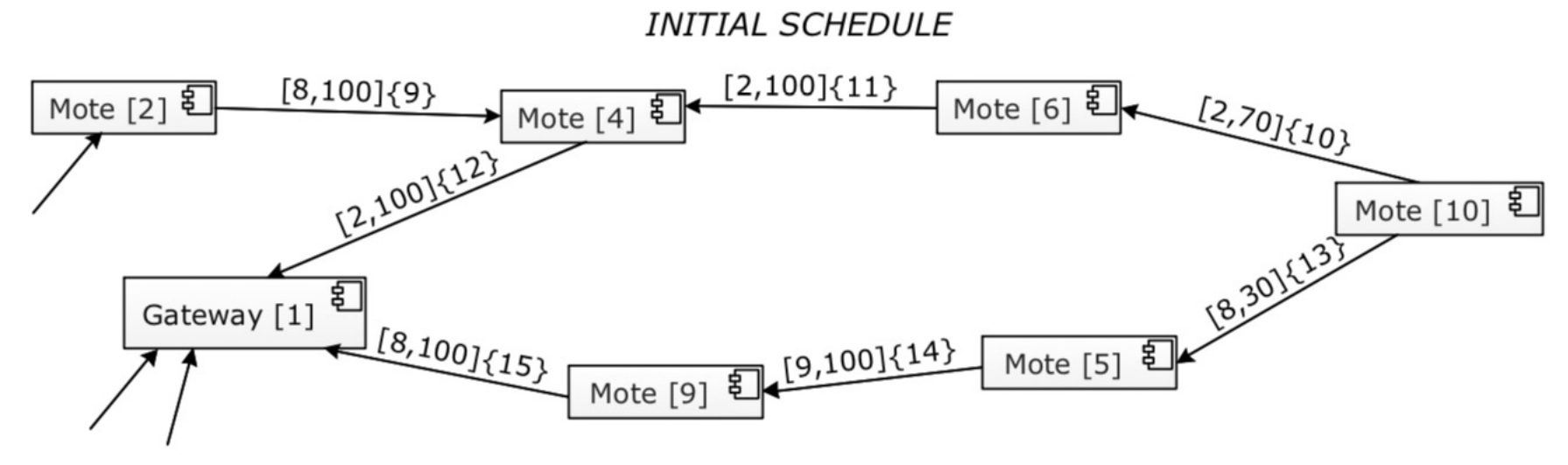
Alternative 3: packet loss 9% < 10% -- latency 3% < 5% -- energy consumption 19 C

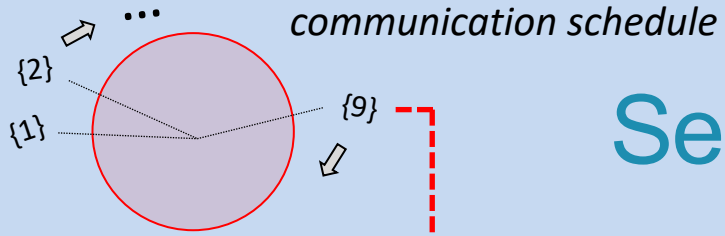
=> switch to alternative 3

Essential maintenance tasks for automation

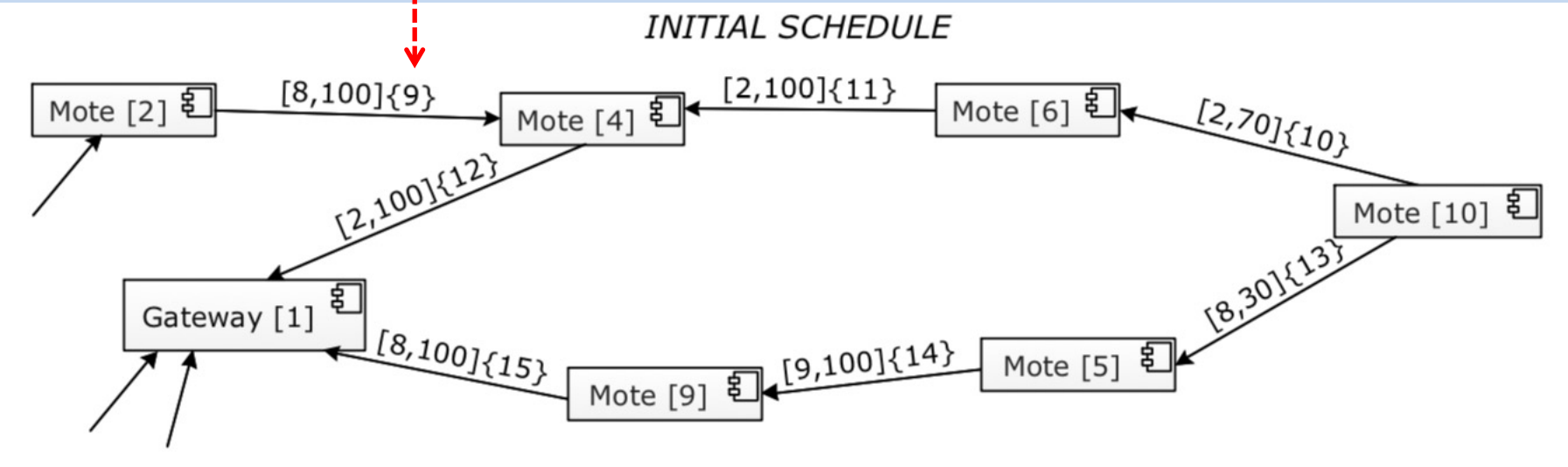
Self-healing is the capability of a system to detect and recover from faults to provide its required quality goals or degrade gracefully otherwise.

Self-healing

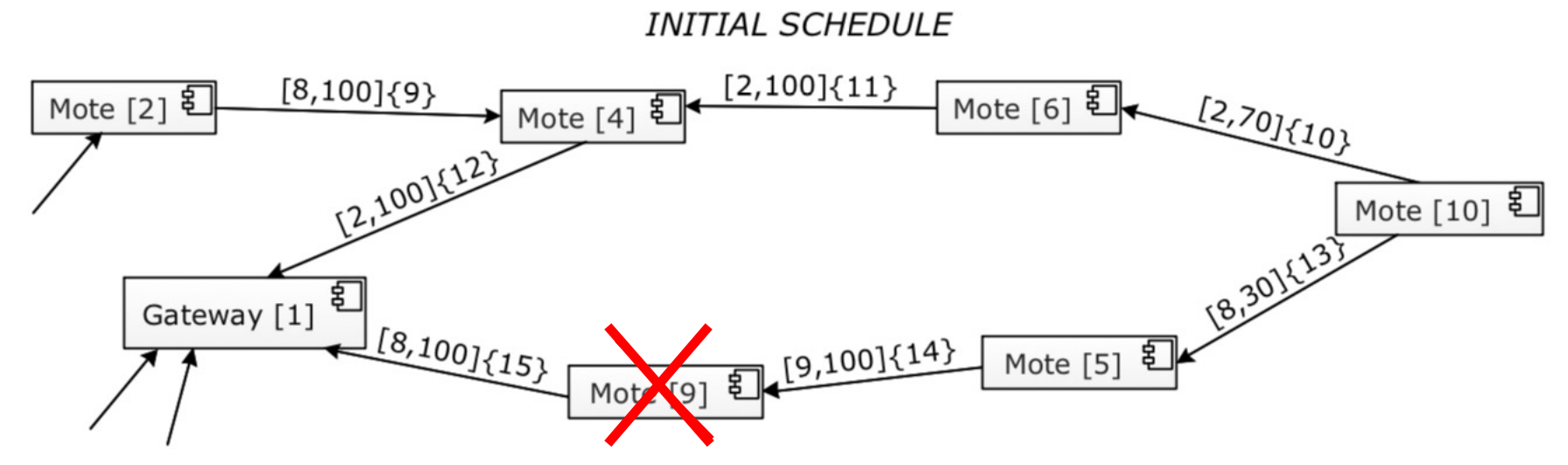




Self-healing

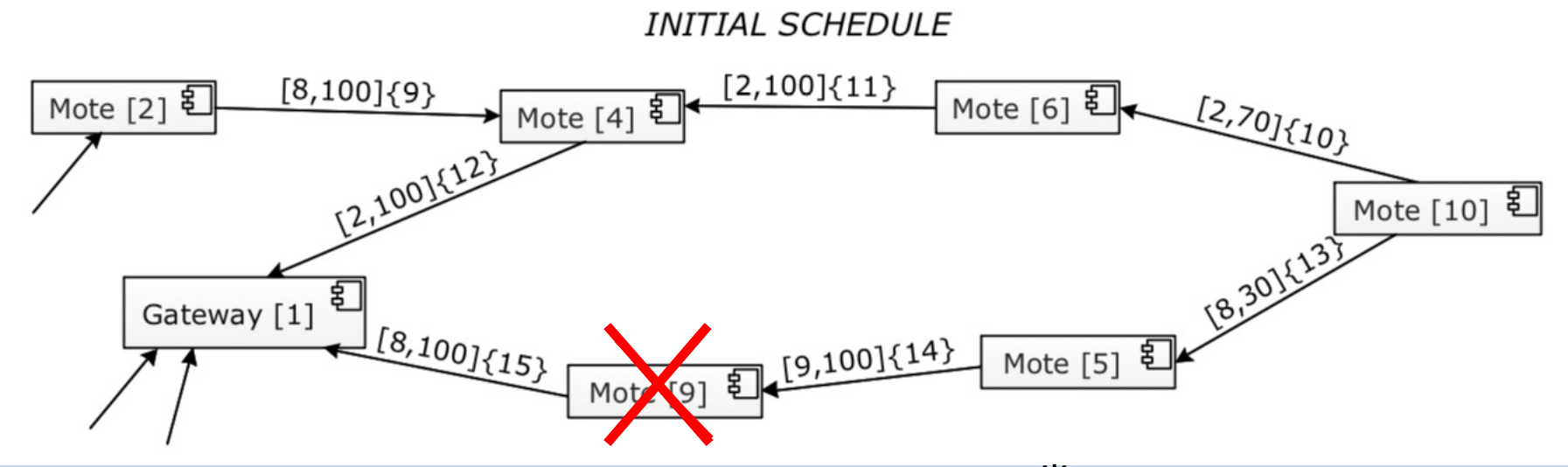


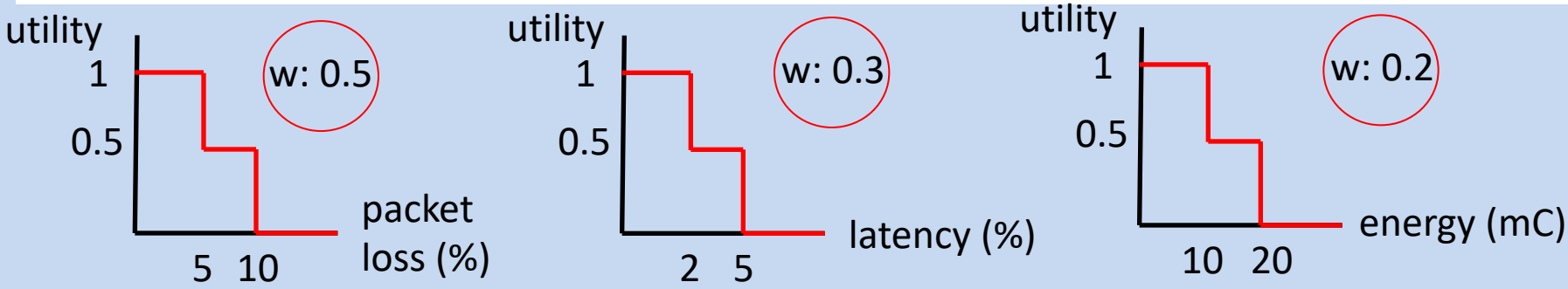
Self-healing



Mote 9 fails

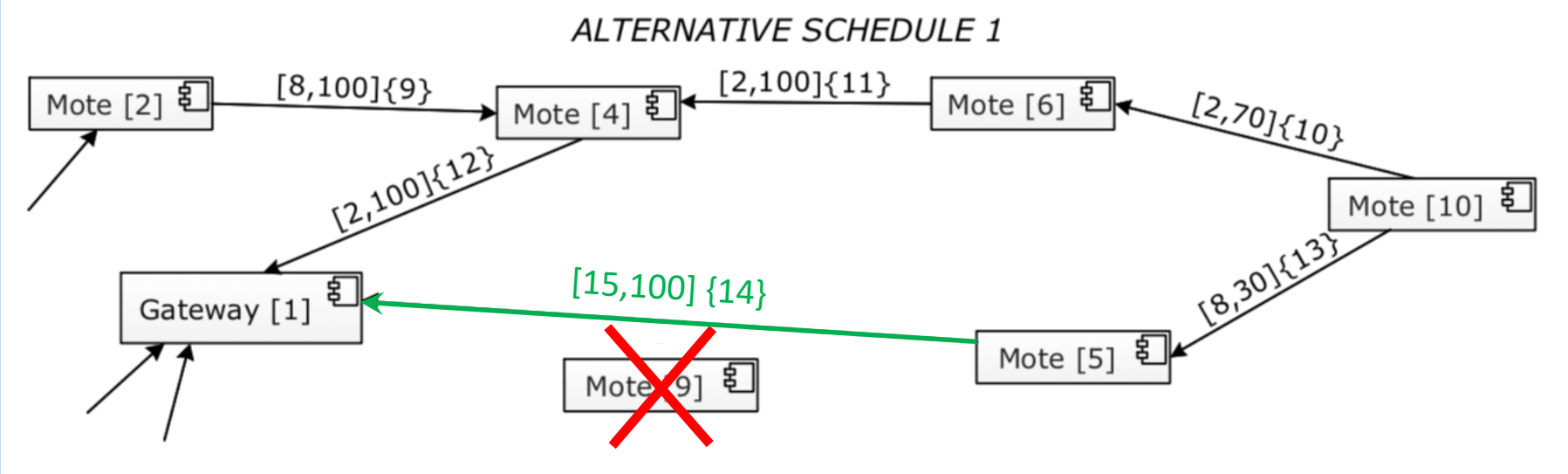
Self-healing

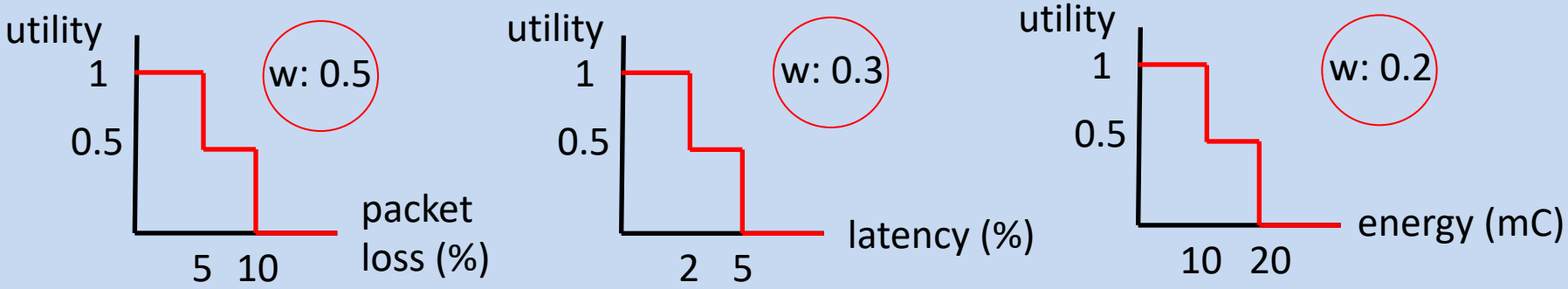




expected utility = sum (w_i x u_i)

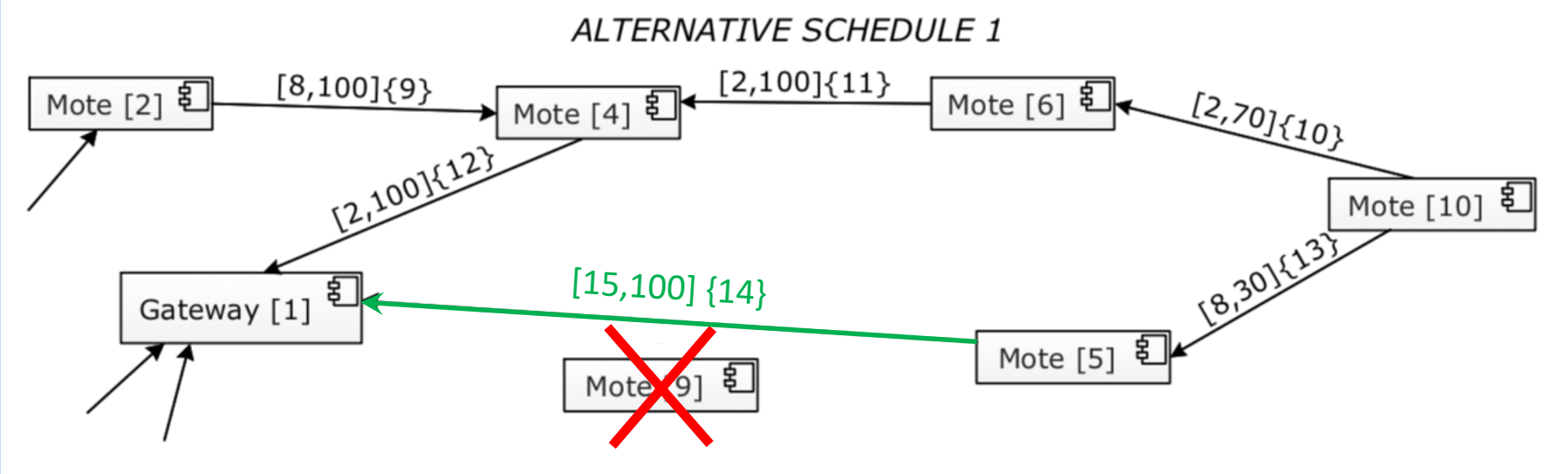
Self-healing

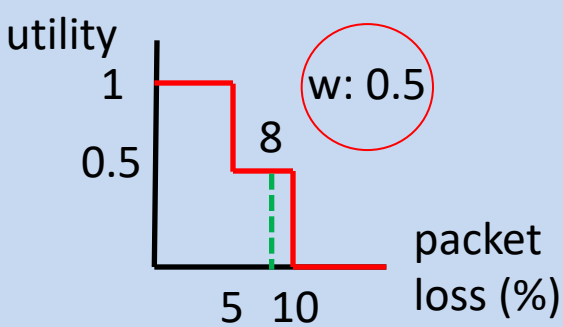


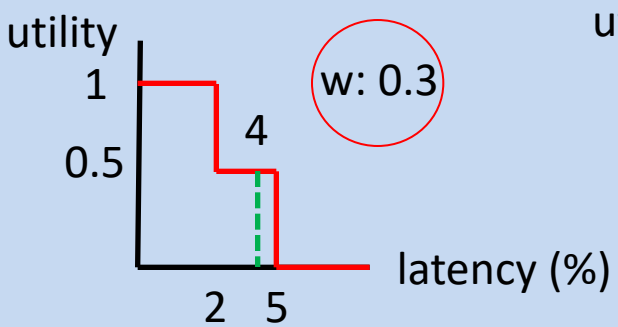


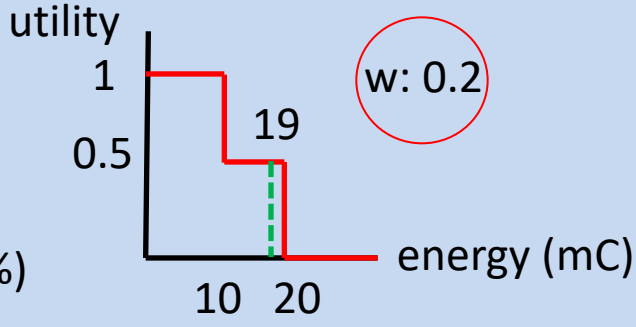
Alternative 1

Self-healing



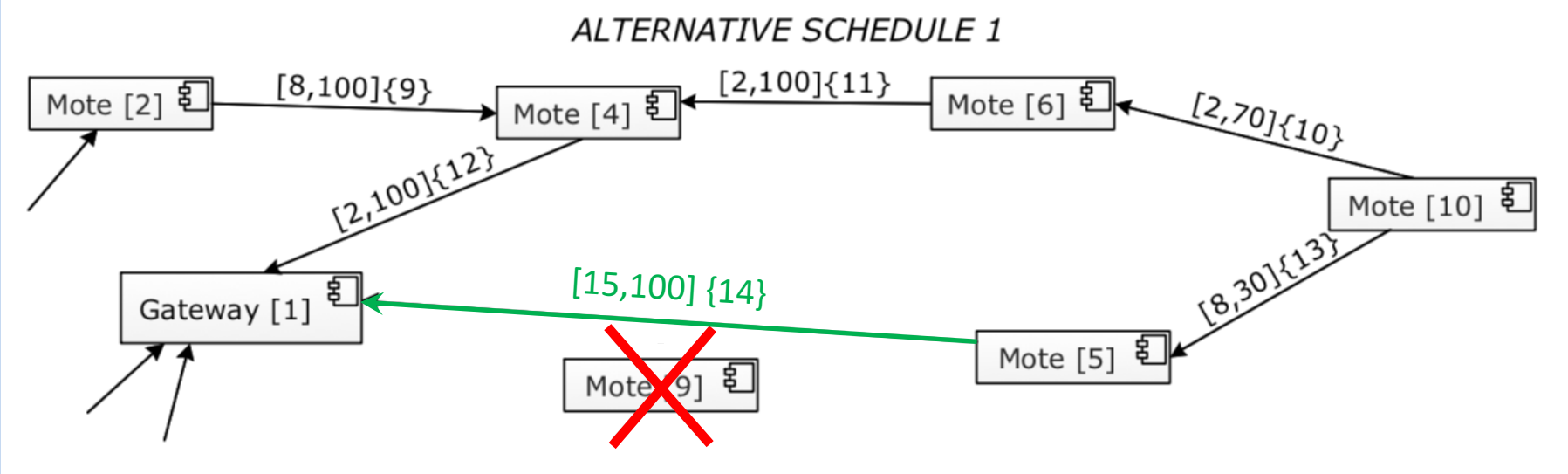


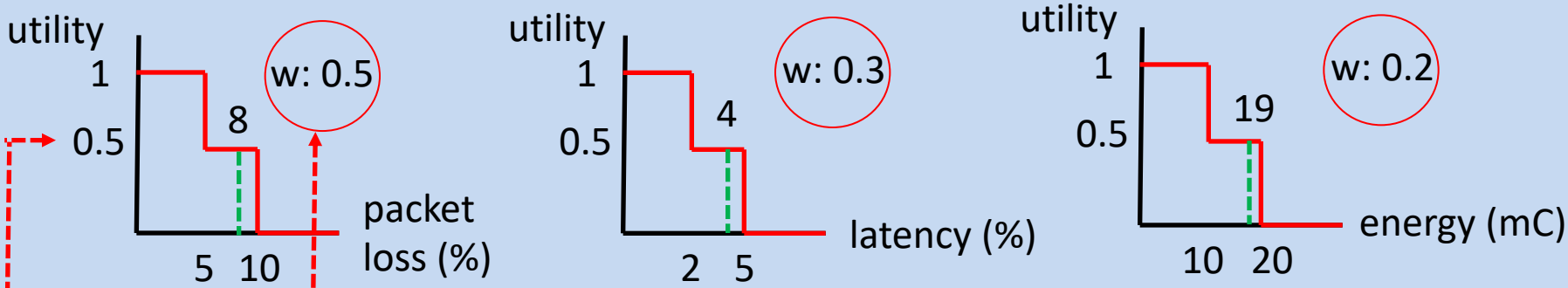




Alternative 1: expected utility = ?

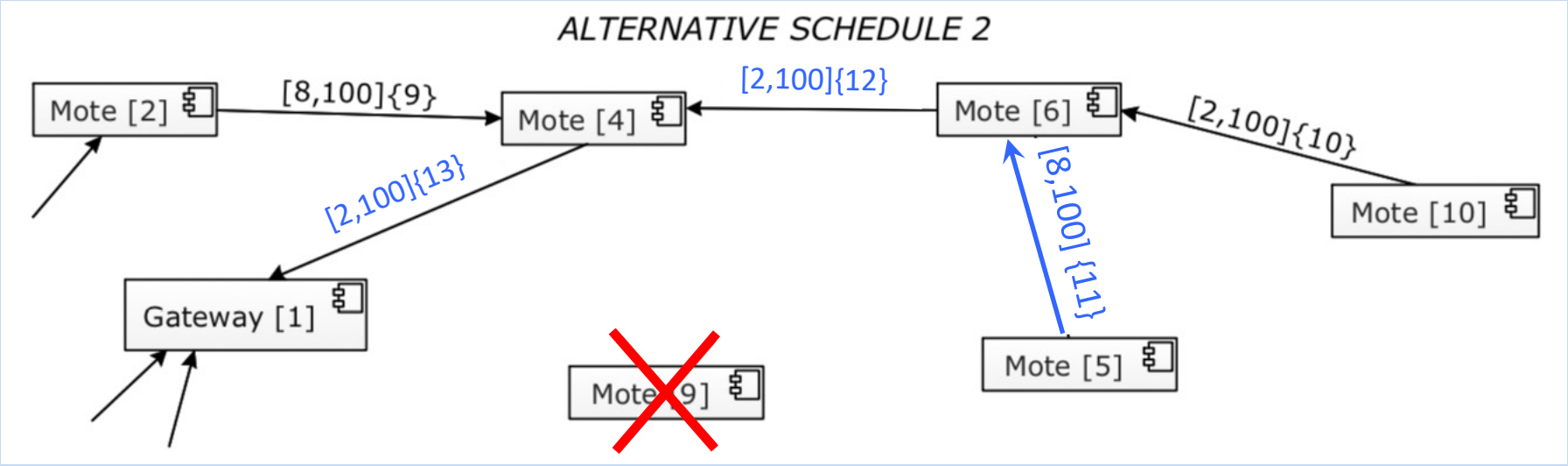
Self-healing

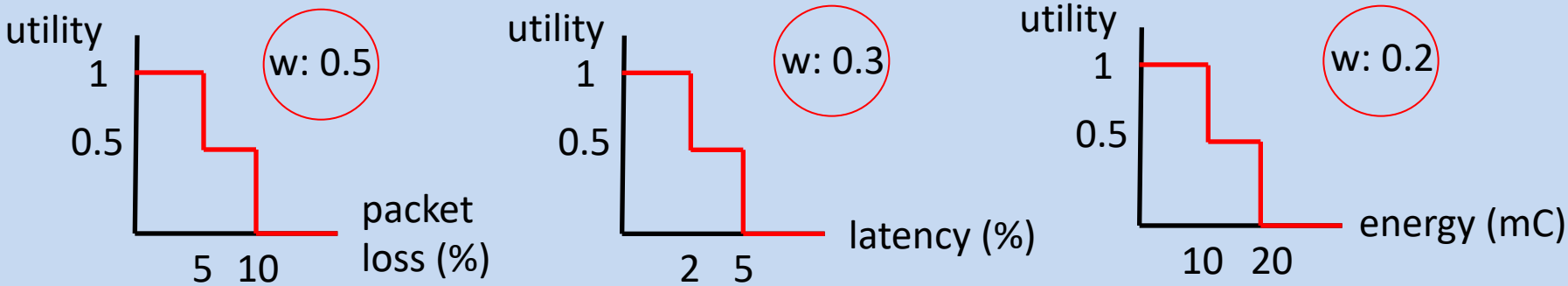




Alternative 1: expected utility = sum (w_i x u_i) => 0.5 x 0.5 + 0.3 x 0.5 + 0.2 x 0.5 = 0.50

Self-healing

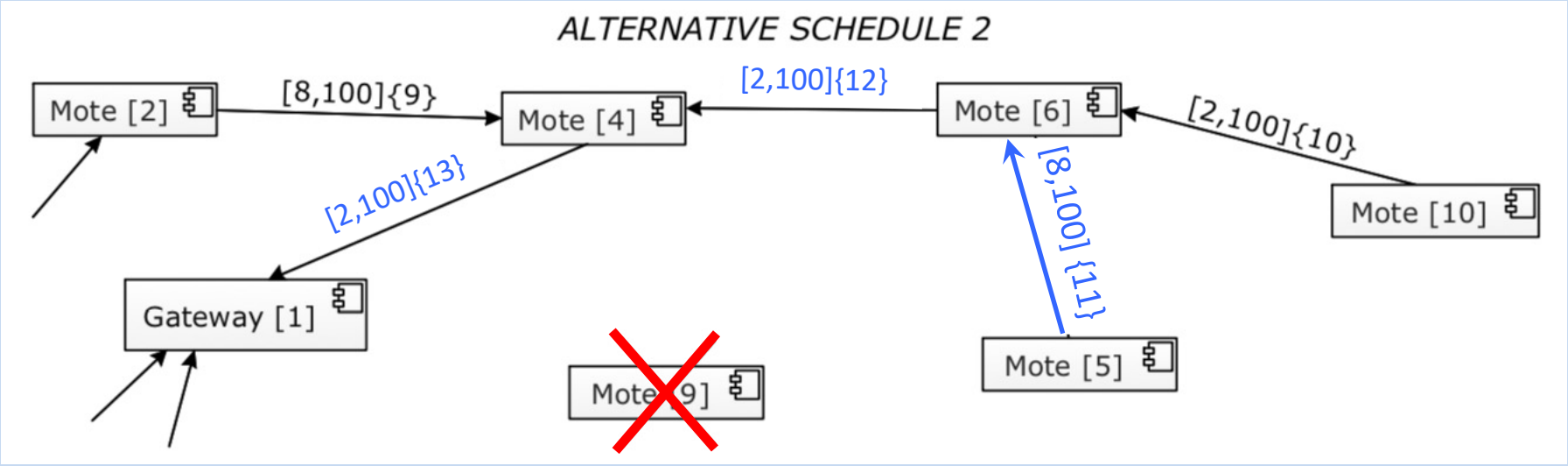


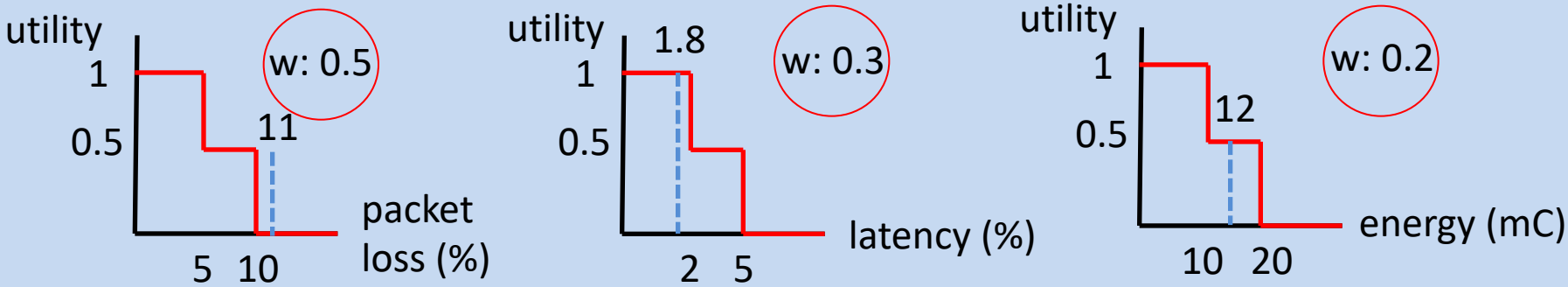


Alternative 1: expected utility = sum (w_i x u_i) => 0.5 x 0.5 + 0.3 x 0.5 + 0.2 x 0.5 = 0.50

Alternative 2

Self-healing

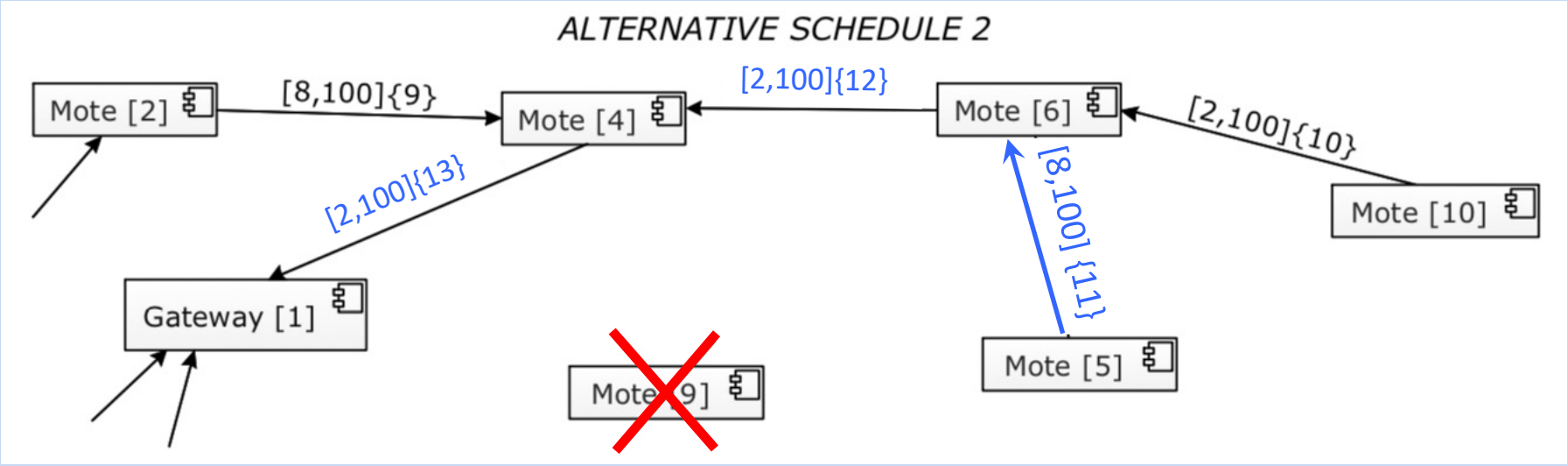


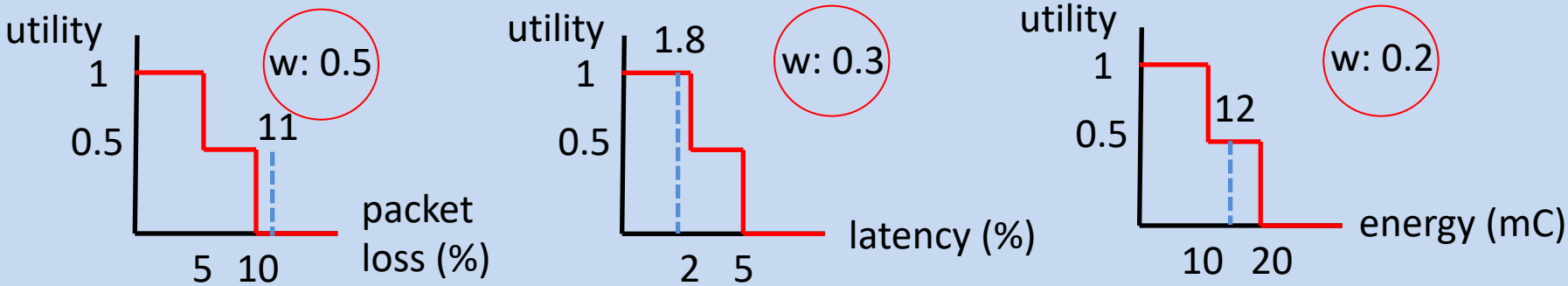


Alternative 1: expected utility = sum (w_i x u_i) => 0.5 x 0.5 + 0.3 x 0.5 + 0.2 x 0.5 = 0.50

Alternative 2: expected utility = ?

Self-healing

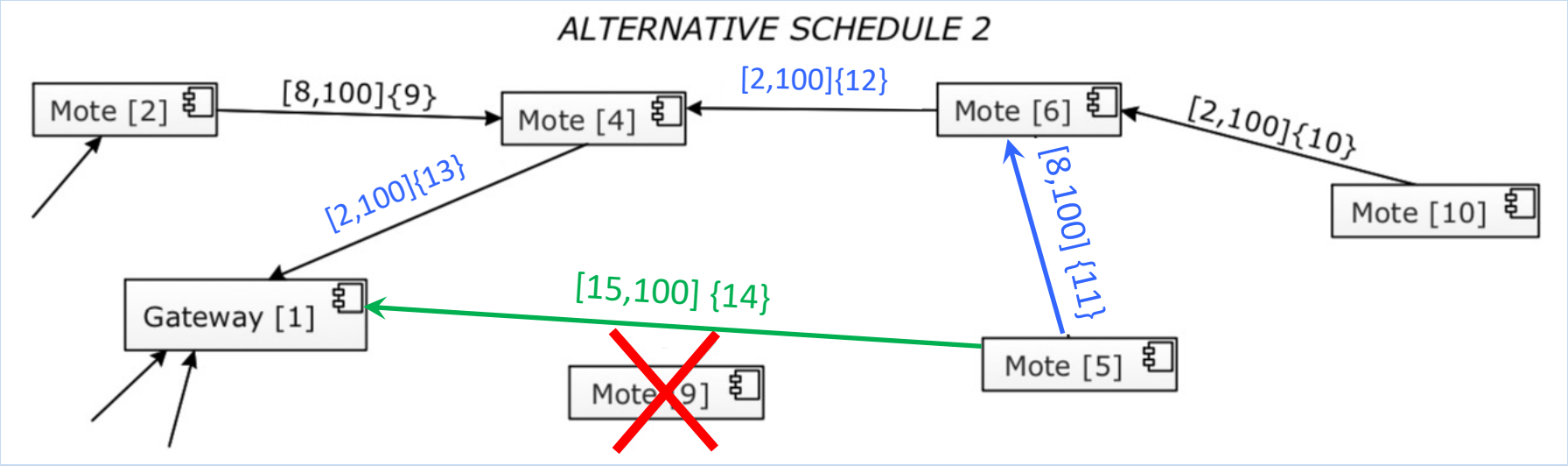


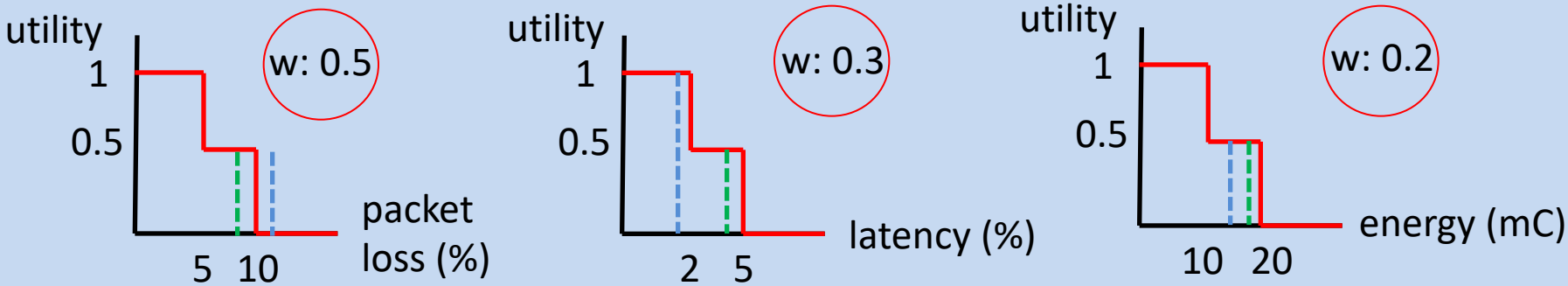


Alternative 1: expected utility = sum (w_i x u_i) => 0.5 x 0.5 + 0.3 x 0.5 + 0.2 x 0.5 = 0.50

Alternative 2: expected utility = sum (w_i x u_i) => 0.5 x 0 + 0.3 x 1.0 + 0.2 x 0.5 = 0.31

Self-healing





Alternative 1: expected utility = sum (w_i x u_i) => 0.5 x 0.5 + 0.3 x 0.5 + 0.2 x 0.5 = 0.50

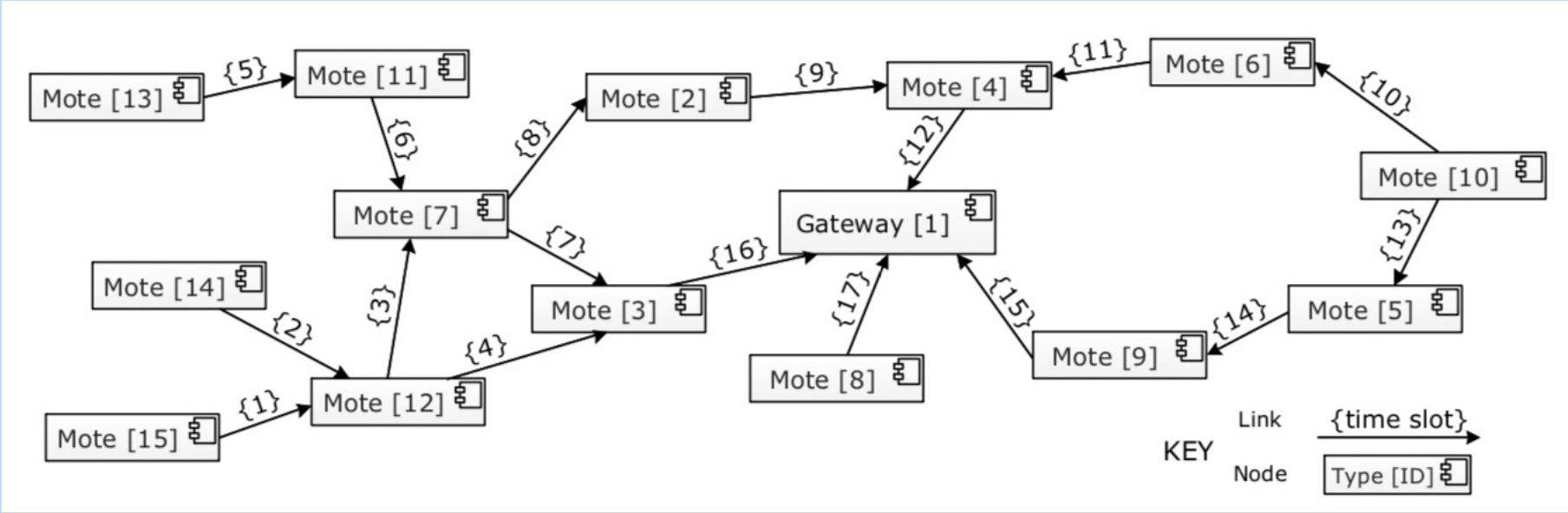
Alternative 2: expected utility = sum (w_i x u_i) => 0.5 x 0 + 0.3 x 1.0 + 0.2 x 0.5 = 0.31

=> switch to alternative 1

Essential maintenance tasks for automation

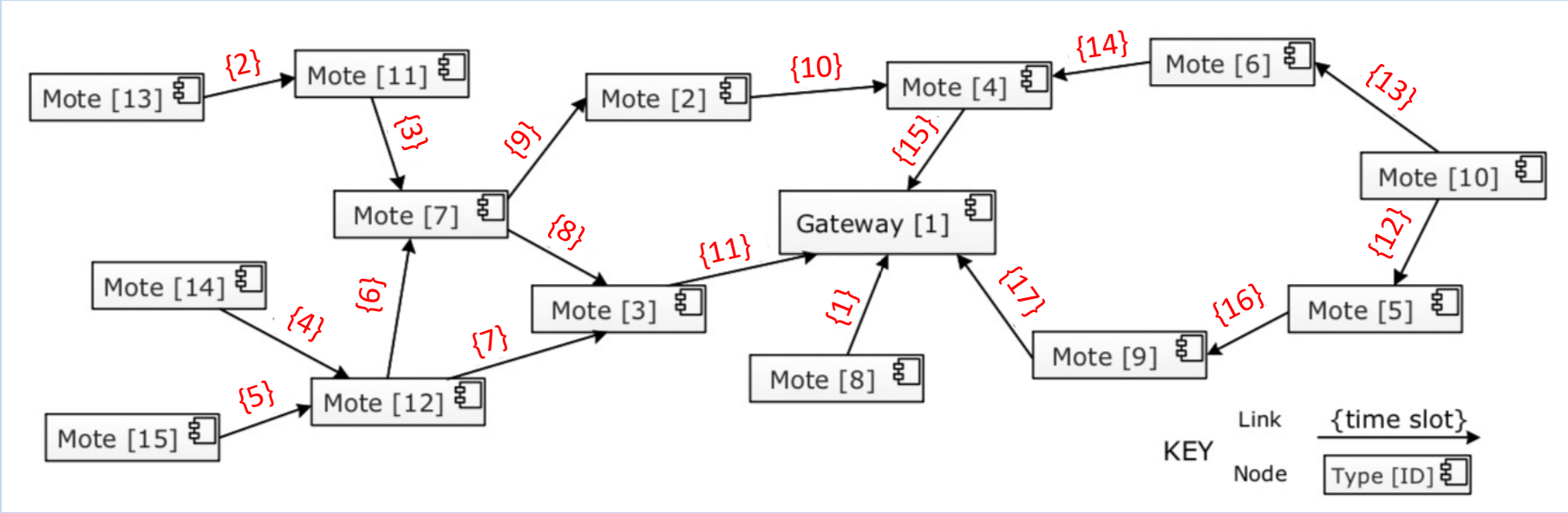
Self-protection = capability of system to defend against attacks and anticipate problems they may cause, to provide its required quality objectives.

Self-protection



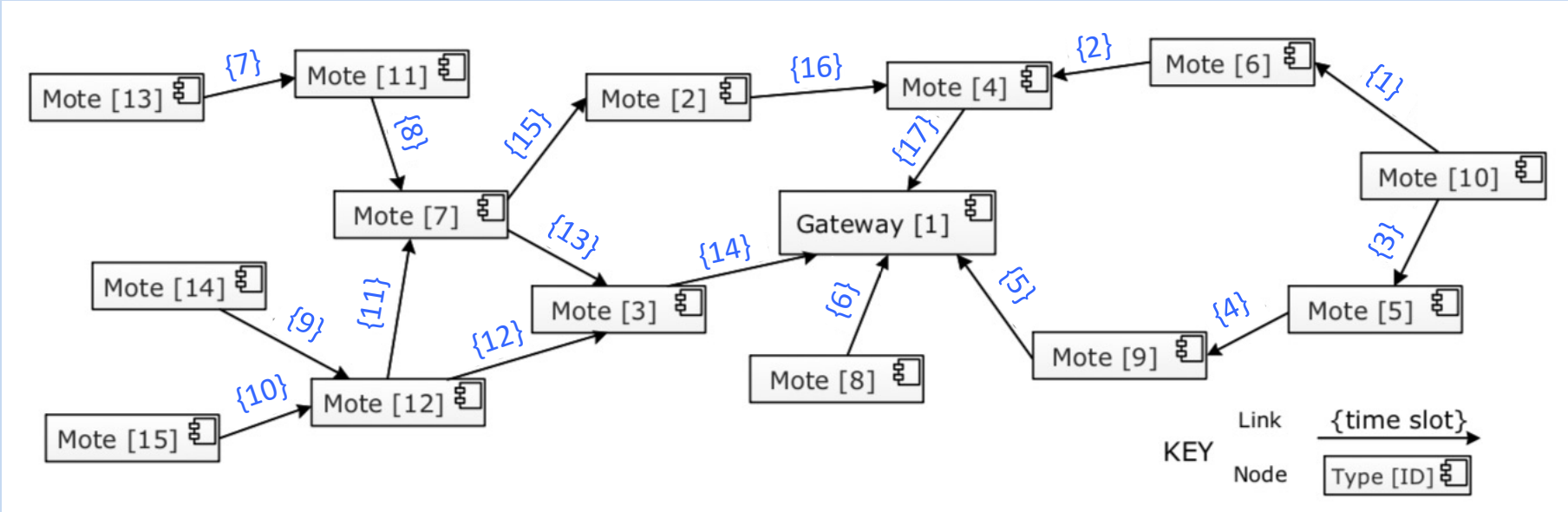
Attacker detect patterns in communication and exploits this to transmit spurious messages (jamming attack)

Self-protection



Attacker detect patterns in communication and exploits this to transmit spurious messages
Switch to alternative schedule when suspicious traffic is discovered

Self-protection



Attacker detect patterns in communication and exploits this to transmit spurious messages

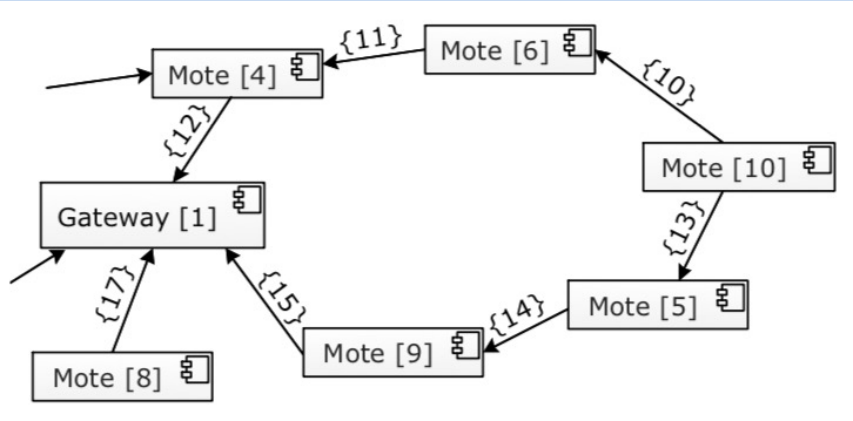
Switch to alternative schedule when suspicious traffic is discovered

Proactively switch to alternative schedule to anticipate attack

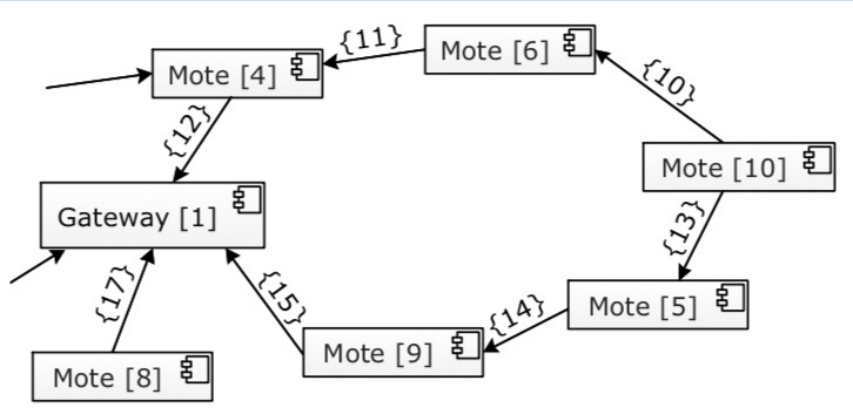
Essential maintenance tasks for automation

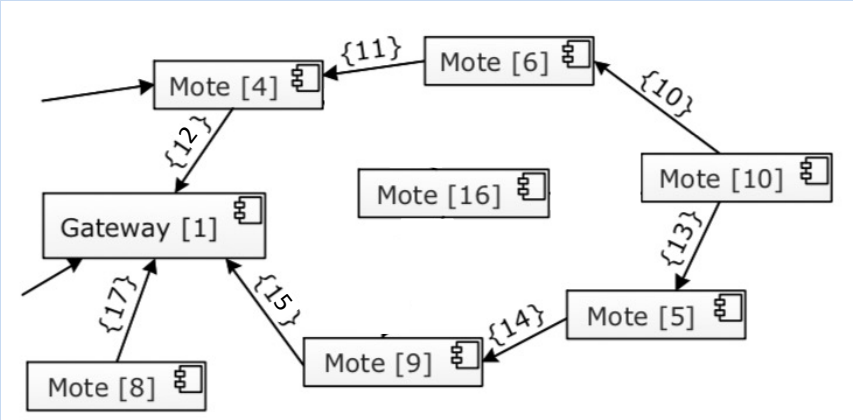
Self-configuration is the capability of a system to automatically integrate new elements, without interrupting the system's normal operation.

Self-configuration



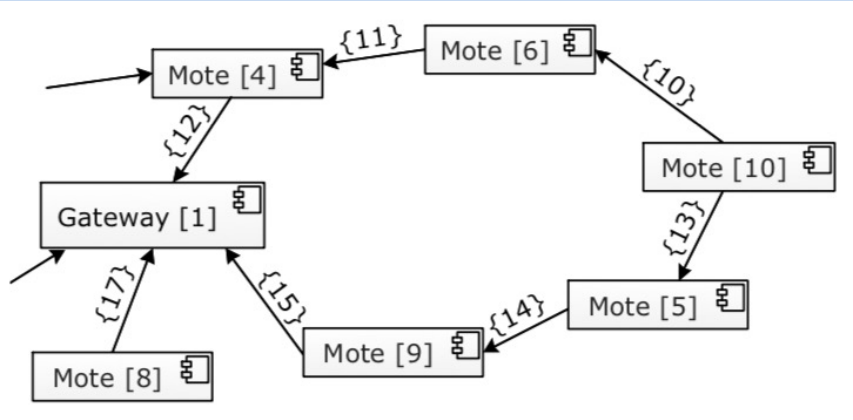
Self-configuration

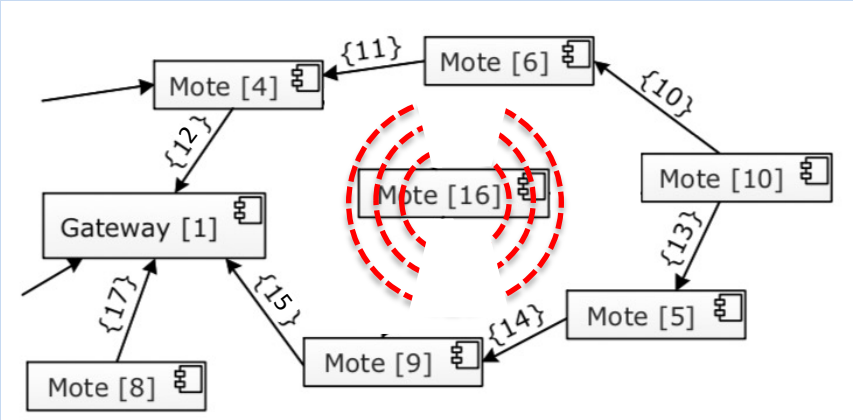




New mote 16 arrives

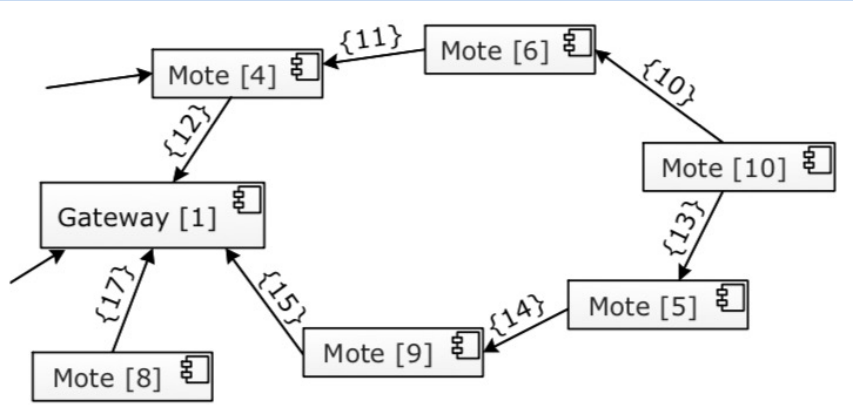
Self-configuration

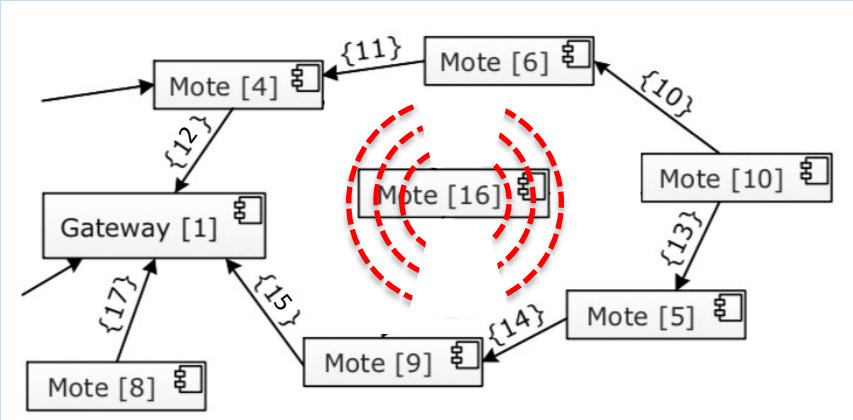




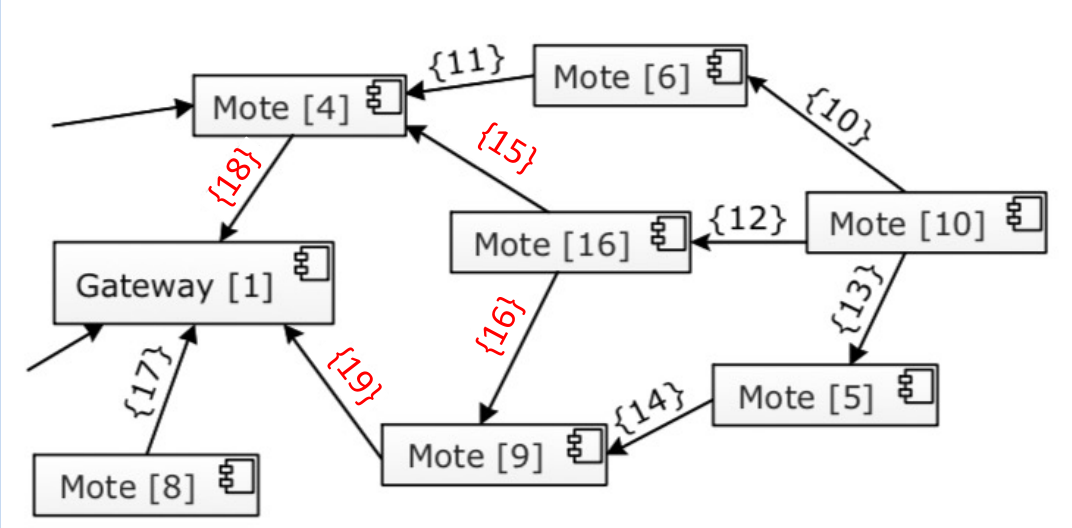
Mote 16 emits discovery message

Self-configuration





Mote 16 emits discovery message



Mote 16 integrated in the network

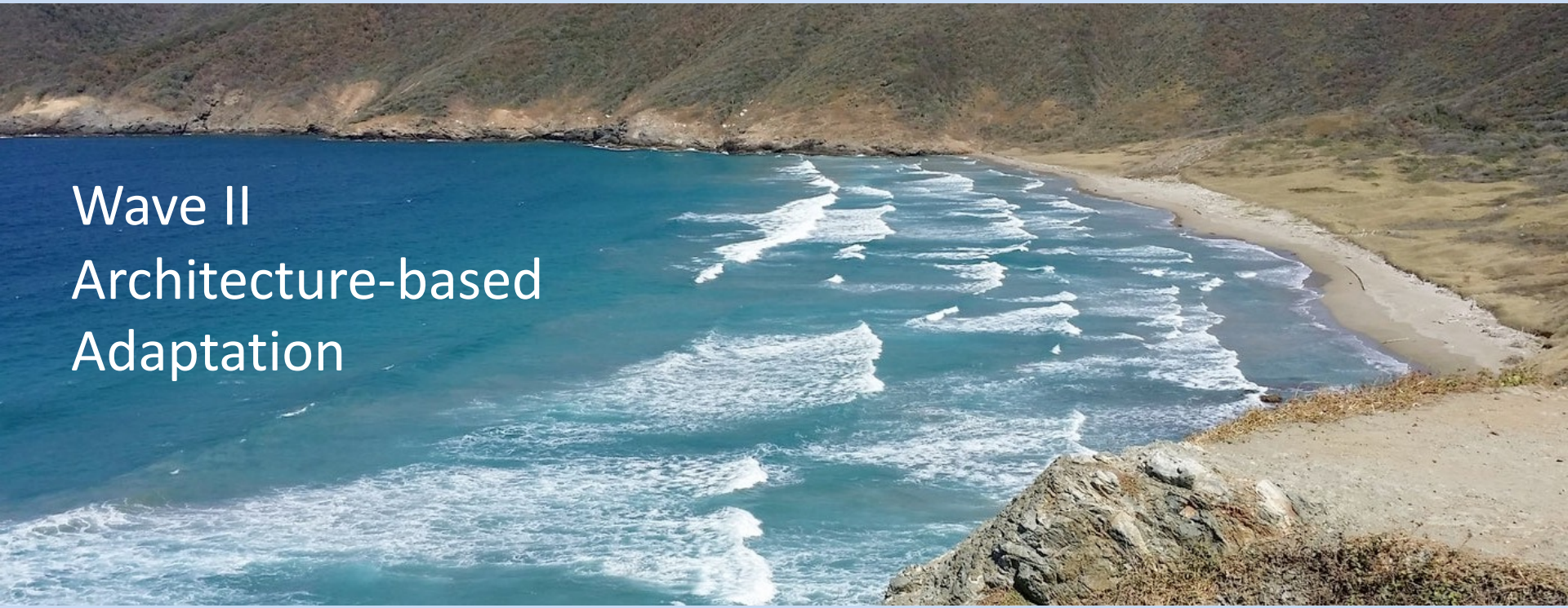
Key insights Wave I

Automating tasks is a key driver for self-adaptation

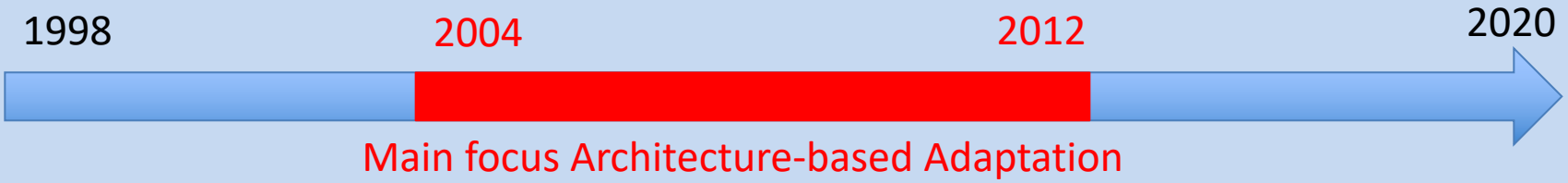
MAPE-K as reference model for a managing system

Four essential types of maintenance tasks:

- self-optimization
- self-healing
- self-protection
- self-configuration.



Wave II Architecture-based Adaptation



Driver: Lack of principled engineering approach

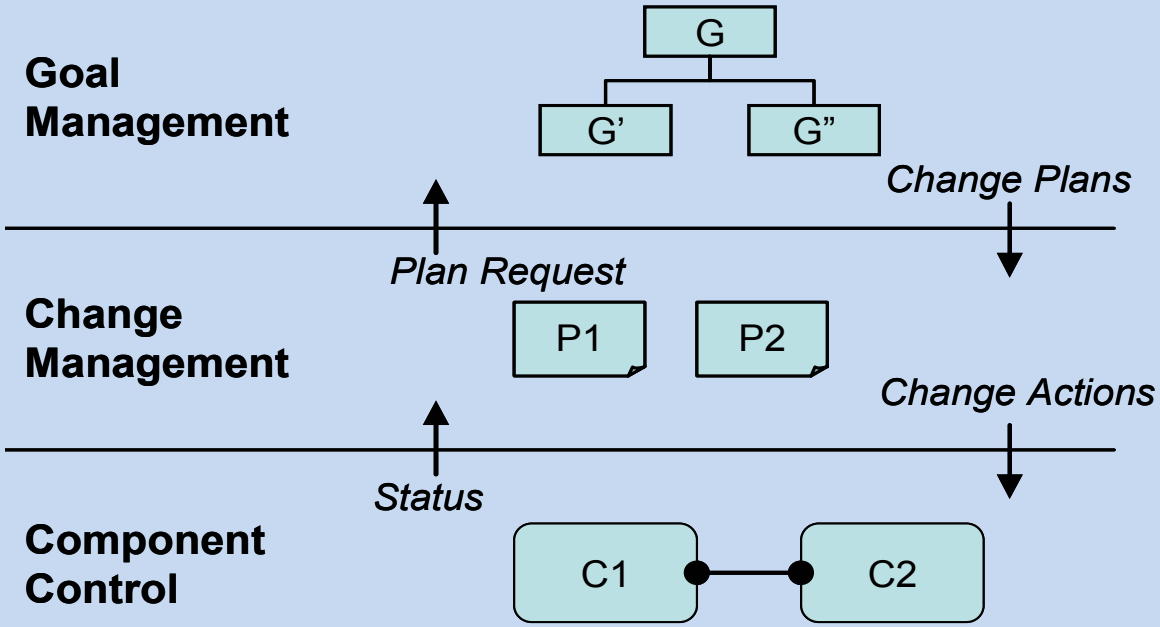
Topic: Architecture perspective on self-adaptation

Enabled:

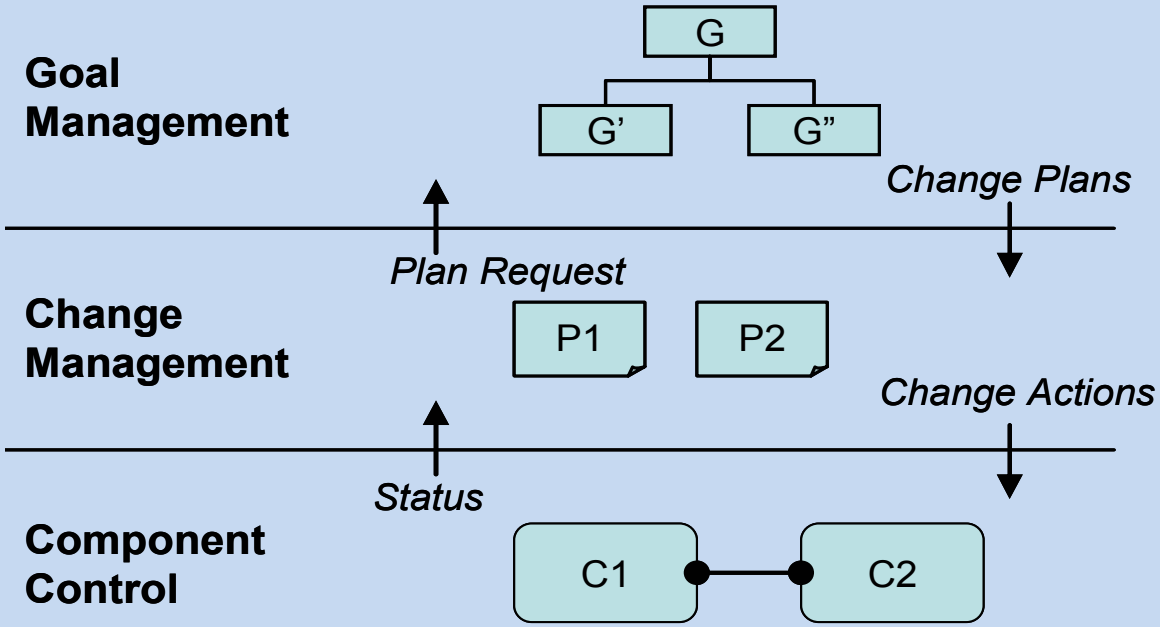
Architecture driver to define self-adaptive systems and to reason about adaptation at runtime

Architecture driver to define self-adaptive systems

3-Layer model

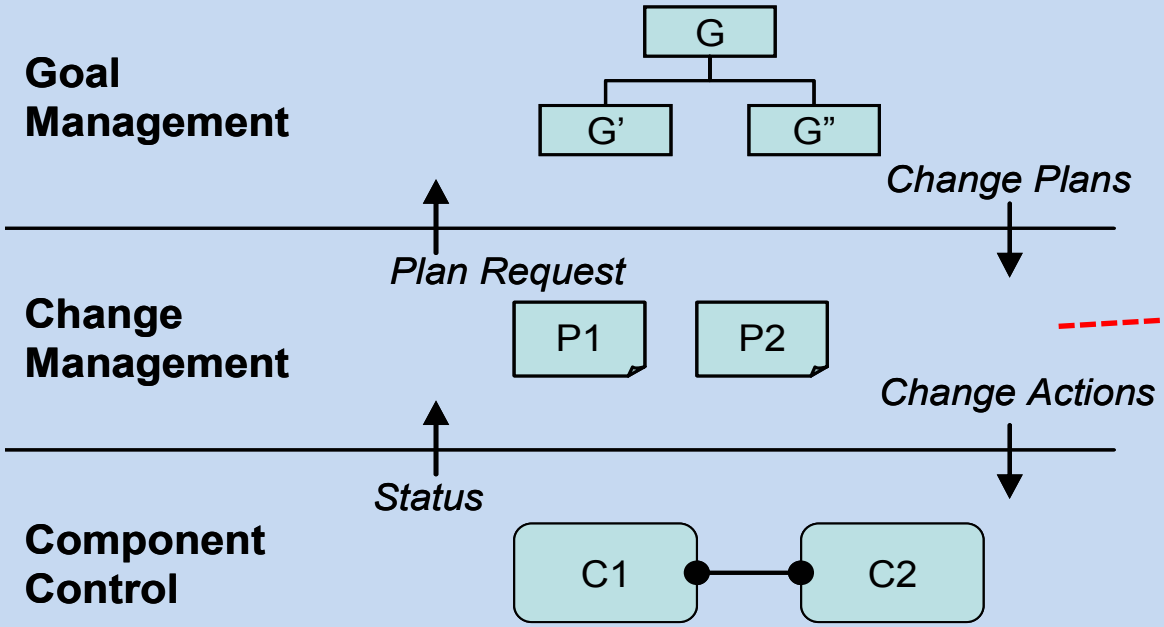


3-Layer model



Accomplishes functionality of application

3-Layer model

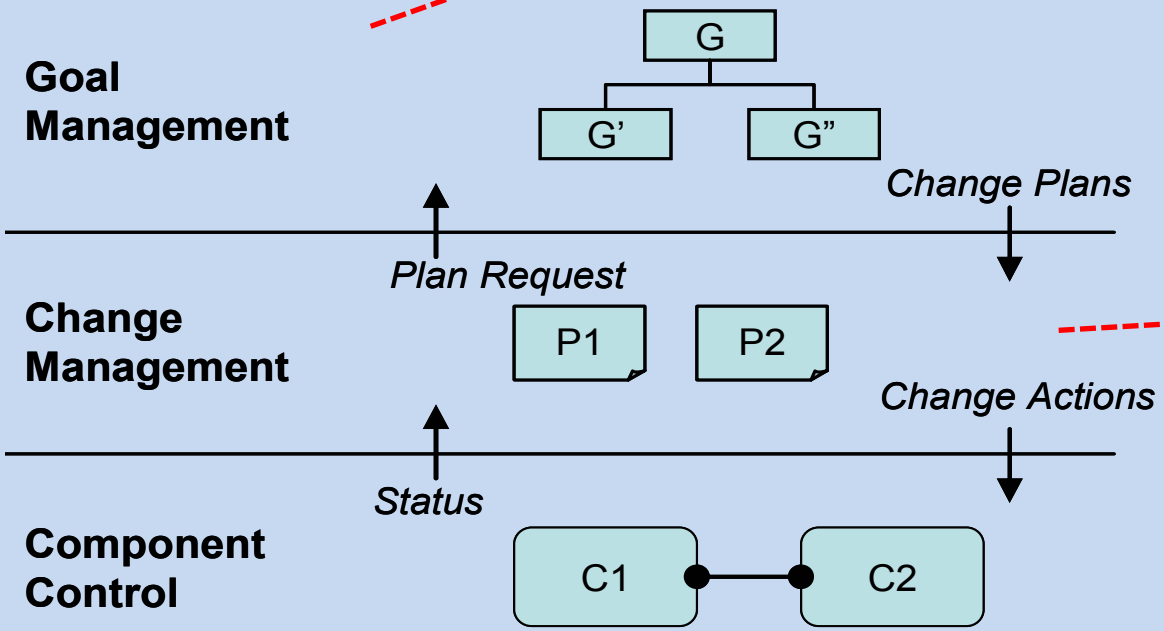


Executes changes in lower layer based on status changes

Accomplishes functionality of application

3-Layer model

Re-plans and deals with changing or new goals

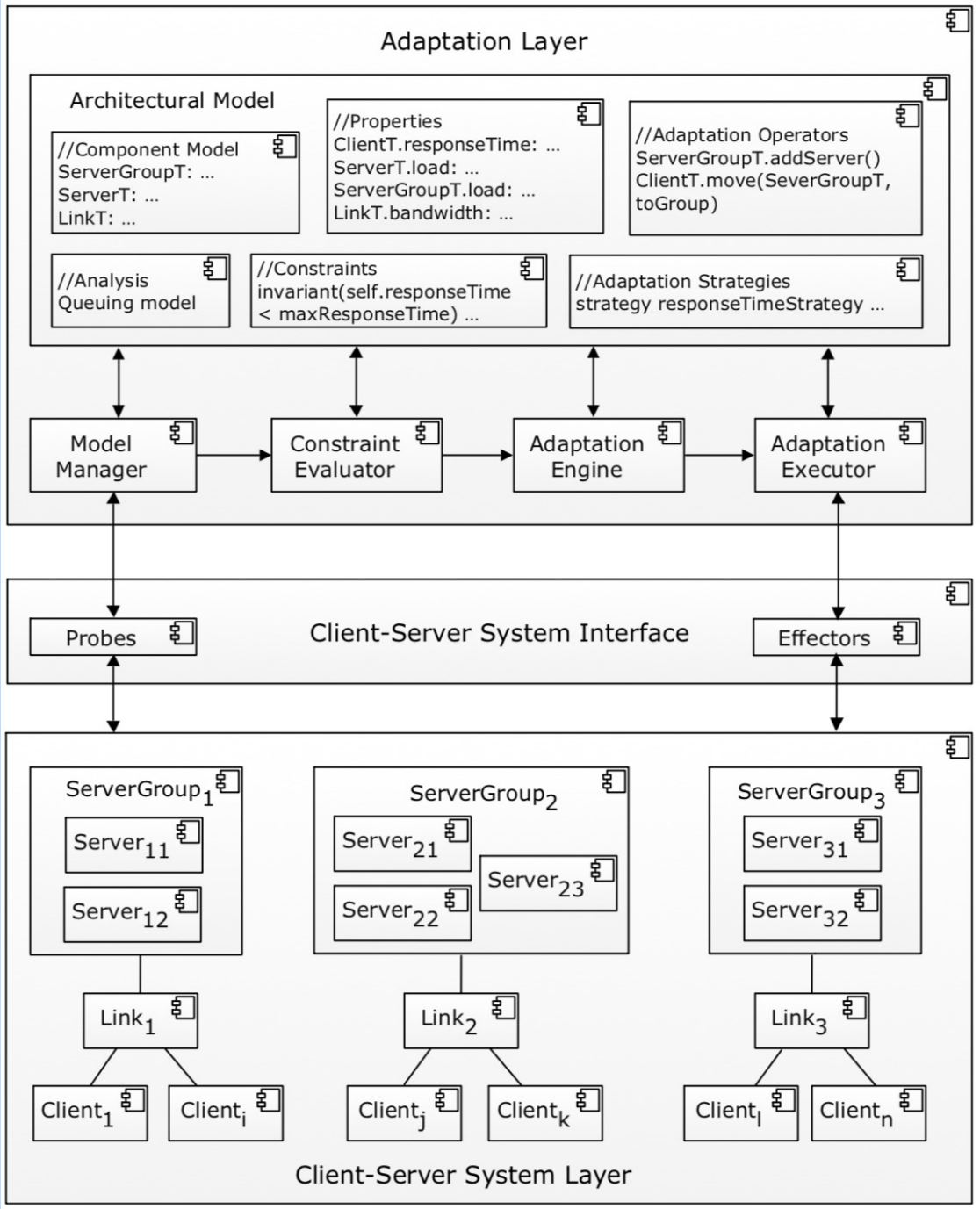


Executes change plans in lower layer based on status changes

Accomplishes functionality of application

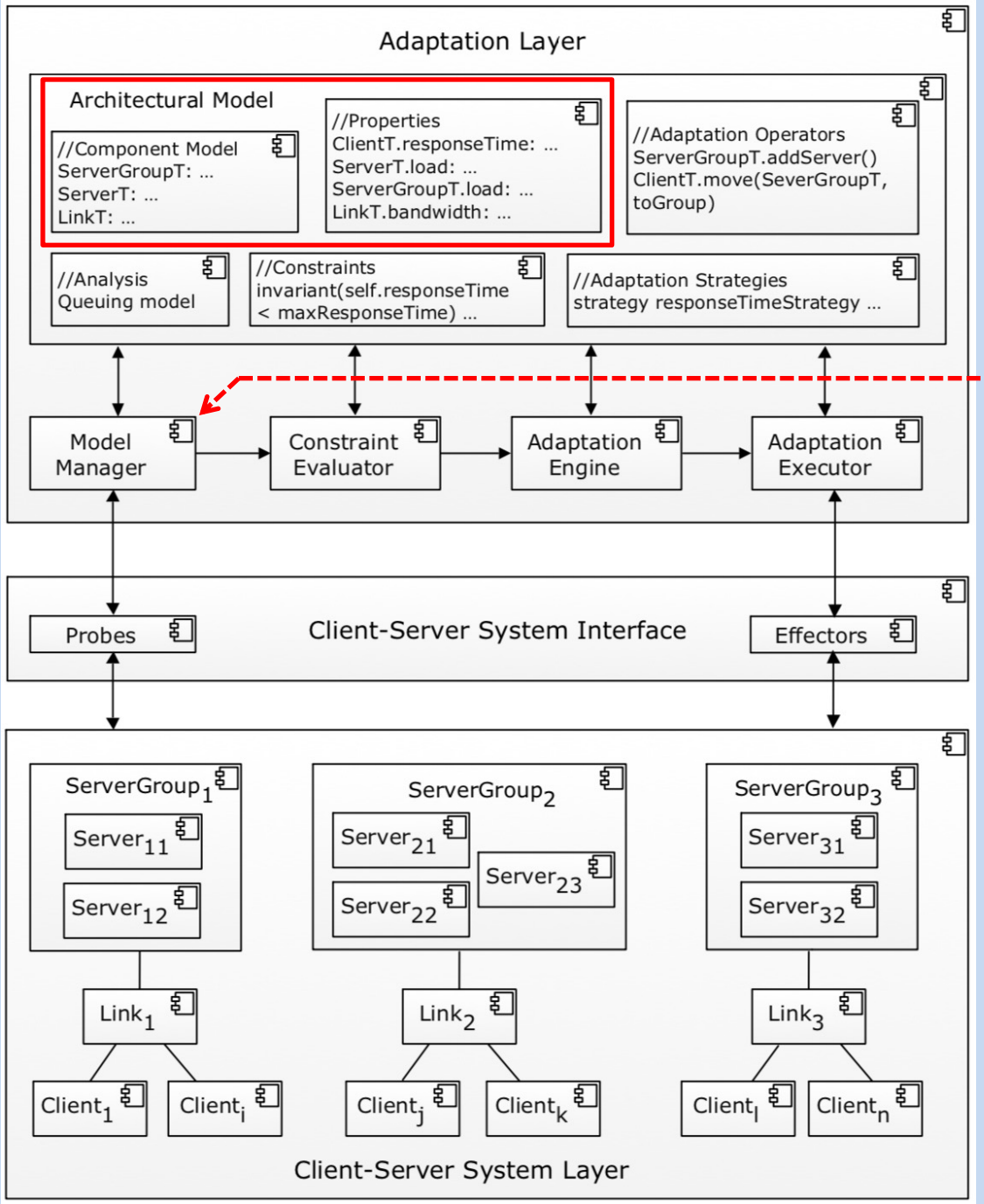
Architecture model to reason about self-adaptation at runtime

The Rainbow framework



Goal: keep the response time below predefined maximum

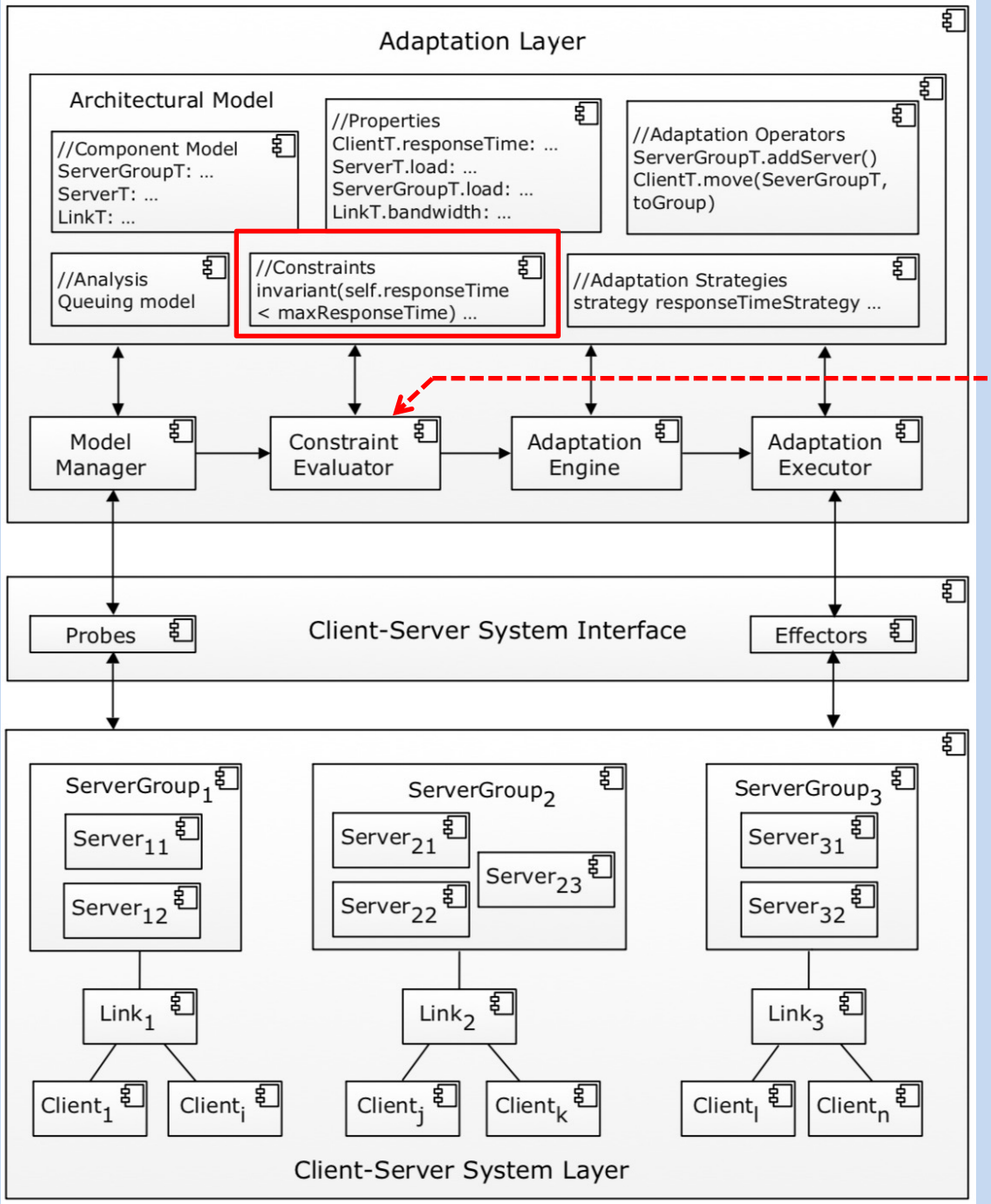
The Rainbow framework



Maintains model of client-server system (response time, bandwidth, …)

Goal: keep the response time below predefined maximum

The Rainbow framework

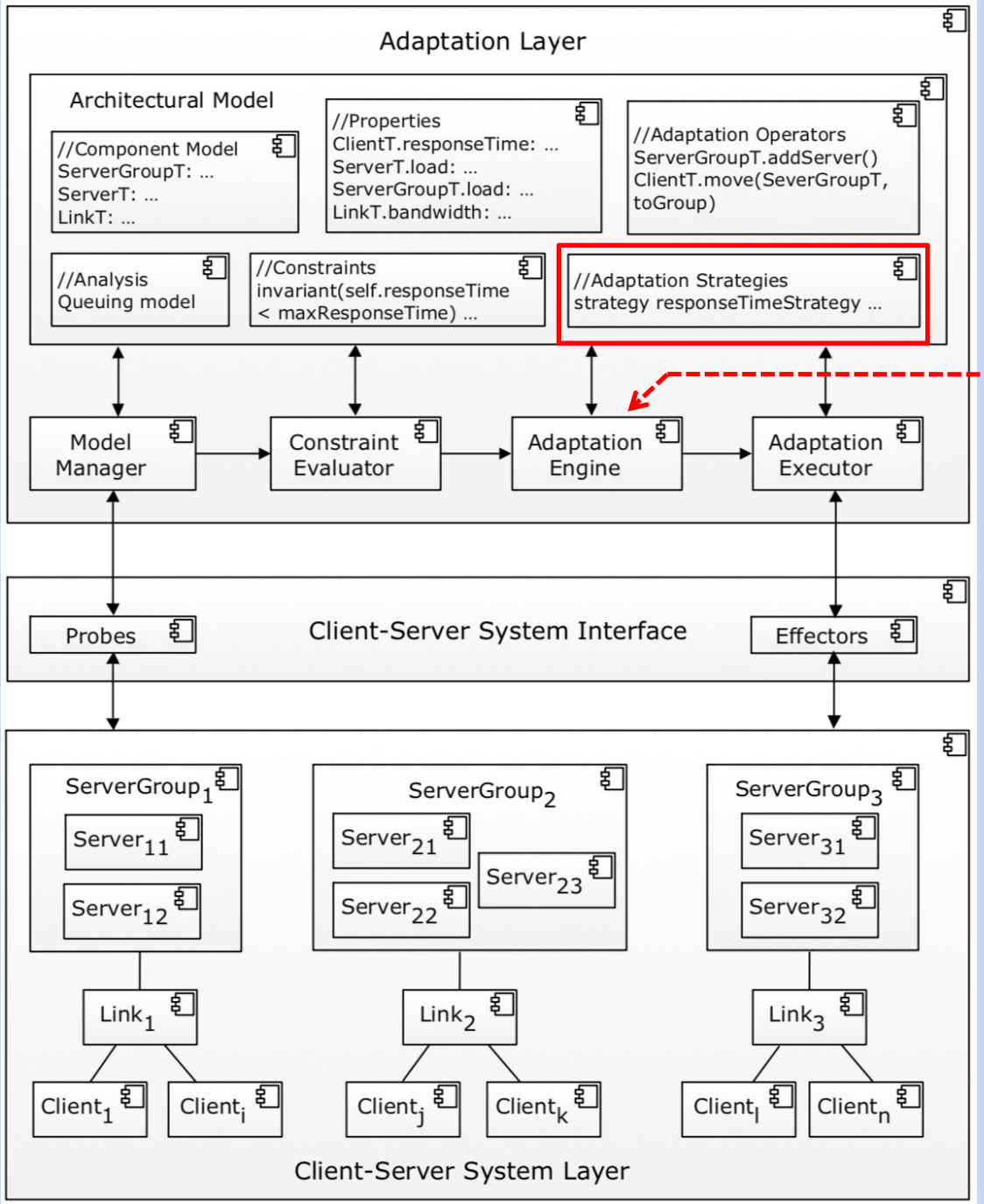


Periodically check response time

If violation -> trigger adaptation engine

Goal: keep the response time below predefined maximum

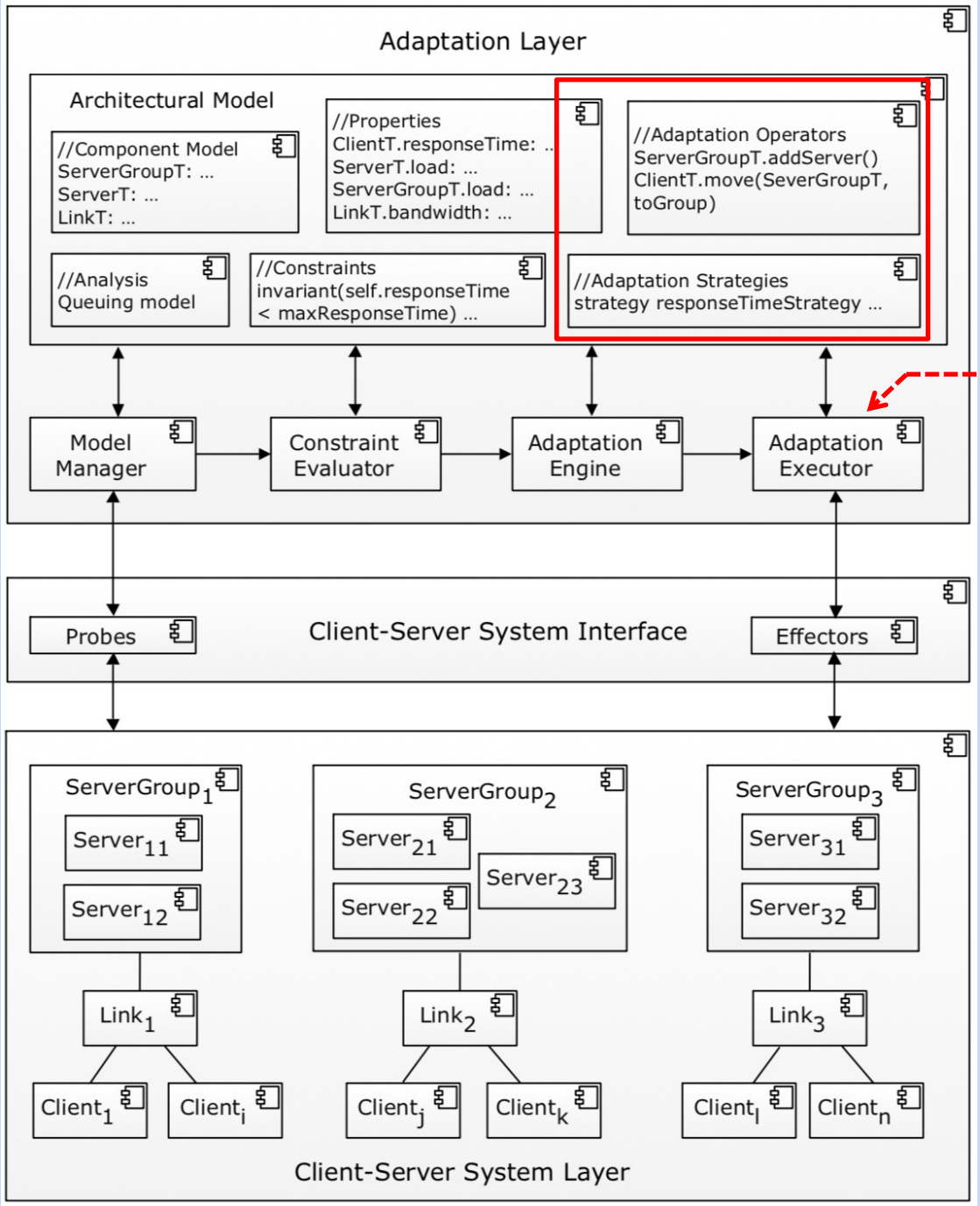
The Rainbow framework



Applies adaptation strategy:
(i) load group too high -> add server;
(ii) available bandwidth of client too low -> move client to other group

Goal: keep the response time below predefined maximum

The Rainbow framework



Executes actions of selected adaptation strategy

Goal: keep the response time below predefined maximum

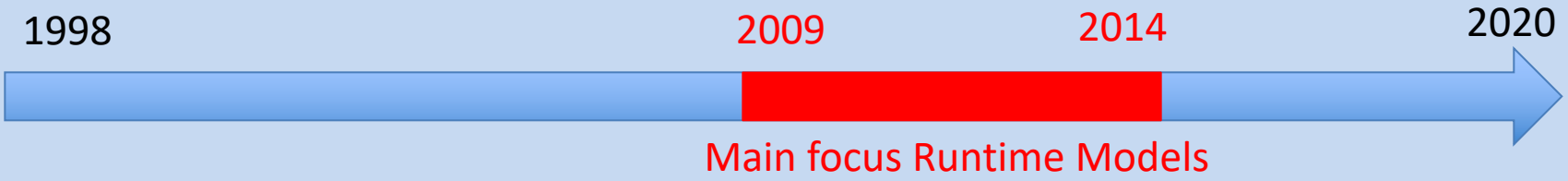
Key insights Wave II

Fundamental architectural concerns of self-adaptation:

- change management
- goal management

Architectural model offers appropriate representation of system to make adaptation decisions at runtime

Wave III
Runtime Models



Driver: Concrete realization self-adaptation is complex

Topic: Model-driven approach extended to runtime to realize self-adaptation

Enabled: Different types of runtime models to support runtime decision-making

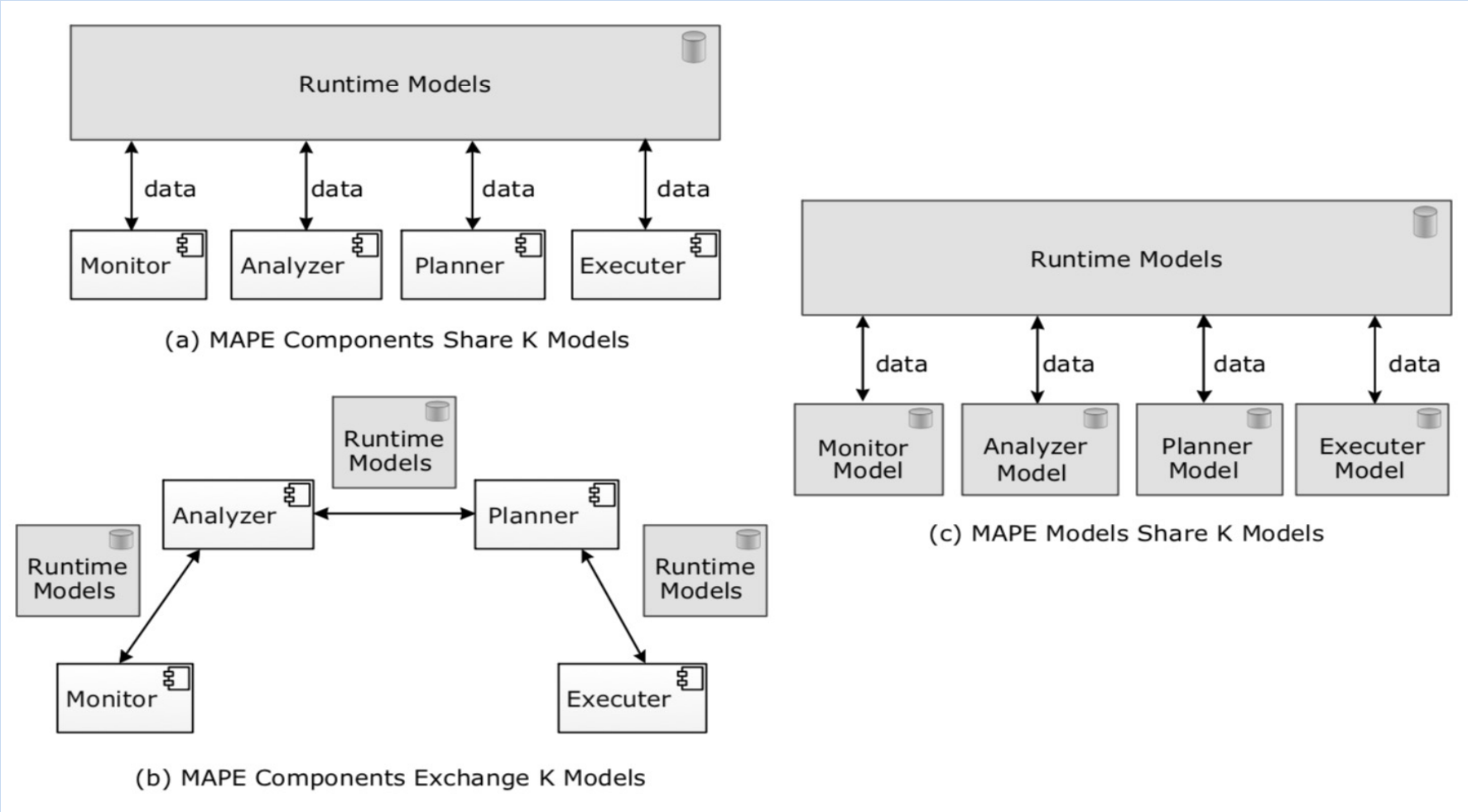
Runtime model

A runtime model is a first-class runtime abstraction of that system or any aspect related to that system that is used for the purpose of realizing self-adaptation

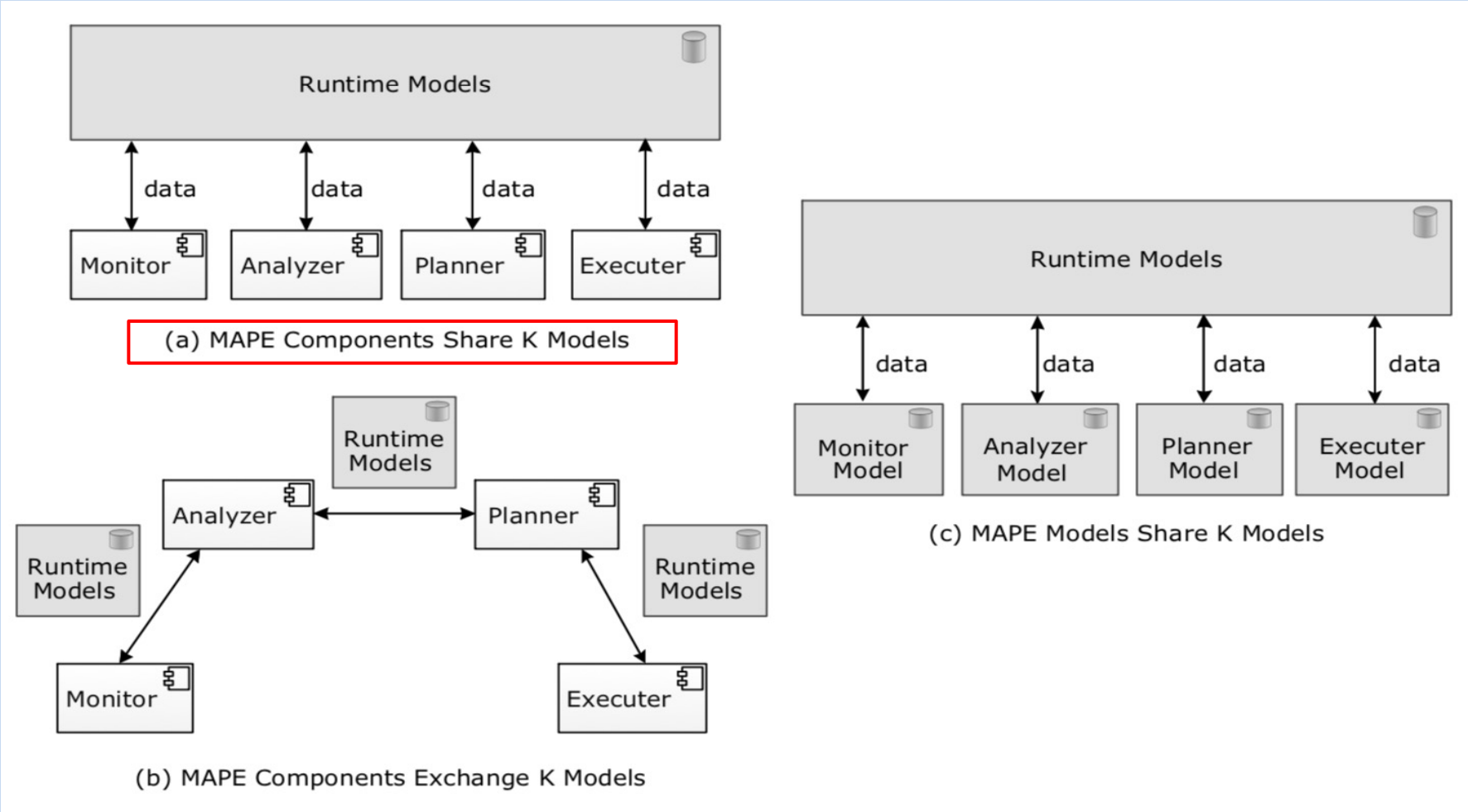
Weak causal connection

Weak causal connection allows a discrepancy between the state of a runtime model and the state of the system that does not invalidate the adaptation decisions

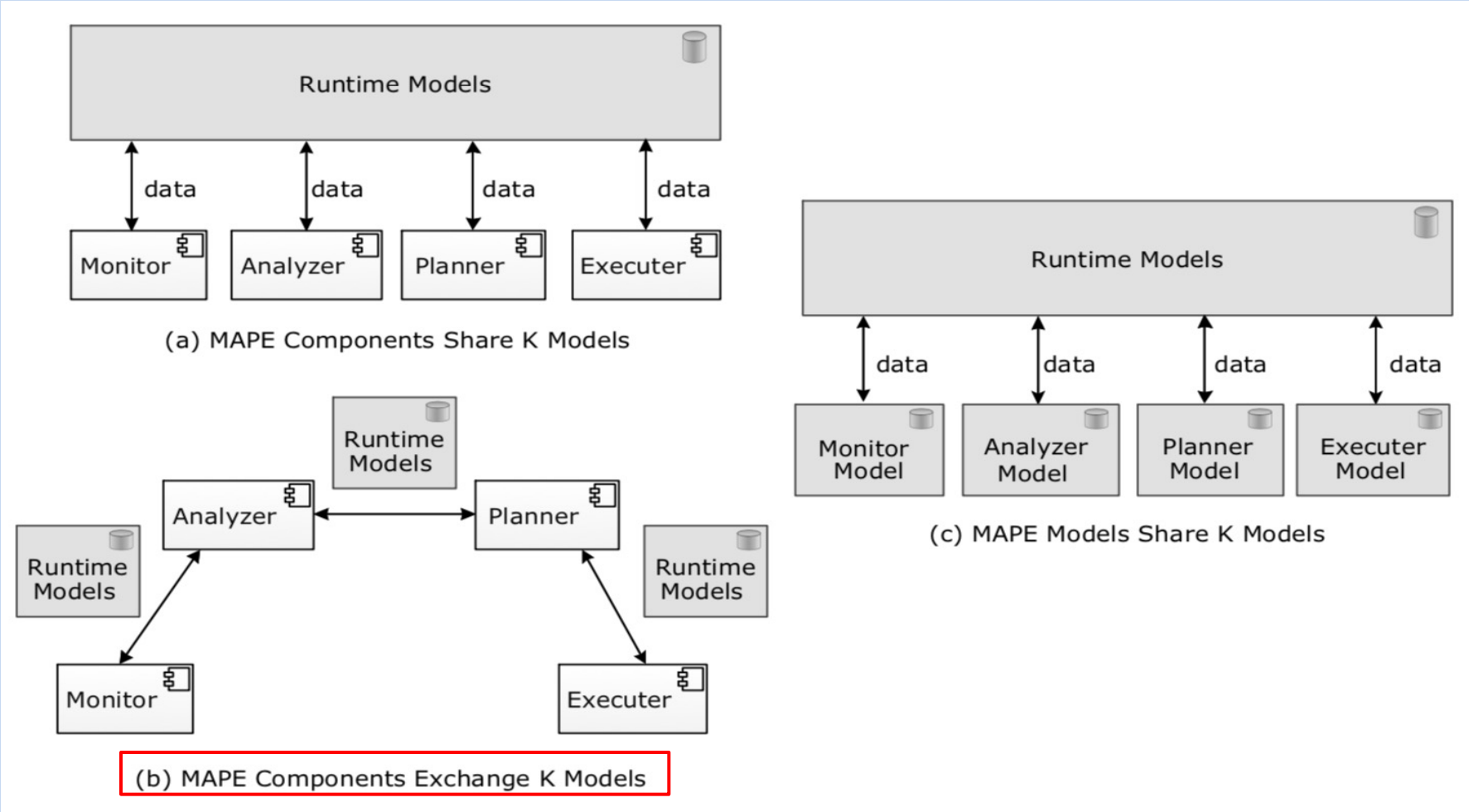
Principle Strategies to Use Runtime Models



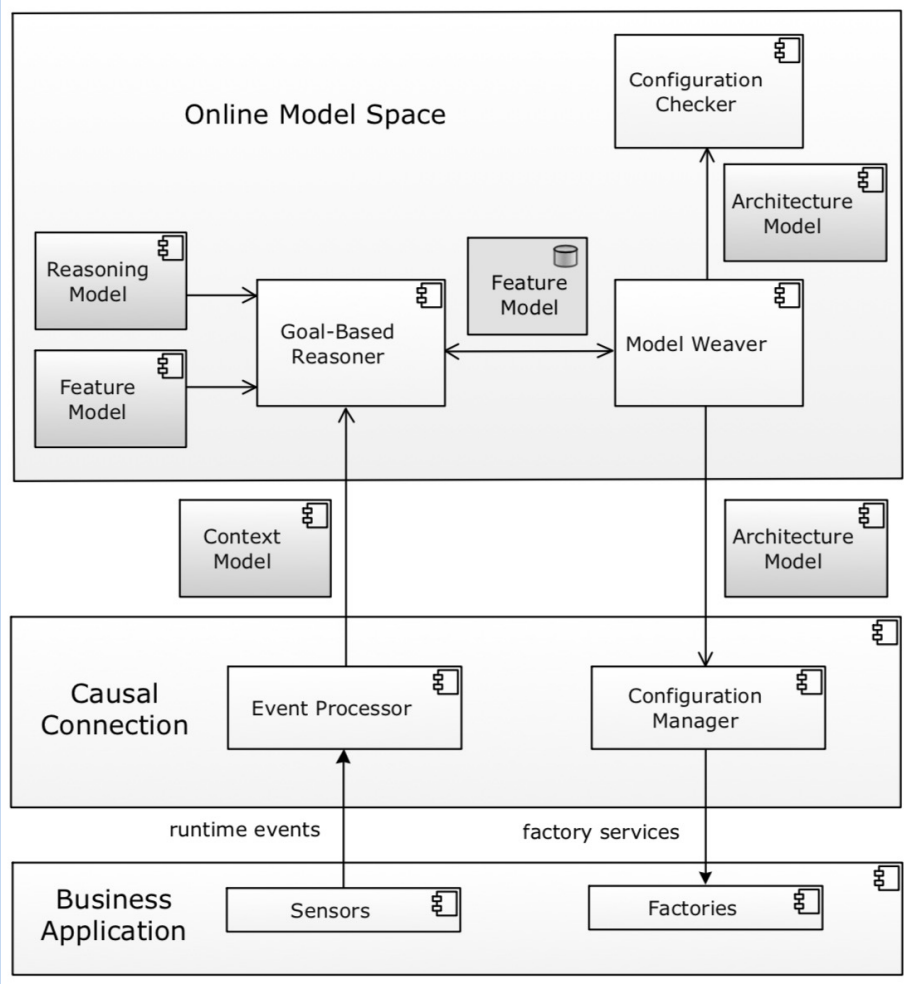
Principle Strategies to Use Runtime Models



Principle Strategies to Use Runtime Models

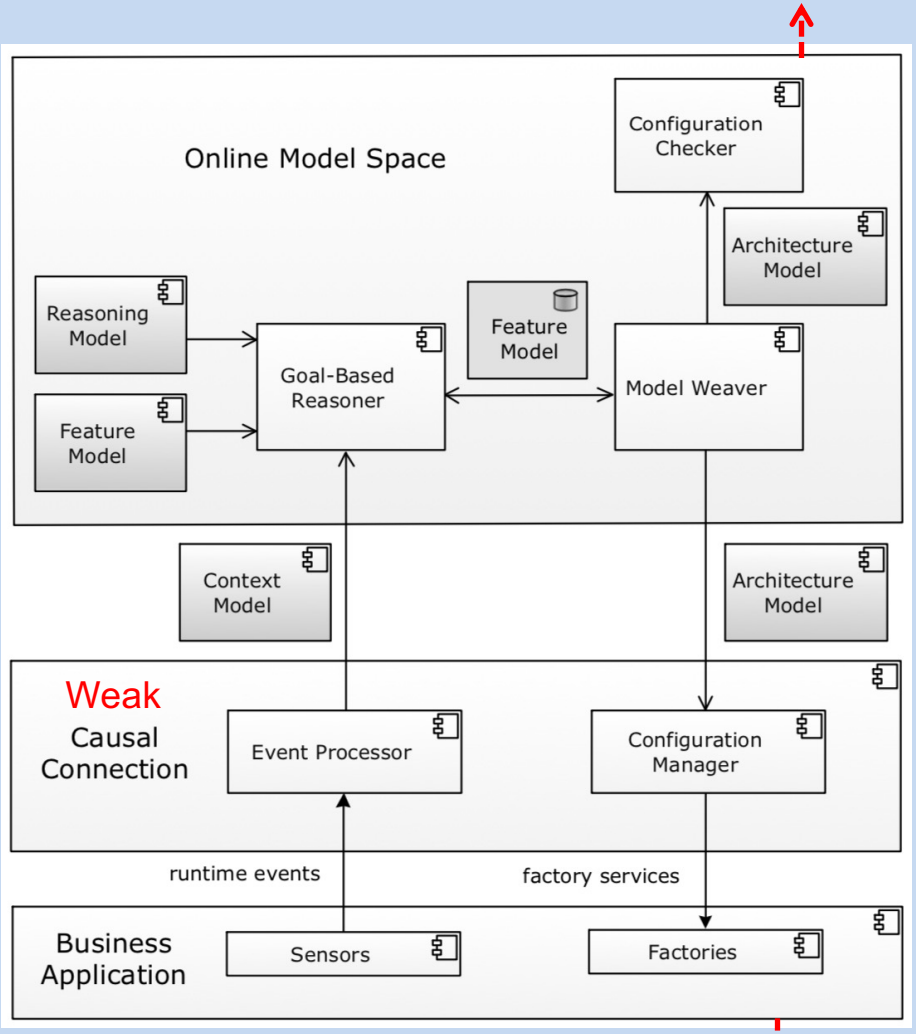


MAPE components exchange K models



MAPE components exchange K models

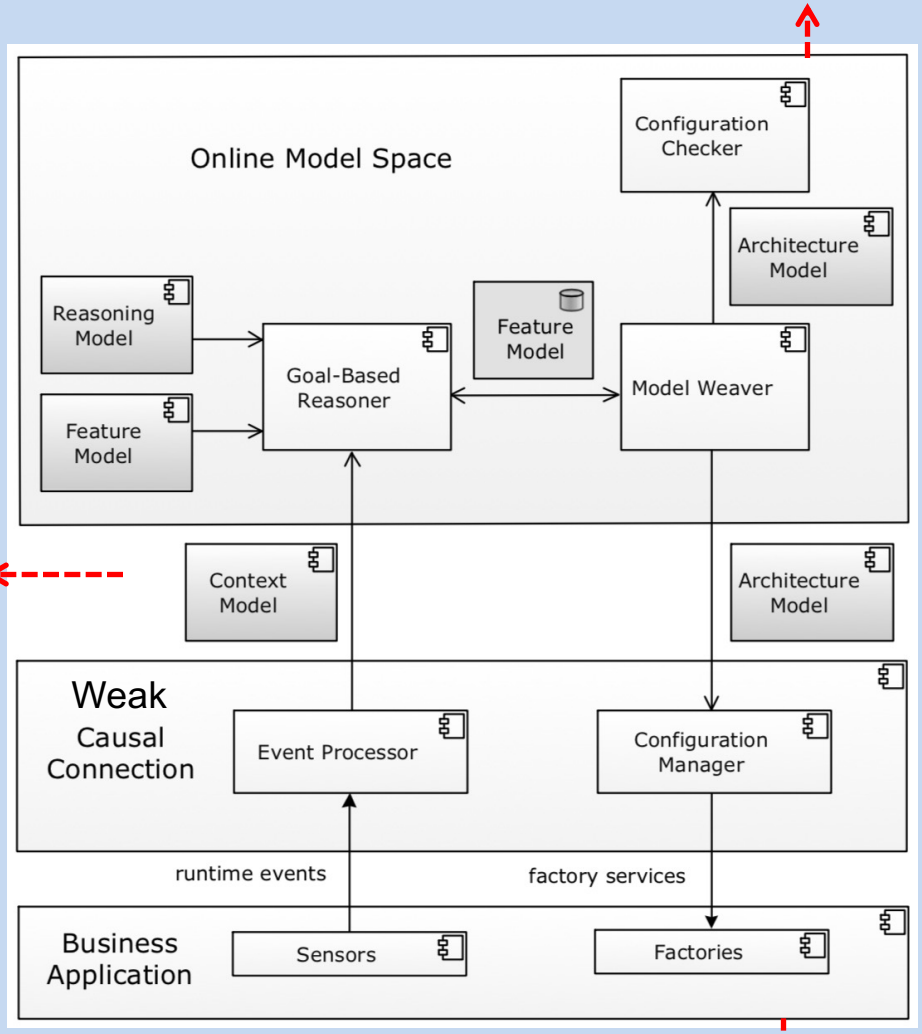
Managing system that reasons over models



Managed system instrumented with probes and effectors

MAPE components exchange K models

Managing system that reasons over models

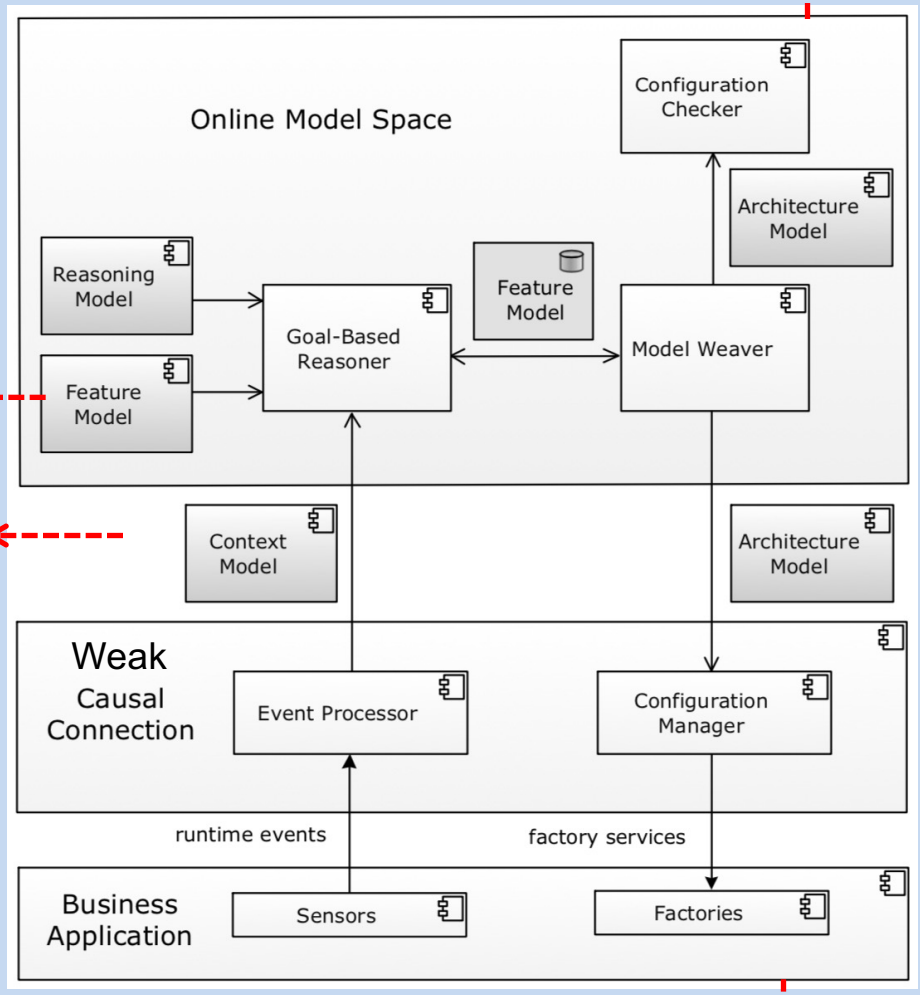


Relevant elements of the environment

Managed system instrumented with probes and effectors

MAPE components exchange K models

Managing system that reasons over models



Variability of the system

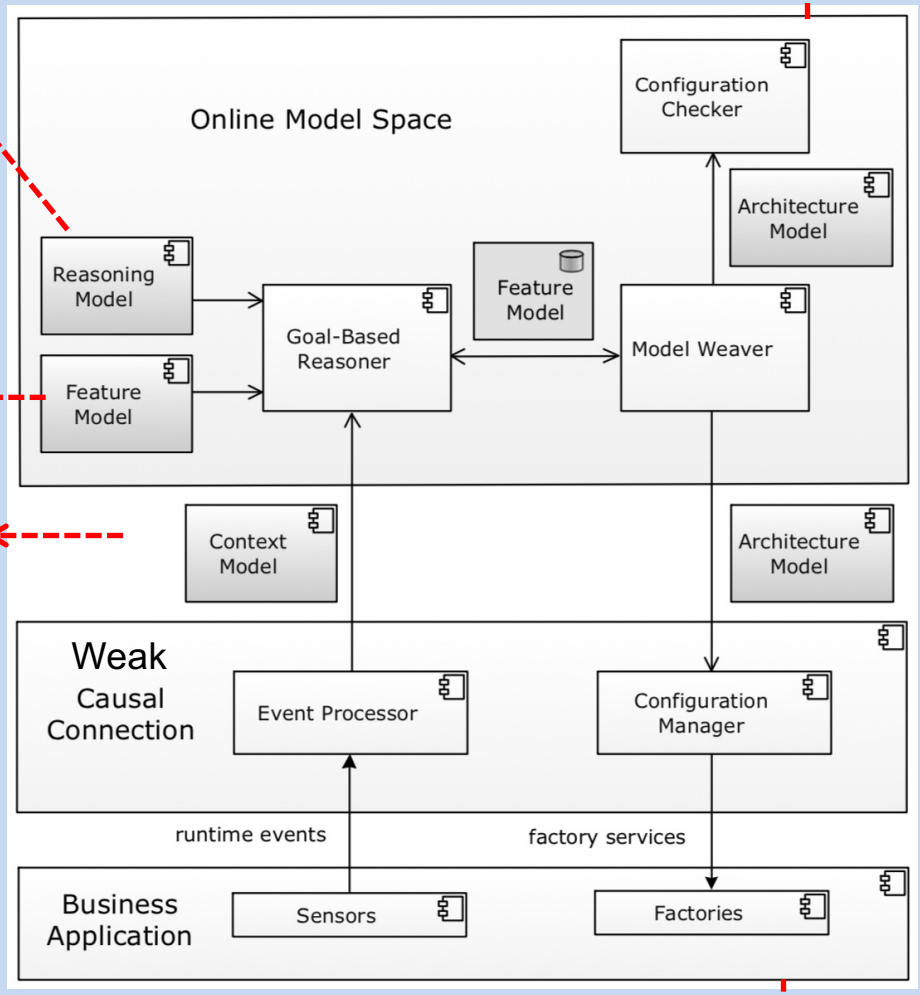
Relevant elements of the environment

Managed system instrumented with probes and effectors

MAPE components exchange K models

Managing system that reasons over models

Associates features with context



Variability of the system

Relevant elements of the environment

Managed system instrumented with probes and effectors

MAPE components exchange K models

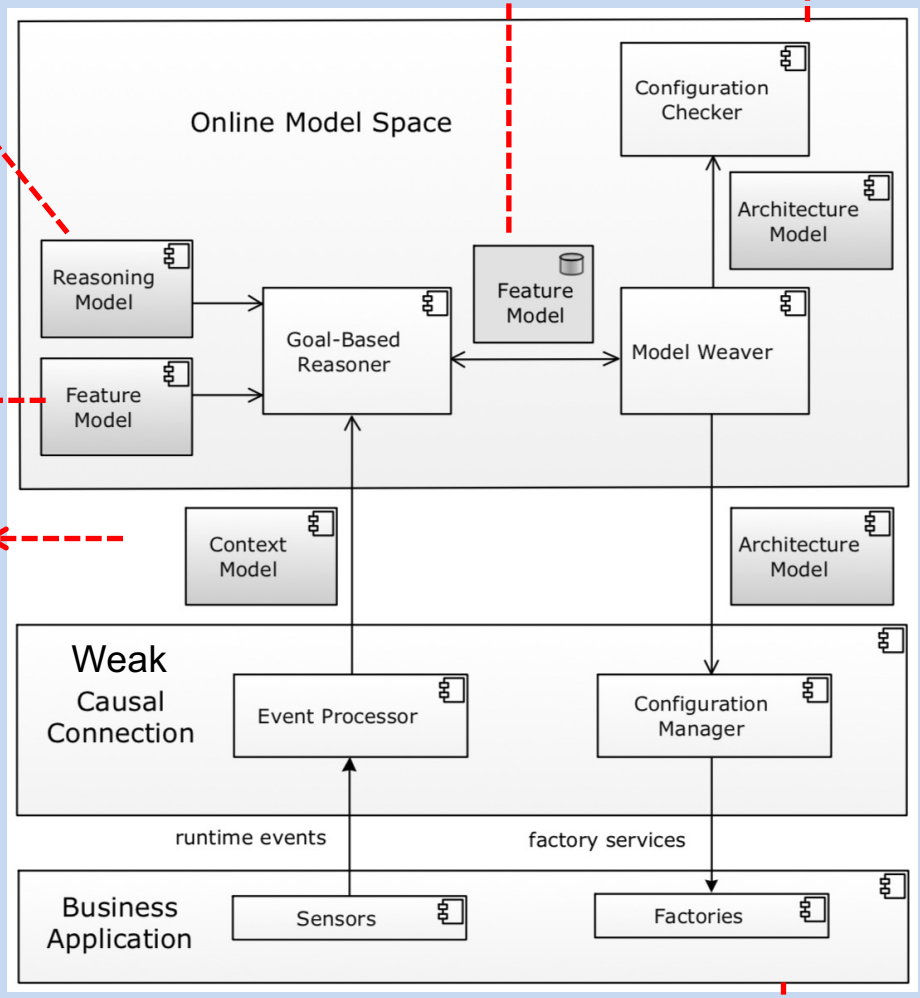
Associates features with context

Features for current context

Managing system that reasons over models

Variability of the system

Relevant elements of the environment



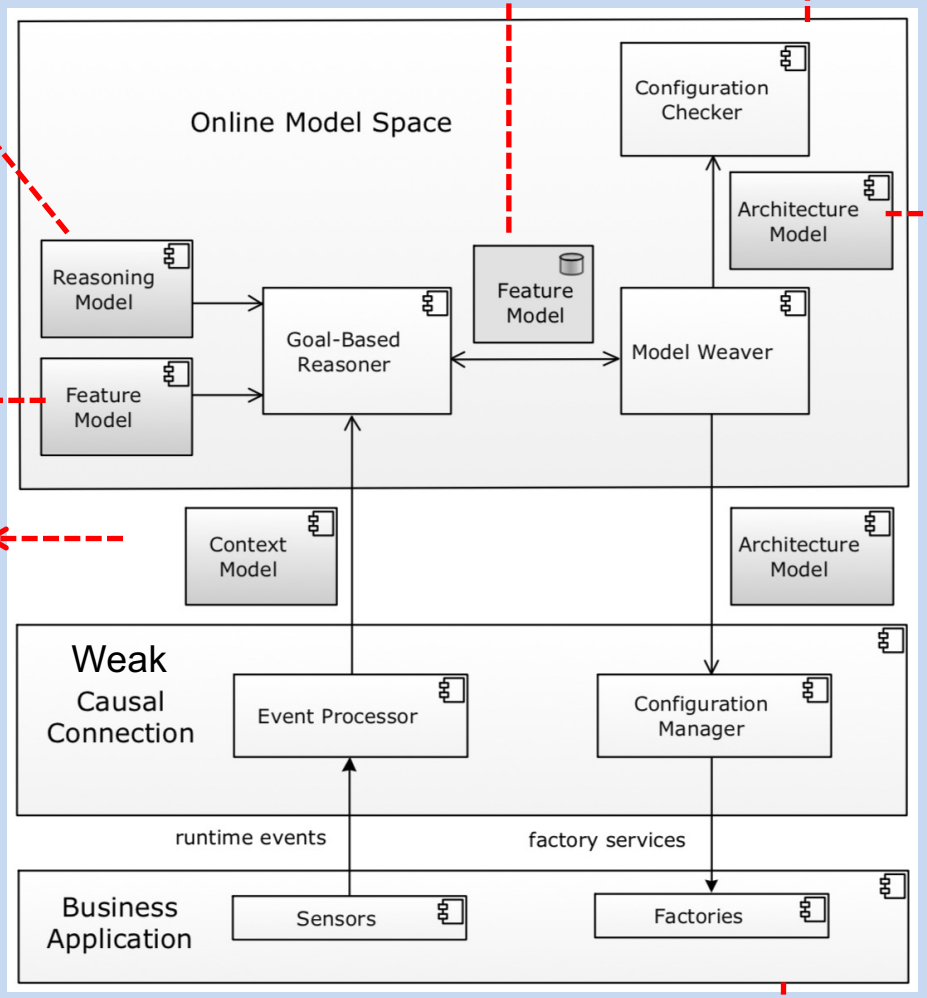
Managed system instrumented with probes and effectors

MAPE components exchange K models

Associates features with context

Features for current context

Managing system that reasons over models



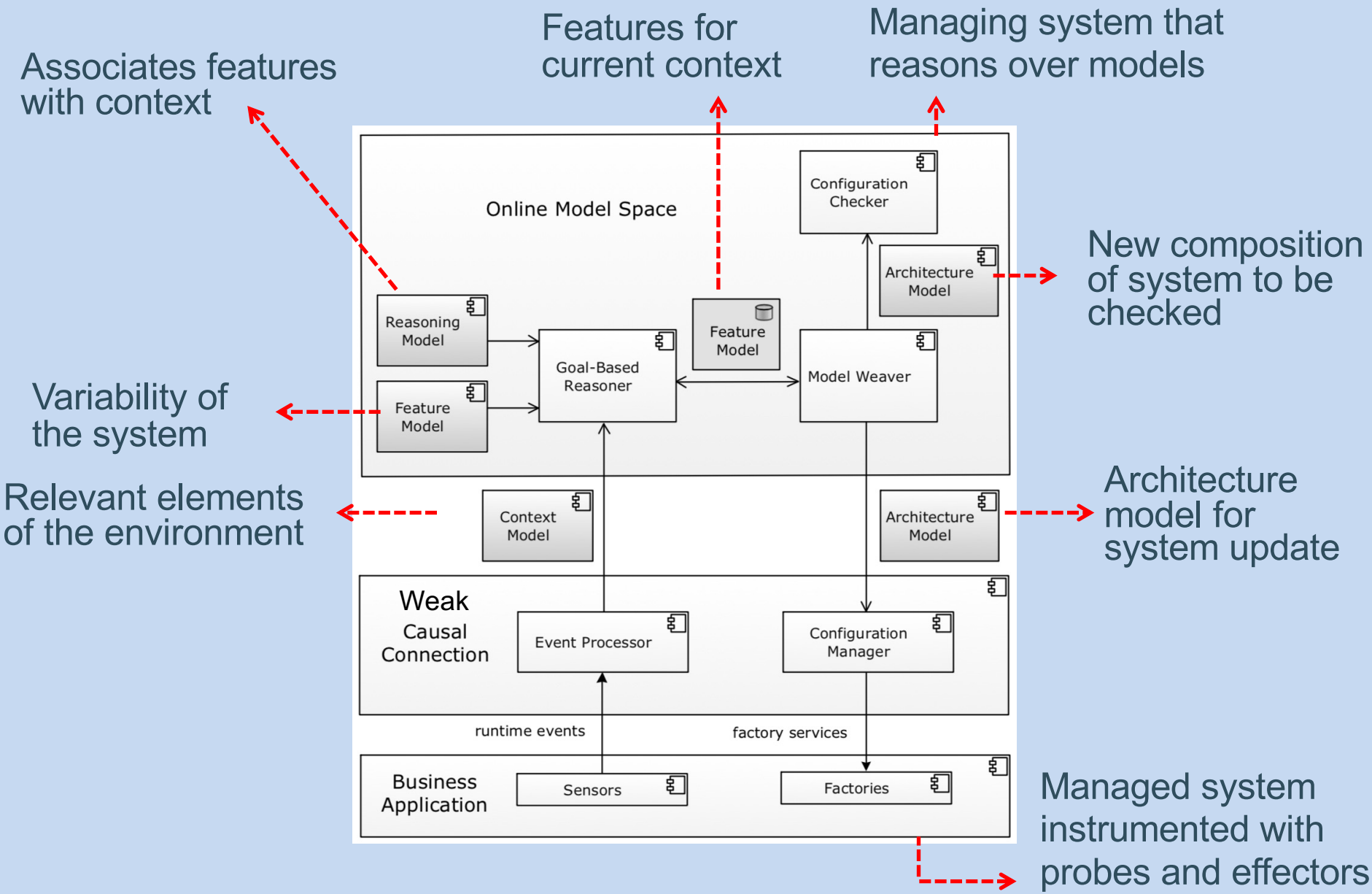
New composition of system to be checked

Variability of the system

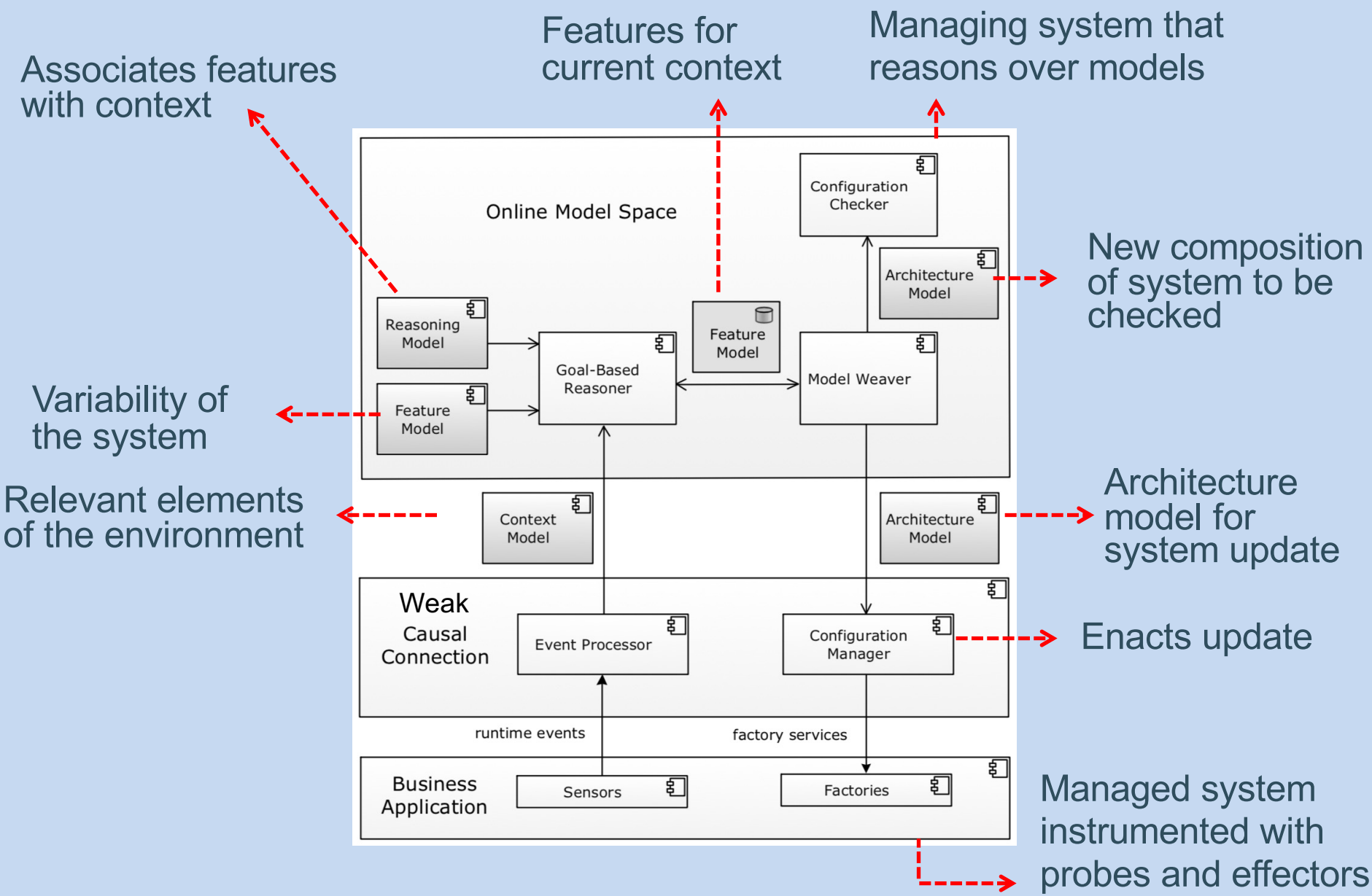
Relevant elements of the environment

Managed system instrumented with probes and effectors

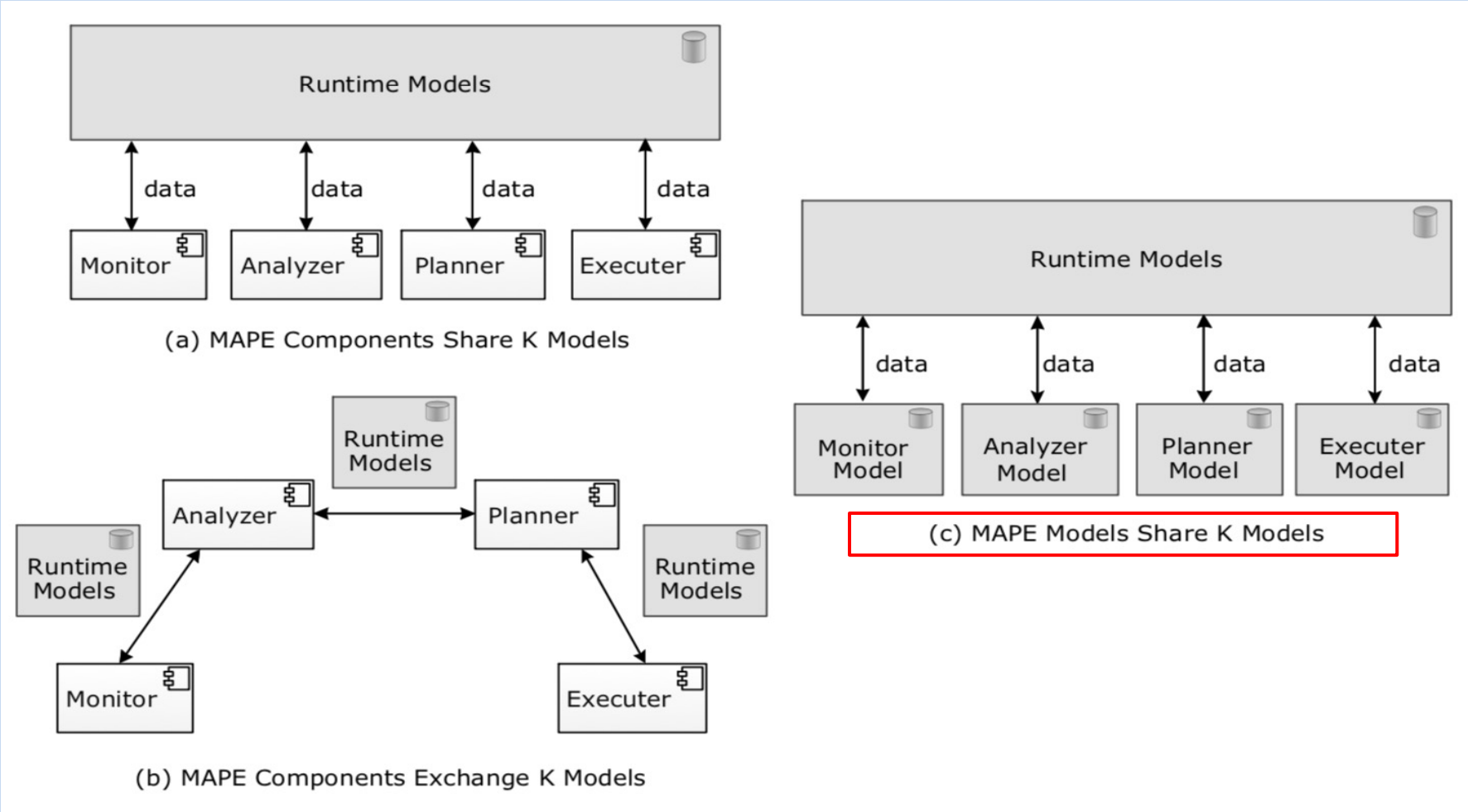
MAPE components exchange K models



MAPE components exchange K models

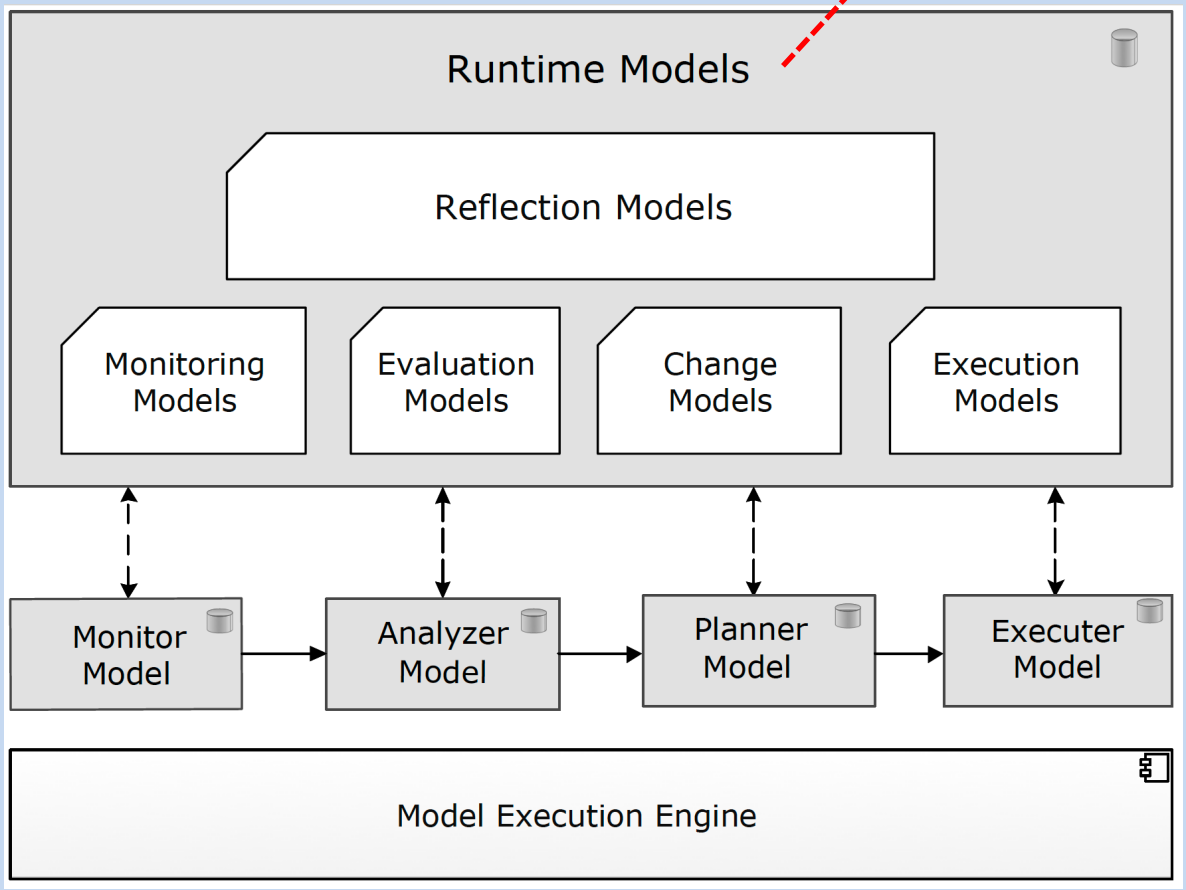


Principle Strategies to Use Runtime Models



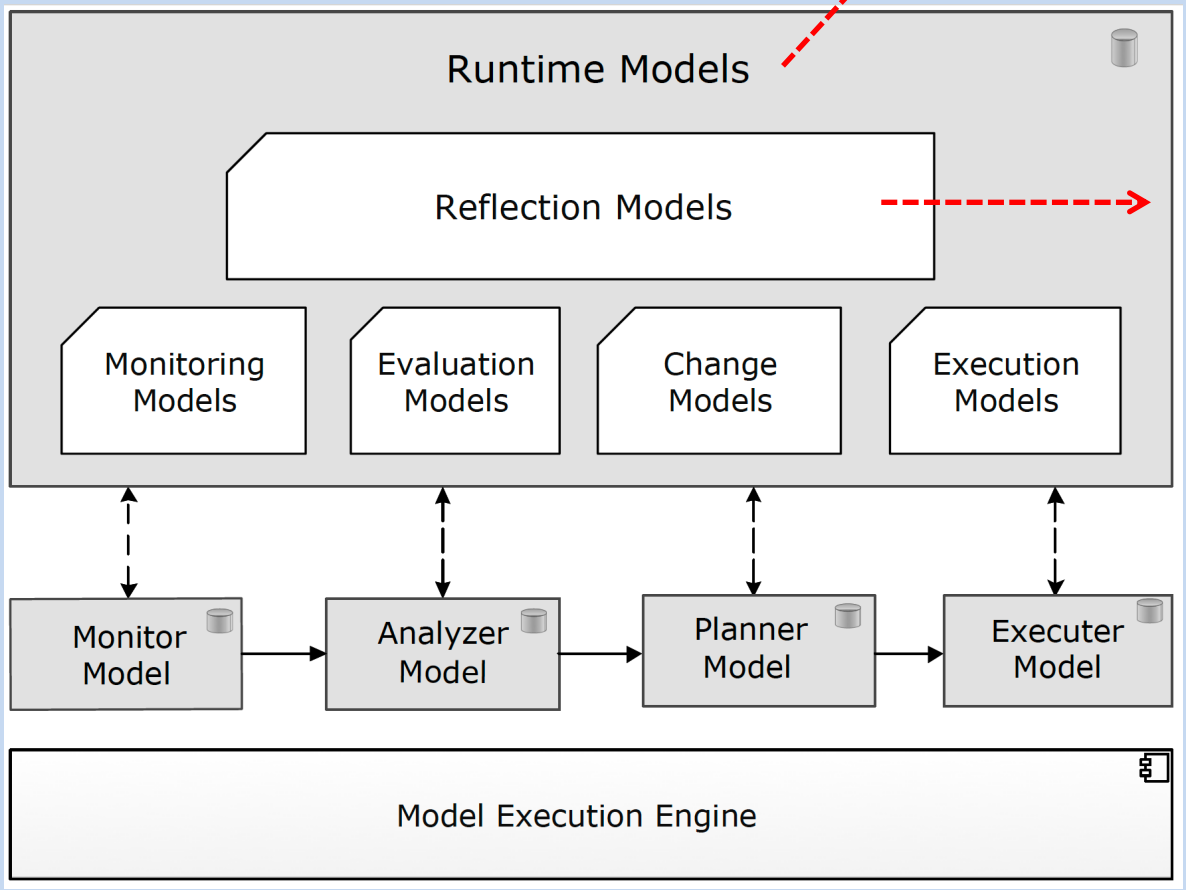
MAPE models share K models

Up-to-date runtime models of system, environment, etc.



MAPE models share K models

Up-to-date runtime models of system, environment, etc.

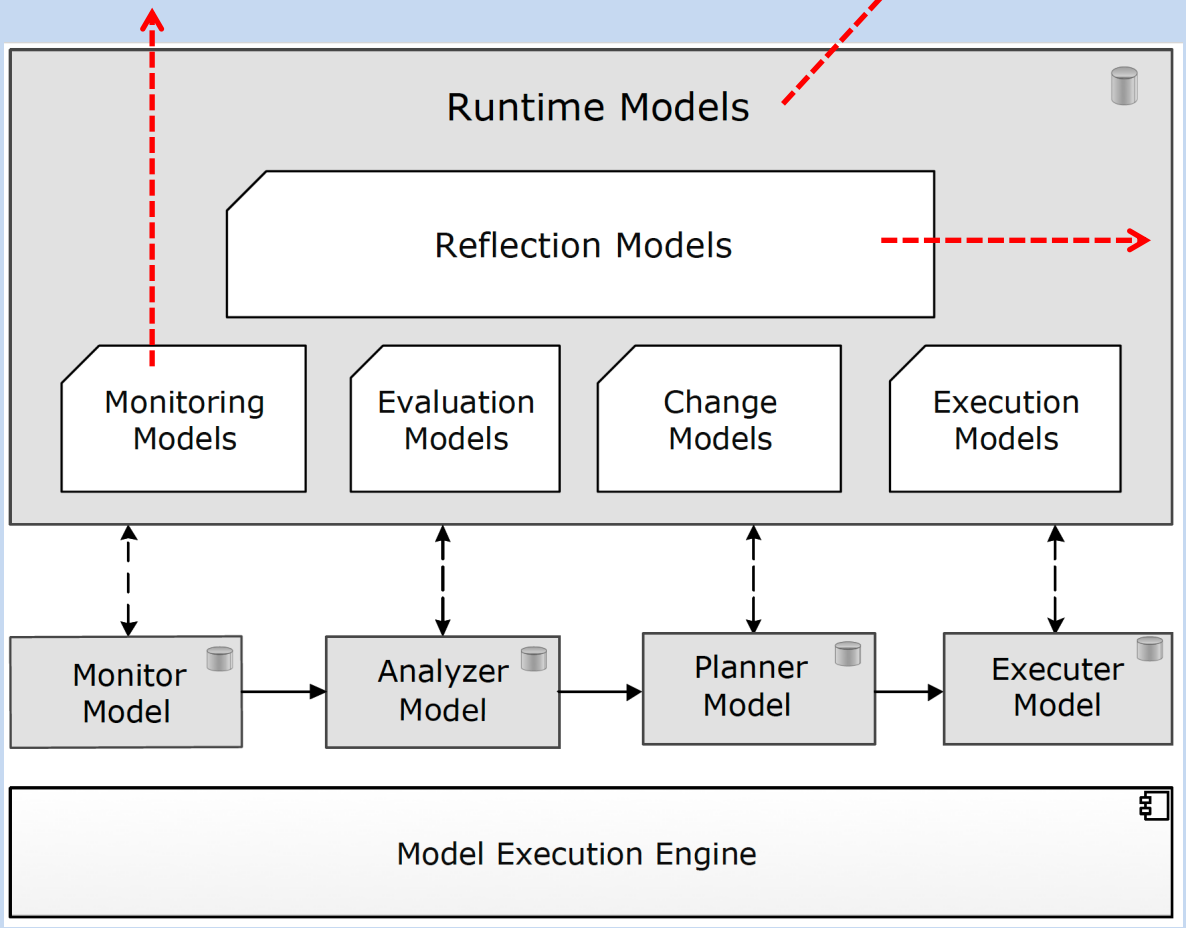


Represents the relevant architectural aspects of the managed system and its environment

MAPE models share K models

Maps observations to abstractions of reflection models

Up-to-date runtime models of system, environment, etc.

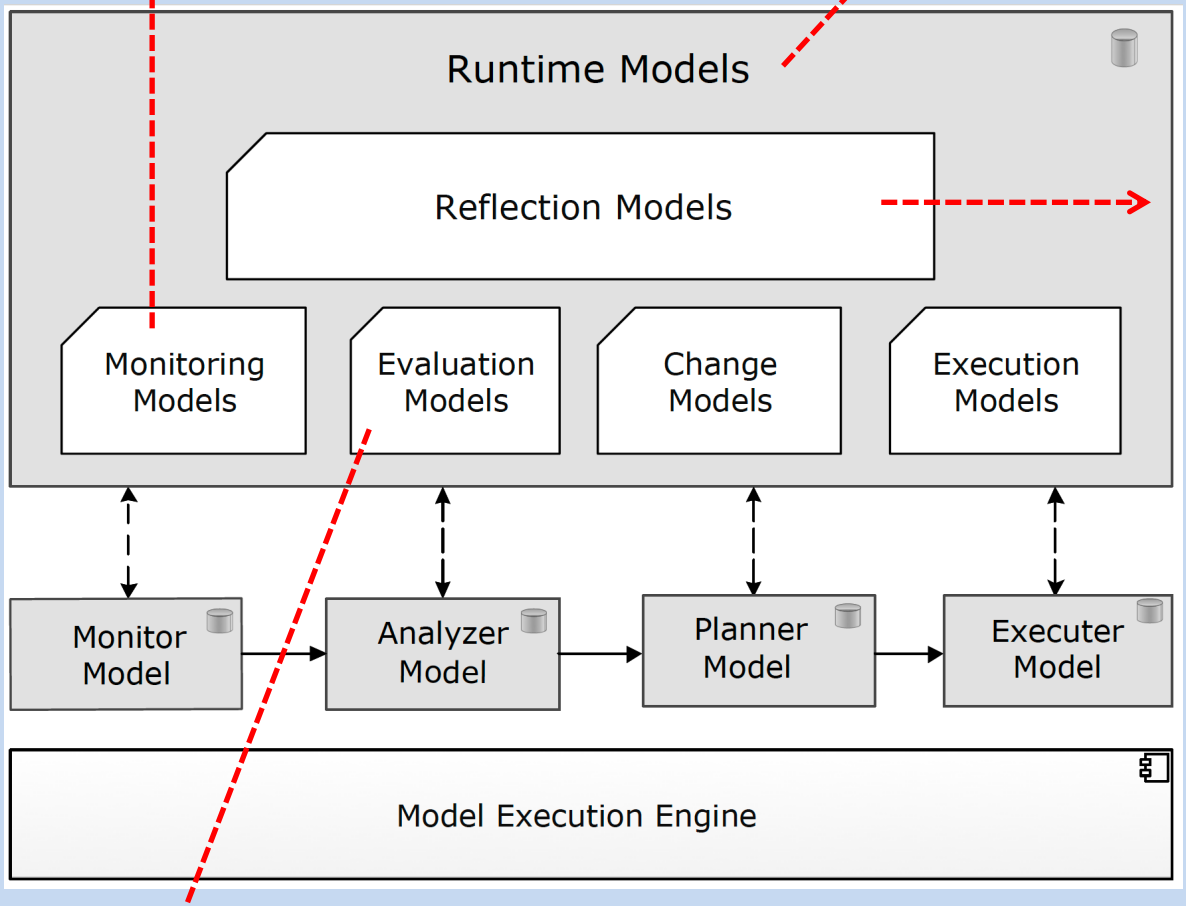


Represents the relevant architectural aspects of the managed system and its environment

MAPE models share K models

Maps observations to abstractions of reflection models

Up-to-date runtime models of system, environment, etc.



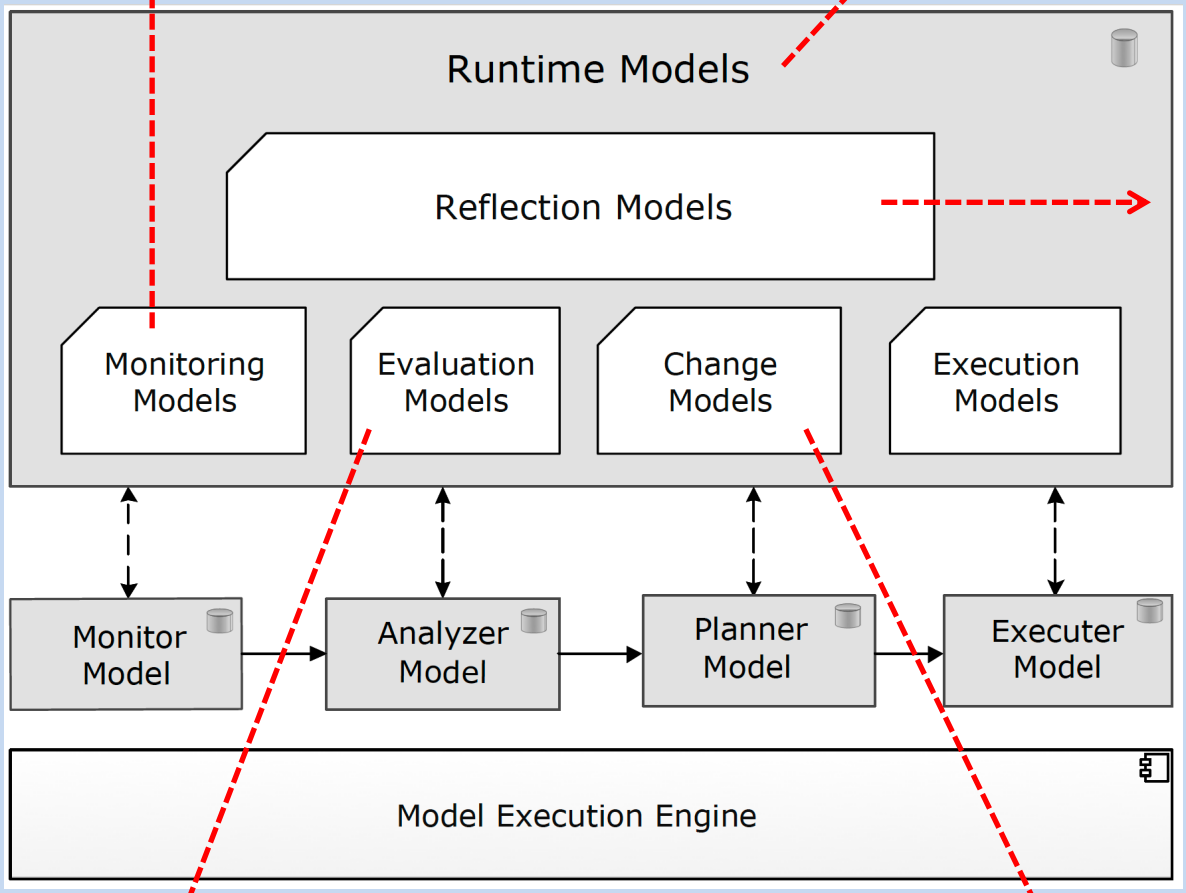
Represents the relevant architectural aspects of the managed system and its environment

Defines allowed configurations and required utility of the managed system

MAPE models share K models

Maps observations to abstractions of reflection models

Up-to-date runtime models of system, environment, etc.



Represents the relevant architectural aspects of the managed system and its environment

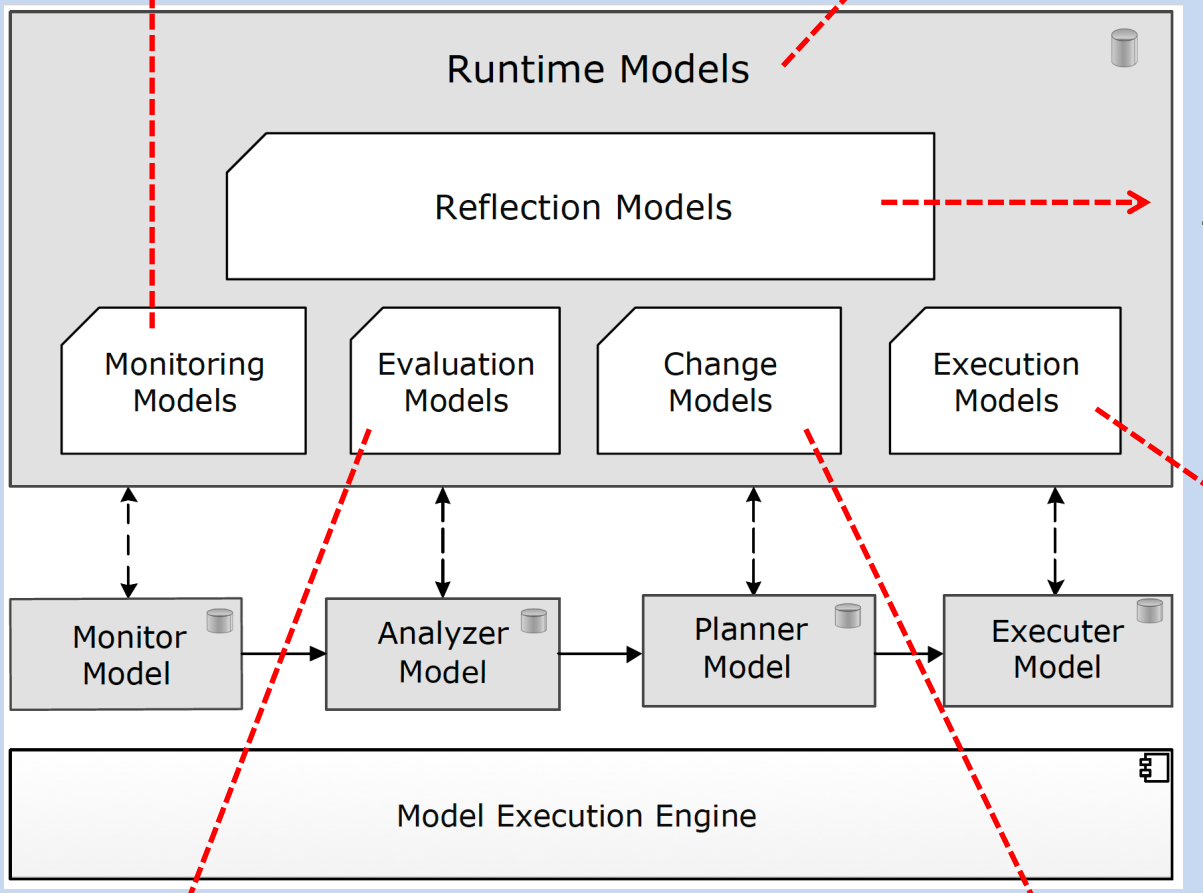
Defines allowed configurations and required utility of the managed system

Defines the space of allowed variability of the system

MAPE models share K models

Maps observations to abstractions of reflection models

Up-to-date runtime models of system, environment, etc.



Represents the relevant architectural aspects of the managed system and its environment

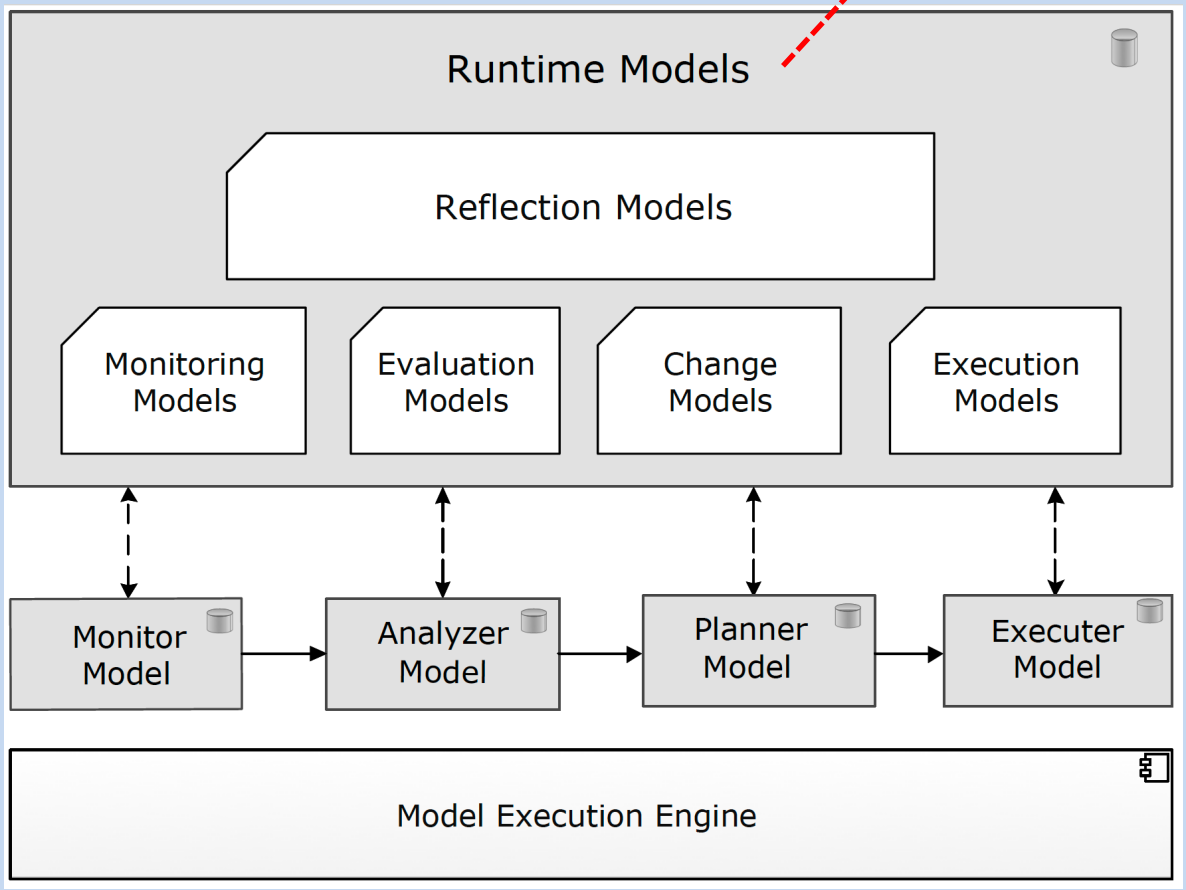
Maps model-level steps of plan to system-level adaptation actions

Defines allowed configurations and required utility of the managed system

Defines the space of allowed variability of the system

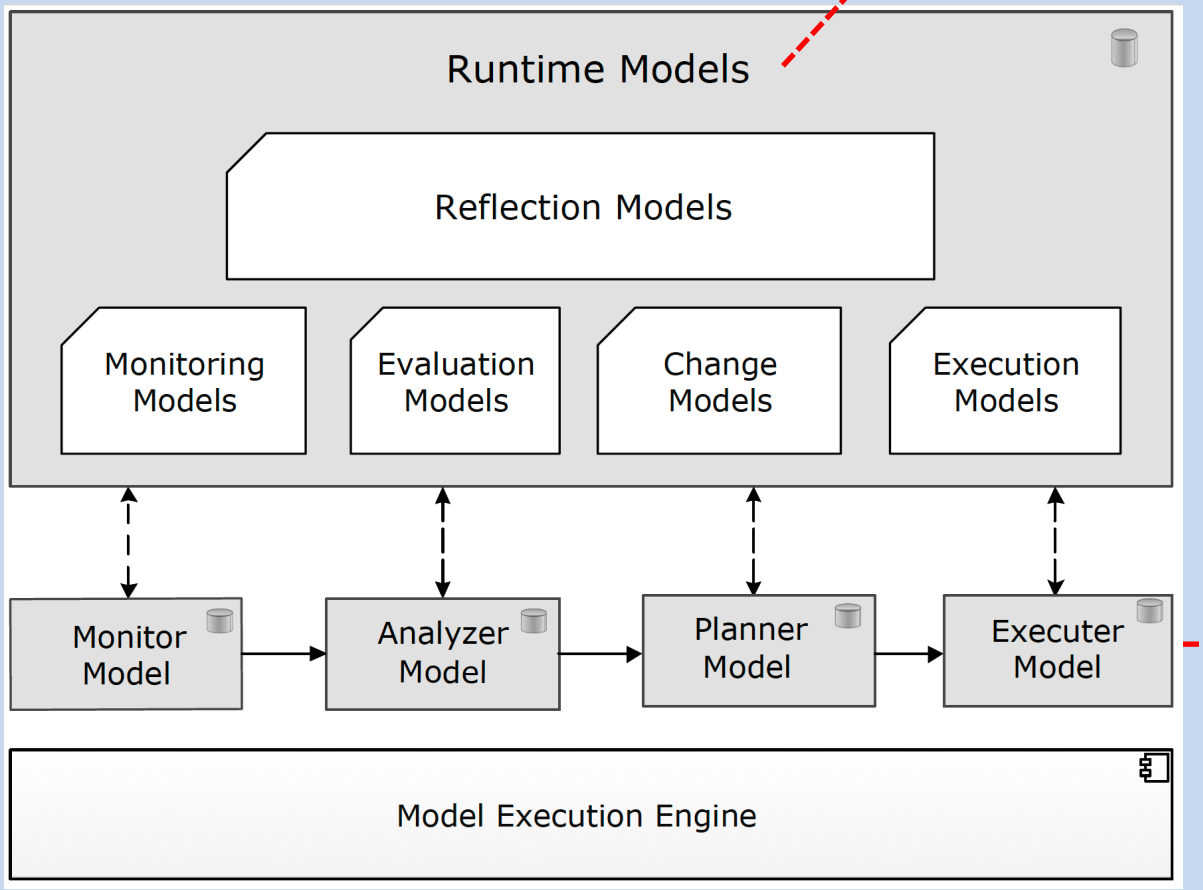
MAPE models share K models

Up-to-date runtime models of system, environment, etc.



MAPE models share K models

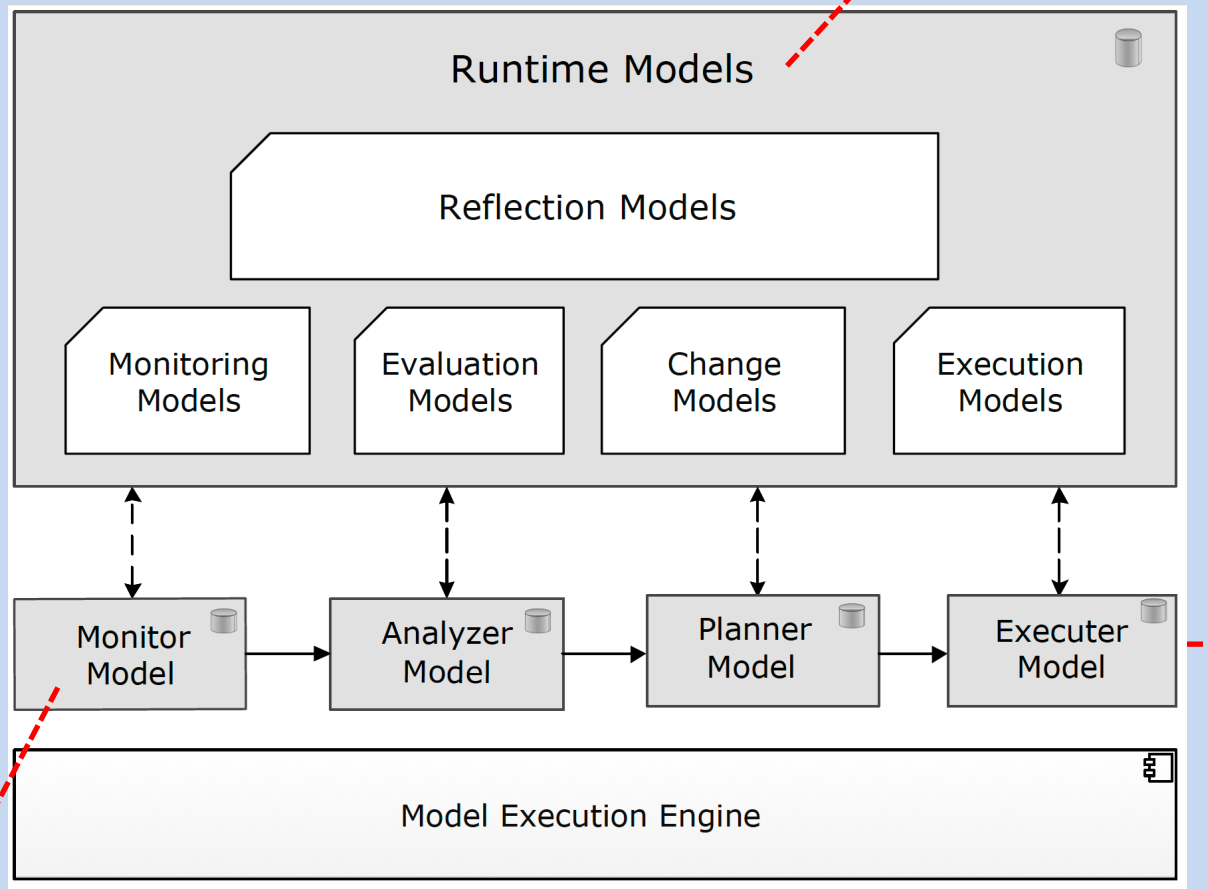
Up-to-date runtime models of system, environment, etc.



MAPE elements as executable runtime models

MAPE models share K models

Up-to-date runtime models of system, environment, etc.

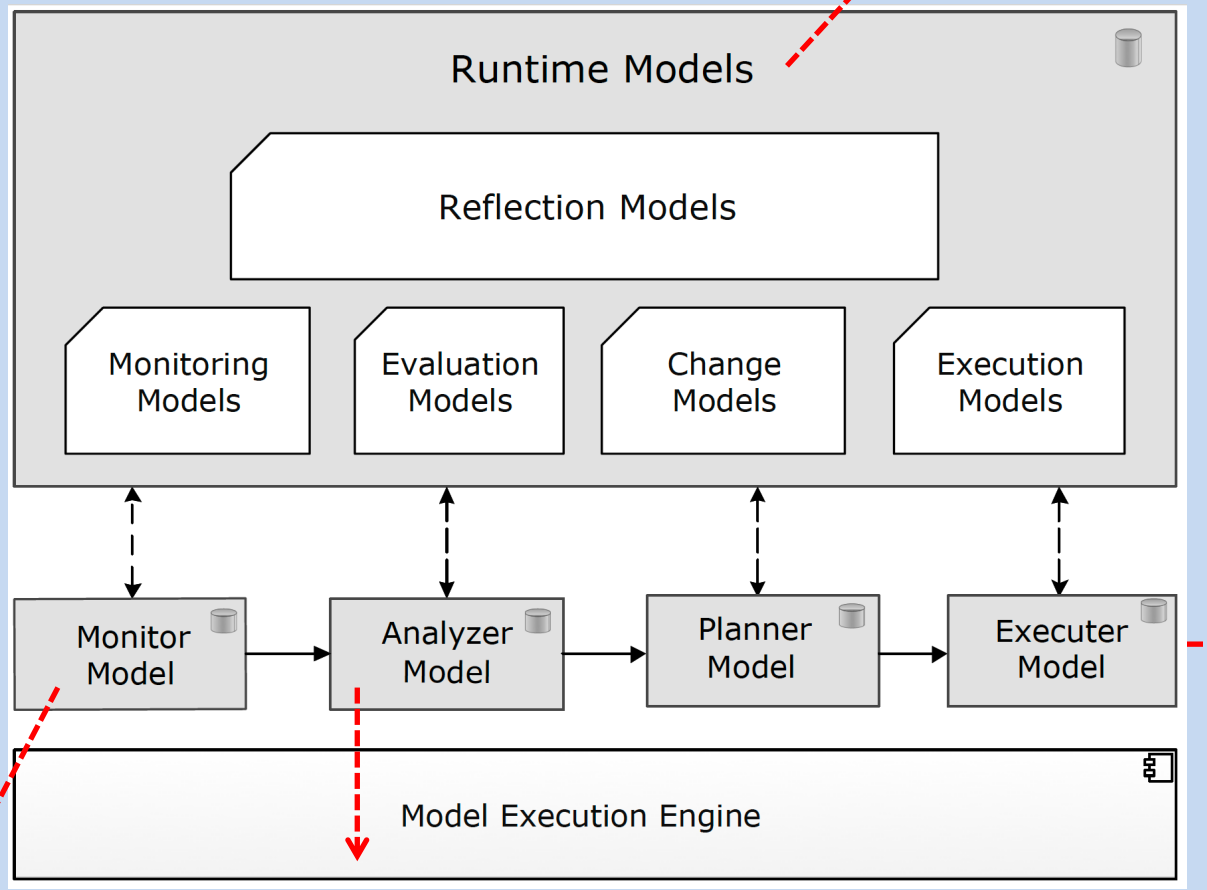


MAPE elements as executable runtime models

Updates reflection models

MAPE models share K models

Up-to-date runtime models of system, environment, etc.



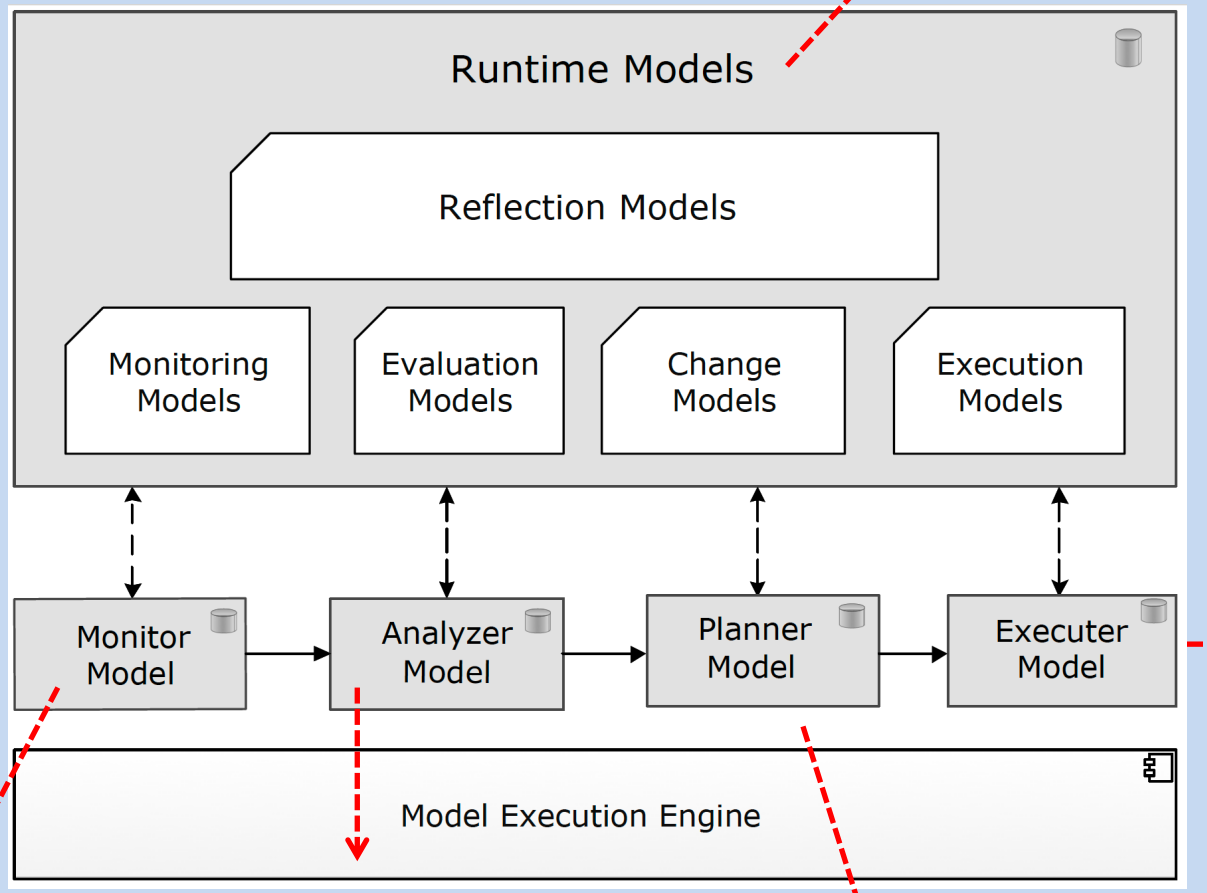
MAPE elements as executable runtime models

Updates reflection models

Analyses reflection models to identify the need for adaptation

MAPE models share K models

Up-to-date runtime models of system, environment, etc.



MAPE elements as executable runtime models

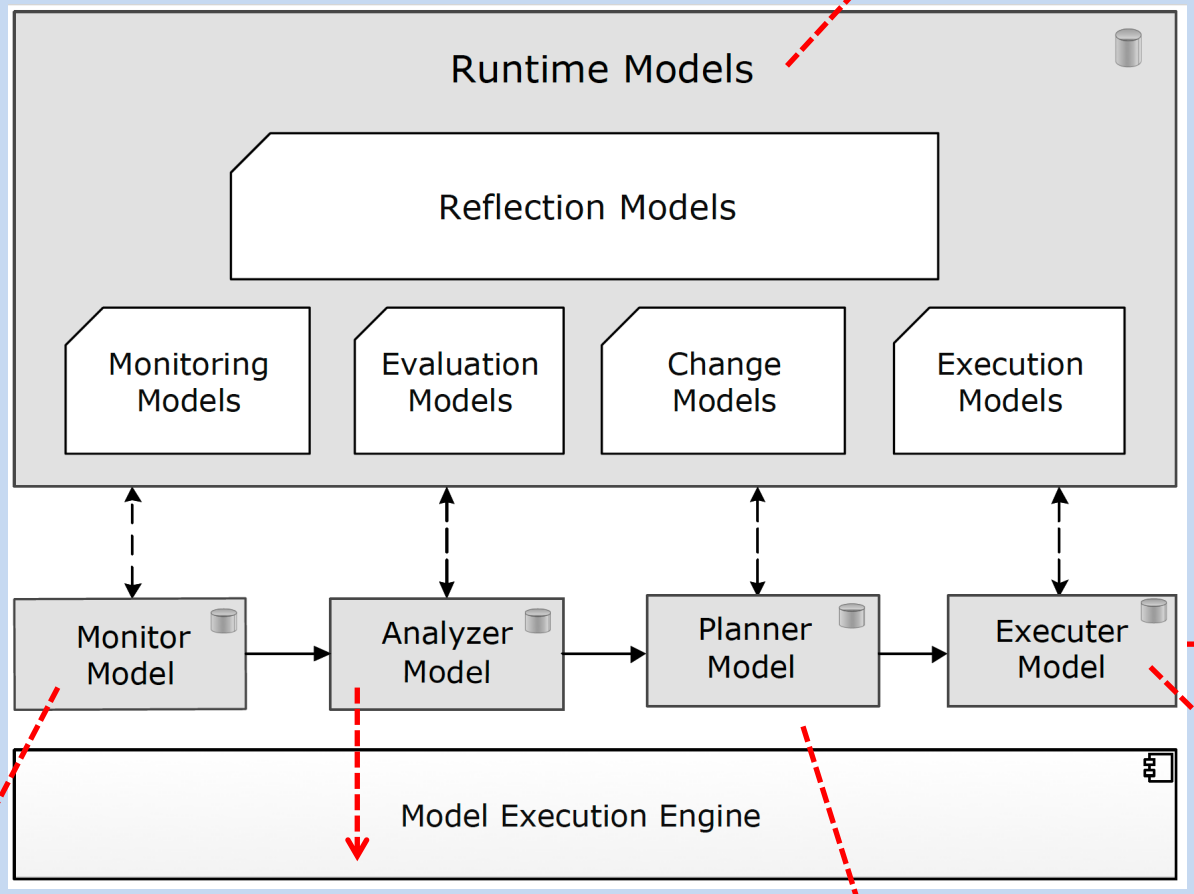
Updates reflection models

Analyses reflection models to identify the need for adaptation

Devises a plan with required changes of the system

MAPE models share K models

Up-to-date runtime models of system, environment, etc.



MAPE elements as executable runtime models

Executes the plan

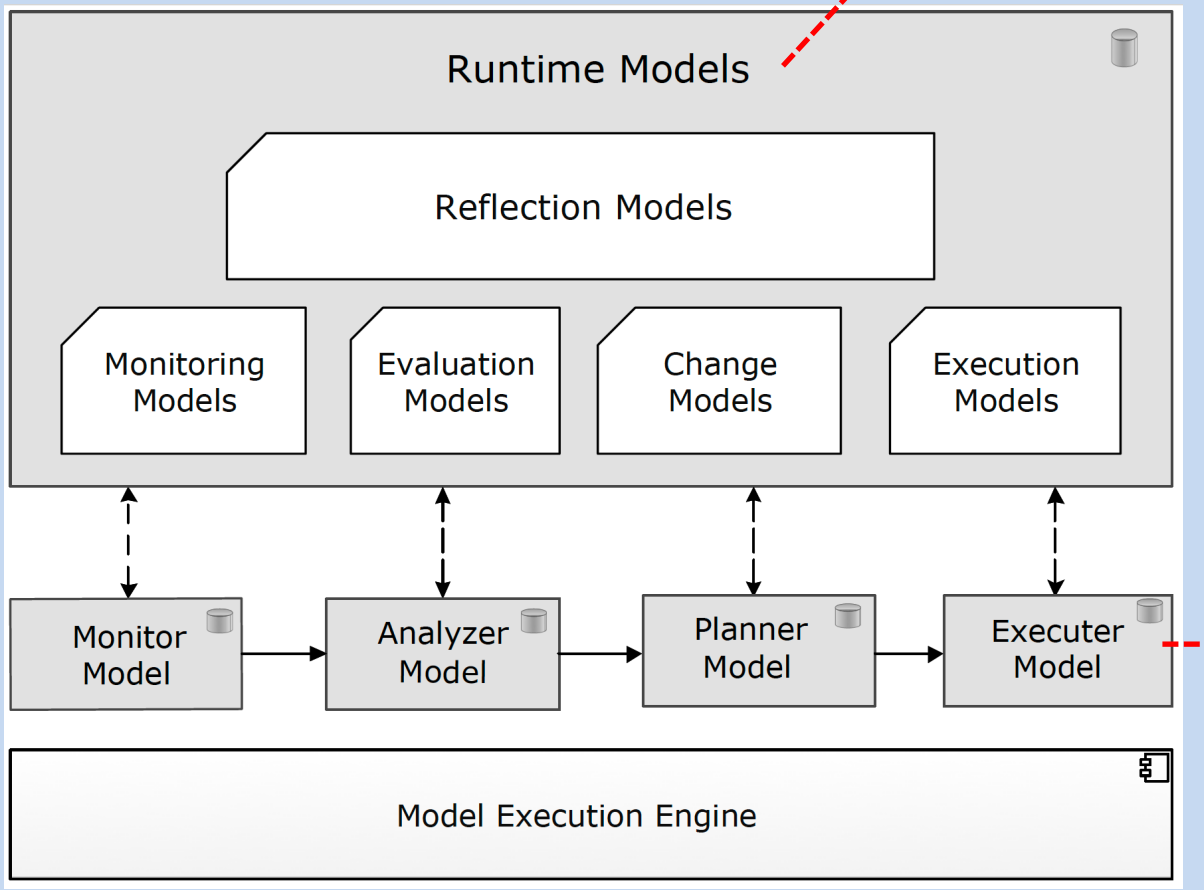
Updates reflection models

Analyses reflection models to identify the need for adaptation

Devises a plan with required changes of the system

MAPE models share K models

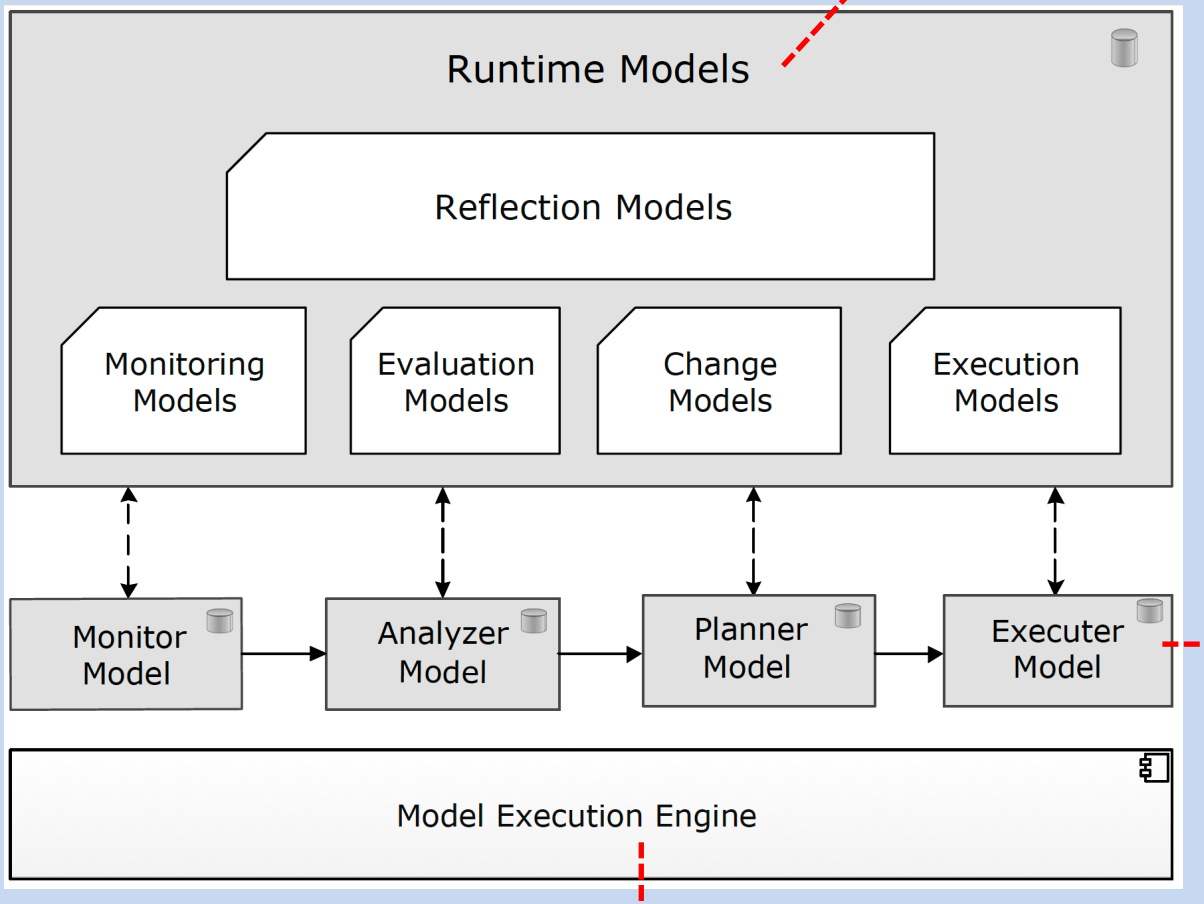
Up-to-date runtime models of system, environment, etc.



MAPE elements as executable runtime models

MAPE models share K models

Up-to-date runtime models of system, environment, etc.



MAPE elements as executable runtime models

Virtual machine executes the MAPE models

Key insights Wave III

Runtime models allow managing complexity of design self-adaptive systems

In self-adaptation we usually use weak causal connection

Three strategies to use runtime models

- MAPE components sharing Knowledge models most common

Wave IV Requirements-driven Adaptation

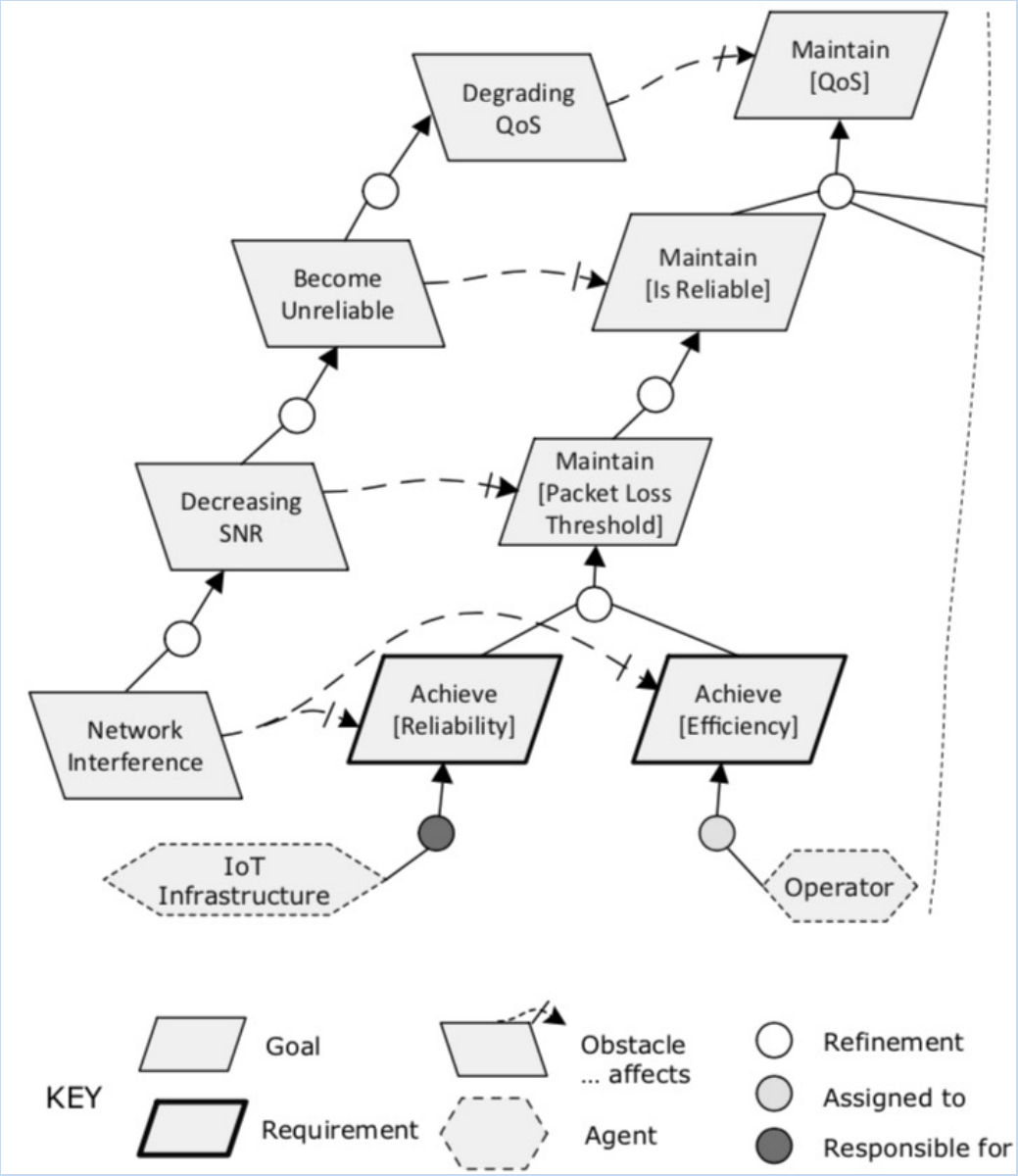
Driver: Understand requirements of feedback loops

Topic: Requirements driving self-adaptation

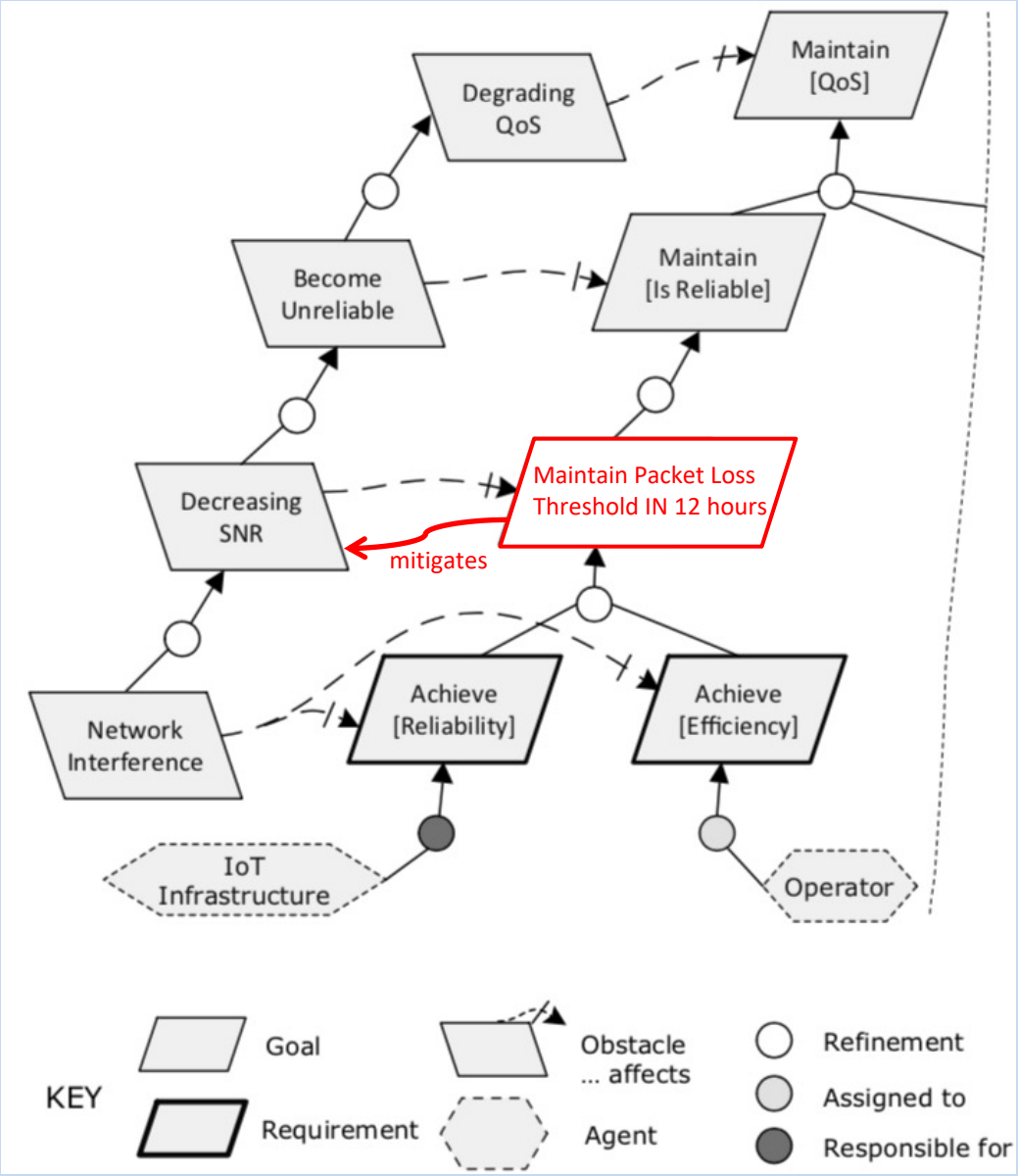
Enabled: Languages and formalisms to specify requirements for self-adaptive systems

Relaxing requirements

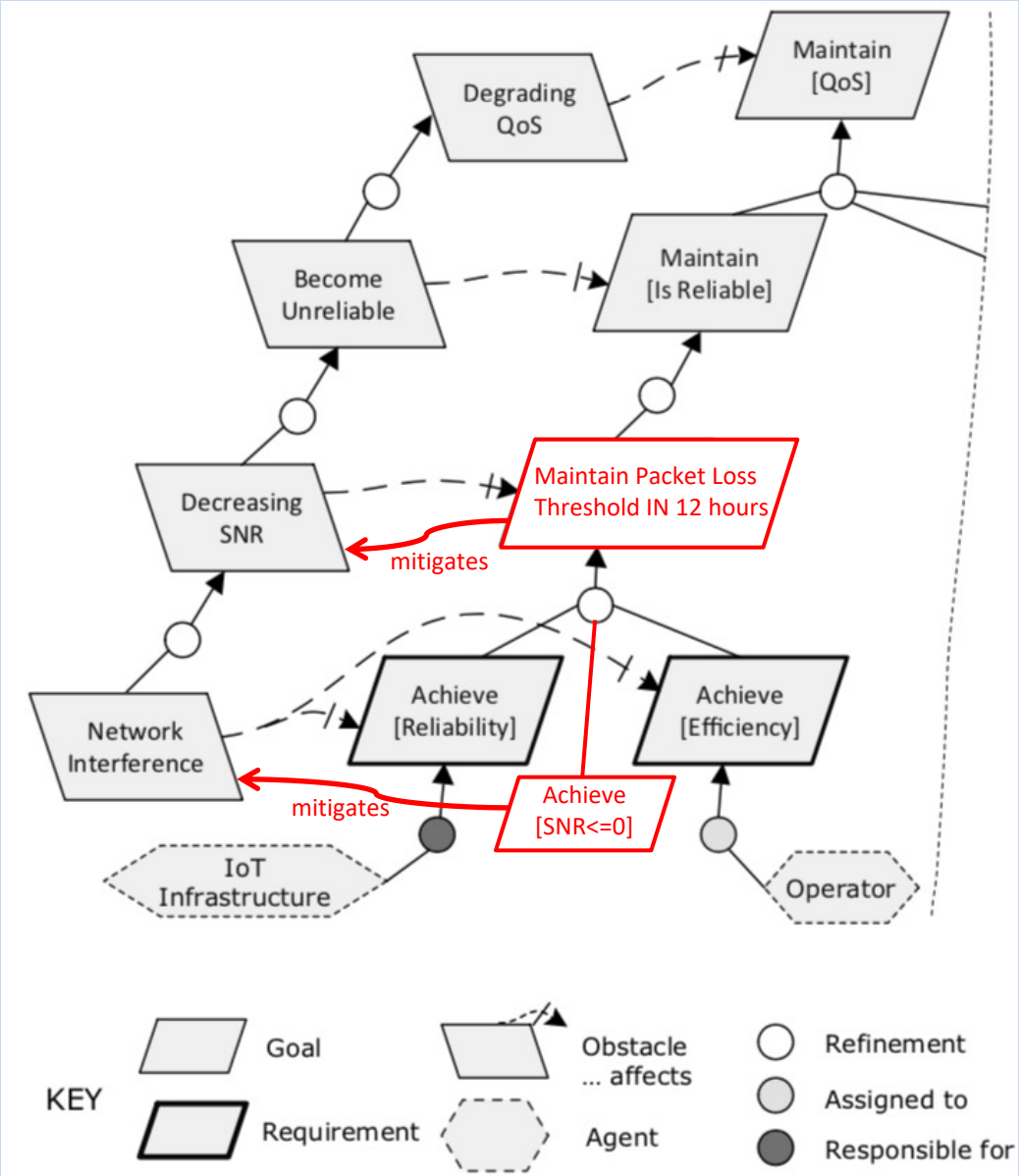
Relaxing requirements



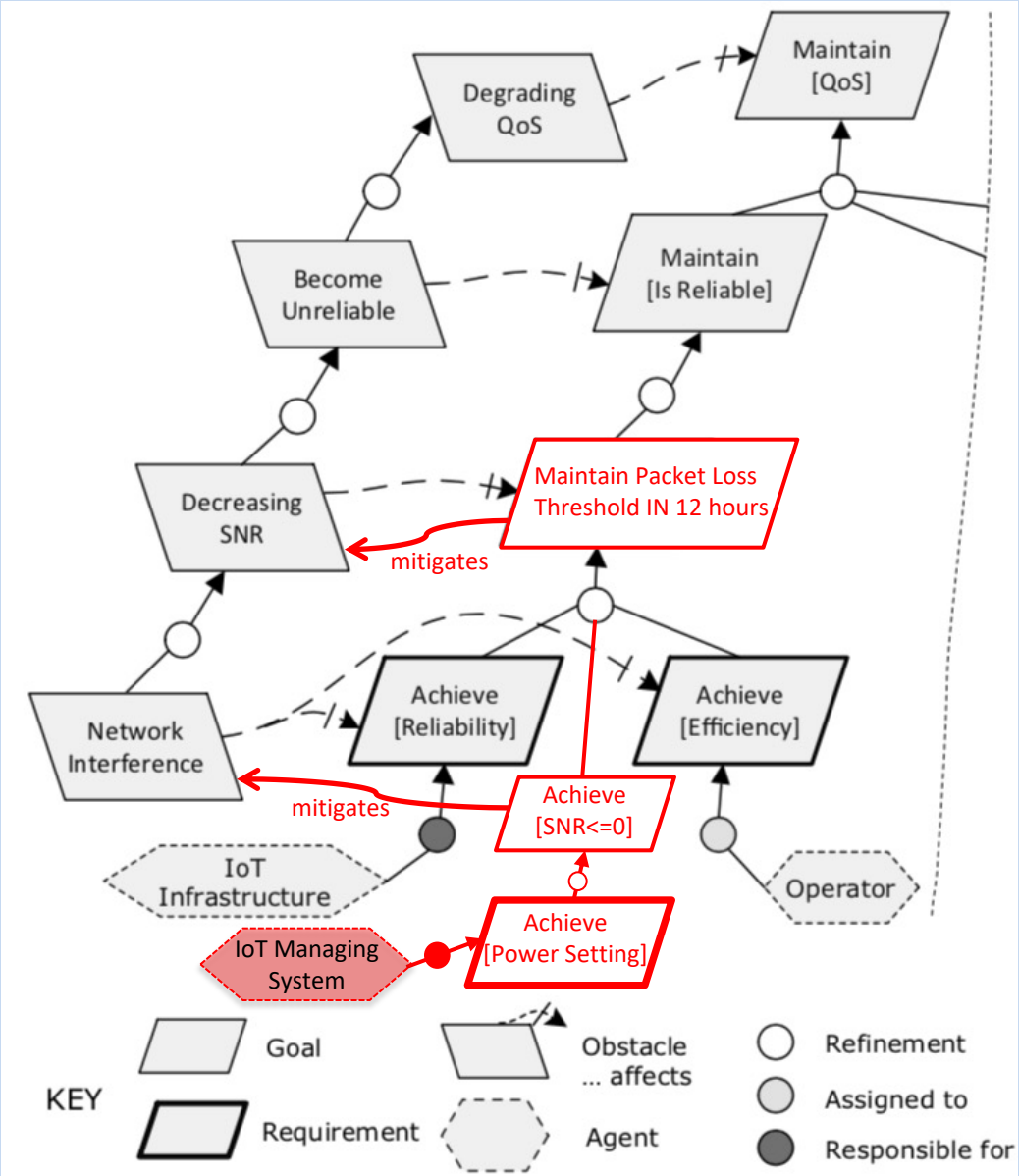
Relaxing requirements



Relaxing requirements

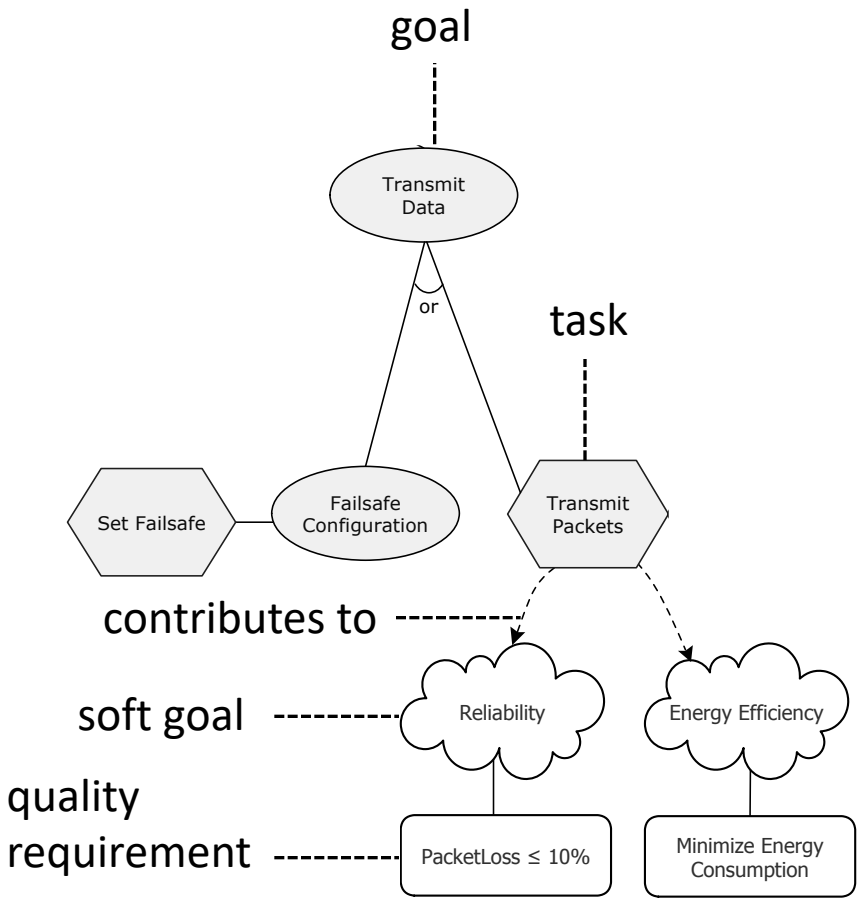


Relaxing requirements

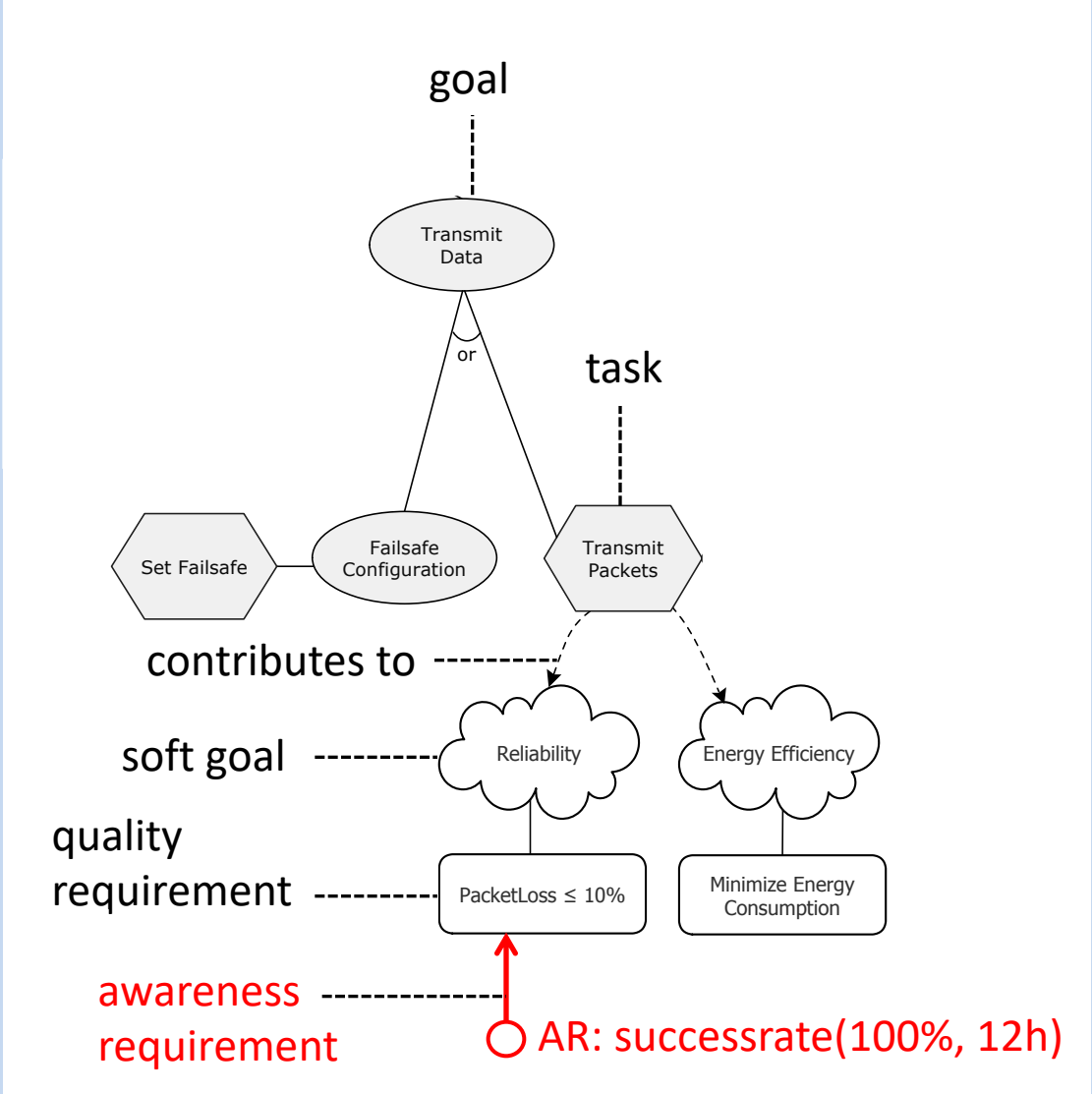


Meta requirements

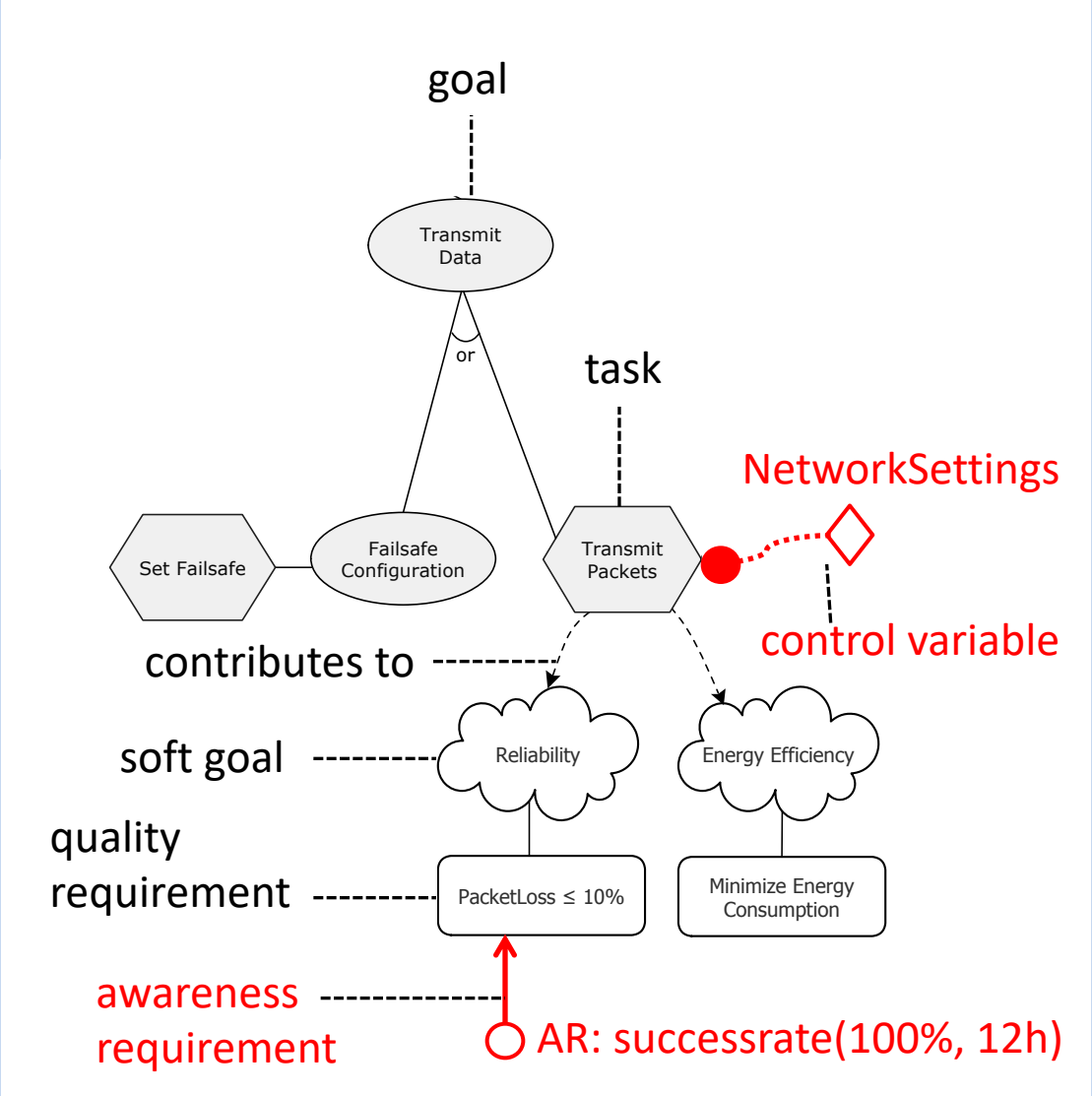
Meta requirements



Meta requirements



Meta requirements



Key insights Wave IV

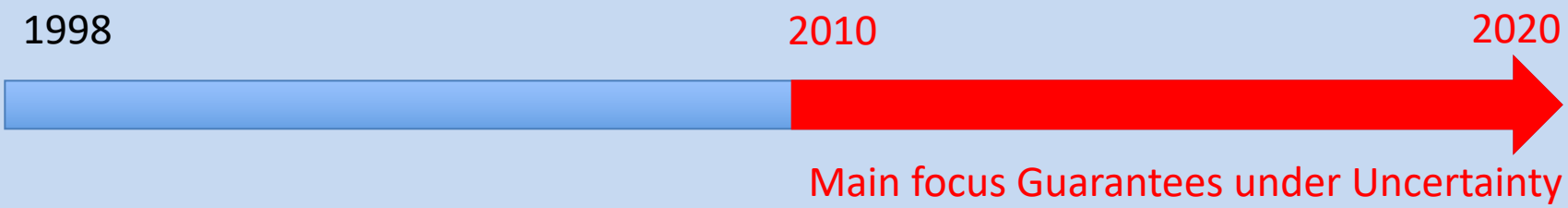
Requirements may not be achievable at all times due to uncertainties

Two techniques to deal with that are:

- Relaxing requirements
- Meta-requirements

Correctness of feedback loop behavior is crucial

Wave V
Guarantees under Uncertainty



Driver: Uncertainty handled in ad-hoc manner

Topic: The role of uncertainty in self-adaptive systems and how to mitigate it

Enabled: Use of formal techniques (at runtime) to guarantee adaptation goals under uncertainty

Uncertainty in self-adaptive systems

Uncertainty in self-adaptive systems is any deviation of deterministic knowledge that may reduce the confidence of adaptation decisions made based on the knowledge.

Taxonomies of uncertainty

Taxonomy of uncertainty
(Ramirez et al.)

Uncertainty in the environment
(Esfahani & Malek)

Uncertainty dimensions: location, level, and nature
(Perez-Palacin & Mirandola)

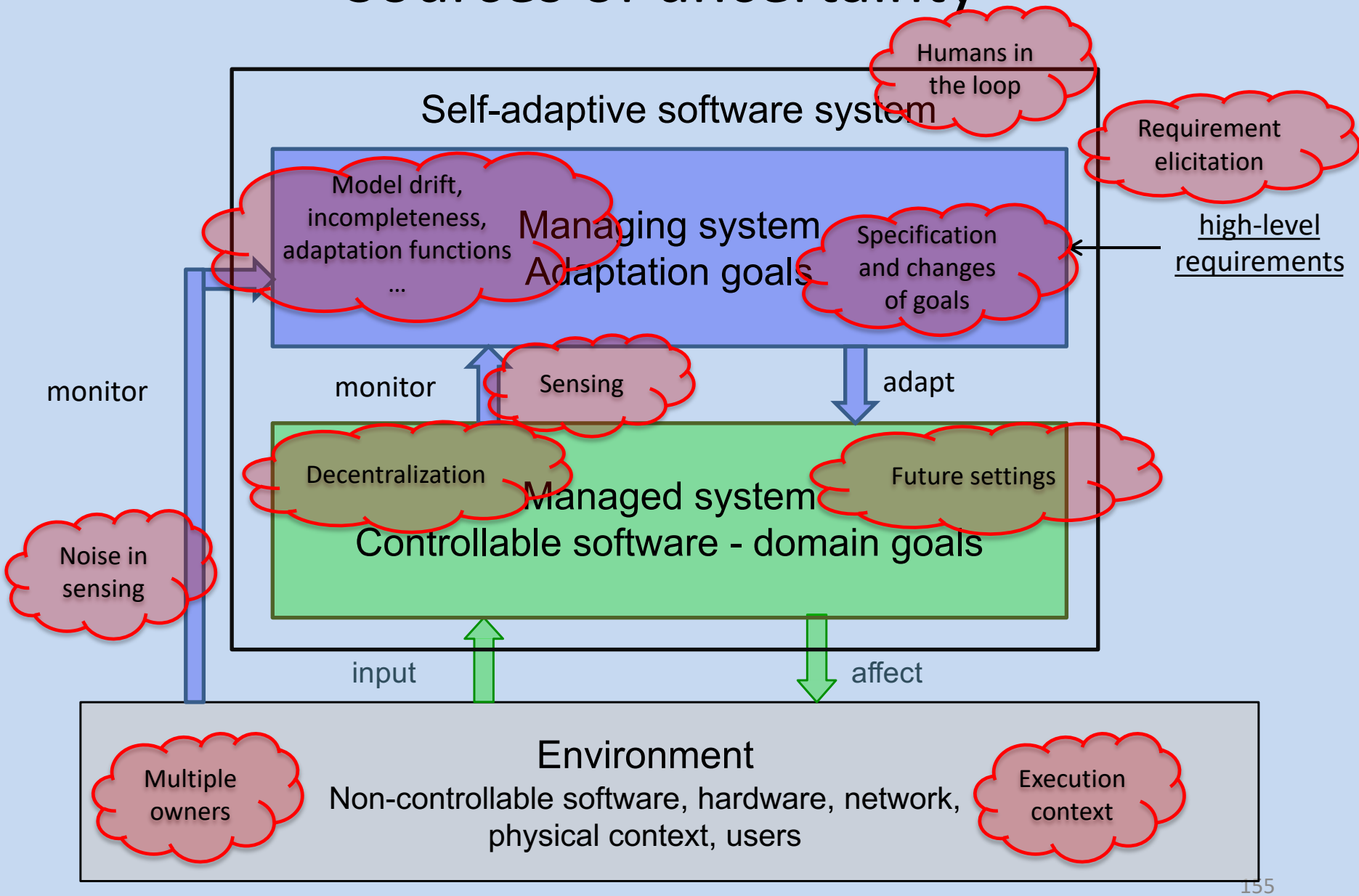
Classification framework for uncertainty and sources
(Mahdavi-Hezavehi et al.)

"Uncertainty in self-adaptive systems is any deviation of deterministic knowledge that may reduce the confidence of adaptation decisions made based on the knowledge."

Group	Sources
Context	Execution context, noise in sensing, different information sources
System	Simplifying assumptions, model drift, decentralization, future settings, adaptation functions, learning
Goals	Requirements elicitation, specification of goals, future goal changes
Humans	Human in the loop, multiple ownership

Ramirez et al., A taxonomy of uncertainty for dynamically adaptive systems, SEAMS 2012
Esfahani & Malek, Uncertainty in Self-Adaptive Software Systems, LNCS Springer, 2013
Perez-Palacin & Mirandola, Uncertainties in the Modelling of Self-adaptive Systems: A Taxonomy, ICPE 2014
Mahdavi-Hezavehi et al. A Classification Framework of Uncertainty in Architecture-Based Self-Adaptive Systems, 2017

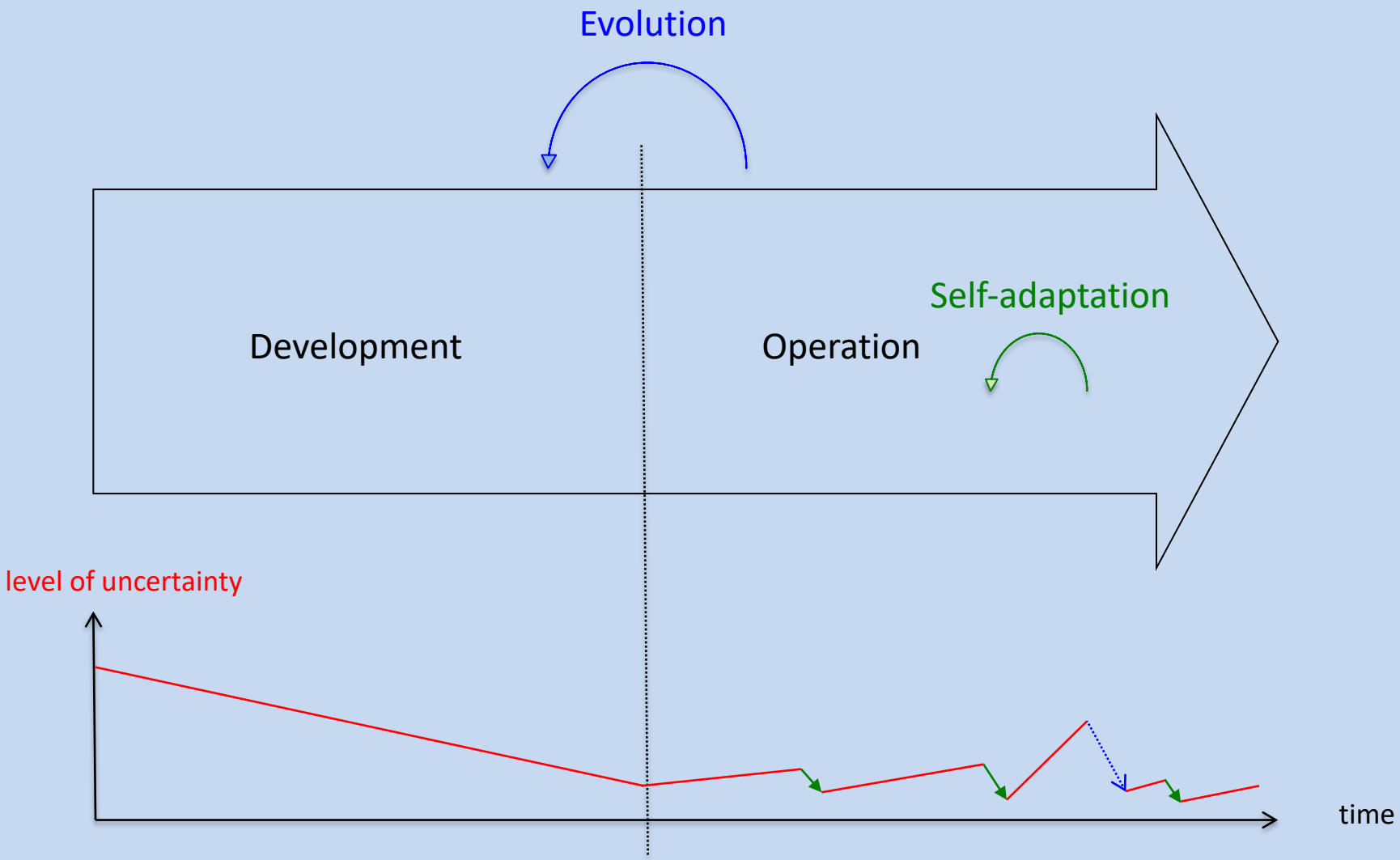
Sources of uncertainty



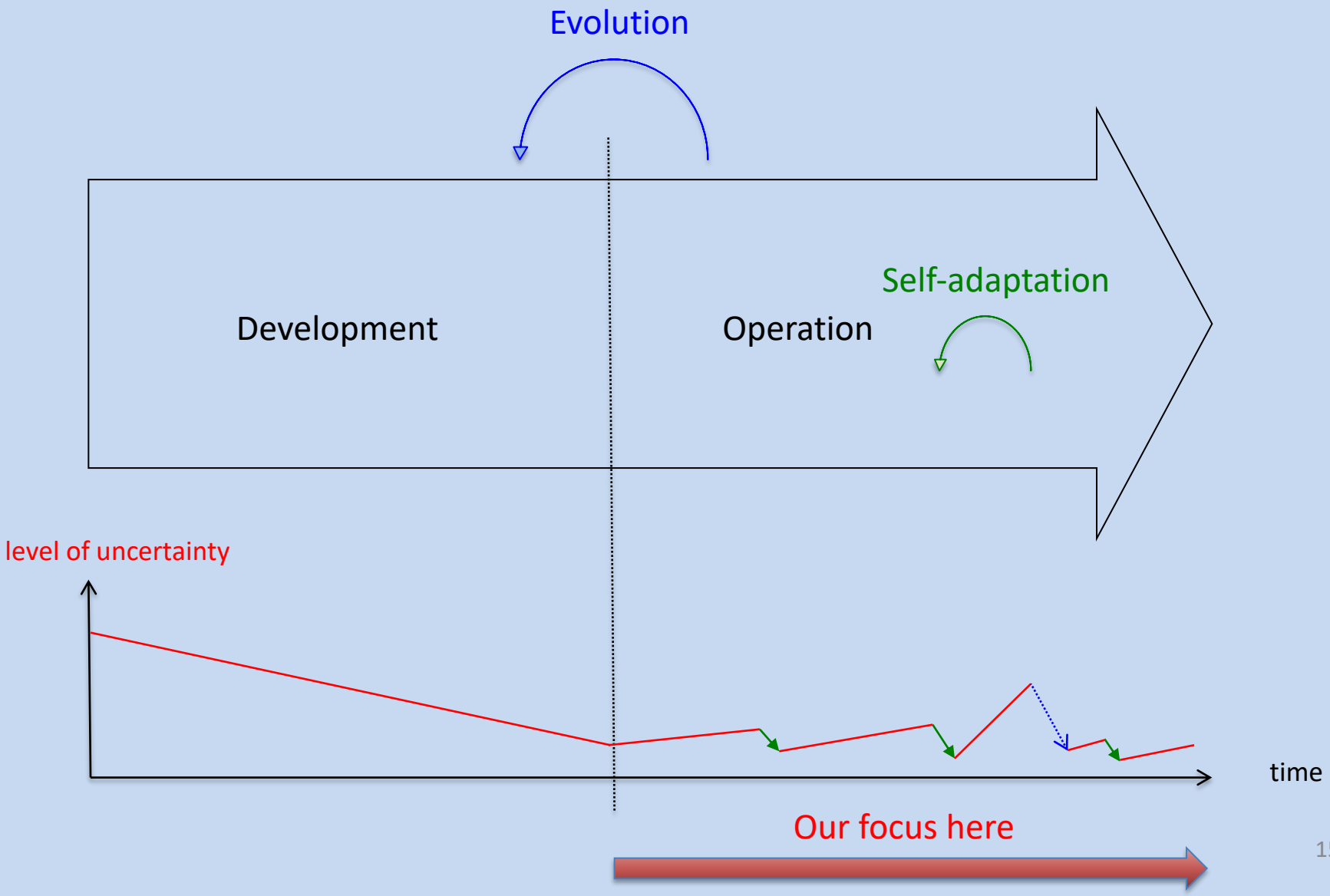
Uncertainty in self-adaptive systems

- Uncertainty is a crosscutting phenomenon in self-adaptive systems
- Not mitigating uncertainty may jeopardize system goals… or worse
- Dichotomy: uncertainty versus guarantees
- “Taming uncertainty”

Taming uncertainty

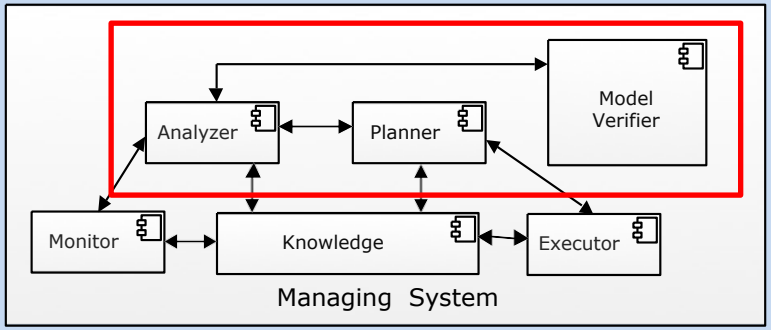


Taming uncertainty



Taming uncertainty using self-adaptation

- Prominent approach relies on applying formal techniques to make adaptation decisions
  - Analysis stage: evaluate the adaptation options for each goal using formal verification
  - Planning stage: select the best option based on the verification results and the adaptation goals

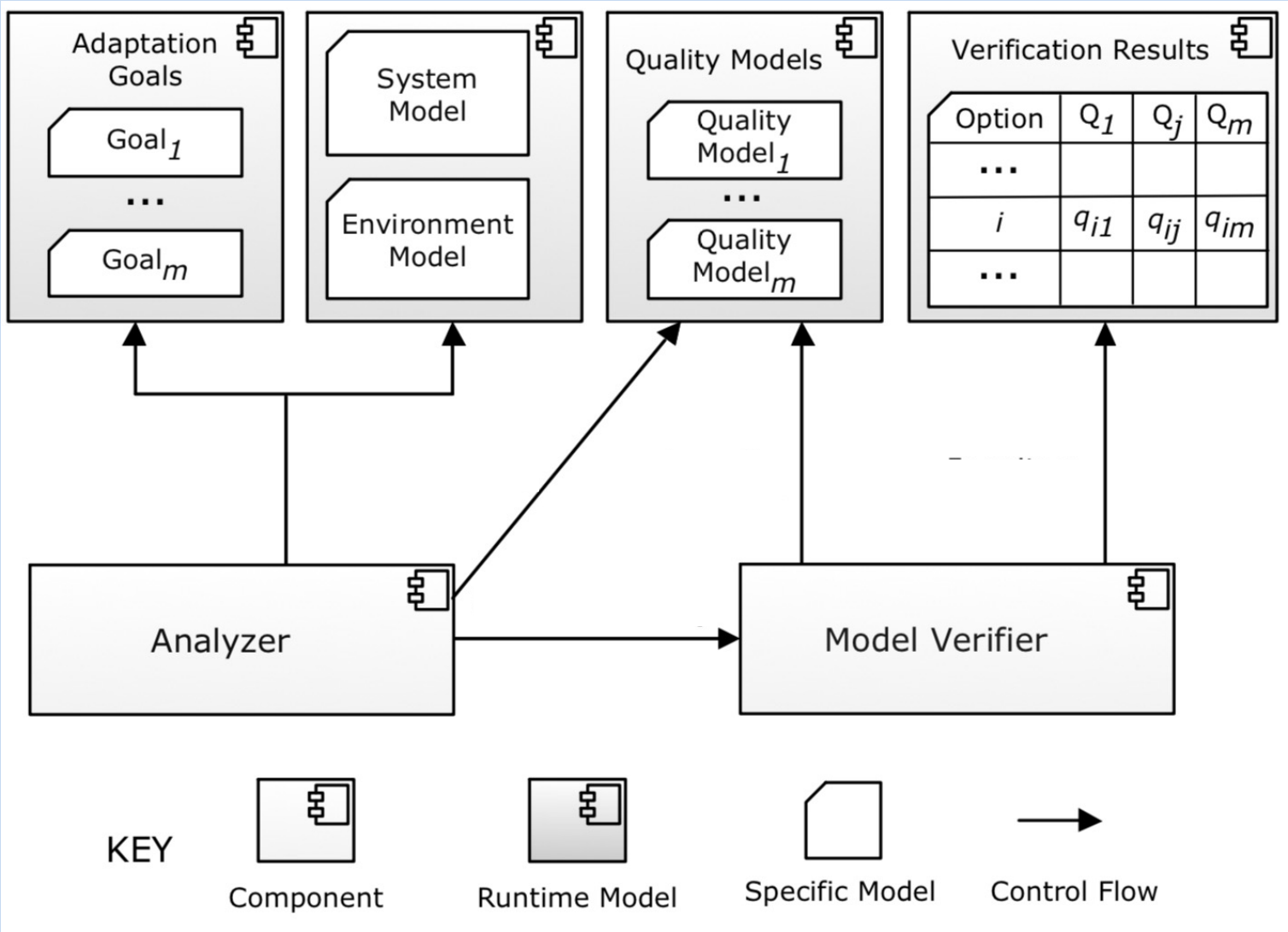


Taming uncertainty in self-adaptive systems

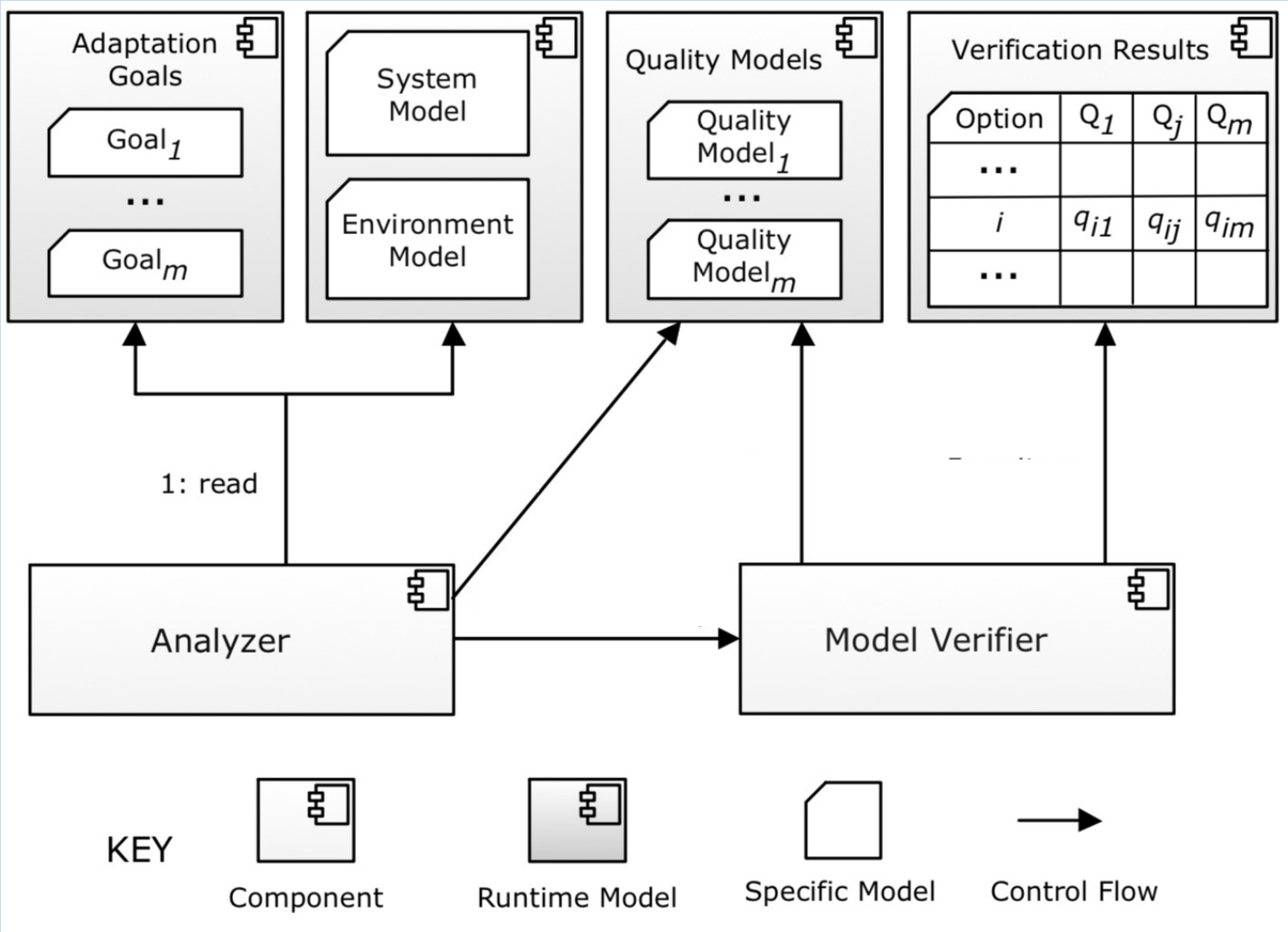
I. Analysis of adaptation options

II. Selection of best adaptation option

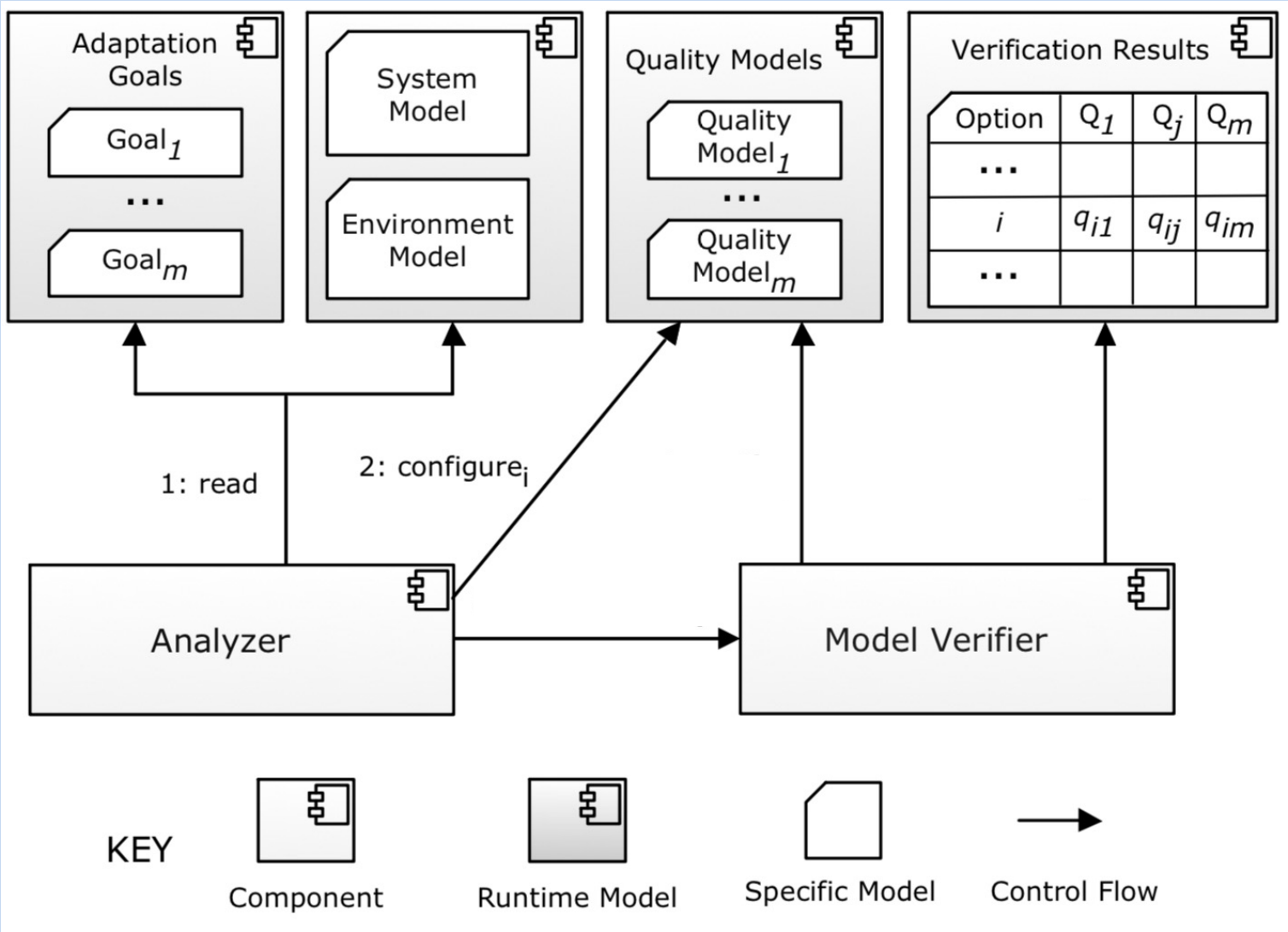
I. Analysis of adaptation options



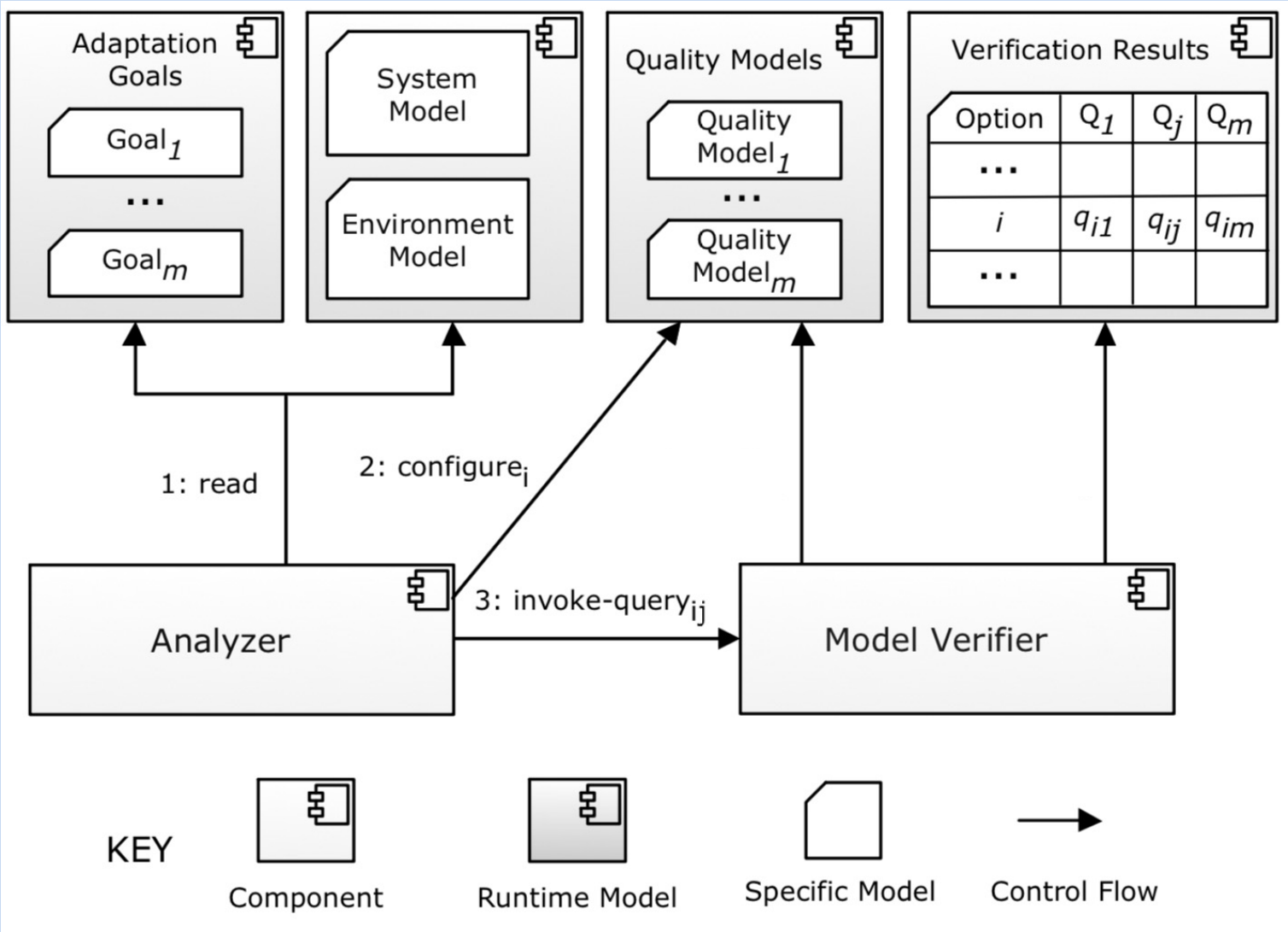
I. Analysis of adaptation options



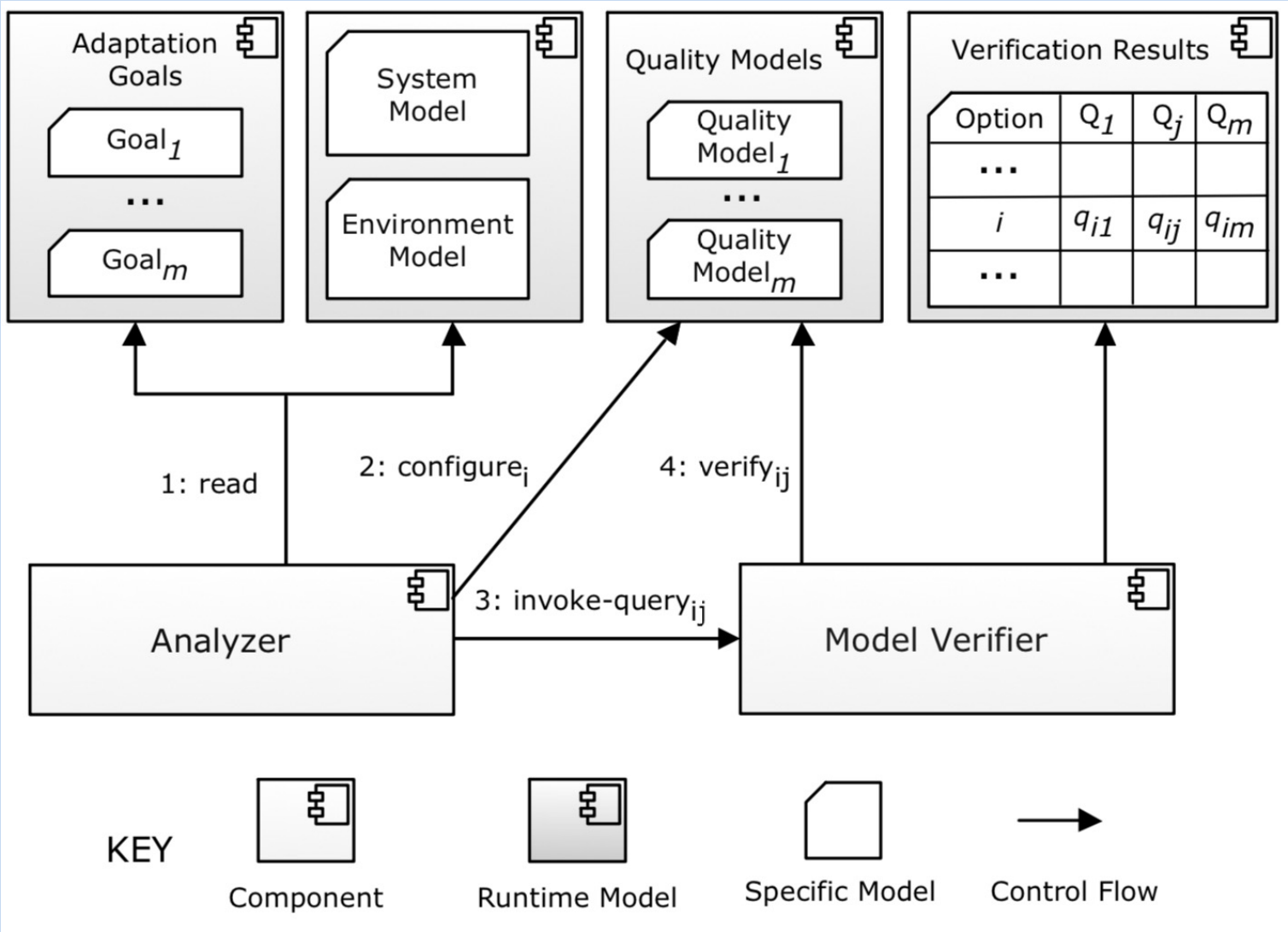
I. Analysis of adaptation options



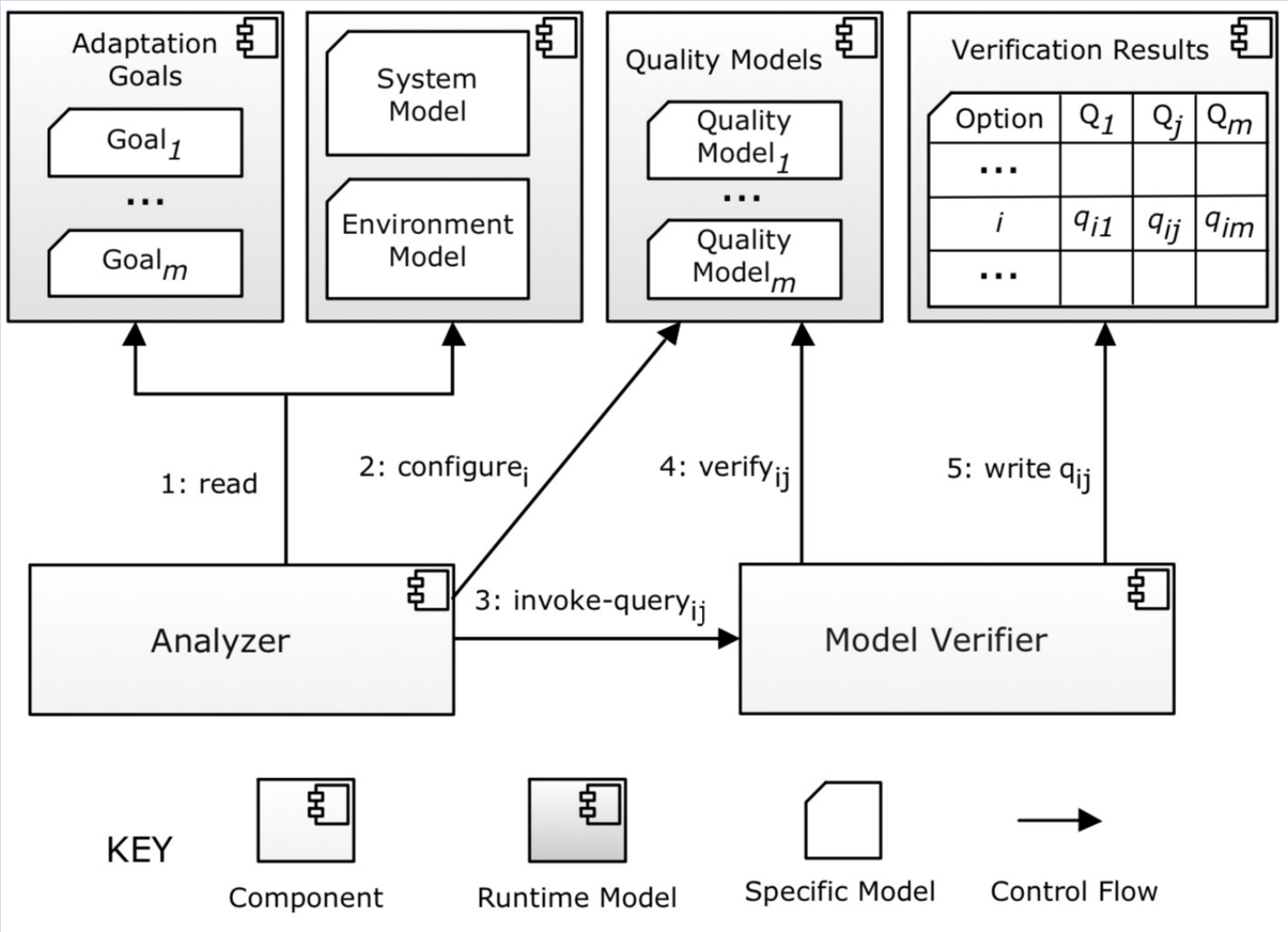
I. Analysis of adaptation options



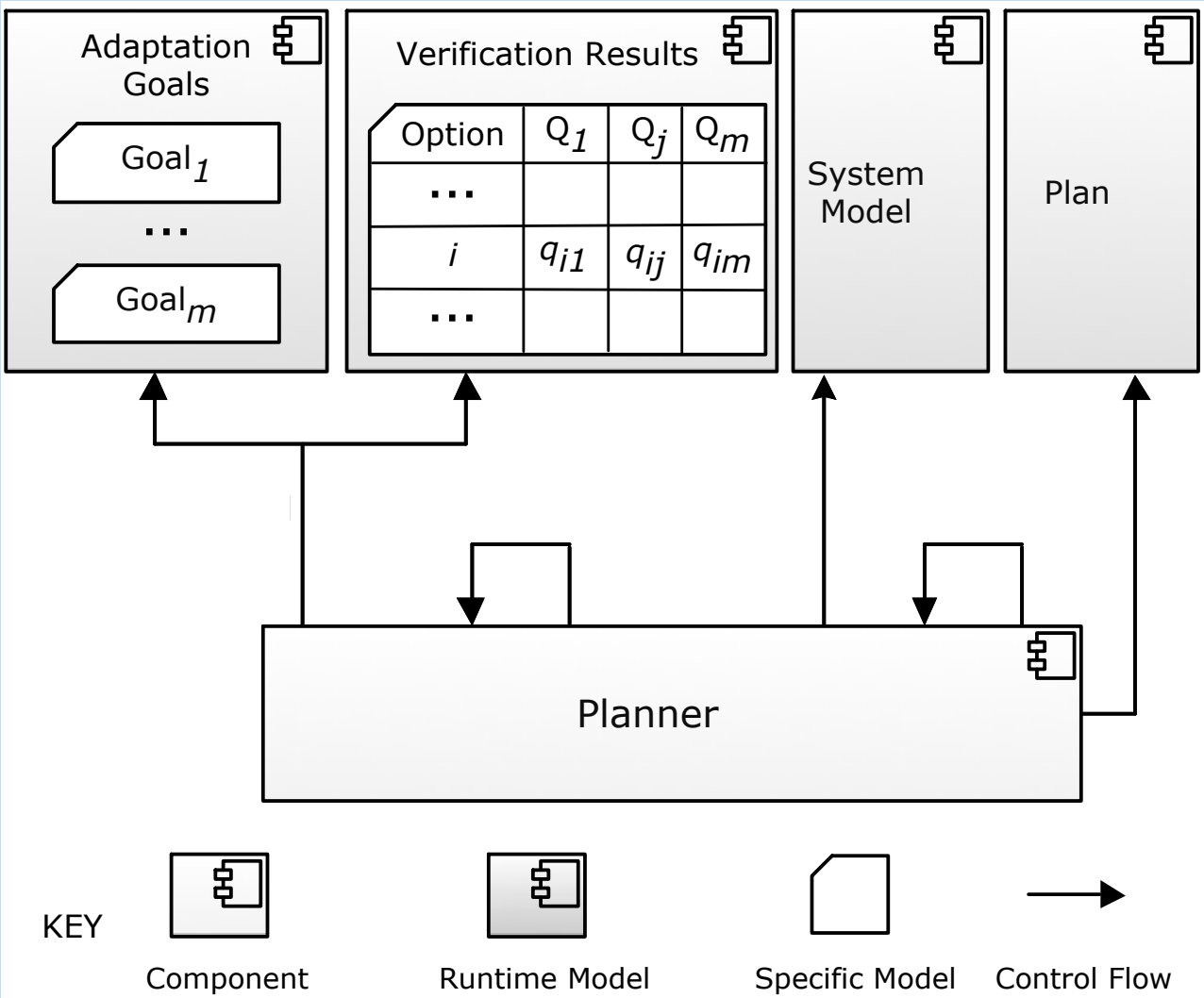
I. Analysis of adaptation options



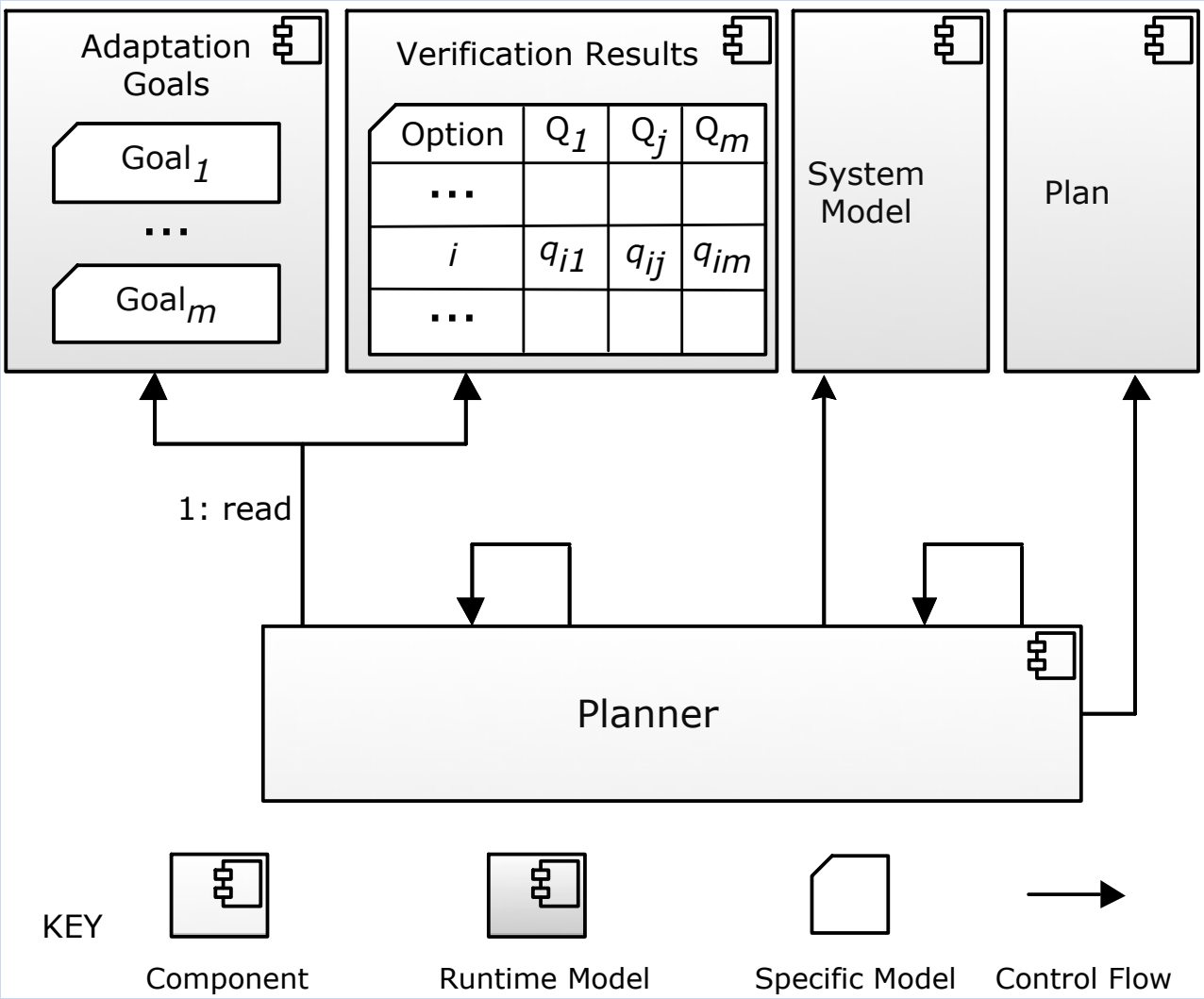
I. Analysis of adaptation options



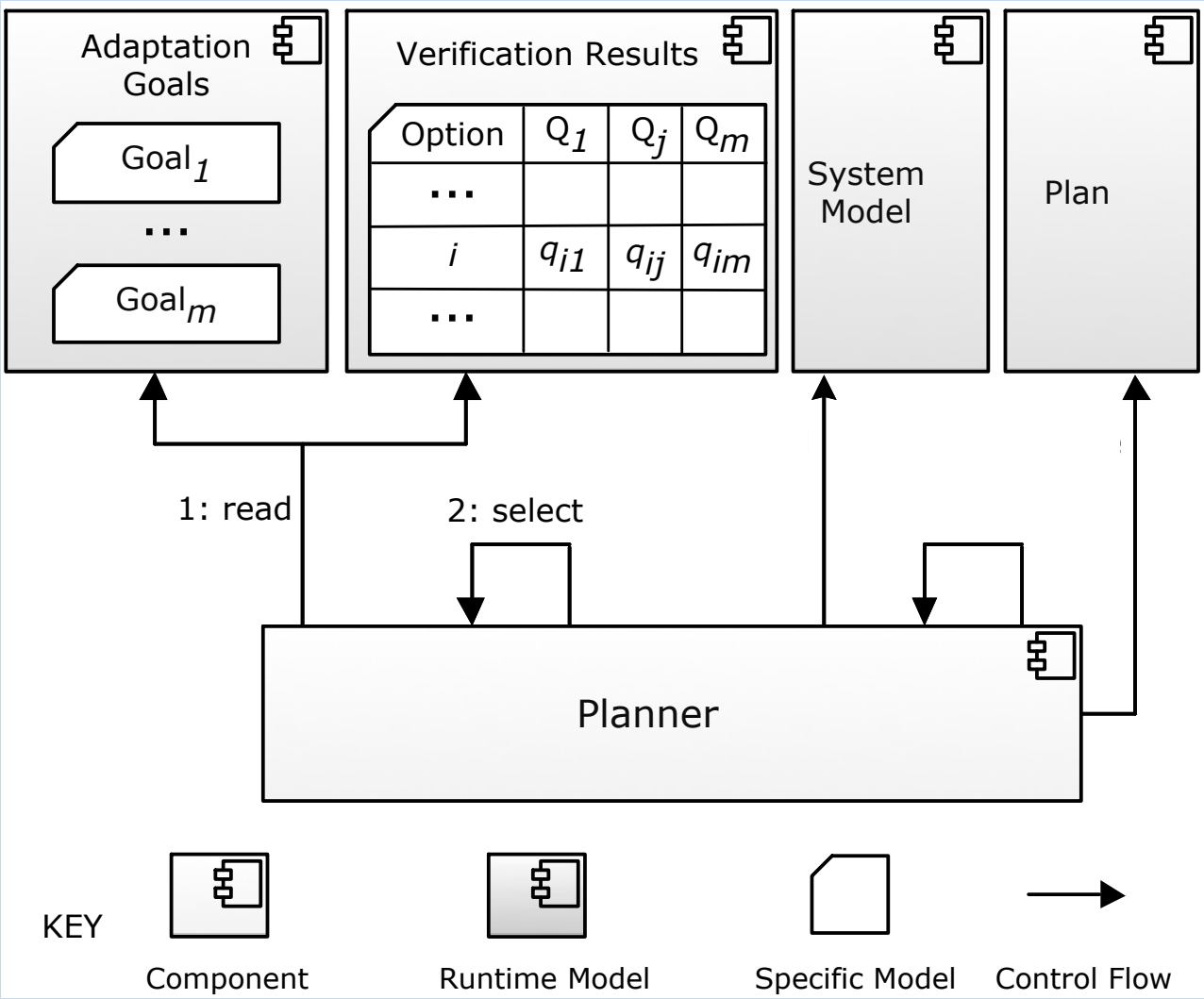
II. Selection of the best adaptation option



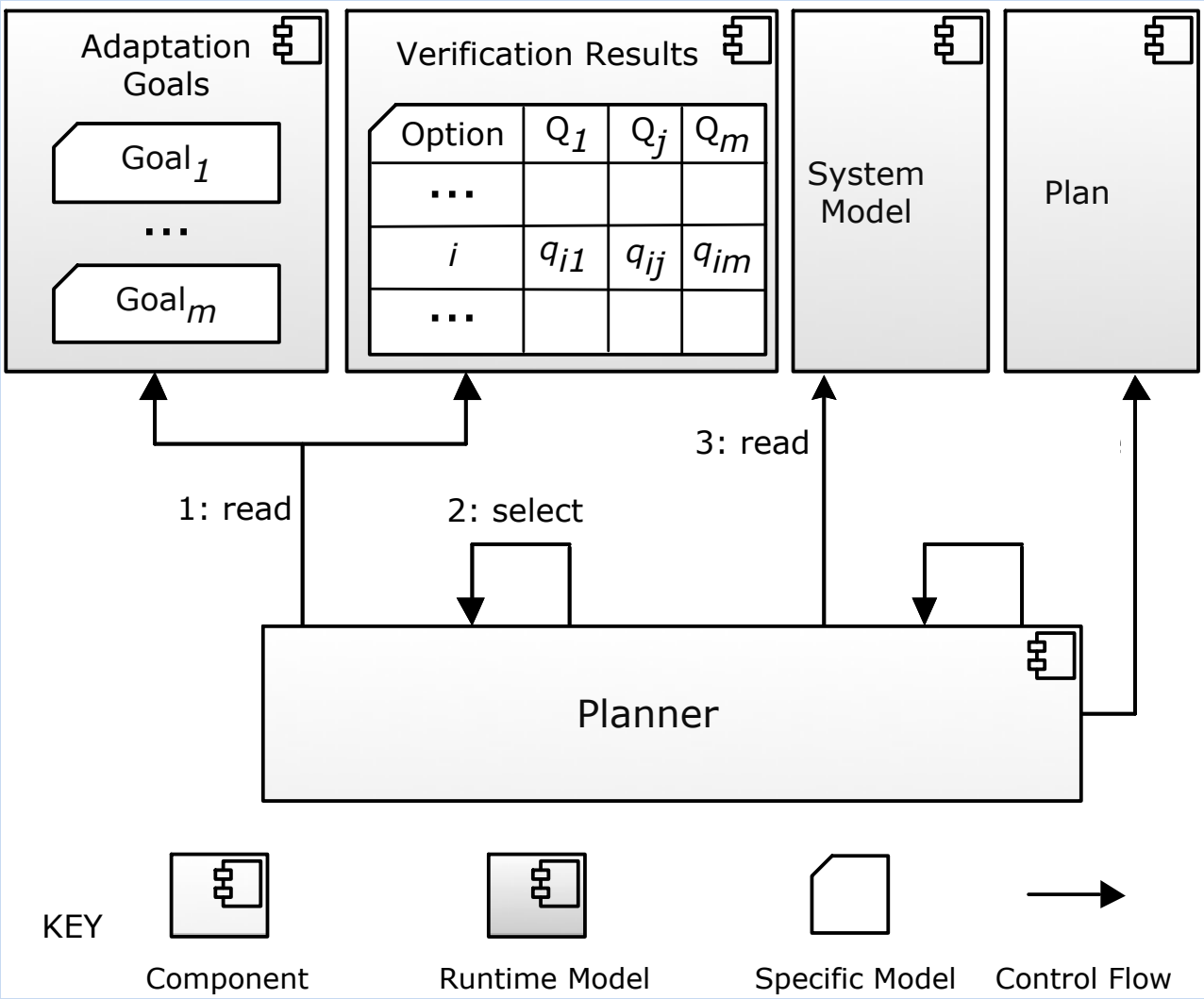
II. Selection of the best adaptation option



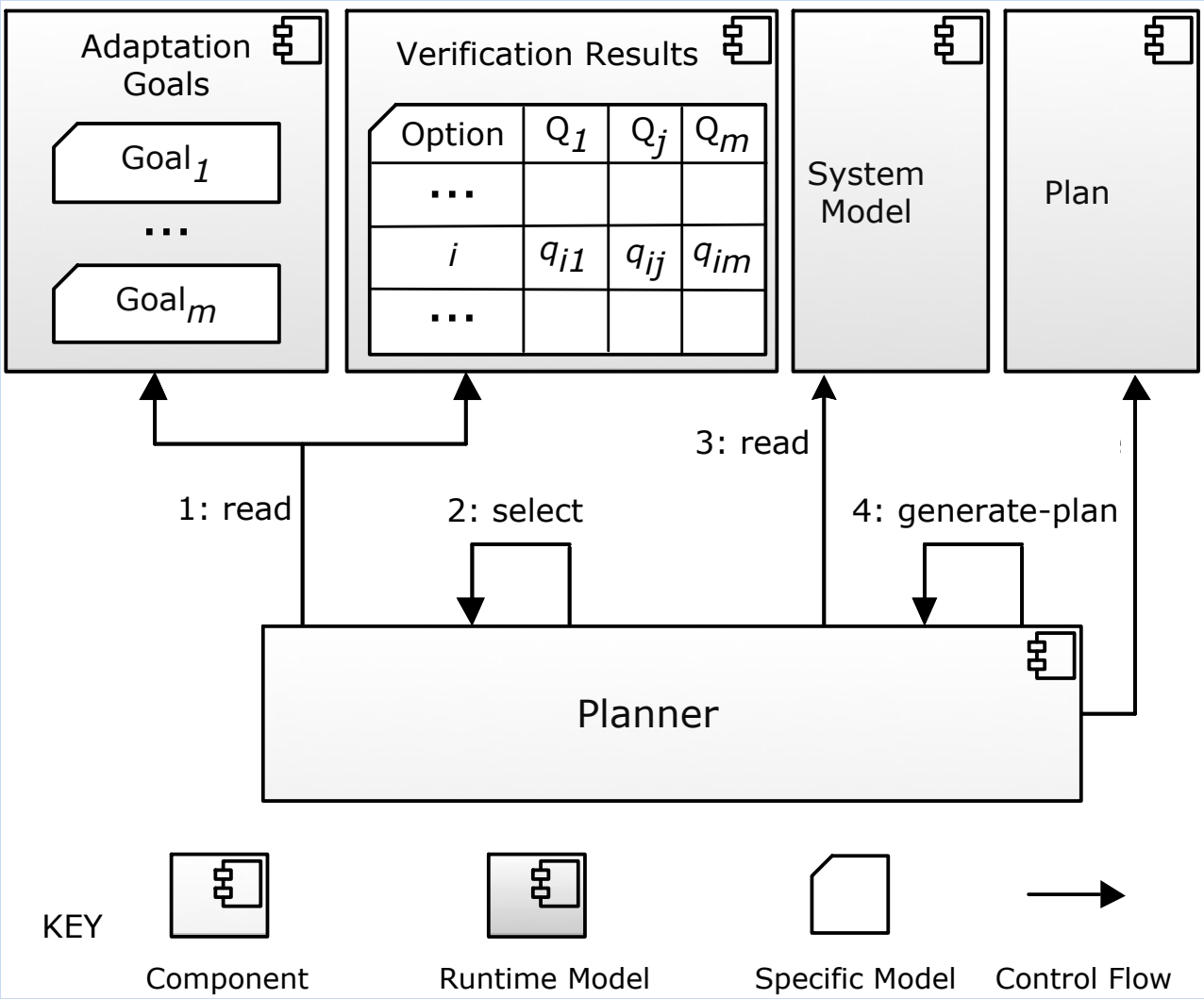
II. Selection of the best adaptation option



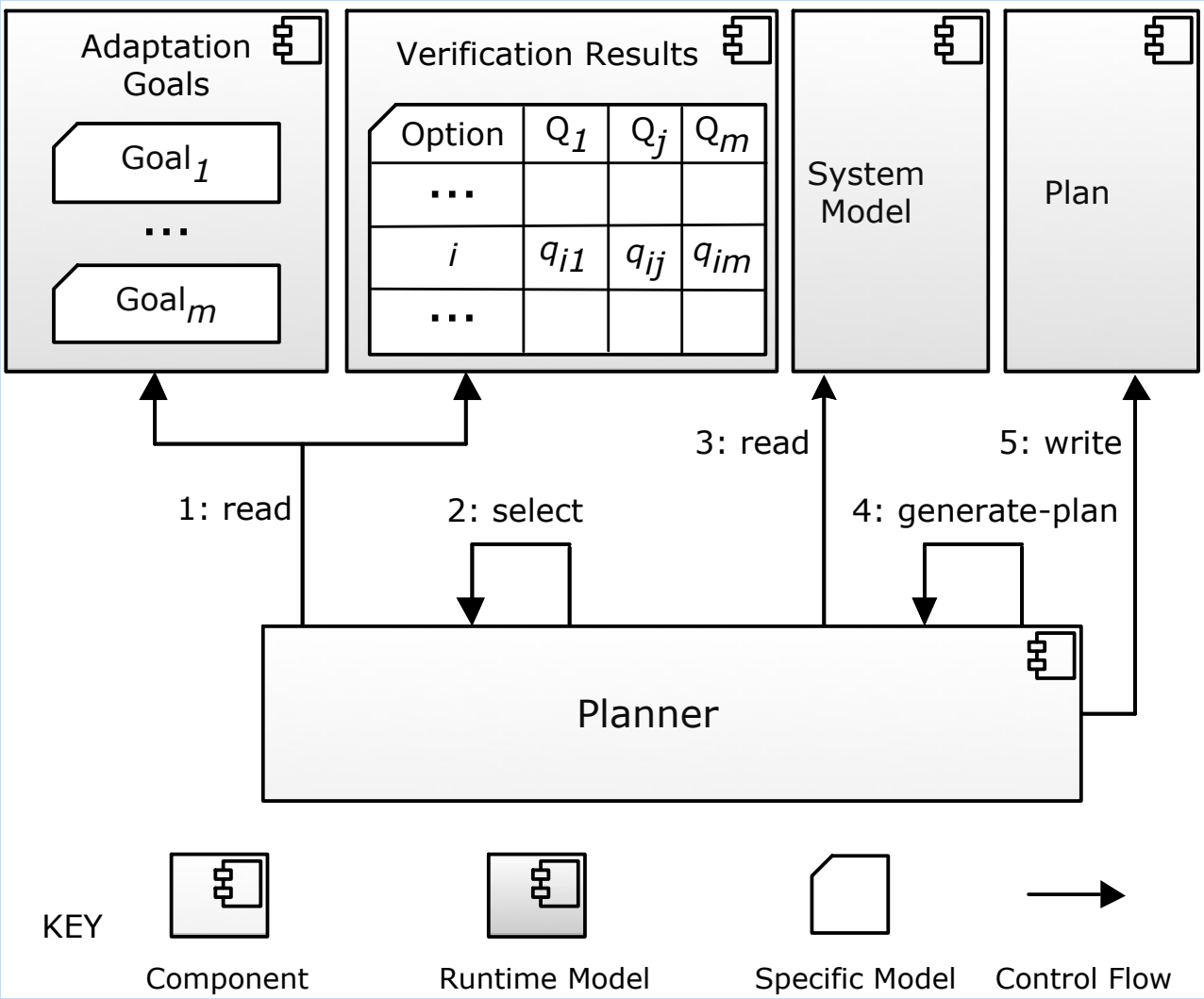
II. Selection of the best adaptation option



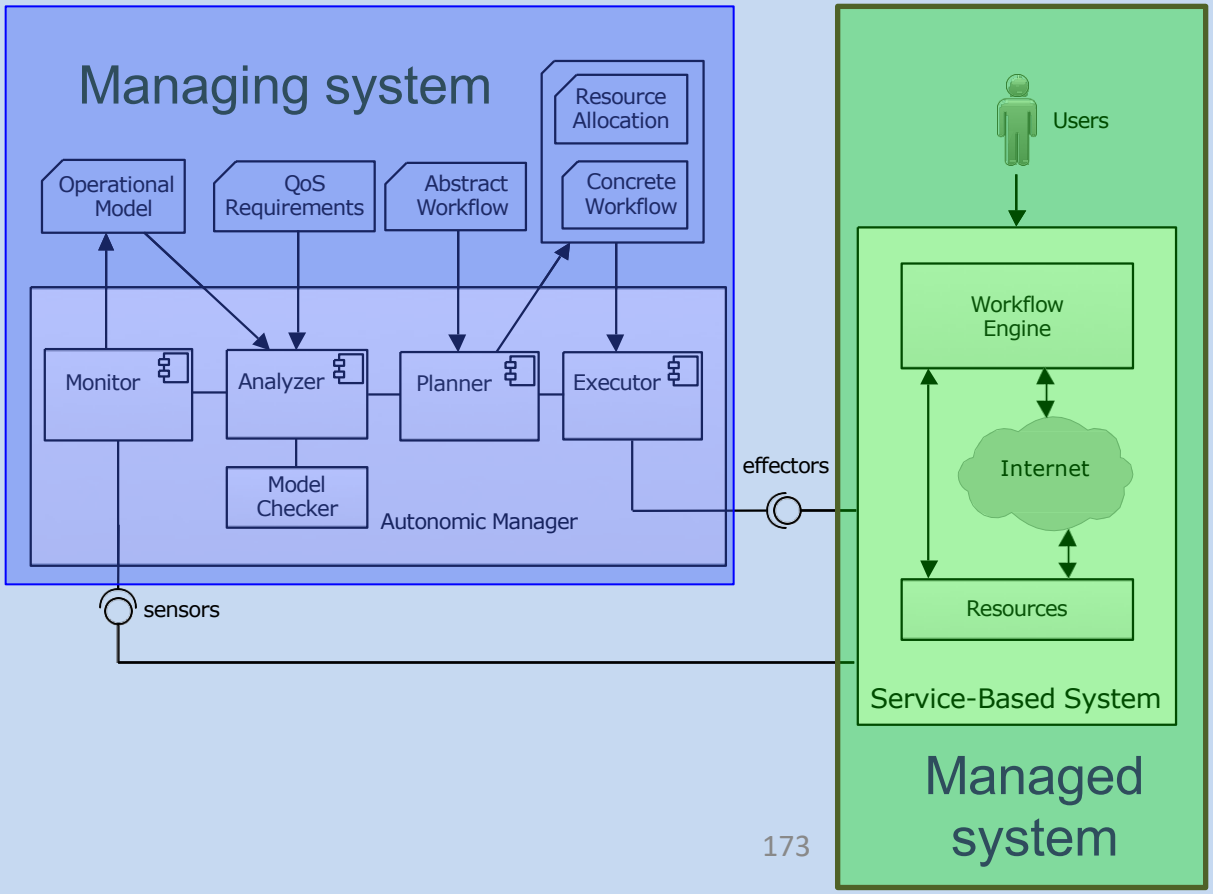
II. Selection of the best adaptation option

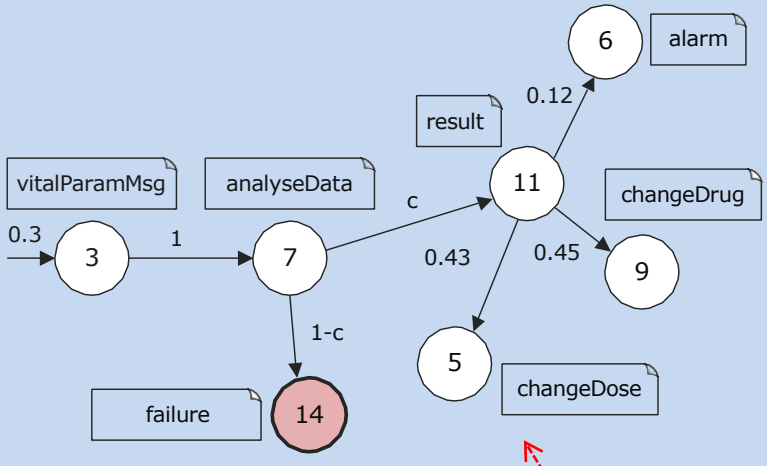


II. Selection of the best adaptation option

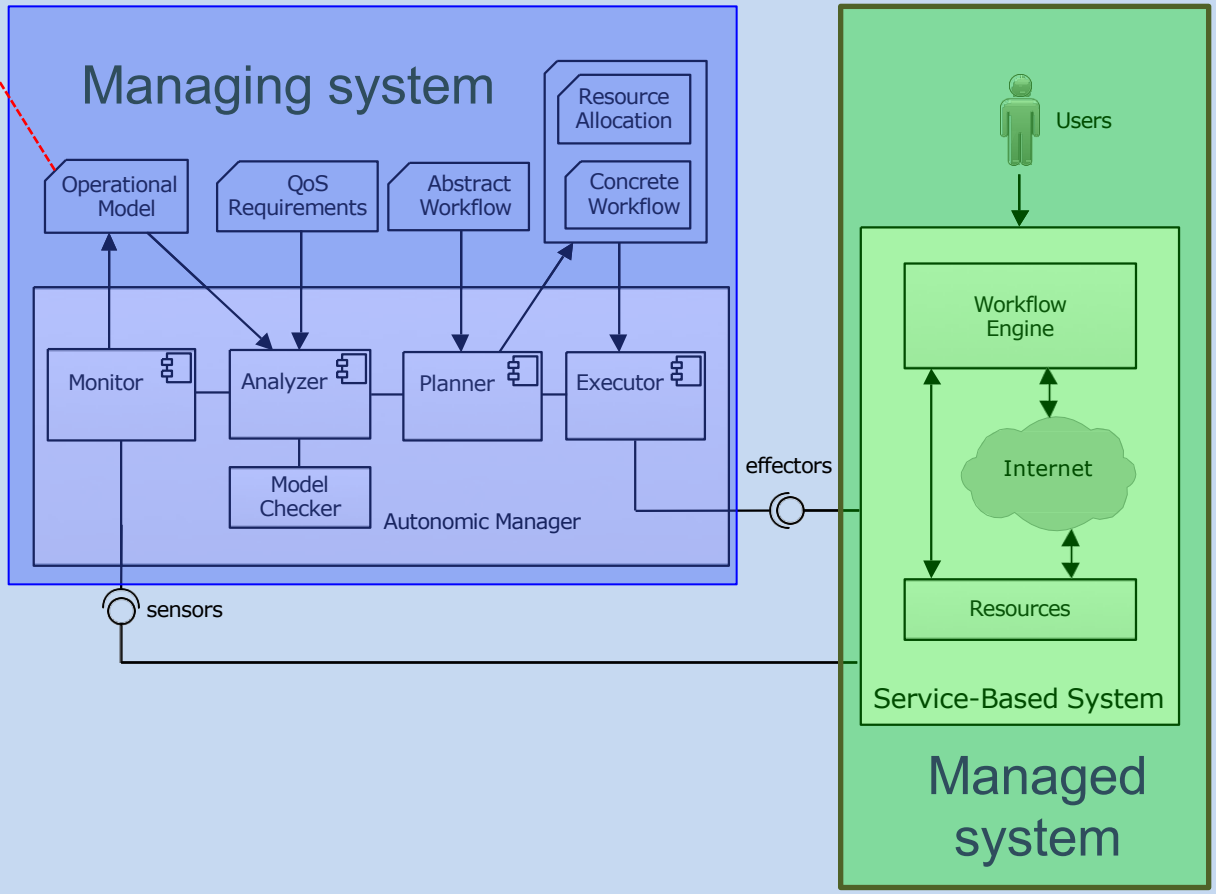


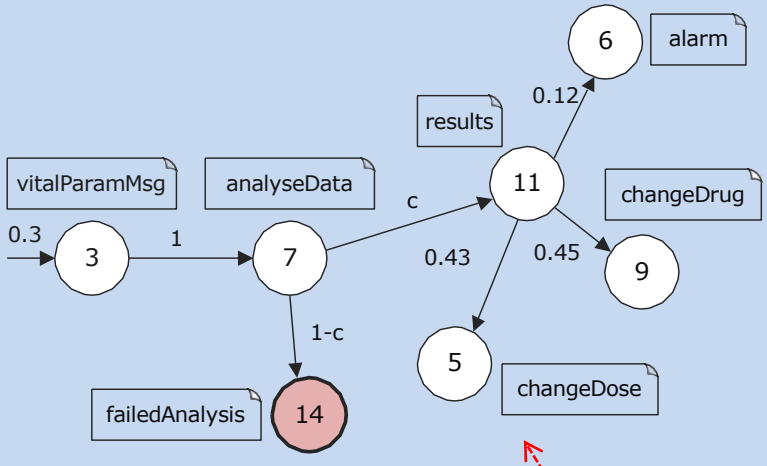
Example


Managing system
Resource Allocation
Concrete Workflow
Operational Model
QoS Requirements
Abstract Workflow
Monitor
Analyzer
Planner
Executor
Model Checker
Autonomic Manager
effectors
sensors
Users
Workflow Engine
Internet
Resources
Service-Based System
Managed system

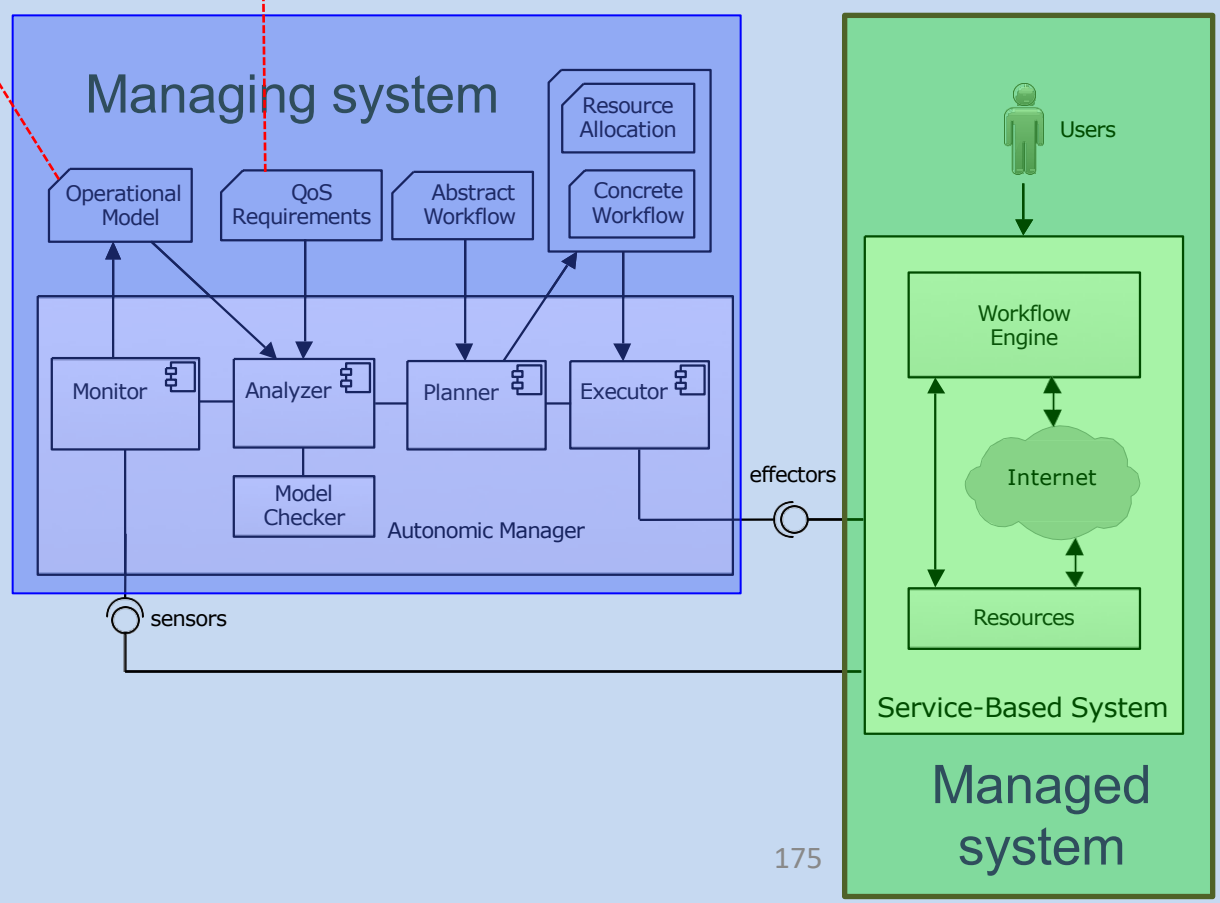


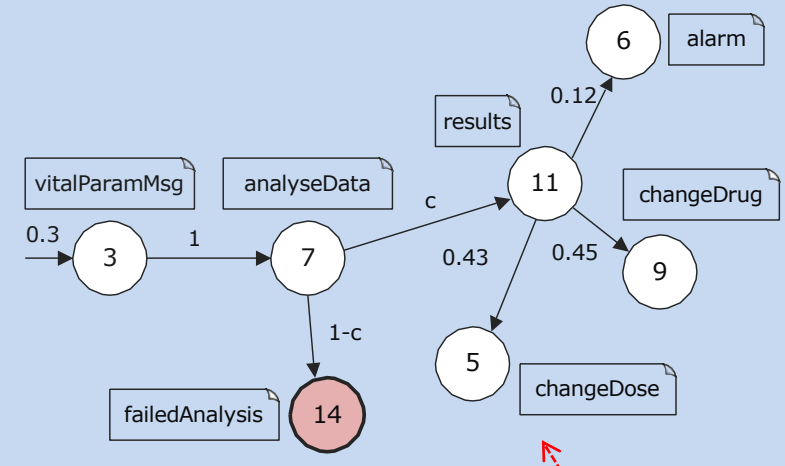
Parameterized model
- Services
- Uncertainties



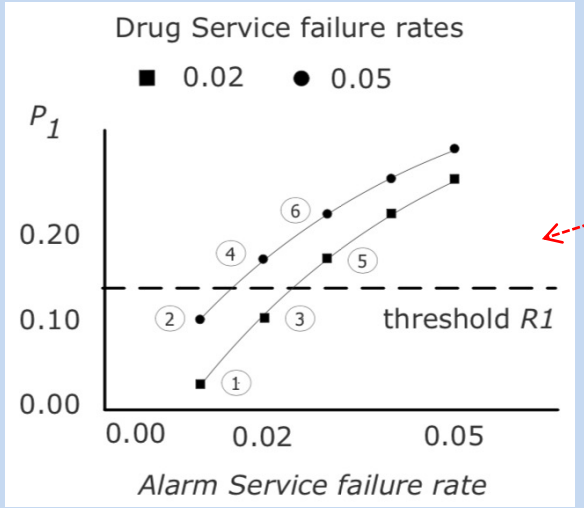


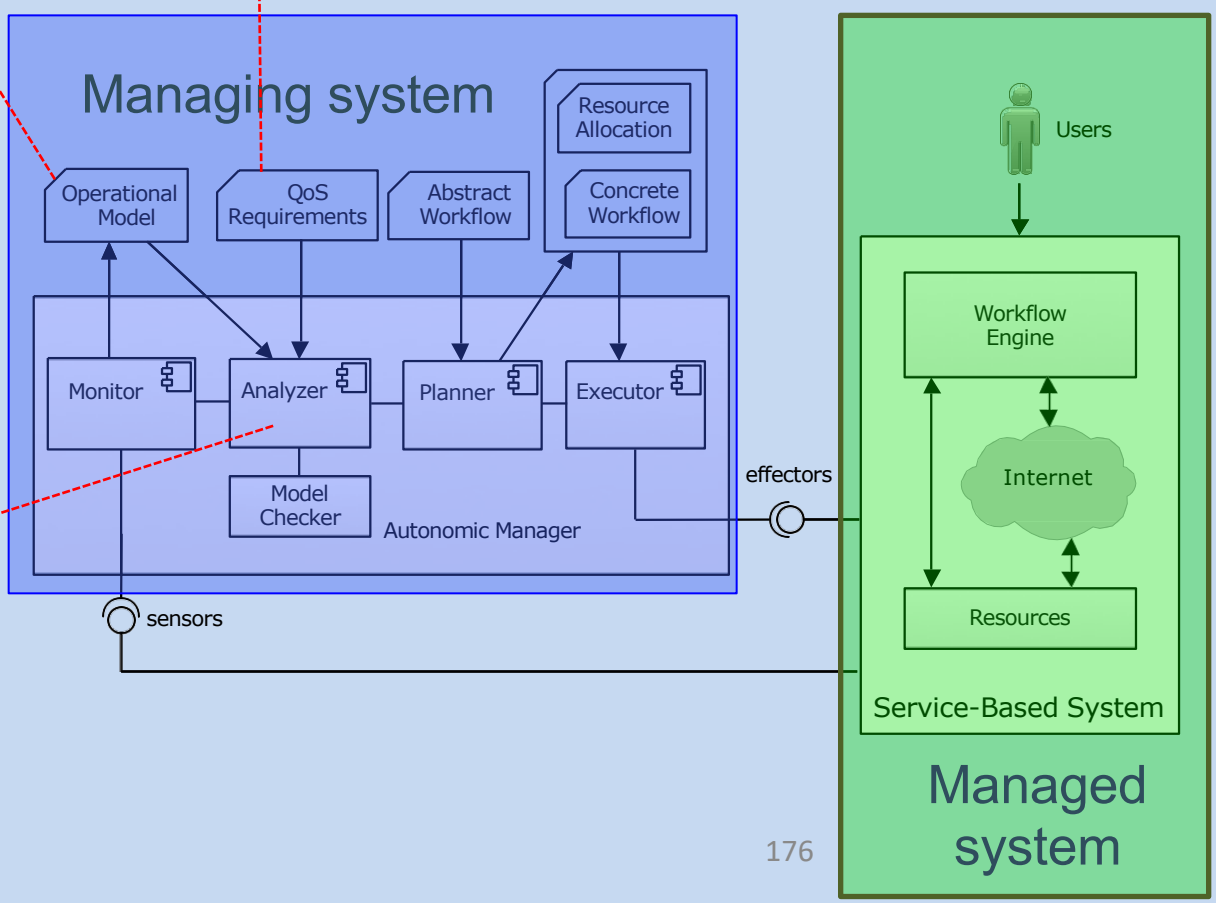
$$R1 : P_{\leq 0.14}[\Diamond \text{“}failure\text{”}]$$

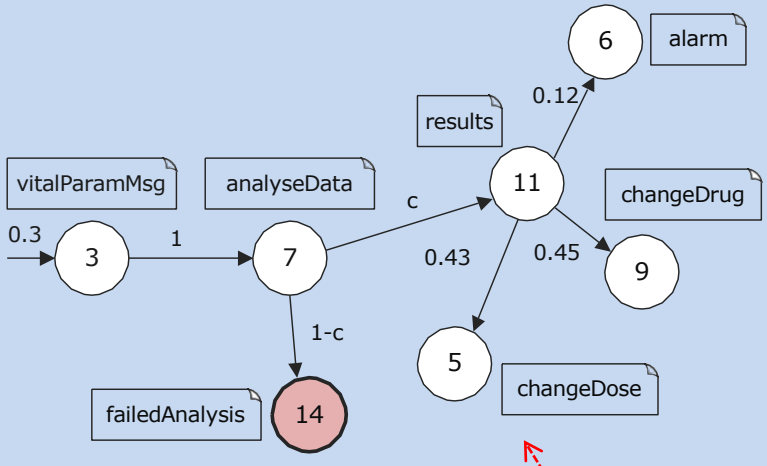




$$R1 : P_{\leq 0.14}[\Diamond \text{“}failure\text{”}]$$

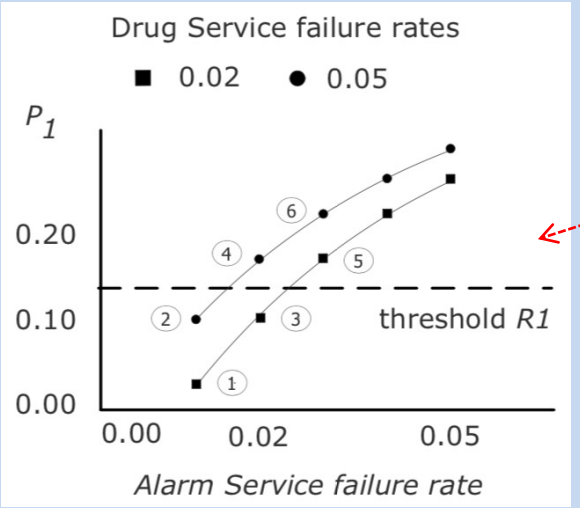


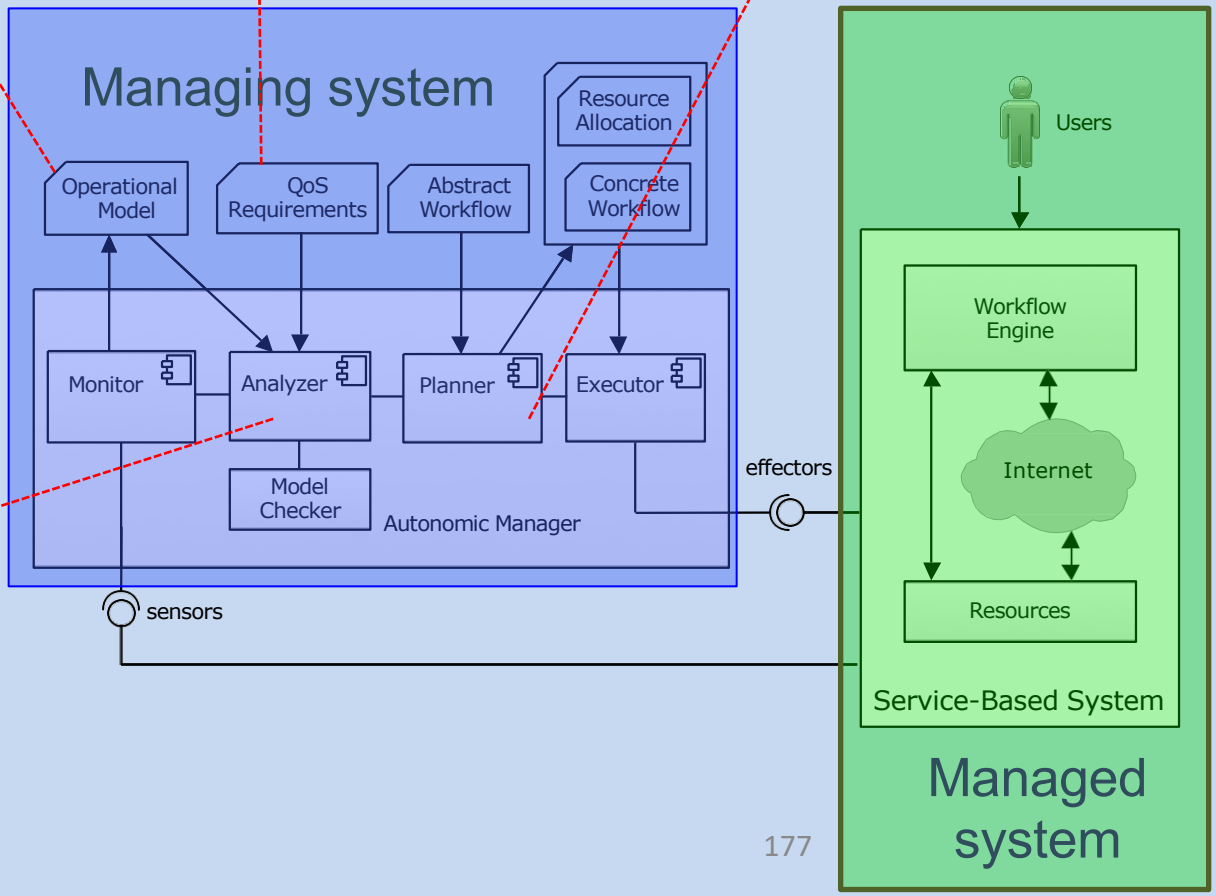




$$R1 : P_{\leq 0.14}[\Diamond \text{“}failure\text{”}]$$

Configuration	Selected service instances	$objective_1$ (failures)	$objective_2$ (cost)	Utility ($\sum w_i . o_i$)
1	{MS1, AS1, DS1}	1	30.8	69.2
2	{MS1, AS1, DS2}	1	27.8	72.2
3	{MS1, AS2, DS1}	1	26.2	73.8
4	{MS1, AS2, DS2}	0	23.2	−23.2
5	{MS1, AS3, DS1}	0	26.2	−26.2
6	{MS1, AS3, DS2}	0	23.2	−23.2





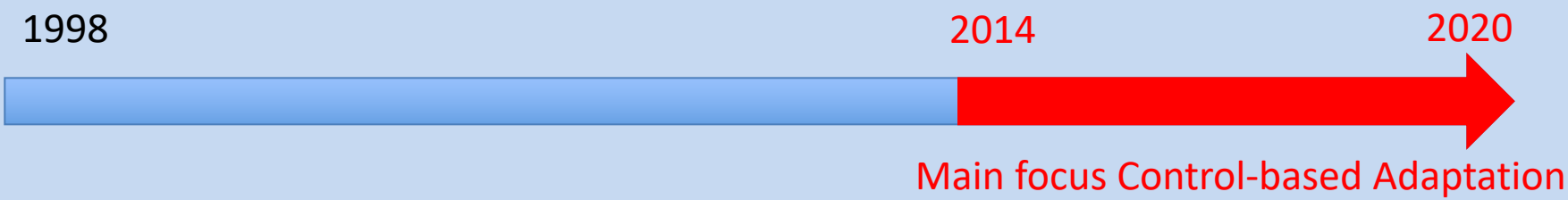
Key insights Wave V

Uncertainty = any deviations in deterministic knowledge that may reduce the confidence in adaptation decisions

Taming uncertainty = providing guarantees for compliance of the system with its adaptation goals.

Verification based on models needs to be complemented by validation of the system

Wave VI
Control-based Adaptation



Driver: Engineering of MAPE-based self-adaptive systems is often complex

Topic: Apply principles from control theory to realize self-adaptation

Enabled: Control theory as a basis for the design and formal analysis of self-adaptive systems

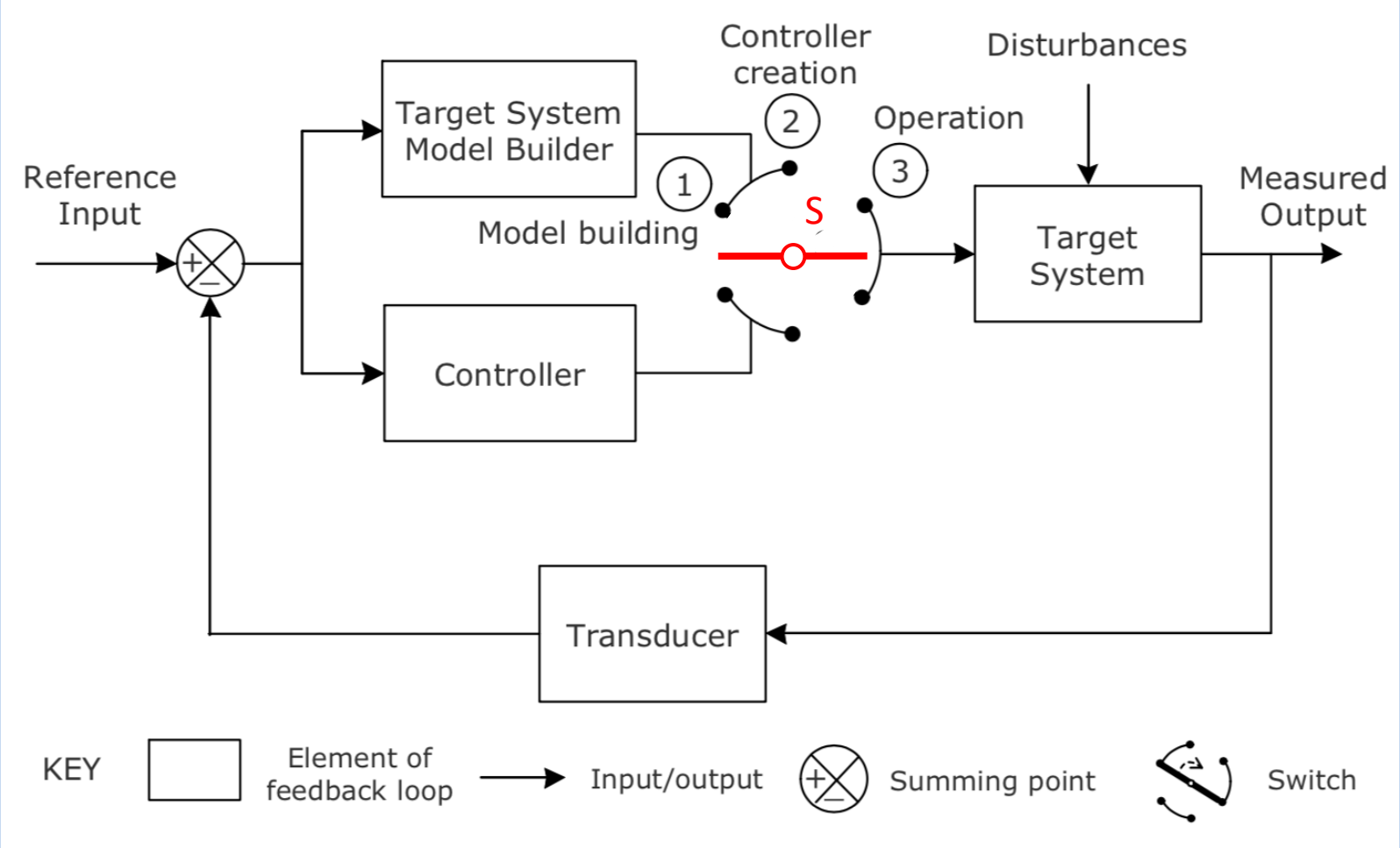
Control theory for self-adaptive systems

Formal guarantees of control properties (stability etc.)

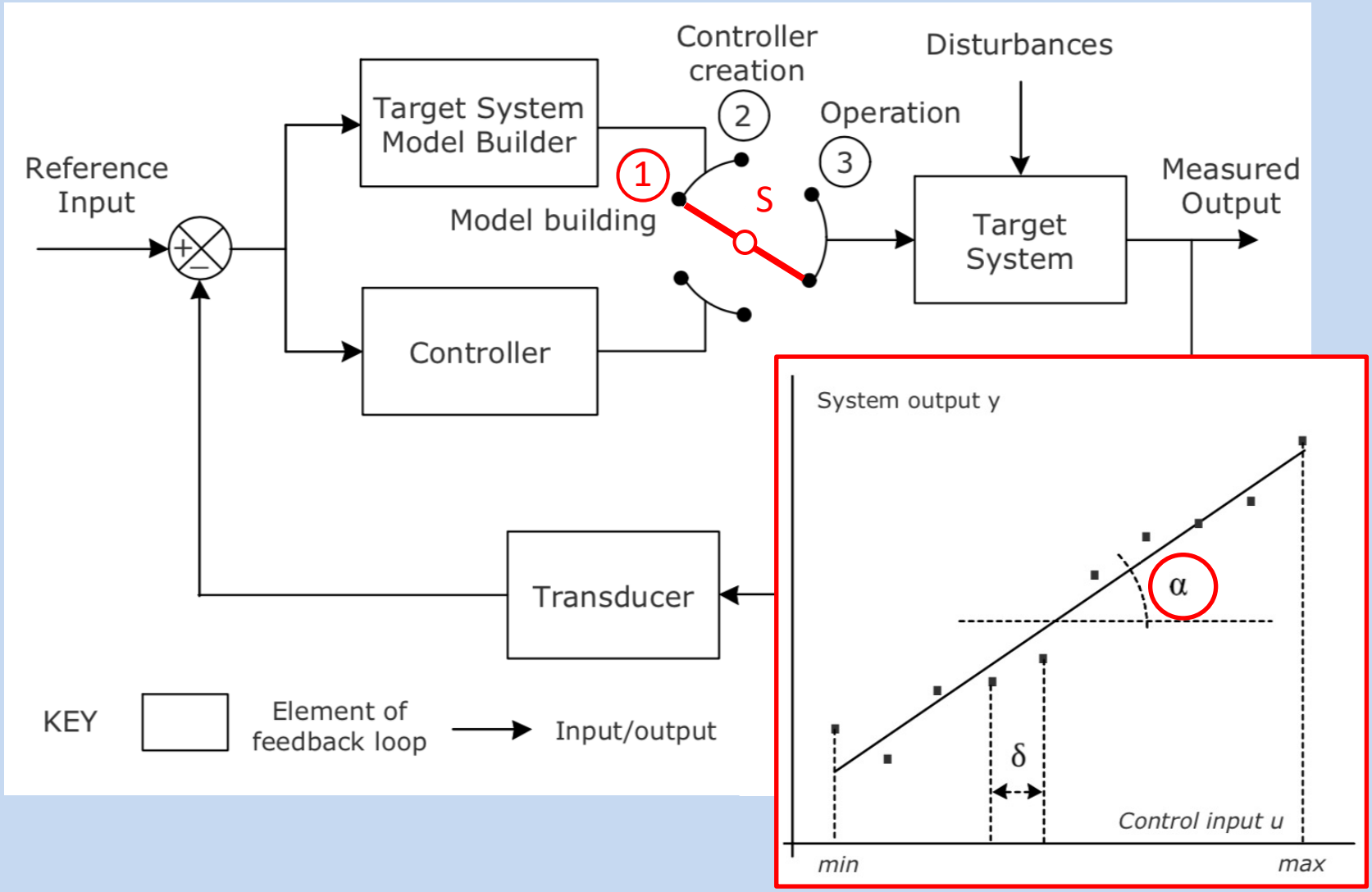
Focus on design and analysis of feedback loop systems

Due to complex mathematics focus on automatic construction of feedback loop systems

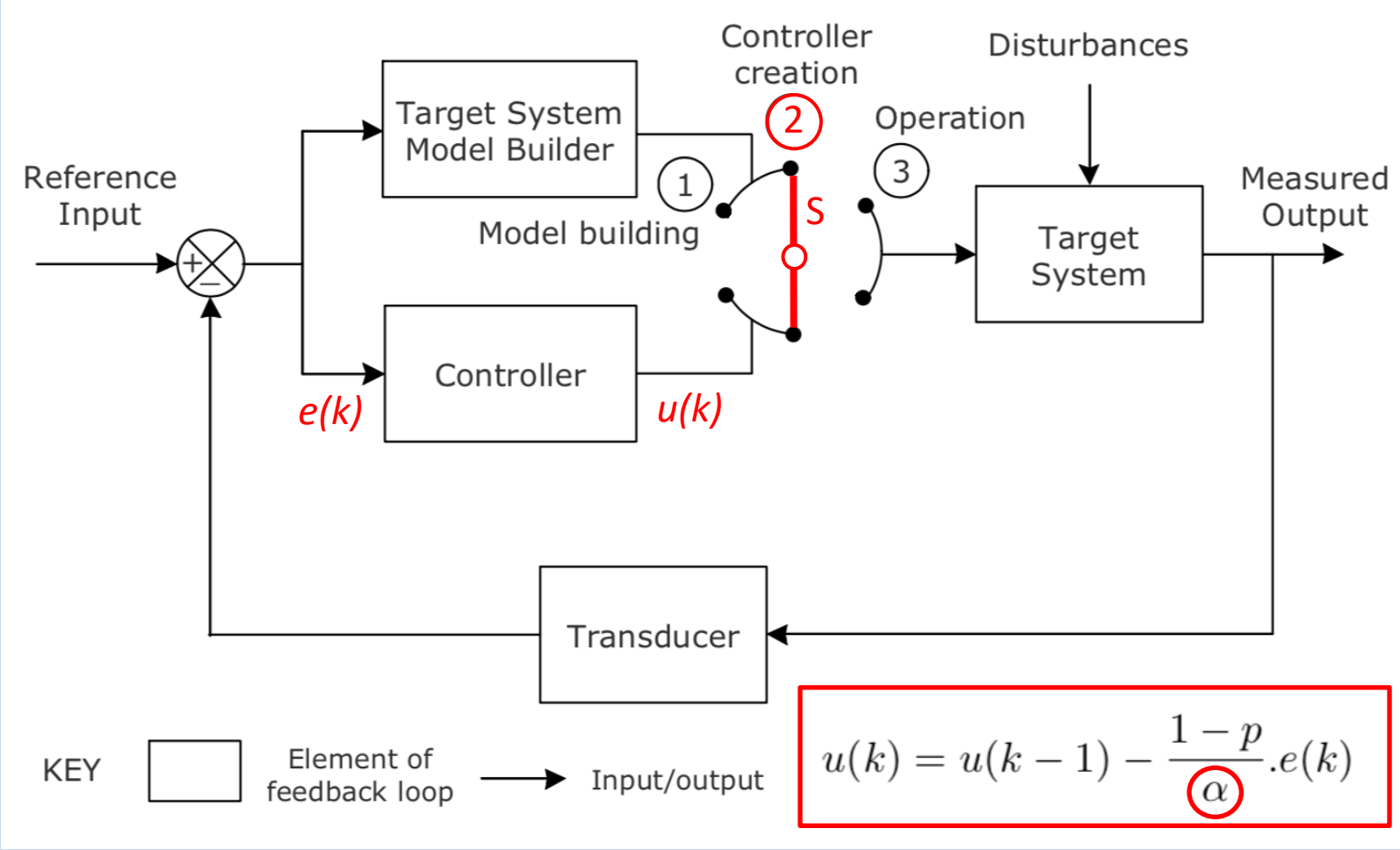
Phases of Controller Construction & Operation



Phases of Controller Construction & Operation

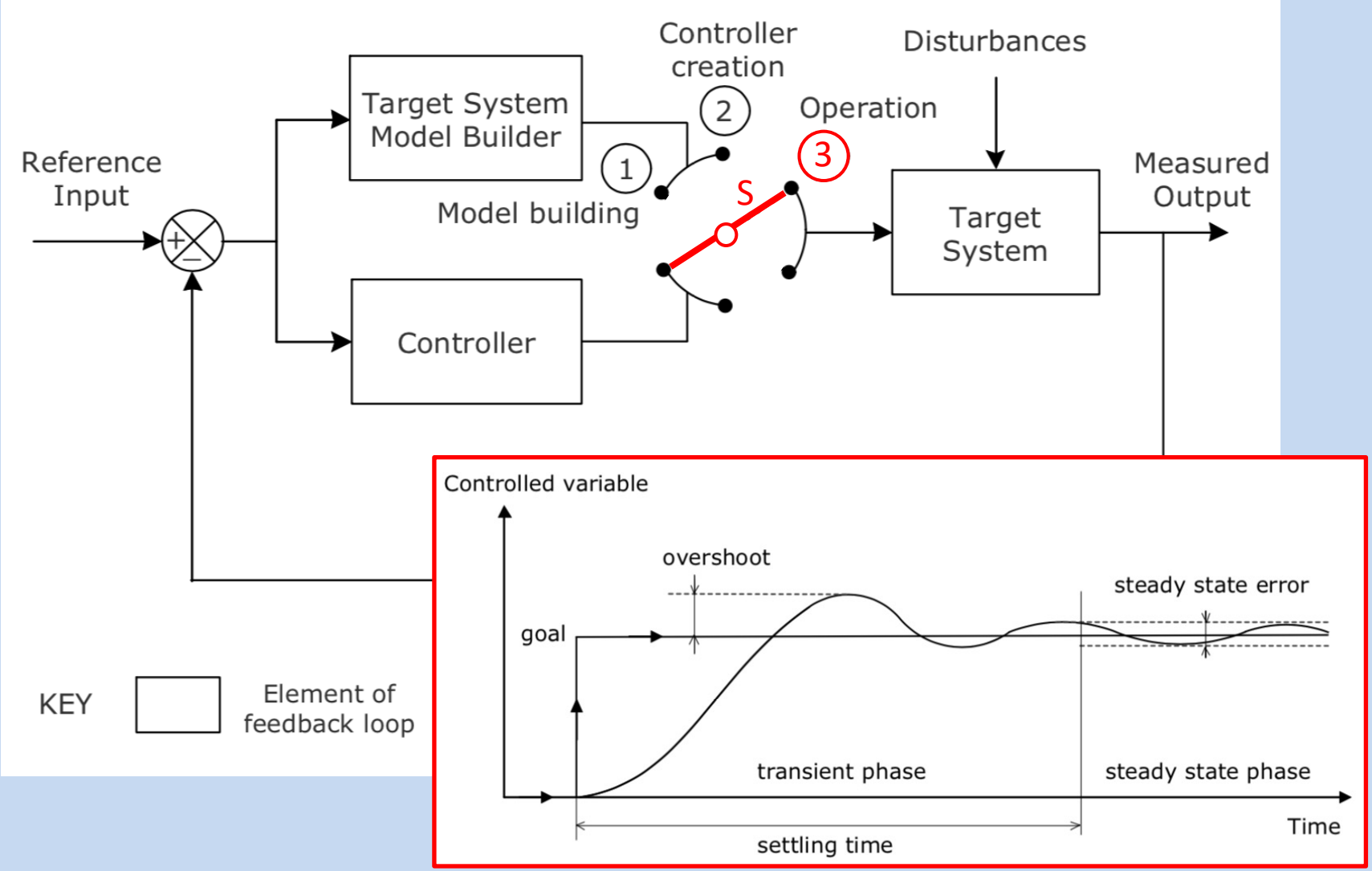


Phases of Controller Construction & Operation

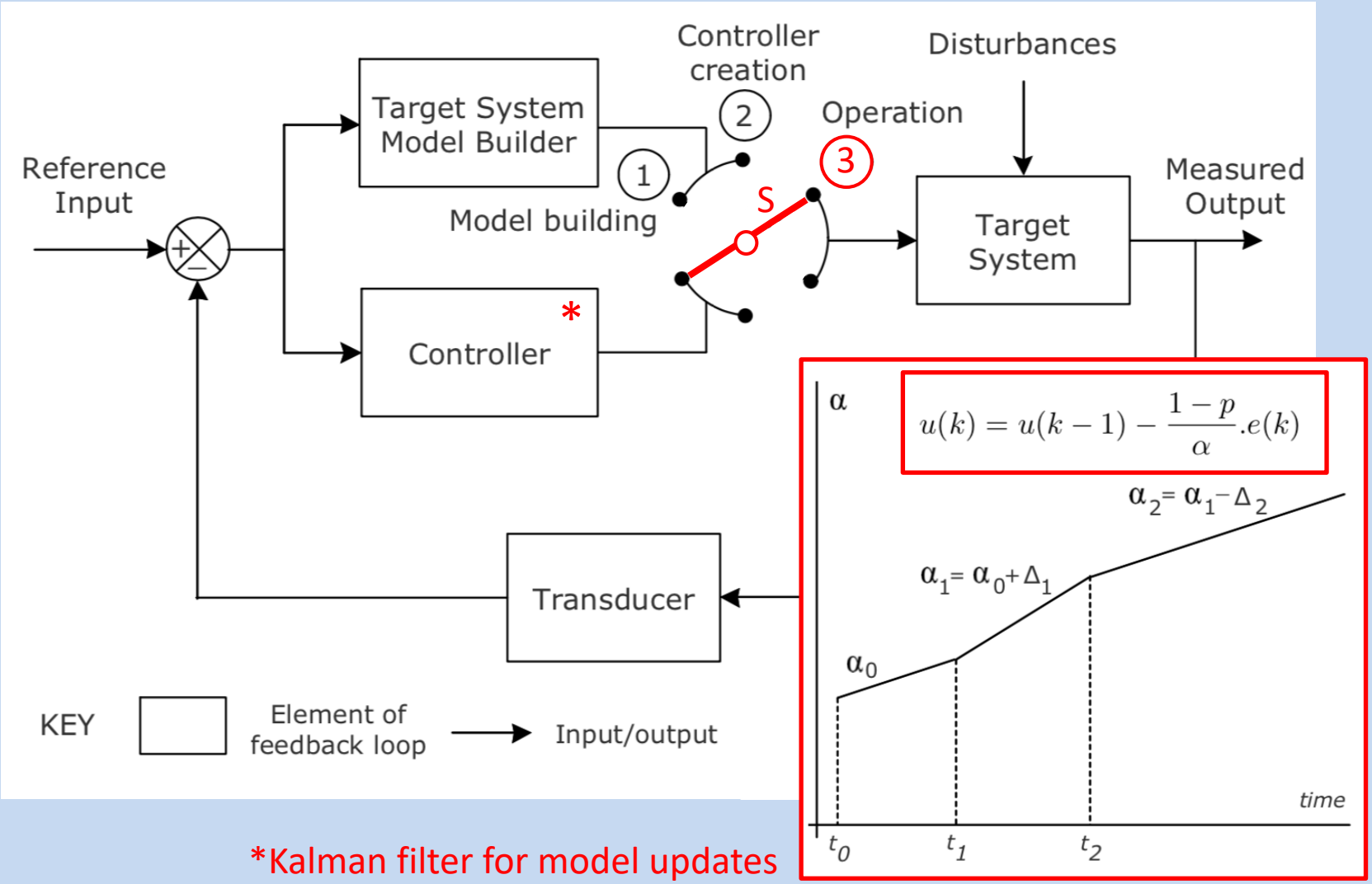


$$u(k) = u(k-1) - \frac{1-p}{\alpha}.e(k)$$

Phases of Controller Construction & Operation



Phases of Controller Construction & Operation

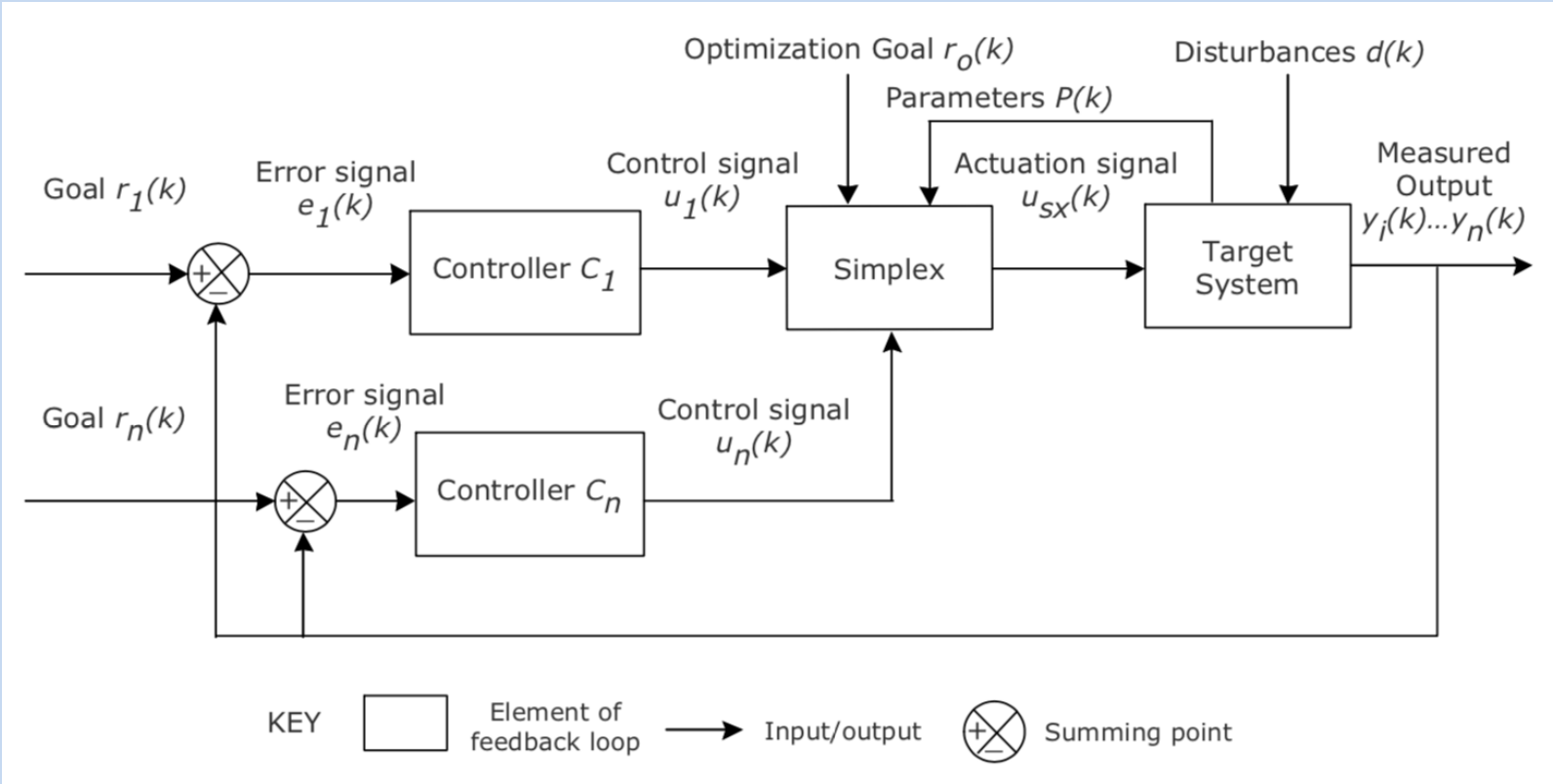


*Kalman filter for model updates

Example control system for multiple goals

Optimization goal

Set point goals



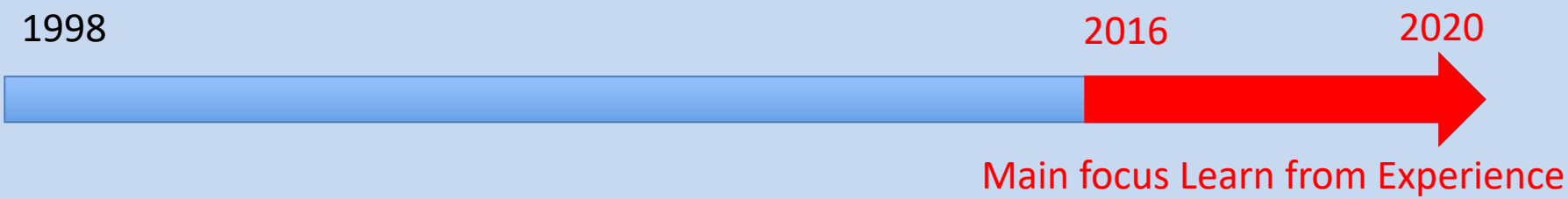
Key insights Wave VI

Control schemas for self-adaptive software can be automatically constructed

Linear model of the system allows various software control problems to be solved

Control guarantees (stability, settling time, …) apply to the model and need to be complemented with validation

Wave VII
Learn from Experience

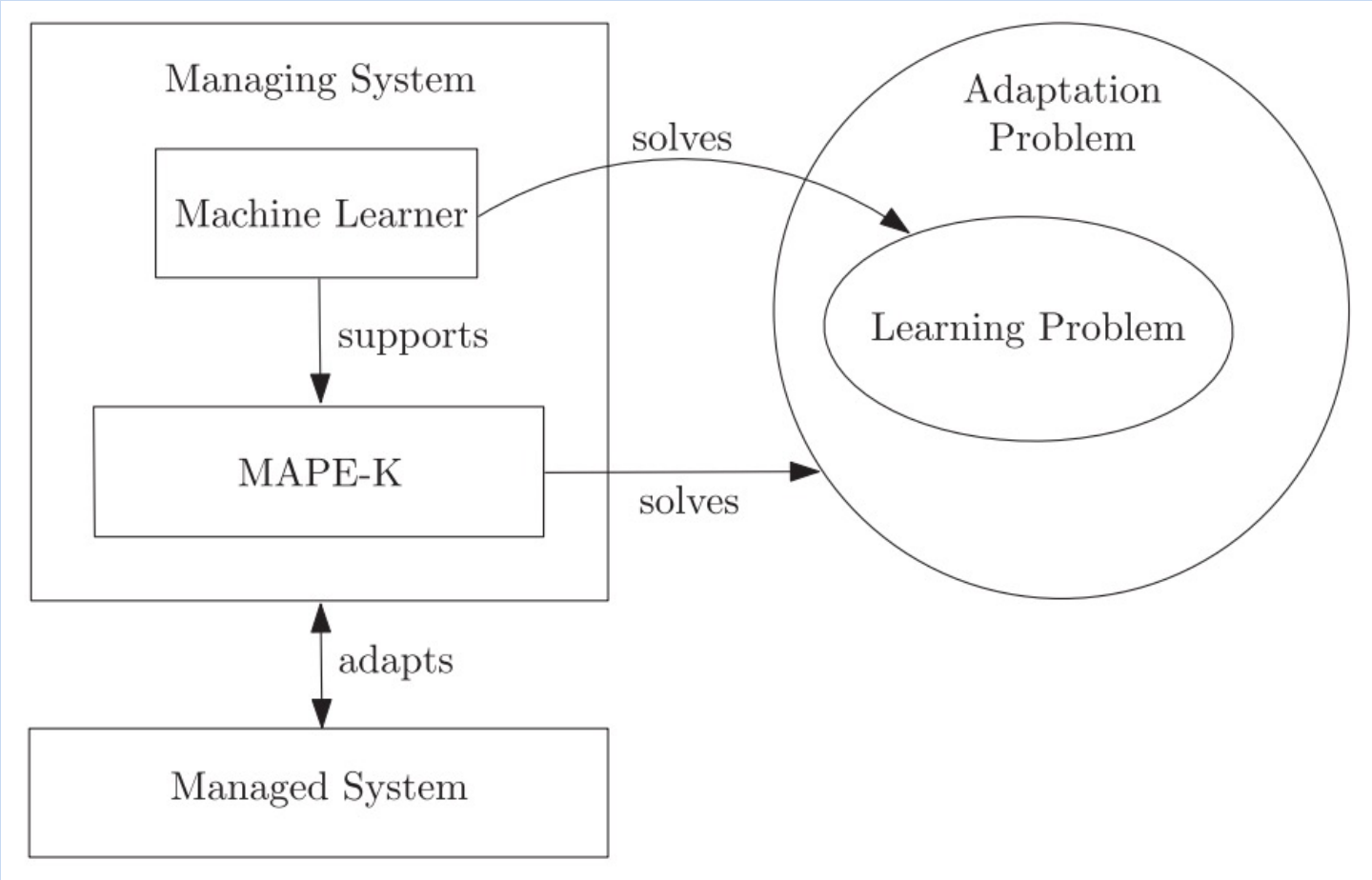


Driver: Growing scale and complex levels of uncertainty

Topic: The use of learning techniques in self-adaptation

Enabled: Learning techniques to support different adaptation functions

Learning for self-adaptation



Learning for self-adaptive systems

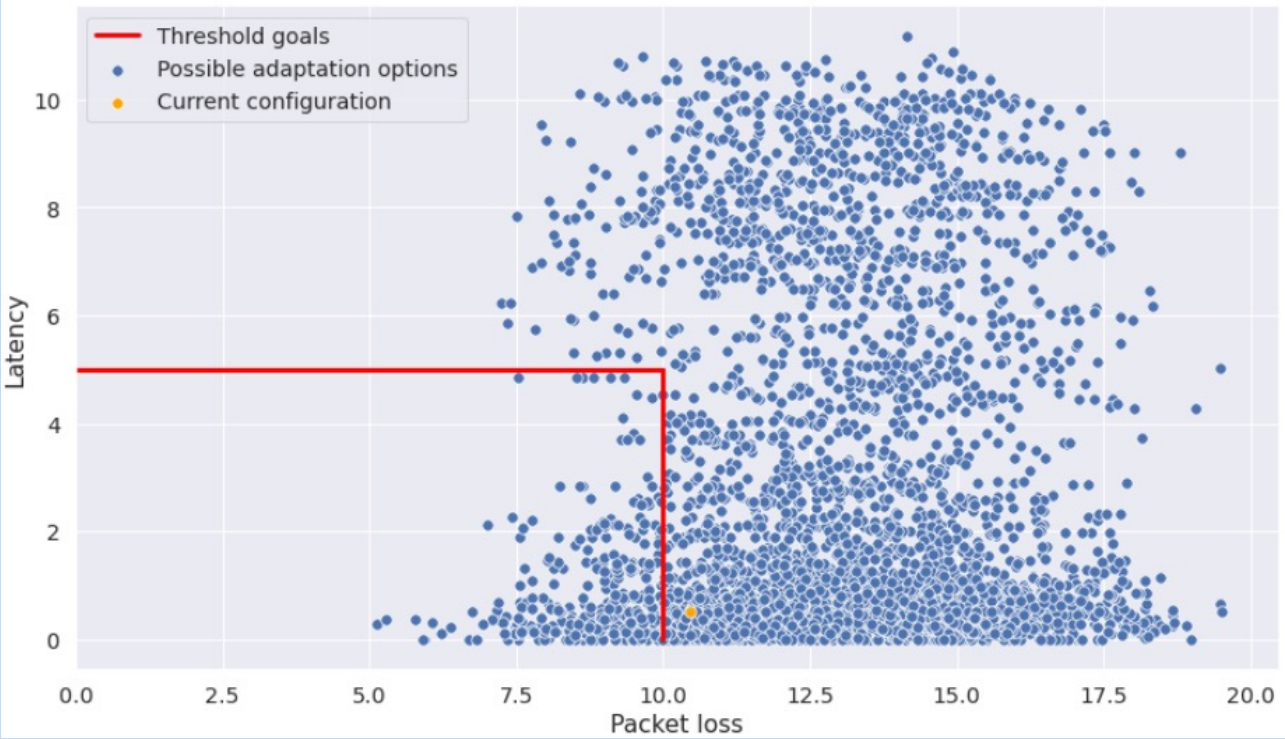
Learning to support different MAPE stages; e.g.

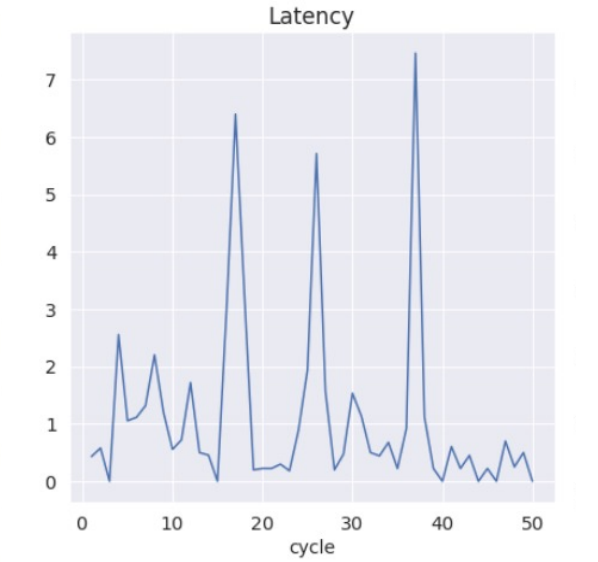
- Bayesian estimator keeps run-time model up to date
- Deep learning reduces large adaptation spaces
- Q-learning improves auto-scaling rules of a Cloud

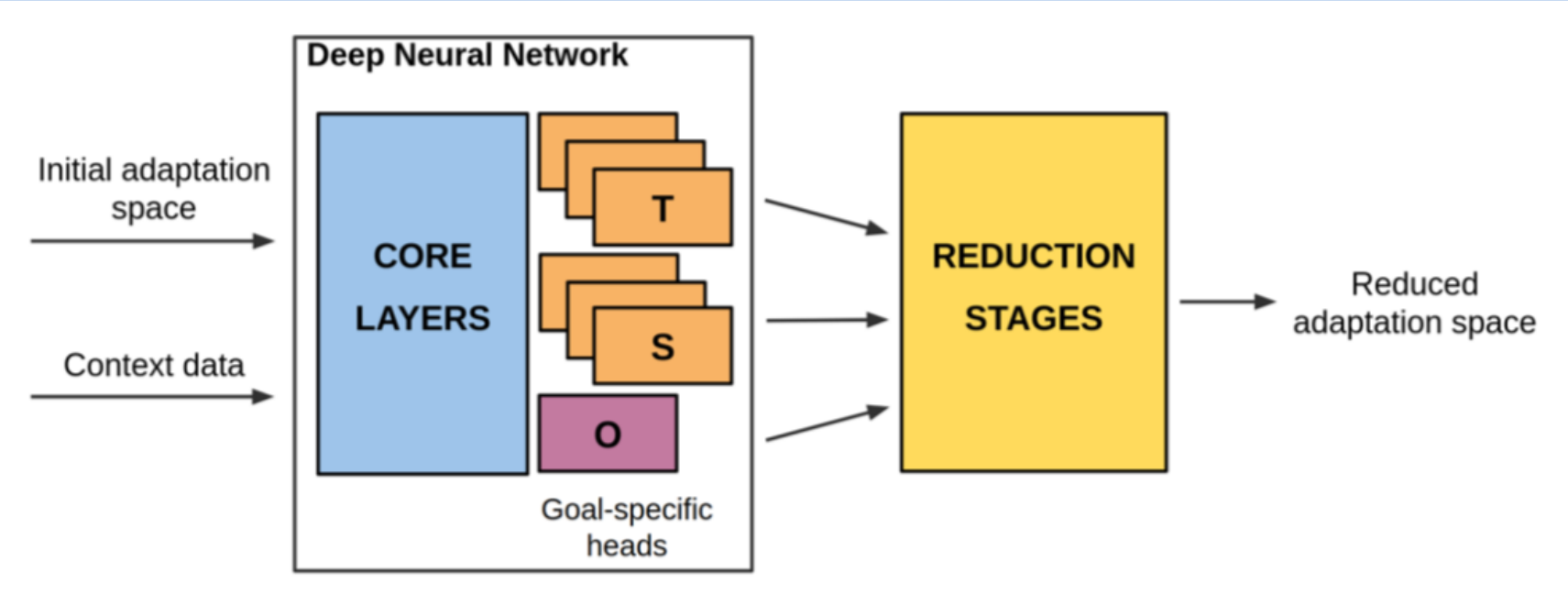
Learning for self-adaptive systems

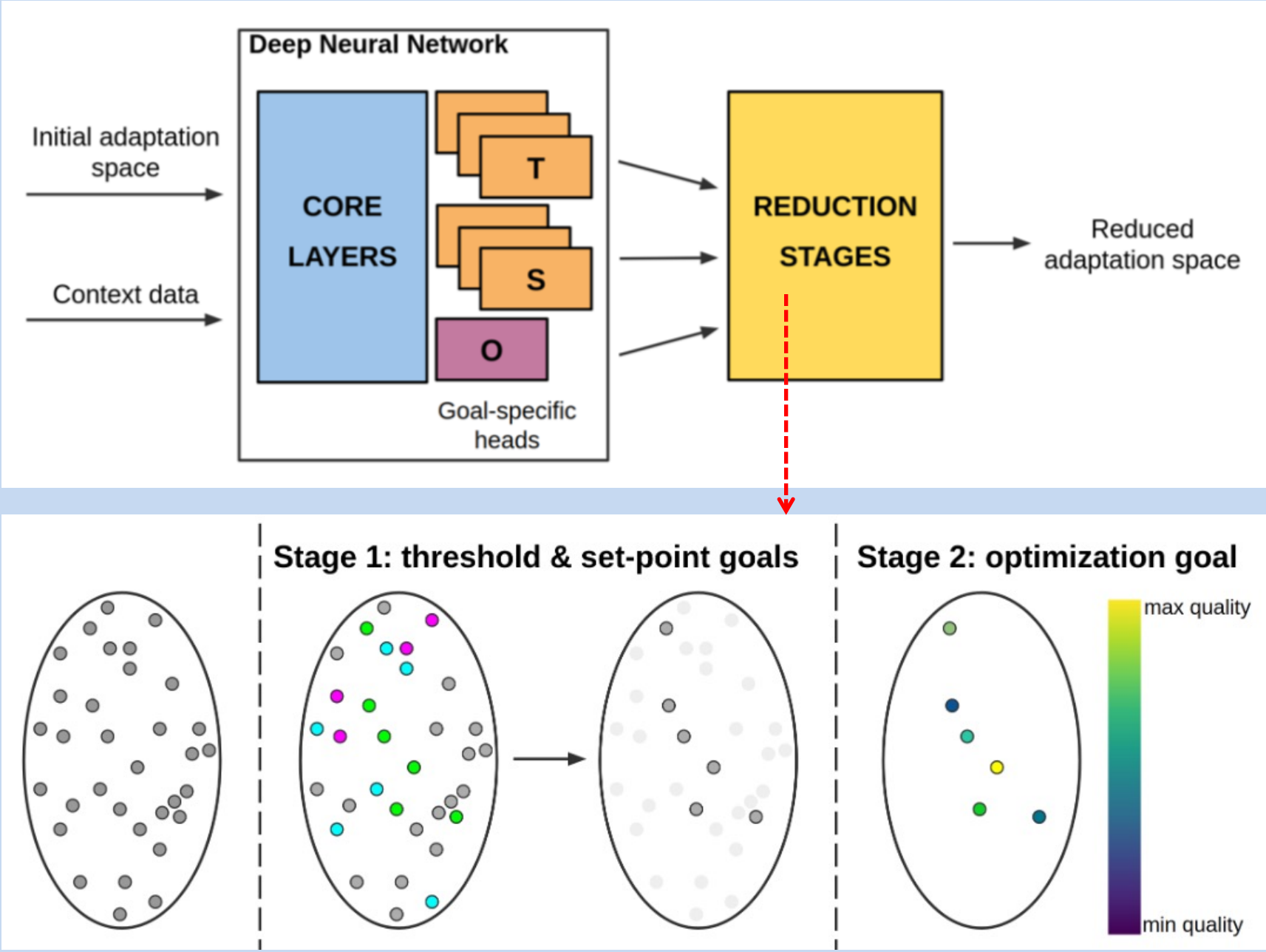
Learning to support different MAPE stages; e.g.

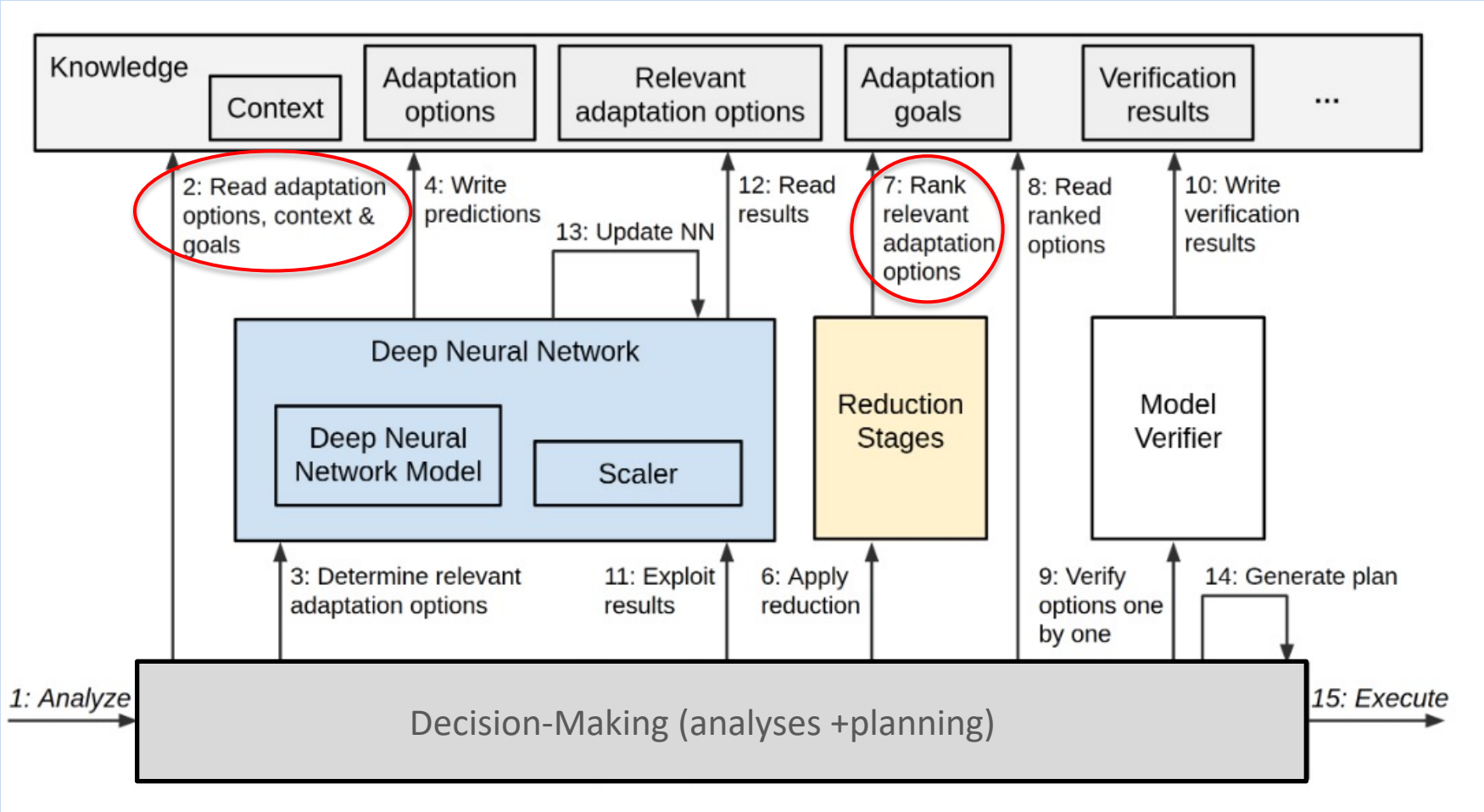
- Bayesian estimator keeps run-time model up to date
- <u>Deep learning reduces large adaptation spaces</u>
- Q-learning improves auto-scaling rules of a Cloud

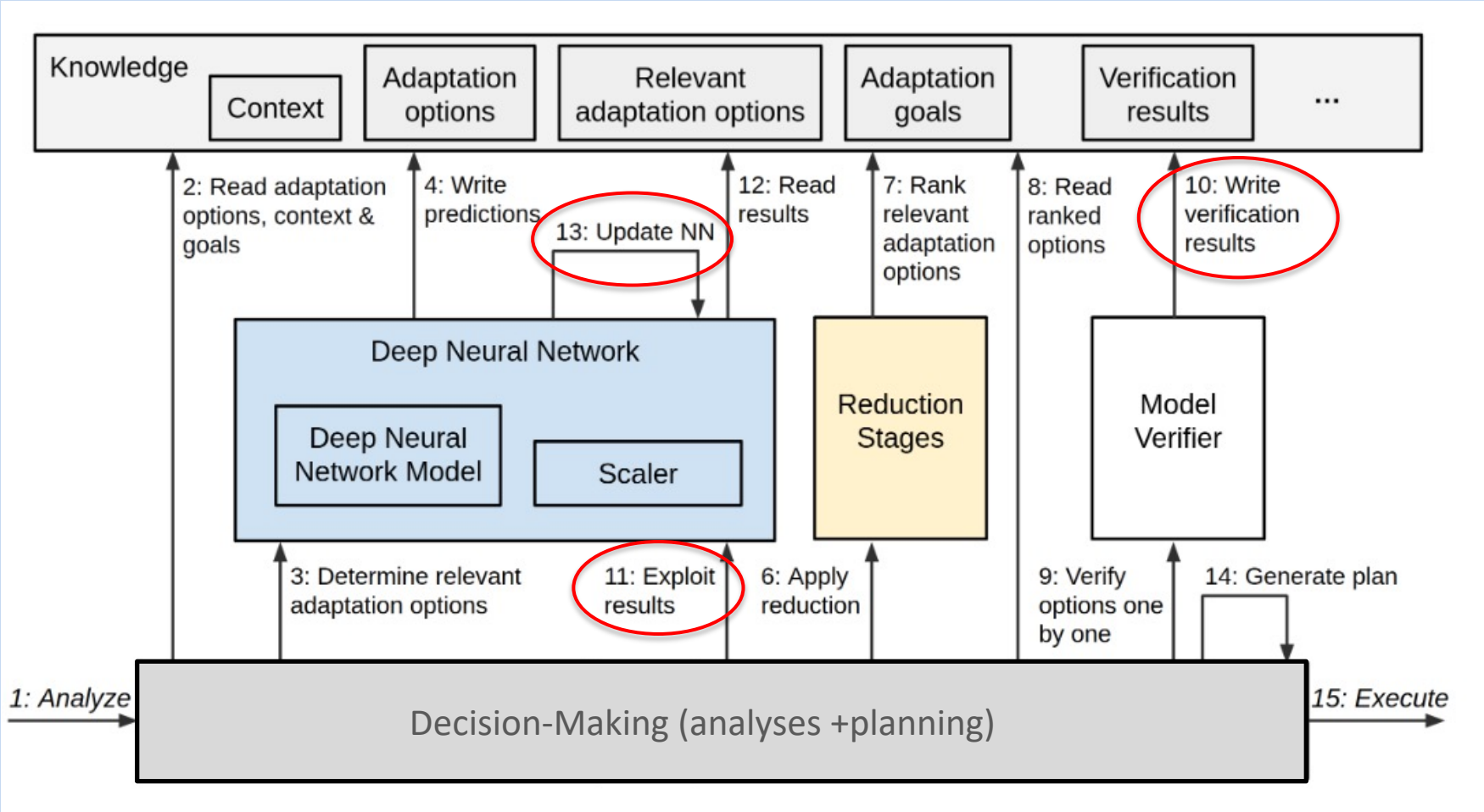

Threshold goals
Possible adaptation options
Current configuration
Latency
Packet loss
0
2
4
6
8
10
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

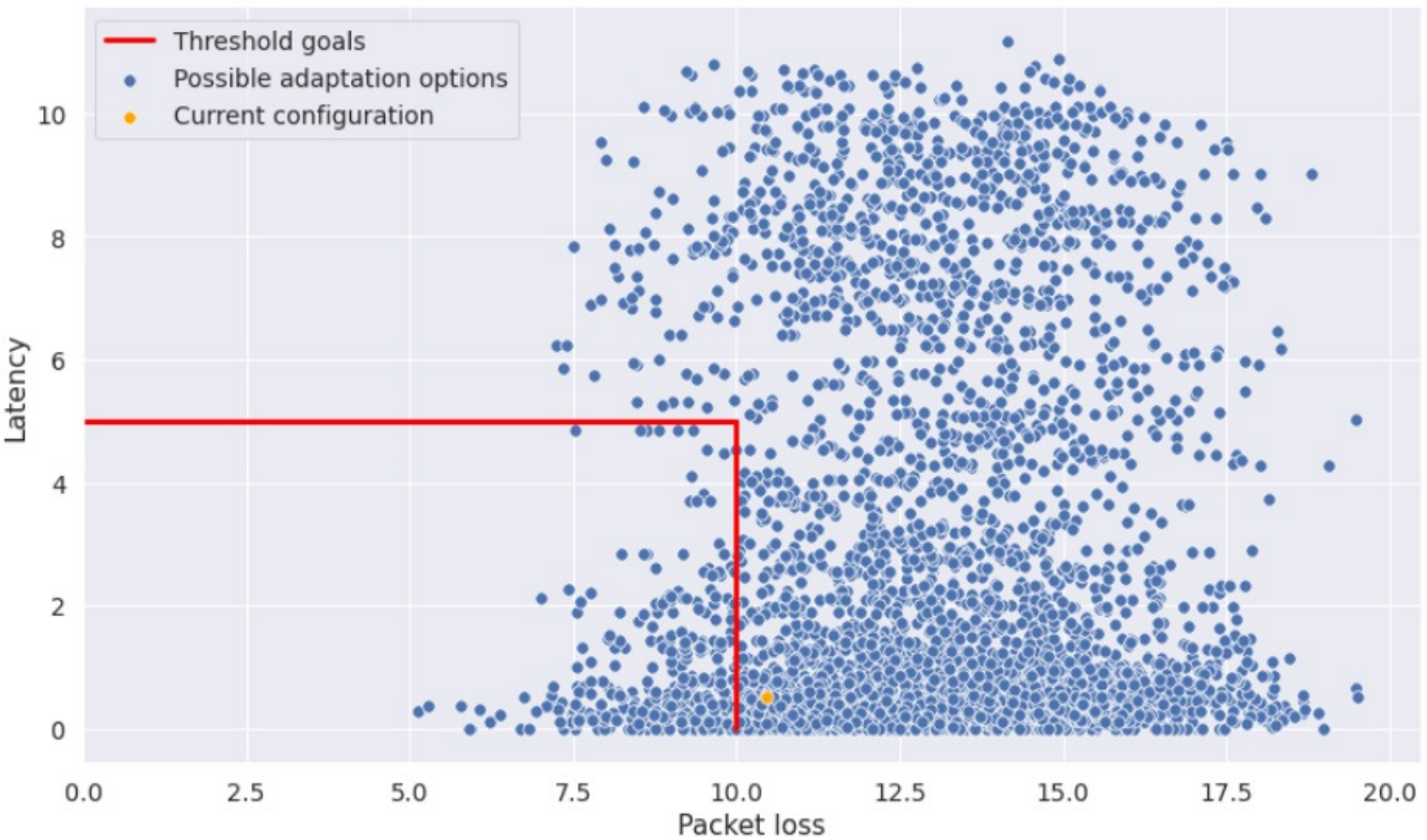

Latency
cycle
0
1
2
3
4
5
6
7
0
10
20
30
40
50


Deep Neural Network
Initial adaptation space
Context data
CORE LAYERS
T
S
O
Goal-specific heads
REDUCTION STAGES
Reduced adaptation space

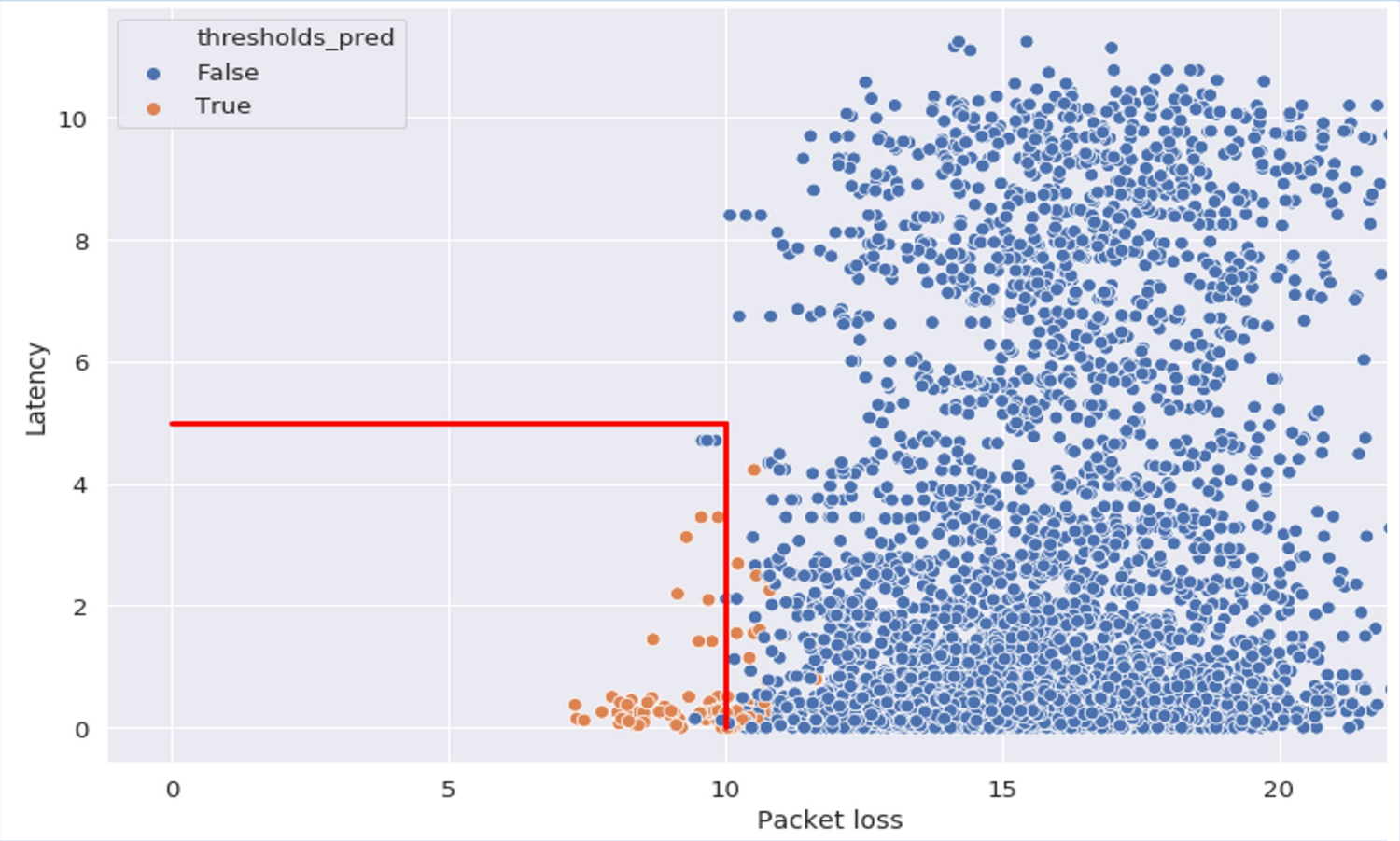

Deep Neural Network
Initial adaptation space
Context data
CORE LAYERS
T
S
O
Goal-specific heads
REDUCTION STAGES
Reduced adaptation space
Stage 1: threshold & set-point goals
Stage 2: optimization goal
max quality
min quality


Knowledge
Context
Adaptation options
Relevant adaptation options
Adaptation goals
Verification results
...
2: Read adaptation options, context & goals
4: Write predictions
13: Update NN
12: Read results
7: Rank relevant adaptation options
8: Read ranked options
10: Write verification results
Deep Neural Network
Deep Neural Network Model
Scaler
Reduction Stages
Model Verifier
3: Determine relevant adaptation options
11: Exploit results
6: Apply reduction
9: Verify options one by one
14: Generate plan
1: Analyze
Decision-Making (analyses +planning)
15: Execute


Knowledge
Context
Adaptation options
Relevant adaptation options
Adaptation goals
Verification results
...
2: Read adaptation options, context & goals
4: Write predictions
13: Update NN
12: Read results
7: Rank relevant adaptation options
8: Read ranked options
10: Write verification results
Deep Neural Network
Deep Neural Network Model
Scaler
Reduction Stages
Model Verifier
3: Determine relevant adaptation options
11: Exploit results
6: Apply reduction
9: Verify options one by one
14: Generate plan
1: Analyze
Decision-Making (analyses +planning)
15: Execute


Threshold goals
Possible adaptation options
Current configuration
Latency
Packet loss
0
2
4
6
8
10
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

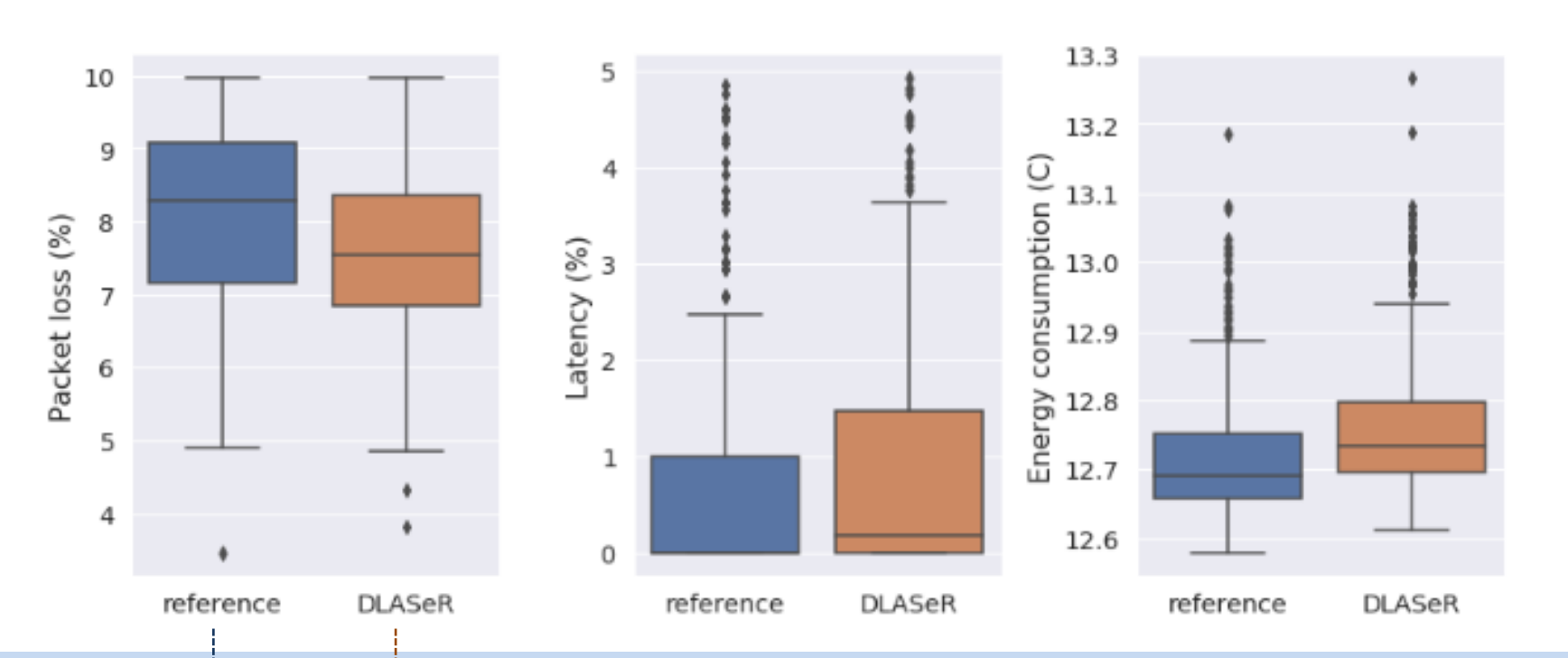
thresholds_pred
False
True
Latency
Packet loss
0
2
4
6
8
10
0
5
10
15
20


thresholds_pred
False
True
Latency
Packet loss
0
2
4
6
8
10
0
5
10
15
20

packet loss < 10%

latency < 5%

min energy consumption



analysis after adaptation
space reduction using learning

exhaustive analysis

Key insights Wave VII

Machine learning enables dealing with large amounts of data and complex levels of uncertainty

Learning can be used to support different functions of a feedback loop

Learning is increasingly being applied in self-adaptation

Overview

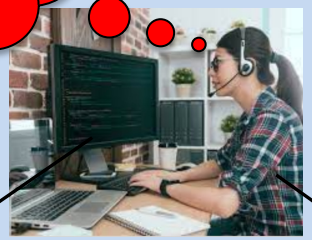
- Why self-adaptation?
- What is self-adaptation?
- Self-adaptation in industry
- How to engineer self-adaptive systems?
- Decision-making with formal techniques at runtime
- Why is “just self-adaptation” not enough?
- Conclusions

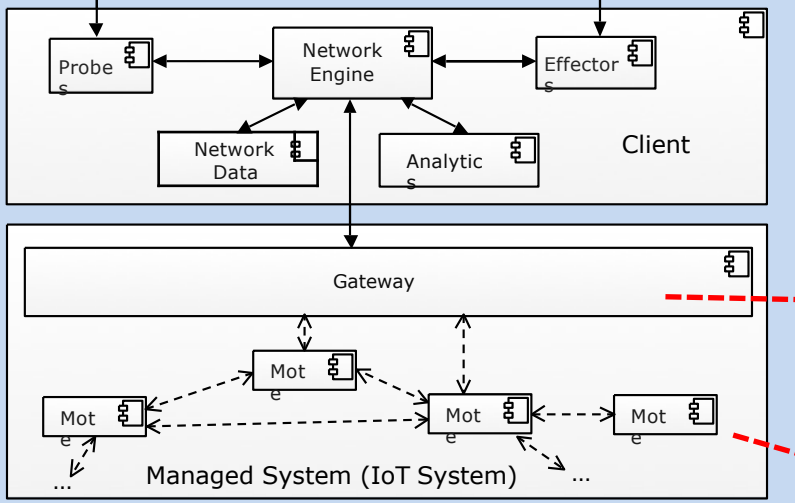
Decision-making with formal techniques at runtime

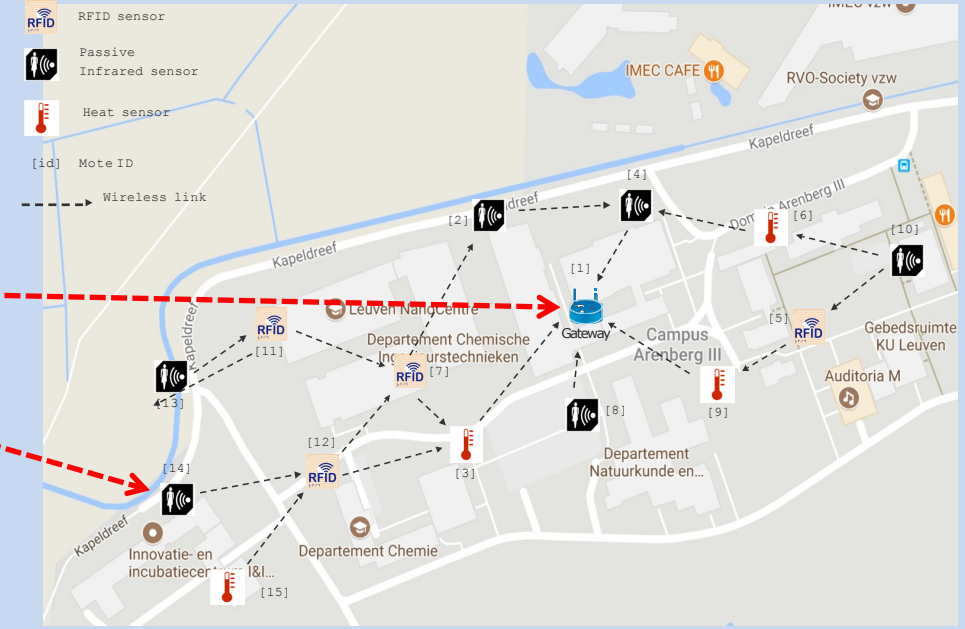
Self-adaptive IoT system

Self-adaptive IoT system

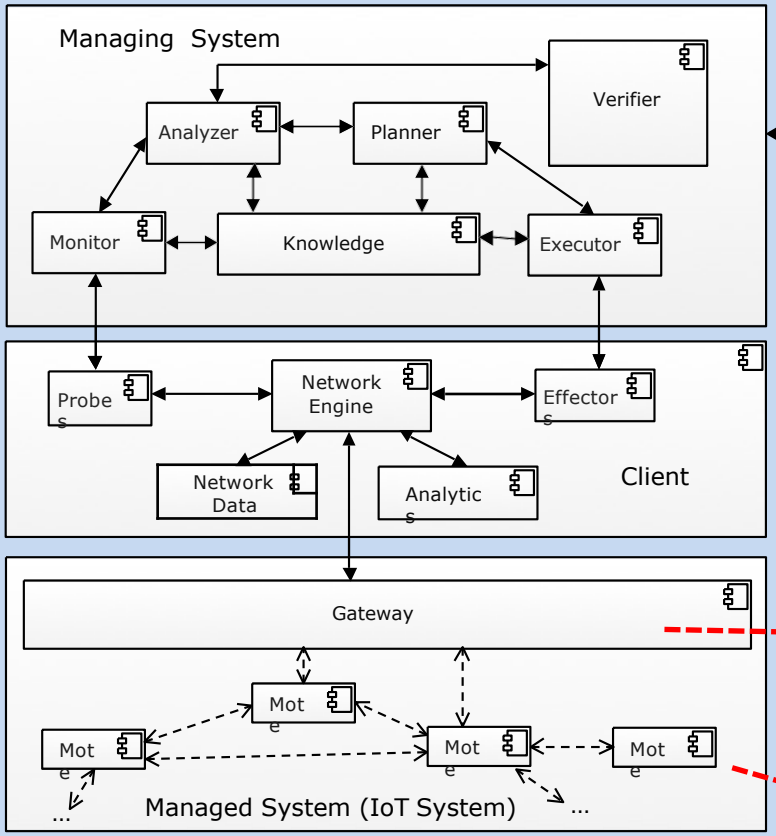
Which configuration?
to use?
- packet loss < 10%
- min energy



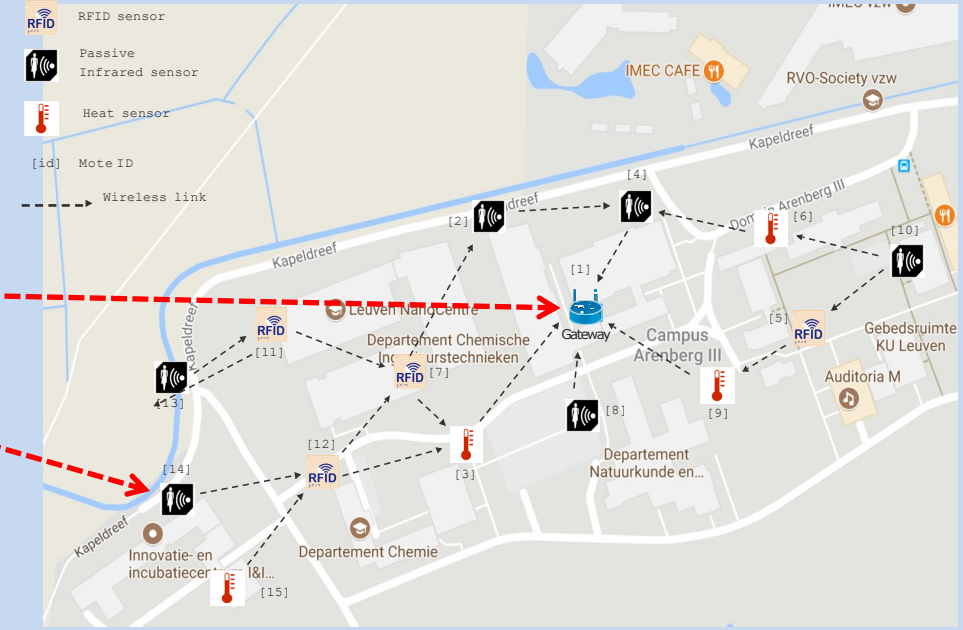




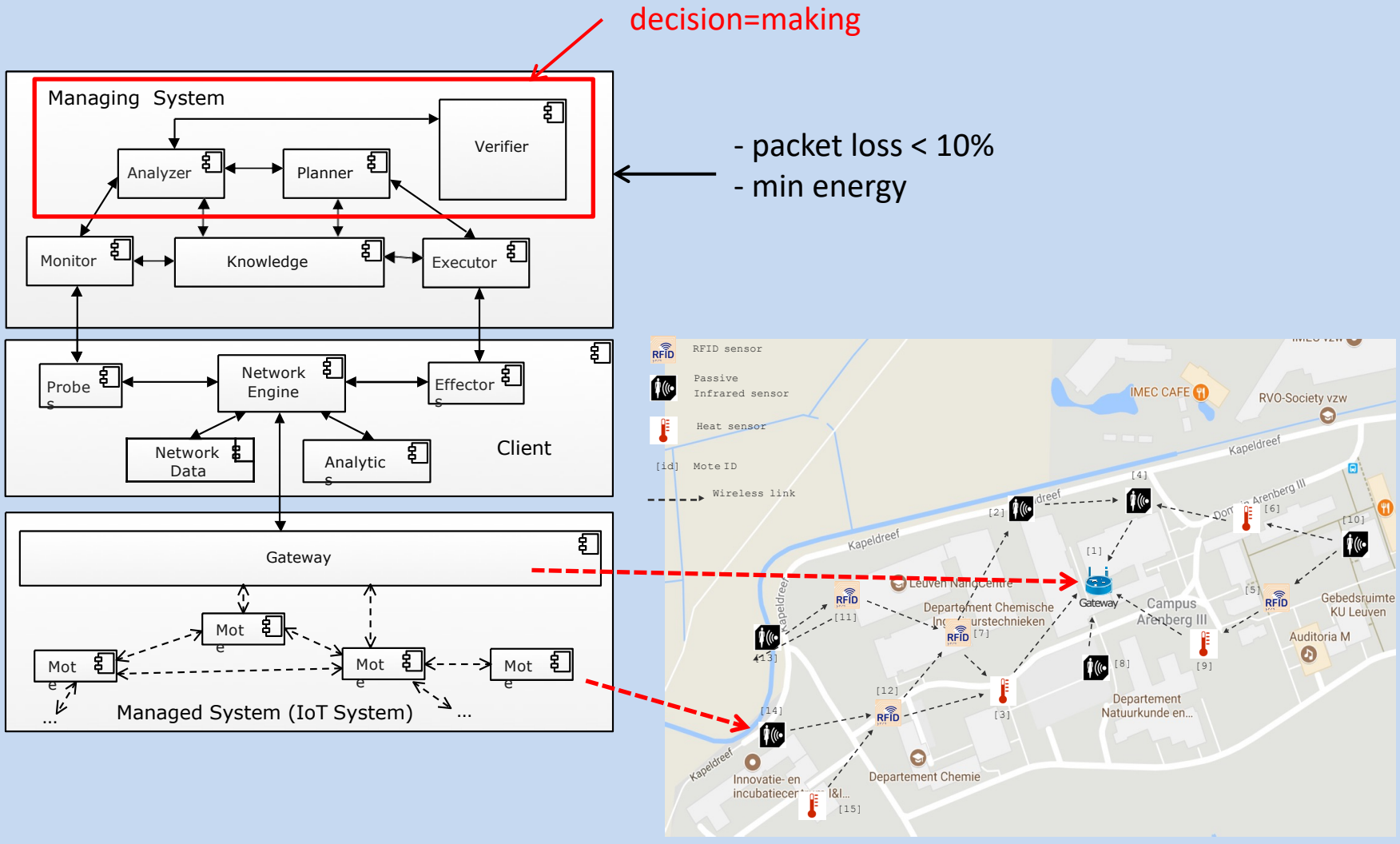
Self-adaptive IoT system



- packet loss < 10%
- min energy

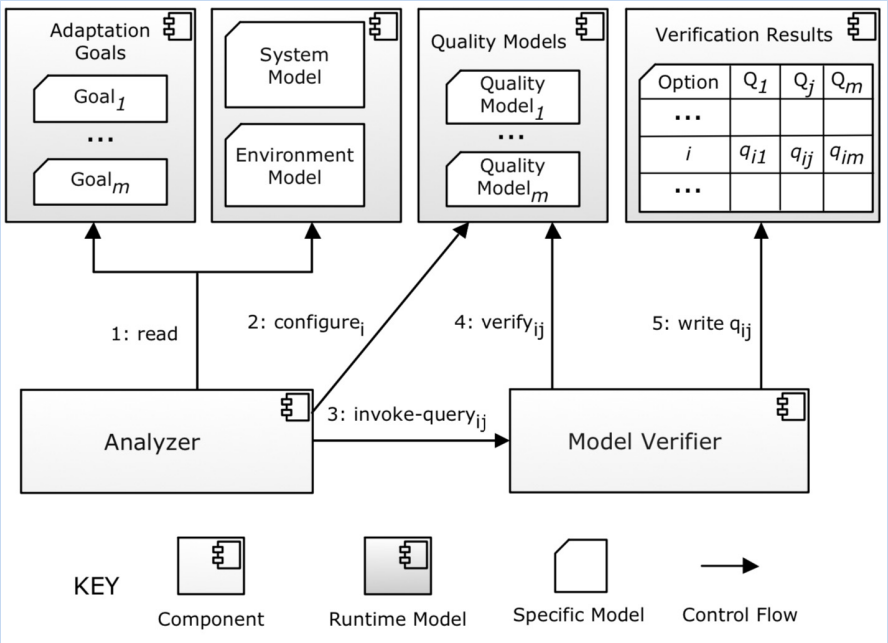


Self-adaptive IoT system

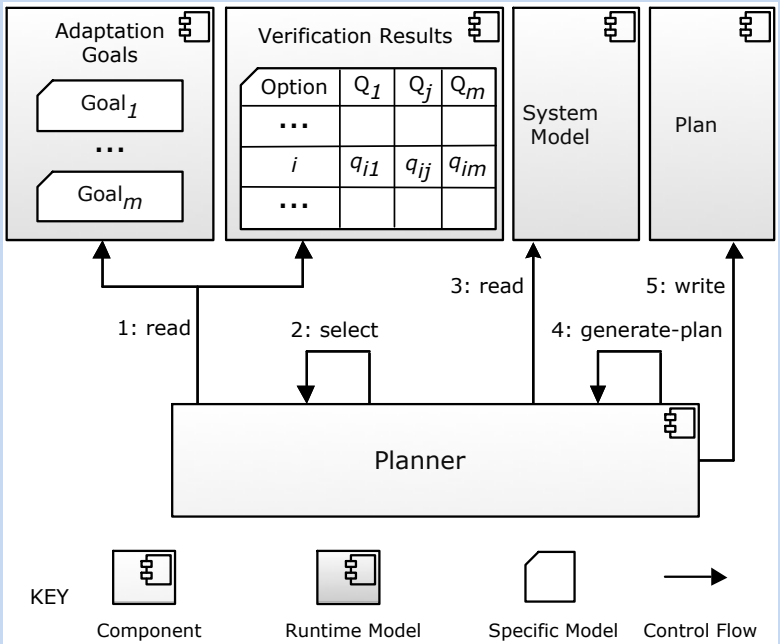


Decision-making in self-adaptive systems using formal techniques at runtime

I. Analysis of adaptation options

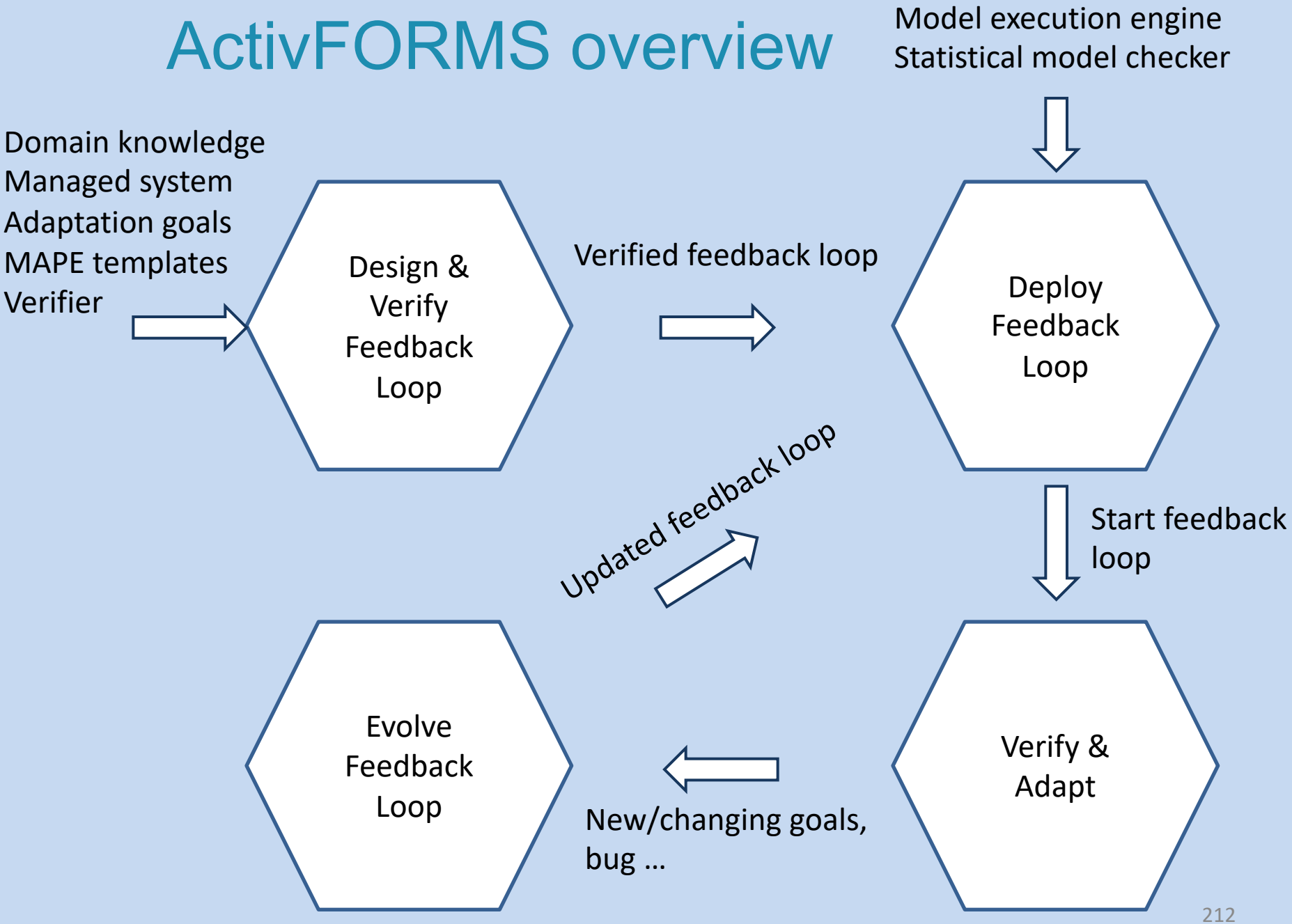


II. Selection of the best adaptation option

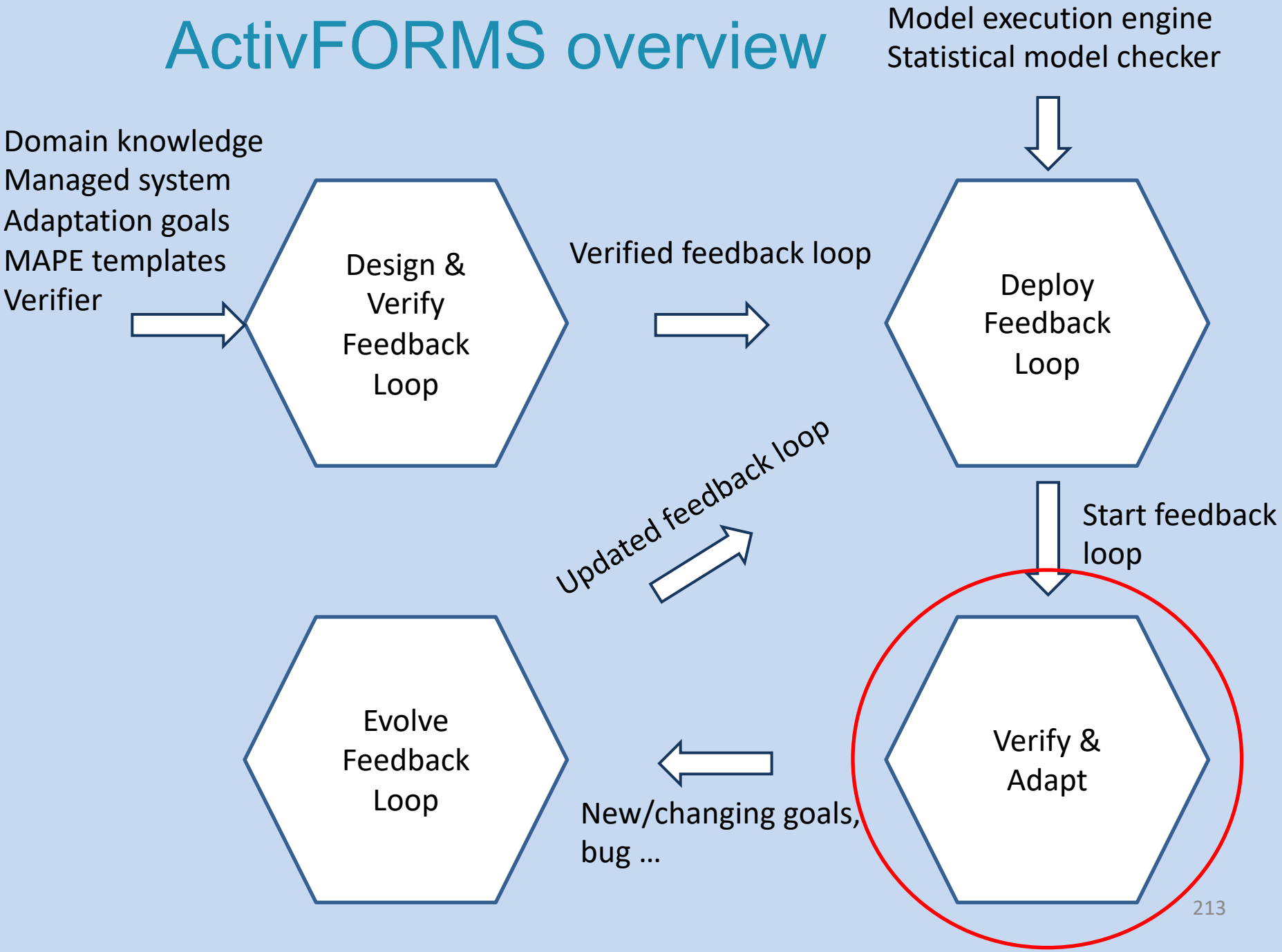


ActivFORMS: executable formal models with statistical model checking at runtime

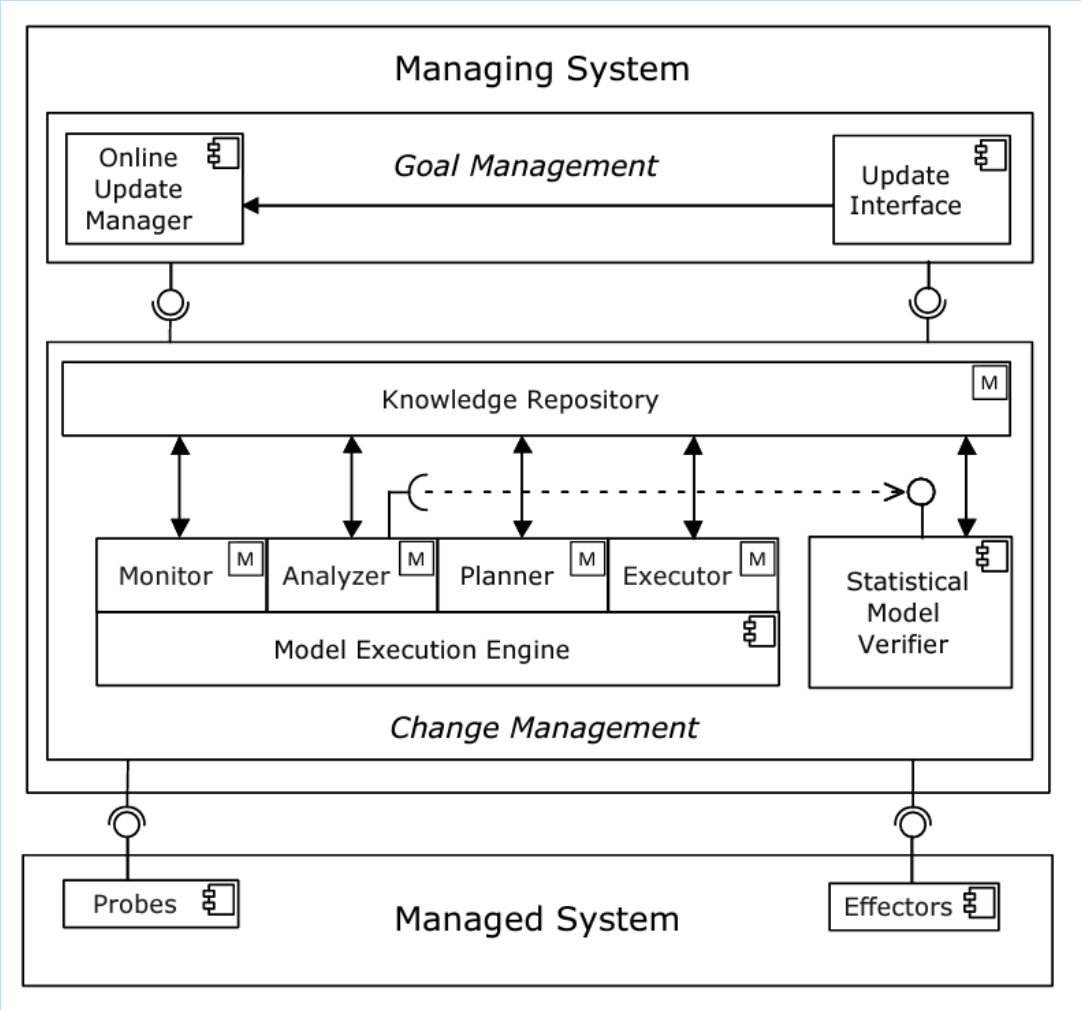
ActivFORMS overview



ActivFORMS overview

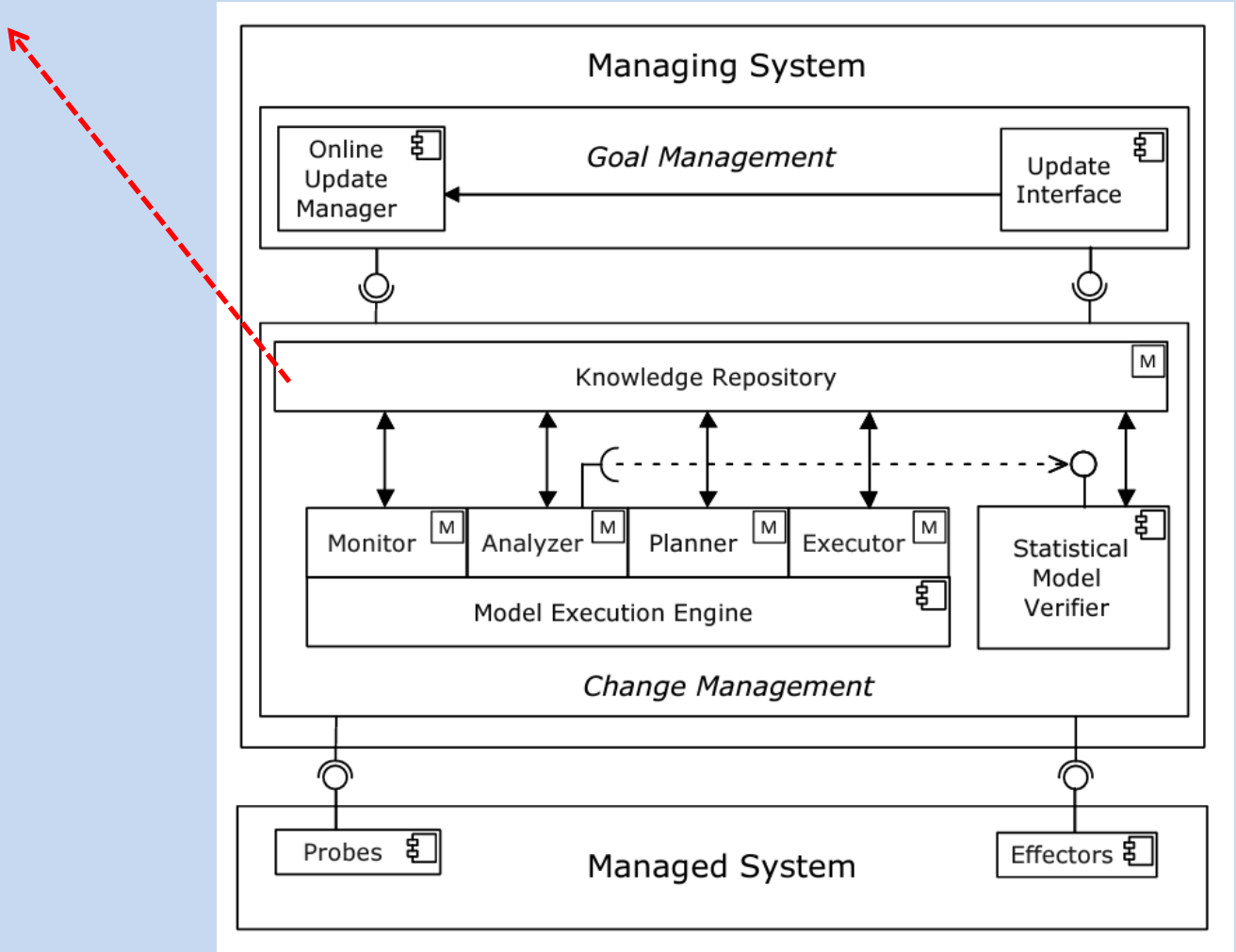


ActivFORMS overview



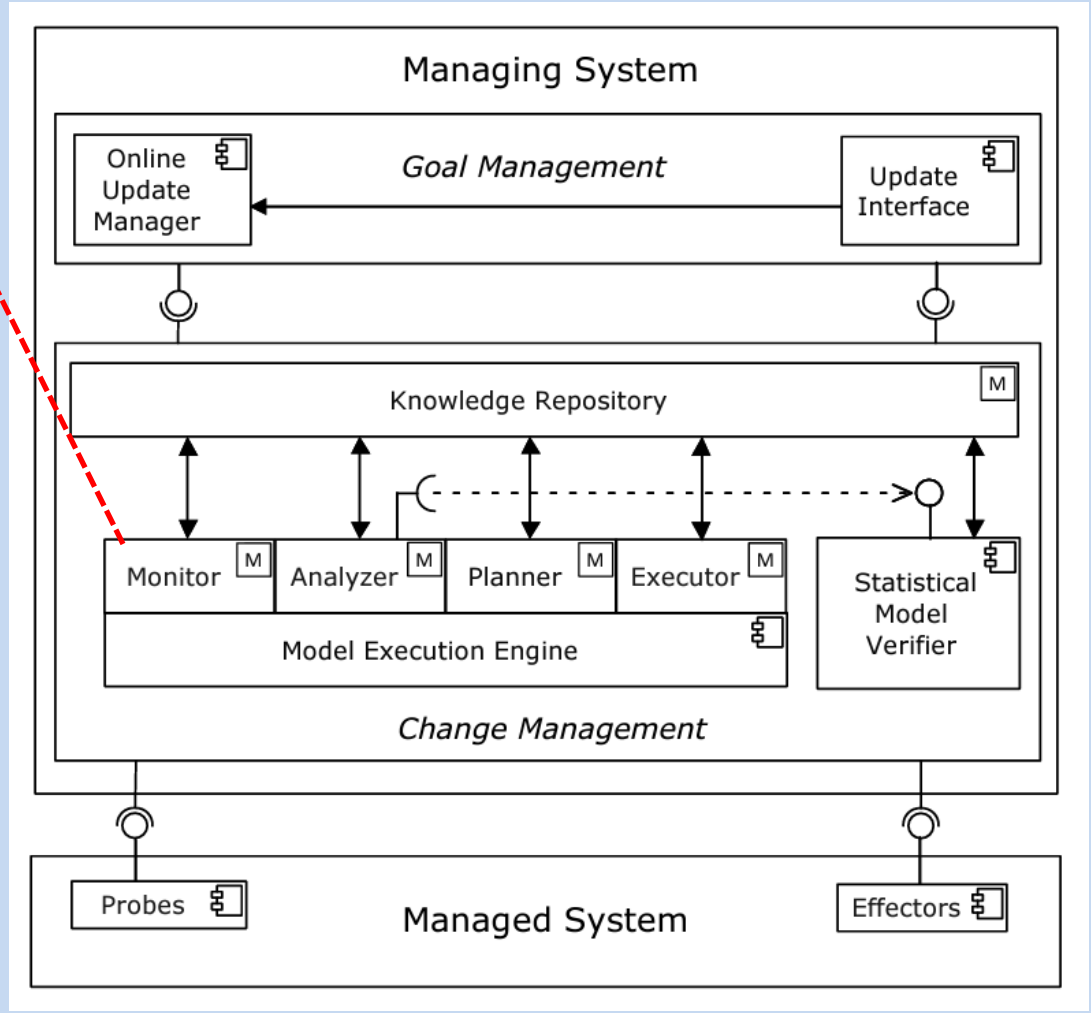
ActivFORMS overview

- Stochastic timed automata, one per quality property of adaptation goal
  Parameters for setting uncertainties and adaptation options
- Adaptation goals as first-class citizens



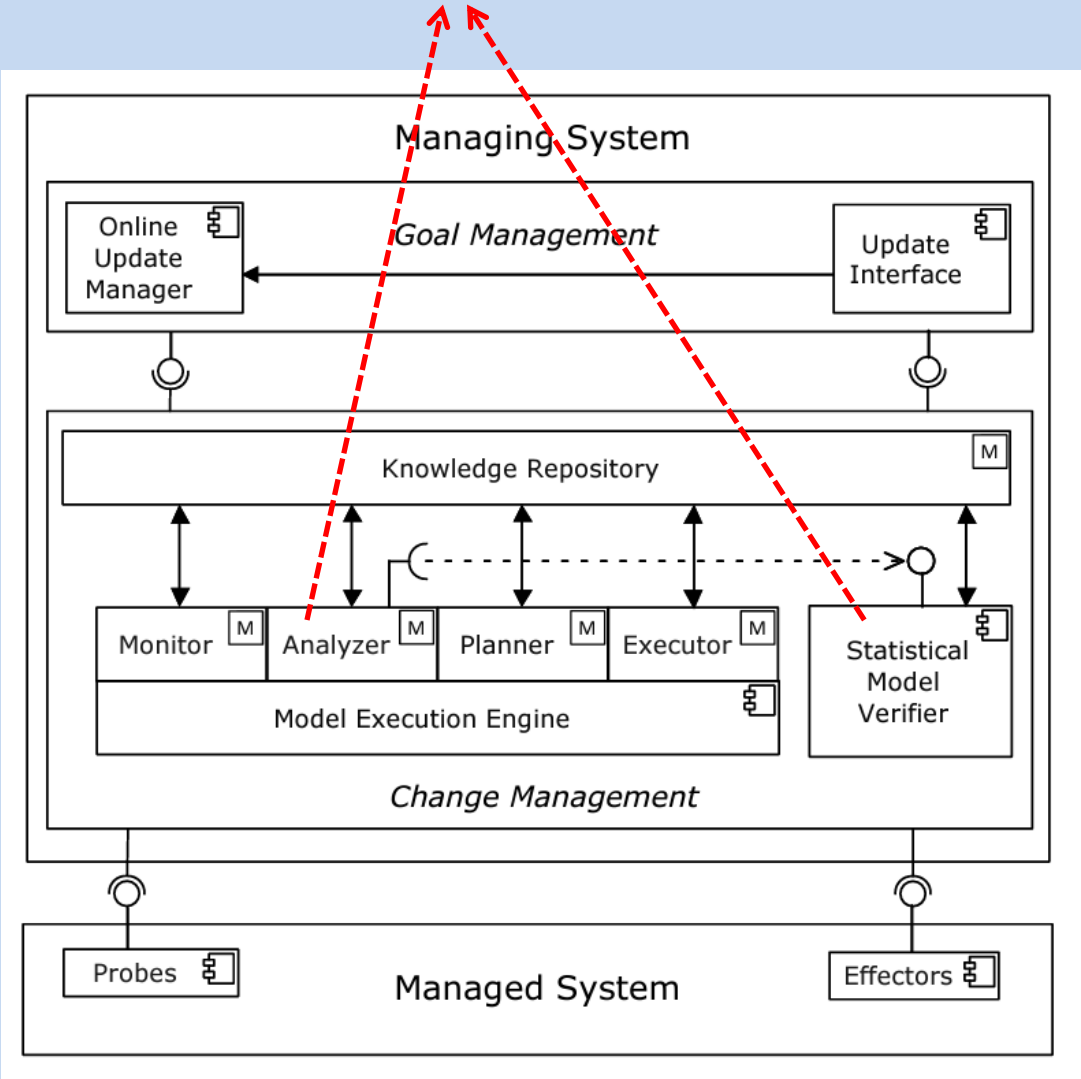
ActivFORMS overview

- Tracks system properties and uncertainties through probes
- Updates knowledge accordingly



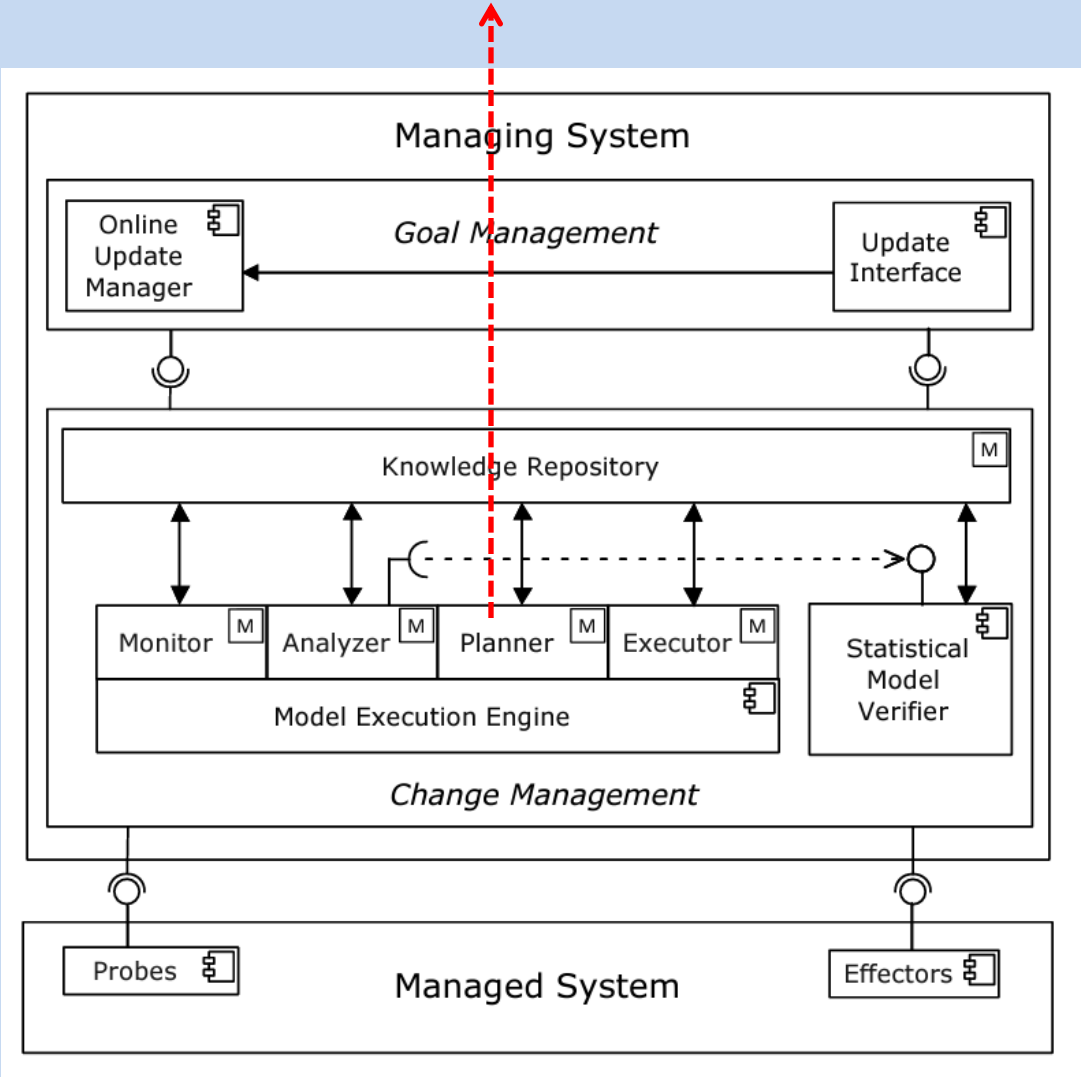
ActivFORMS overview

- Configures quality models for each adaptation option
- Exploits statistical model checker to determine expected qualities of each adaptation option



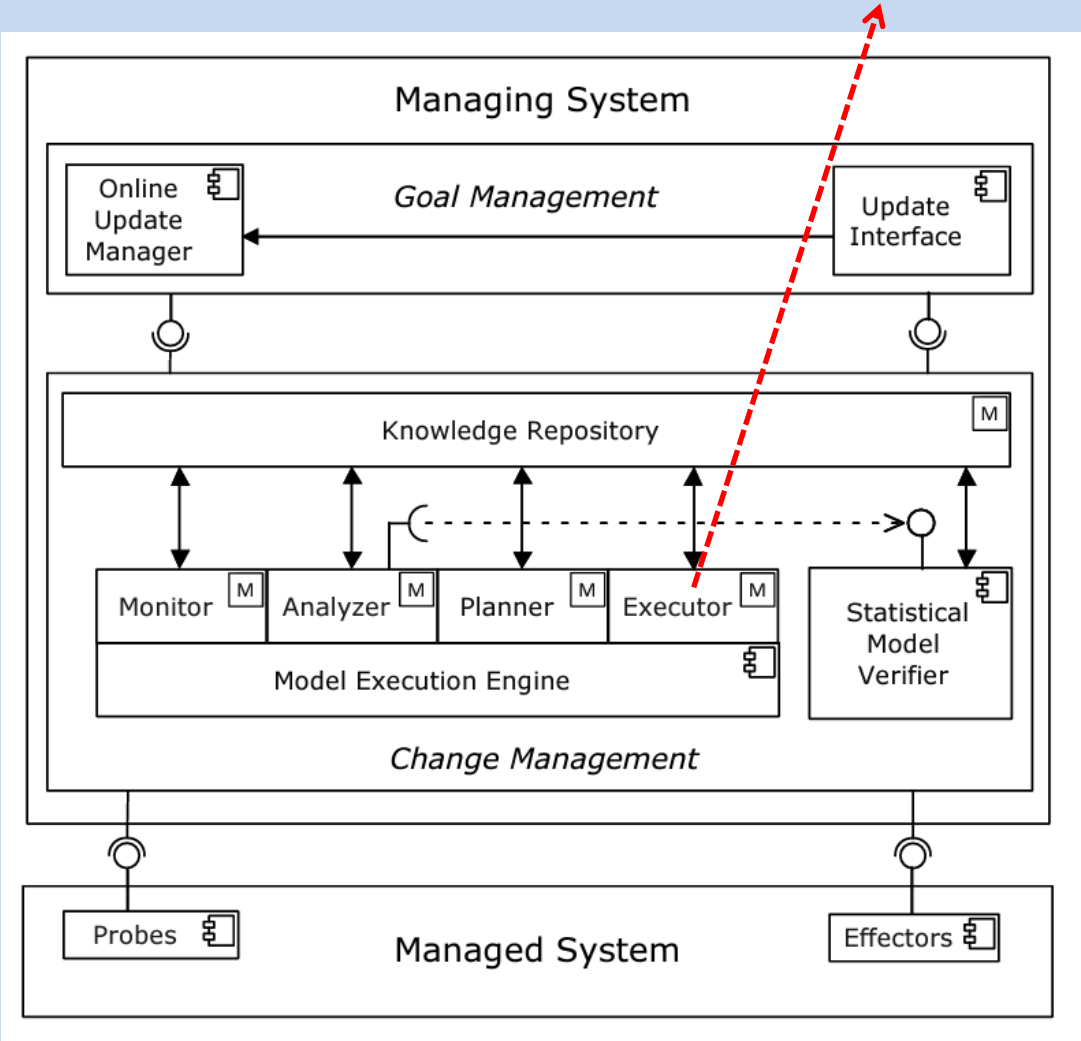
ActivFORMS overview

- Determines the best adaptation option based on analysis results and adaptation goals
- Creates a plan to adapt the system from current configuration to new configuration



ActivFORMS overview

- Executes the adaptation plan though effectors

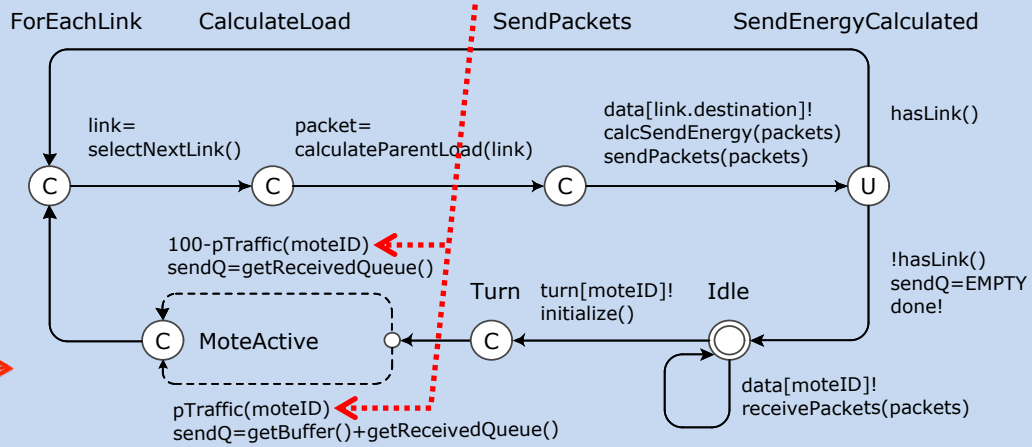


Uncertainties: traffic generated by motes and interference links

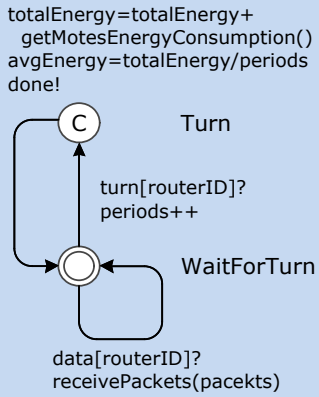
Environment, system, and quality models

Monitor

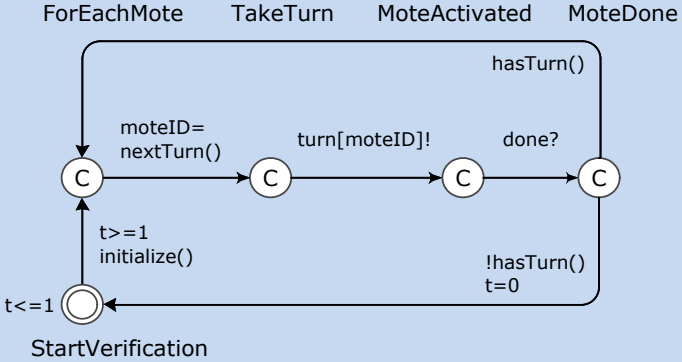
update state & uncertainties



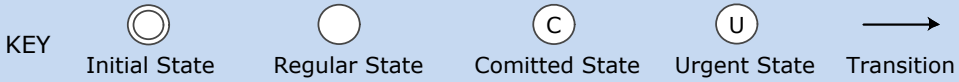
(a) Mote



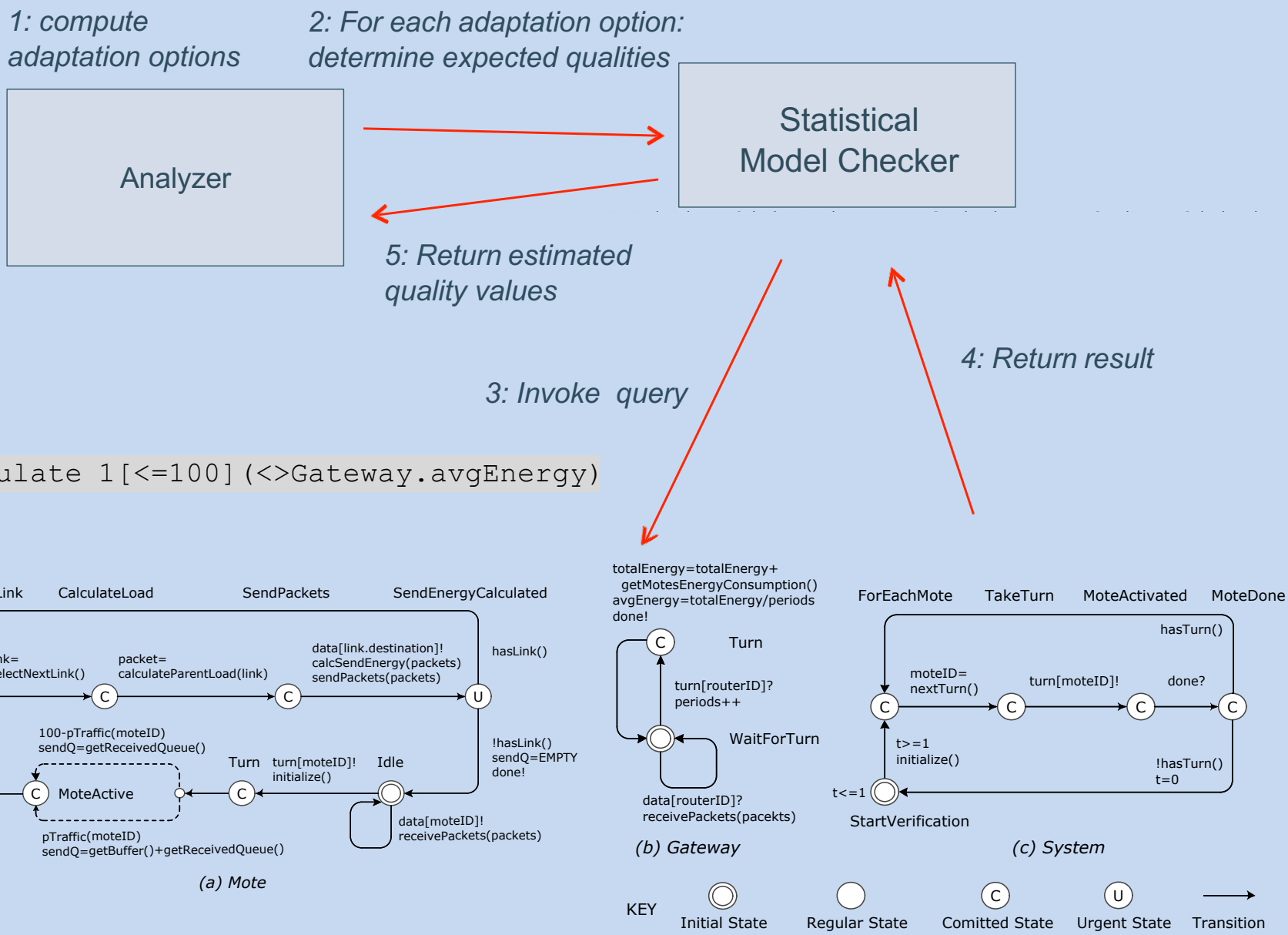
(b) Gateway



(c) System

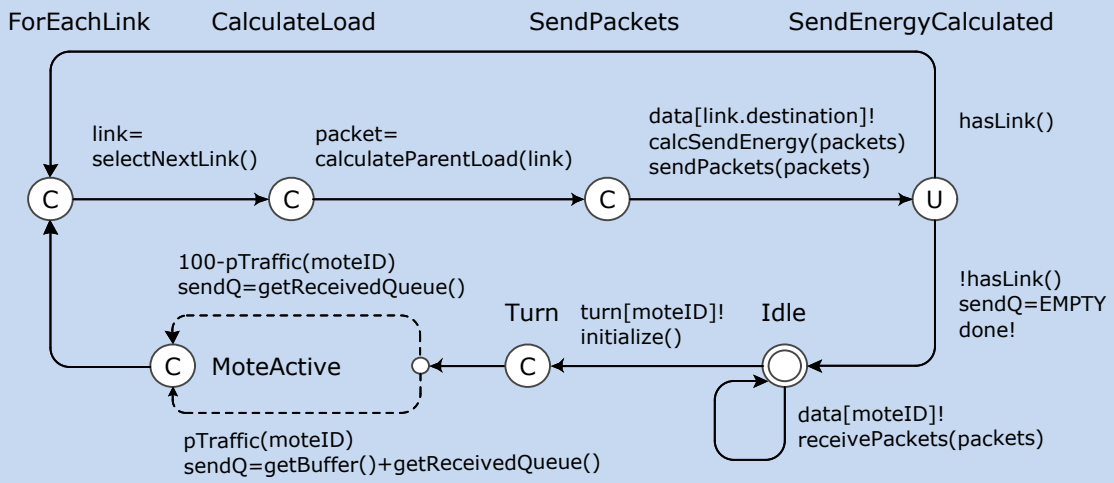


Track:
- *Traffic of nodes*
- *Signal to noise ratio*
- *Etc.*

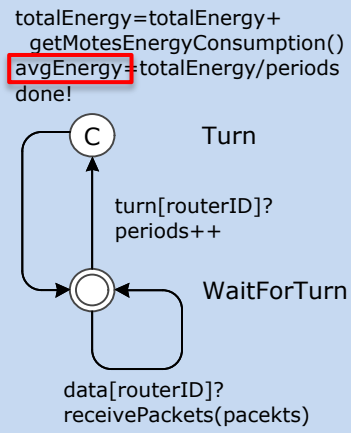


Number of simulation runs determines accuracy and confidence of result

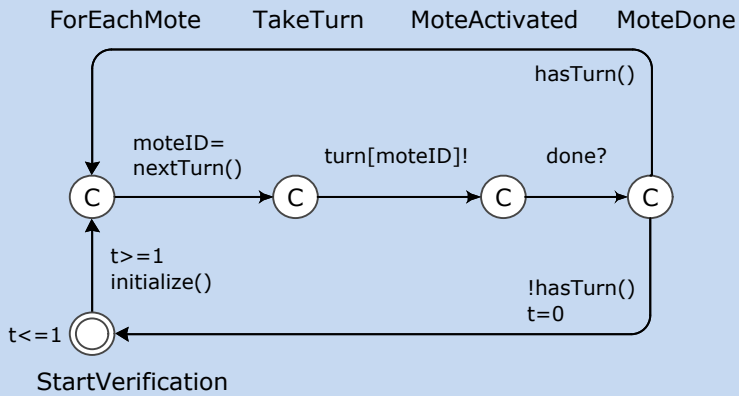
Uppaal verification query: simulate 1 [<=100] (<> Gateway.avgEnergy)



(a) Mote

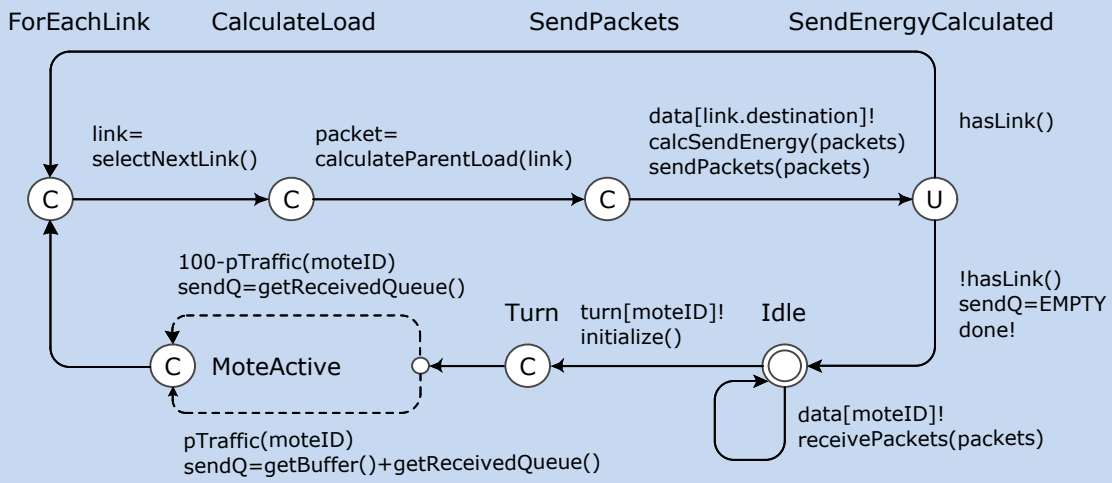


(b) Gateway

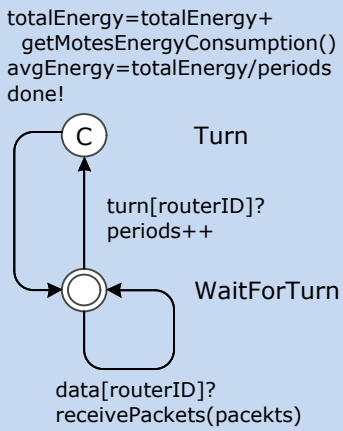


(c) System

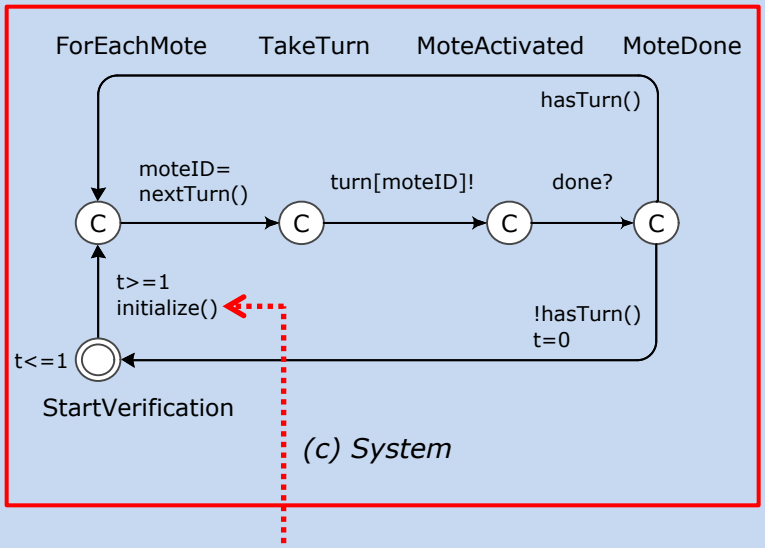
Uppaal verification query: simulate 1 [<=100] (<> Gateway.avgEnergy)



(a) Mote



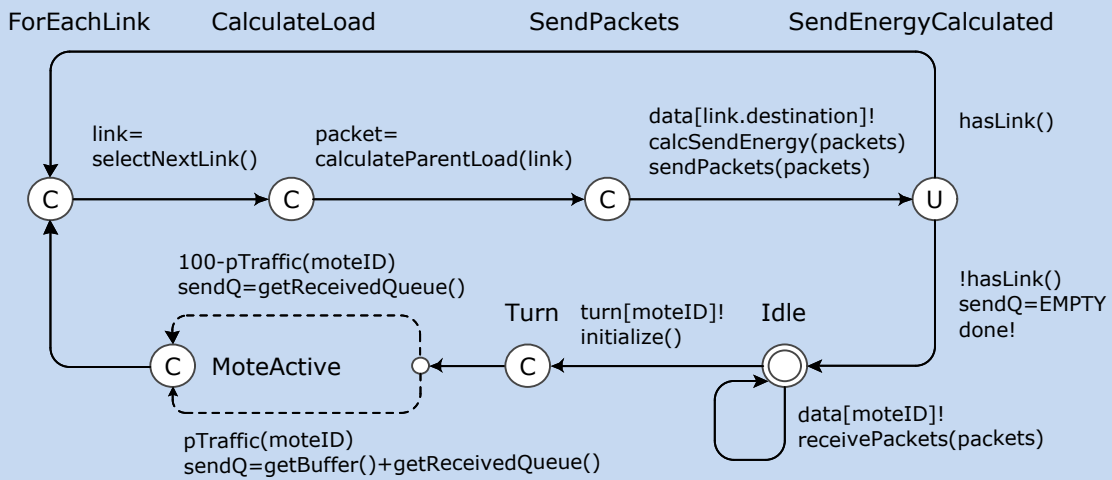
(b) Gateway



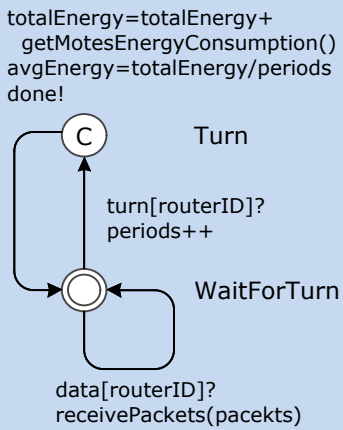
(c) System

Configure adaptation option: power settings motes and % messages sent along links

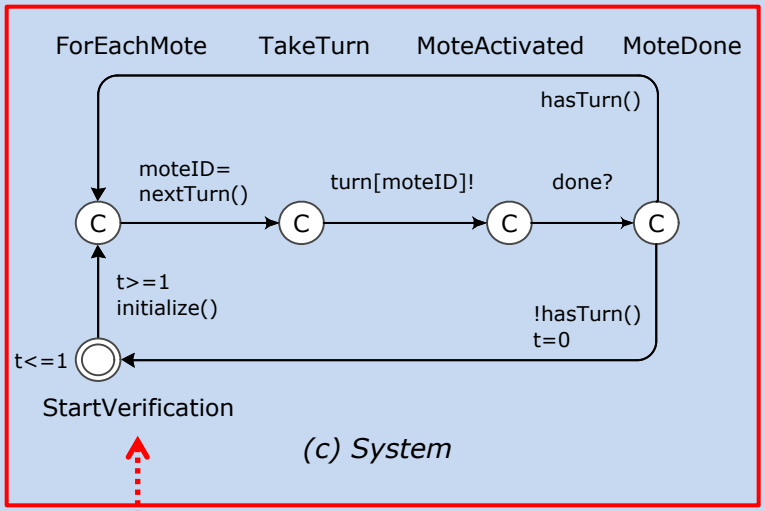
Uppaal verification query: simulate 1 [<=100] (<> Gateway.avgEnergy)



(a) Mote



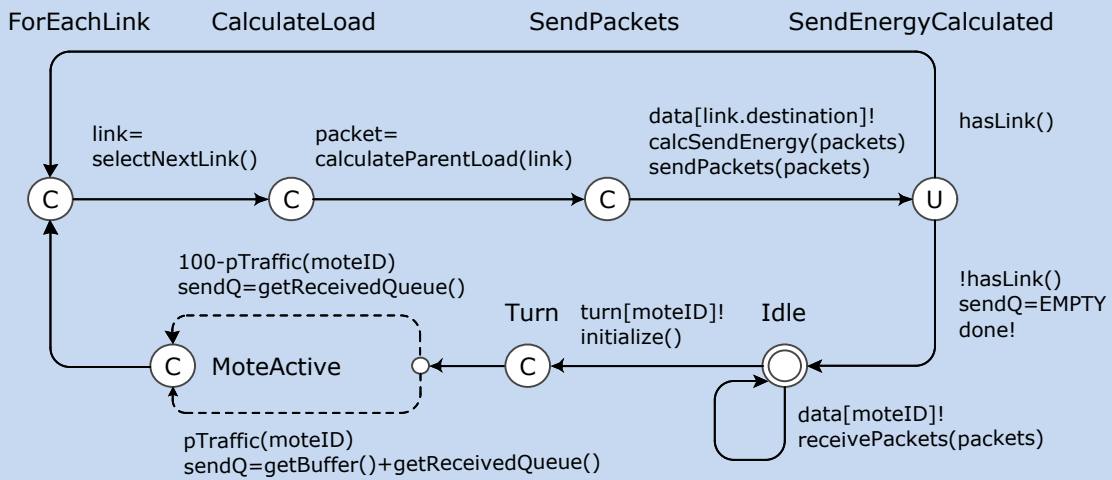
(b) Gateway



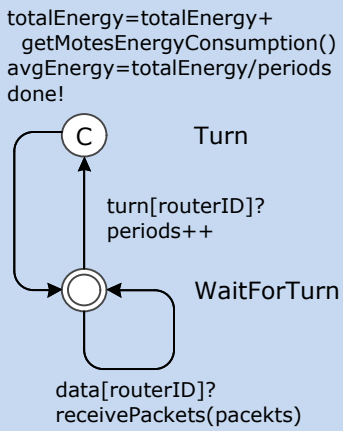
(c) System

Start simulation for adaptation option

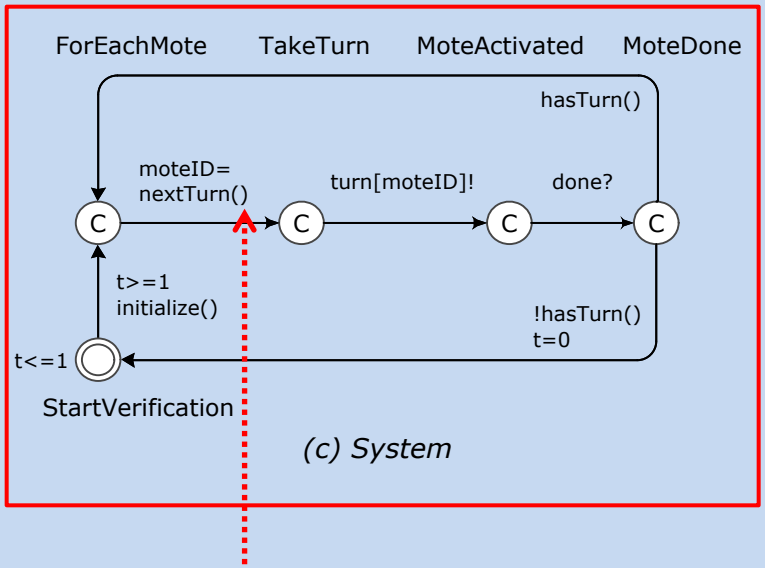
Uppaal verification query: simulate 1 [<=100] (<> Gateway.avgEnergy)



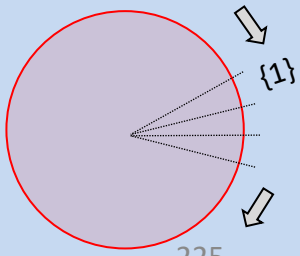
(a) Mote



(b) Gateway

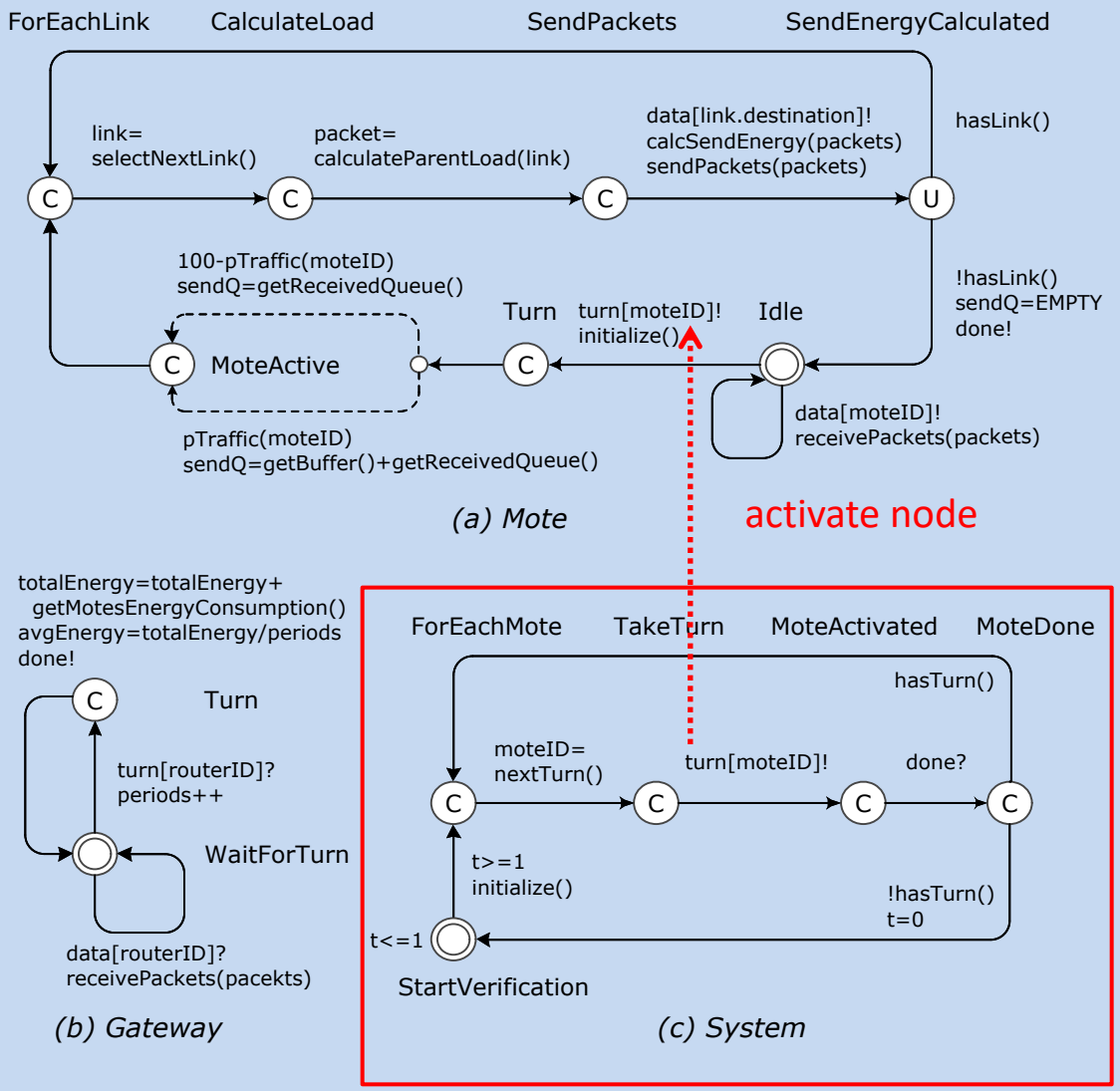


(c) System

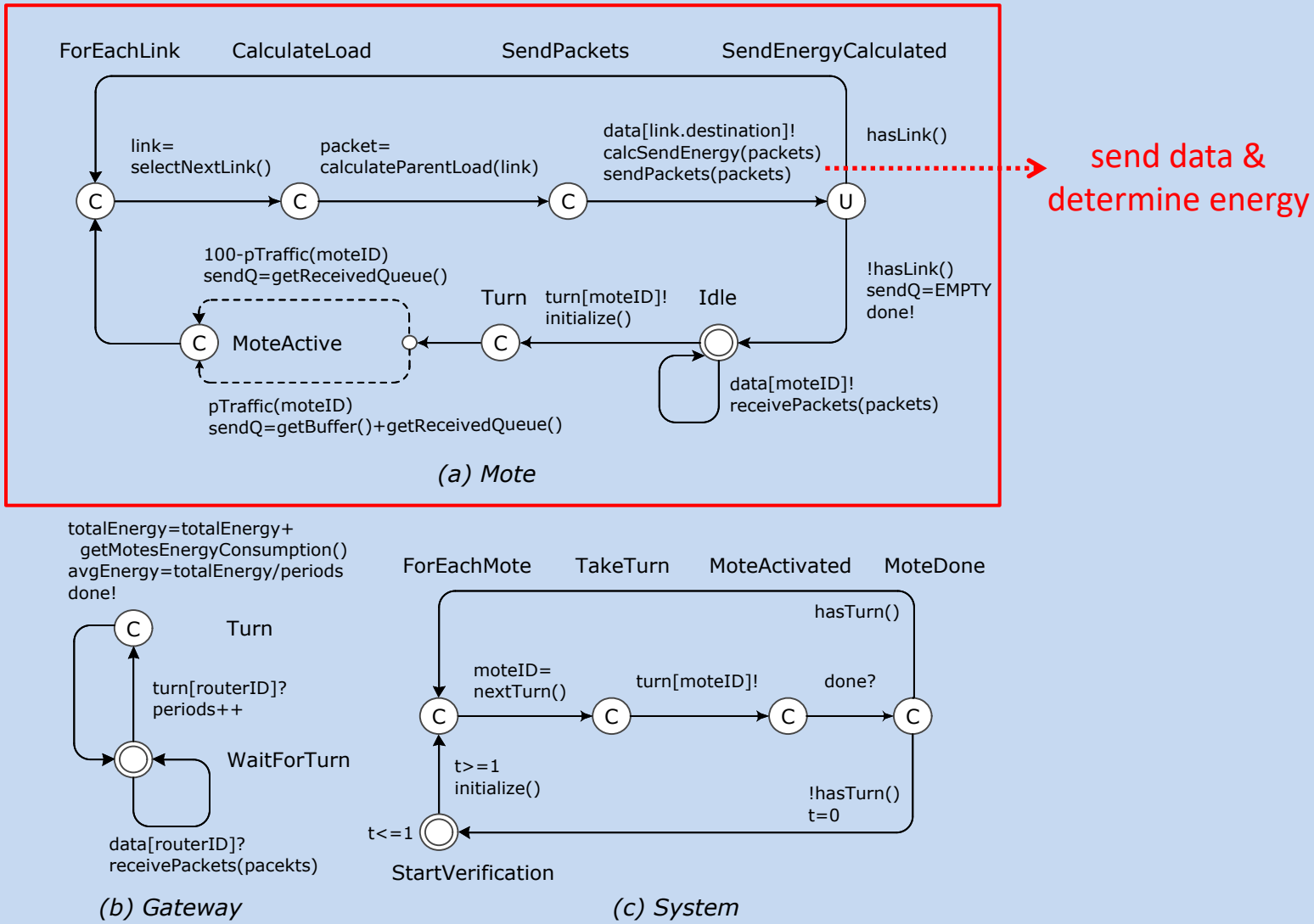


simulate a communication cycle, activating subsequent nodes

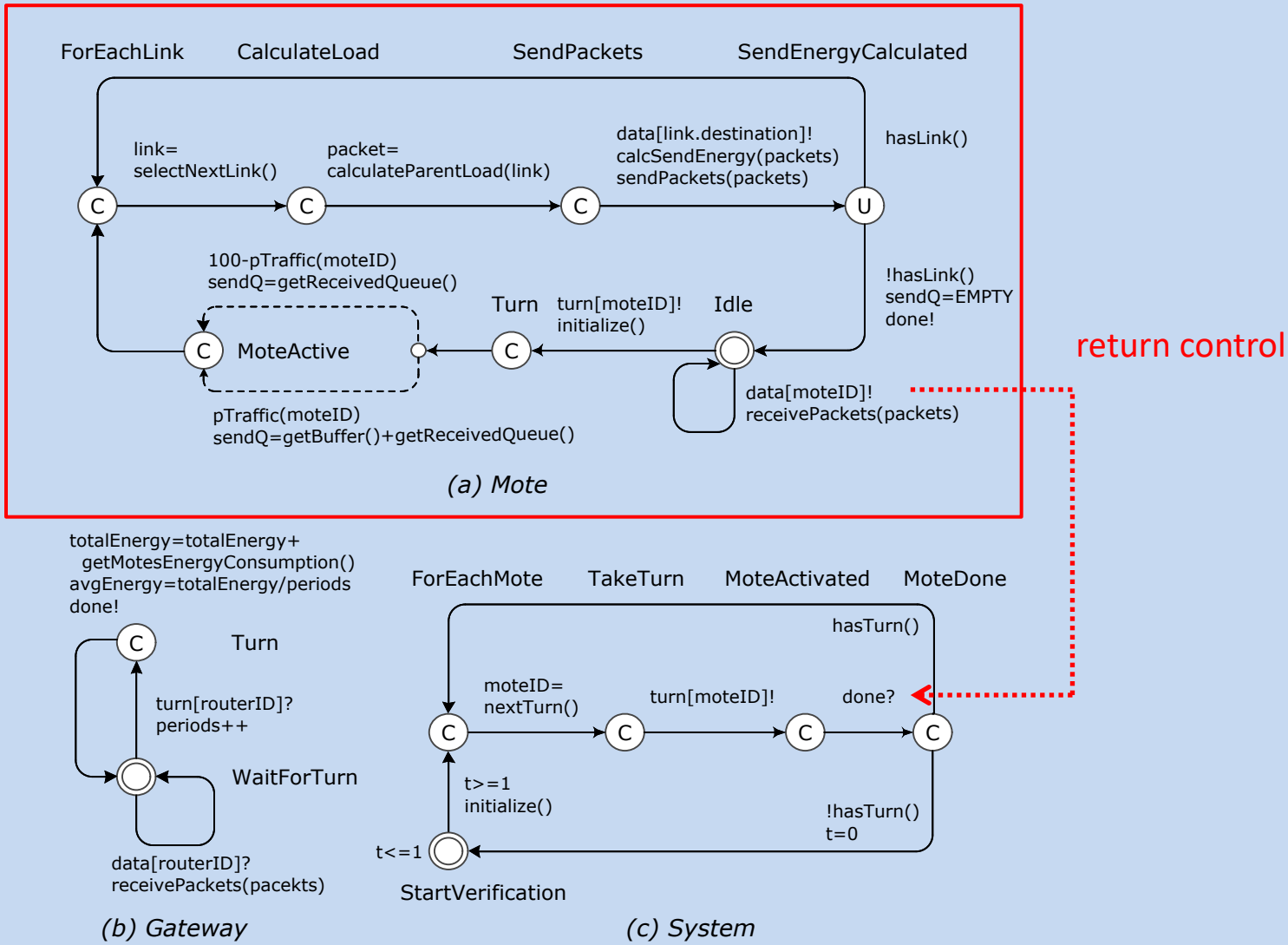
Uppaal verification query: simulate 1 [<=100] (<> Gateway.avgEnergy)



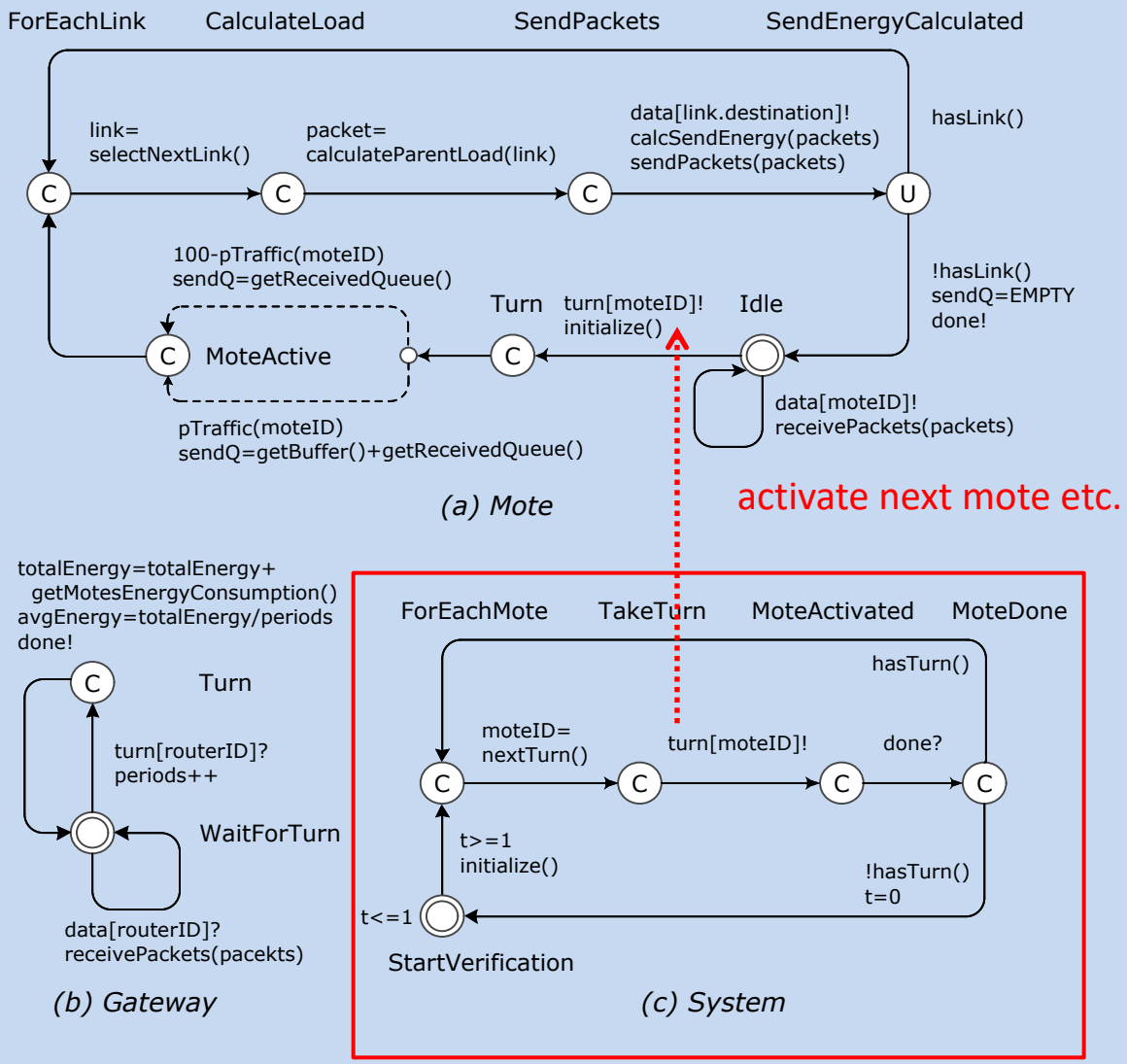
Uppaal verification query: simulate 1 [<=100] (<> Gateway.avgEnergy)



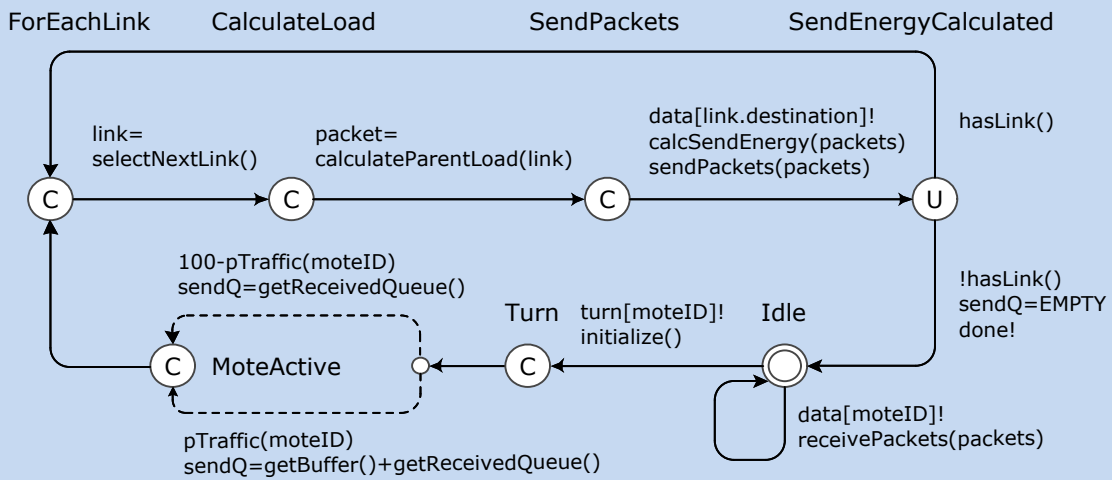
Uppaal verification query: simulate 1 [<=100] (<> Gateway.avgEnergy)



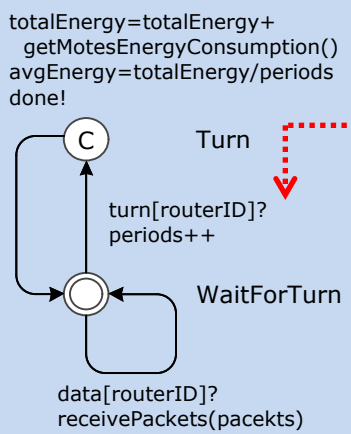
Uppaal verification query: simulate 1 [<=100] (<> Gateway.avgEnergy)



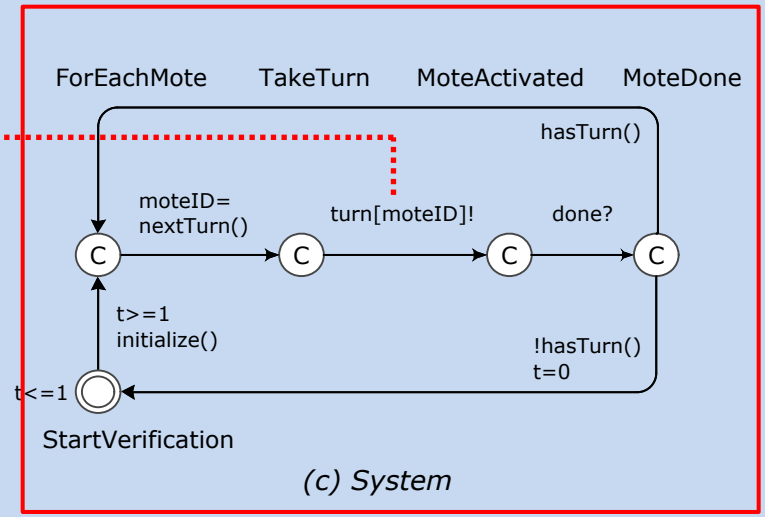
Uppaal verification query: simulate 1 [<=100] (<> Gateway.avgEnergy)



(a) Mote



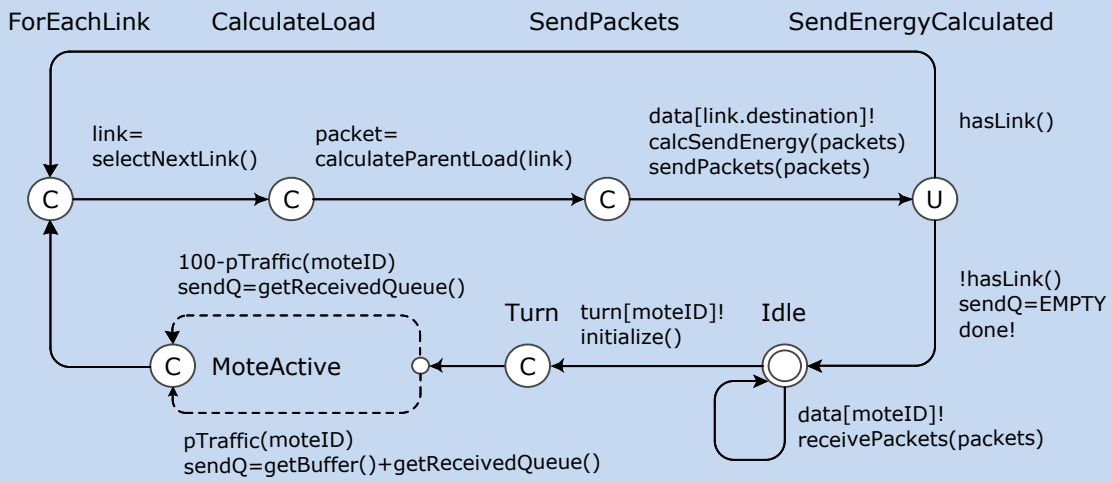
(b) Gateway



(c) System

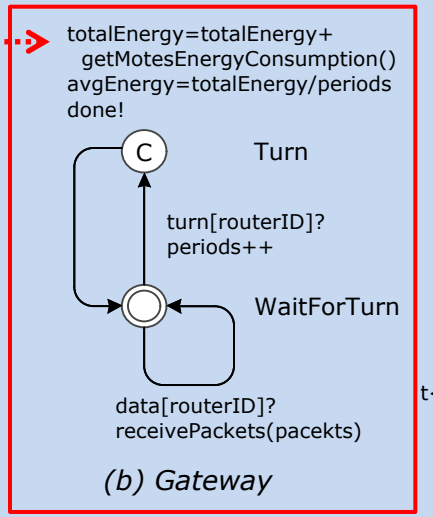
activate gateway

Uppaal verification query: simulate 1 [<=100] (<> Gateway.avgEnergy)

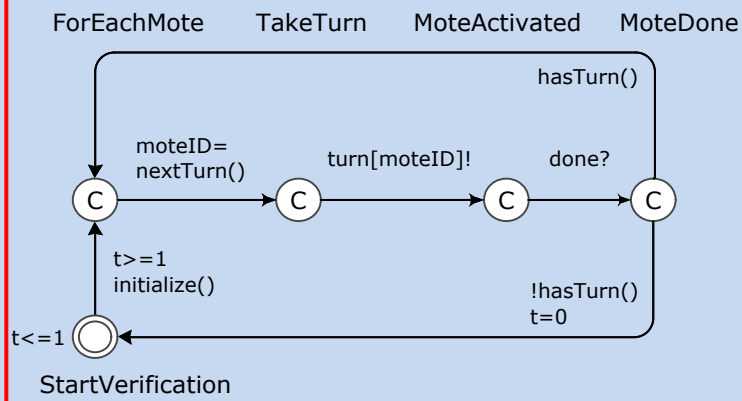


(a) Mote

determine energy

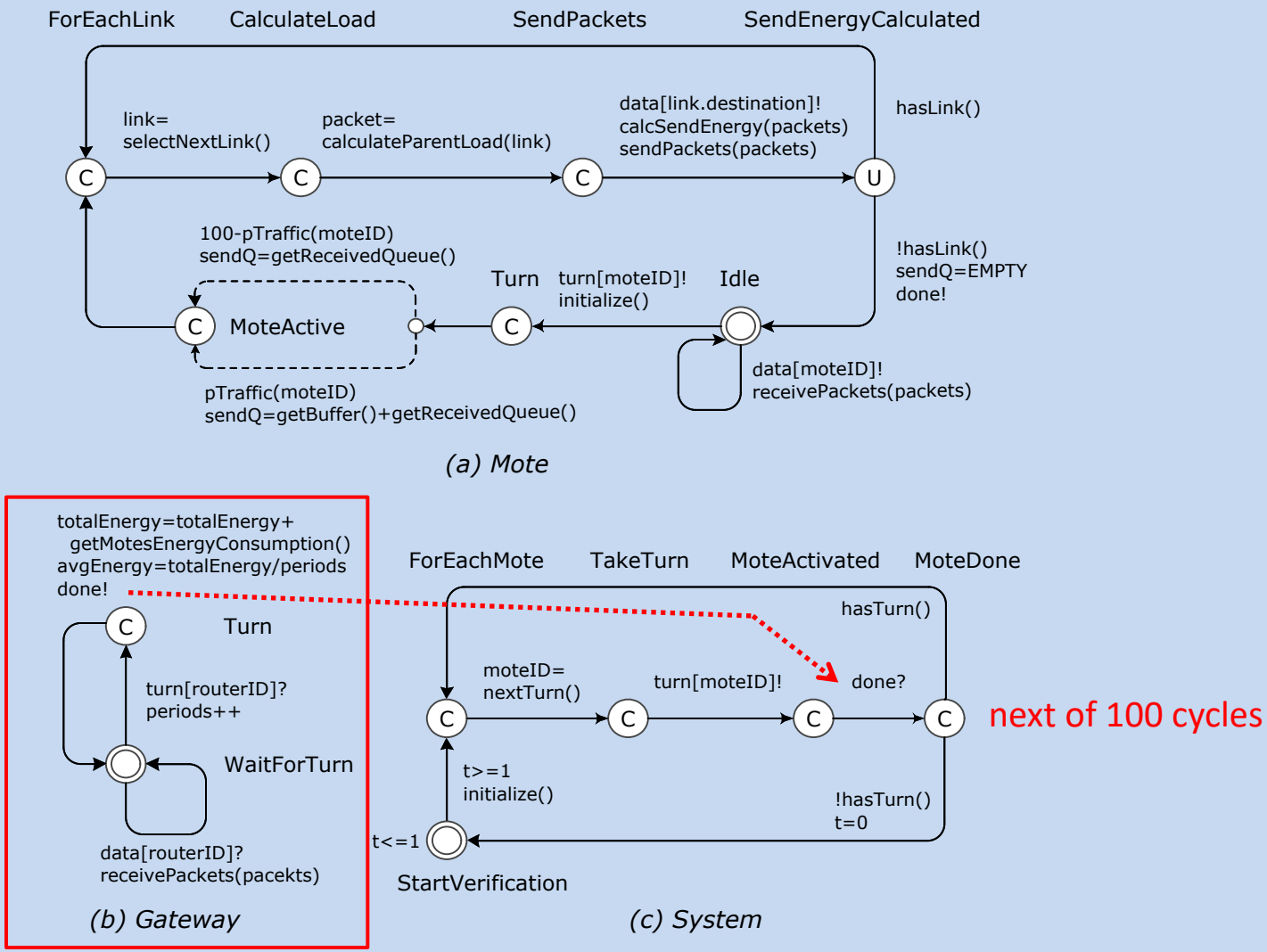


(b) Gateway

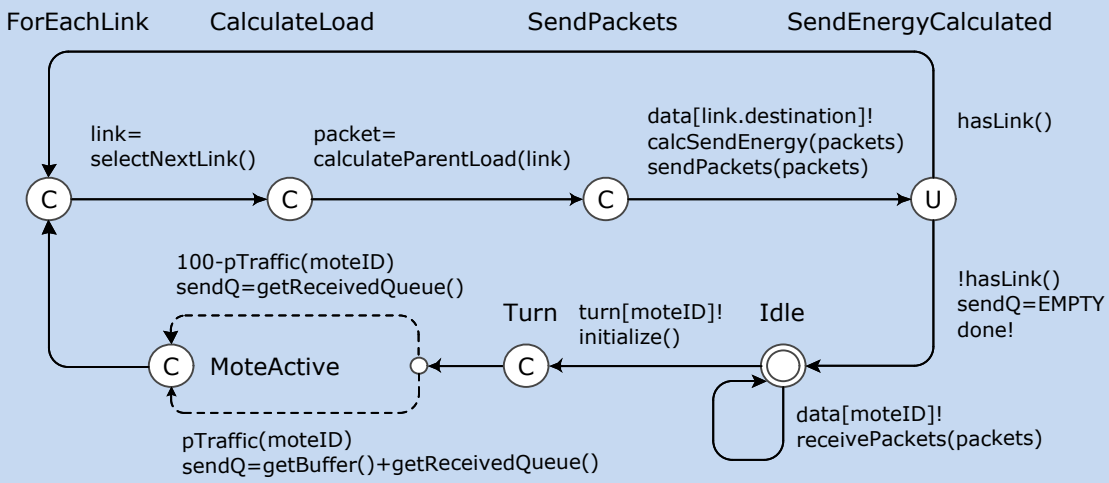


(c) System

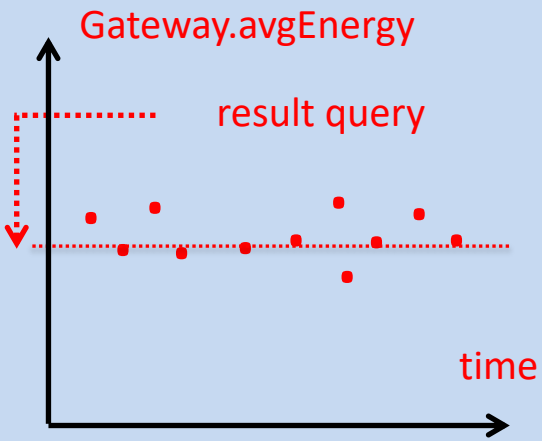
Uppaal verification query: simulate 1 [<=100] (<> Gateway.avgEnergy)

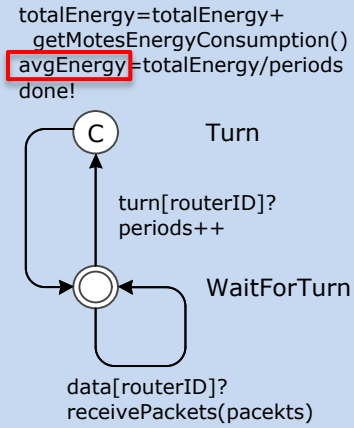


Uppaal verification query: simulate 1 [<=100] (<> Gateway.avgEnergy)

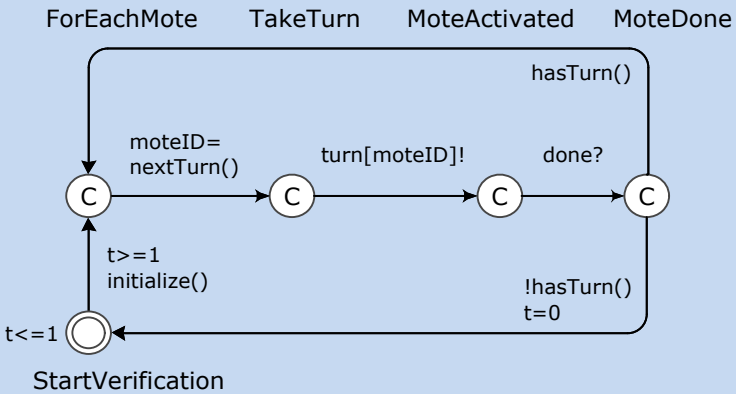


(a) Mote

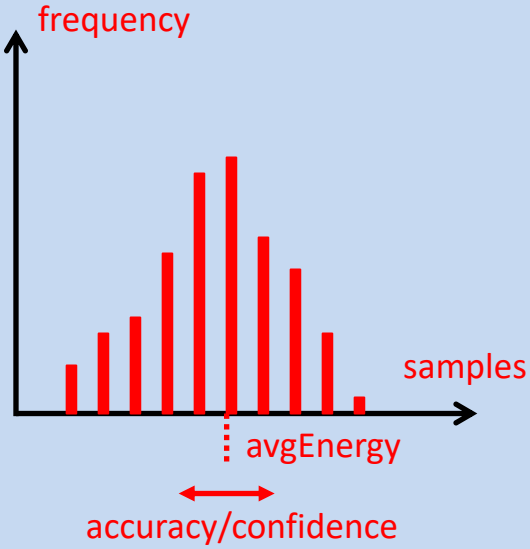




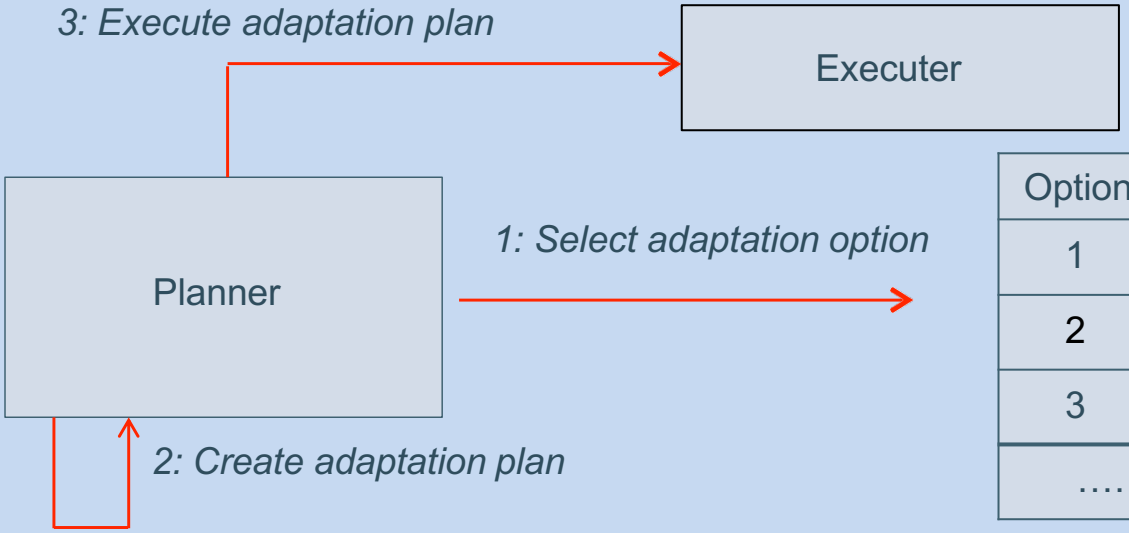
(b) Gateway



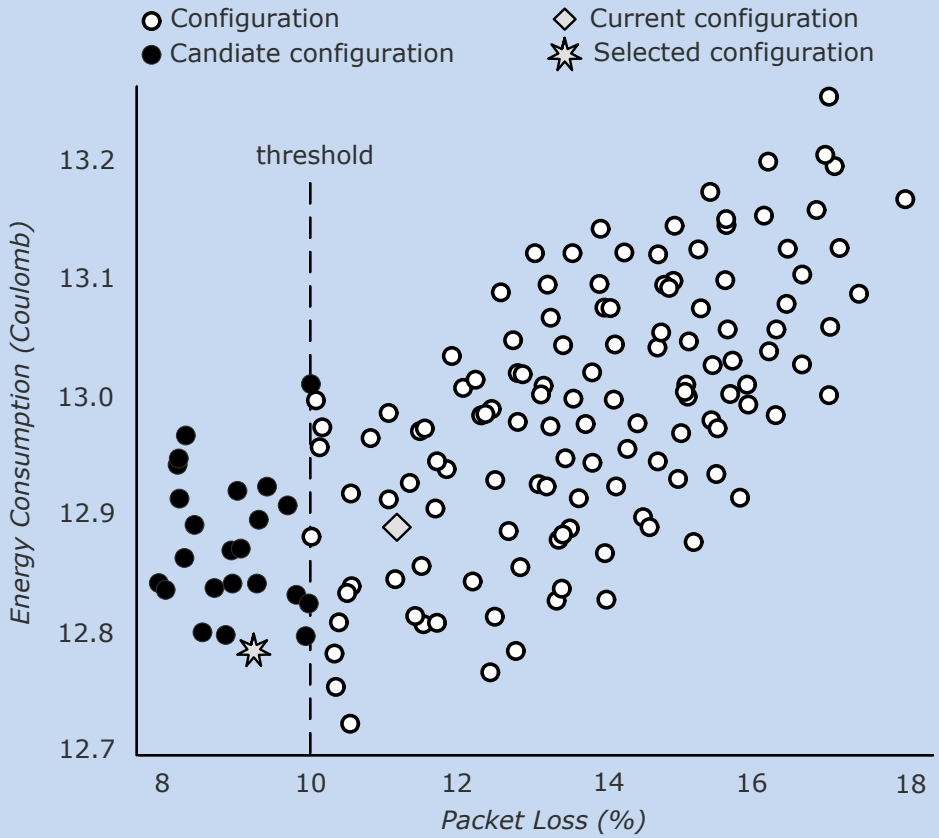
(c) System

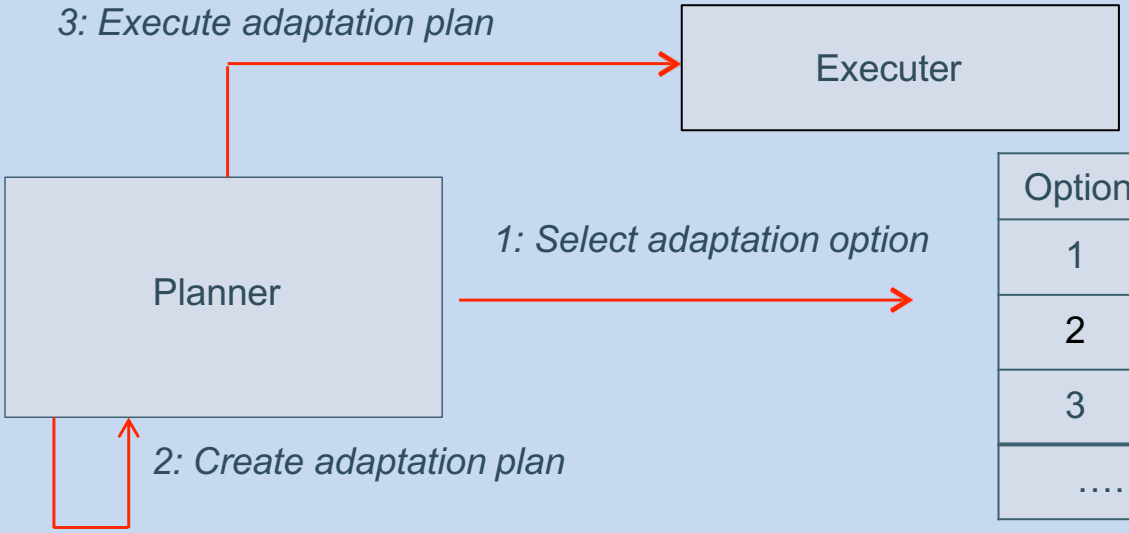


KEY

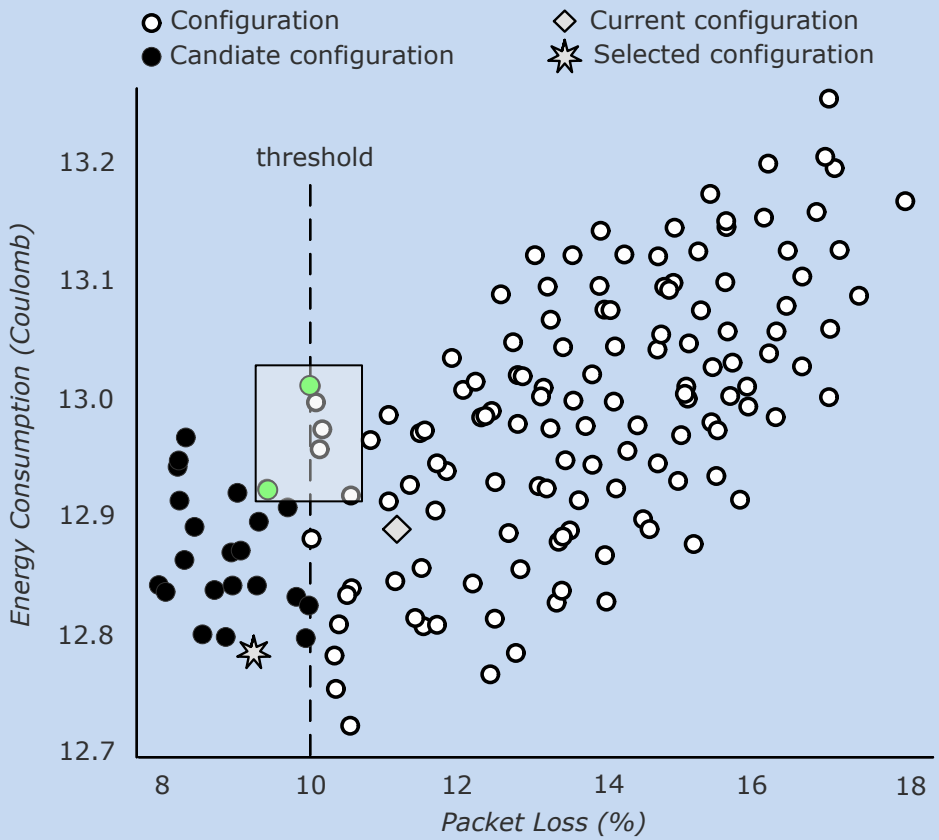


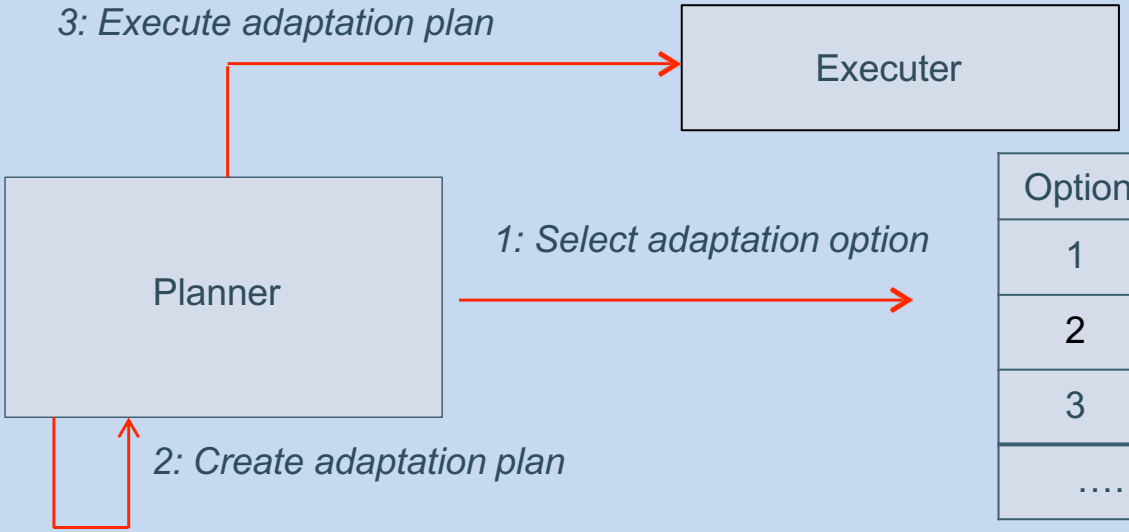
Option	Packet Loss	Energy
1	8.2 %	12.9 mC
2	9.7 %	12.7 mC
3	10.0 %	13.0 mC
……		



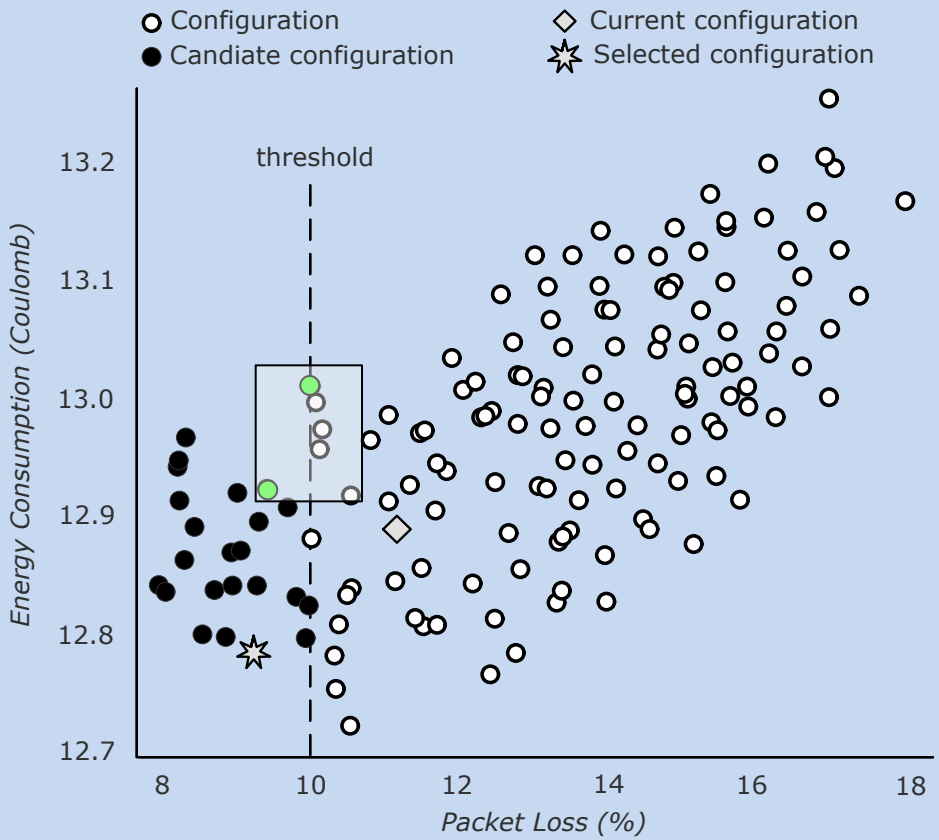


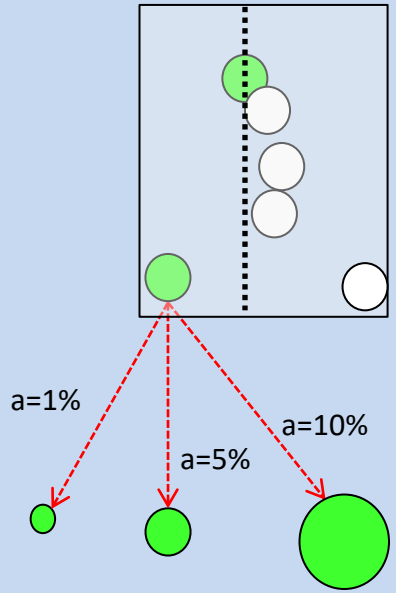
Option	Packet Loss	Energy
1	8.2 %	12.9 mC
2	9.7 %	12.7 mC
3	10.0 %	13.0 mC
……		



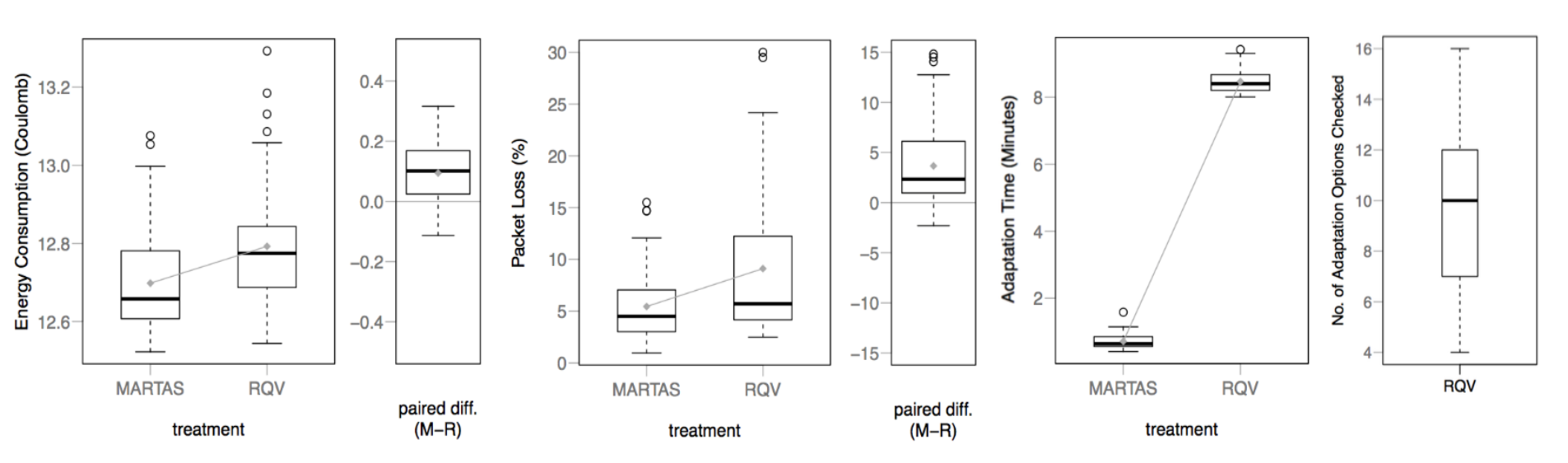


Option	Packet Loss	Energy
1	8.2 %	12.9 mC
2	9.7 %	12.7 mC
3	10.0 %	13.0 mC
……		





Comparison with exhaustive verification



Results for energy consumption

Results for packet loss

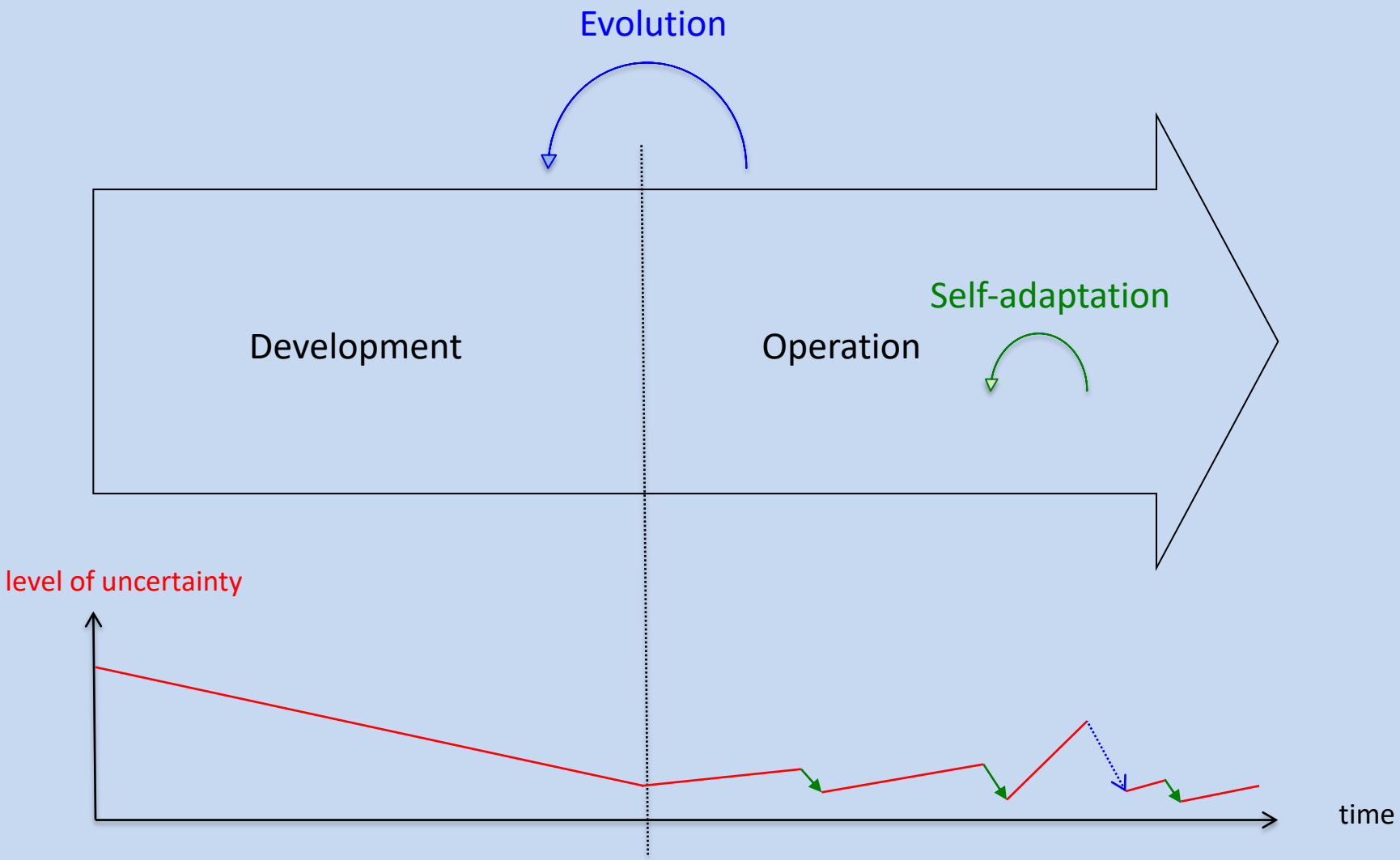
Significant less adaptation time

Overview

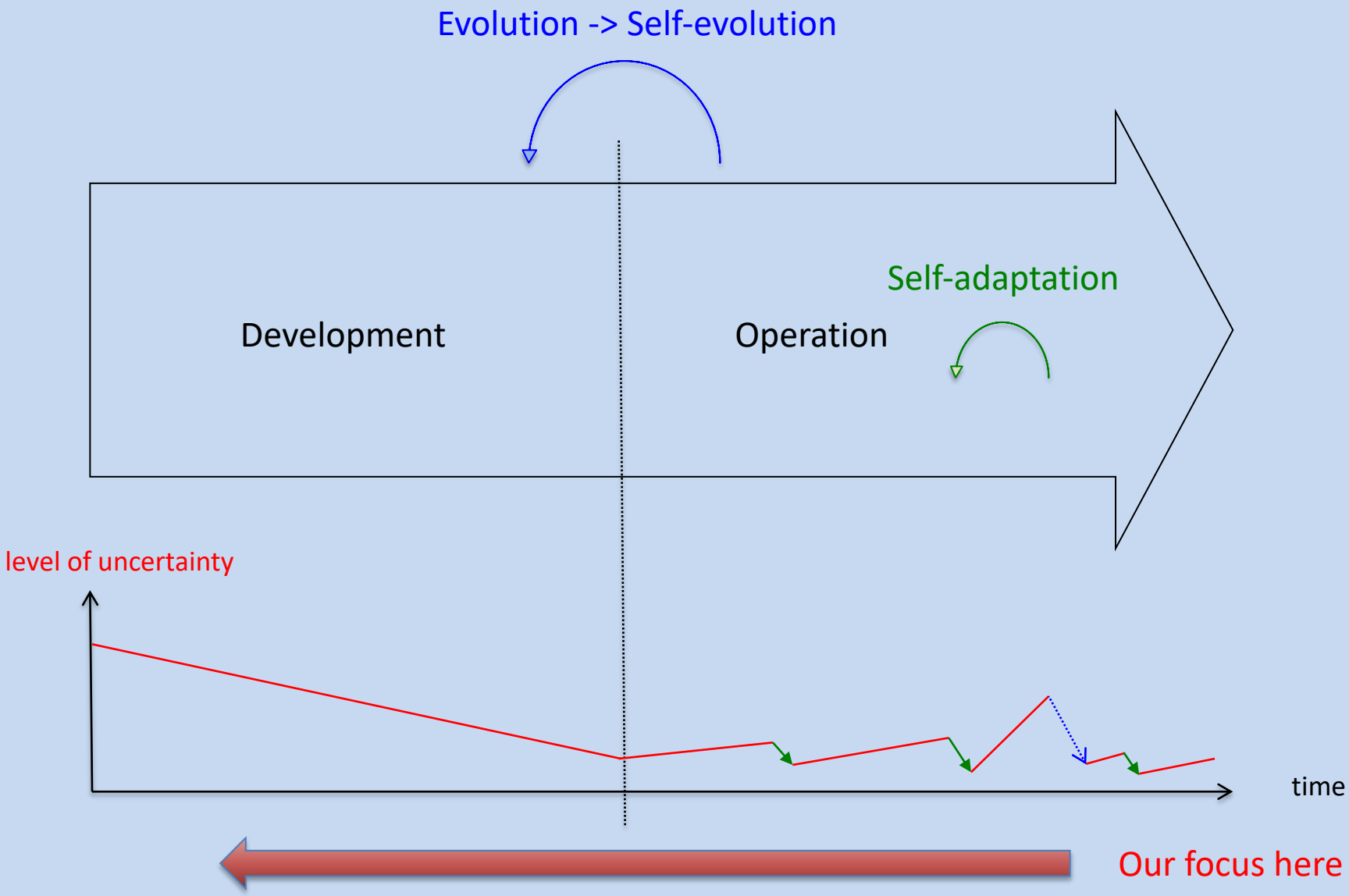
- Why self-adaptation?
- What is self-adaptation?
- Self-adaptation in industry
- How to engineer self-adaptive systems?
- Guarantees at runtime
- Why is “just self-adaptation” not enough?
- Conclusions

Why is “just self-adaption” not enough?

Taming uncertainty



From self-adaptation to self-evolution

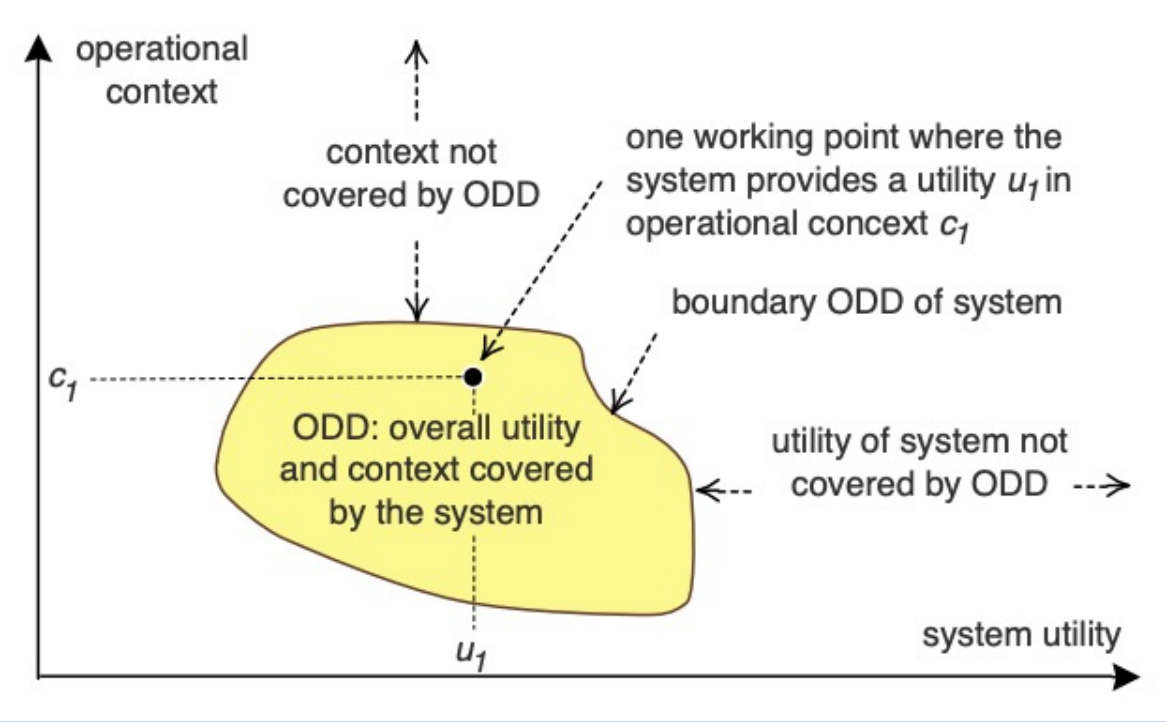


Operational Design Domain (ODD)

- We define the ODD of a computing system as the conditions for which it was built
  - Requirement R
  - Constraints B
  - Context C

  - $ODD_S = \{(u,c) \mid (u,c).sat(R,B,C\}_S \wedge c \in C, u \in U\}$

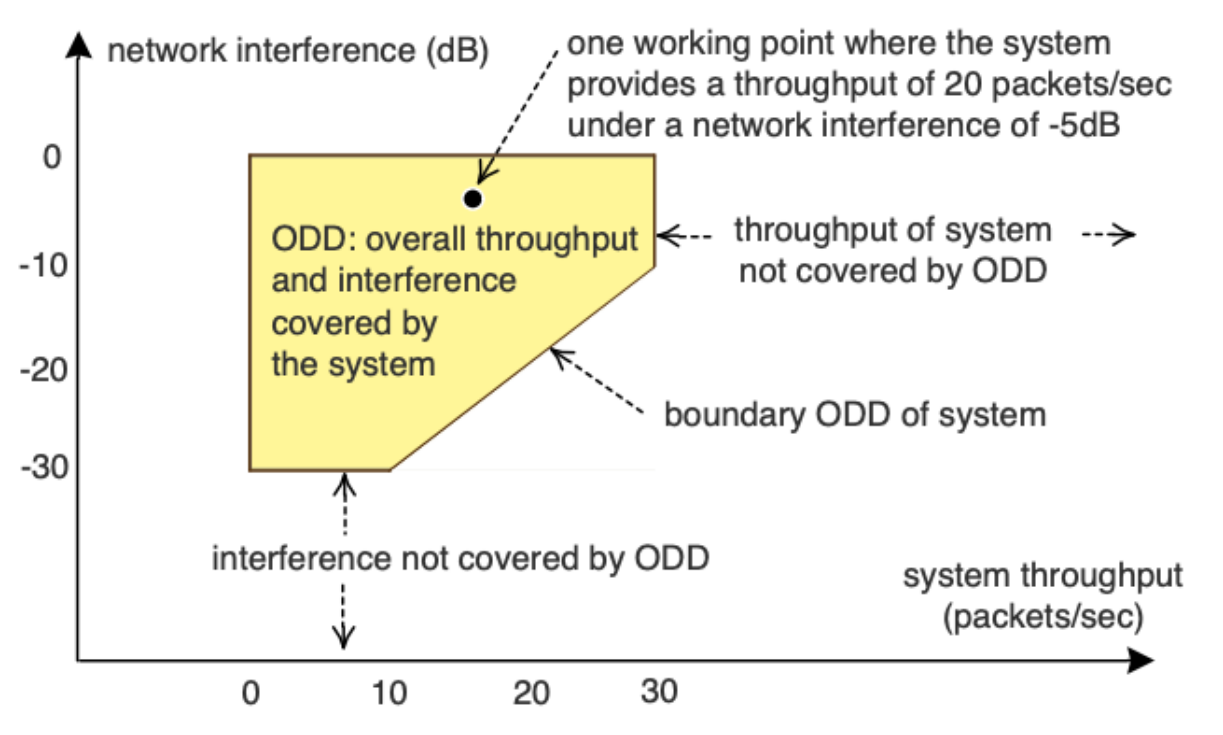
ODD regular computing system



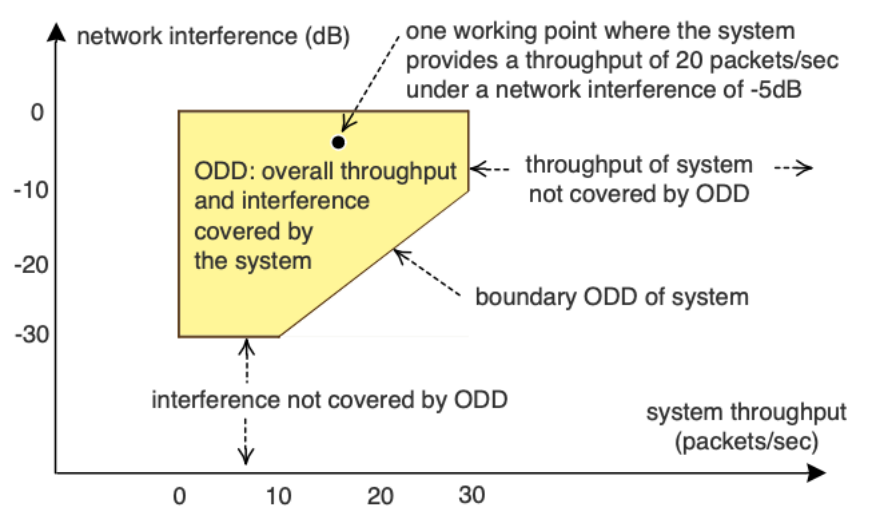
Consider simplified IoT system

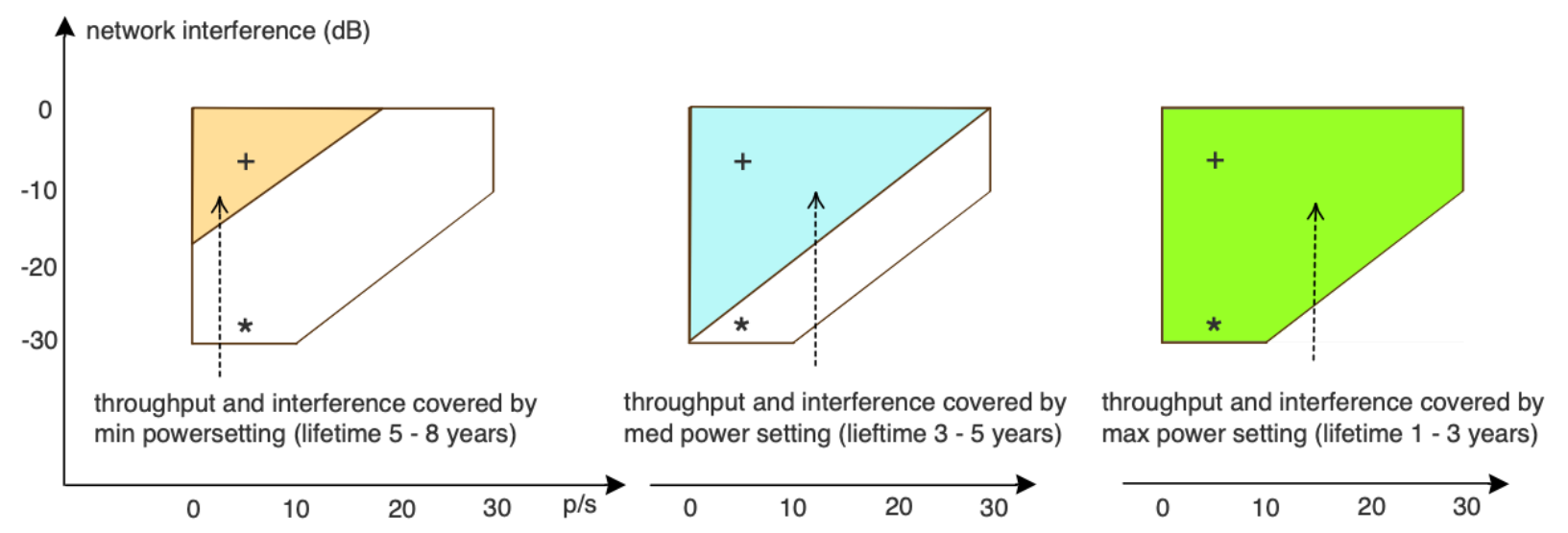
- Single uncertainty: variable interference network
- Adaptation goals: deliver packets with minimum packet loss while maximizing battery lifetime
- Adaptation options: 3 levels of power settings of nodes: low, medium, high

ODD of simplified IoT system

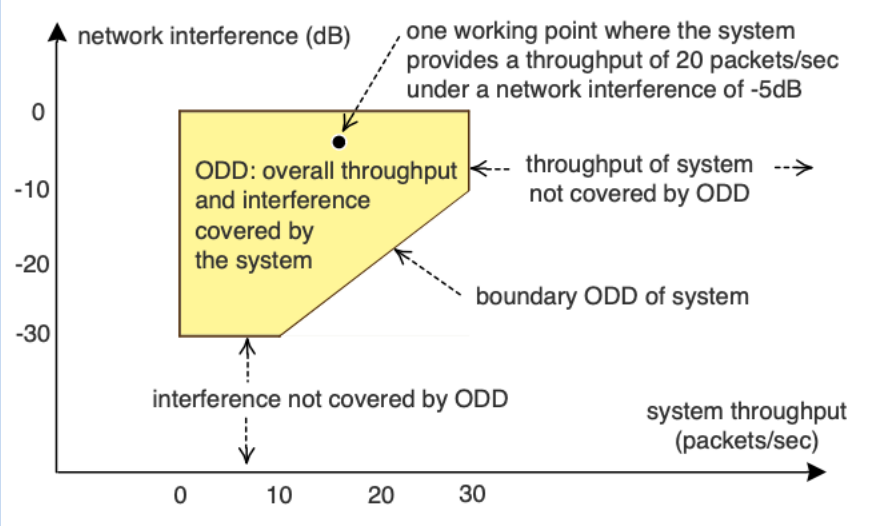


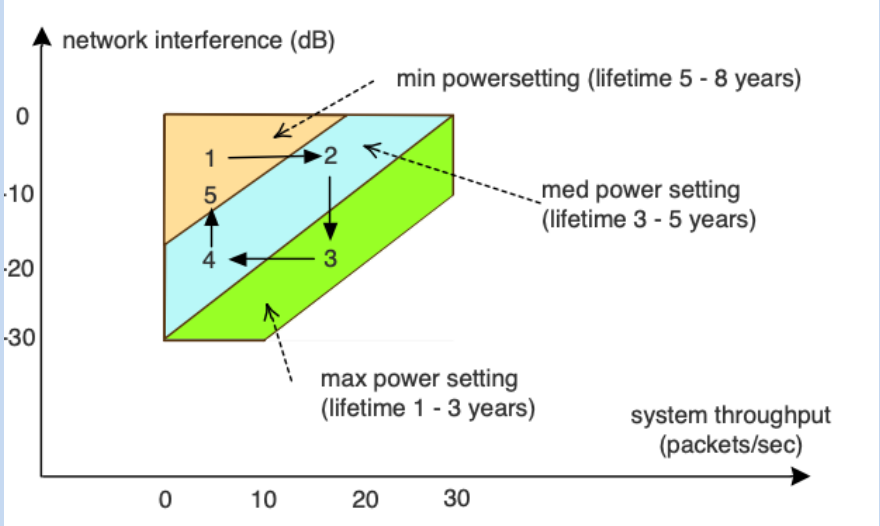
ODD of simplified IoT system





ODD of simplified <u>self-adaptive</u> IoT system





Operational Design Domain (ODD)

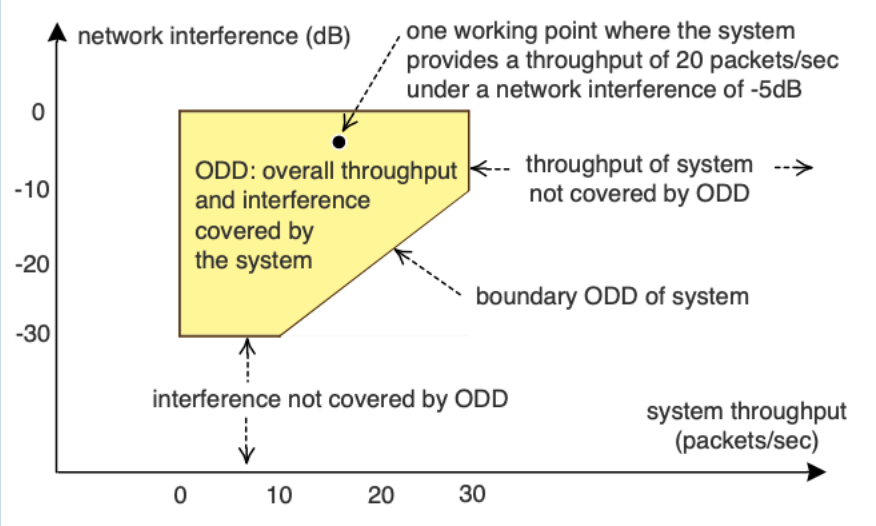
With self-adaptation the part of the ODD covered by operators is now covered by the system itself

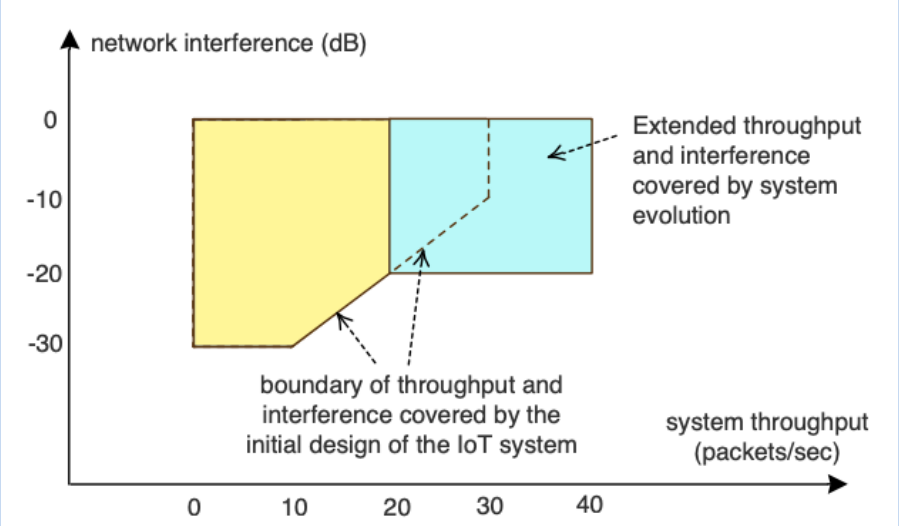
Yet, self-adaptation is not enough to extend the ODD

Self-adaptive systems cannot deal with uncertainty beyond what was anticipated (i.e., what the system was designed and built for)

Such uncertainty requires <u>evolution</u>

ODD and evolution simplified IoT system





ODD and system evolution

- Evolving a system = change its ODD
- Scenario: extending the ODD
  - $ODD_{Se} = ODD_S \cup ODD_e$

- Self-evolution requires:
  - Determine ODD_e
  - Determine new element that satisfies ODD_e
  - Enact new configuration that satisfies ODD_{Se}

Self-evolving computing

Current approaches to software evolution

= human-driven

Not sustainable in the future:

- Technologies and requirement advance rapidly
- Systems need to be available 24/7
- More sensors collecting heterogenous & complex data
- Dynamic operating conditions possibly unanticipated

How can be build computing systems that can evolve autonomously?

- Peek into the future
- Explorative/speculative

System layer

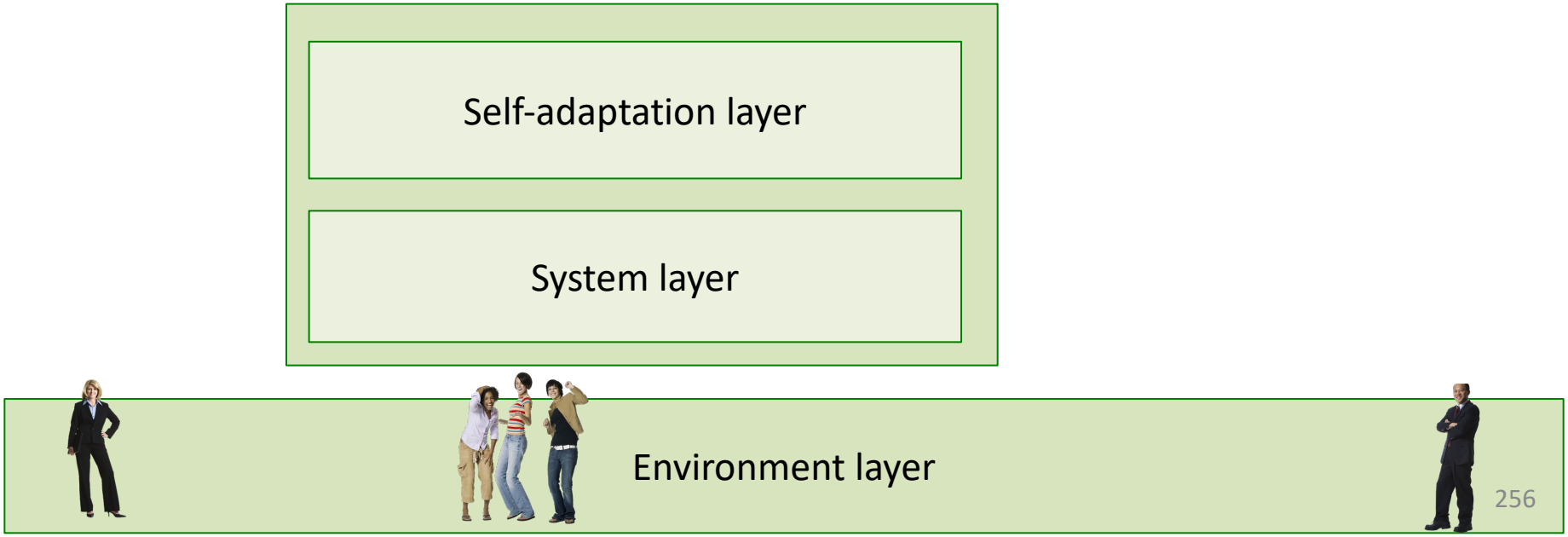
Environment layer

- Heterogeneous computing system comprising regular computing elements and learning elements

System layer

Environment layer

- Heterogeneous computing system comprising regular computing elements and learning elements
- Equipped with self-adaptation layer to deal with anticipated changes => self-adaptive system
- May involve humans in support



- Auto-evolution enabled computing and learning elements
- Populated by software companies
- “Catalog” to select elements
- Elements provided with “Usage guide” (specification, prerequisites, rights, deployment procedure, contract, etc.)
- Ready to be integrated by running systems

Computing Warehouse

Self-adaptation layer

System layer

Environment layer

Data Store

- Stores and maintains relevant data
- Produced and consumed by running systems
- Deployment: private, ecosystem, open platform

Computing Warehouse

Self-adaptation layer

System layer

Environment layer

- Self-learning platform performs evolution of the underling system autonomously
- Deals with unanticipated changes
- Exploits data store and computing warehouse

Data Store

Computing Warehouse

Self-evolving layer

Novelty & anomaly detection

Evolutionary Synthesis

Experimentation

Evolution

Self-awareness & Context-awareness

Self-adaptation layer

System layer

Evolution enactment

Environment layer

- Up to date architectural runtime models
- Self-awareness: runtime model of system and its goals
- Context-awareness: runtime model of environment

Data Store

Computing Warehouse

Self-evolving layer

Novelty & anomaly detection

Evolutionary Synthesis

Experimentation

Evolution

Self-awareness & Context-awareness

Self-adaptation layer

System layer

Evolution enactment

Environment layer

- Anomaly detection: identify suspicious issues
- Novelty detection: discover novel phenomena
- Including new goals added to the system by humans

Data Store

Computing Warehouse

Self-evolving layer

Novelty & anomaly detection

Evolutionary Synthesis

Experimentation

Evolution

Self-awareness & Context-awareness

Self-adaptation layer

System layer

Evolution enactment

Environment layer

- Assesses and diagnoses novelty/anomaly/goals
- Synthesis new architecture using runtime models, data of store, spec auto-enabled evolution elements

Data Store

Computing Warehouse

Self-evolving layer

Novelty & anomaly detection

Evolutionary Synthesis

Experimentation

Evolution

Self-awareness & Context-awareness

Self-adaptation layer

System layer

Evolution enactment

Environment layer

- Runs experiments with new candidate architectures in sandbox
- Centered on evolutionary algorithm
- Saves relevant data to store for future usage

Data Store

Computing Warehouse

Self-evolving layer

Novelty & anomaly detection

Evolutionary Synthesis

Experimentation

Evolution

Self-awareness & Context-awareness

Self-adaptation layer

System layer

Evolution enactment

Environment layer

- Identifies most promising architecture
- Guides synthesis and experimentation
- Saves relevant data to store for future usage

Data Store

Computing Warehouse

Self-evolving layer

Novelty & anomaly detection

Evolutionary Synthesis

Experimentation

Evaluation

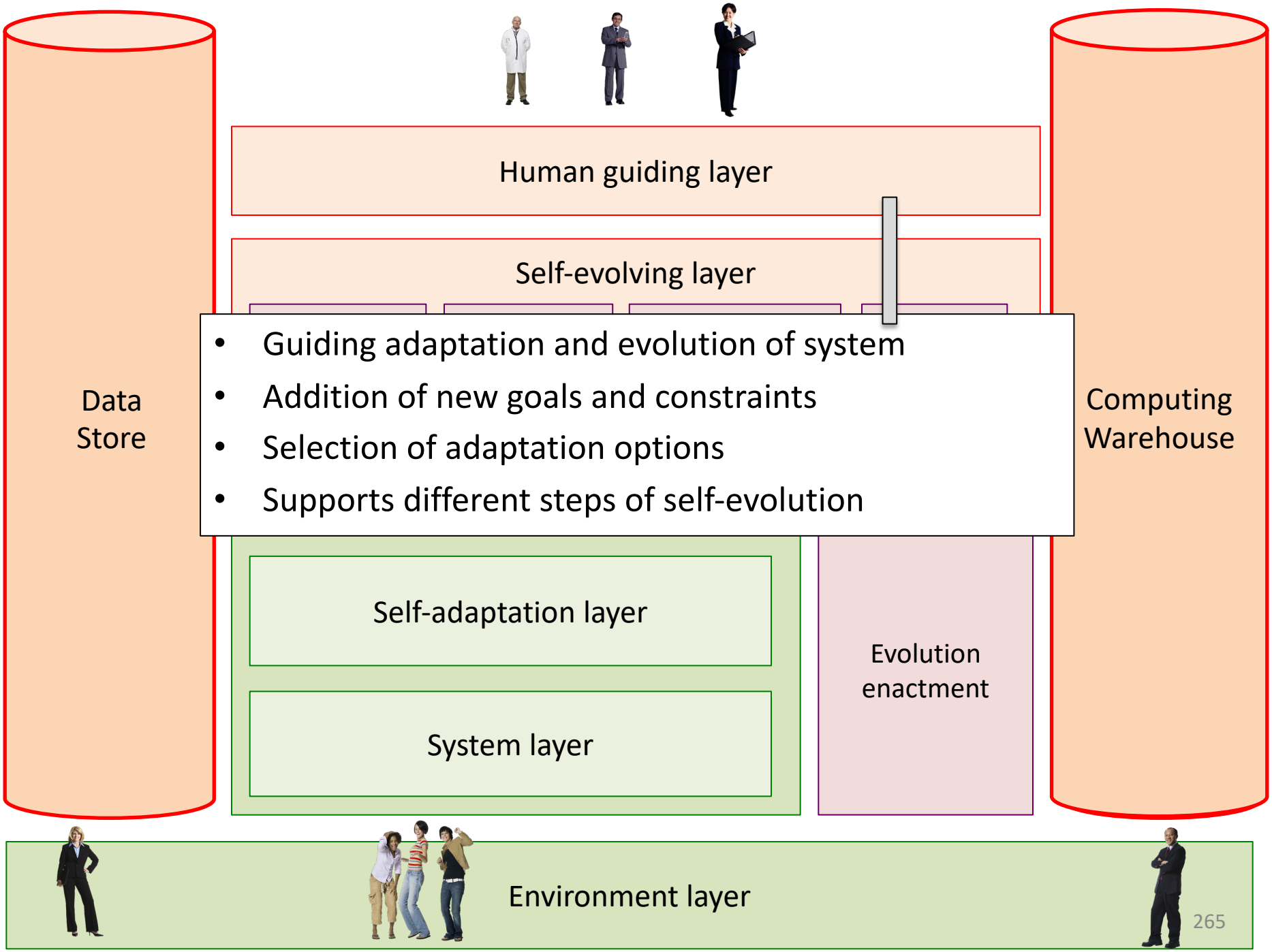
Self-awareness & Context-awareness

Self-adaptation layer

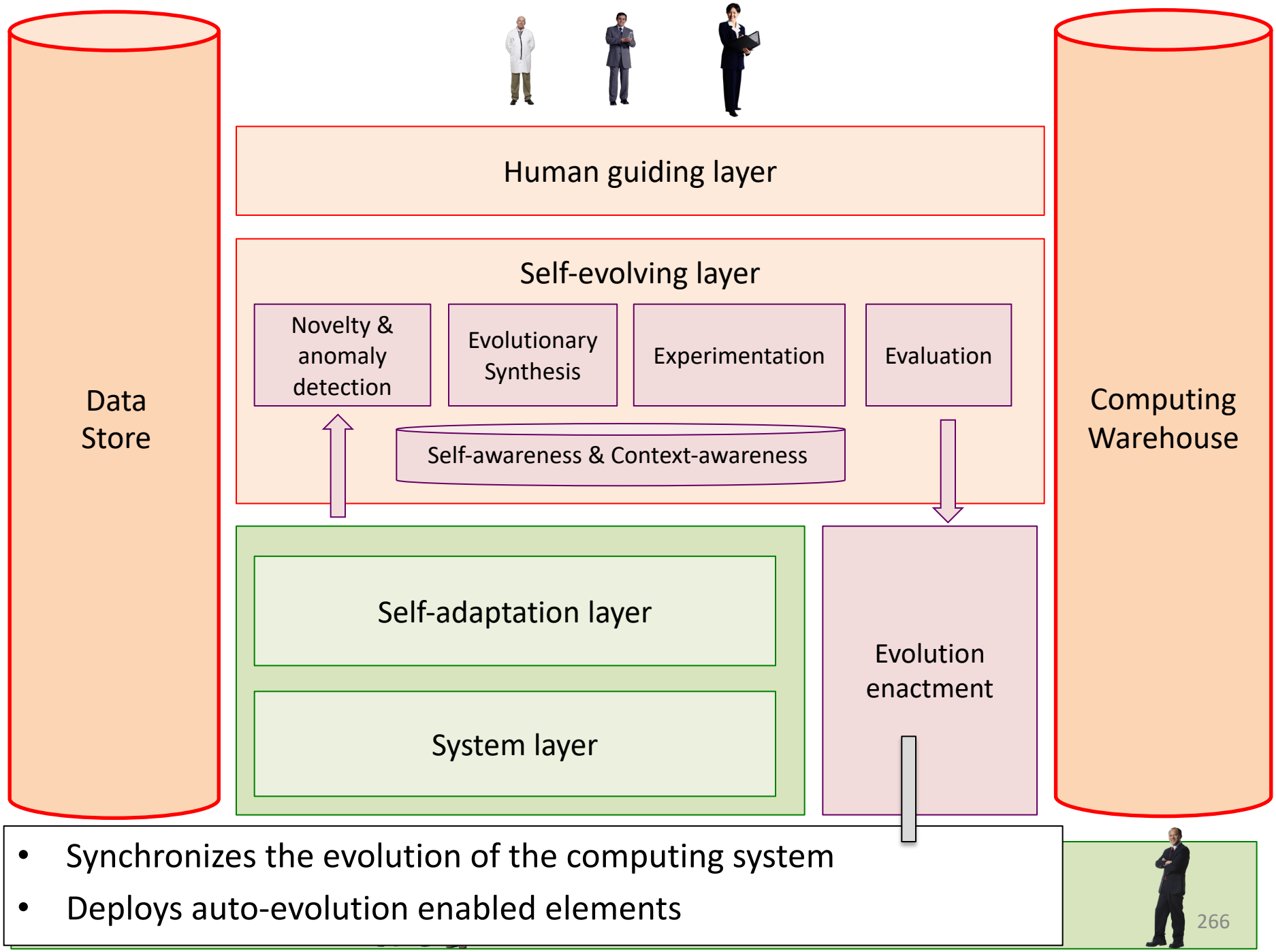
System layer

Evolution enactment

Environment layer



- Guiding adaptation and evolution of system
- Addition of new goals and constraints
- Selection of adaptation options
- Supports different steps of self-evolution



- Synchronizes the evolution of the computing system
- Deploys auto-evolution enabled elements

Self-Evolving Computing System

Data Store

Computing Warehouse

Human guiding layer

Self-evolving layer

- Novelty & anomaly detection
- Evolutionary Synthesis
- Experimentation
- Evaluation

Self-awareness & Context-awareness

Self-adaptation layer

System layer

Evolution enactment

Environment layer

Numerous open challenges

- Anomaly detection and novelty discovery mechanisms, including for complex data streams
- Concept of “auto-evolution enabled elements” and “computing warehouse”
- Online evolutionary learning framework
- Semantics for automatic composition
- Notations and mechanisms for involving humans
- Management of (historical) data of ”data store”
- (Formal) guarantees that systems comply with goals
- Integration of AI techniques

Overview

- Why self-adaptation?
- What is self-adaptation?
- Self-adaptation in industry
- How to engineer self-adaptive systems?
- Guarantees at runtime
- Why is “just self-adaptation” not enough?
- Conclusions

Conclusions

Recap goals of the course

- Illustrate the need for self-adaptation
- Explain the basic principles of self-adaptation
- Understand the use of self-adaptation in industry
- Understand the waves of research on self-adaptation
- Explain how to guarantee that the system complies with its goals at runtime
- Understand some of the open challenges in the field of self-adaptation

Conclusions

Growing need for automation and handling uncertainty in software systems

Self-Adaptation can contribute to high-quality software

Road ahead

- Knowledge transfer research <> industry
- Automate evolution to handle unanticiapted change
- Integrate with recent AI developments

Selection of advised reading

Selection of advised reading

P. Oreizy, M. Gorlick, R. Taylor, et al. An Architecture-Based Approach to Self-Adaptive Software. IEEE Intelligent Systems, 14(3):54–62, 1999

Pioneering and visonary work

- Academic motivation of the need for self-adaptation
- Architecture perspective
- Seemless integration of adaptation and evolution

Selection of advised reading

Kephart and Chess, The vision of autonomic Computing, IEEE Computer, January 2003

Pioneering and visionary work

- Industrial motivation of the need for self-adaptation
- Principles of autonomic computing
- MAPE-K

Selection of advised reading

D. Garlan, S-W. Cheng, A.C. Huang, B. Schmerl, P. Steenkiste, Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure, IEEE Computer, October 2004

Pioneering work

- Motivated the need for architecture-models at runtime
- Reusable framework that realised MAPE-K

Selection of advised reading

J. Kramer and J. Magee. Self-Managed Systems: An Architectural Challenge. In Future of Software Engineering, ICSE 2007

Pioneering work

- Motivated the need for separating change management from goal management
- 3 layer reference model for self-adaptation

Selection of advised reading

R. Calinescu, L. Grunske, M. Kwiatkowska, R. Mirandola, and G. Tamburrelli. Dynamic QoS Management and Optimization in Service-Based Systems, IEEE TSE 2011

Pioneering work

- Use of online learning (of quality model)
- Application of quantitative verification at runtime

Selection of advised reading

G. Moreno, J. Cámara, D. Garlan, B. Schmerl, Proactive self-adaptation under uncertainty: a probabilistic model checking approach, FSE 2015

Runtime modelling and verification

- Proactive self-adaptation
- Adaptation decisions with a look-ahead horizon

Selection of advised reading

R. Calinescu, D. Weyns, S. Gerasimou, U. Iftikhar, I. Habli, T. Kelly, Engineering Trustworthy Self-Adaptive Software with Dynamic Assurance Cases, IEEE TSE 44(11), 2018

Design-time and runtime modelling and verification

- ENTRUST methodology
- Industry-adopted assurance cases

Selection of advised reading

D. Weyns and U. Iftikhar, ActivFORMS: A Formally-Founded Model-Based Approach to Engineer Self-Adaptive Systems, ACM TOSEM 32(1), 2023

Design-time and runtime modelling and verification

- ActivFORMS methodology
- Statistical model checking at runtime

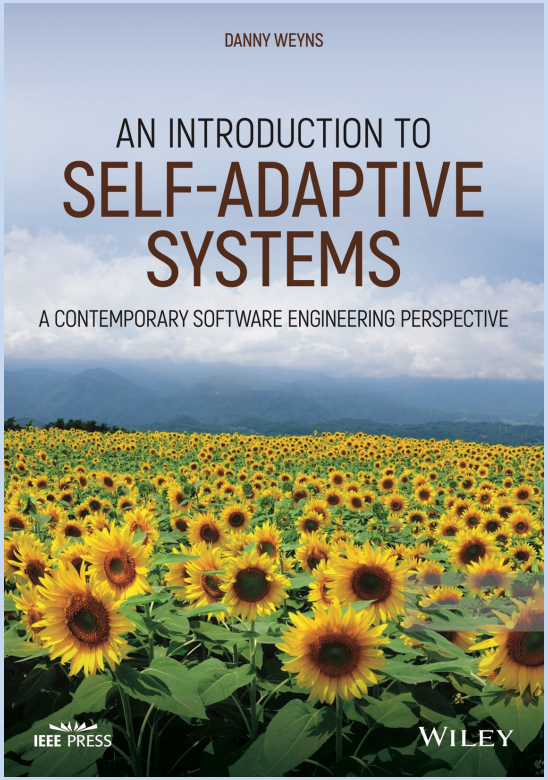
Selection of advised reading

Jialong Li, Mingyue Zhang, Nianyu Li, Danny Weyns, Zhi Jin, Kenji Tei, Generative AI for Self-Adaptive Systems: State of the Art and Research Challenges, ACM TAAS, Sept 2024

Explores how Generative AI may support self-adaptation

- Leverages use of generative AI in software engineering
- Outlines roadmap for future research

Selection of advised reading



Selection of advised reading

Home SEAMS Dagstuhl Seminars **Exemplars** Mailing List News

Software Engineering for Self-Adaptive Systems

self-adaptive.org

Exemplars

Self-Adaptive Systems Artifacts and Model Problems

This site contains a set of exemplars for self-adaptive systems. An exemplar can be quite generic such as an artifact or rather specific such as a model problem in self-adaptive systems. The goal is to provide a repository of examples, challenge problems, and solutions that the software engineering for self-adaptive systems community can use to motivate research, exhibit and evaluate solutions and techniques, and compare results. Artifacts of interest include but are not limited to:

- *Testbeds / Model Problems / Managed Systems*, which are implementations or detailed specifications of systems that pose and highlight fundamental or characteristic challenges that self-adaptive systems should address.
- *Datasets*, which are data (e.g., logging data, sensor data, system traces, survey raw data) that can be used to develop, evaluate, and compare self-adaptation approaches.
- *Frameworks*, which are tools and services illustrating and implementing self-adaptation techniques or algorithms that are potentially useful in different contexts and that other researchers could use and customize to specific contexts.

EXEMPLARS

- Znn.com
- ATRP
- Hogna
- DEECo
- TAS
- Feed me, Feed me
- Hadoop-Benchmark
- Self-Adaptive Video Encoder
- UNDERSEA
- DeltaIoT
- CrowdNav and RTX
- Intelligent Ensembles
- Lotus@Runtime
- mRUBiS
- K8-Scalar
- SWIM
- TRAPP
- PiStarGODA-MDP
- OCCI Monitoring
- DingNet
- Dragonfly
- DARTSim

Artifacts for self-adaptation: https://www.hpi.uni-potsdam.de/giese/public/selfadapt/exemplars/
Guidelines for industry-relevant artifacts for self-adaptation: https://arxiv.org/abs/2206.12492

Bibliography

- Kephart and Chess, The vision of autonomic Computing, IEEE Computer, January 2003
- IBM. An Architectural Blueprint for Autonomic Computing, November 2019. URL: https://www-03.ibm.com/autonomic/pdfs/ACBlueprintWhitePaperV7.pdf
- P. Oreizy, M. Gorlick, R. Taylor, et al. An Architecture-Based Approach to Self-Adaptive Software. IEEE Intelligent Systems, 14(3):54–62, 1999
- D. Garlan, S-W. Cheng, A.C. Huang, B. Schmerl, P. Steenkiste, Rainbow: Architecture- Based Self-Adaptation with Reusable Infrastructure, IEEE Computer, October 2004
- D. Weyns, U. Iftikhar, and J. Soderland. Do External Feedback Loops Improve the Design of Self-adaptive Systems? A Controlled Experiment, Symposium in *Software Engineering for Adaptive and Self-Managing Systems,* 2013
- D. Weyns et al., On Patterns for Decentralized Control in Self-Adaptive Systems. Software Engineering for Self-Adaptive Systems II, Springer, 2013
- J. Kramer and J. Magee, Self-adaptation: an architecture challenge, Future of Software Engineering, Future of Software Engineering, FOSE 2007
- D. Weyns, S. Malek, J. Andersson, FORMS: Formal reference model for self-adaptation, ACM Transactions on Autonomous and Adaptive Systems, TAAS 7(1), 2012
- G. Blair, N. Bencomo, and R. B. France. Models@run.time. Computer, 42(10):22–27, 2009
- B. Cheng et al. A Goal-Based Modeling Approach to Develop Requirements of an Adaptive System with Environmental Uncertainty. Model Driven Engineering Languages and Systems, 2009
- J. Whittle, P. Sawyer, N. Bencomo, B. Cheng, and J. Bruel. RELAX: Incorporating Uncertainty into the Specification of Self-Adaptive Systems. In *17th IEEE International Requirements Engineering Conference*, 2009
- V. Souza, A. Lapouchnian, W. Robinson, and J. Mylopoulos. Awareness Requirements for Adaptive Systems. Symposium on Software Engineering for Adaptive and Self-Managing Systems, 2011
- N. Esfahani and S. Malek. Uncertainty in Self-Adaptive Software Systems. Software Engineering for Self-Adaptive Systems II, Springer, 2013

Bibliography

- S. Hezavehi, D. Weyns, P.Avgeriou, R. Calinescu, R. Mirandola, D. Perez-Palacin, Uncertainty in Self-adaptive Systems: A Research Community Perspective, ACM Trans. on Autonomous & Adaptive Systems, 15(10) 2020
- R. Calinescu, L. Grunske, M. Kwiatkowska, R. Mirandola, and G. Tamburrelli. Dynamic QoS Management and Optimization in Service-Based Systems, IEEE Transactions on Software Engineering, TSE 2011
- Z. Coker, D. Garlan, and C. Le Goues. SASS: Self-adaptation Using Stochastic Search. In 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, 2015
- T. Vogel and H. Giese. Model-Driven Engineering of Self-Adaptive Software with EUREMA. ACM Transactions on Autonomous and Adaptive Systems, 8(4):18:1–18:33, 2014
- U. Iftikhar, D. Weyns: ActivFORMS: Active formal models for self-adaptation, International Symposium in Software Engineering of Adaptive and Self-Managing Systems, SEAMS 2014
- D. Weyns, U. Iftikhar, ActivFORMS: A Formally-Founded Model-Based Approach to Engineer Self-Adaptive Systems, ACM Trans. on Software Engineering & Methodology (https://arxiv.org/abs/1908.11179) 2022
- A. Filieri et al. Software Engineering Meets Control Theory. In 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, 2015
- F. Quin, D. Weyns, T. Bamelis, S. Buttar, and S. Michiels. Efficient Analysis of Large Adaptation Spaces in Self-adaptive Systems Using Machine Learning. Software Engineering for Adaptive and Self-Managing Systems, 2019
- D. Weyns, O. Gheibi, F. Quin, J. Van Der Donckt, Deep Learning for Effective and Efficient Reduction of Large Adaptation Spaces in Self-Adaptive Systems, ACM Trans. on Autonomous and Adaptive Systems 17(1-2) 2022
- D. Weyns, Software Engineering of Self-Adaptive Systems: Chapter for Handbook of Software Engineering, Editors: Richard Taylor, Kyo Chul Kang and Sungdeok Cha, Springer 2018
- D. Weyns, T. Bäck, R. Vidal, X. Yao, N. Belbachir: The Vision of Self-Evolving Computing Systems. Journal of Integrated Design and Process Science, CoRR abs/2204.06825, 2022
- D. Weyns, Ilias Gerostathopoulos et al, Self-Adaptation in Industry, ACM TAAS 18(2), 2023
- D. Weyns, An Introduction to Self-Adaptive Systems: A Contemporary Software Engineering Perspective, Wiley, 2020
- Jialong Li, Mingyue Zhang, Nianyu Li, Danny Weyns, Zhi Jin, Kenji Tei, Generative AI for Self-Adaptive Systems: State of the Art and Research Challenges, ACM TAAS, Sept 2024