Efficient Utility-Driven Self-Healing Employing Adaptation Rules for Large Dynamic Architectures

Sona Ghahremani

&

Holger Giese, Thomas Vogel

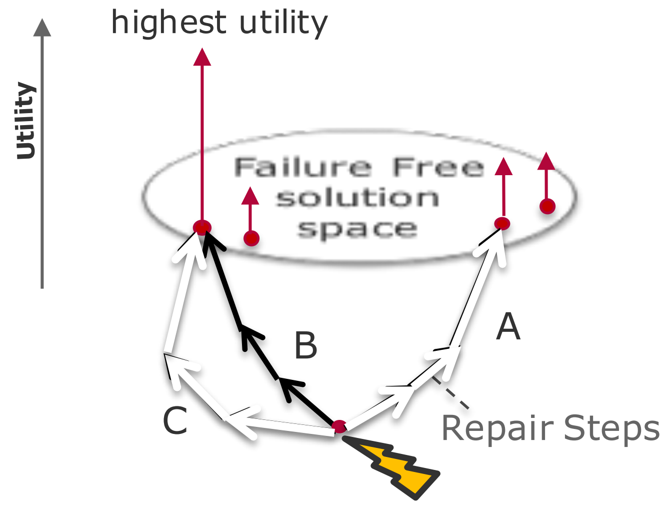
Hasso Plattner Institute, Potsdam, Germany

sona.ghahremani@hpi.de

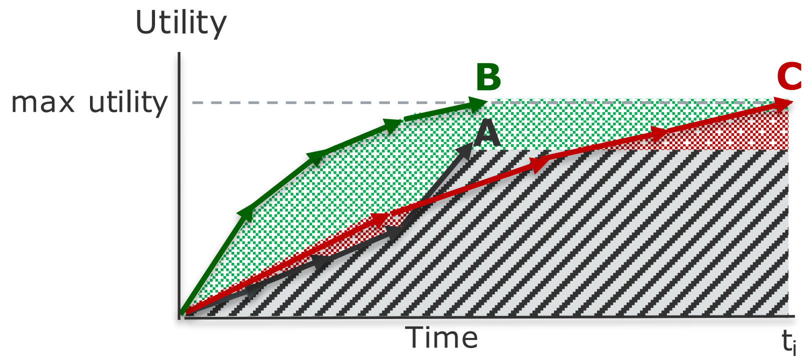
Overview

- **Direction**
  - Focusing on self-healing among all the self-* properties
  - Targeting architectural self-healing
  - Linking adaptation rules to utility
  - Defining architectural utility for dynamic architectures
- **Implementation**
  - MAPE-K Feedback loop maintains a runtime model representing the architecture of the system under adaptation
  - Employing MDE techniques such as model transformation
- **Evaluation**
  - mRUBiS as case study: an online marketplace modeled after eBay [RUBiS]

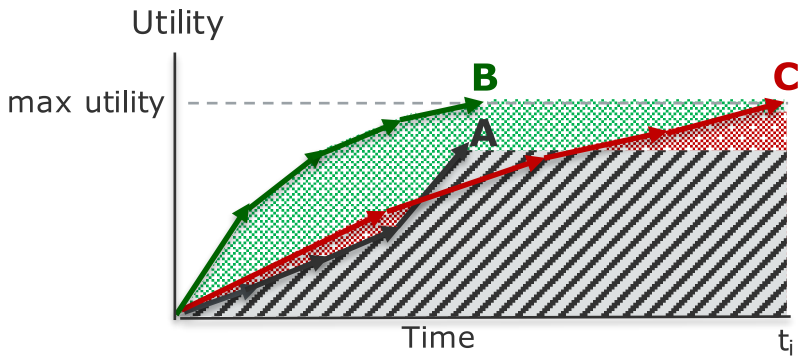
Motivation: Linking Adaptation Rules to Utility



- Sequence of repair steps -> **failure free solution space**
- Final configuration with **highest utility**
- **Optimal order of repairs-> highest Reward**



Motivation: Combining Two Ends of the Spectrum



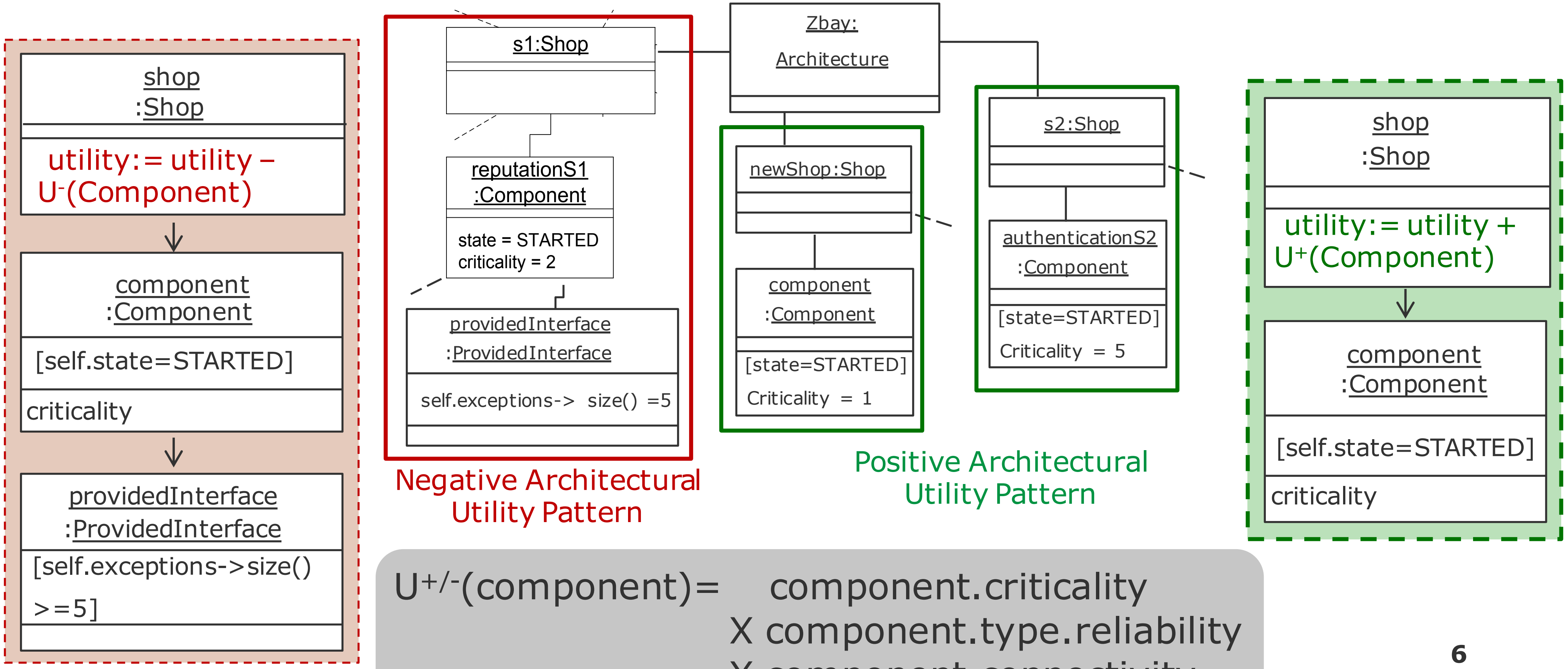
+

	Optimization -based (C)	Rule -based (A)	
Optimal order of repairs	✓	✗	✓
Scalable	✗	✓	✓
Maximum Utility	✓	✗	✓
Expressiveness	+	–	+/–

Assumptions

- **A1:** Considering only repair rules that are triggered by failures in contrast to optimization rules
- **A2**: The repair rules are effective in healing the failures and therefore executing them achieves the intended improvement of the utility
- **A3:** Rules are independent of each other with respect to their applicability and their impacts on the overall utility

Defining Pattern-based Utility

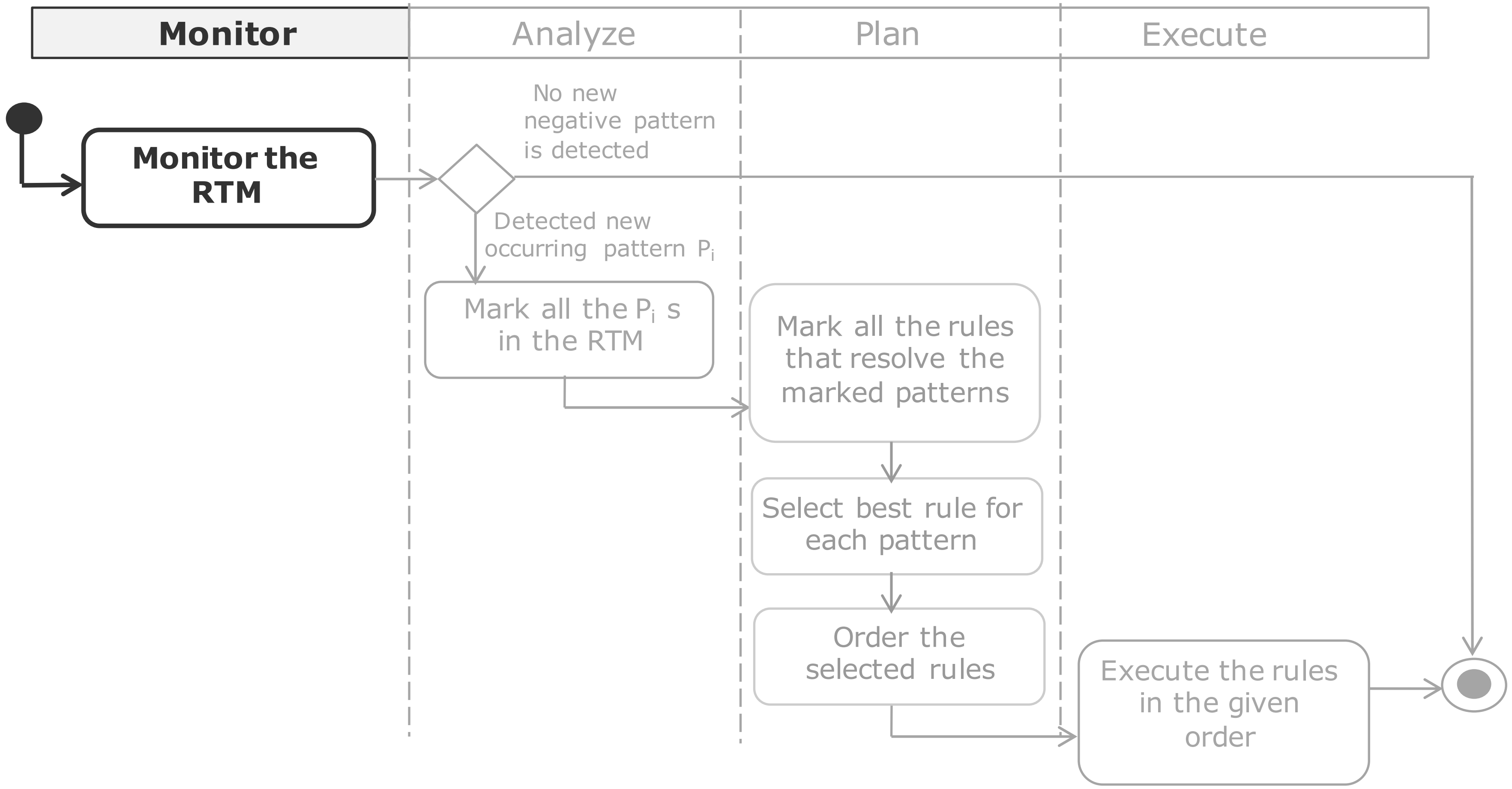


Negative Architectural Utility Pattern

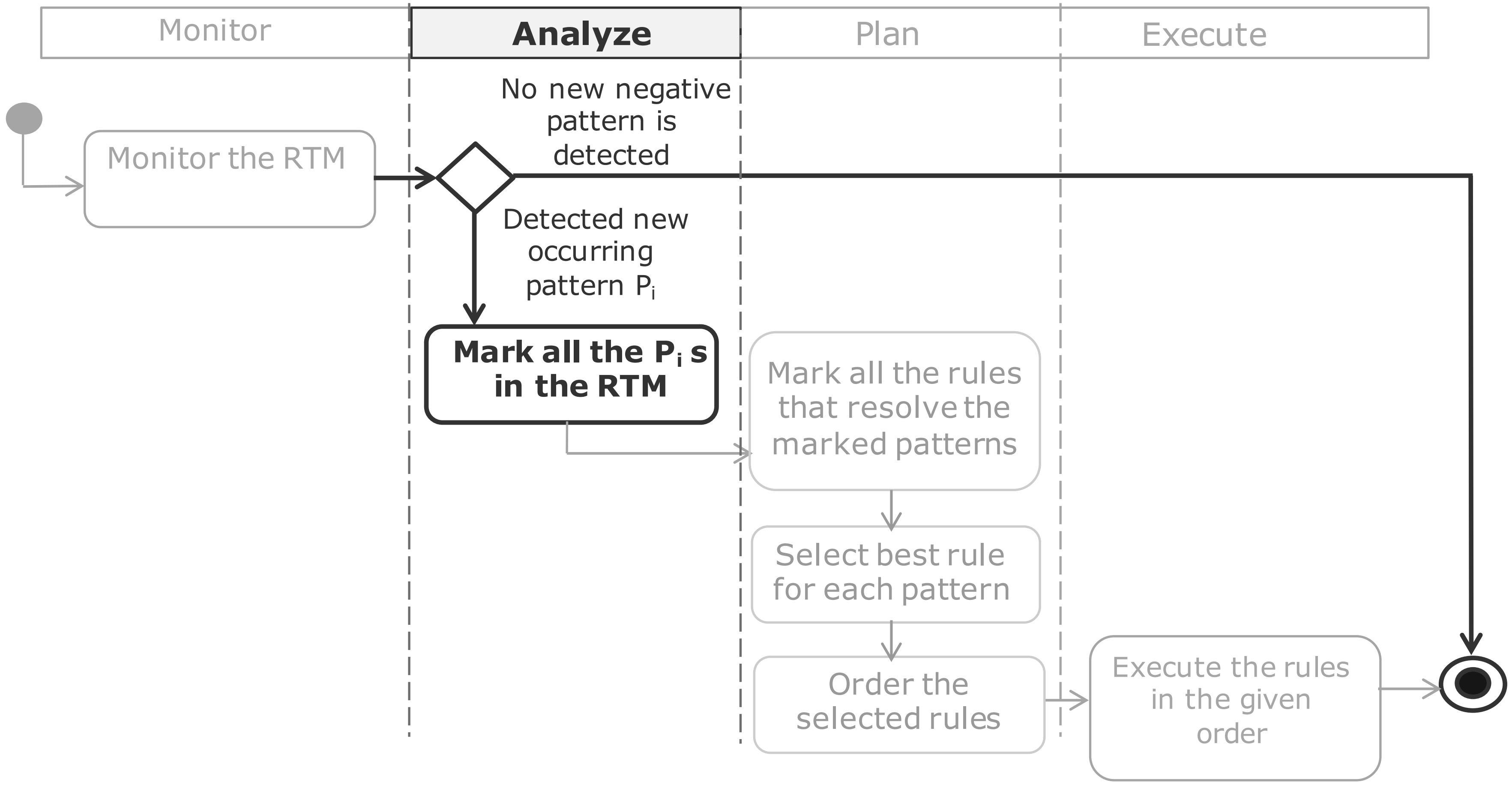
Positive Architectural Utility Pattern

$$U^{+/-}(\text{component}) = \text{component.criticality} \times \text{component.type.reliability} \times \text{component.connectivity}$$

Monitor

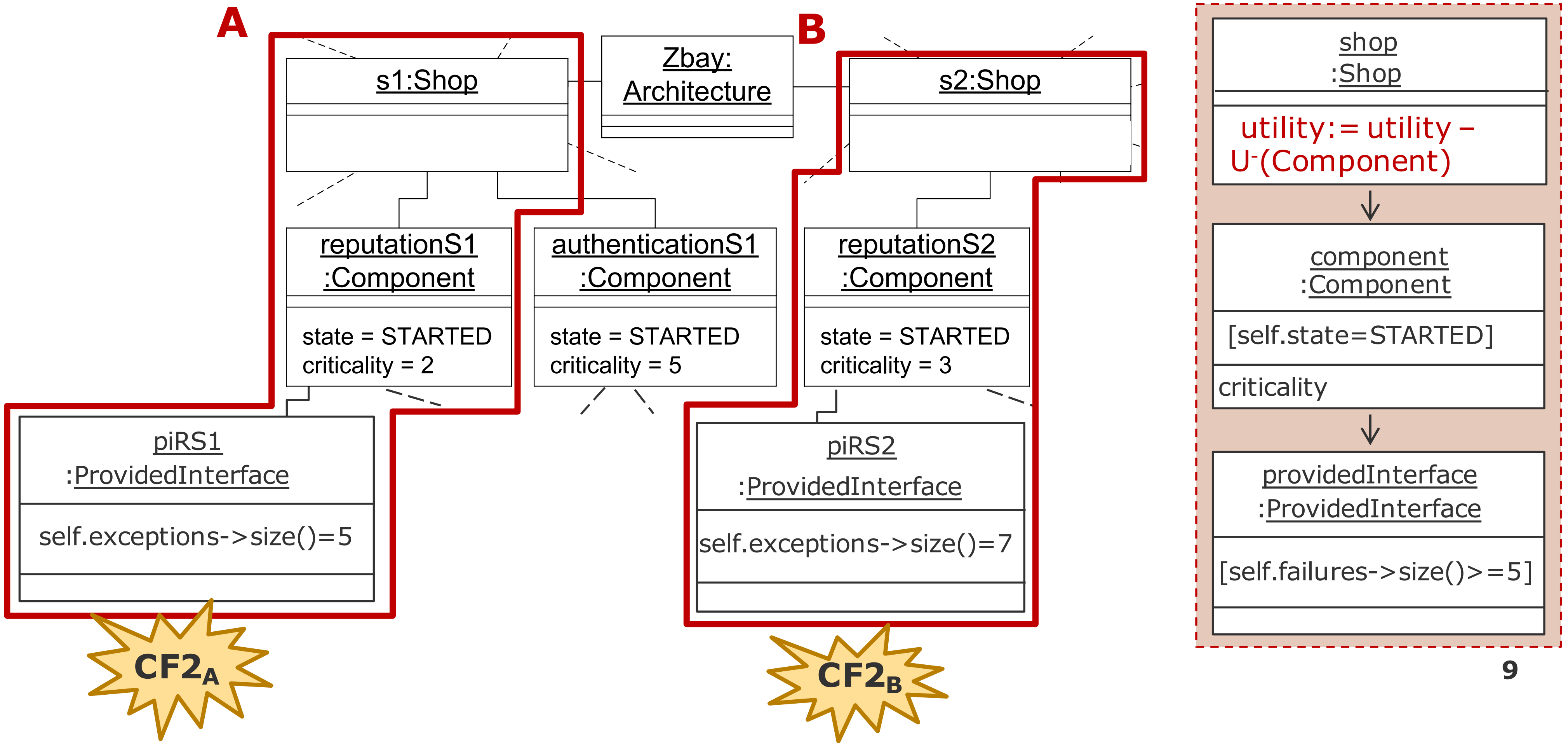

Monitor
Analyze
Plan
Execute
Monitor the RTM
No new negative pattern is detected
Detected new occurring pattern P_i
Mark all the P_i s in the RTM
Mark all the rules that resolve the marked patterns
Select best rule for each pattern
Order the selected rules
Execute the rules in the given order

Analyze

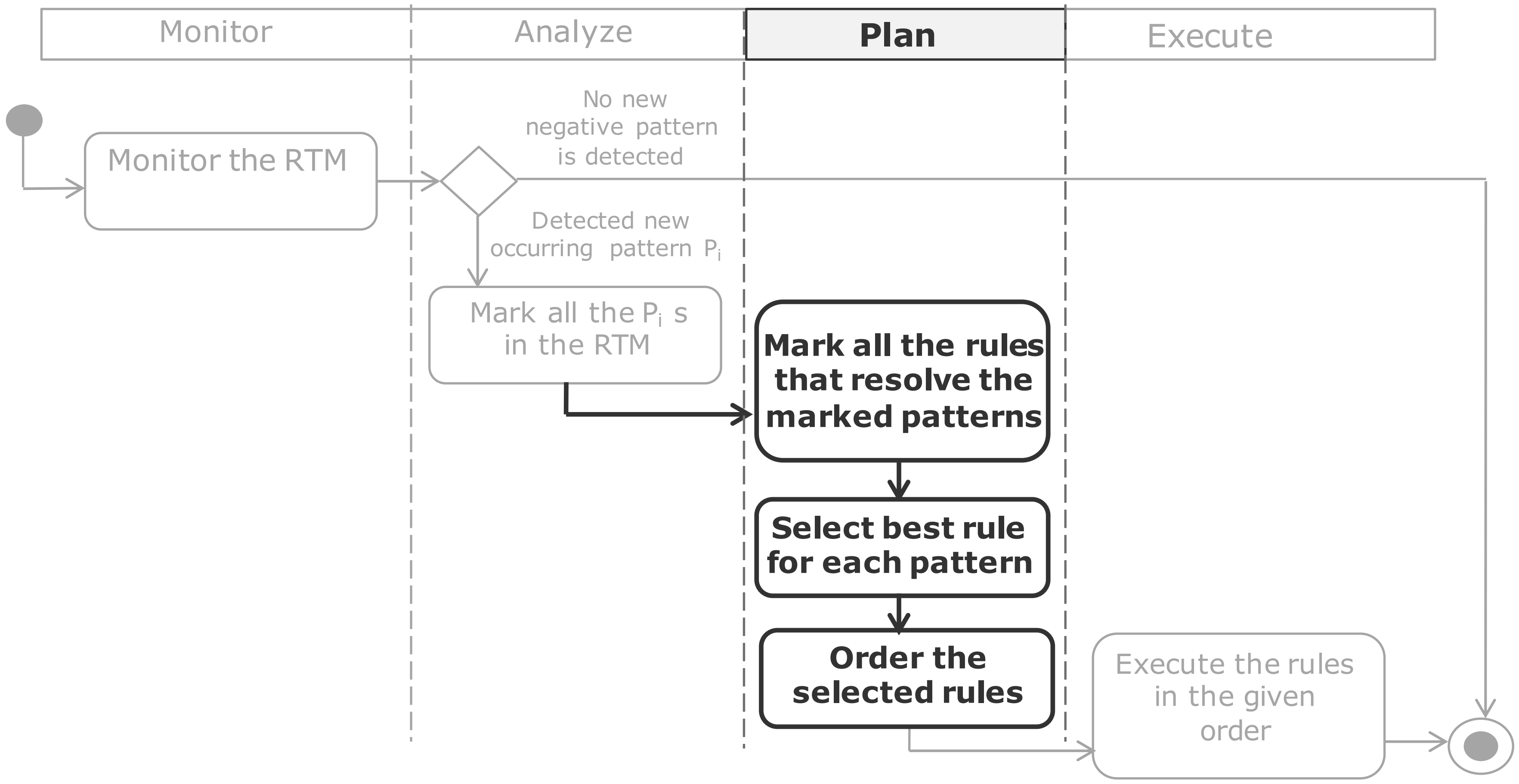


Monitor
Analyze
Plan
Execute
Monitor the RTM
No new negative pattern is detected
Detected new occurring pattern P_i
Mark all the P_i s in the RTM
Mark all the rules that resolve the marked patterns
Select best rule for each pattern
Order the selected rules
Execute the rules in the given order

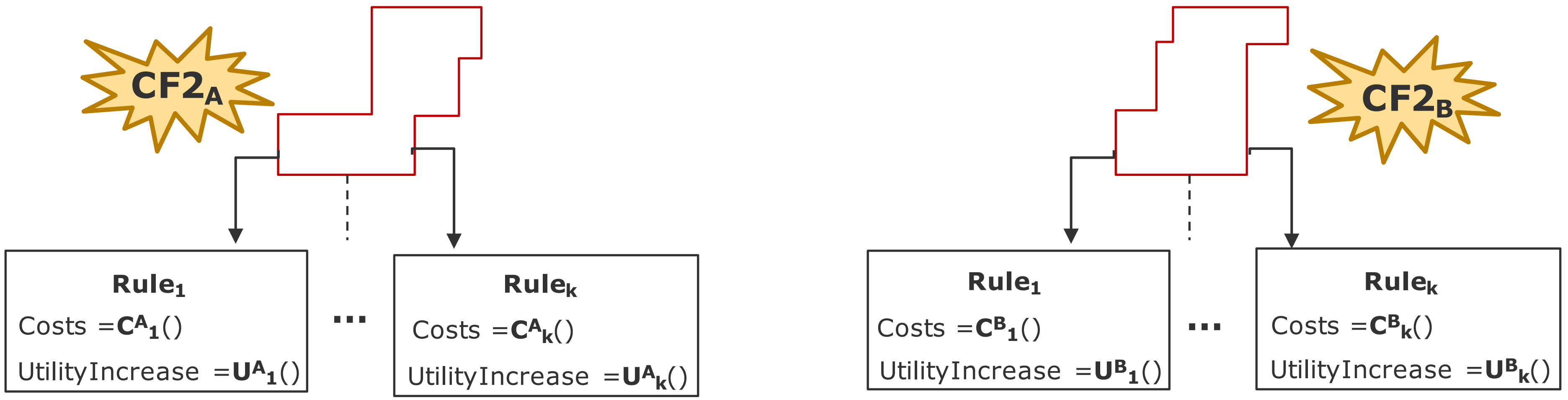
Analyzing the Patterns



Plan


Monitor
Analyze
Plan
Execute
Monitor the RTM
No new negative pattern is detected
Detected new occurring pattern P_i
Mark all the P_i s in the RTM
Mark all the rules that resolve the marked patterns
Select best rule for each pattern
Order the selected rules
Execute the rules in the given order

Two Step Planning



$CF2_A$
$Rule_1$
Costs $= \mathbf{C^A_1}()$
UtilityIncrease $= \mathbf{U^A_1}()$
...
$Rule_k$
Costs $= \mathbf{C^A_k}()$
UtilityIncrease $= \mathbf{U^A_k}()$
$CF2_B$
$Rule_1$
Costs $= \mathbf{C^B_1}()$
UtilityIncrease $= \mathbf{U^B_1}()$
...
$Rule_k$
Costs $= \mathbf{C^B_k}()$
UtilityIncrease $= \mathbf{U^B_k}()$

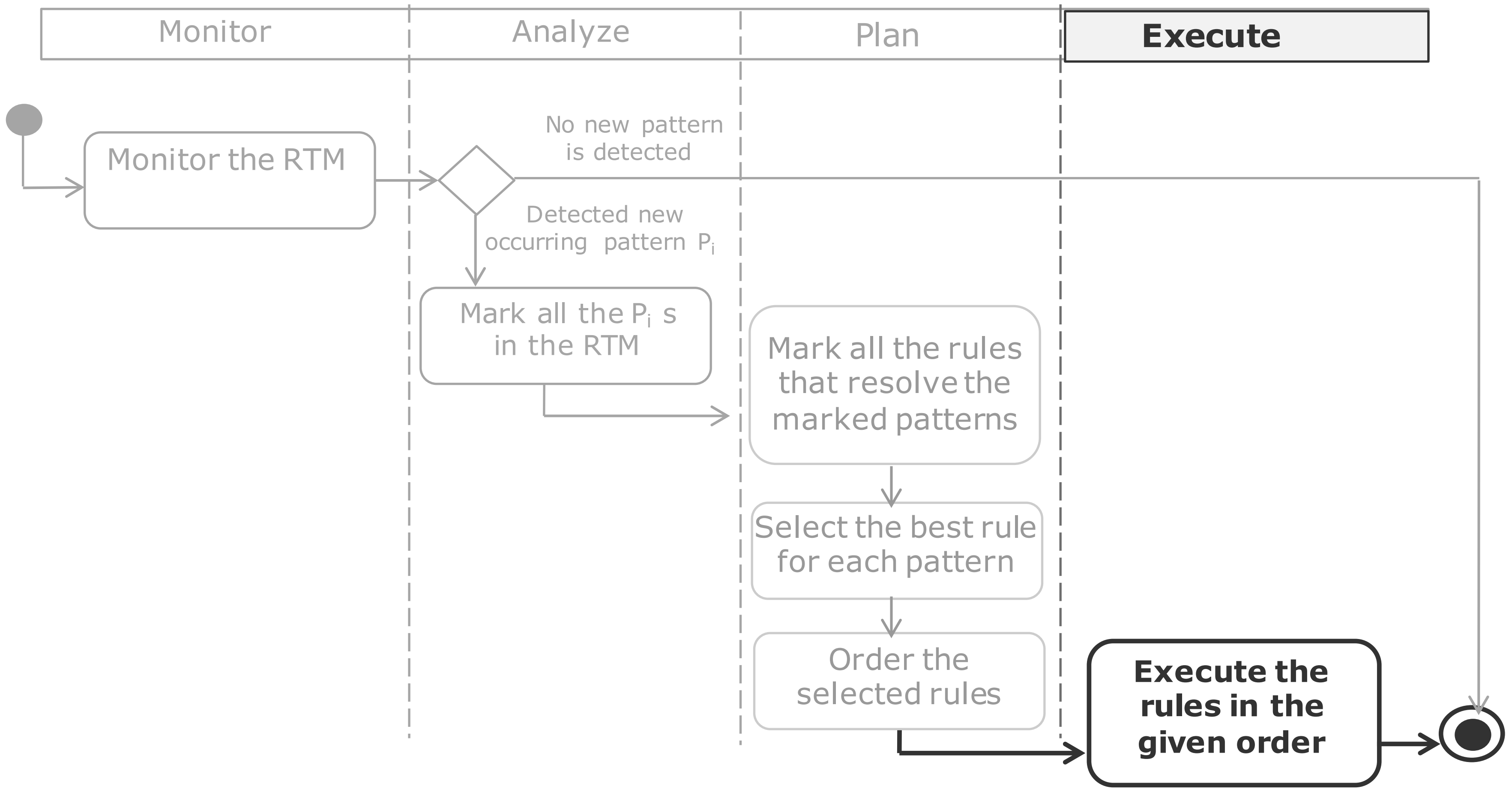
1. Select the rule which has the **max UtilityIncrease**

Select the rule which has the **max UtilityIncrease**

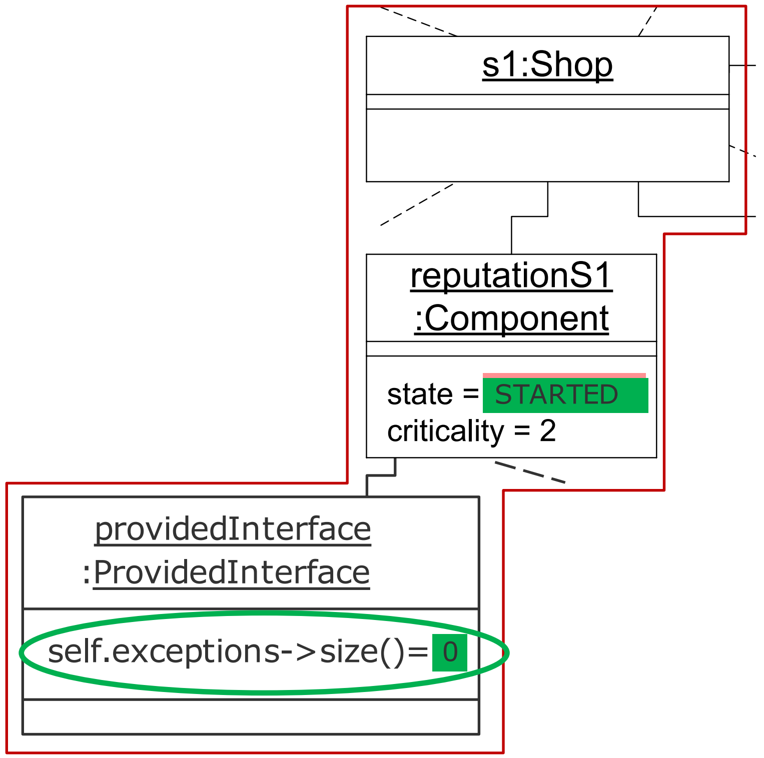
2. Order the selected Rules

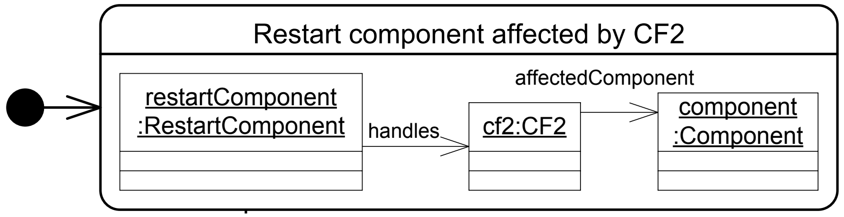
$$\frac{U^A{}_{max}()}{C^A()} > \frac{U^B{}_{max}()}{C^B()} > \dots$$

Execute


Monitor
Analyze
Plan
Execute
Monitor the RTM
No new pattern is detected
Detected new occurring pattern P_i
Mark all the P_i s in the RTM
Mark all the rules that resolve the marked patterns
Select the best rule for each pattern
Order the selected rules
Execute the rules in the given order

Executing the Rules

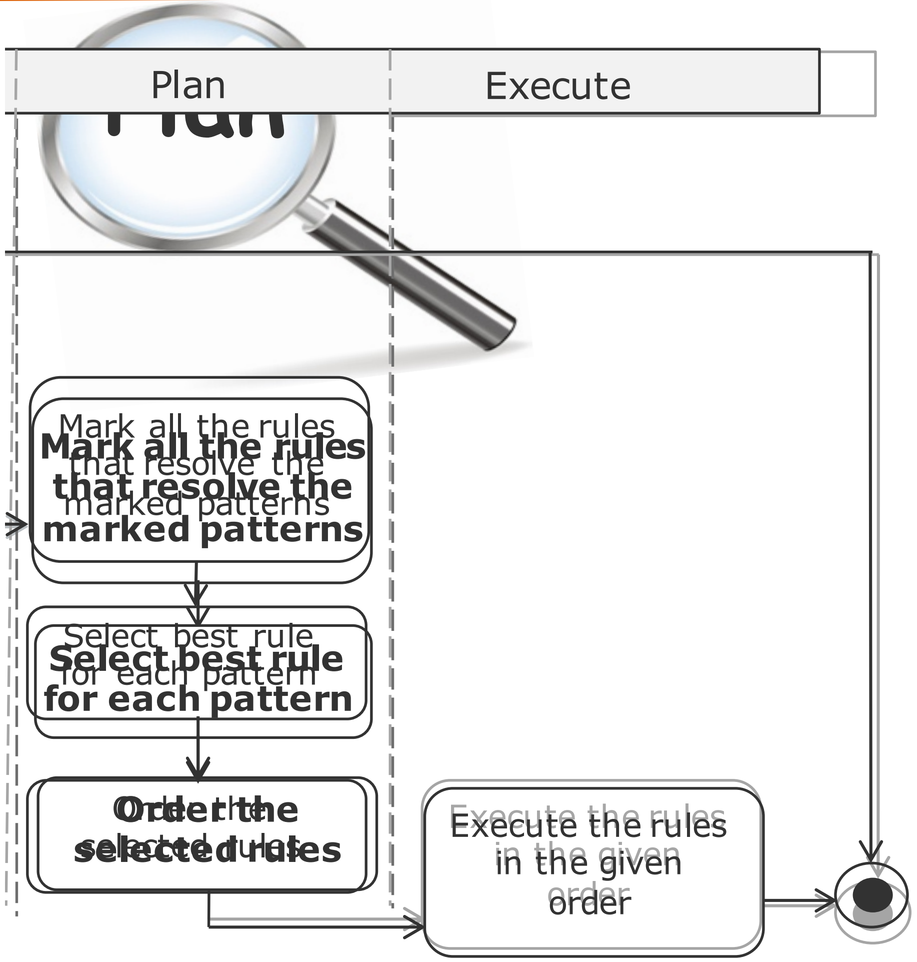




What do We Evaluate?

Variants:

- Rule-based Approach: A static rule-based approach employing static priorities and assignments without any utility function
- Optimization-based Approach: IBM ILOG CPLEX constraint solver optimizing an objective function at runtime [IBM ILoG]
- Utility-driven Approach (our approach): computing the impact of different adaptation rules at runtime using a utility function


Plan
Execute
Mark all the rules that resolve the marked patterns
Select best rule for each pattern
Order the selected rules
Execute the rules in the given order

Scalability of the Approaches

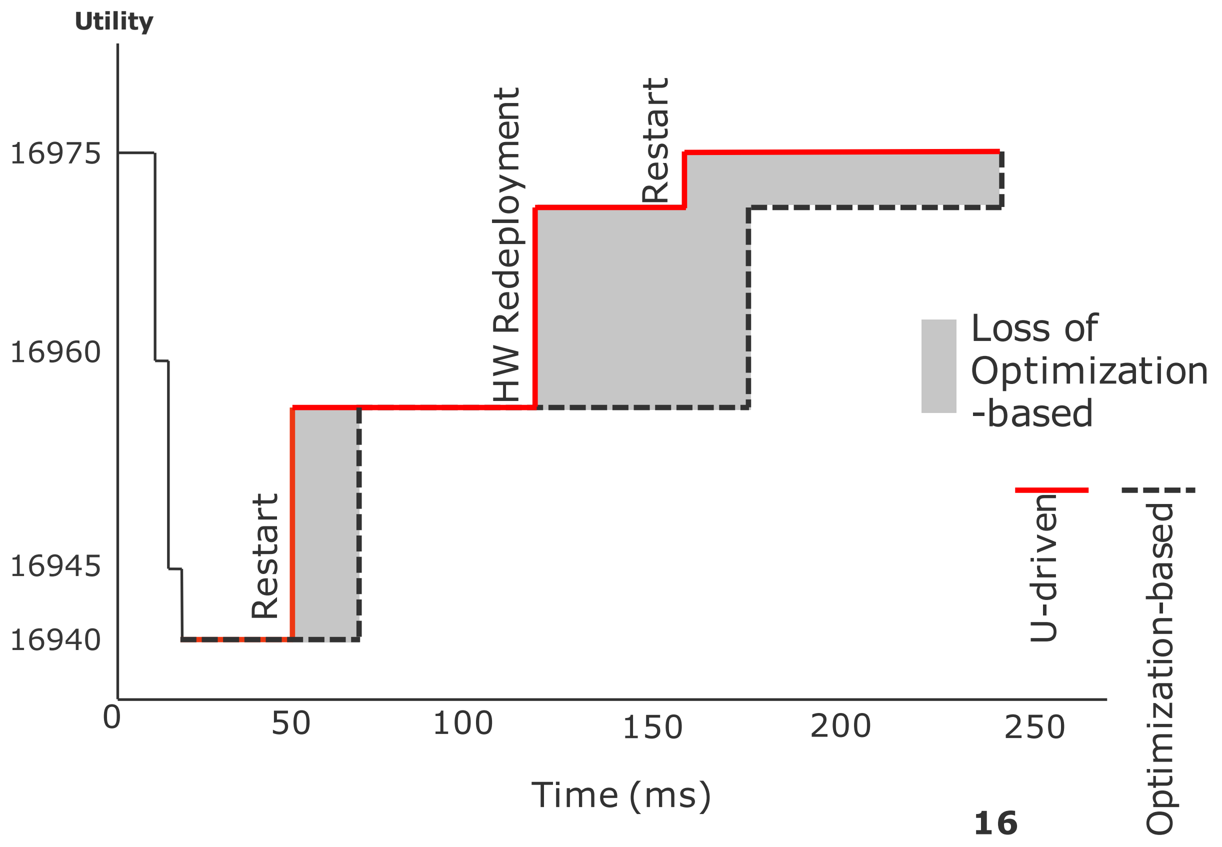
Number of Components	1 Failure			10 Failure			100 Failure			1000 Failure		
	Rule-based	U-driven	Opt.-based	Rule-based	U-driven	Opt.-based	Rule-based	U-driven	Opt.-based	Rule-based	U-driven	Opt.-based
18	0.76	0.89	5.02	10.37	14.36	56.68	NA	NA	NA	NA	NA	NA
180	0.68	0.89	5.01	9.71	13.58	59.07	14.22	17.70	219.54	NA	NA	NA
1800	0.61	0.74	4.83	10.60	13.47	58.24	13.82	26.65	211.09	54.50	60.09	3216.60
18000	0.65	0.71	4.90	10.14	13.87	71.93	21.80	26.38	271.51	127.80	171.31	3611.95

(ms)

	Optimization -based	**Rule -based**	**U-driven**
Optimal order of changes	✔	✘	**?**
Scalable	✘	✔	✔
Maximum Utility	✔	✘	**?**

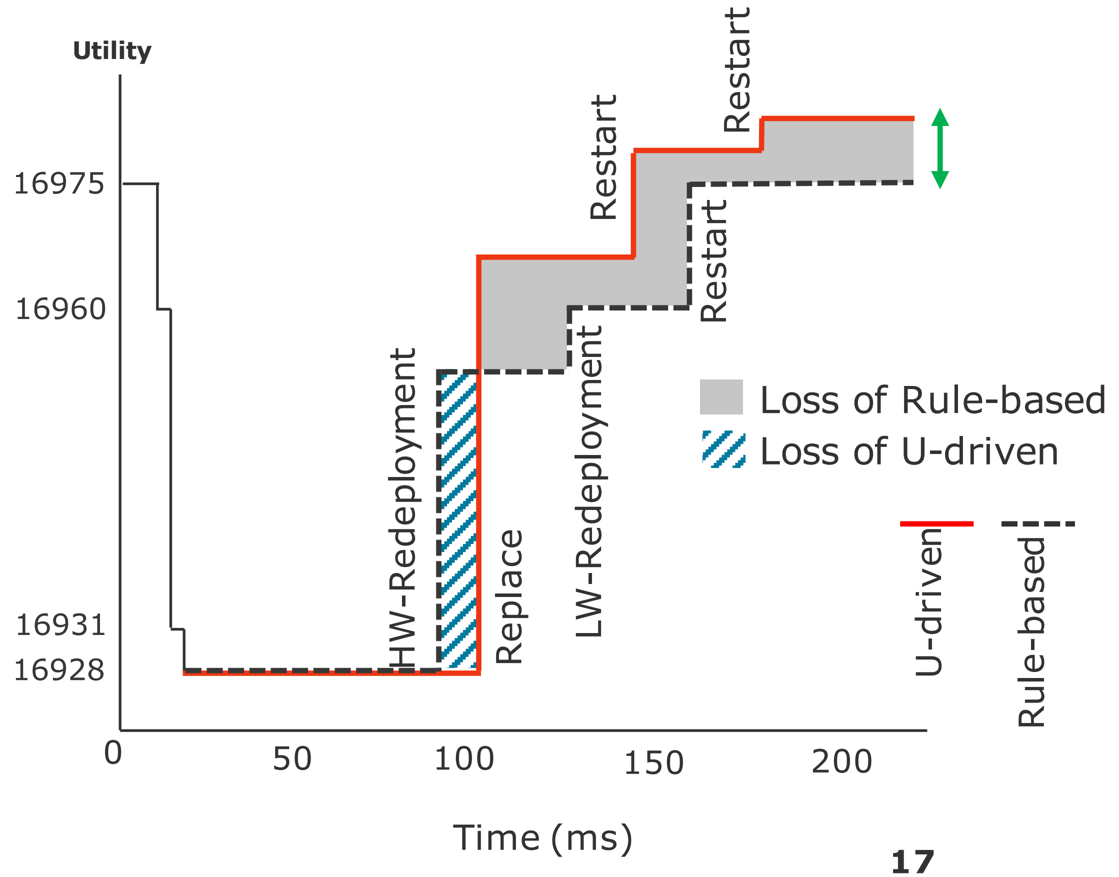
Lost Reward Due to Overhead

	Optimization -based	Rule -based	U-driven
Optimal order of repairs	✔	✘	✔
Scalable	✘	✔	✔
Maximum Utility	✔	✘	?



Lost Reward Due to Non-optimal Selection of Repair Steps [Wrong Ordering of Changes]

	Optimization -based	Rule -based	U-driven
Optimal order of repairs	✔	✘	✔
Scalable	✘	✔	✔
Maximum Utility	✔	✘	✔



Conclusion and Future Work

- Conclusion:
  - Defined utility functions for dynamic architectures and linking them to the adaptation rules
  - Proposed a novel approach to improve the self-healing reward while reducing the computation efforts for planning self-adaptation
  - Achieved optimal adaptation decisions online within a reasonable time

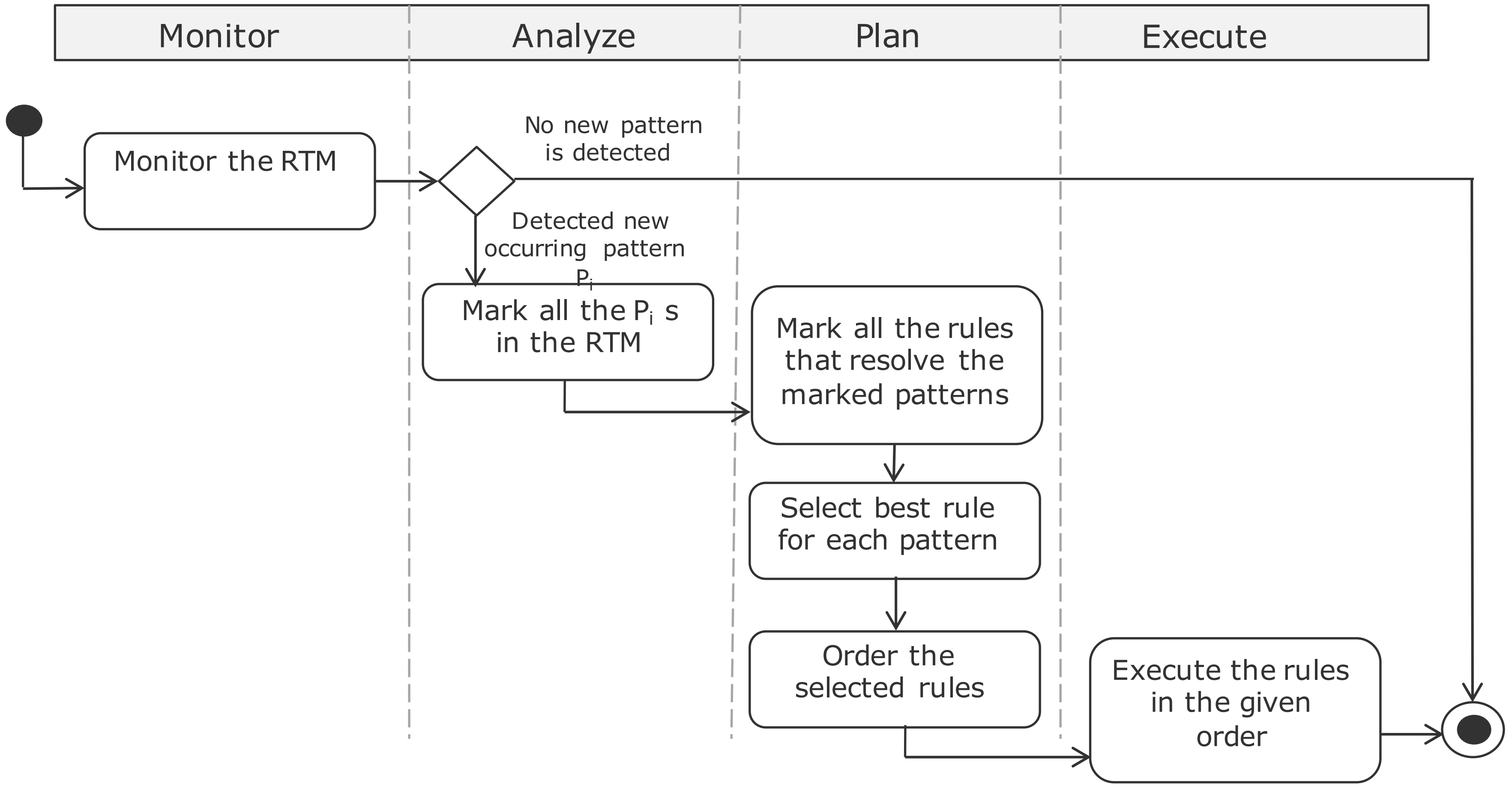
	Optimization -based	Rule -based	**U-driven**
Optimal order of repairs	✓	✗	✓
Scalable	✗	✓	✓
Maximum Utility	✓	✗	✓
Expressiveness	+	−	+/−

- Future work:
  - Weakening some of the assumptions made such as including conflicts among issues and rules
  - Support more complex class of utility functions such as non-linear utility functions

IT Systems Engineering | Universität Potsdam

END

Overview of the Approach



Monitor
Analyze
Plan
Execute
Monitor the RTM
No new pattern is detected
Detected new occurring pattern P_i
Mark all the P_i s in the RTM
Mark all the rules that resolve the marked patterns
Select best rule for each pattern
Order the selected rules
Execute the rules in the given order

Utility of Different Self-healing Approaches in Presence of Different Failure Profile Models

Number of Failures = 1

15<Number of Failures <40

1<Number of Failures <500

150<Number of Failures <600

Model Approach	Single	Uniform	Burst	BigBurst
Static	1.94E+09	1.93E+09	1.79E+09	1.82E+09
Solver	1.99E+09	1.94E+09	1.82E+09	1.81E+09
U-driven	1.99E+09	1.95E+09	1.83E+09	1.84E+09