Technische Universität Dresden
Fakultät Informatik
Lehrstuhl Softwaretechnologie

Cross-Layer Adaptation in Multi-Layer Autonomic Systems

Uwe Aßmann, Dominik Grzelak, Johannes Mey, Dmytro Pukhkaiev, René Schöne, Christopher Werner
Georg Püschel (wandelbots.de)

January 30, 2019

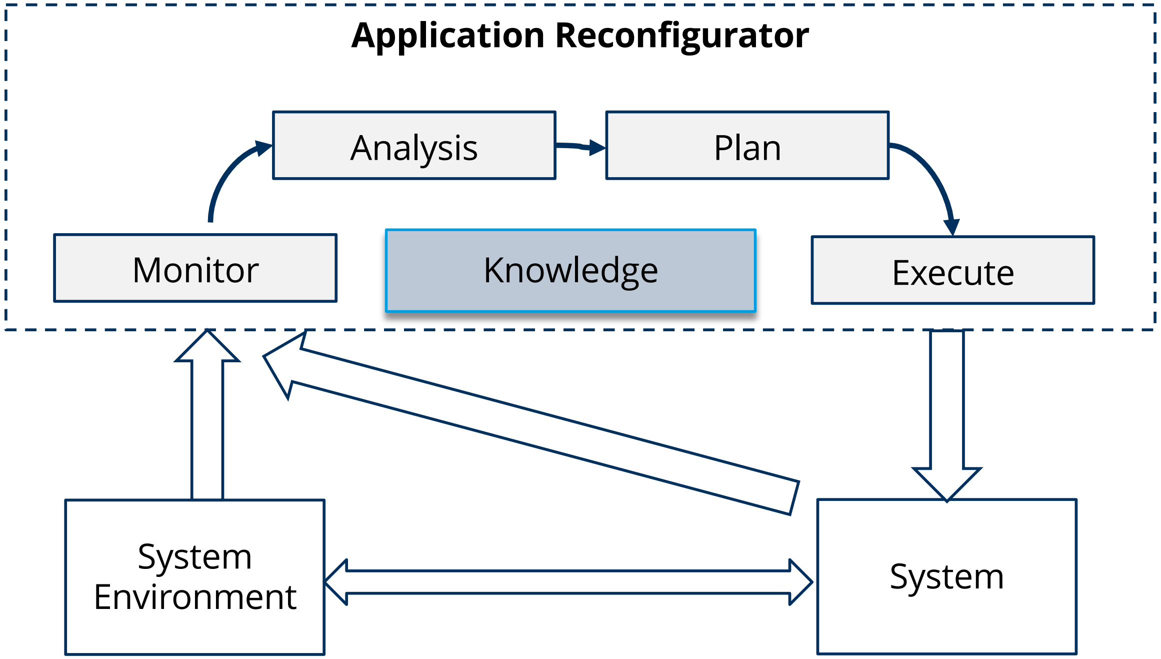
SOFSEM, Novy Smokovec

Overview

1. Self-Adaptive and Autonomic Systems
    1. Robotic Coworking Cells
    2. Highly-Adaptive Energy-Efficient Servers
2. Simple Autonomic Systems
    1. SMAGS for Robotic Coworking
    2. MQuAT for Highly-Adaptive Energy-Efficient Servers
3. Multi-Layer Automonic Systems (MuLAS)
4. Context-Controlled Autonomic Controllers (ConAC)
    1. In Action for Cinderella
5. Quality-Context-Controlled Autonomic Controllers (qConAC)
6. Energy-Context-Controlled Autonomic Controllers (eConAC)

1. Why Do We Need Self-Adaptive Systems (Autonomic Systems)

MAPE-K Loop in Autonomic Systems

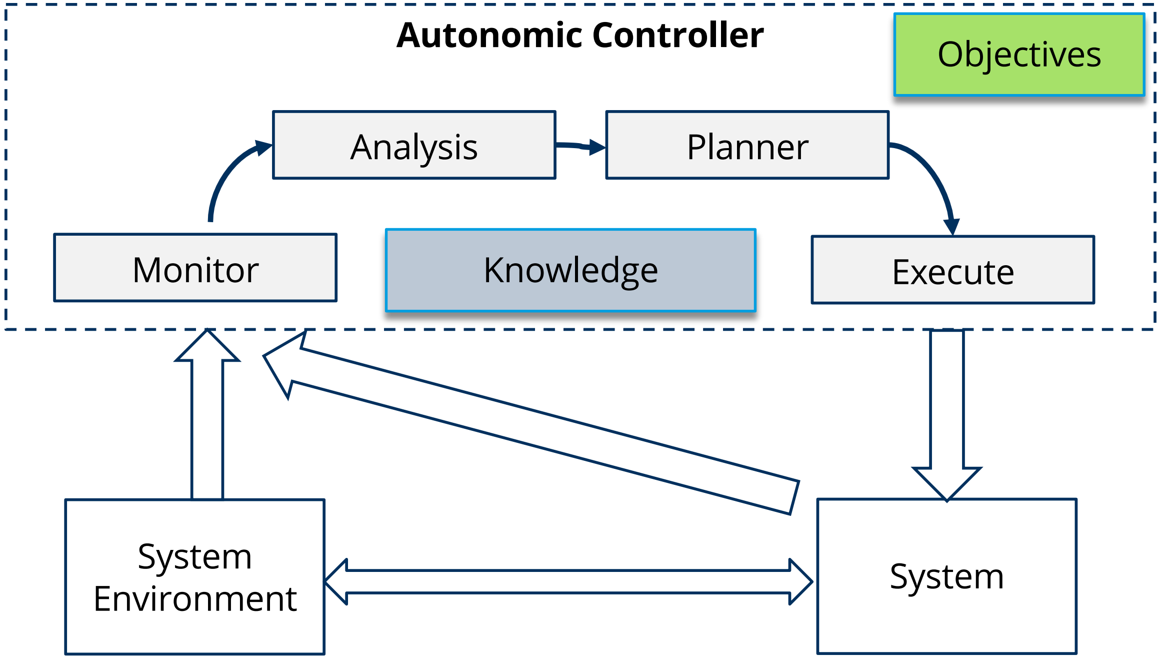


Definitions of Autonomic Software Systems

- An **Autonomic Controller** object runs a MAPE-K loop to control other objects
- An **Autonomic System** is a self-adaptive system controlled by an autonomic controller
- An **Autonomic Software Product Line or Family (ASPL)** is a variant family whose dynamic reconfiguration is controlled by a MAPE-K loop [Abbas 2010]
- **MAPE-K patterns** are design patterns for MAPE-K loops in ASPL [Weyns 2013]

MAPE-K Loop of Self-Optimizing Systems

- An autonomic system is called **self-optimizing**, if it has an explicit objective function (objective model).

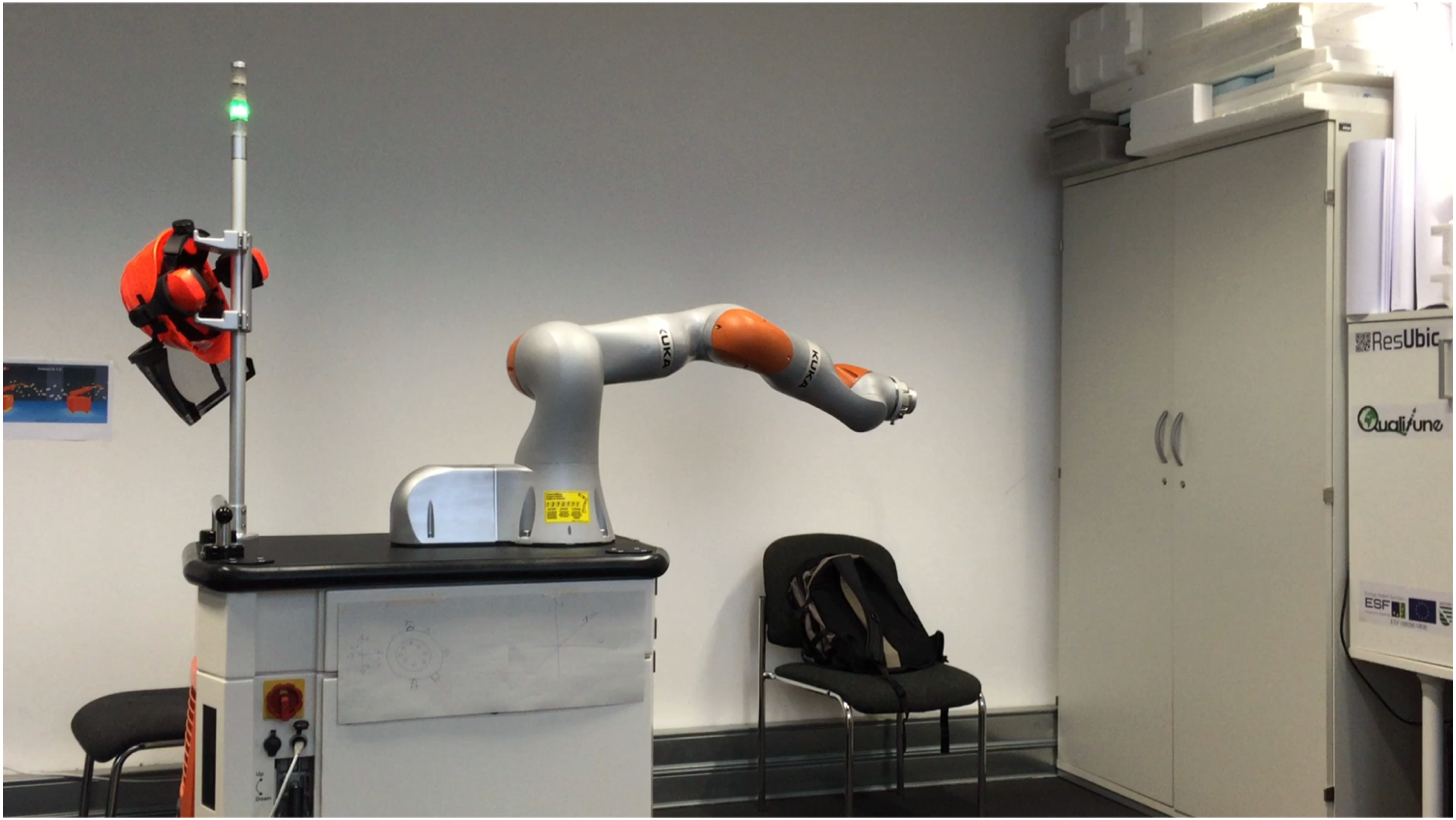


TECHNISCHE UNIVERSITÄT DRESDEN

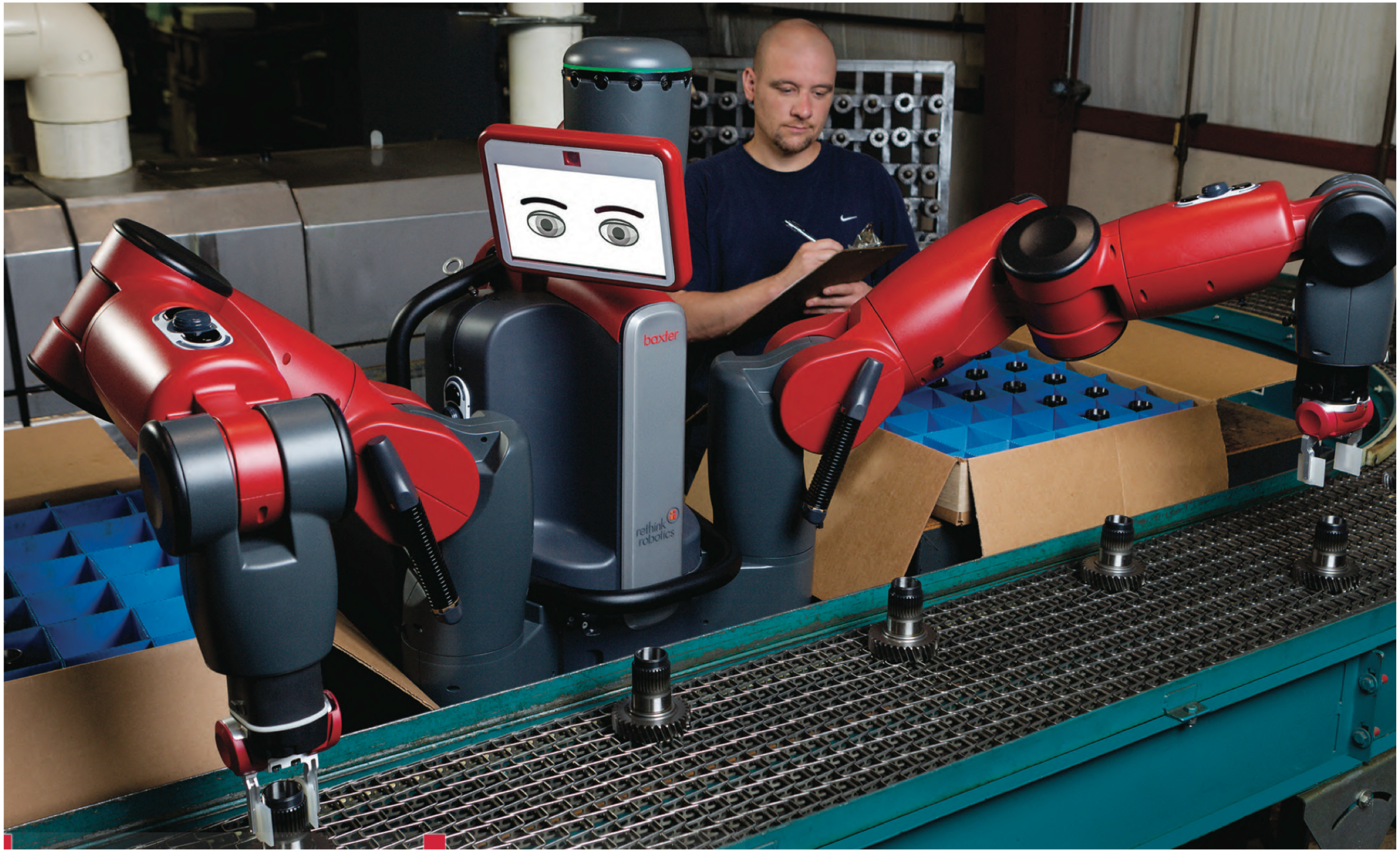
DRESDEN concept

1.1. Example 1: Robotic Coworking

KUKA LBR iiwa in our Lab

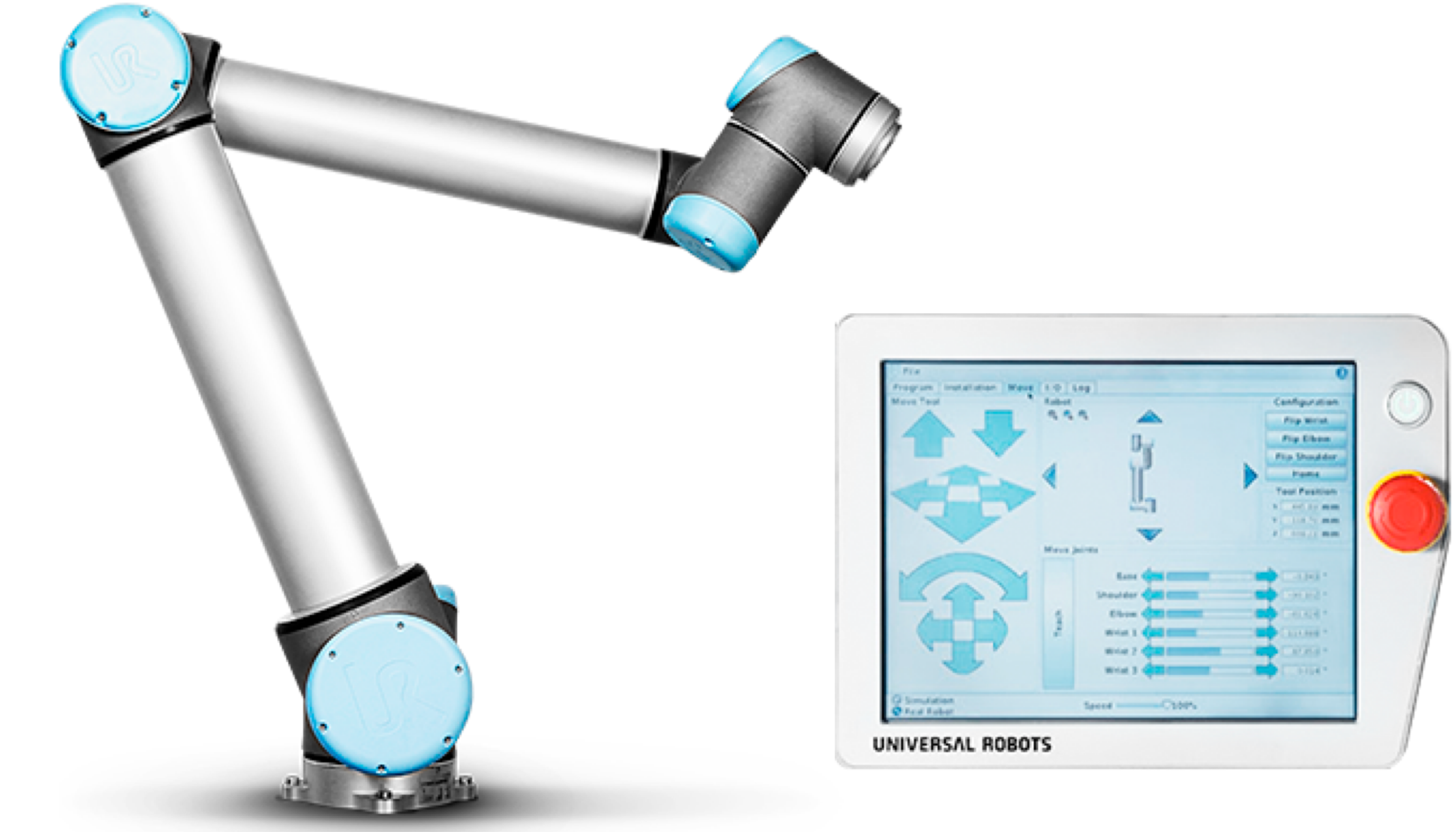


Baxter (Rethink Robotics)



[Baxter Cutsheet http://cdn-staging.rethinkrobotics.com/wp-content/uploads/2014/08/Baxter_Cutsheet_2014.pdf]

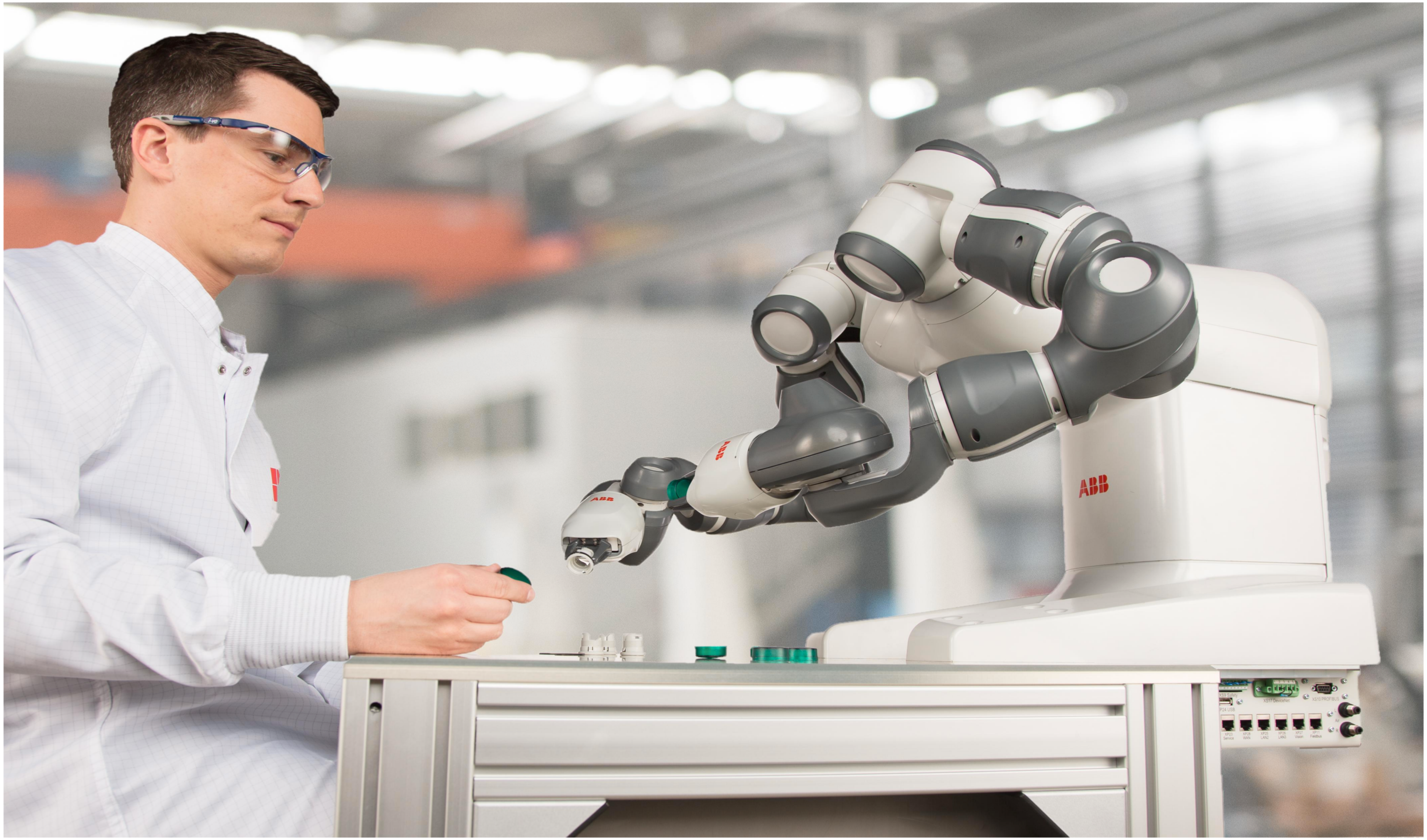
UR10 (Universal Robots)



http://www.universal-robots.com/products/ur10-robot

http://www.universal-robots.com/products/ur-robot-benefits

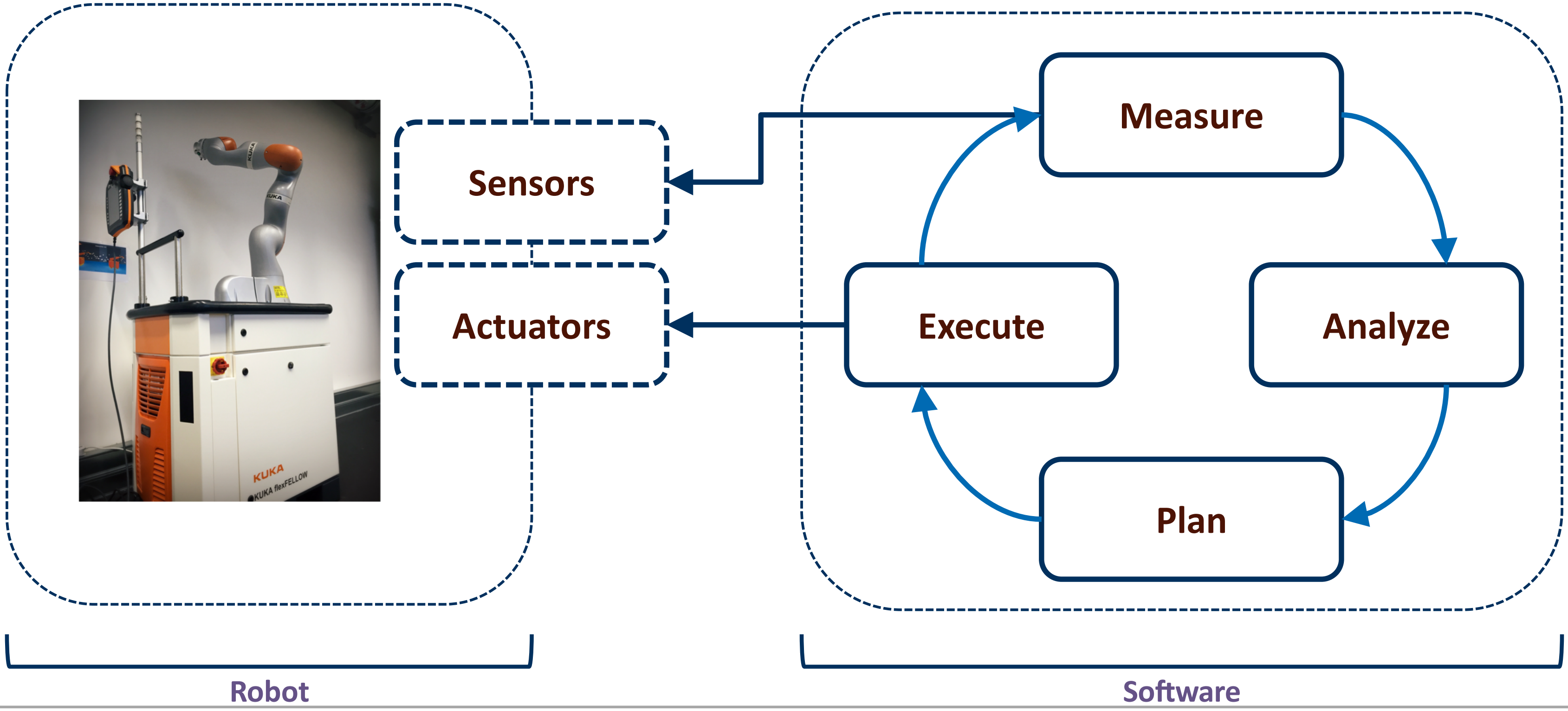
ABB YuMi



http://new.abb.com/products/robotics/de/yumi

Robotic Co-Workers are Self-Adaptive Systems (MAPE Loop)

External triggers for MAPE loop (asynchronous events)



Costs of a Single Co-Worker Robot in a Co-Assembly Line

Cost = Invest + Electricity + Maintentance:

0.3€/hr electricity

1000€ / year maintenance

7 years, 24/7

	Kuka LBR iiwa	Baxter	ABB Yumi
Invest	90000€	$23000	40000€
Invest/hr	1.47€	0.38$	0.65€
Electricity/hr	0.30€	0.30€	0.30€
Mainten./hr	0.11€	0.11€	0.11€
Cost/hour	1.89€	0.79€	1.07€

https://www.reddit.com/r/AskEngineers/comments/38u7n4/what_does_the_operation_of_an_average_assembly/

Robotic Co-Workers

Do not need high investments

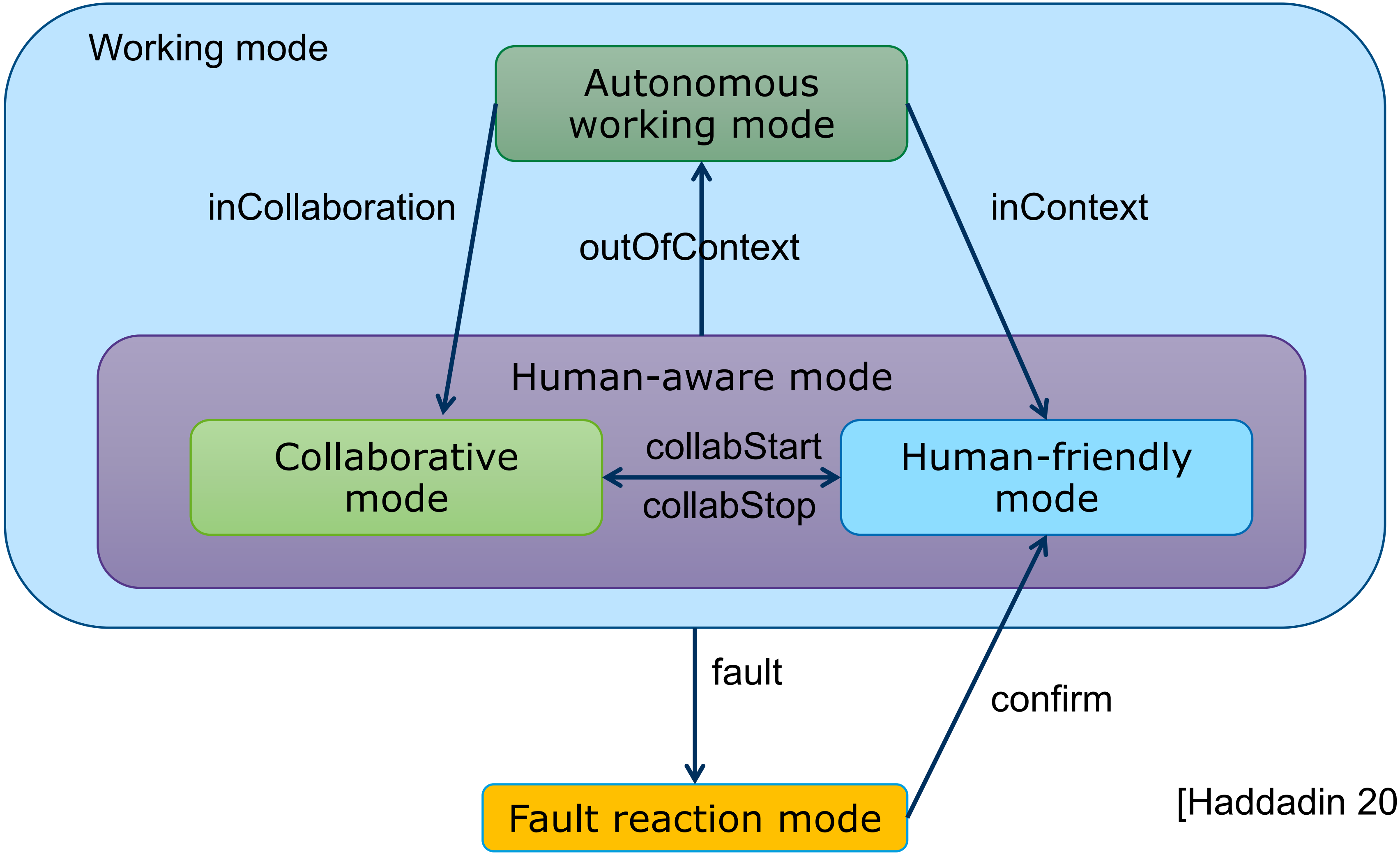
Do not fail an assembly line

Can easily be re-targeted

Smart Gloves in Smart Rooms
Our KUKA Case Study for the Hannover Fair

Example 1b: Designing Coworking Cells

Haddadin‘s Safety Automaton for Cobots



Working mode

Autonomous working mode

inCollaboration

outOfContext

inContext

Human-aware mode

Collaborative mode

collabStart

collabStop

Human-friendly mode

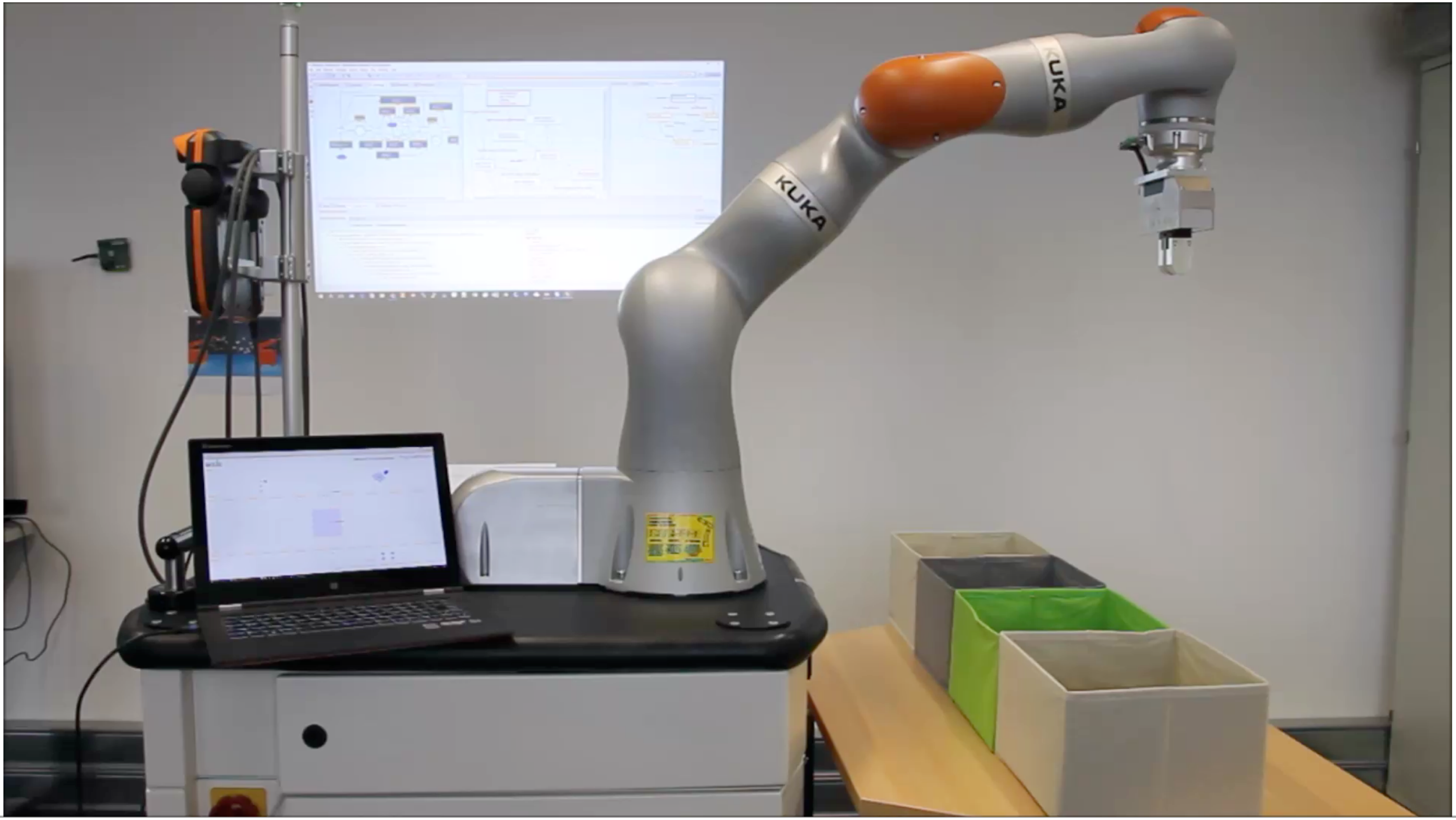
fault

confirm

Fault reaction mode

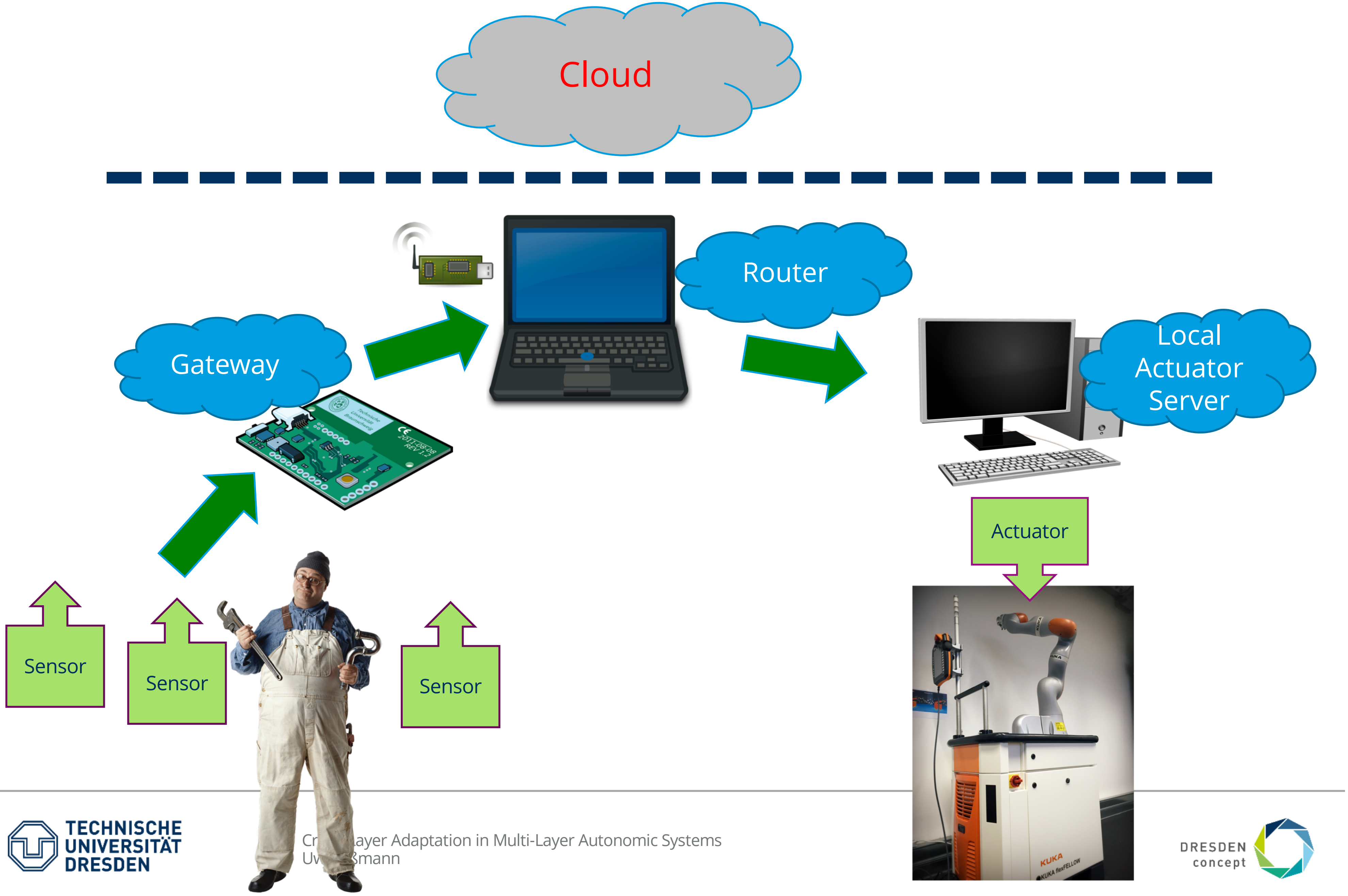
[Haddadin 2009]

Co-Working Cell „Cinderella“; Demo with MATE



Co-Working Cell „Cinderella“

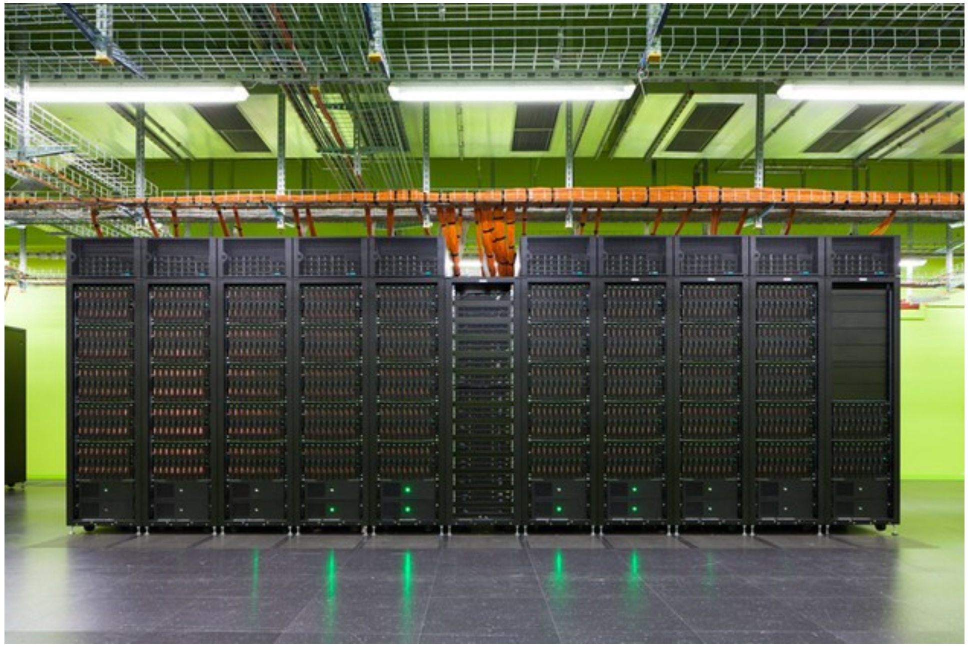
The Smart Environment of Robotic Coworking



How can we organize the smart environments of robotic coworking?

1.2. Example 2: Highly-Adaptive Energy-Efficient Computing Servers

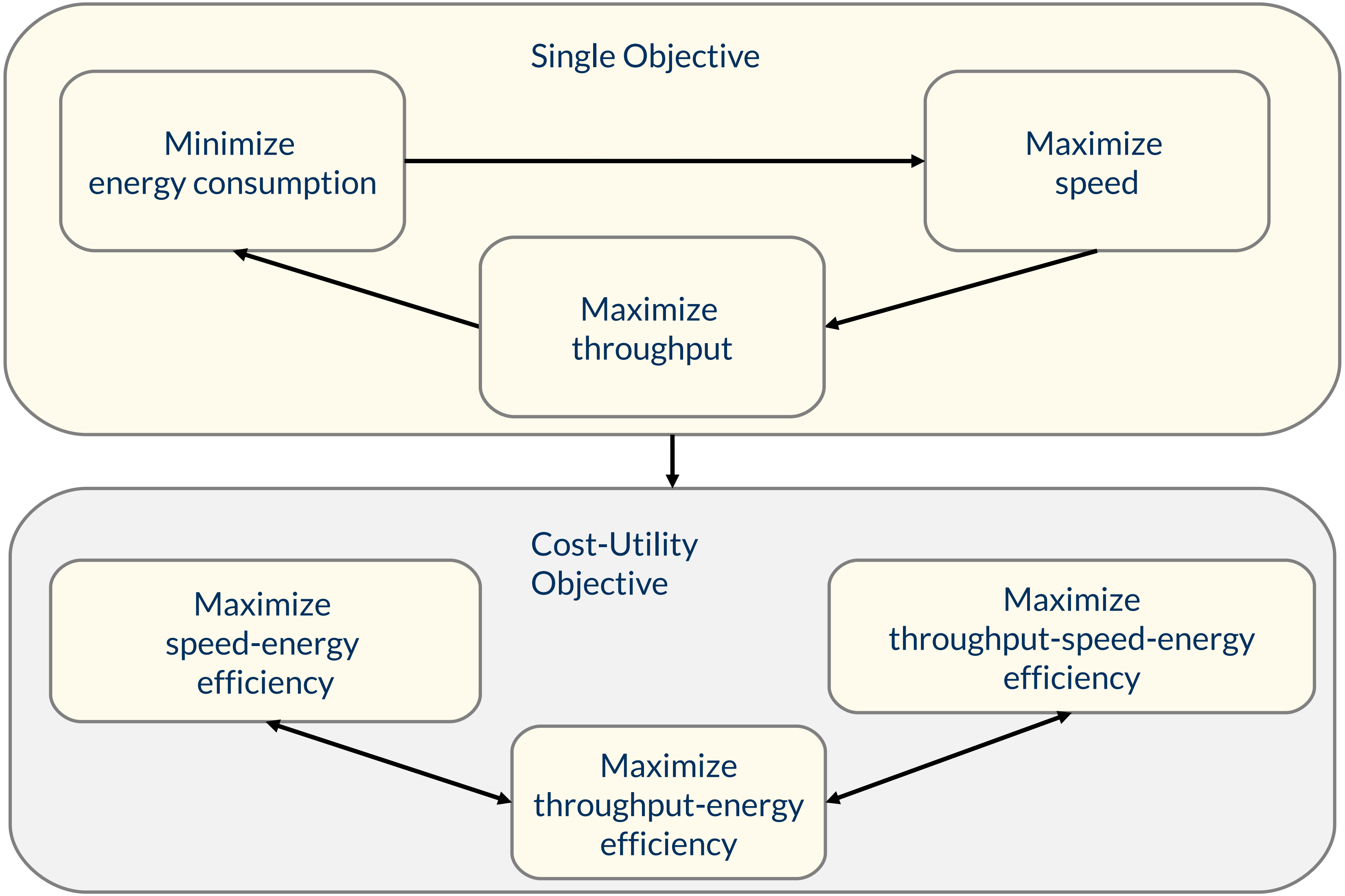
Natural-Energy Driven Processes



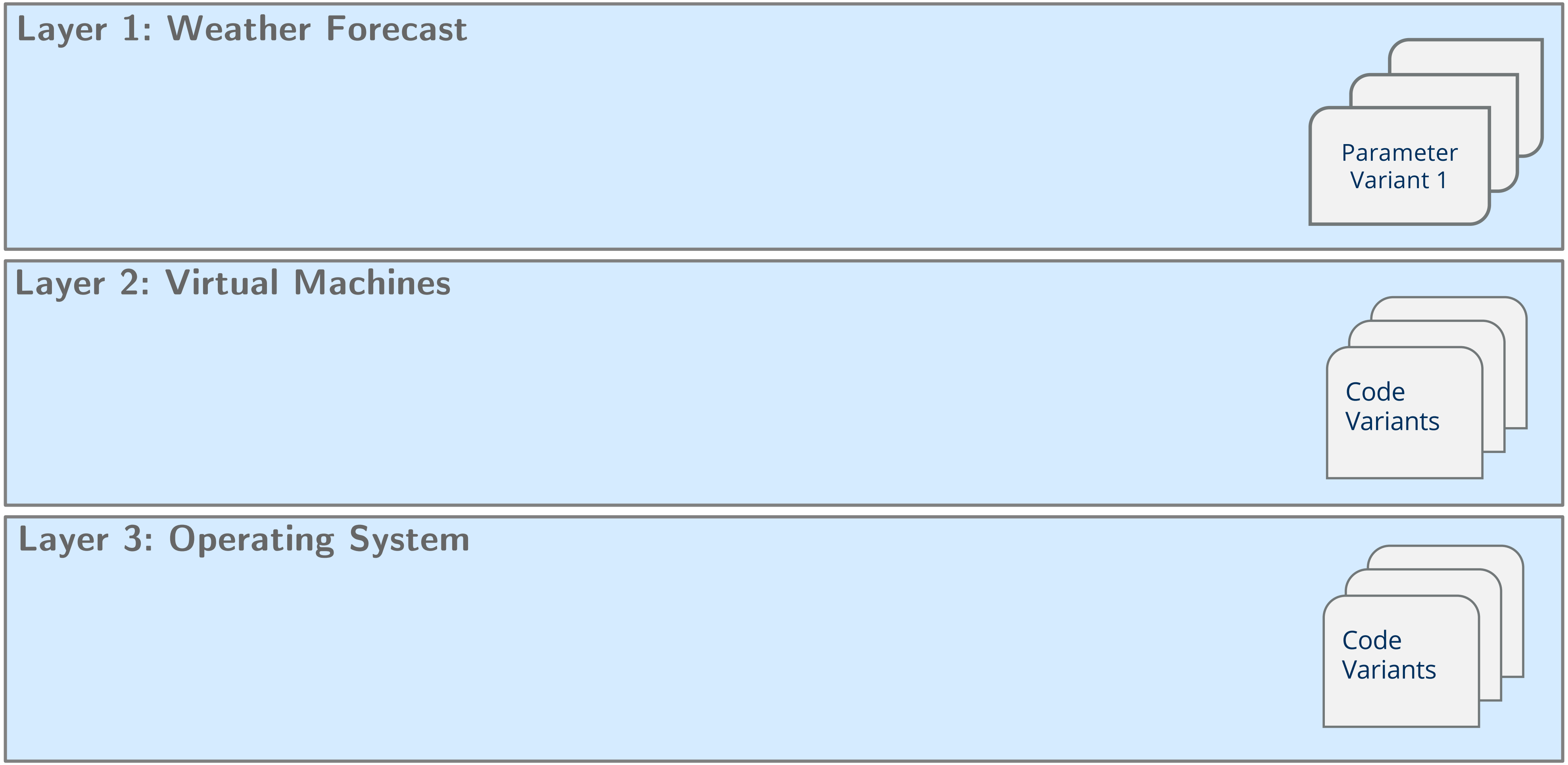
https://tu-dresden.de/zih/ressourcen/bilder/hrsk-ii_taurus_gommlich.jpg/@@images/b69d7257-b9bb-43c8-910a-fbe742ecddd5.jpeg

PD from www.pexels.com

An Adaptation Automaton for a Highly-Adaptive Energy-Efficient Server


Single Objective
Minimize energy consumption
Maximize speed
Maximize throughput
Cost-Utility Objective
Maximize speed-energy efficiency
Maximize throughput-speed-energy efficiency
Maximize throughput-energy efficiency

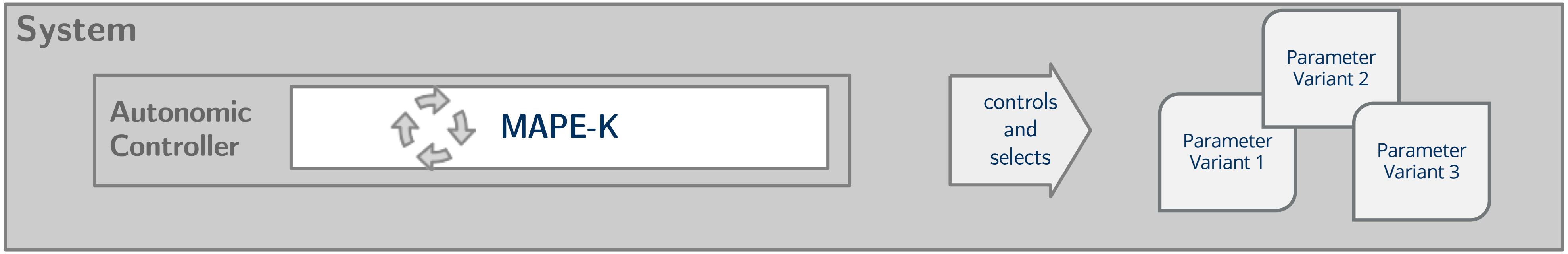
Complex Layer Structure


Layer 1: Weather Forecast
Parameter Variant 1
Layer 2: Virtual Machines
Code Variants
Layer 3: Operating System
Code Variants

How can we organize
the vast servers of the future
for energy-efficiency and adaptivity?

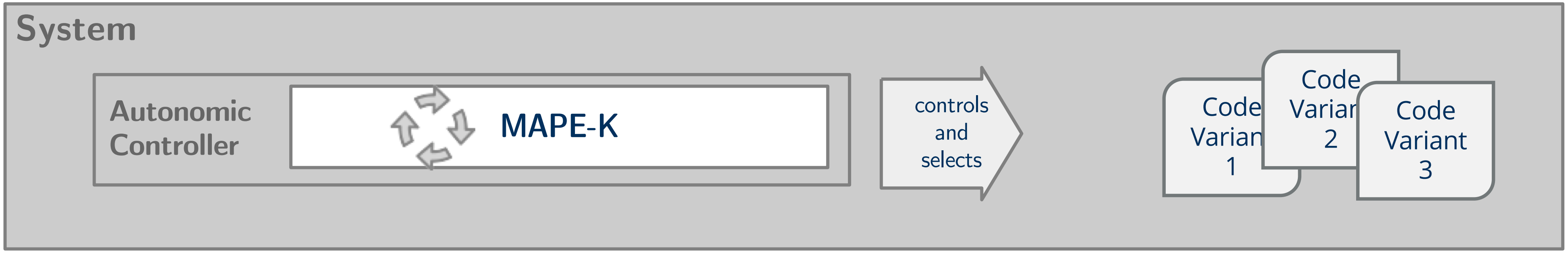
2. Simple Autonomic Systems and Autonomic Product Lines

Autonomic Systems

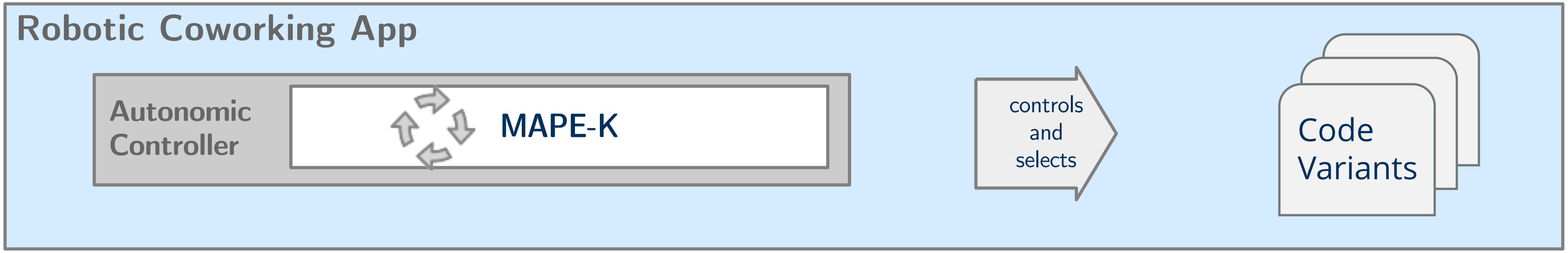


Autonomic Software Product Lines (ASPL)

[Abbas 2010]

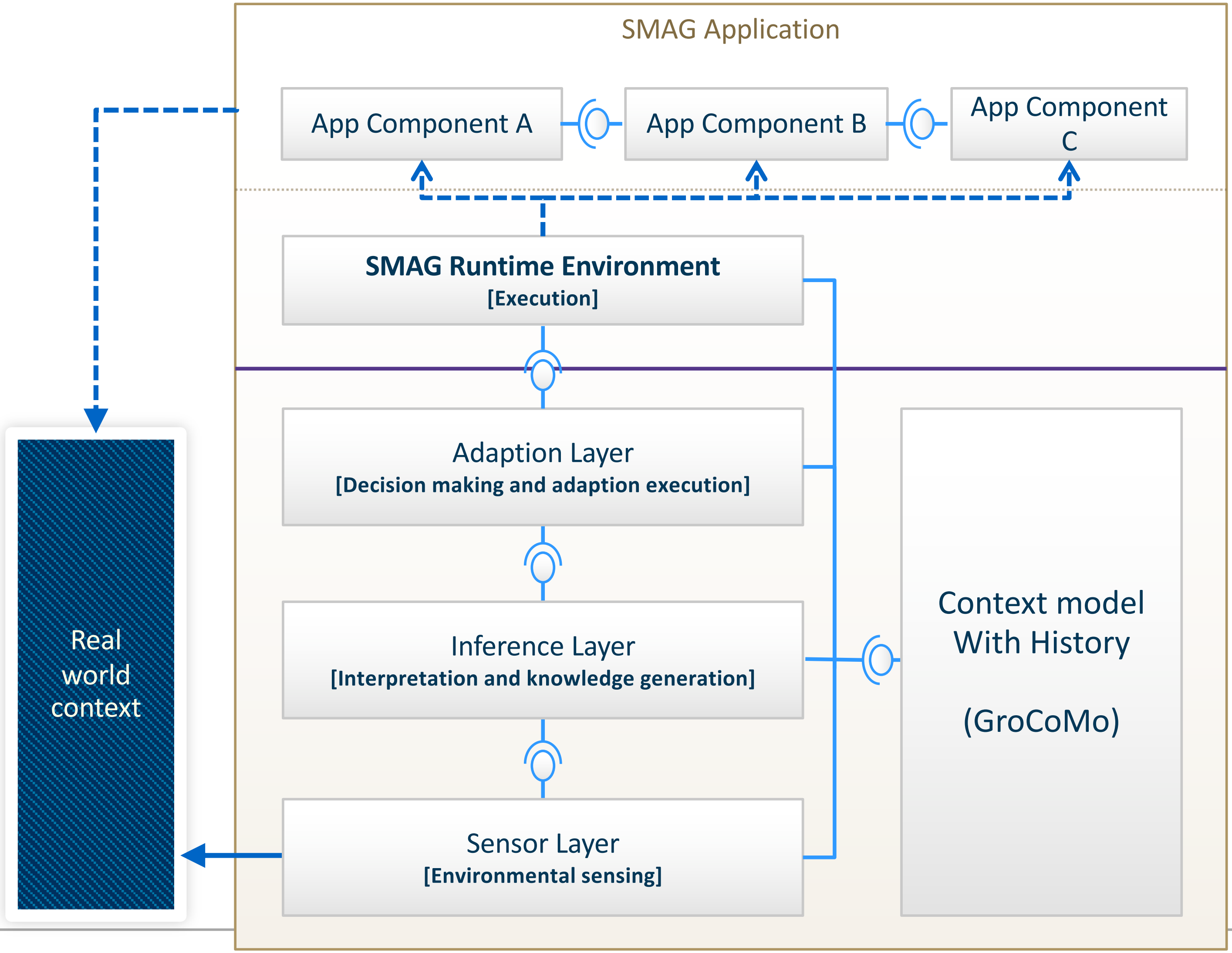


Coworking Apps are
Autonomic Software Product Lines (ASPL)



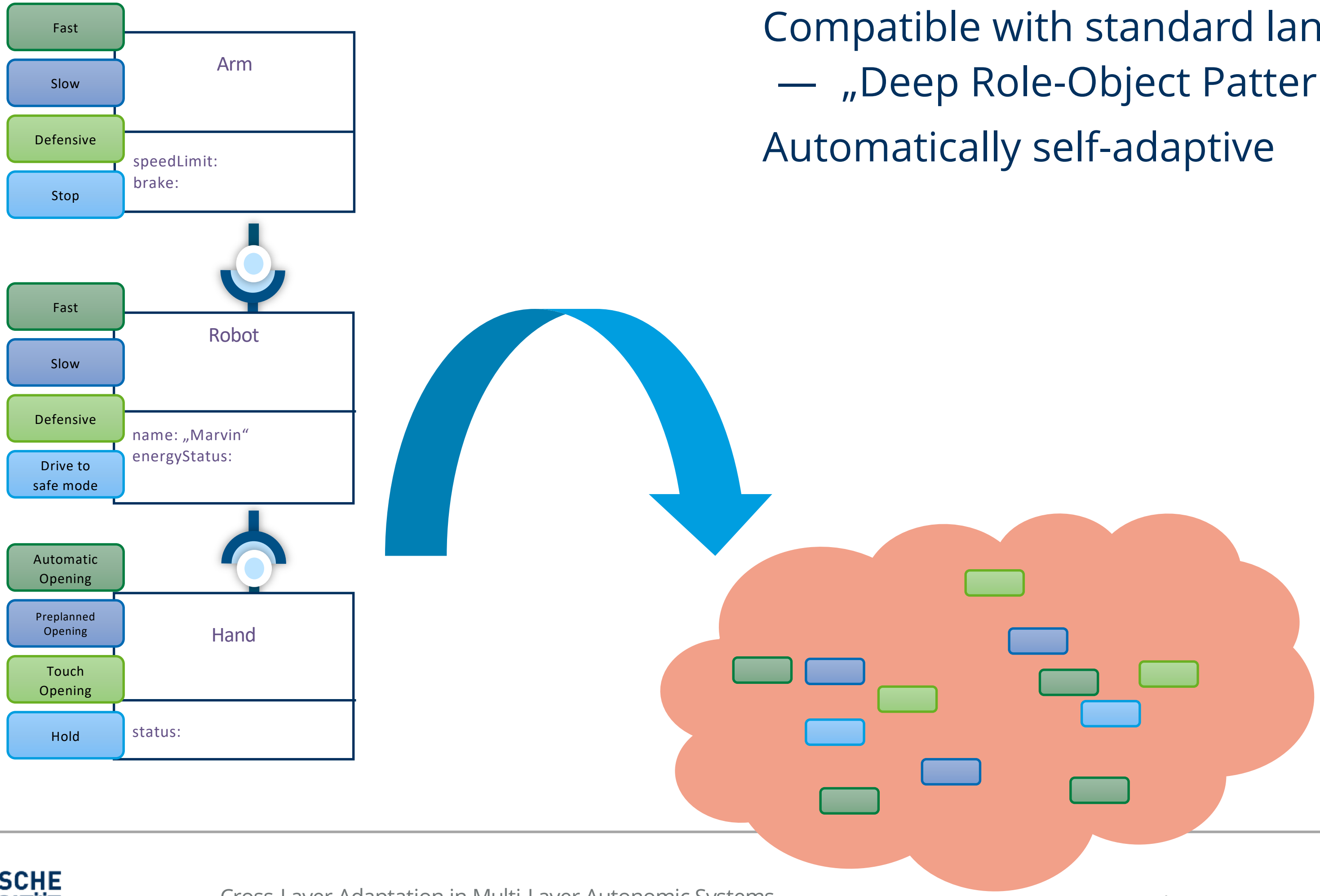
[Aßmann 2017]

Smart Applications on Smart App Grid Infrastructure (SMAGS) for ASPL



[http://st.inf.tu-dresden.de/smags]

SMAGS: Self-Adaptive Architecture Language with Classic Code Generation



Code generation for several platforms possible

Compatible with standard languages

- „Deep Role-Object Pattern“

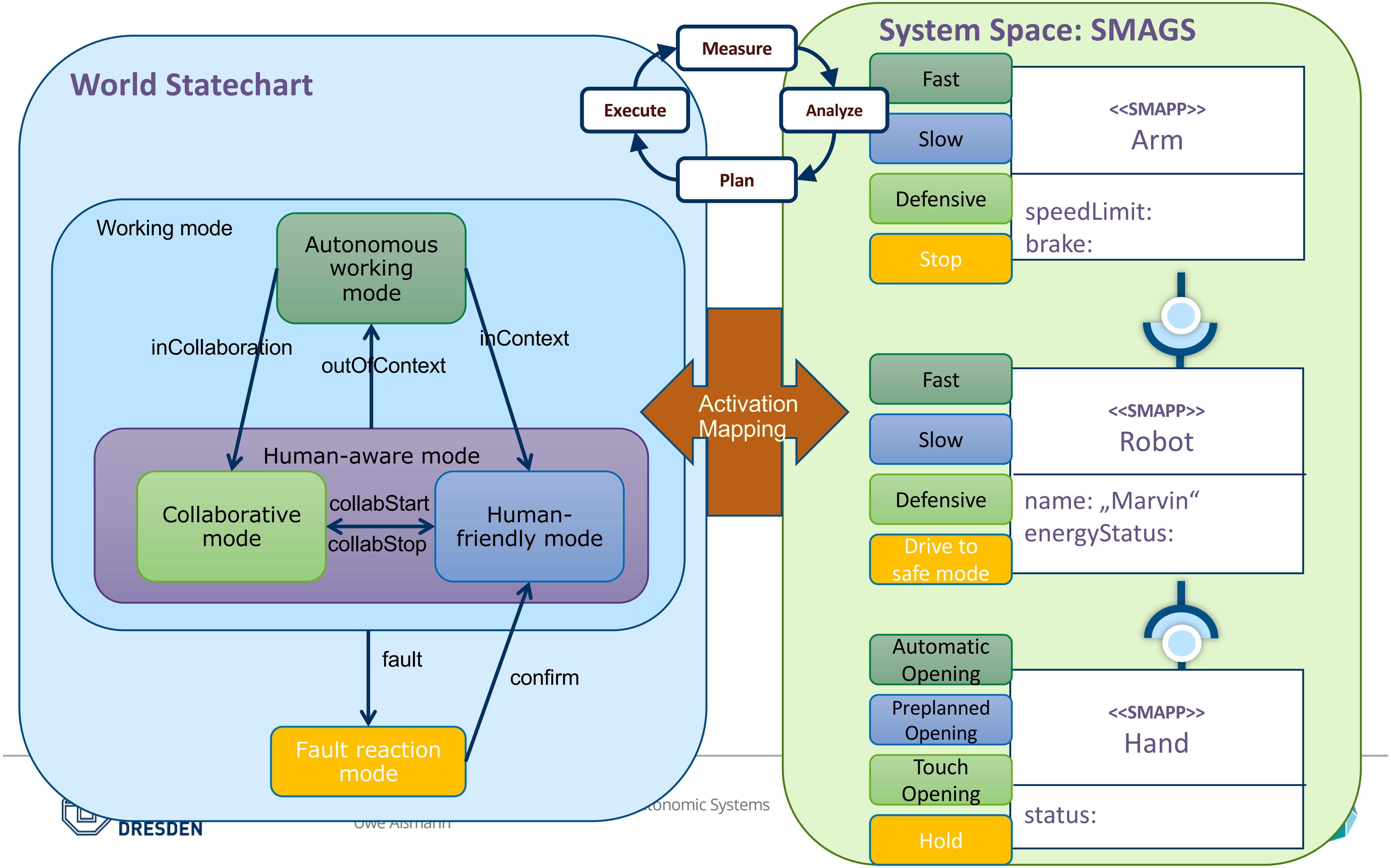
Automatically self-adaptive

[http://st.inf.tu-dresden.de/smags]

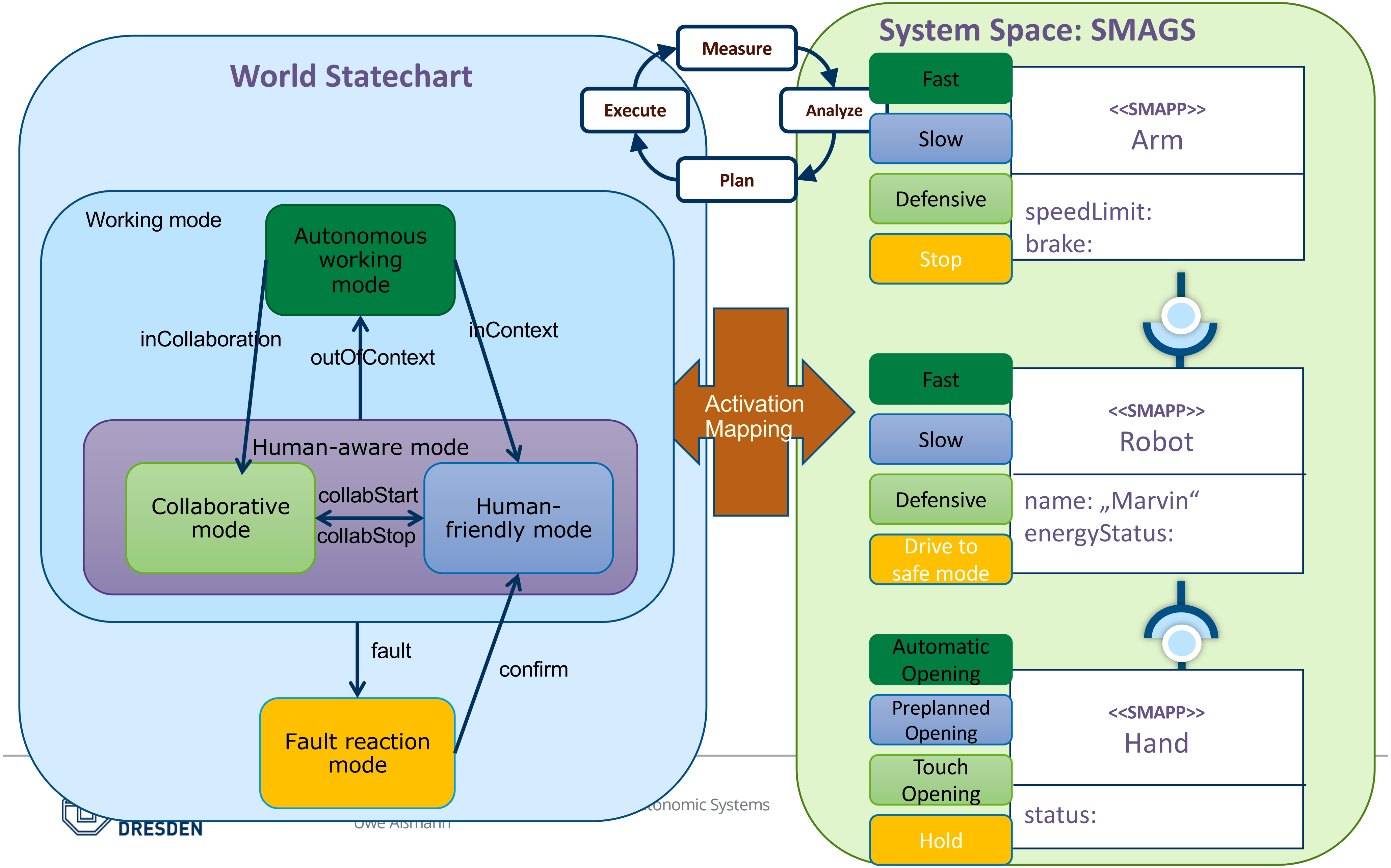
2.1. A Single-Layer ASPL for Coworking

[Aßmann 2017]

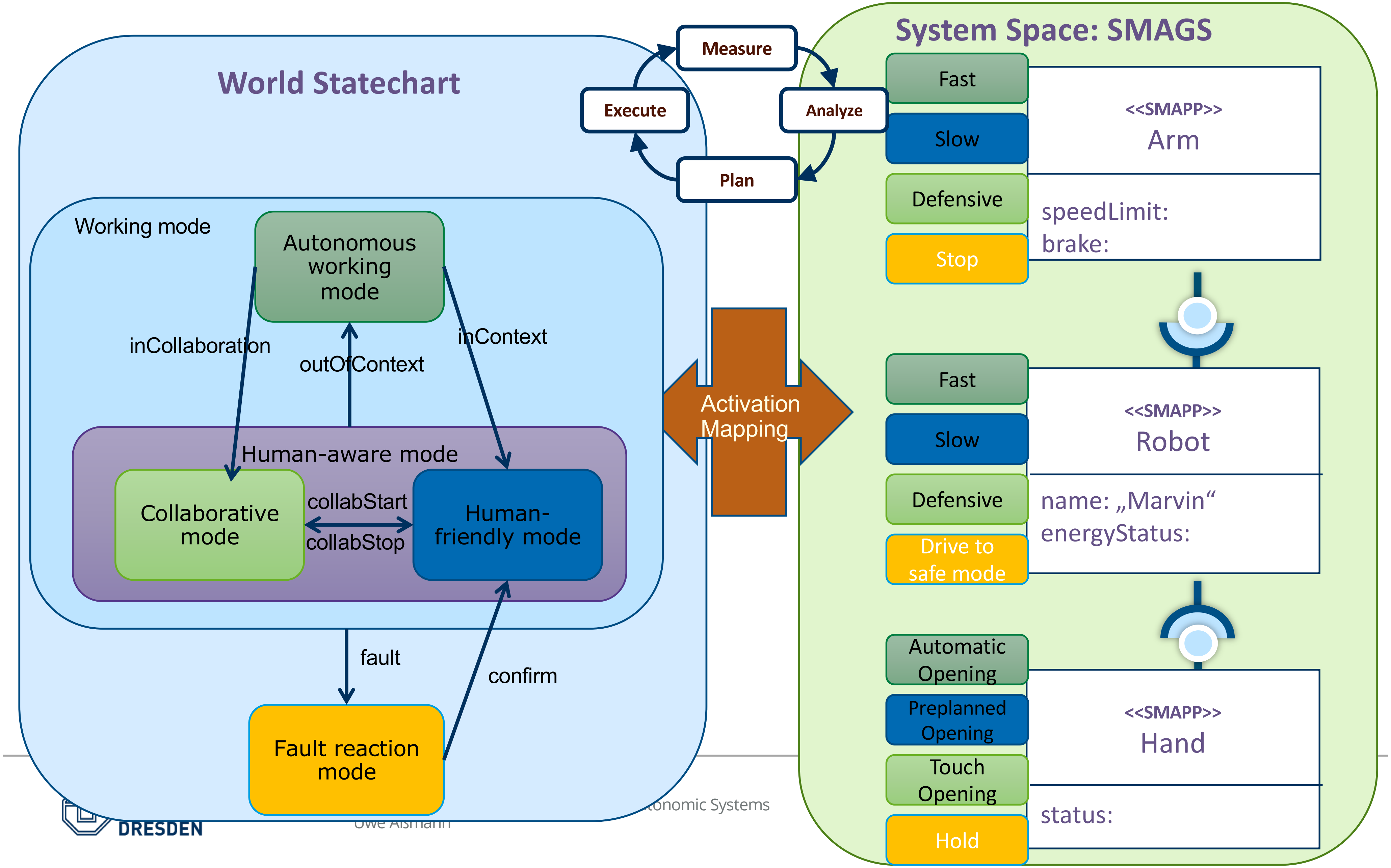
Open Smart Space = World Statechart + SMAGS



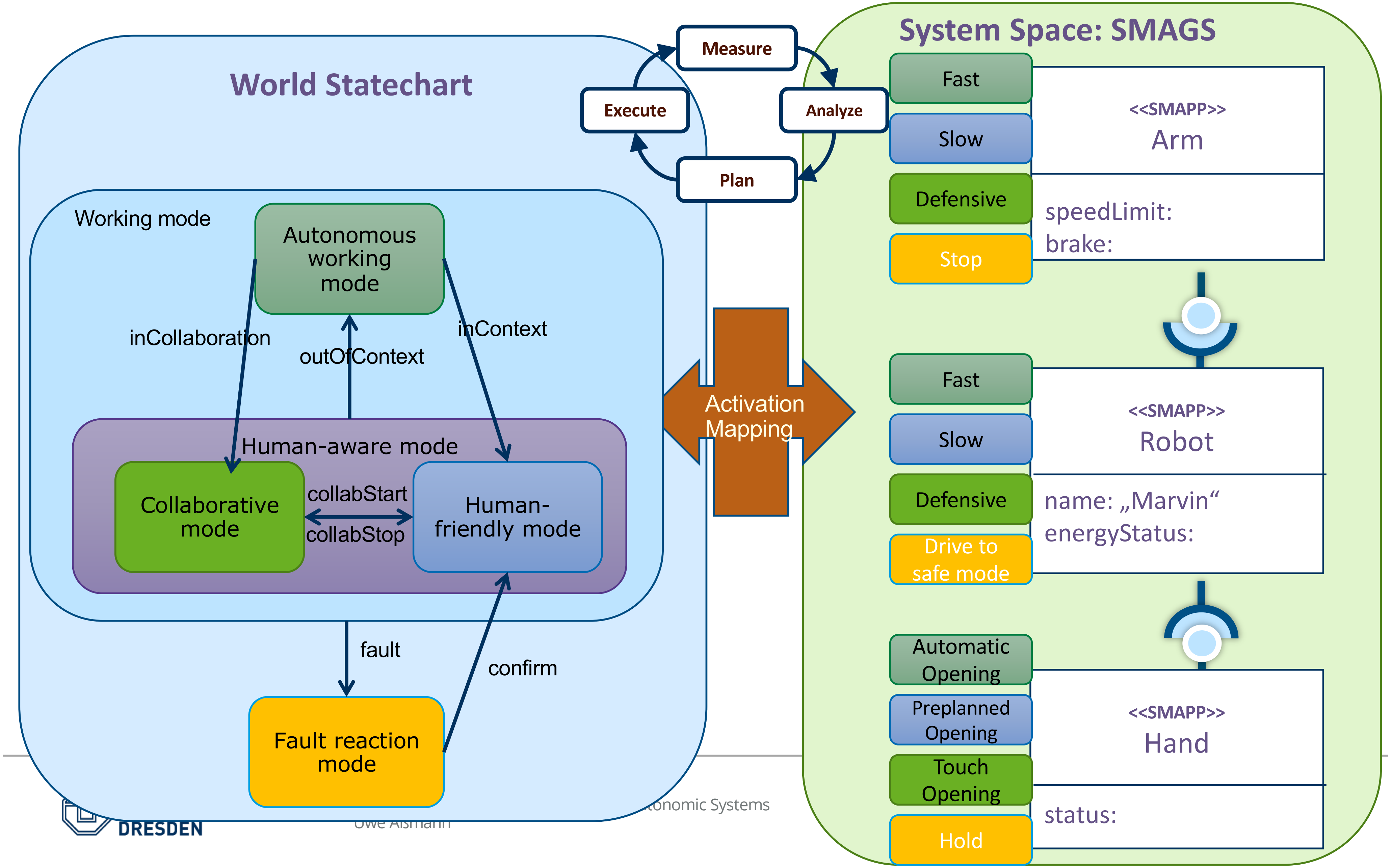
Open Smart Space in Co-Worker Mode “Autonomous”



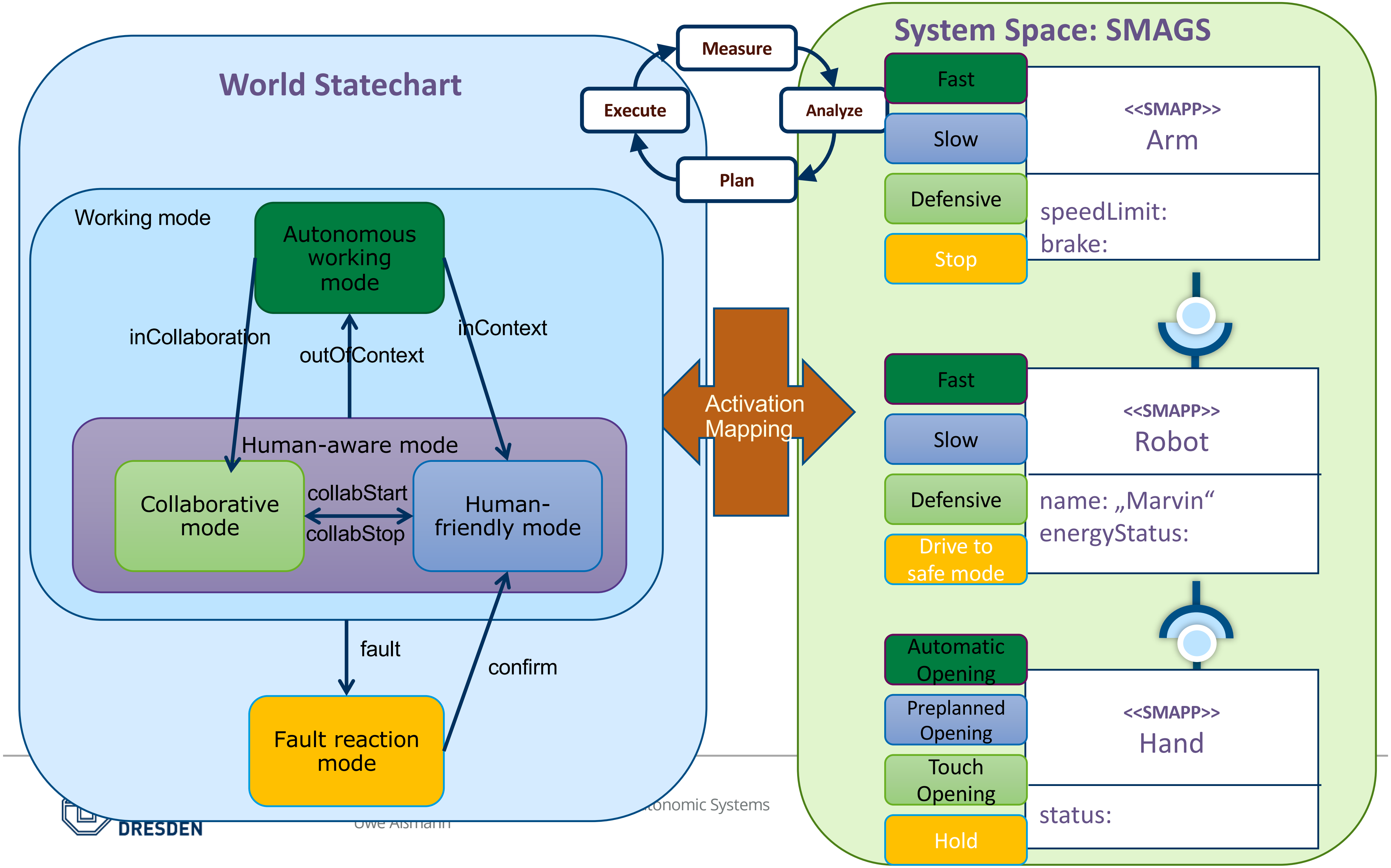
Open Smart Space in Co-Worker Mode “Human-Friendly”



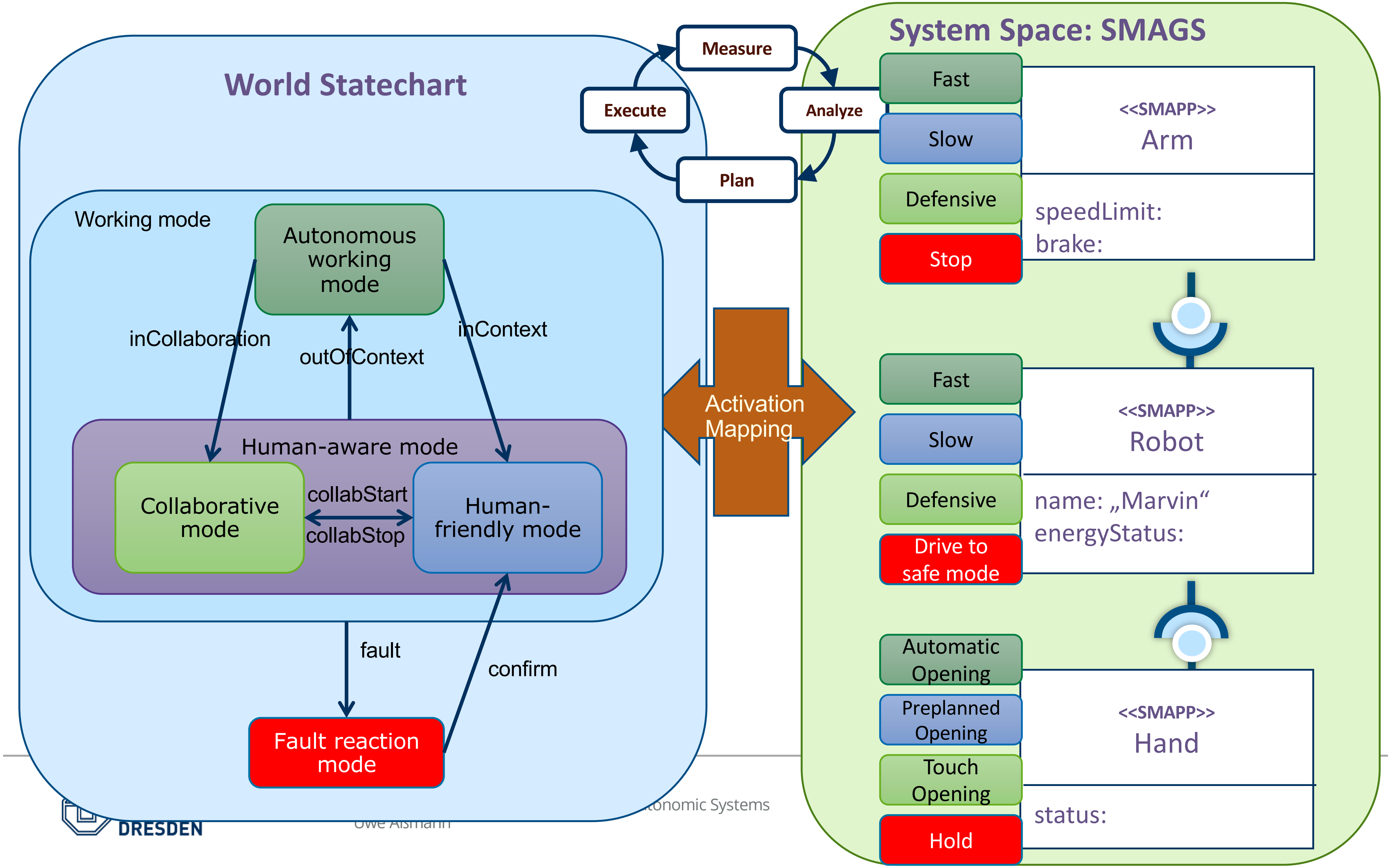
Open Smart Space in Co-Worker Mode “Collaborative”



Open Smart Space in Co-Worker Mode “Autonomous”



Open Smart Space in Co-Worker Mode “Fault Reaction”

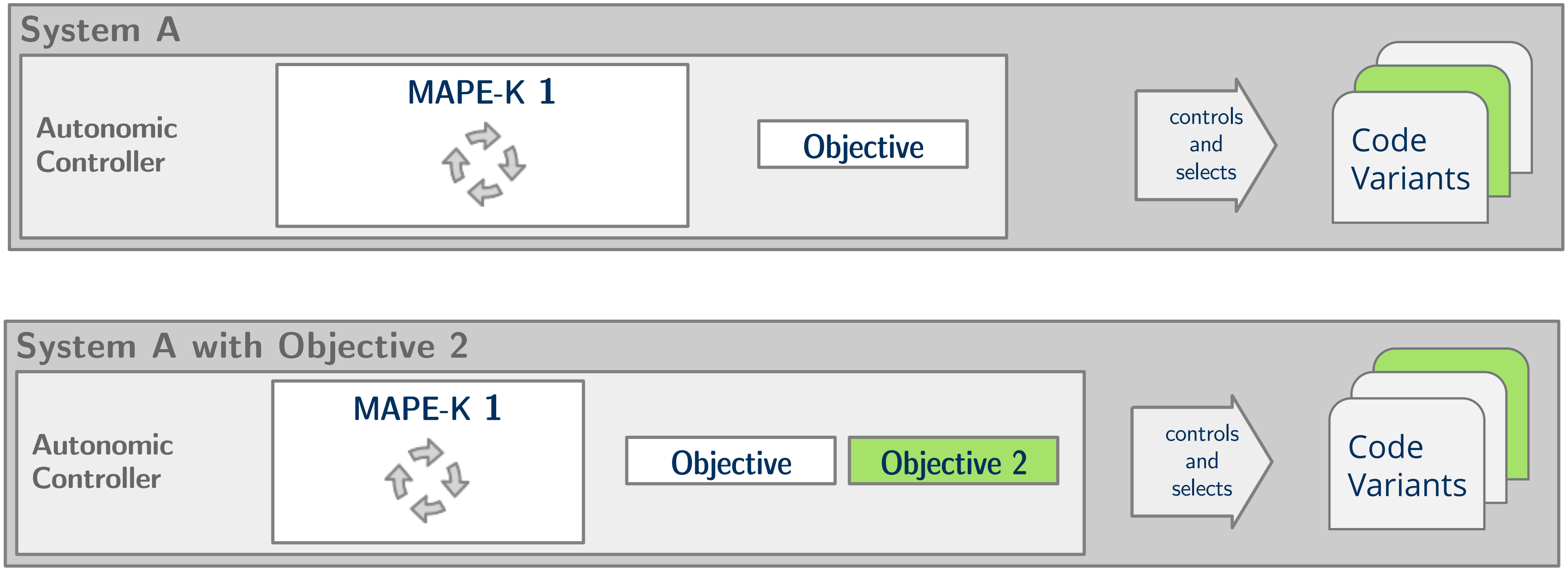


TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

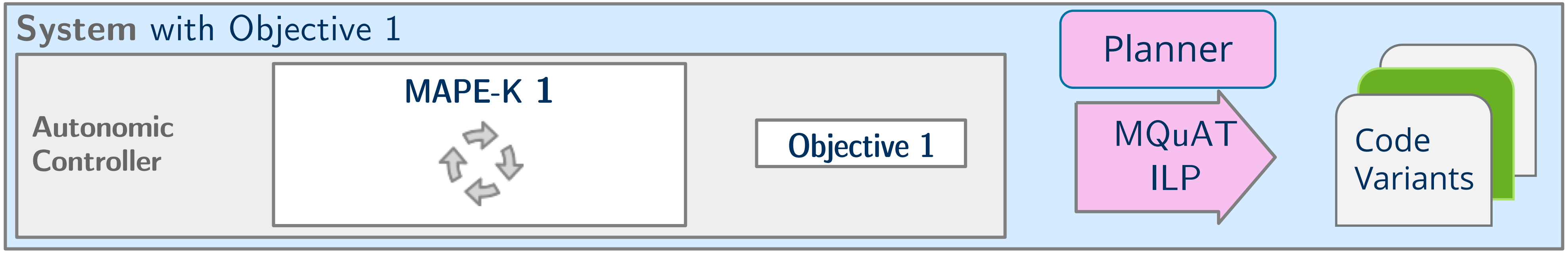
2.2 Single-Layer Self-Optimizing Systems

Self-Optimizing Software Product Lines (OSPL)



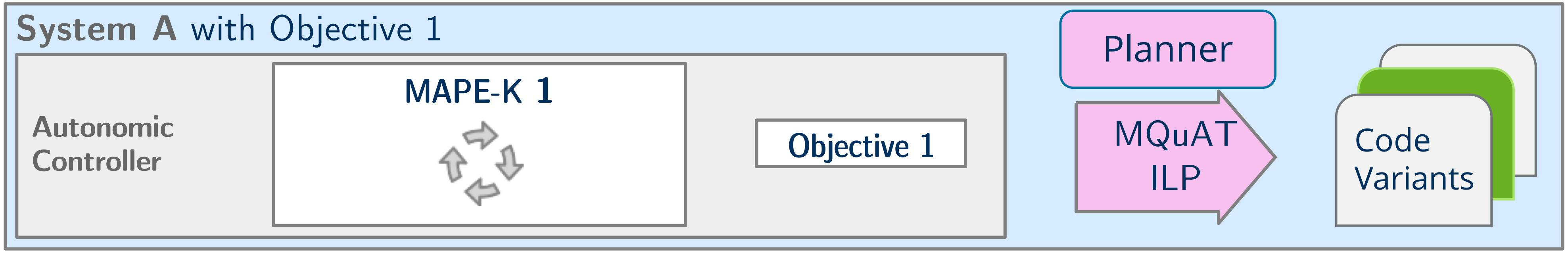
Multi-Quality Auto-Tuning (MQuAT)

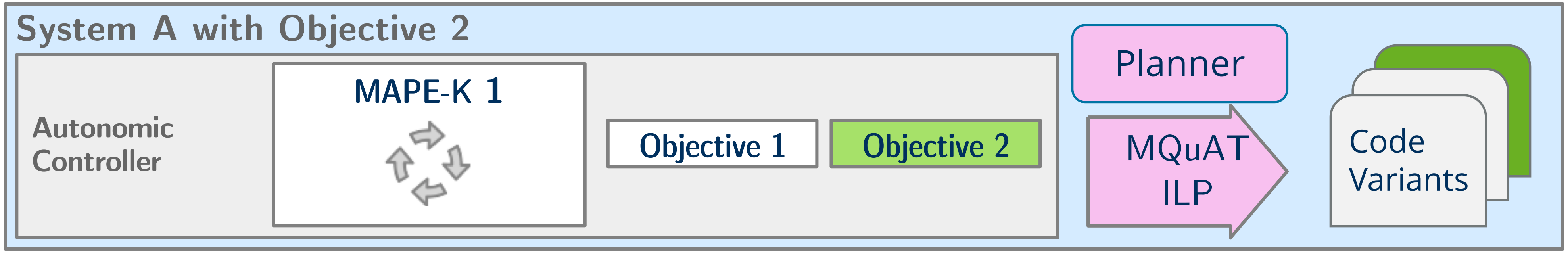
[Götz 2013]



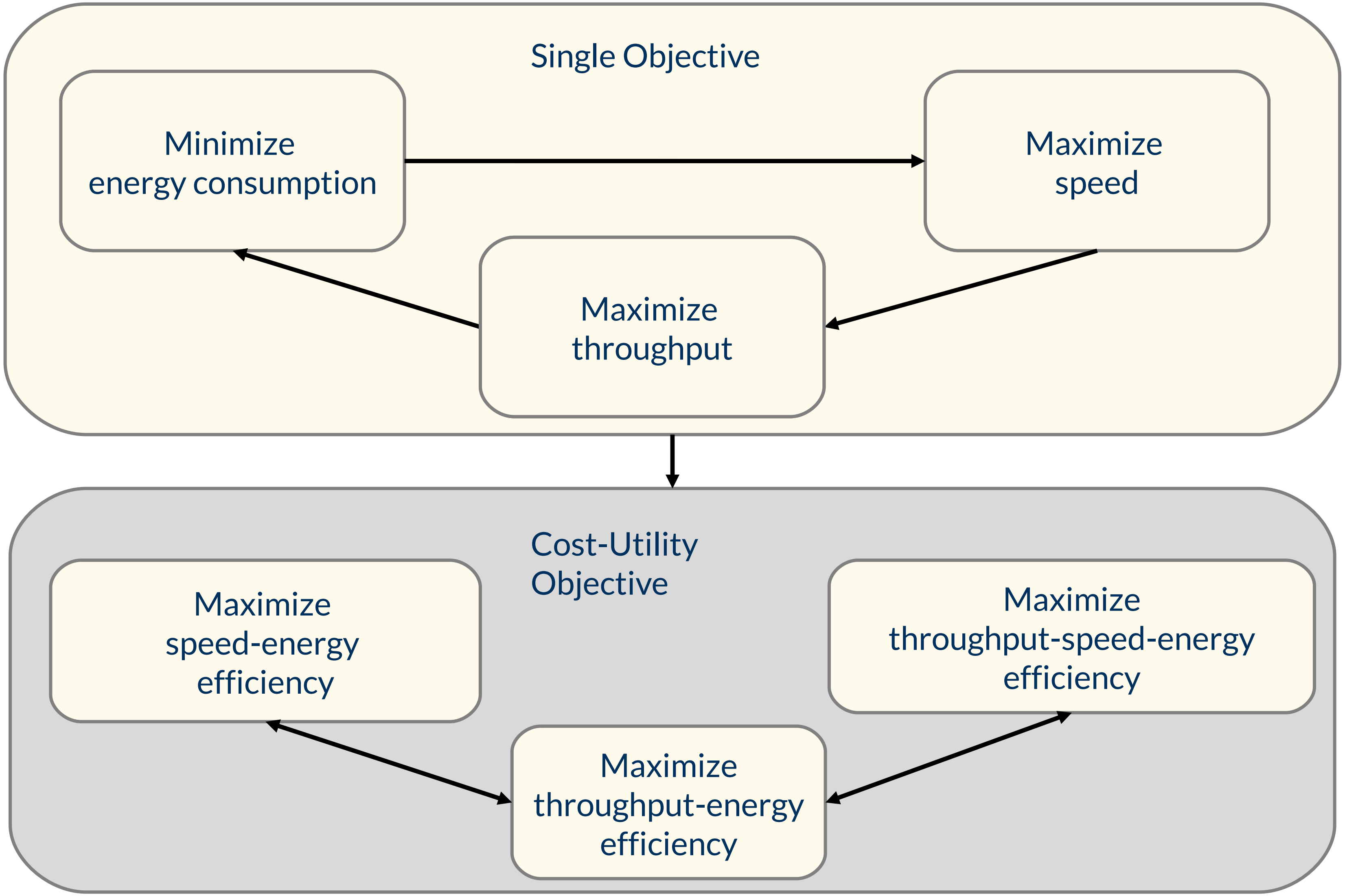
- Strategy
  - resource allocation **for pack-and-switch-off decisions**

Multi-Quality Auto-Tuning (MQuAT)





MQuAT Can Change Objectives for a Highly-Adaptive Energy-Efficient Server


Single Objective
Minimize energy consumption
Maximize speed
Maximize throughput
Cost-Utility Objective
Maximize speed-energy efficiency
Maximize throughput-speed-energy efficiency
Maximize throughput-energy efficiency

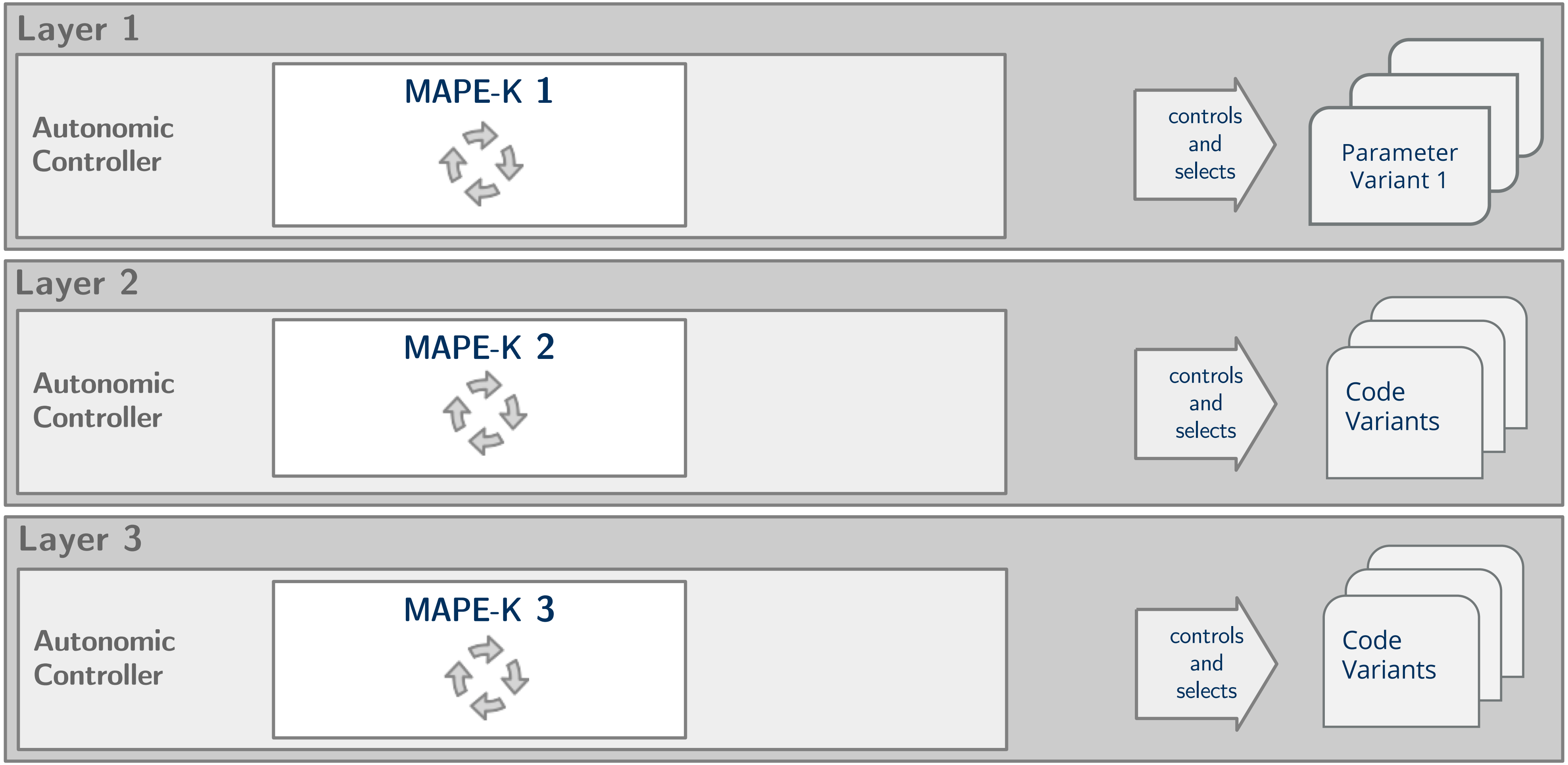
Multi-Quality Auto-Tuning (MQuAT)

- uses ILP systems for self-optimization
- determines the optimal configuration of a system
- with regard to an objective function

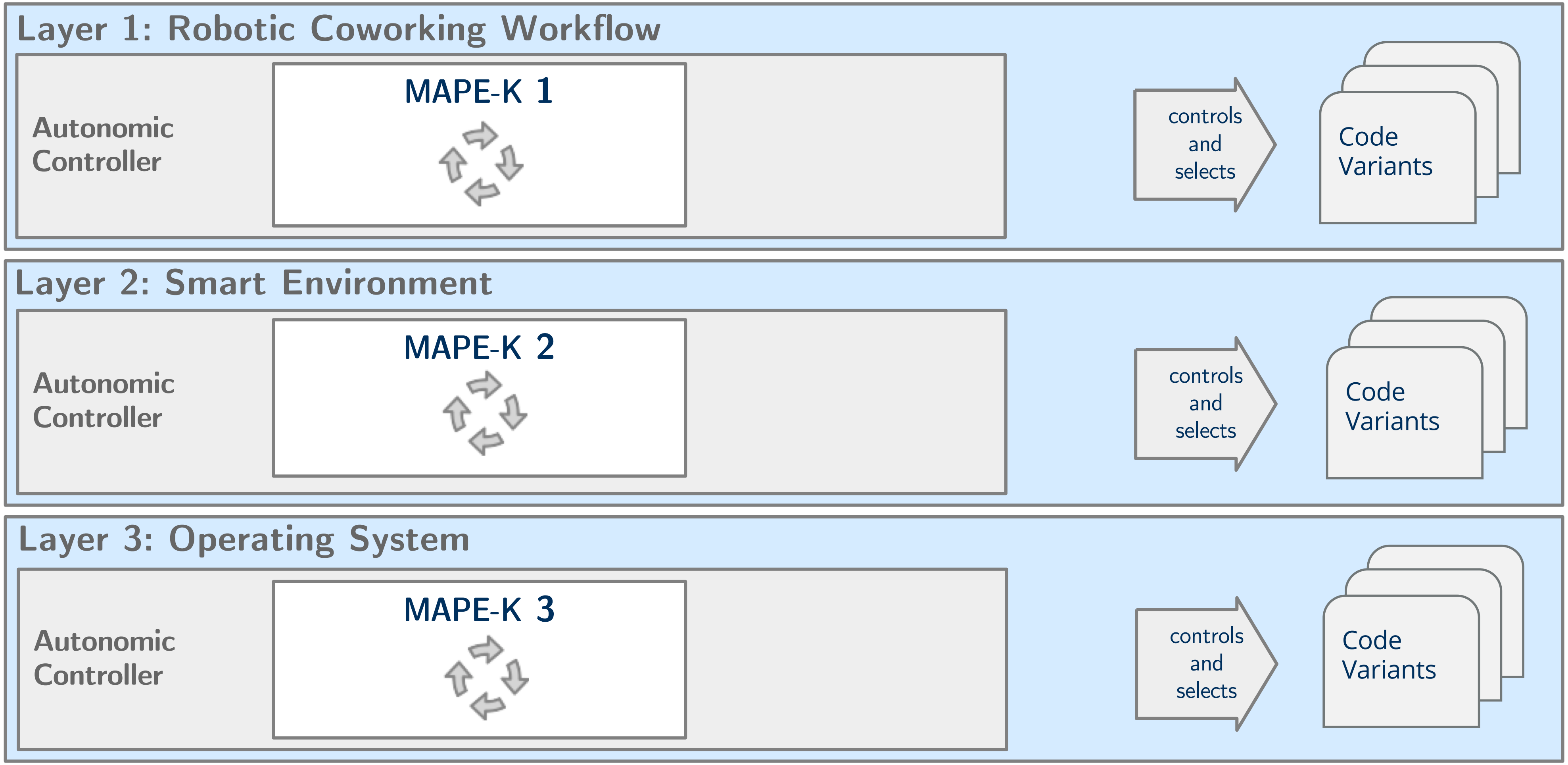
- Change the objective function to get different reconfiguration planning

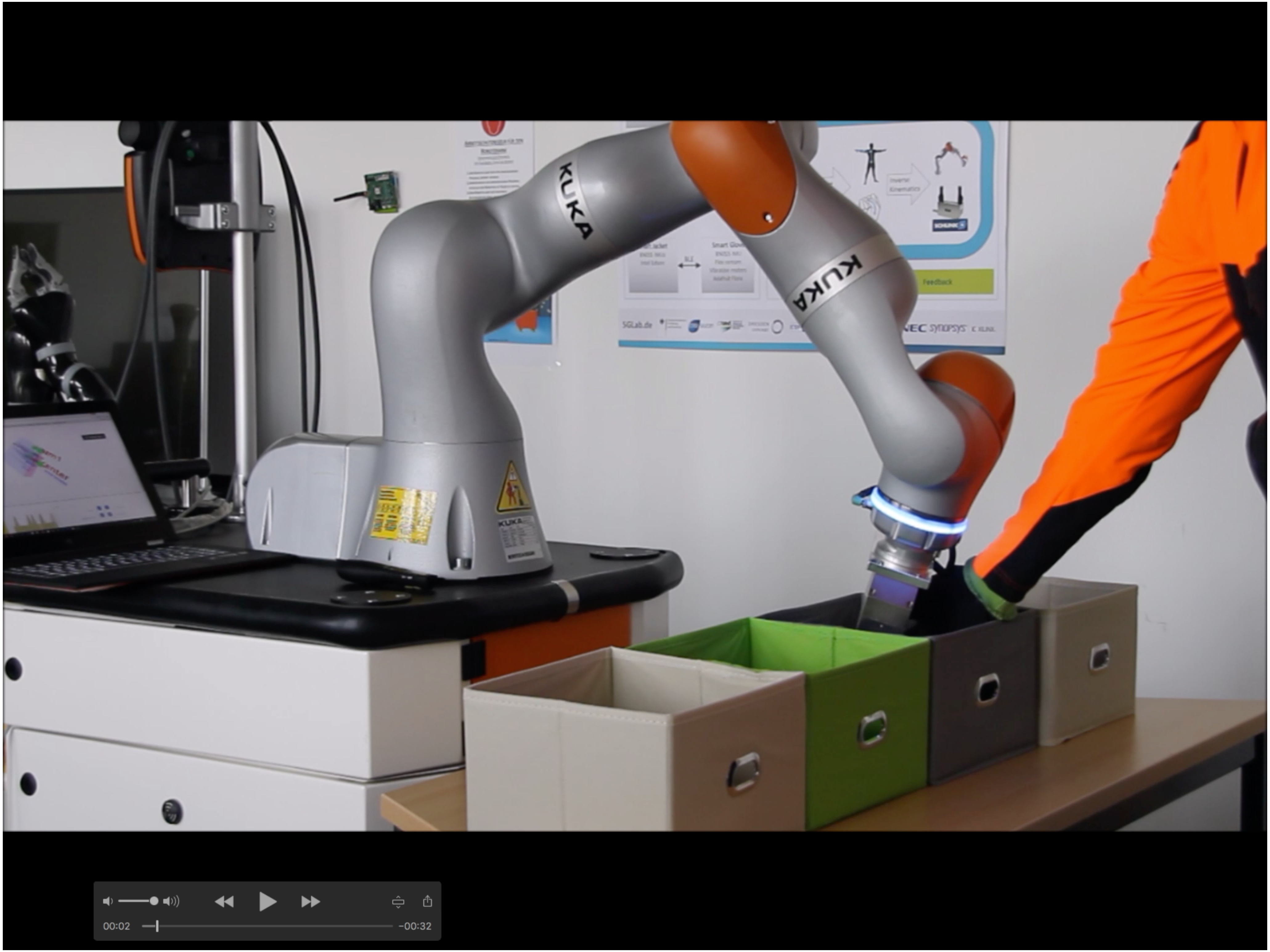
3. How to Tame Multi-Layer Autonomic Systems

Multi-Layer Autonomic System (MuLAS)

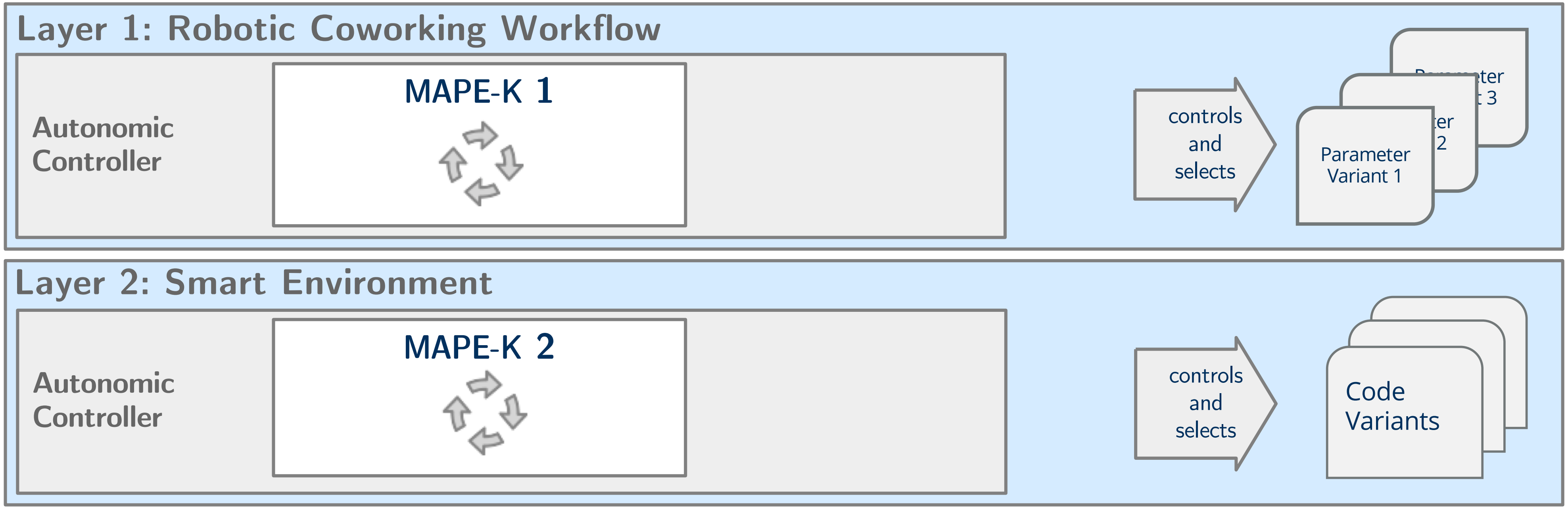


Example: Robotic Coworking Smart Room (MuLAS)

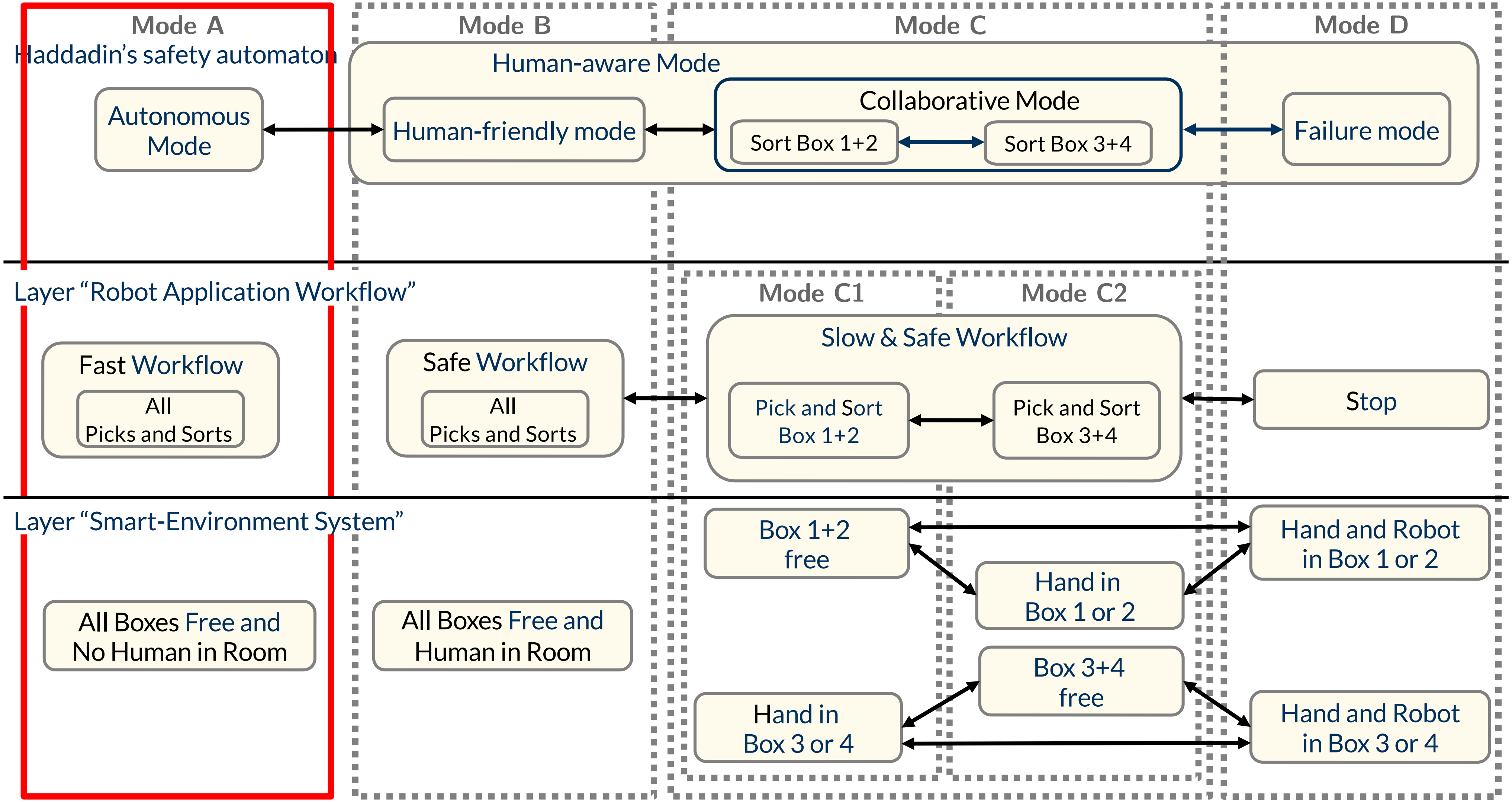



KUKA
KUKA
Feedback
00:02
-00:32

Example: Cinderella Case Study



Cinderella Mode A is Active



Mode A

Haddadin's safety automaton

Autonomous Mode

Mode B

Mode C

Mode D

Human-aware Mode

Human-friendly mode

Collaborative Mode

Sort Box 1+2

Sort Box 3+4

Failure mode

Layer "Robot Application Workflow"

Mode C1

Mode C2

Fast Workflow

All Picks and Sorts

Safe Workflow

All Picks and Sorts

Slow & Safe Workflow

Pick and Sort Box 1+2

Pick and Sort Box 3+4

Stop

Layer "Smart-Environment System"

All Boxes Free and No Human in Room

All Boxes Free and Human in Room

Box 1+2 free

Hand and Robot in Box 1 or 2

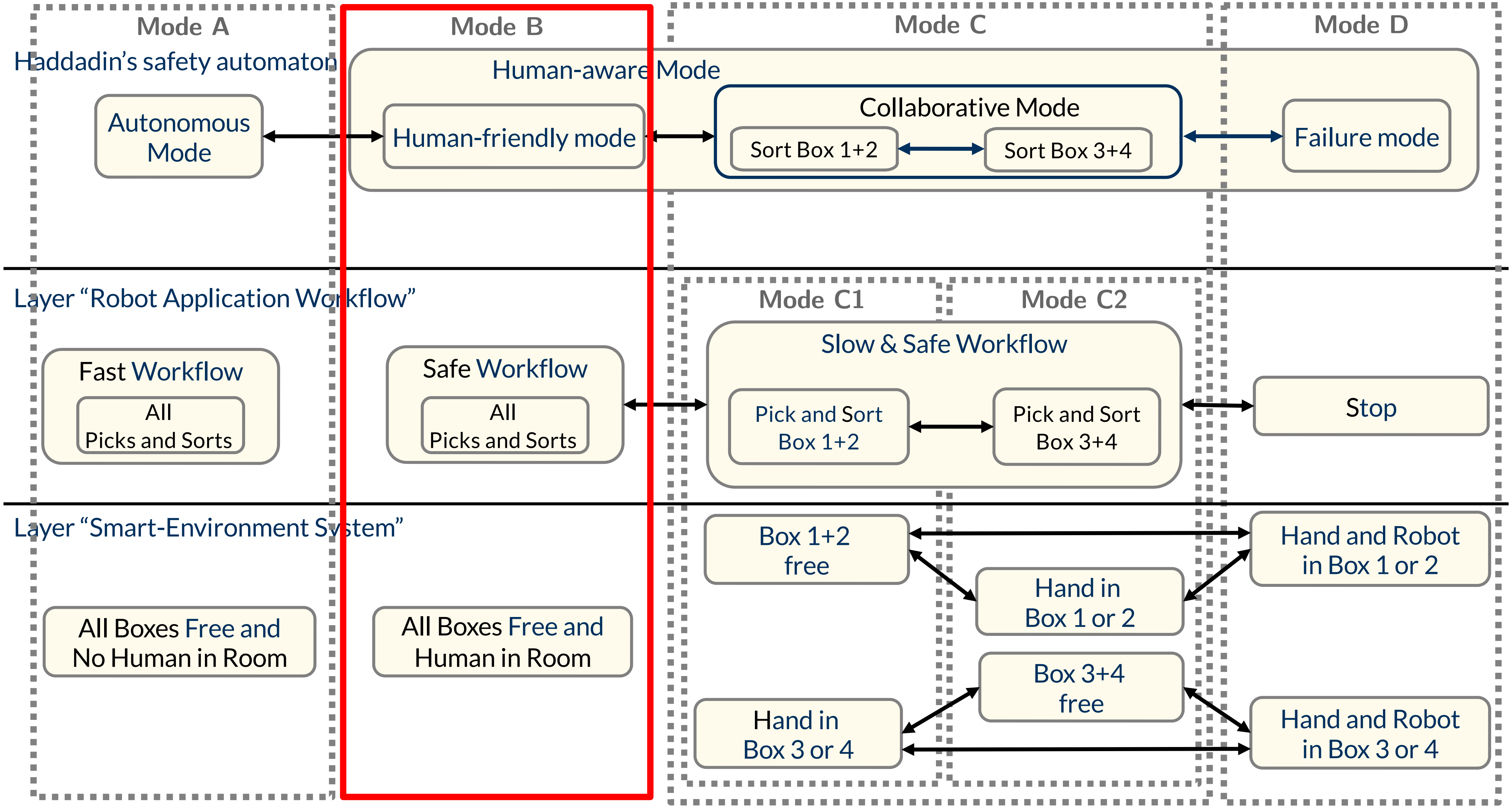
Hand in Box 1 or 2

Box 3+4 free

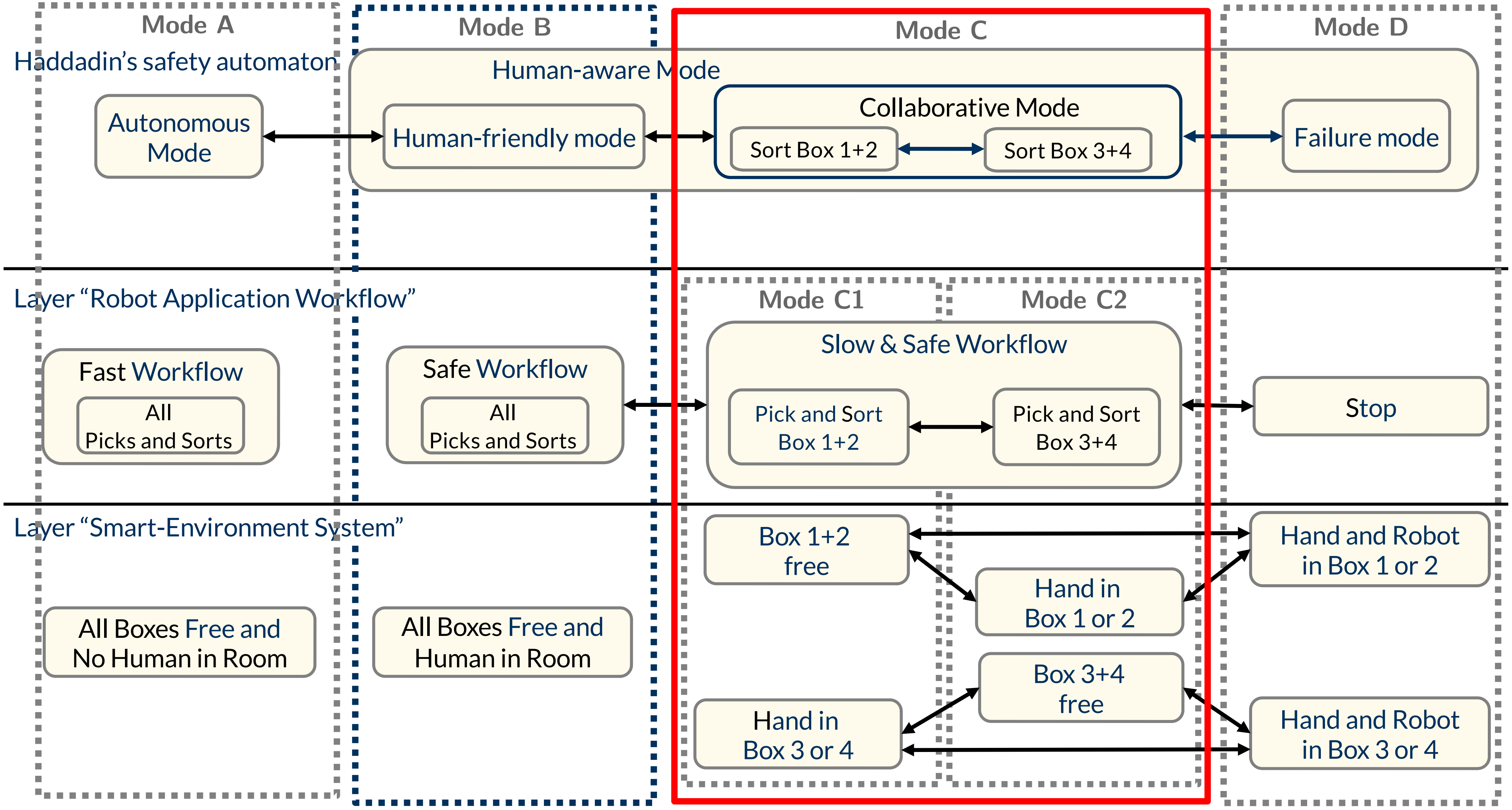
Hand in Box 3 or 4

Hand and Robot in Box 3 or 4

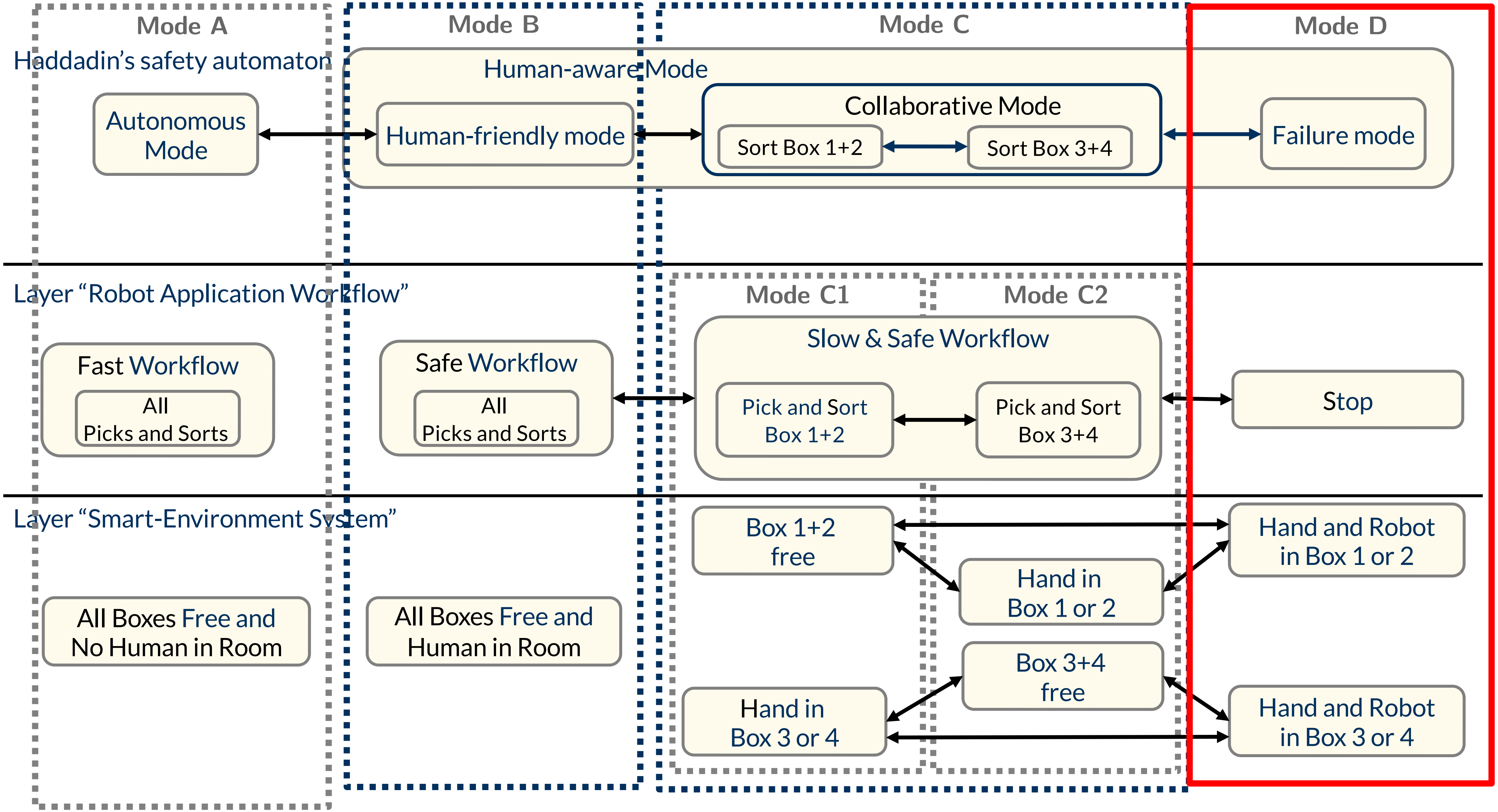
Cinderella Mode B is Active



Cinderella Mode C is Active



Cinderella Mode D is Active



Lean Robotics

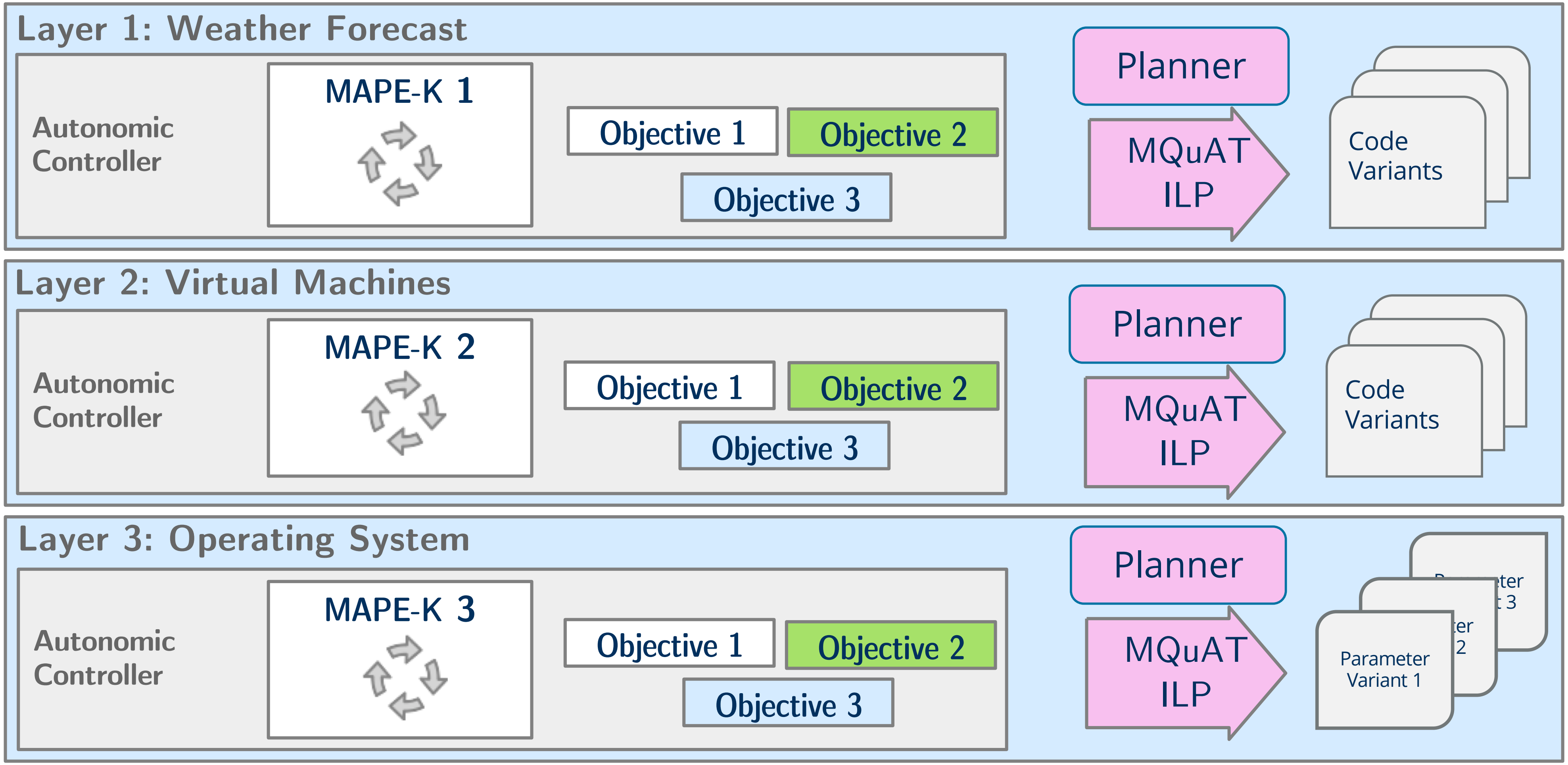
Robotic coworking needs specifically designed **robotic coworking cells**

[Lean Robotics, Bouchard 2017]

Every future robotic coworking cell needs a MuLAS

3.2 How to Tame Multi-Layer Optimizing Systems (MuLOS)

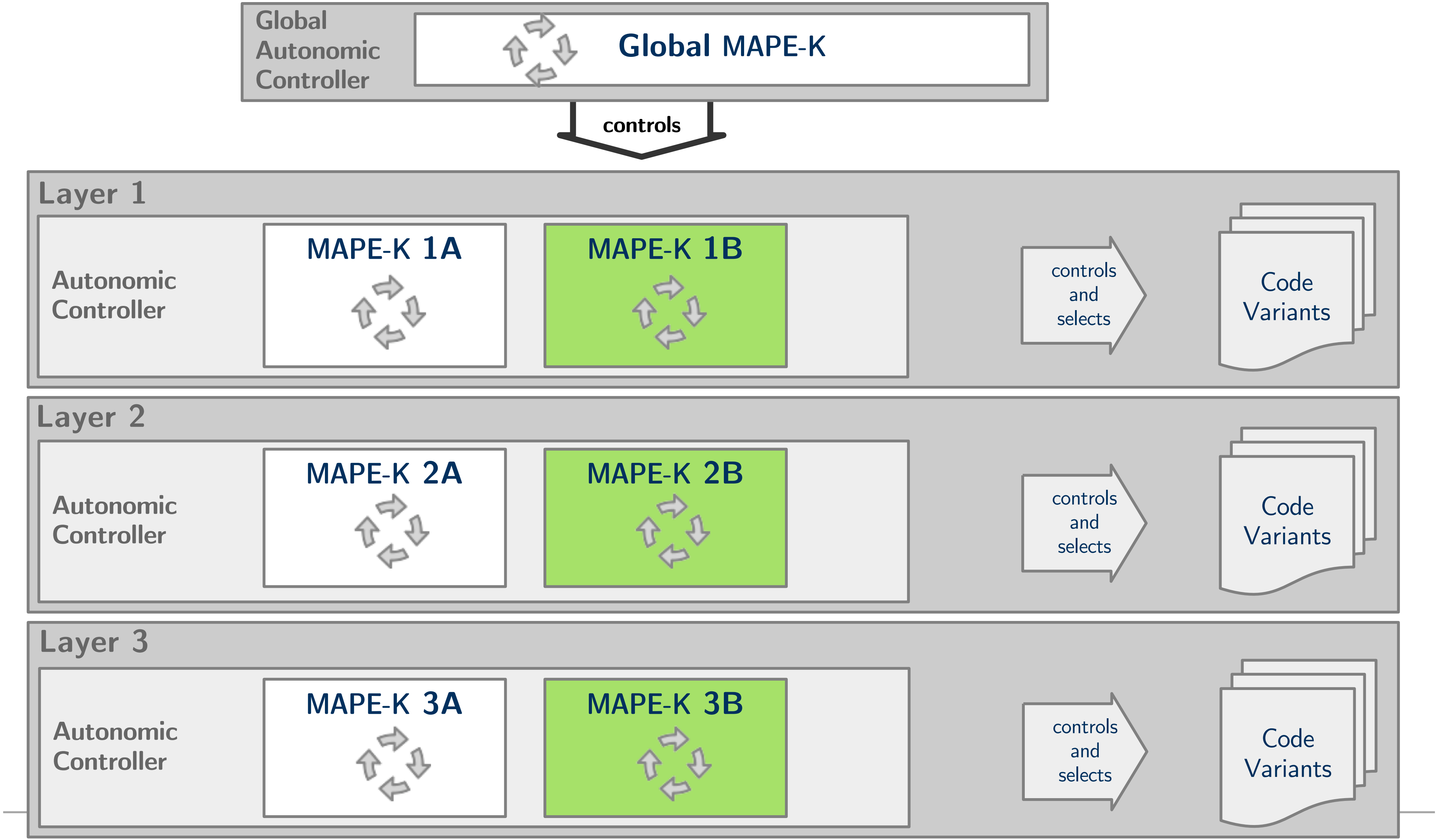
Example: Energy-Adaptive Multi-Layer Self-Optimizing System



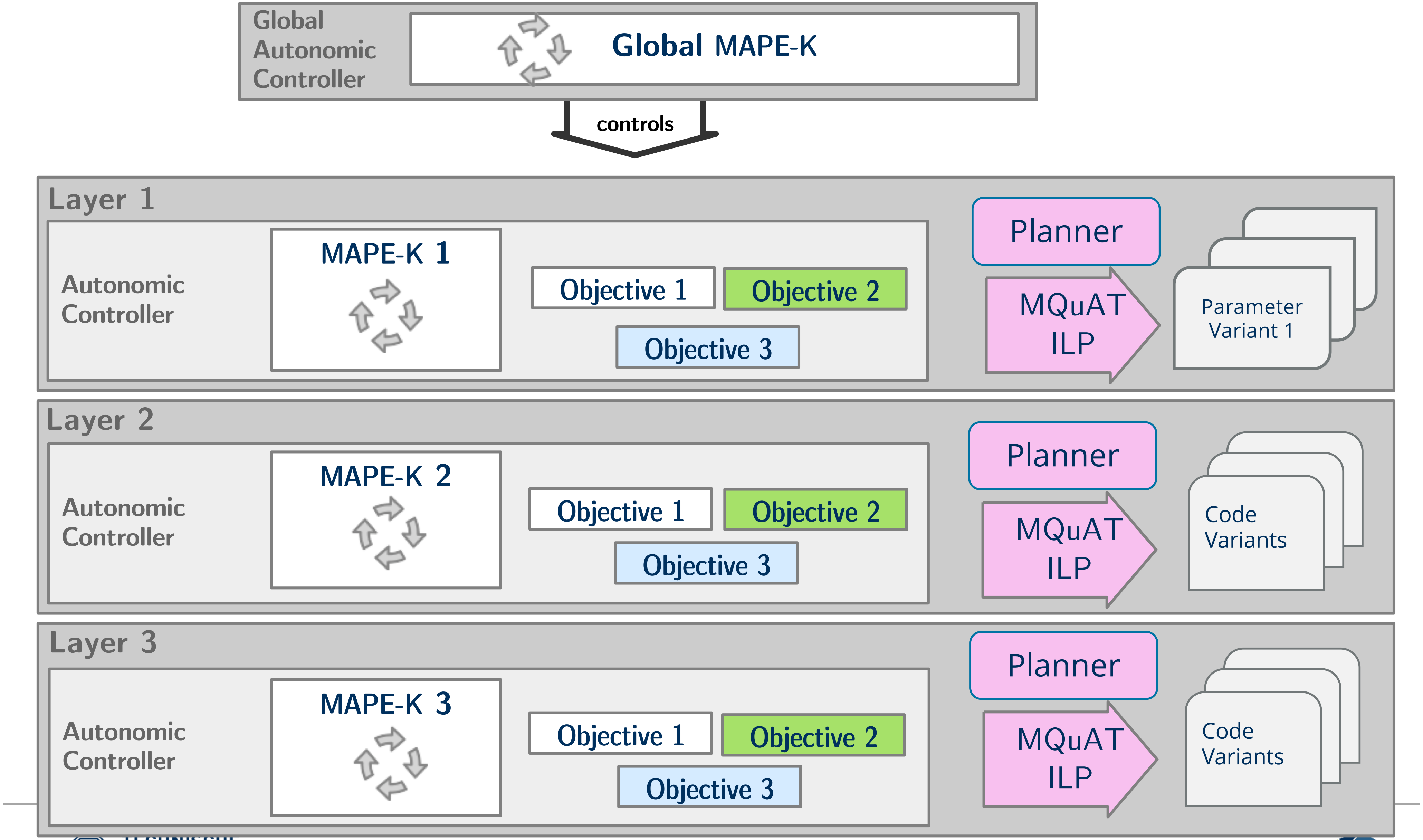
How can we coordinate the changes of the

- Objectives?
- The MAPE-K loops?

Global Autonomic Controller in a MuLAS (Meta-Adaptation)



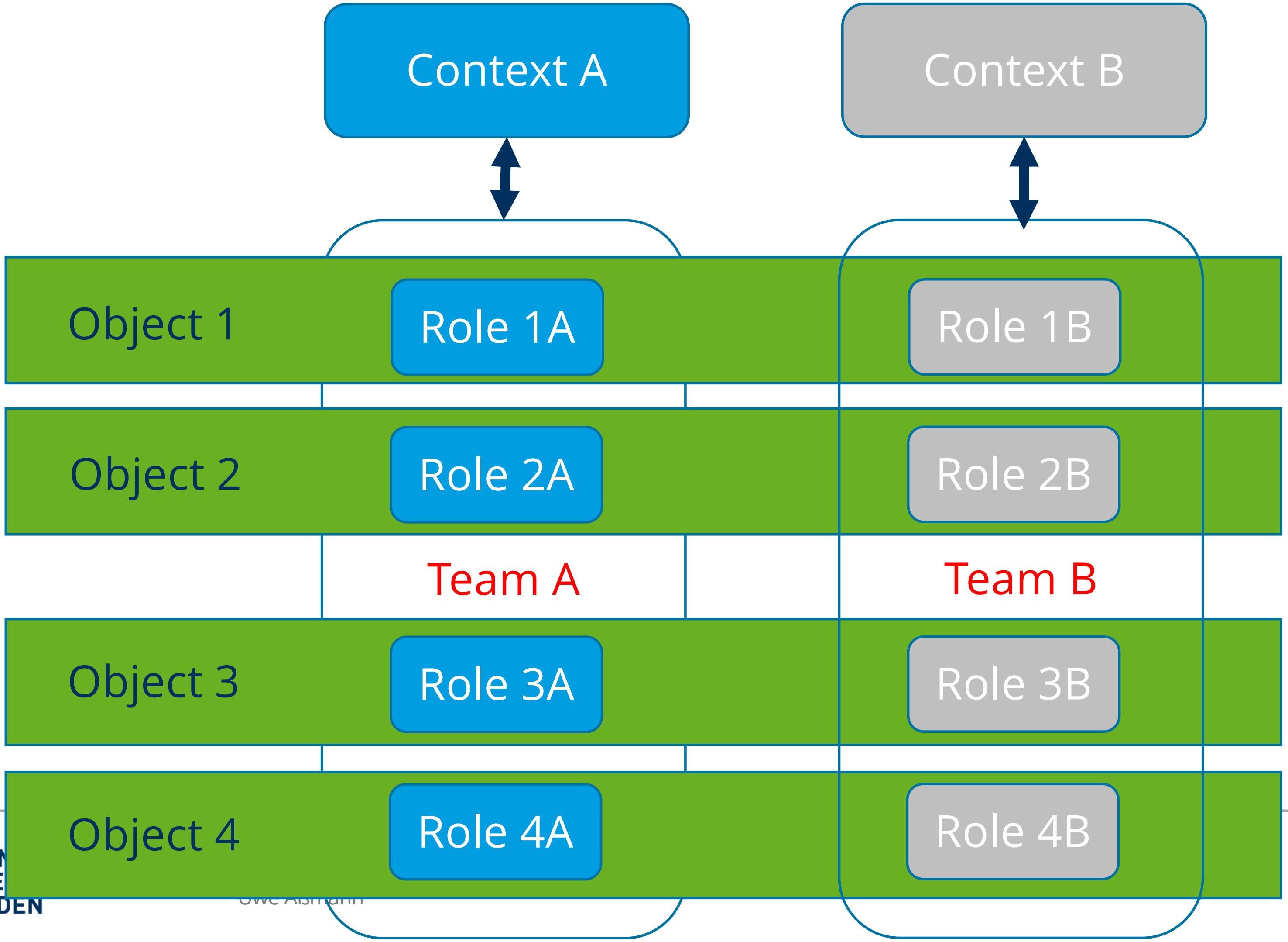
Global Autonomic Controller in a MuLOS (Meta-Optimization)

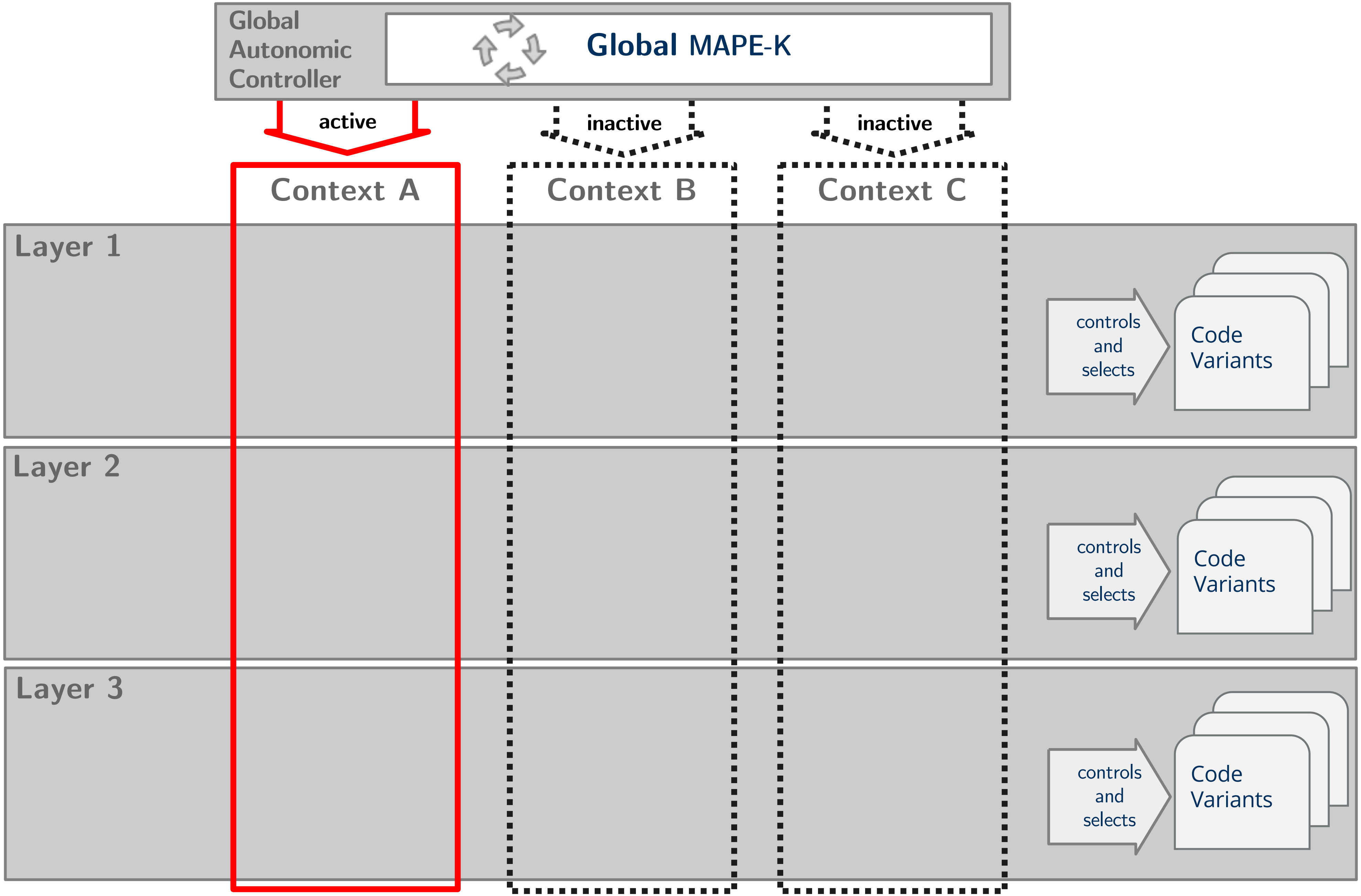


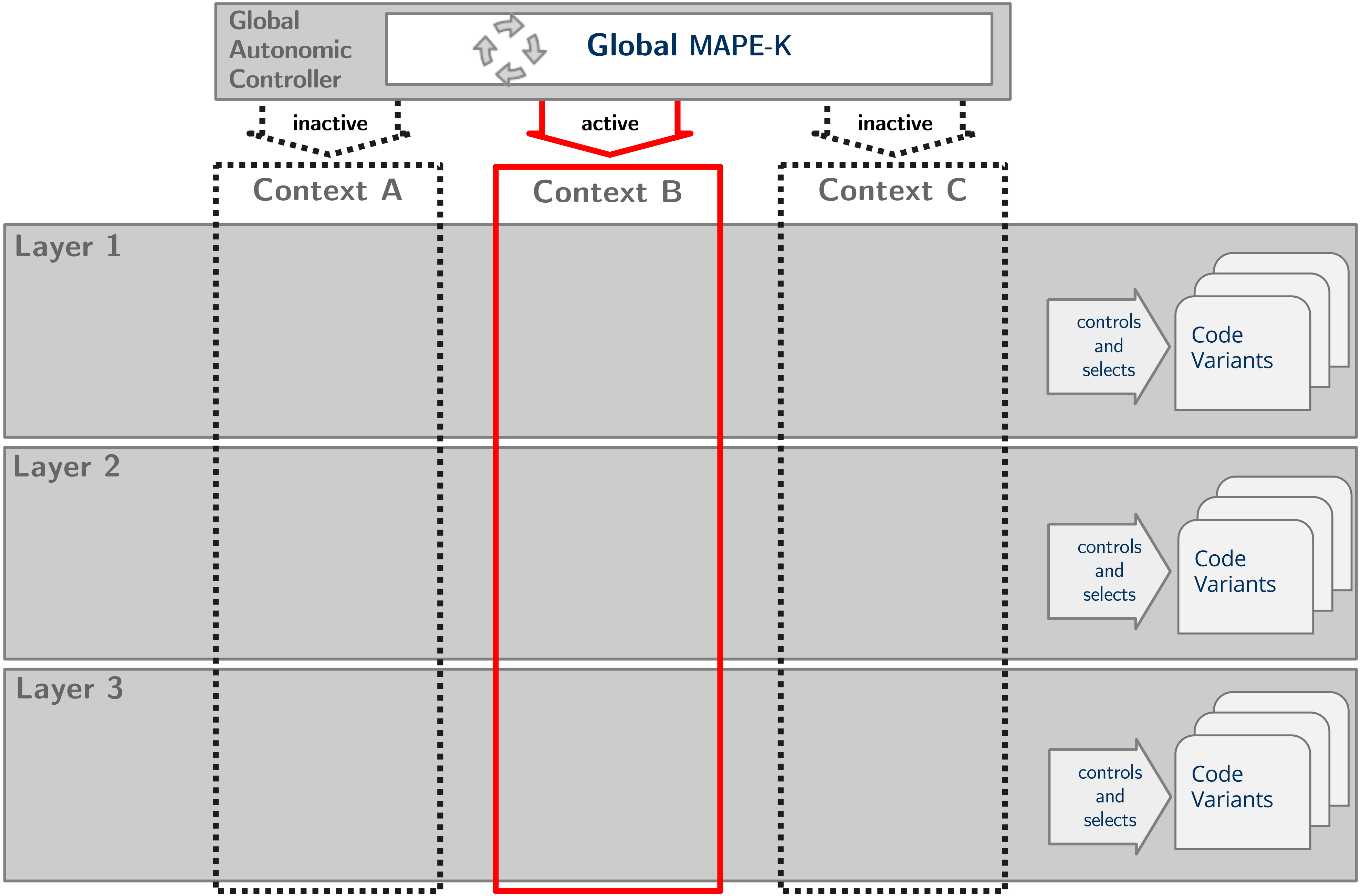
4. Context-Controlled Autonomic Controllers (ConAC Architectural Pattern)

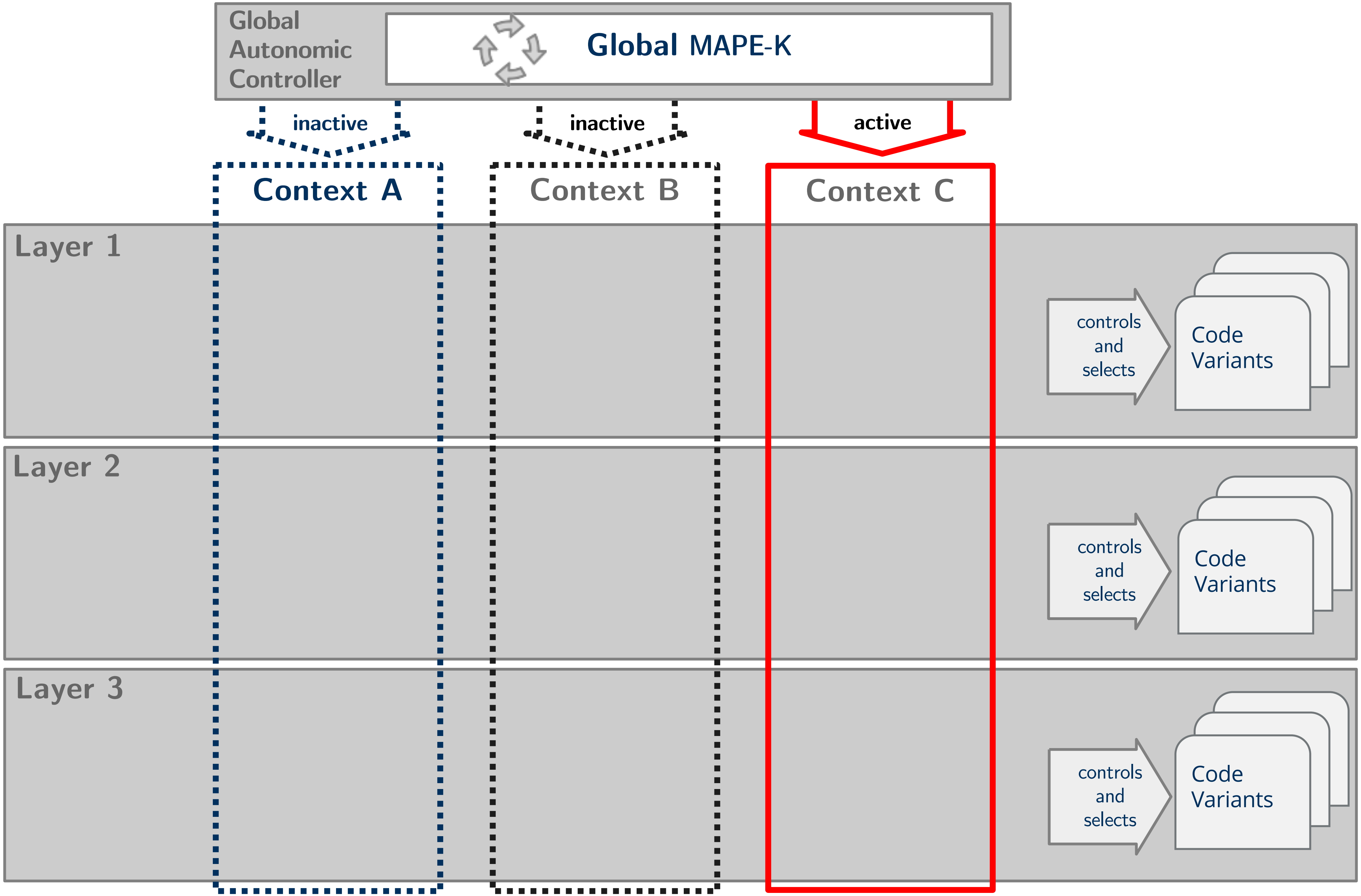
Context-Oriented Programming (COP) for Self-Adaptive Programming

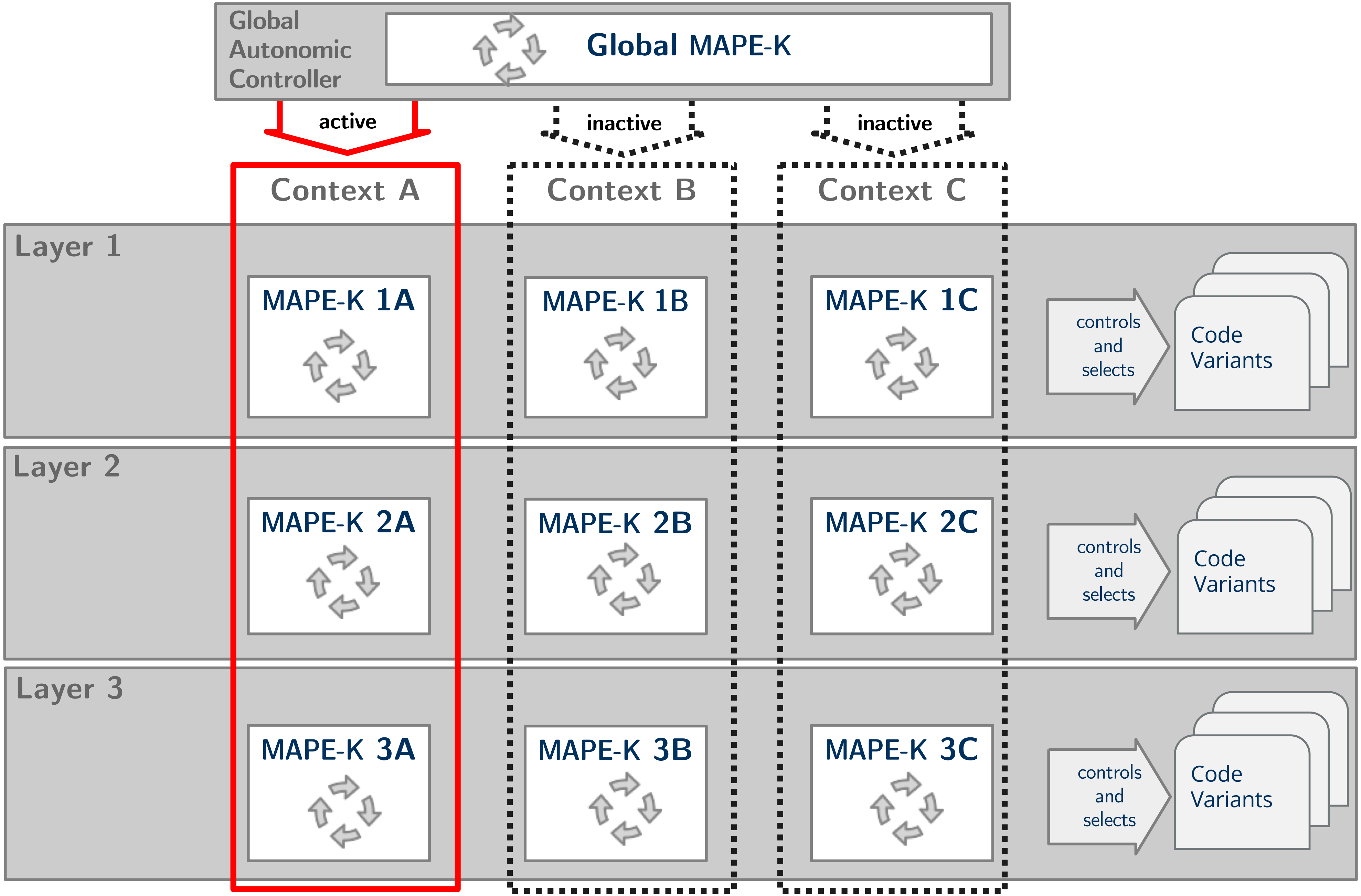
Context, Role, and **Team Objects** [Kühn 2014]

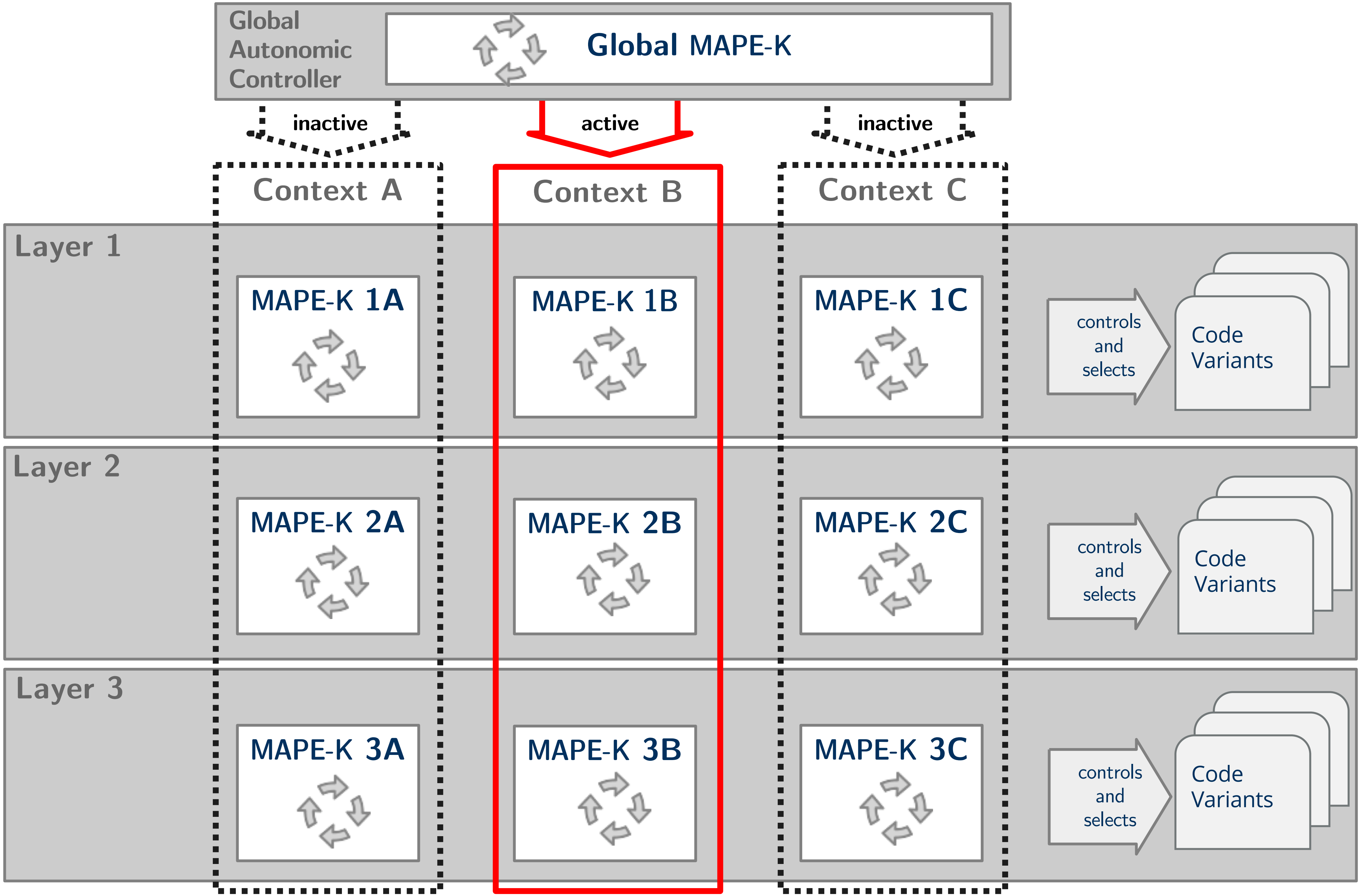


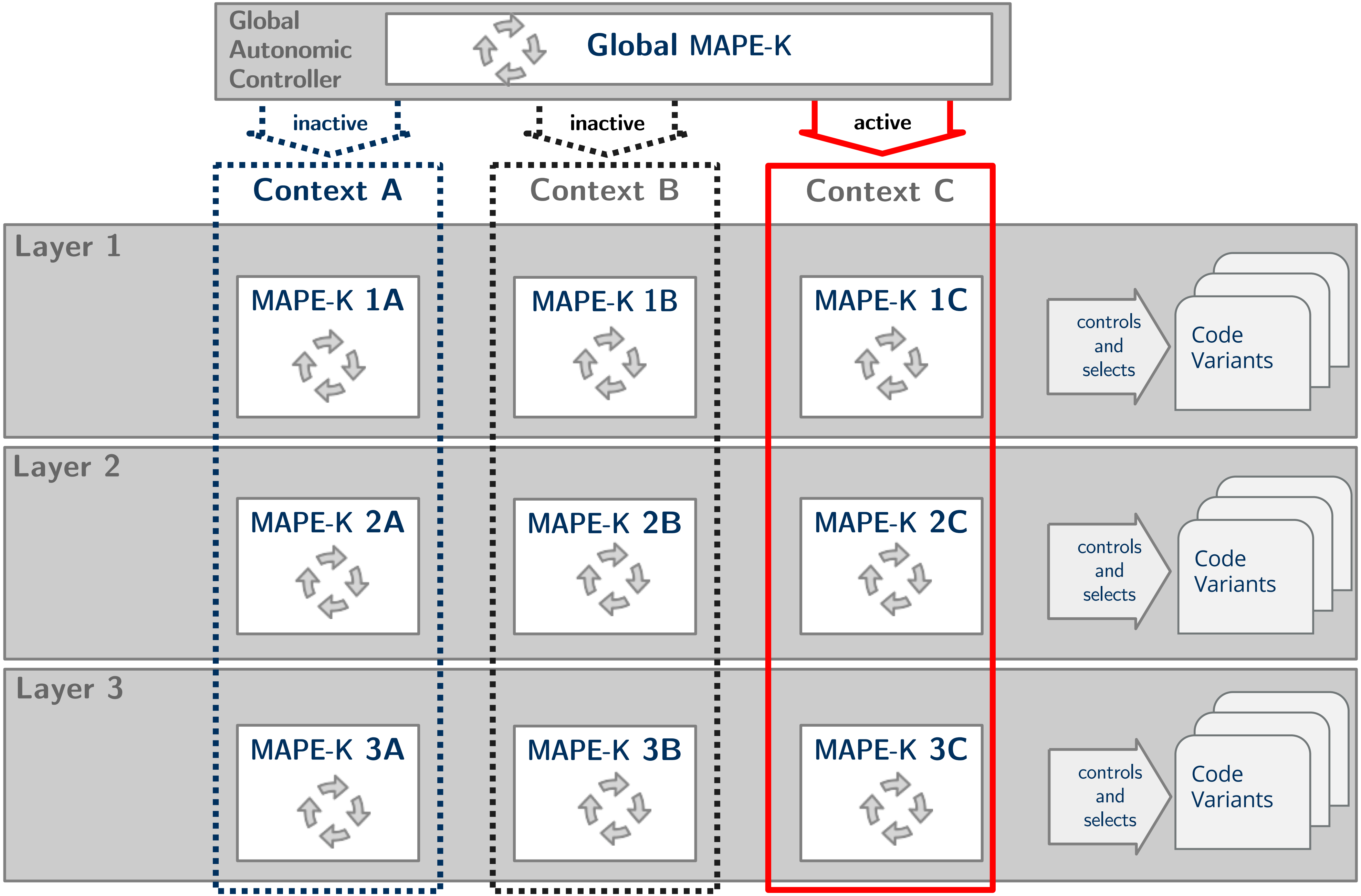

Global Autonomic Controller
Global MAPE-K
active
inactive
inactive
Context A
Context B
Context C
Layer 1
Layer 2
Layer 3
controls and selects
Code Variants
controls and selects
Code Variants
controls and selects
Code Variants


Global Autonomic Controller
Global MAPE-K
inactive
active
inactive
Context A
Context B
Context C
Layer 1
Layer 2
Layer 3
controls and selects
Code Variants
controls and selects
Code Variants
controls and selects
Code Variants


Global
Autonomic
Controller
Global MAPE-K
inactive
inactive
active
Context A
Context B
Context C
Layer 1
Layer 2
Layer 3
controls and selects
Code Variants
controls and selects
Code Variants
controls and selects
Code Variants

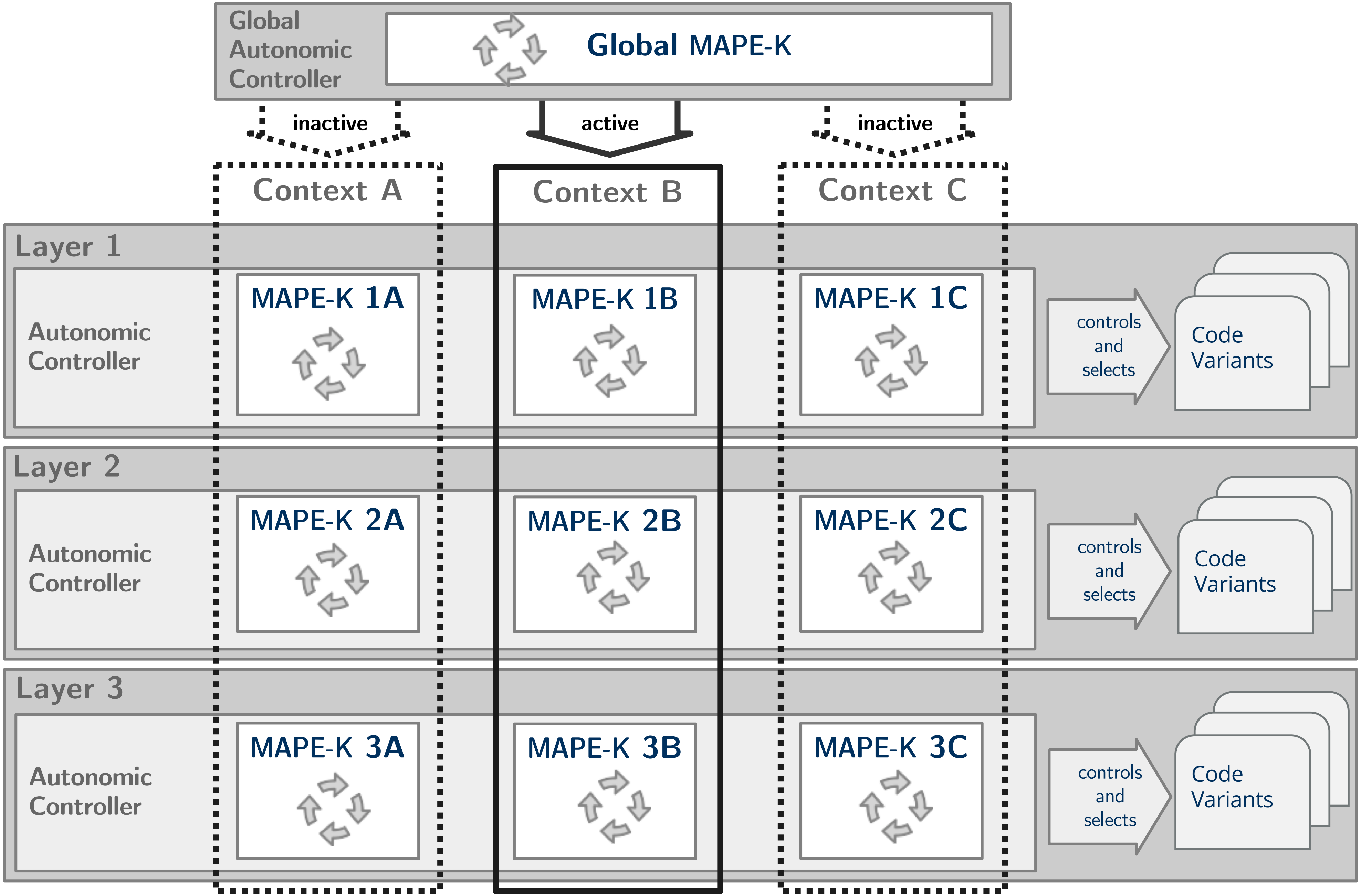

Global Autonomic Controller
Global MAPE-K
active
inactive
inactive
Context A
Context B
Context C
Layer 1
MAPE-K 1A
MAPE-K 1B
MAPE-K 1C
controls and selects
Code Variants
Layer 2
MAPE-K 2A
MAPE-K 2B
MAPE-K 2C
controls and selects
Code Variants
Layer 3
MAPE-K 3A
MAPE-K 3B
MAPE-K 3C
controls and selects
Code Variants


Global Autonomic Controller
Global MAPE-K
inactive
active
inactive
Context A
Context B
Context C
Layer 1
MAPE-K 1A
MAPE-K 1B
MAPE-K 1C
controls and selects
Code Variants
Layer 2
MAPE-K 2A
MAPE-K 2B
MAPE-K 2C
controls and selects
Code Variants
Layer 3
MAPE-K 3A
MAPE-K 3B
MAPE-K 3C
controls and selects
Code Variants


Global Autonomic Controller
Global MAPE-K
inactive
inactive
active
Context A
Context B
Context C
Layer 1
MAPE-K 1A
MAPE-K 1B
MAPE-K 1C
controls and selects
Code Variants
Layer 2
MAPE-K 2A
MAPE-K 2B
MAPE-K 2C
controls and selects
Code Variants
Layer 3
MAPE-K 3A
MAPE-K 3B
MAPE-K 3C
controls and selects
Code Variants

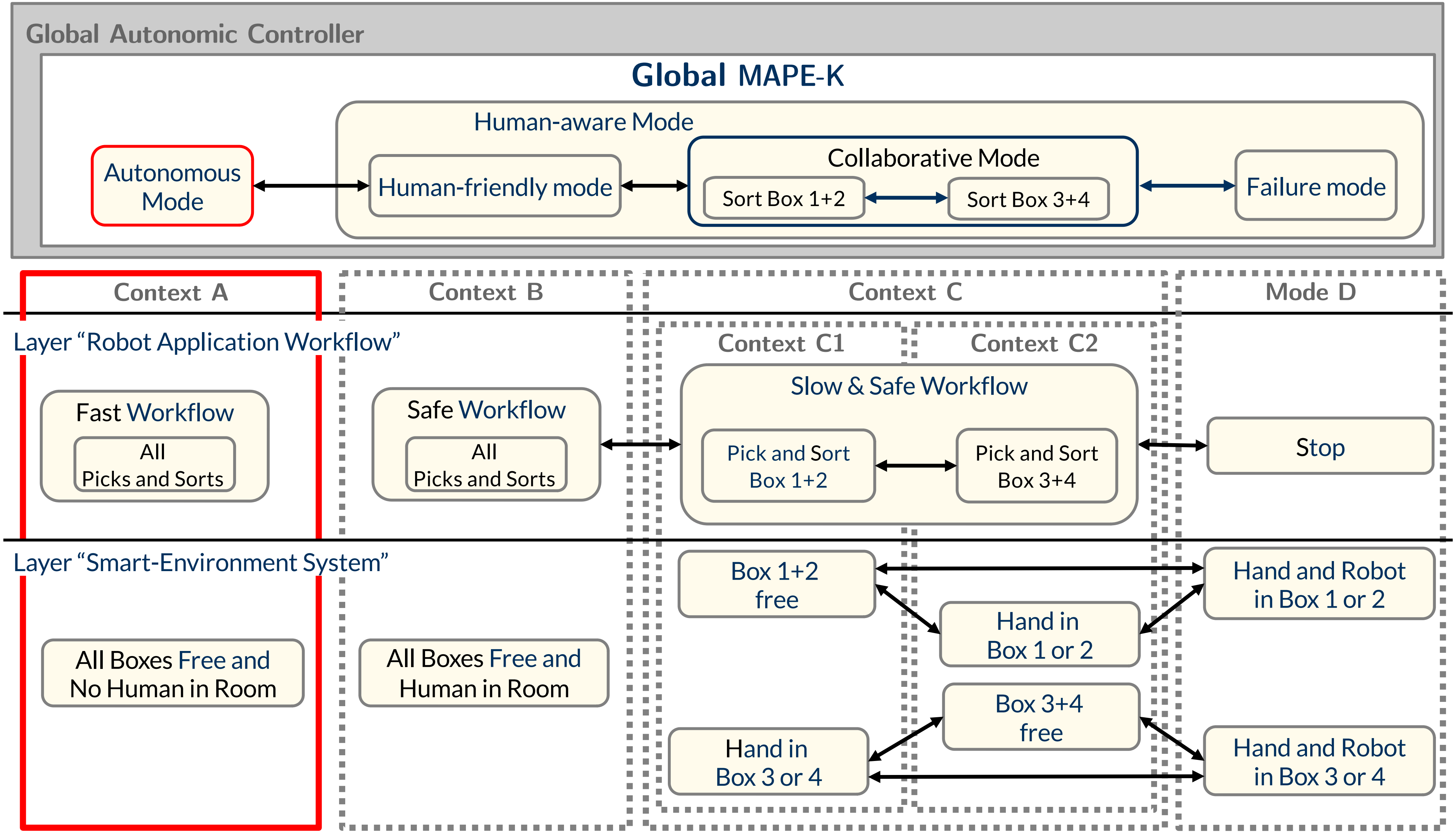
4.2 The ConAC Pattern on 1 Slide

2 dimensional software product line

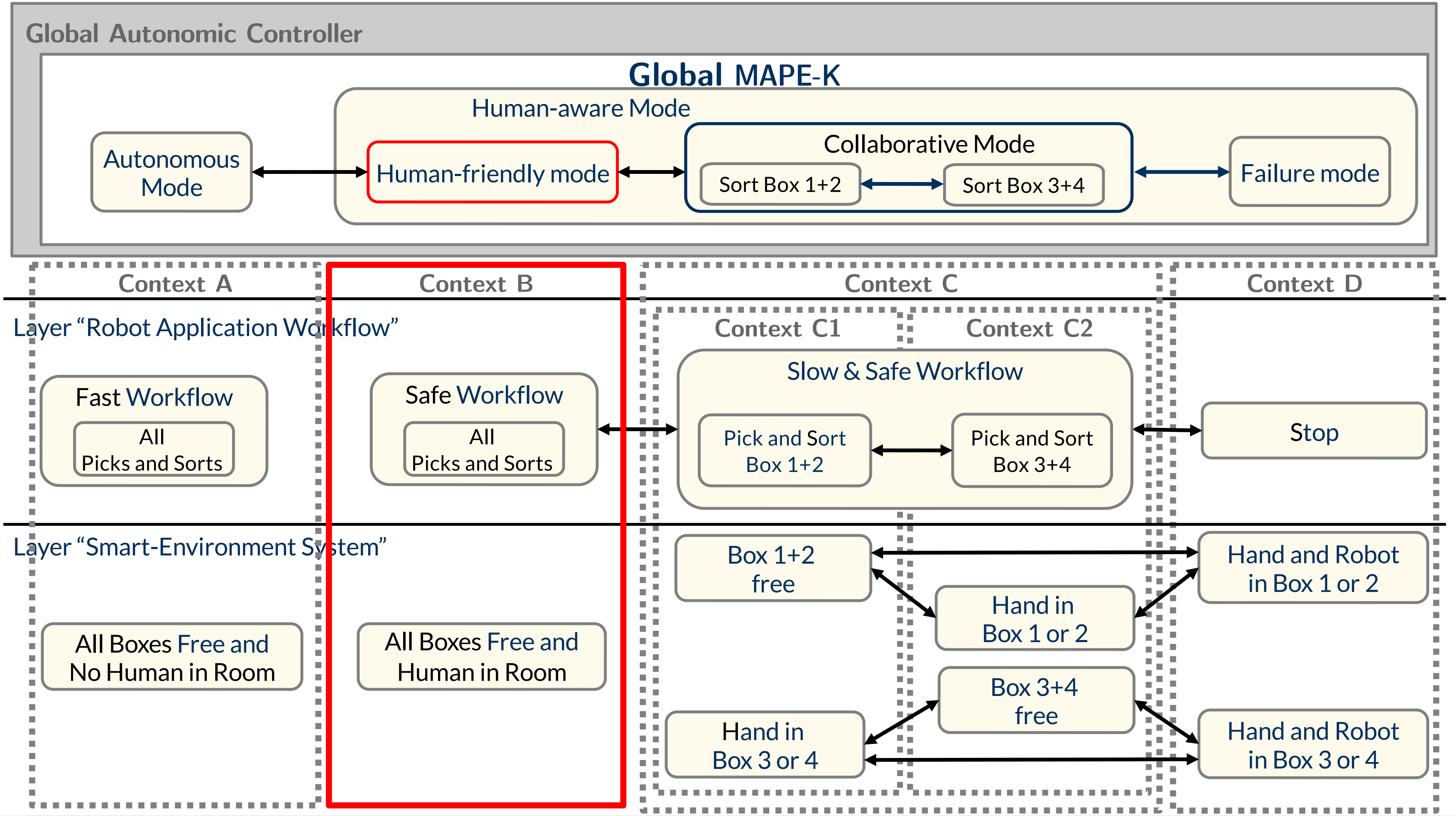

Global Autonomic Controller
Global MAPE-K
inactive
active
inactive
Context A
Context B
Context C
Layer 1
Autonomic Controller
MAPE-K 1A
MAPE-K 1B
MAPE-K 1C
controls and selects
Code Variants
Layer 2
Autonomic Controller
MAPE-K 2A
MAPE-K 2B
MAPE-K 2C
controls and selects
Code Variants
Layer 3
Autonomic Controller
MAPE-K 3A
MAPE-K 3B
MAPE-K 3C
controls and selects
Code Variants

4.3 The ConAC Pattern In Action in Cinderella Robotic Coworking Cell

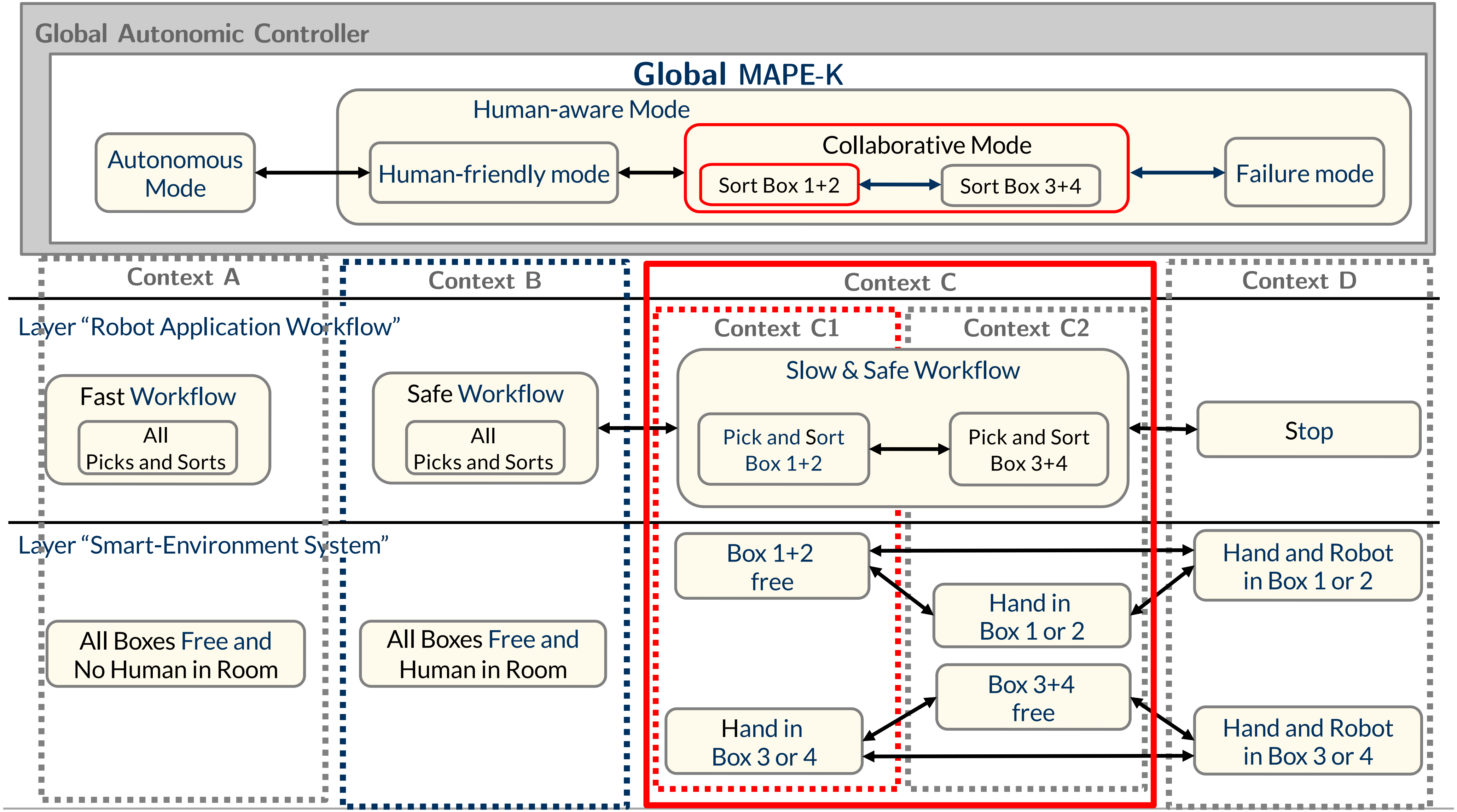
Cinderella Context A is Active



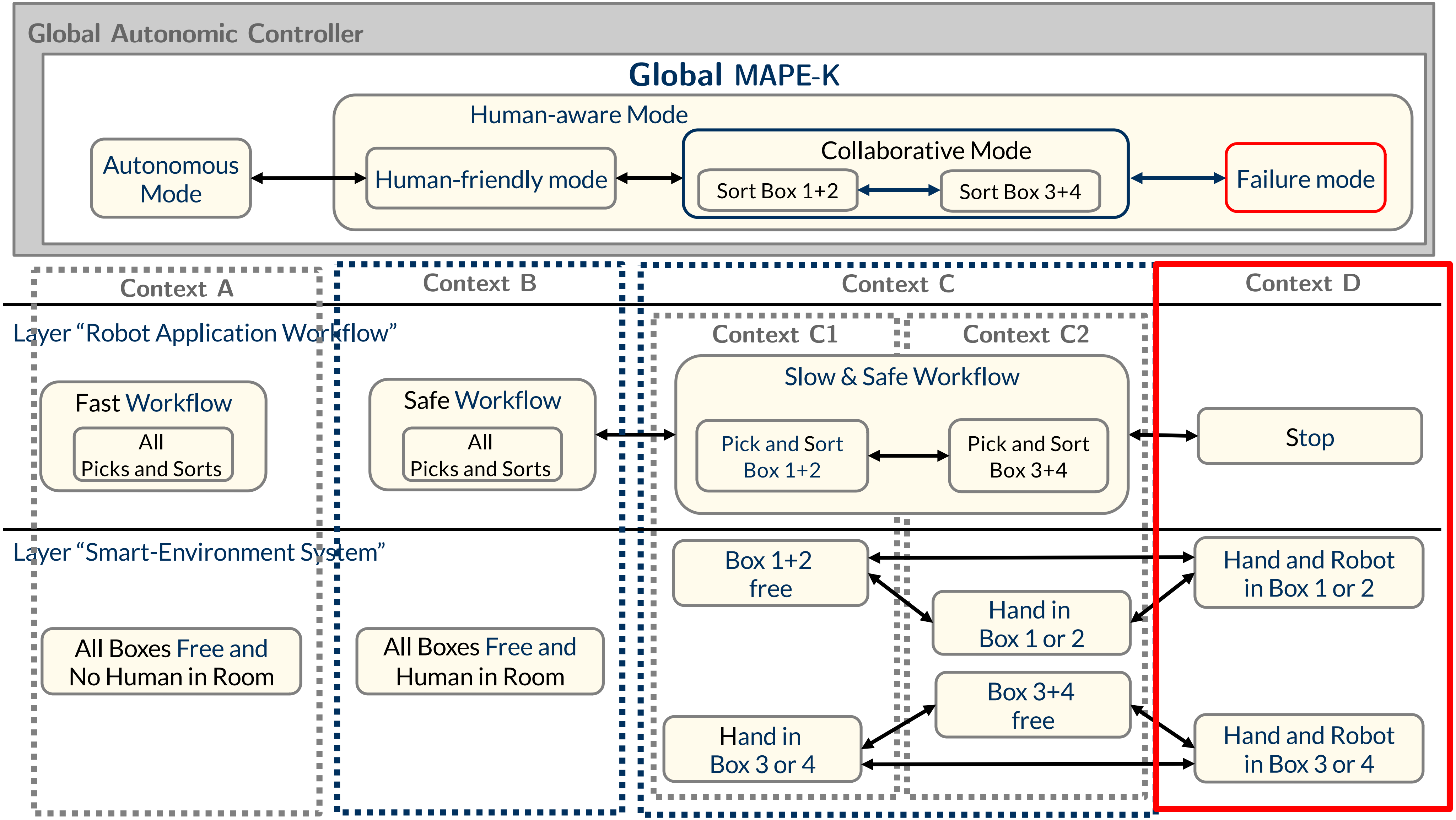
Cinderella Context B is Active



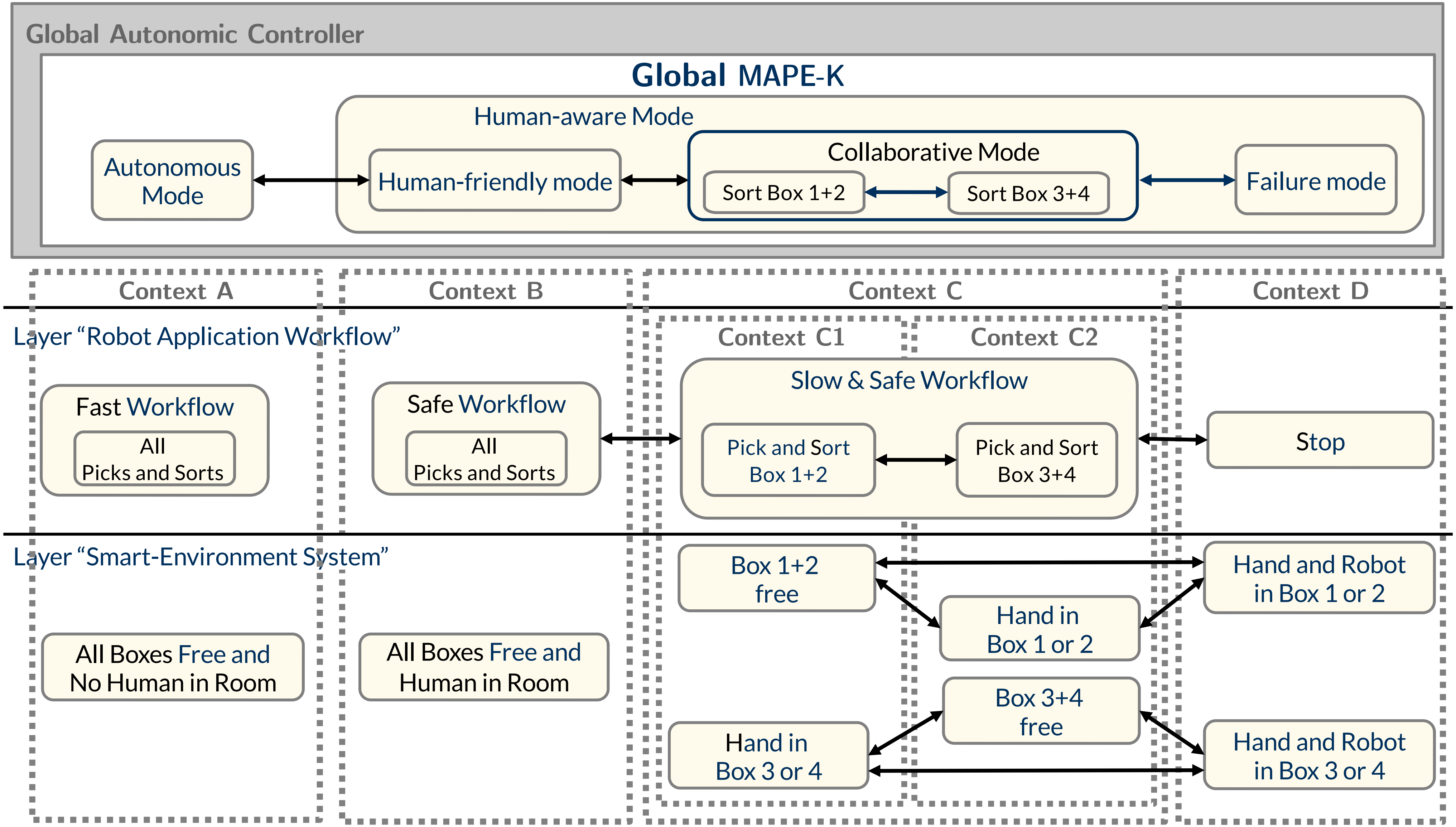
Cinderella Context C1 is Active; Collaboration in the Box 3+4 Possible



Cinderella Context D is Active



ConAC for Cinderella on 1 Page



ConAC Applicability

- **ConAC is a 2dimensional software product line pattern**
  - 1st dimension varies the code and parameter variants
  - 2nd dimension varies the MAPE-K autonomic controllers
- Consistent variation of complex multi-layer self-adaptive systems
  - Robotic coworking cells
  - Autonomous cars
  - Human cyber-physical systems
- Strategy conflicts in meta-adaptation can be avoided
  - the adaptation strategies of the layers can be consistently varied
  - by changing the global context together with the related MAPE-K team of the layer-local autonomic controllers

TECHNISCHE UNIVERSITÄT DRESDEN

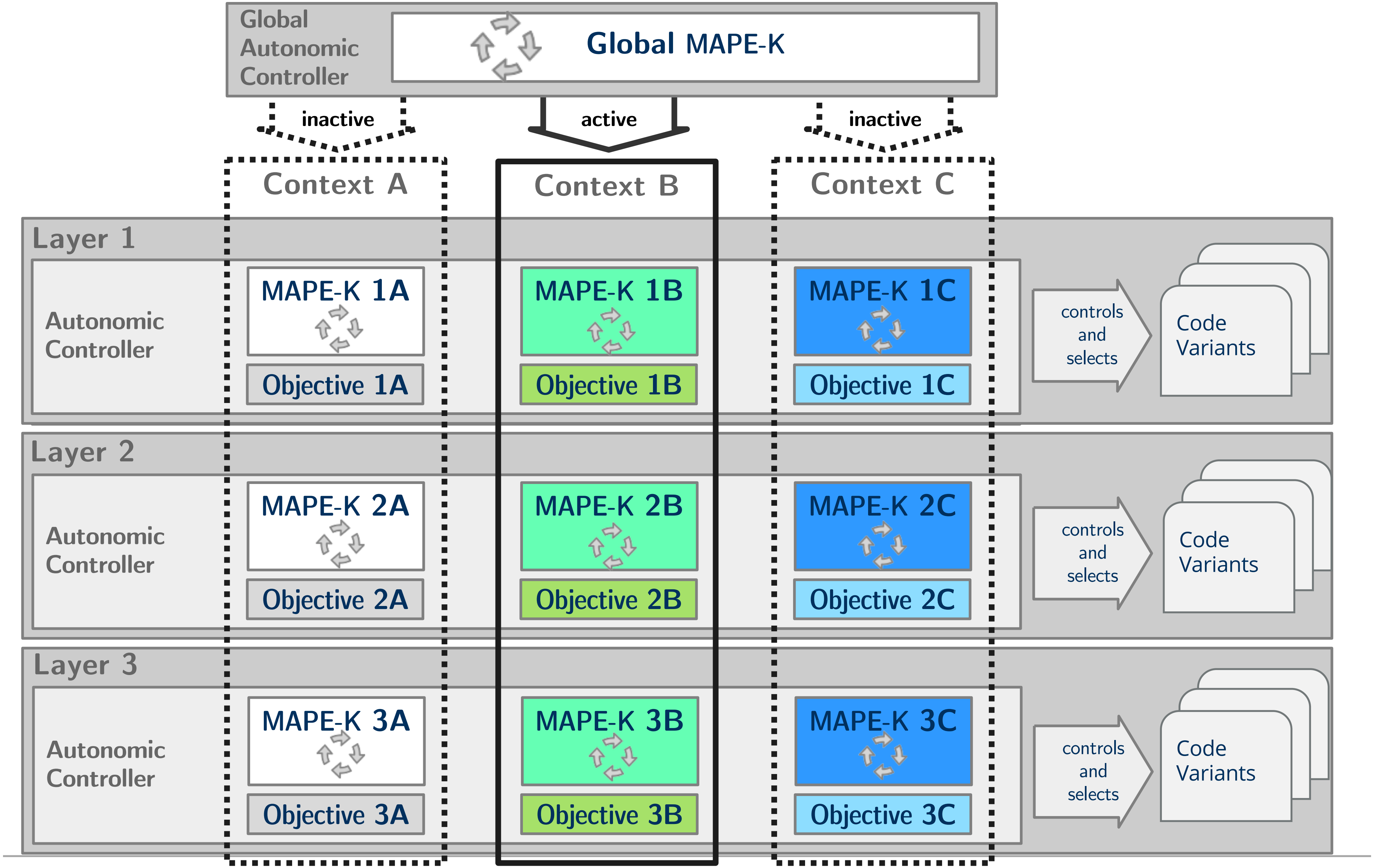
DRESDEN concept

5. Quality-Context-Controlled Autonomic Controllers (qConAC)

MuLOS Structured According To qConAC

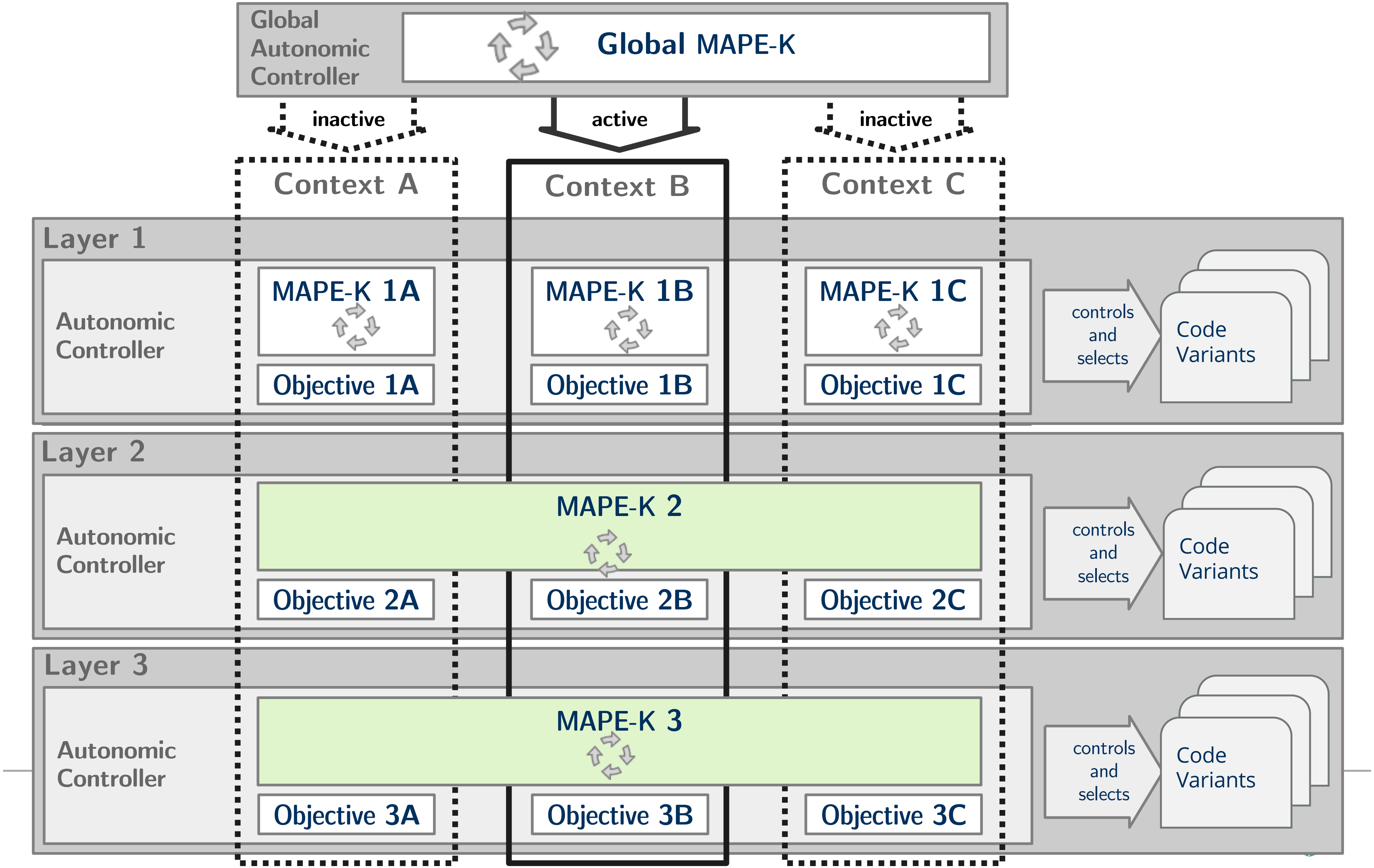
- **Separation of MAPE-K and Objectives**
  - 2nd dimension has two independent teams to vary

- All layers run a MAPE-K-O loop with MAPE-K and O
- **Two crosscutting teams** of the autonomic controllers of each layer,
  - one for MAPE-K (autonomic management, **MAPE-K team**)
  - one for O (objective management, **O team**)

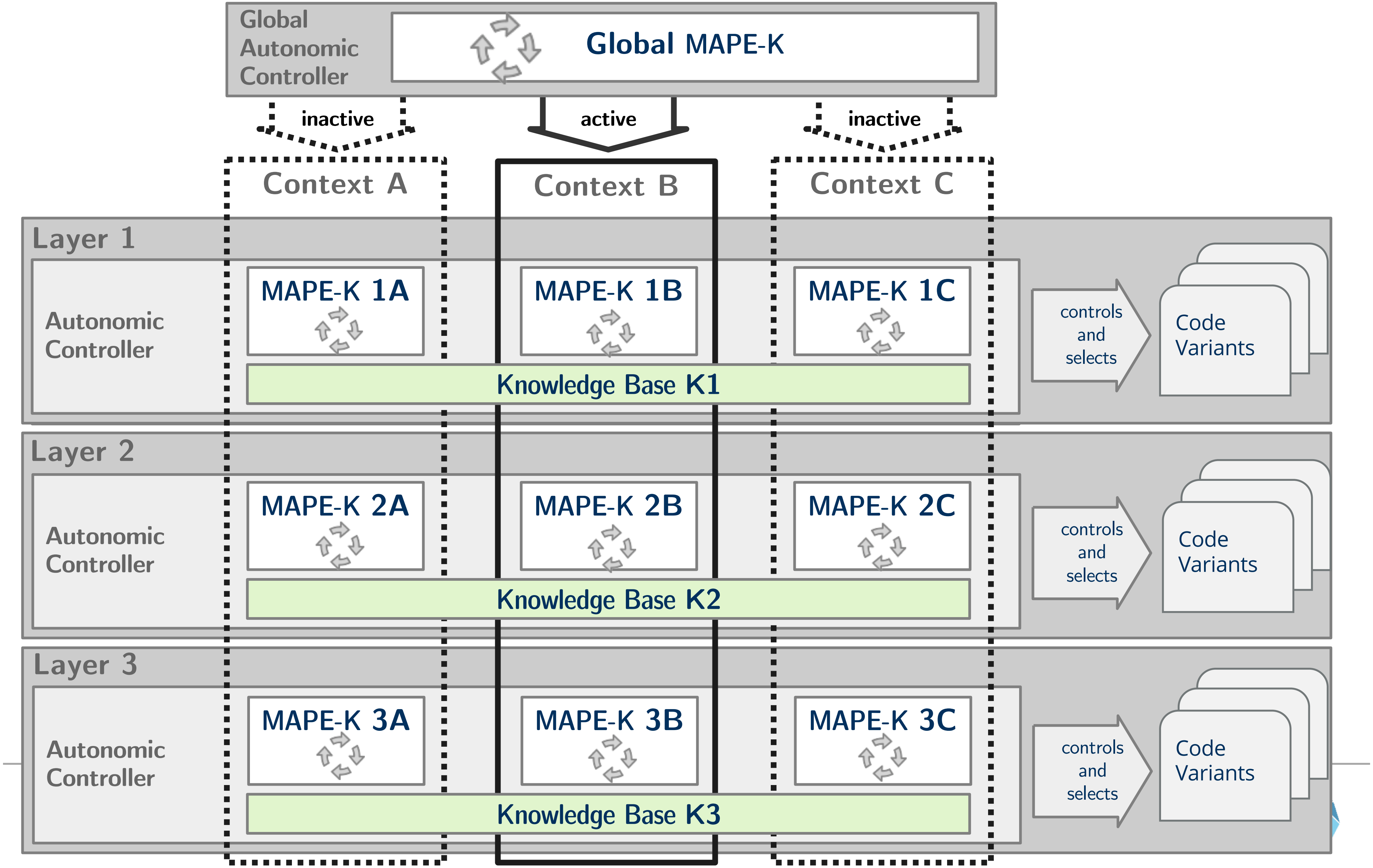

Global Autonomic Controller
Global MAPE-K
inactive
active
inactive
Context A
Context B
Context C
Layer 1
Autonomic Controller
MAPE-K 1A
Objective 1A
MAPE-K 1B
Objective 1B
MAPE-K 1C
Objective 1C
controls and selects
Code Variants
Layer 2
Autonomic Controller
MAPE-K 2A
Objective 2A
MAPE-K 2B
Objective 2B
MAPE-K 2C
Objective 2C
controls and selects
Code Variants
Layer 3
Autonomic Controller
MAPE-K 3A
Objective 3A
MAPE-K 3B
Objective 3B
MAPE-K 3C
Objective 3C
controls and selects
Code Variants

5.2 Other Variants of qConAC

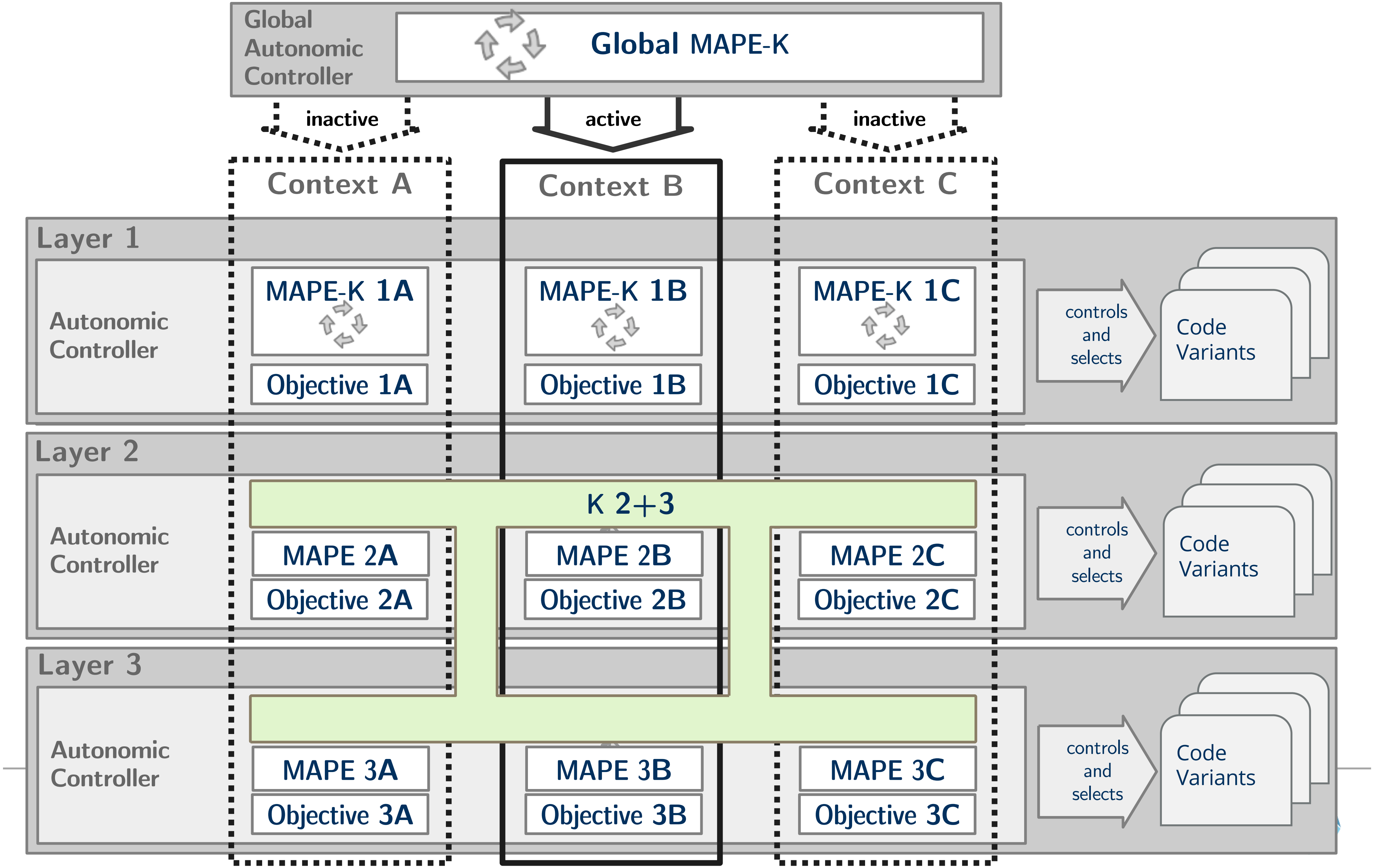
Some Layers May Share Layer Controllers



ConAC with Layer-Shared Knowledge Base

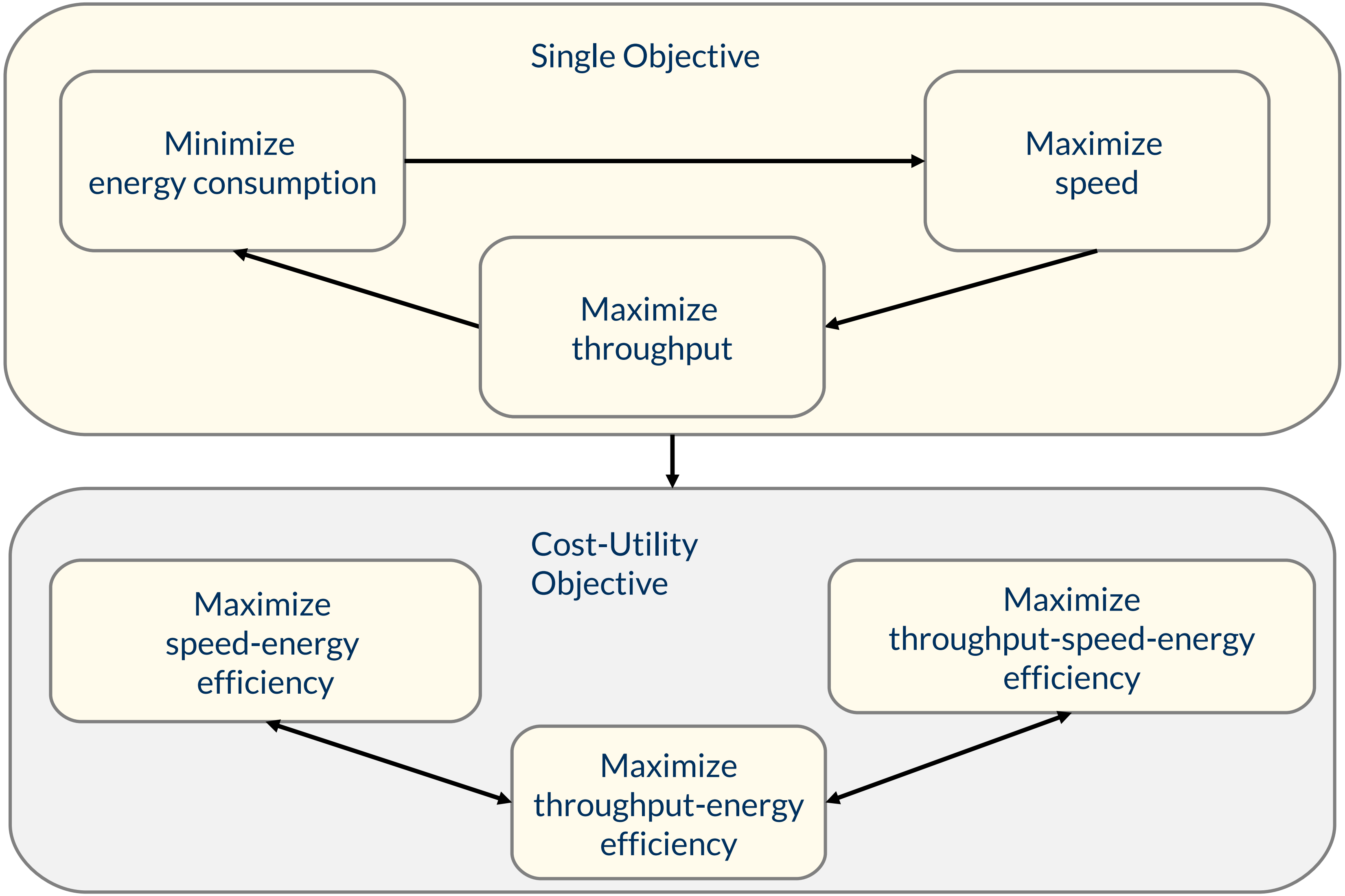


Even Layers May Share Knowledge



6. eConAC for Highly-Adaptive Energy-Efficient Compute Servers (HAEC Servers)

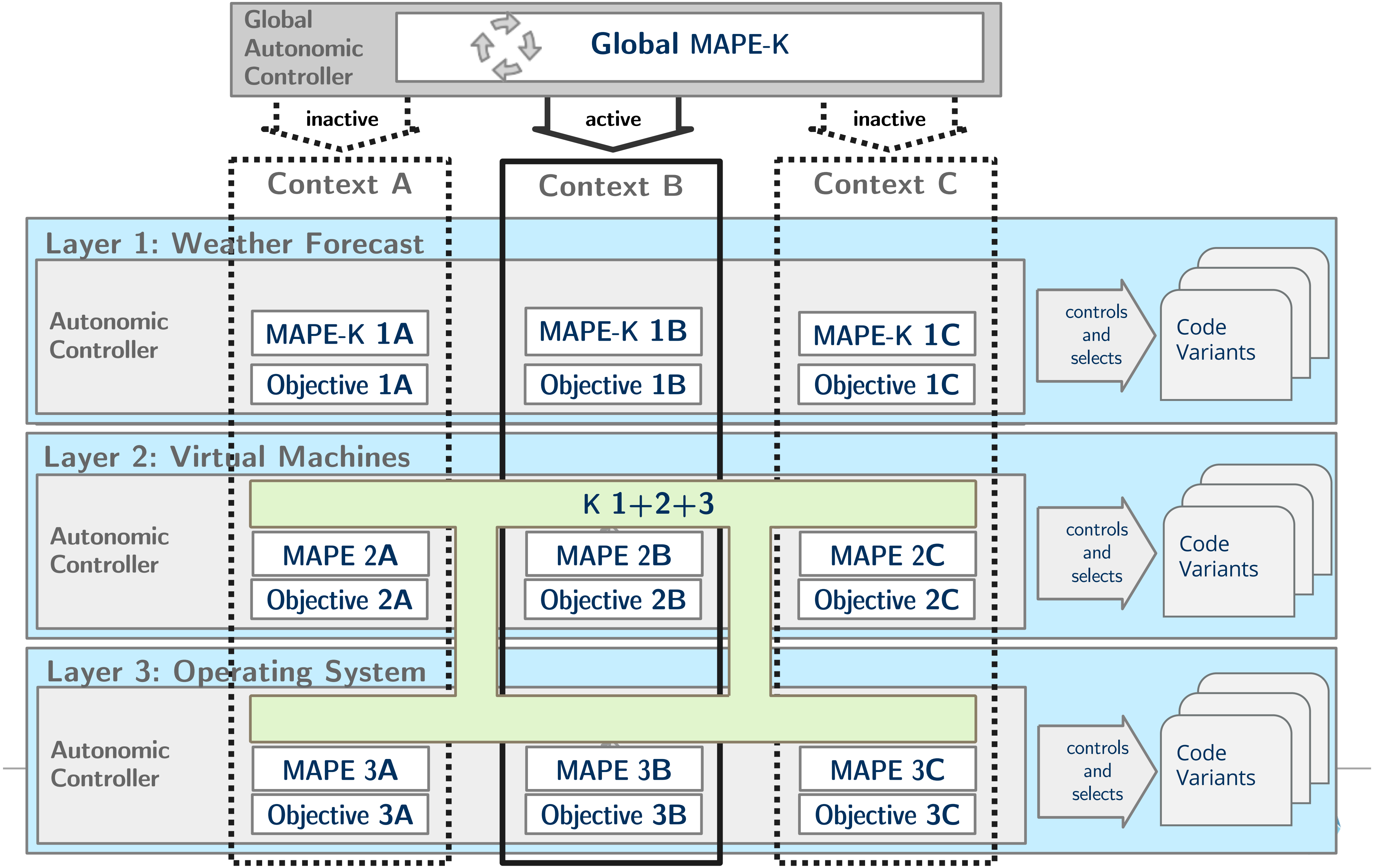
An Global Adaptation Automaton for a Highly-Adaptive Energy-Efficient Server


Single Objective
Minimize energy consumption
Maximize speed
Maximize throughput
Cost-Utility Objective
Maximize speed-energy efficiency
Maximize throughput-speed-energy efficiency
Maximize throughput-energy efficiency

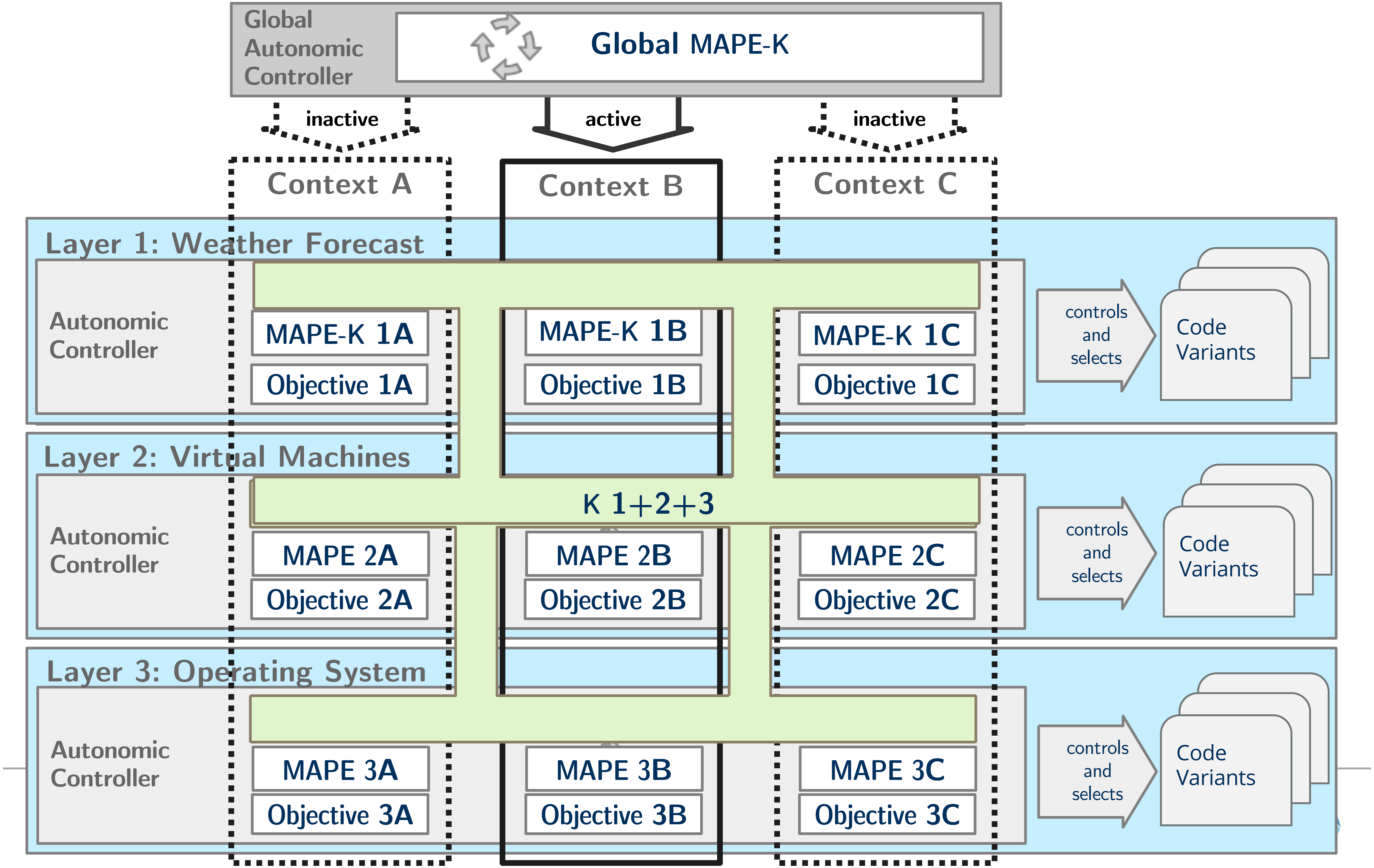
Highly-Adaptive Energy-Efficient Computing Servers (HAEC) with eConAC

- Global energy optimization needs **shared global status**
  - **Communication of global parameters**, such as load or resource pressure
  - for the **local planning of resource allocation**
  - for consistent **global pack-and-switch-off decisions**
  - – that do not only depend on energy objectives,
  - – but also on the system's status
- MAPE-K-O-S loop ensures 3 independent contextual teams K-O-S
  - **S-Team**: manages shared knowledge
- eConAC for energy-adaptive MuLOS provides
  - consistent reconfiguration of energy objectives on all layers
  - precise, shared knowledge of resource pressure of all layers

In a HAEC System, Some Layers Must Share Knowledge



In a HAEC System, All Layers Can Share Knowledge



Global Autonomic Controller

Global MAPE-K

inactive

active

inactive

Context A

Context B

Context C

Layer 1: Weather Forecast

Autonomic Controller

MAPE-K 1A

Objective 1A

MAPE-K 1B

Objective 1B

MAPE-K 1C

Objective 1C

controls and selects

Code Variants

Layer 2: Virtual Machines

Autonomic Controller

K 1+2+3

MAPE 2A

Objective 2A

MAPE 2B

Objective 2B

MAPE 2C

Objective 2C

controls and selects

Code Variants

Layer 3: Operating System

Autonomic Controller

MAPE 3A

Objective 3A

MAPE 3B

Objective 3B

MAPE 3C

Objective 3C

controls and selects

Code Variants

Achievements of the MAPE-K Pattern Language

ConAC

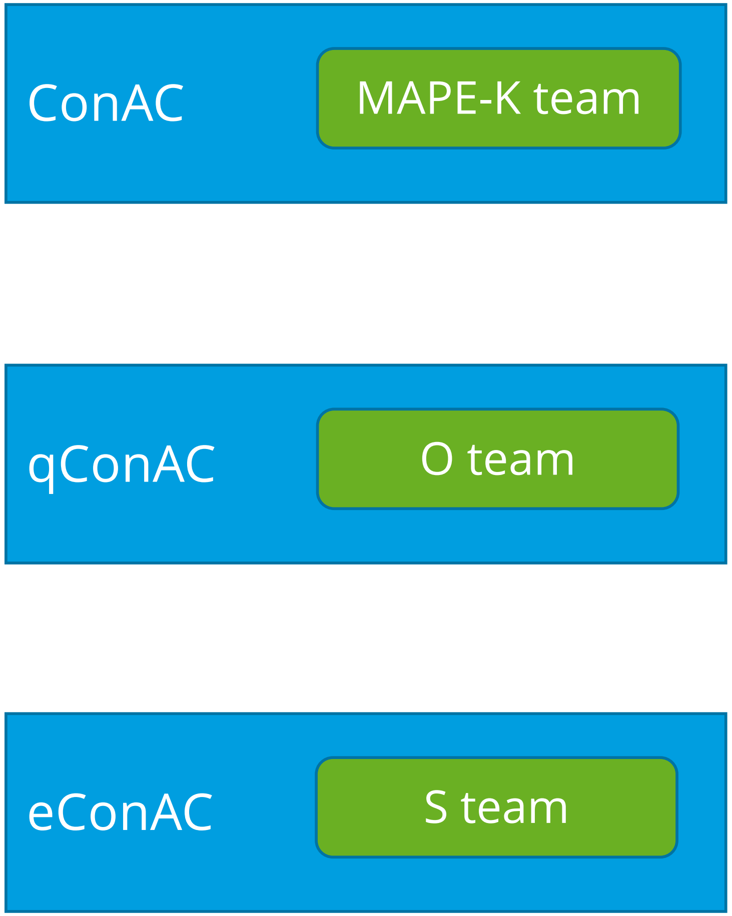
- Helps to build large consistently meta-adapted MuLAS

qConAC

- Meta-adapts objective functions separately
- Large consistently meta-adapted MuLOS

eConAC

- Meta-adapts the knowledge separately
- Large energy-adaptive MuLOS



References

[Abbas 2010] Nadeem Abbas, Jesper Andersson, and Welf Löwe. Autonomic software product lines (ASPL). Software Architecture, 4th European Conference, ECSA 2010, Copenhagen, Denmark, August 23-26, 2010. Companion Volume, ACM.

[Aßmann 2017] Uwe Aßmann, Christian Piechnick, Georg Püschel, Maria Piechnick, Jan Falkenberg, and Sebastian Werner. Modelling the world of a smart room for robotic co-working. 5th International Conference on Model-Driven Engineering and Software Development (MODELSWARD), Cham, 2018. Springer

[Götz 2013] Sebastian Götz. Multi-Quality Auto-Tuning by Contract Negotiation. PhD thesis, Technische Universität Dresden, Fakultät Informatik, July 2013. http://st.inf.tu-dresden.de/sgoetz/

[Haddadin 2009] Sami Haddadin, Michael Suppa, Stefan Fuchs, Tim Bodenmüller, Alin Albu-Schäffer, and Gerd Hirzinger. Towards the robotic co-worker. ISRR, volume 70 of Springer Tracts in Advanced Robotics, Springer, 2009.

References

[Kühn 2014] Thomas Kühn, Max Leuthäuser, Sebastian Götz, Christoph Seidl, and Uwe Aßmann. A metamodel family for role-based modeling and programming languages. SLE, volume 8706 of LNCS. Springer, 2014.

[Weyns 2013] Danny Weyns, Bradley Schmerl, Vincenzo Grassi, Sam Malek, Raffaela Mirandola, Christian Prehofer, Jochen Wuttke, Jesper Andersson, Holger Giese, and Karl M. Göschka. On Patterns for Decentralized Control in Self-Adaptive Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.