Runtime adaptation as a programming pattern in service composition


Carlos G. Lopez Pombo@


Pablo Montepagano@


Emilio Tuosto@

Coordination
Urbino 10 June, 2026

Take-away message

Take-away message

Runtime architectural adaptation

Take-away message

Runtime architectural adaptation
= Monitoring

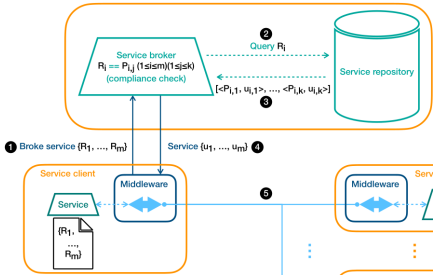
Take-away message

Runtime architectural adaptation
= Monitoring
+ Decision policy

Take-away message

Runtime architectural adaptation
= Monitoring
+ Decision policy
+ Semantic-based service composition

from

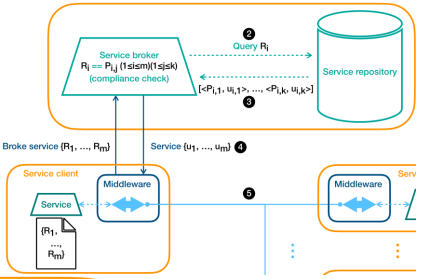


SEAdS @ Coord 2024

Take-away message

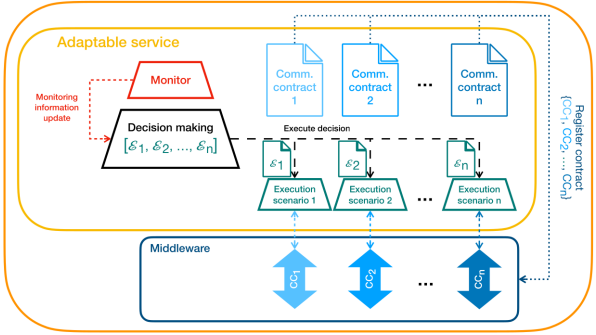
Runtime architectural adaptation
= Monitoring
+ Decision policy
+ Semantic-based service composition

from



SEAMS @ Ecoud 2024

to



Menu

Menu

- A bird-eye watch of SEArch

Menu

- A bird-eye watch of SEArch
- Adaptation (as we see it)

Menu

- A bird-eye watch of SEArch
- Adaptation (as we see it)
- Our programming pattern

Menu

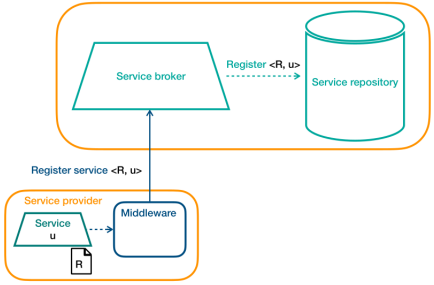
- A bird-eye watch of SEArch
- Adaptation (as we see it)
- Our programming pattern
- Some conclusions

Menu

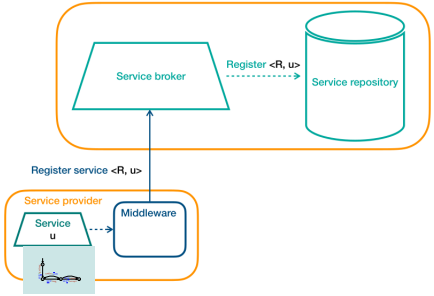
- A bird-eye watch of SEArch
- Adaptation (as we see it)
- Our programming pattern
- Some conclusions

– Prelude –

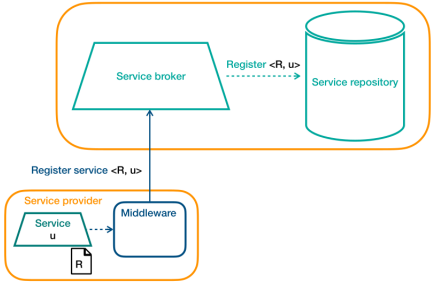
SEArch basics



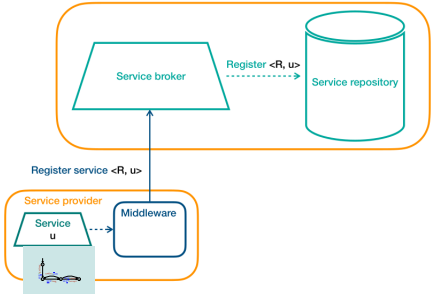
SEArch basics



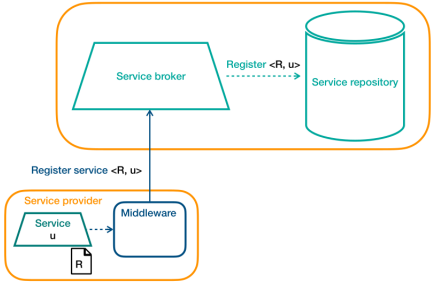
SEArch basics



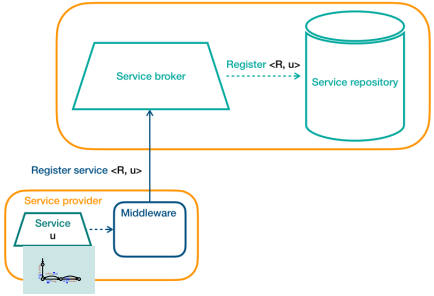
SEArch basics



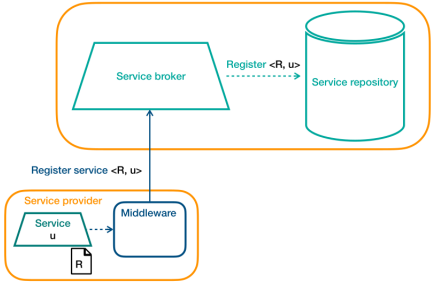
SEArch basics



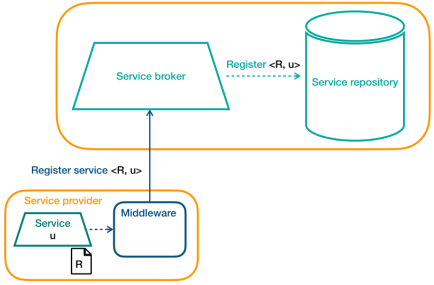
SEArch basics

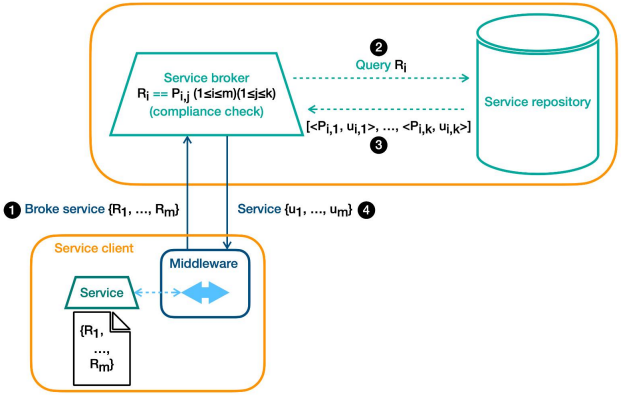


SEArch basics

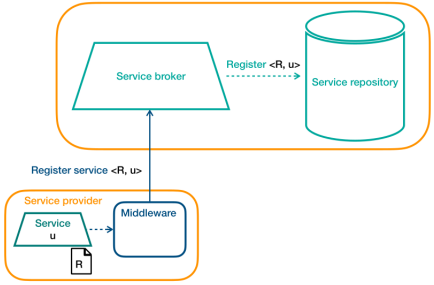


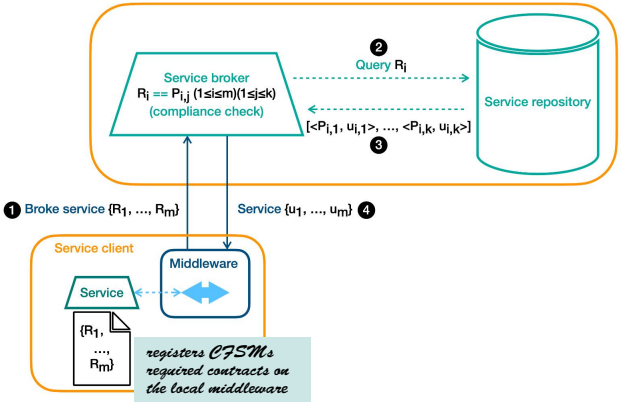
SEArch basics



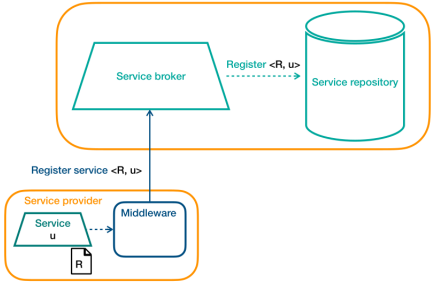


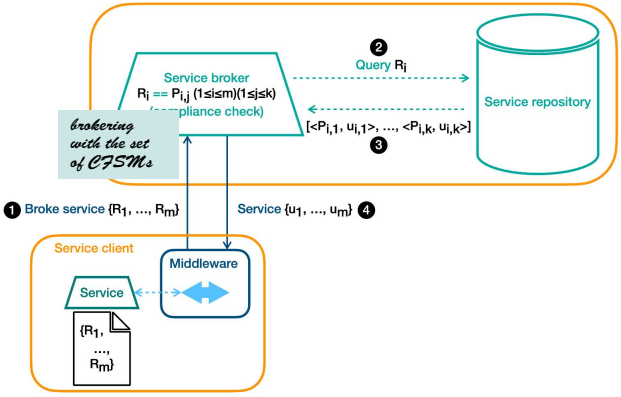
SEArch basics



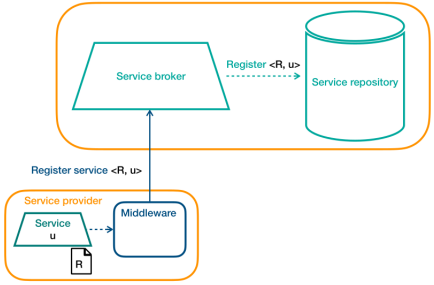


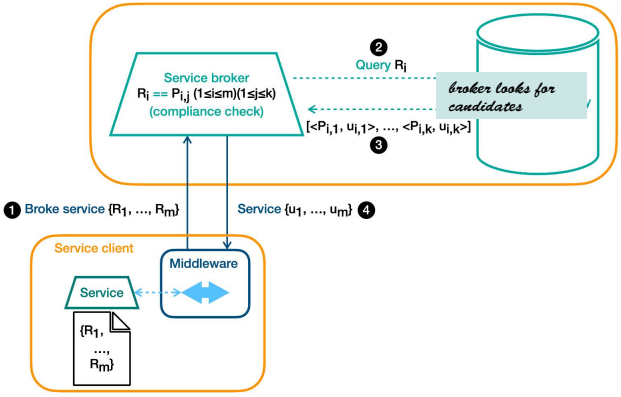
SEArch basics



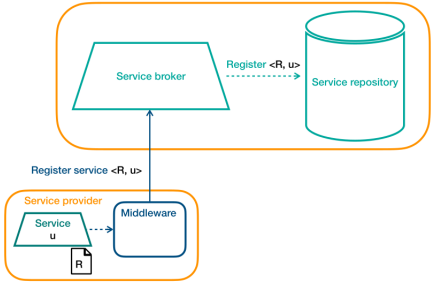


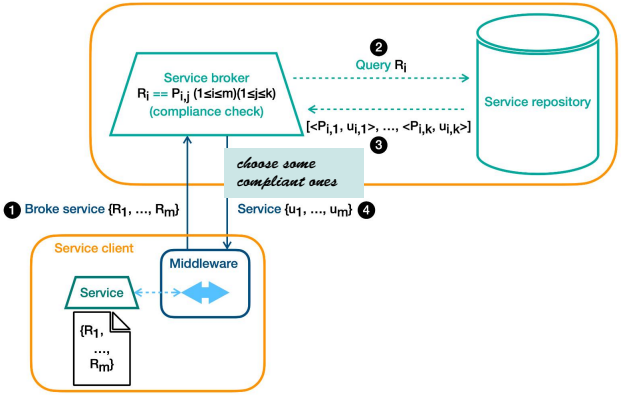
SEArch basics



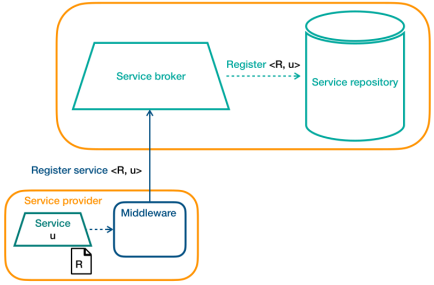


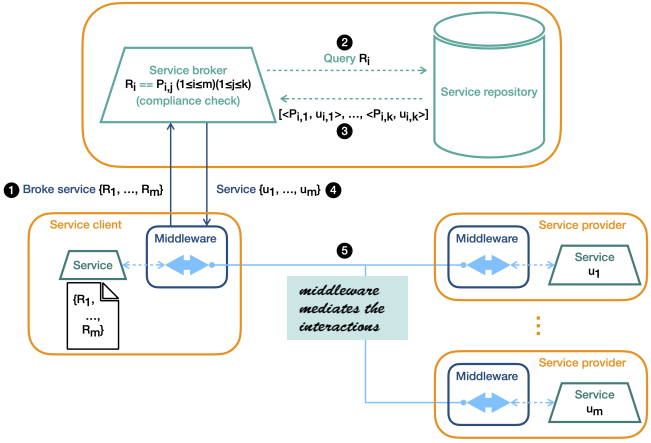
SEArch basics





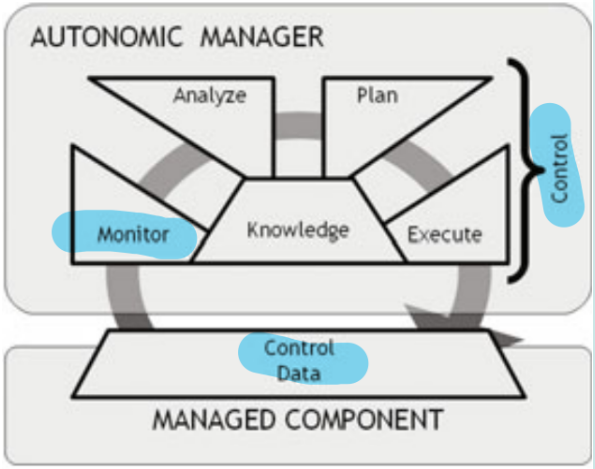
SEArch basics





– On adaptation –

Adaptation basics



Components' behaviour = control + data

- *adaptation = process/modify specific computational/physical resources*
- *control data [2]*[a]
  - *memory usage*
  - *network traffic*
  - *room temperature*
  - ...

Control data are an "adaptation"-dependent design choice

[a] Picture borrowed from [2]

A benchmark for adaptation

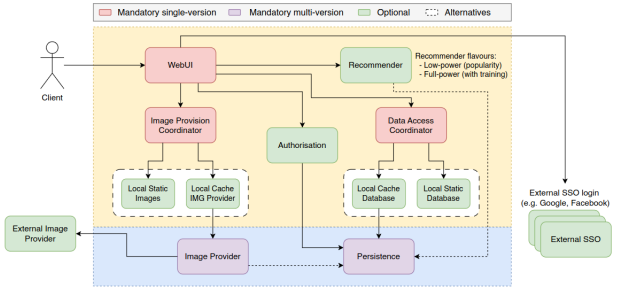


Figure 2: Adaptable TeaStore Architecture (functionalities shown within the coloured areas are internal, those in the blue area are outsourceable, in the yellow area—non-outsourceable (cf. Table 1))

"a broad catalogue of reference adaptation scenarios centred around Adaptable TeaStore, useful to evaluate the ability of a given adaptation technology to address conditions such as component unavailability, cyberattacks, provider outages, benign/-malicious traffic increases, and user-triggered reconfigurations" [1]

A running example

[1] Adapted from [1, 7]

A running example

The adaptable TeaStore benchmark[1]

- a webUI (app users' entry point)
- a remote image provider
- a locally hosted DB of images

[1] Adapted from [1, 7]

A running example

The adaptable TeaStore benchmark[1]

- a webUI (app users' entry point)
- a remote image provider
- a locally hosted DB of images

According to the specification

- providers process images (e.g., applying filters, resizing, etc.);
- webUI automatically generates webpages with images
  - from image providers
  - or from DB for runtime adaptation
  - or else directly from the hard drive

[1] Adapted from [1, 7]

Our adaptable TeaStore

Our adaptable TeaStore

Adaptation triggered by service degradation depending on

- network Latency
- Responsiveness of image provider
- CPU load of webUI

$$\mathcal{E}_1 = \forall x : \mathsf{L} < x \implies \mathsf{R} < x^2$$

$$\mathcal{E}_2 = \neg\mathcal{E}_1 \wedge 0 \leq \mathsf{C} \leq .9$$

$$\mathcal{E}_3 = \neg\mathcal{E}_1 \wedge \mathsf{C} > .9$$

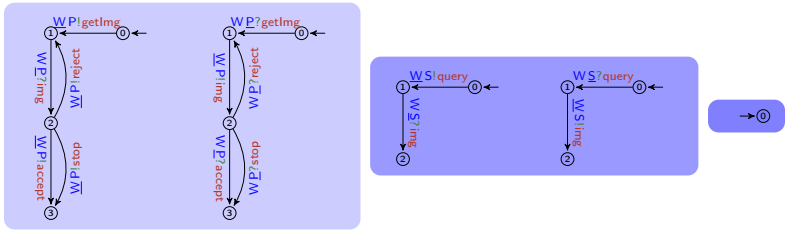
Our adaptable TeaStore

Adaptation triggered by service degradation depending on

- network Latency
- Responsiveness of image provider
- CPU load of webUI

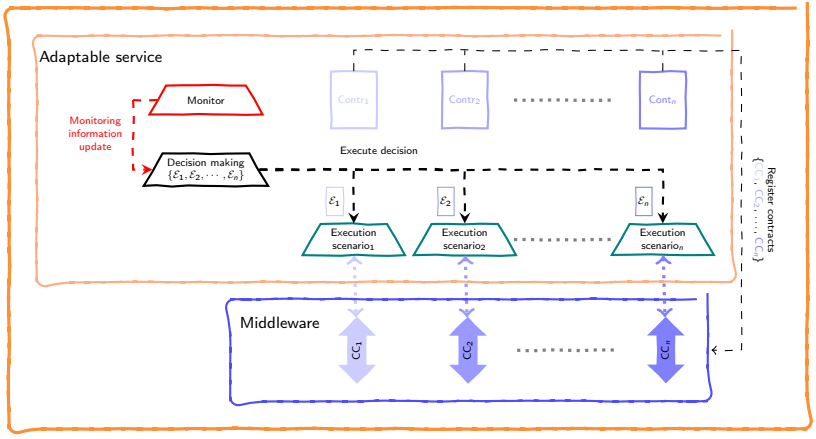
$$\mathcal{E}_1 = \forall x : \mathsf{L} < x \implies \mathsf{R} < x^2$$
$$\mathcal{E}_2 = \neg\mathcal{E}_1 \wedge 0 \leq \mathsf{C} \leq .9$$
$$\mathcal{E}_3 = \neg\mathcal{E}_1 \wedge \mathsf{C} > .9$$

Contracts for the 3 scenarios:

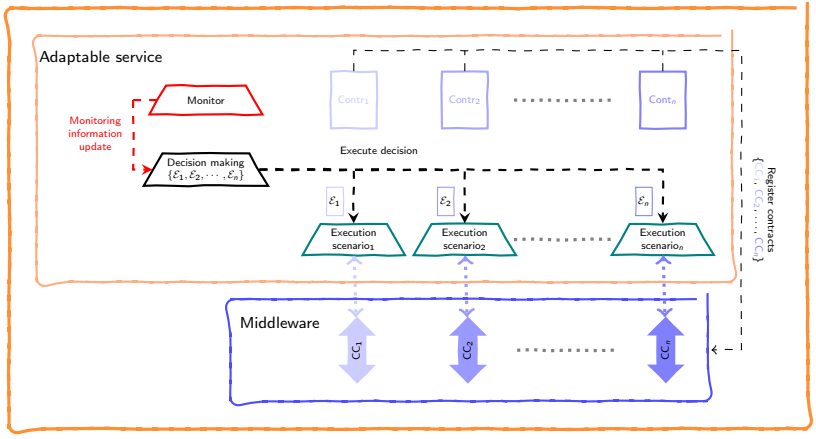


– Adaptation as service composition –

Architecture of our adaptation pattern



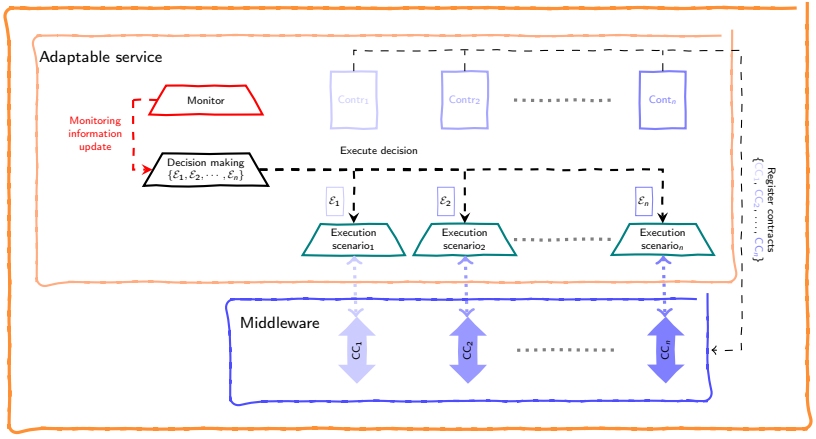
Architecture of our adaptation pattern



Middleware

- mediates interactions
- through ports' connections

Architecture of our adaptation pattern



Adaptable service

- a contract for each channel
- execution scenarios imp.
- adaptation policy

Middleware

- mediates interactions
- through ports' connections

Control loop

while ...:
  m = monitor's last update of control data
  next_scenario = *choose*(m)
  if next_scenario != running_scenario:
    stop running_scenario
    trigger next_scenario

$$m : \begin{cases} L \mapsto 2 \\ R \mapsto 3 \\ C \mapsto .5 \end{cases}$$

and e.g., rely on RCF [6, 4, 3] which uses a SMT solver

A control loop as "service combinator"

```
1  from threading import Thread
2  from ExcScenarios import ES, done
3
4  done = False
5
6  while ⟨not done⟩:
7    stop, scenario = False, None
8    i = 0
9    while scenario == None:
10     m = monitor_update()
11     if not SMT(not ℰ_i[a ↦ m(a)]_{a∈𝒜}) ==  SAT:
12       scenario = Thread(target = ES[i])
13       scenario.start()
14     else:
15       i = (i + 1) mod len(ES)
16
17   while not (done or stop):
18     old_m = m
19     m = monitor_update()
20     if not old_m == m:
21       stop = SMT(not ℰ_i[a ↦ m(a)]_{a∈𝒜}) ==  SAT
22     if stop:
23       scenario.terminate()
```

ES *list of functions each implementing an execution scenario*

done *shared for coordinated termination*

A control loop as "service combinator"

```
1  from threading import Thread
2  from ExcScenarios import ES, done
3
4  done = False
5
6  while ⟨not done⟩:
7    stop, scenario = False, None
8    i = 0
9    while scenario == None:
10     m = monitor_update()
11     if not SMT(not ℰ_i[a ↦ m(a)]_{a∈𝒜}) ==  SAT:
12       scenario = Thread(target = ES[i])
13       scenario.start()
14     else:
15       i = (i + 1) mod len(ES)
16
17   while not (done or stop):
18     old_m = m
19     m = monitor_update()
20     if not old_m == m:
21       stop = SMT(not ℰ_i[a ↦ m(a)]_{a∈𝒜}) ==  SAT
22     if stop:
23       scenario.terminate()
```

ES *list of functions each implementing an execution scenario*

done *shared for coordinated termination atomic access*

A control loop as "service combinator"

```
1  from threading import Thread
2  from ExcScenarios import ES, done
3
4  done = False
5
6  while ⟨not done⟩:
7    stop, scenario = False, None
8    i = 0
9    while scenario == None:
10     m = monitor_update()
11     if not SMT(not ℰ_i[a ↦ m(a)]_{a∈𝒜}) ==  SAT:
12       scenario = Thread(target = ES[i])
13       scenario.start()
14     else:
15       i = (i + 1) mod len(ES)
16
17   while not (done or stop):
18     old_m = m
19     m = monitor_update()
20     if not old_m == m:
21       stop = SMT(not ℰ_i[a ↦ m(a)]_{a∈𝒜}) ==  SAT
22     if stop:
23       scenario.terminate()
```

ES *list of functions each implementing an execution scenario*

done *shared for coordinated termination atomic access*

get control data

A control loop as "service combinator"

```
1  from threading import Thread
2  from ExcScenarios import ES, done
3
4  done = False
5
6  while ⟨not done⟩:
7    stop, scenario = False, None
8    i = 0
9    while scenario == None:
10     m = monitor_update()
11     if not SMT(not ℰ_i[a ↦ m(a)]_{a∈𝒜}) ==  SAT:
12       scenario = Thread(target = ES[i])
13       scenario.start()
14     else:
15       i = (i + 1) mod len(ES)
16
17   while not (done or stop):
18     old_m = m
19     m = monitor_update()
20     if not old_m == m:
21       stop = SMT(not ℰ_i[a ↦ m(a)]_{a∈𝒜}) ==  SAT
22     if stop:
23       scenario.terminate()
```

ES *list of functions each implementing an execution scenario*

done *shared for coordinated termination atomic access*

get control data

spawn the chosen scenario ES[i], *if enabled*

A control loop as "service combinator"

```
1  from threading import Thread
2  from ExcScenarios import ES, done
3
4  done = False
5
6  while ⟨not done⟩:
7    stop, scenario = False, None
8    i = 0
9    while scenario == None:
10     m = monitor_update()
11     if not SMT(not ℰ_i[a ↦ m(a)]_{a∈𝒜}) ==  SAT:
12       scenario = Thread(target = ES[i])
13       scenario.start()
14     else:
15       i = (i + 1) mod len(ES)
16
17   while not (done or stop):
18     old_m = m
19     m = monitor_update()
20     if not old_m == m:
21       stop = SMT(not ℰ_i[a ↦ m(a)]_{a∈𝒜}) ==  SAT
22     if stop:
23       scenario.terminate()
```

ES *list of functions each implementing an execution scenario*

done *shared for coordinated termination atomic access*

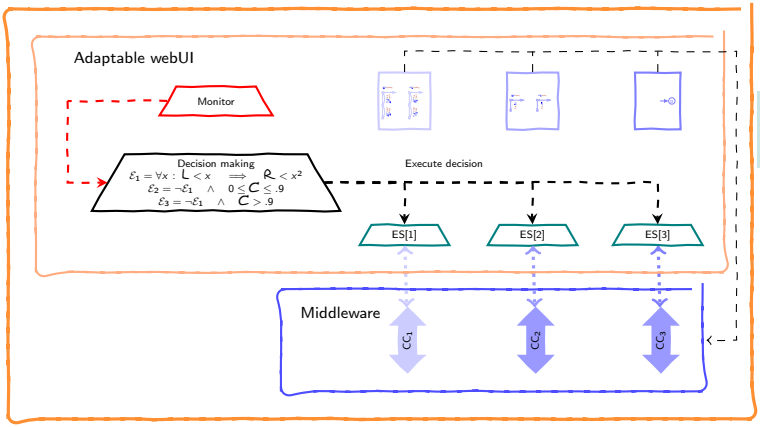
get control data

spawn the chosen scenario ES[i], *if enabled*

keep going until the scenario is done *or adaptation is needed*

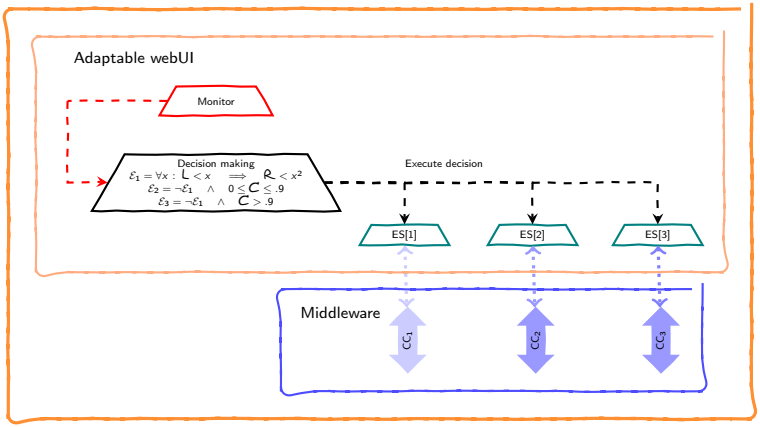
– Our pattern at work –

The pattern in action

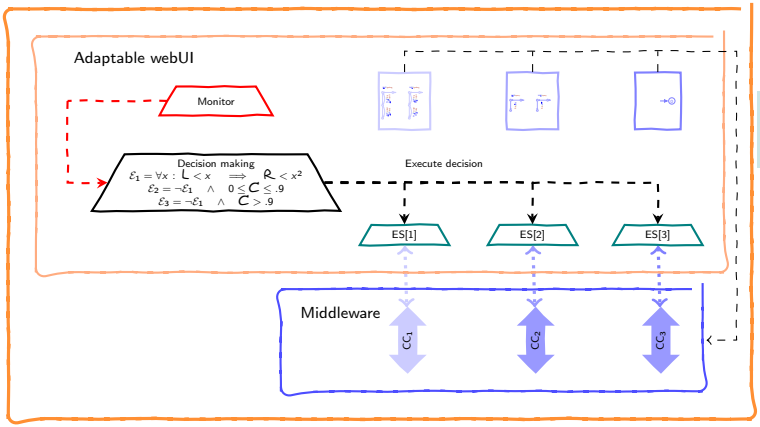


Comm. contracts registered on local middleware

The pattern in action

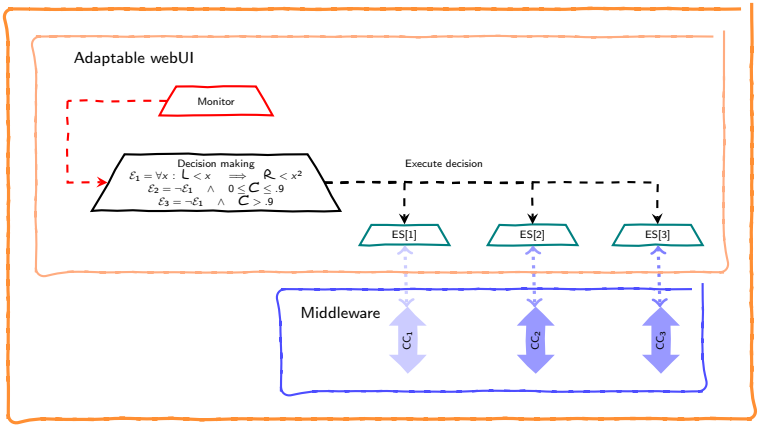


The pattern in action

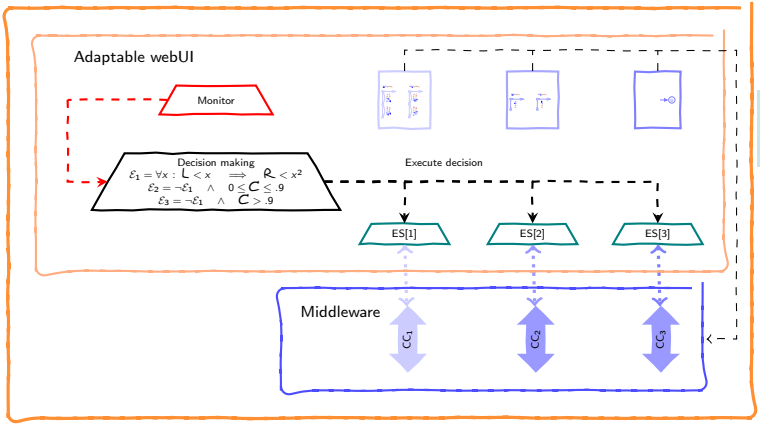


Comm. contracts registered on local middleware

The pattern in action

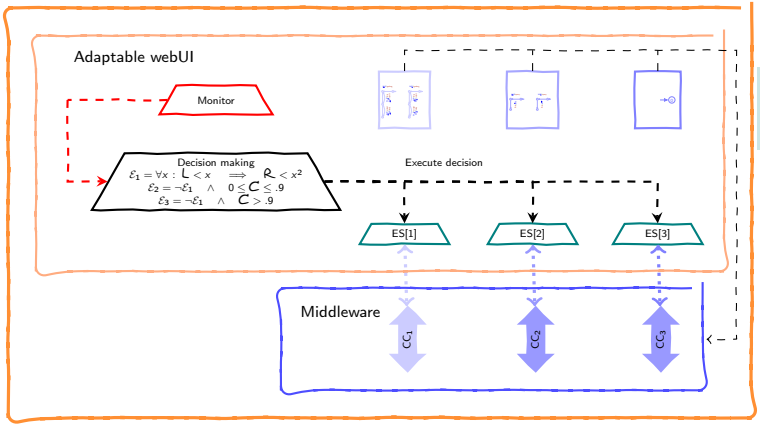


The pattern in action



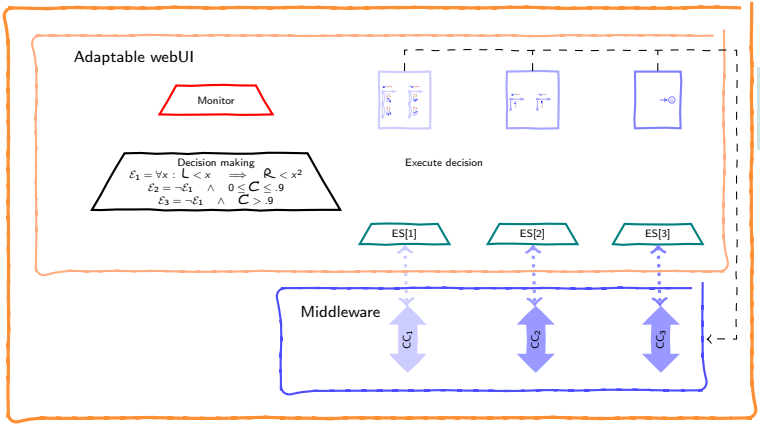
Comm. contracts registered on local middleware

The pattern in action



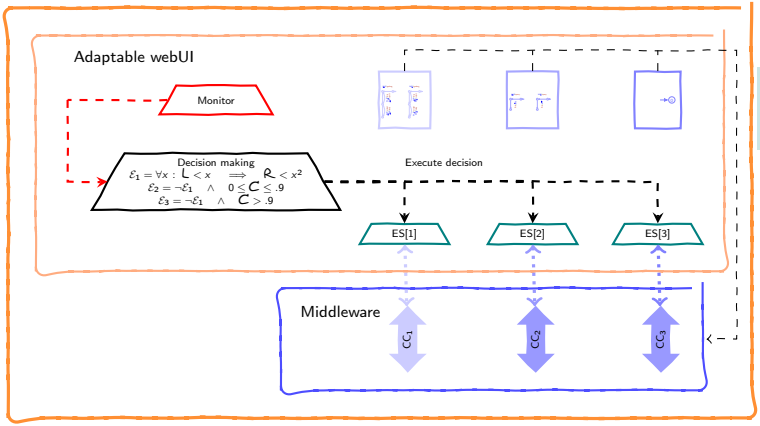
Monitor and decision procedure spawned

The pattern in action



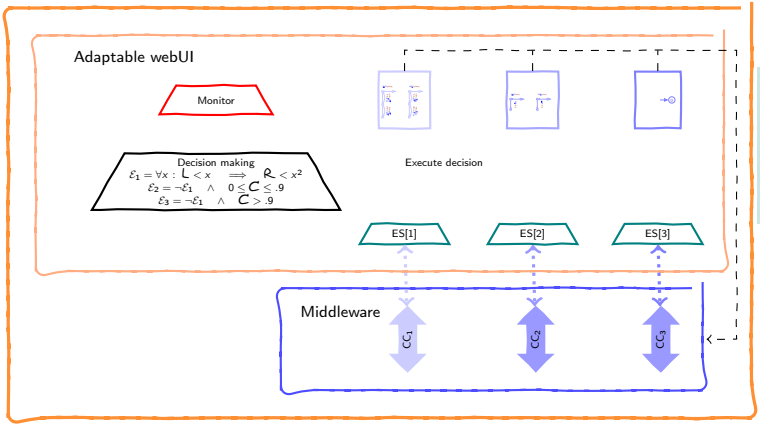
Monitor and decision procedure spawned

The pattern in action



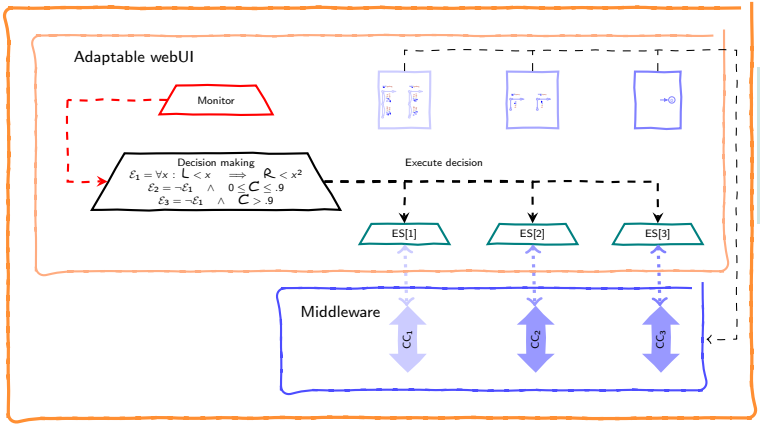
Monitor and decision procedure spawned

The pattern in action



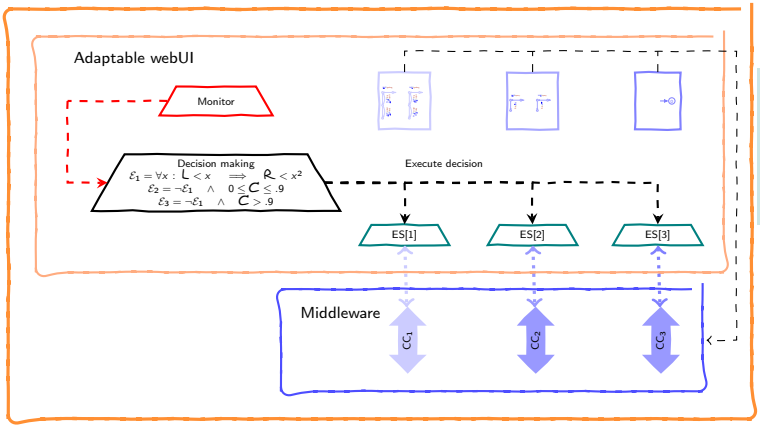
Monitor and decision procedure spawned
Control data acquired:
$\mathsf{L} \mapsto 2, \mathsf{R} \mapsto 3, \mathsf{C} \mapsto 0.54$

The pattern in action



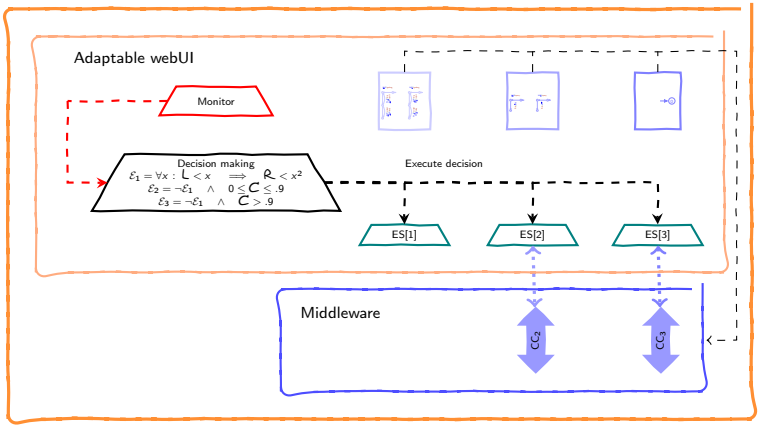
Monitor and decision procedure spawned
Control data acquired:
$\mathsf{L} \mapsto 2, \mathsf{R} \mapsto 3, \mathsf{C} \mapsto 0.54$

The pattern in action

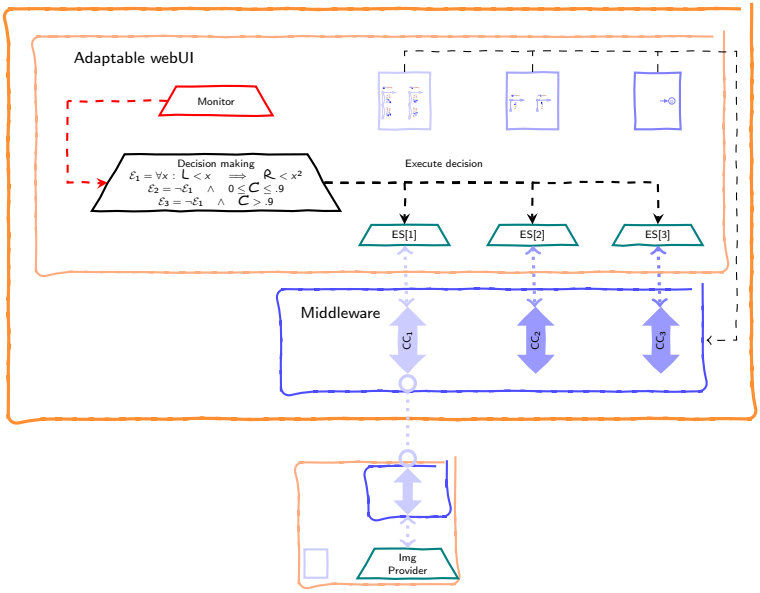


Monitor and decision procedure spawned
Control data acquired:
$\mathsf{L} \mapsto 2, \mathsf{R} \mapsto 3, \mathsf{C} \mapsto 0.54$

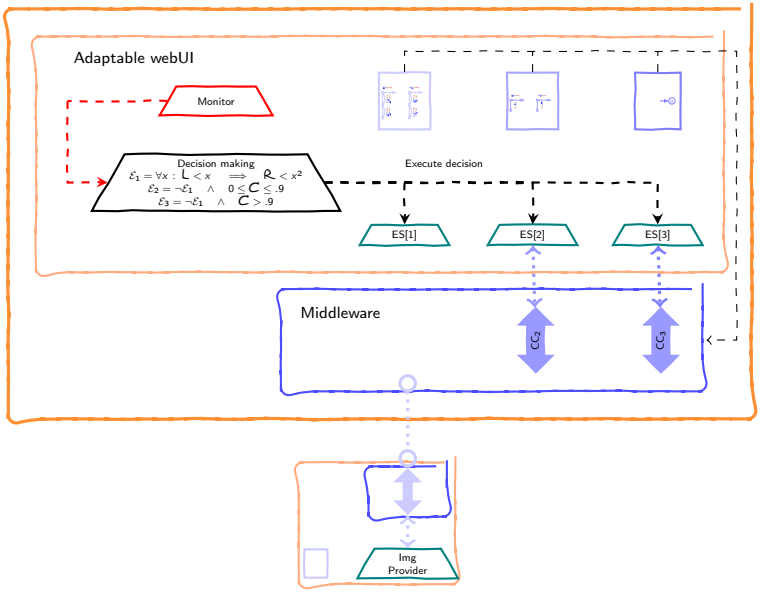
The pattern in action



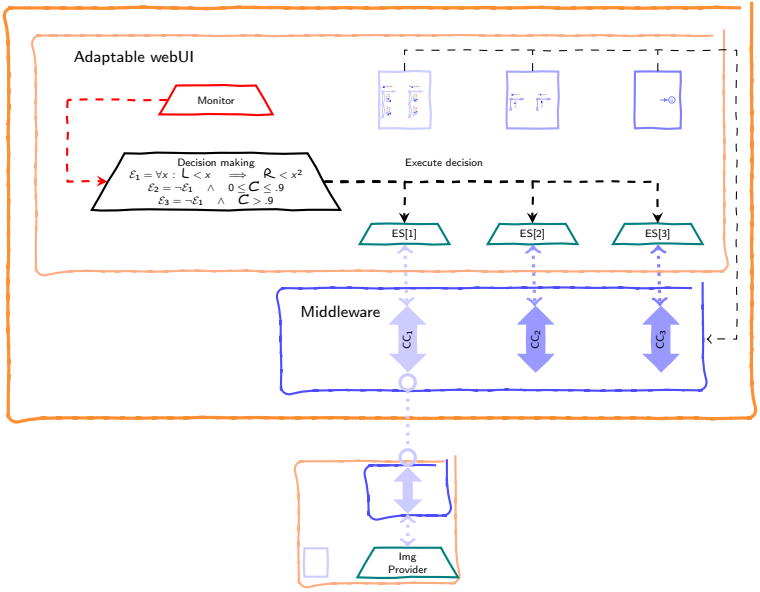
The pattern in action



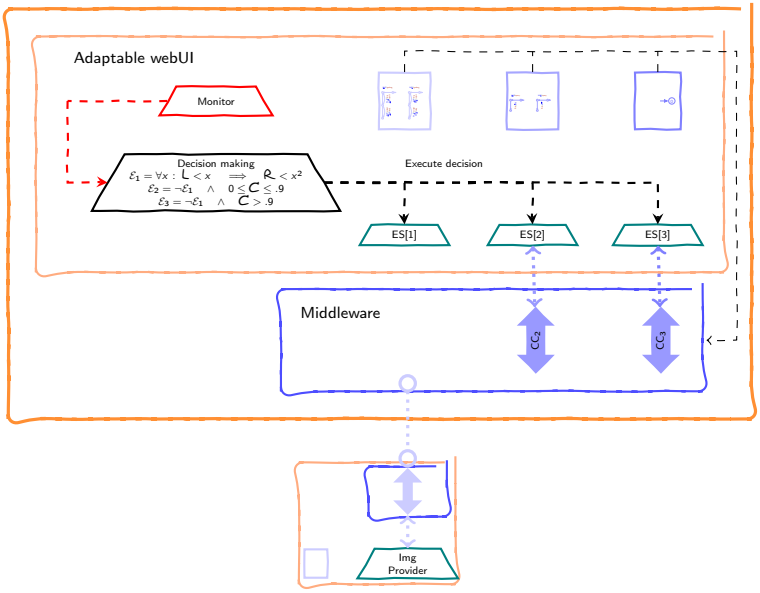
The pattern in action



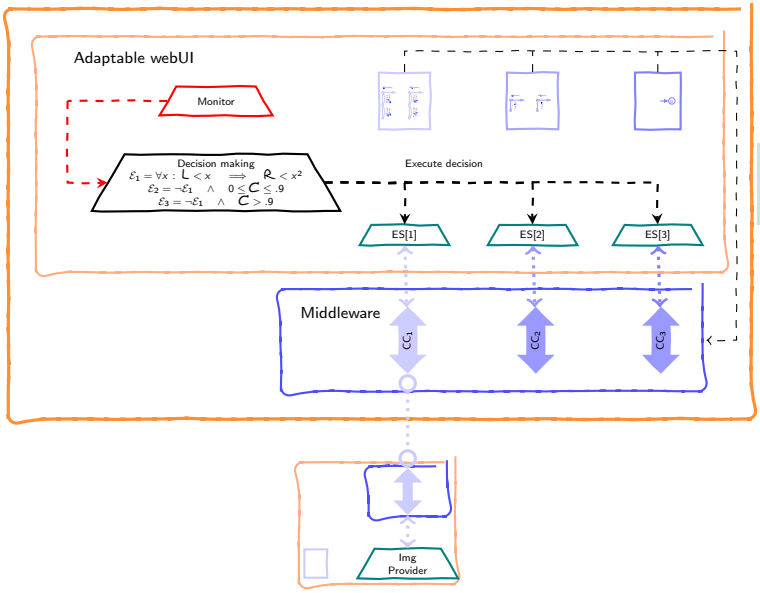
The pattern in action



The pattern in action



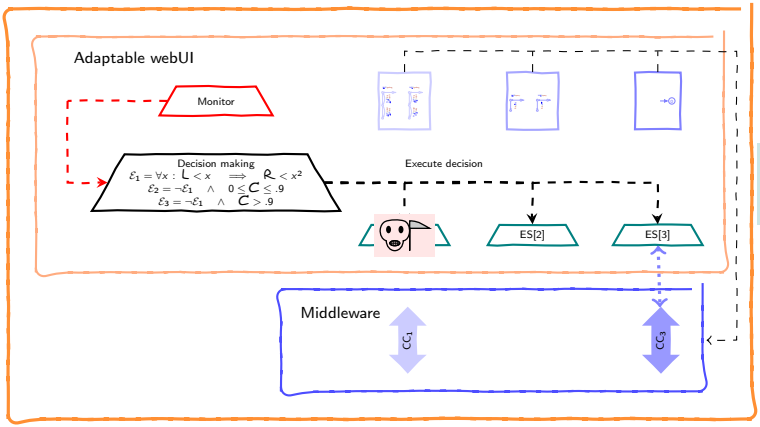
The pattern in action



Control data acquired:

$\mathsf{L} \mapsto 2, \mathsf{R} \mapsto 5, \mathsf{C} \mapsto 0.54$

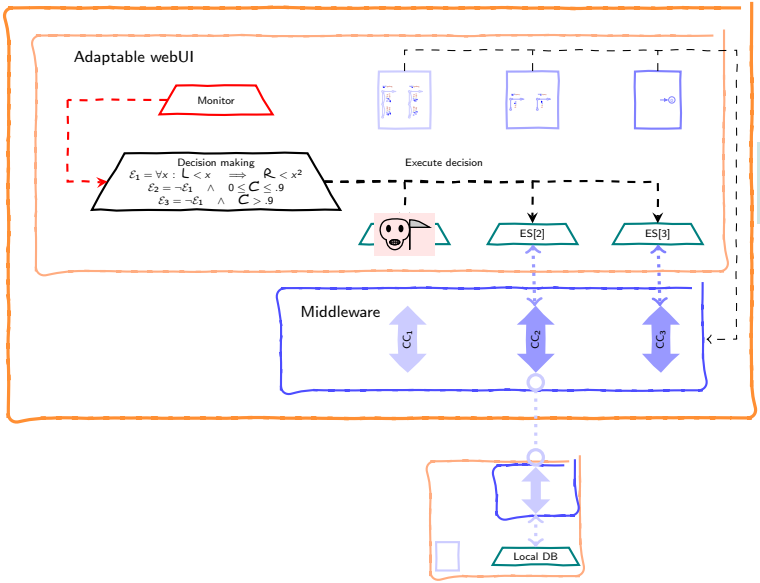
The pattern in action



Control data acquired:

$\mathsf{L} \mapsto 2, \mathsf{R} \mapsto 5, \mathsf{C} \mapsto 0.54$

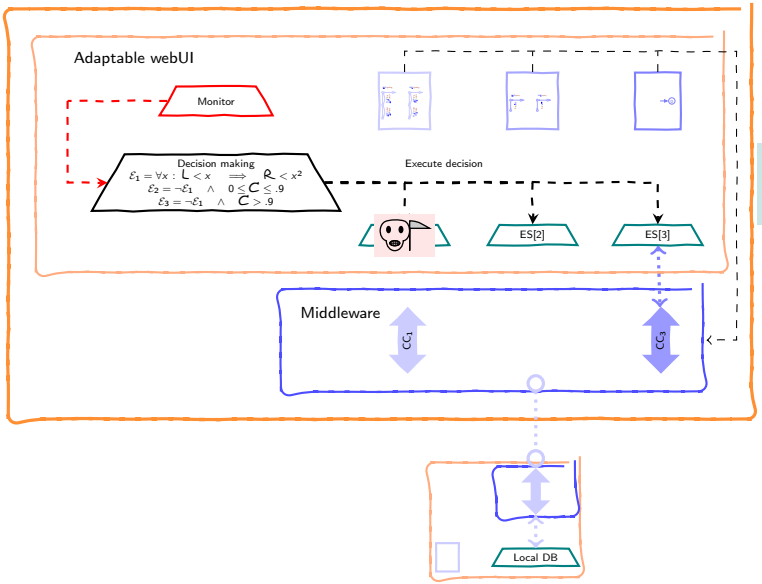
The pattern in action



Control data acquired:

$\mathsf{L} \mapsto 2, \mathsf{R} \mapsto 5, \mathsf{C} \mapsto 0.54$

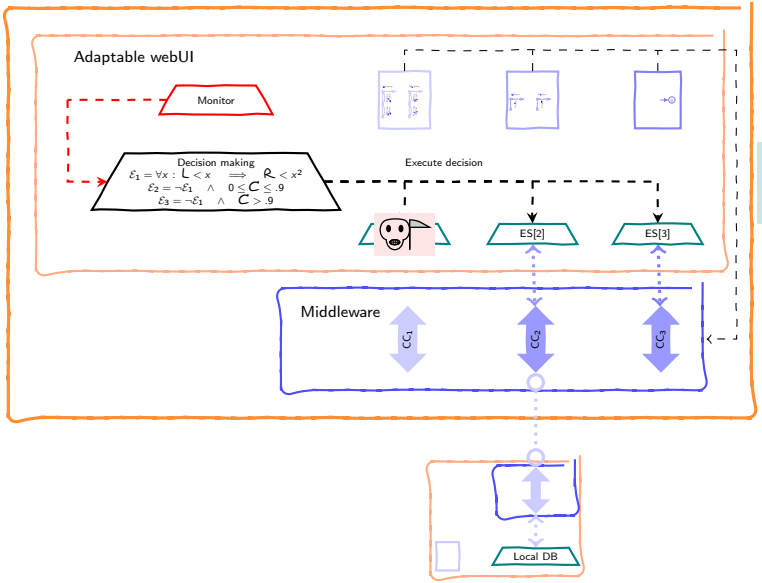
The pattern in action



Control data acquired:

$\mathsf{L} \mapsto 2, \mathsf{R} \mapsto 5, \mathsf{C} \mapsto 0.54$

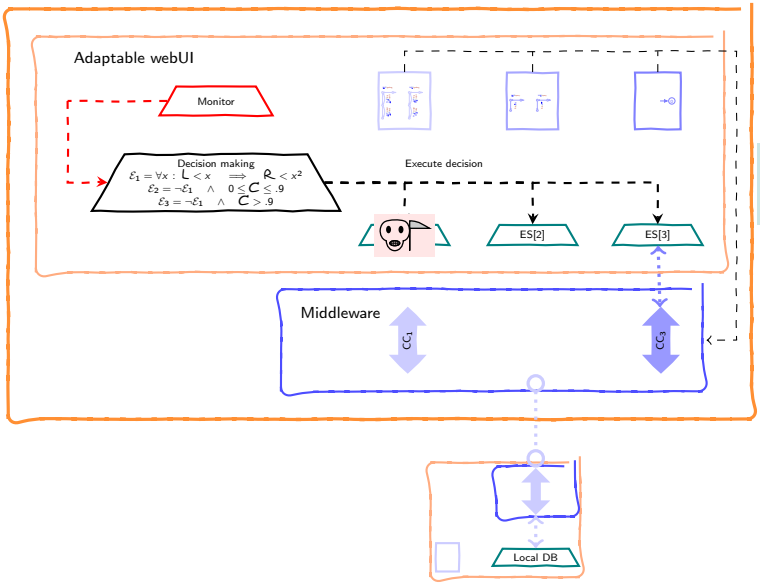
The pattern in action



Control data acquired:

$\mathsf{L} \mapsto 2, \mathsf{R} \mapsto 5, \mathsf{C} \mapsto 0.54$

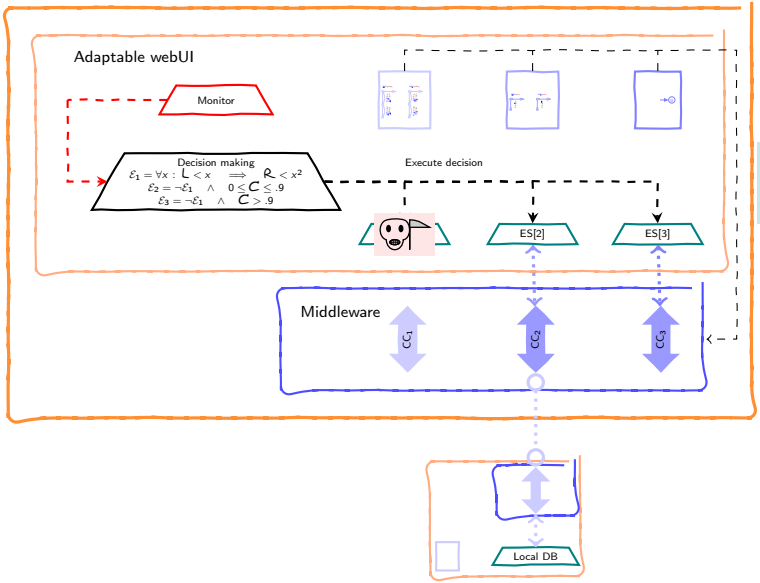
The pattern in action



Control data acquired:

$\mathsf{L} \mapsto 2, \mathsf{R} \mapsto 5, \mathsf{C} \mapsto 0.54$

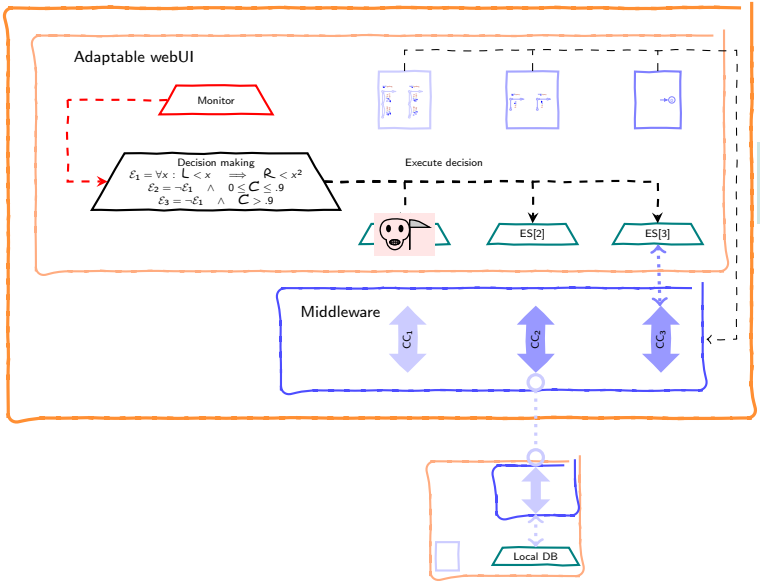
The pattern in action



Control data acquired:

$\mathsf{L} \mapsto 2, \mathsf{R} \mapsto 5, \mathsf{C} \mapsto 0.54$

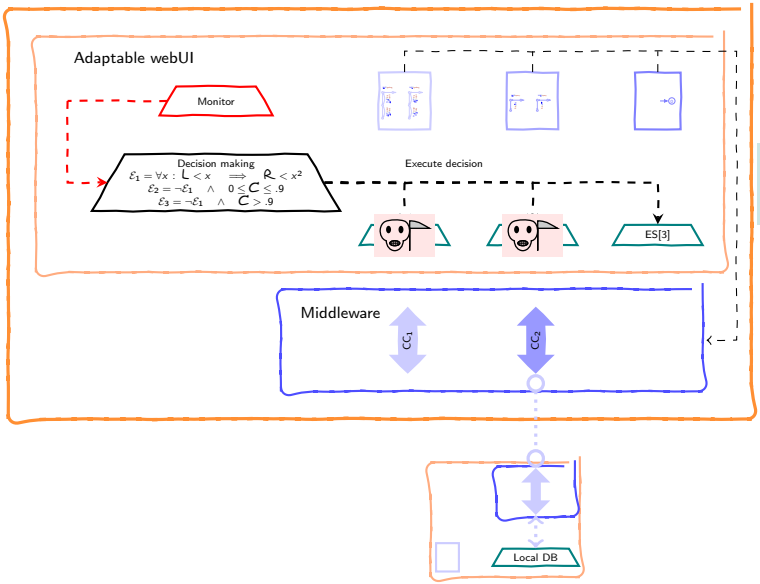
The pattern in action



Control data acquired:

$\mathsf{L} \mapsto 2, \mathsf{R} \mapsto 5, \mathsf{C} \mapsto 0.97$

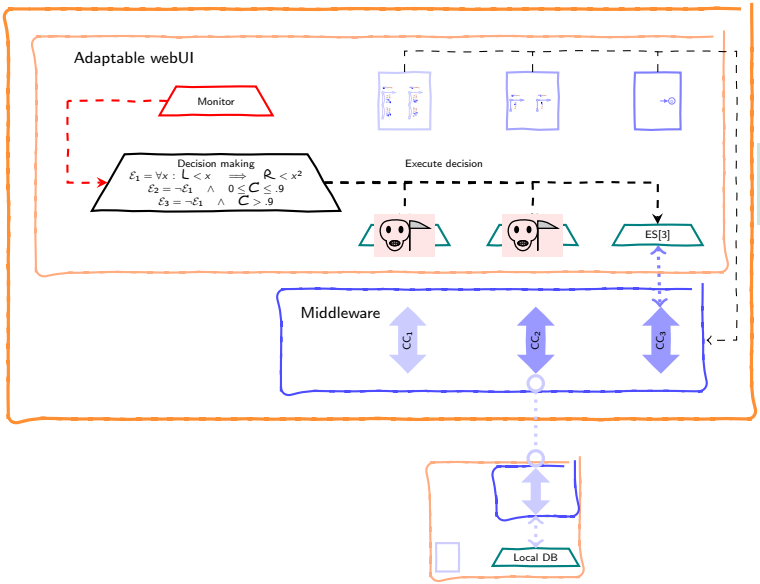
The pattern in action



Control data acquired:

$\mathsf{L} \mapsto 2, \mathsf{R} \mapsto 5, \mathsf{C} \mapsto 0.97$

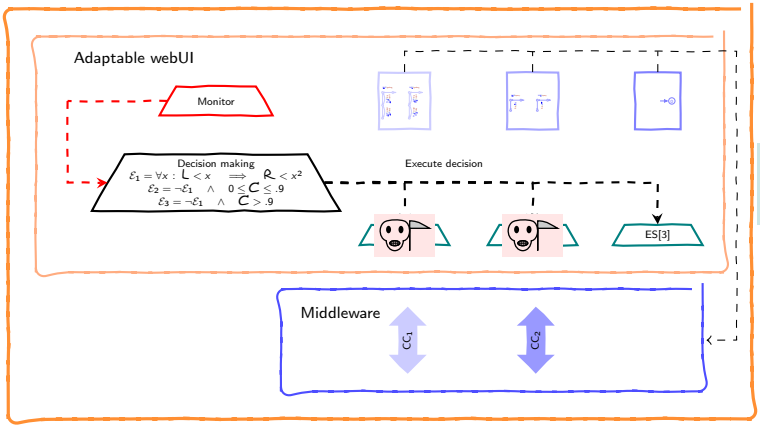
The pattern in action



Control data acquired:

$\mathsf{L} \mapsto 2, \mathsf{R} \mapsto 5, \mathsf{C} \mapsto 0.97$

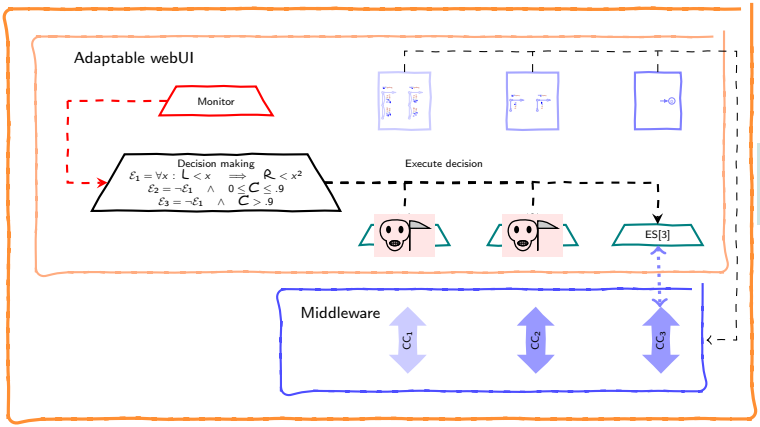
The pattern in action



Control data acquired:

$\mathsf{L} \mapsto 2, \mathsf{R} \mapsto 5, \mathsf{C} \mapsto 0.97$

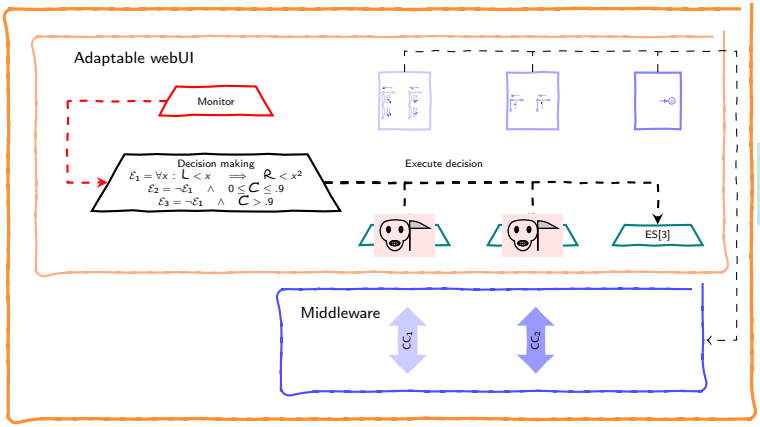
The pattern in action



Control data acquired:

$\mathsf{L} \mapsto 2, \mathsf{R} \mapsto 5, \mathsf{C} \mapsto 0.97$

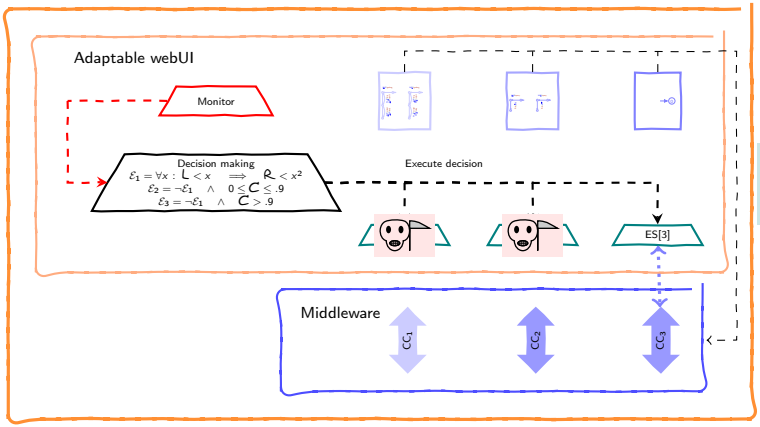
The pattern in action



Control data acquired:

$L \mapsto 2, R \mapsto 5, C \mapsto 0.97$

The pattern in action



Control data acquired:

$\mathsf{L} \mapsto 2, \mathsf{R} \mapsto 5, \mathsf{C} \mapsto 0.97$

– Epilogue –

Wrapping up

Wrapping up

- Runtime architectural adaptation as service combinators relying on
  - separation of concerns (monitoring + decision making)
  - semantic compliance

Wrapping up

- Runtime architectural adaptation as service combinators relying on
  - separation of concerns (monitoring + decision making)
  - semantic compliance

- SEArch provides tool support
  - offers the basic infrastructure
  - uniformly covers many scenarios

Wrapping up

- Runtime architectural adaptation as service combinators relying on
  - separation of concerns (monitoring + decision making)
  - semantic compliance

- SEArch provides tool support
  - offers the basic infrastructure
  - uniformly covers many scenarios

- Take a look at the paper for
  - considerations about the control loop
  - details about adaptation of the running example

Future work

Some limitations due to

Future work

Some limitations due to

- bisimulation is too a strong relation
  - move to trace-based compliance or beh. subtyping

Future work

Some limitations due to

- bisimulation is too a strong relation
  - move to trace-based compliance or beh. subtyping
- exogenous control data only
  - add endogenous control data
    - entangling the monitor in the communication contracts
    - QoS-based adaptation (e.g., image quality degradation taking advantage of RCF)

Future work

Some limitations due to

- bisimulation is too a strong relation
  - move to trace-based compliance or beh. subtyping
- exogenous control data only
  - add endogenous control data
    - entangling the monitor in the communication contracts
    - QoS-based adaptation (e.g., image quality degradation taking advantage of RCF)
- more general combinators (e.g., monitor passing data to execution scenarios)

Future work

Some limitations due to

- bisimulation is too a strong relation
  - move to trace-based compliance or beh. subtyping
- exogenous control data only
  - add endogenous control data
    - entangling the monitor in the communication contracts
    - QoS-based adaptation (e.g., image quality degradation taking advantage of RCF)
- more general combinators (e.g., monitor passing data to execution scenarios)
- abrupt termination
  - move to graceful adaptation (e.g., ensuring transactional behaviour)

THANK YOU!

THANK YOU!

R1: "the only way we have to defend ourselves from AI frauds is to be completely transparent about our tools"

- Indeed! SEArch is available at https://github.com/pmontepagano/search, cf. [5]

THANK YOU!

R1: "the only way we have to defend ourselves from AI frauds is to be completely transparent about our tools"

- Indeed! SEArch is available at https://github.com/pmontepagano/search, cf. [5]

R2: "why for each extended CFSM in the contract the number of candidates is the same?" & "Why a list instead of a set of candidates?"

- at each iteration it is different & bad wording (we do mean 'sets')...sorry

THANK YOU!

R1: "the only way we have to defend ourselves from AI frauds is to be completely transparent about our tools"

- Indeed! SEArch is available at https://github.com/pmontepagano/search, cf. [5]

R2: "why for each extended CFSM in the contract the number of candidates is the same?" & "Why a list instead of a set of candidates?"

- at each iteration it is different & bad wording (we do mean 'sets')...sorry

R3: Presentation

- Apologies (it was a rush)...thank for the great help in improving the presentation


Thank you!

References I

[1] Simon Bliudze, Giuseppe de Palma, Saverio Giallorenzo, Ivan Lanese, Gianluigi Zavattaro, and Brice Arleon Zemtsop Ndadji. Adaptable TeaStore.
On-line, 2024. Available at https://arxiv.org/abs/2412.16060.

[2] Roberto Bruni, Andrea Corradini, Fabio Gadducci, Alberto Lluch Lafuente, and Andrea Vandin. A conceptual framework for adaptation.
In Juan de Lara and Andrea Zisman, editors, *Fundamental Approaches to Software Engineering*, pages 240–254, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[3] Carlos G. Lopez Pombo, Agustín Eloy Martinez Suñé, and Emilio Tuosto. A dynamic temporal logic for quality of service in choreographic models.
In Erika Ábrahám, Clemens Dubslaff, and Silvia Lizeth Tapia Tarifa, editors, *Proceedings of 20th International Colloquium on Theoretical Aspects of Computing - ICTAC 2023*, volume 14446 of *LNCS*, pages 119–138, Lima, Perú, December 2023. Springer.

[4] Carlos G. Lopez Pombo, Agustín Eloy Martinez Suñé, and Emilio Tuosto. Automated static analysis of quality of service properties of communicating systems.
In André Platzer, Kristin Yvonne Rozier, Matteo Pradella, and Matteo Rossi, editors, *Proceedings of 26th. International Symposium on Formal Methods (FM 2024) - Part II*, volume 14934 of *LNCS*, pages 84–103. Springer, September 2025.

[5] Carlos Gustavo López Pombo, Pablo Montepagano, and Emilio Tuosto. Search: An execution infrastructure for service-based software systems.
In Ilaria Castellani and Francesco Tiezzi, editors, *Coordination Models and Languages - 26th IFIP WG 6.1 International Conference, COORDINATION 2024, Held as Part of the 19th International Federated Conference on Distributed Computing Techniques, DisCoTec 2024, Groningen, The Netherlands, June 17-21, 2024, Proceedings*, volume 14676 of *Lecture Notes in Computer Science*, pages 314–330. Springer, 2024.

[6] Carlos López Pombo, Agustín E. Martinez Suñé, and Emilio Tuosto. A dynamic temporal logic for quality of service in choreographic models.
TCS, 1043:115247, 2025.

References II

[7] Jóakim von Kistowski, Simon Eismann, Norbert Schmitt, André Bauer, Johannes Grohmann, and Samuel Kounev. Teastore: A micro-service reference application for benchmarking, modeling and resource management research.
In *26th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2018)*, pages 223–236. IEEE Computer Society, September 2018.