

ASTON UNIVERSITY

---

**Requirements-aware Models to Support  
Better Informed Decision-making for  
Self-adaptation using Partially  
Observable Markov Decision Processes**

---

Luis Hernán García Paucar

Doctor of Philosophy

December 20, 2019

©Luis Hernán García Paucar, 2019

Luis Hernán García Paucar asserts her moral right to be identified as the author of  
this thesis

This copy of the thesis has been supplied on condition that anyone who consults it  
is understood to recognise that its copyright belongs to its author and that no  
quotation from the thesis and no information derived from it may be published  
without appropriate permission or acknowledgement.

ASTON UNIVERSITY

**Requirements-aware Models to Support Better Informed Decision-making for  
Self-adaptation using Partially Observable Markov Decision Processes**

Luis Hernán García Paucar

Doctor of Philosophy

December, 2019

A self-adaptive system (SAS) is a system that can adapt its behaviour in response to environmental fluctuations at runtime and its own changes. Therefore, the decision-making process of a SAS is challenged by the underlying uncertainty. In this dissertation, the author focuses on the kind of uncertainty associated with the satisficement levels of non-functional requirements (NFRs) given a set of design decisions reflected on a SAS configuration. Specifically, the focus of this work is on the specification and runtime handling of the uncertainty related to the levels of satisficement of the NFRs when new evidence is collected, and that may create the need of adaptation based on the reconfiguration of the system. Specifically, this dissertation presents two approaches that address decision-making in SASs in the face of uncertainty. First, we present RE-STORM, an approach to support decision-making under uncertainty, which uses the current satisficement level of the NFRs in a SAS and the required trade-offs, to therefore guide its self-adaptation. Second, we describe ARRoW, an approach for the automatic reassessment and update of initial preferences in a SAS based on the current satisficement levels of its NFRs. We evaluate our proposals using a case study, a Remote Data Mirroring (RDM) network. Other cases have been used as well in different publications. The results show that under uncertain environments, which may have not been foreseen in advance, it is feasible that: (a) a SAS reassess the preferences assigned to certain configurations and, (b) reconfigure itself at runtime in response to adverse conditions, in order to keep satisficing its requirements.

**Keywords:** uncertainty, runtime models, POMDPs, NFR preferences

*To Norka Mailin, who taught me the meaning of "discovering the warm water", and I would experience it many times during my research, but most importantly, for being the blessing of my life.*

## *Acknowledgements*

I would like to acknowledge the contributions of many people who have helped me to go through this journey, and to whom I am sincerely grateful. First and foremost, I want to express my gratitude and appreciation to my supervisor, Nelly Bencomo, who has played a key role by her continuous support, her invaluable guidance and her insightful advice for my research as well as my career.

I also thank Kevin Yuen for many enlightening and helpful conversations about multiple-criteria decision-making, which enriched the work presented in Chapter 5.

My experience as a graduate student would have been much poorer but for the fellow students at Aston. For their friendship, for many enlightening discussions and all the laughs, I would like to thank particularly Nasser Amaitik, Arezoo Vejdanparast, Aman Soni, Thomas Carr, Thomas Griffiths, Noa Garcia, Renato Barros, Juan Parra, Owen Reynolds, Huma Samin, and Deepeka Garg.

I am also grateful to my qualifying report examiners, Yulan He and Aniko Ekart, and my Ph.D. examiners, Tony Clark and Jean-Michel Bruel for the challenging discussions and helpful advice. Their comments and suggestions have improved the thesis. More generally, I thank the school of Engineering and Applied Science and the Software Engineering at Aston (SEA) research group for the stimulating environment to learn and carry out research.

This work and my career as a researcher have also benefited from conversations with Peter Sawyer, Antonio Garcia-Dominguez, Elizabeth Wanner, Diego Faria, Peter Lewis, Ulysses Bernardet, Geoffrey Morrison, and Betty Cheng, and my deepest gratitude goes to them also.

Finally, my special thanks to my family. In particular my parents, Blanca and Hernan, Crucita, and my brother Jorge, for their patience, encouragement, and continuous support. I have been so lucky for having them!!

*Luis*



# Contents

<b>Acknowledgements</b>	<b>4</b>
<b>1 Introduction</b>	<b>14</b>
1.1 Motivation . . . . .	14
1.2 Overall aim and Research questions . . . . .	15
1.3 Contributions . . . . .	16
1.4 Research strategy . . . . .	17
1.5 Overview of the work developed . . . . .	18
1.6 Dissertation outline . . . . .	19
<b>2 Research Baseline</b>	<b>21</b>
2.1 Uncertainty . . . . .	21
2.2 Models and uncertainty . . . . .	22
2.2.1 Runtime models . . . . .	22
2.2.2 Kinds of Runtime models . . . . .	22
System Models . . . . .	23
Context Models . . . . .	23
Requirement Models . . . . .	23
2.3 MAPE-K loop and runtime models . . . . .	24
2.4 Types of systems using runtime models . . . . .	26
2.4.1 Context-aware systems . . . . .	26
2.4.2 Requirements-Aware systems . . . . .	27
2.4.3 Self-adaptive Systems (SASs) . . . . .	27
2.5 SASs and runtime uncertainty management . . . . .	28
2.5.1 Probability theory and uncertainty management in SASs . . . . .	28
2.6 POMDPs and decision-making under uncertainty . . . . .	28

2.6.1	POMDPs . . . . .	29
2.7	RE-STORM: a Reinforcement Learning POMDP approach . . . . .	31
2.8	Summary . . . . .	32
<b>3</b>	<b>State of the art on Decision-Making under Uncertainty</b>	<b>34</b>
3.1	Techniques for decision-making under uncertainty ( $C_1$ ) . . . . .	35
3.2	Requirements representation for decision-making ( $C_2$ ) . . . . .	36
3.3	Specification of preferences at design-time ( $C_3$ ) . . . . .	38
3.4	Reassessment and update of preferences at runtime ( $C_4$ ) . . . . .	39
3.5	Summary . . . . .	40
<b>4</b>	<b>RE-STORM: Requirements Trade-offs for Self-adaptation using POMDPs</b>	<b>42</b>
4.1	Overview . . . . .	42
4.2	Case study: Remote data mirroring (RDM) system . . . . .	43
4.3	NFRs decision-making as a POMDP . . . . .	44
4.3.1	RDM SAS - POMDP representation . . . . .	45
4.3.2	From Requirements to POMDPs: a requirements-aware model	46
4.3.3	From requirements to POMDPs: NFRs inference evolution . . .	51
4.4	MAPE-K loop and POMDPs for decision-making in self-adaptation .	56
4.4.1	RE-STORM: MAPE-K loop activities . . . . .	57
4.4.2	RE-STORM: Details on the online planning activity using POMDPs	59
4.4.3	RE-STORM: Approximated optimal policy . . . . .	63
4.5	Summary . . . . .	64
<b>5</b>	<b>ARRoW: <u>A</u>utomatic <u>R</u>untime <u>R</u>eappraisal of <u>W</u>eights for Self-Adaptation</b>	<b>66</b>
5.1	Overview . . . . .	66
5.2	A motivating example . . . . .	67
5.3	P-CNP for reassessing NFR weights in Self-adaptation . . . . .	68
5.3.1	Step 1: Problem definition . . . . .	68
5.3.2	Step 2: Weights assignment to NFRs . . . . .	70
5.3.3	Step 3: Comparison of alternatives $alt \in Alt$ with respect to NFRs	72
5.3.4	Step 4: Information fusion and global weights . . . . .	73
5.4	<u>A</u> utomatic <u>R</u> untime <u>R</u> eappraisal of <u>W</u> eights: ARRoW . . . . .	74

5.4.1	ARRoW Initial Setup . . . . .	74
5.4.2	The ARRoW runtime process . . . . .	77
	Step 1: Monitoring NFR satisfaction to detect poor levels . . .	77
	Step 2: Balancing weights of NFRs and actions . . . . .	78
5.5	Summary . . . . .	81
<b>6</b>	<b>Evaluation</b>	<b>83</b>
6.1	Infrastructure used during the evaluation . . . . .	84
6.2	Initial setup of experiments with the RDM SAS . . . . .	85
6.2.1	Identification of sources of uncertainty and their treatment . . .	85
6.2.2	Setting of the initial preferences and Service Level Agreements (SLAs) for NFRs . . . . .	87
6.3	Implementation and evaluation of RE-STORM and ARRoW . . . . .	88
6.3.1	Configuration file of RE-STORM . . . . .	89
6.3.2	Configuration file of the dynamic contexts $DC_i$ to $DC_6$ . . . . .	90
6.3.3	DESPOT simulation model . . . . .	92
6.4	Experiments for decision-making under uncertainty related to dynamic changes in the environment . . . . .	93
6.4.1	Template for the specification of dynamic contexts . . . . .	94
6.4.2	RDM SAS under stable conditions . . . . .	95
6.4.3	RDM SAS dynamic contexts . . . . .	97
6.5	Results of experiments . . . . .	100
6.5.1	Summary and review of results of each of the 6 dynamic context	101
6.5.2	Aggregated view of results . . . . .	107
6.6	Further analysis: behaviour of the RDM SAS under different Service Level Agreements (SLAs) . . . . .	108
6.7	Time-aware queries for internal evaluations of the RDM SAS . . . . .	110
6.8	Comparison with related work . . . . .	110
6.9	Towards a simulation tool supported by RE-STORM and ARRoW . . .	114
6.10	Summary . . . . .	115
<b>7</b>	<b>Conclusions and Future Research Agenda</b>	<b>118</b>
7.1	Research Questions . . . . .	118

7.2	Exploited Research Collaborations . . . . .	121
7.3	Future Research Agenda . . . . .	122
7.3.1	Optimization of parameters for self-adaptation . . . . .	122
7.3.2	Optimization of the decision-making process . . . . .	123
<b>A</b>	<b>Publications</b>	<b>125</b>
A.1	Conferences . . . . .	125
A.2	Workshops . . . . .	126
<b>B</b>	<b>Example of time-aware query for the RDM SAS</b>	<b>128</b>
B.1	Time-aware query to show proactiveness in self-adaptation . . . . .	128
<b>C</b>	<b>Dynamic Contexts to represent unexpected changes in the environment</b>	<b>131</b>
C.1	Dynamic Context $DC_1$ : changes in the environment during the execution of the MST topology are introduced to reduce the reliability of the system: $P(MR_{t+1} = \text{True}   NFR_t, MST_t)$ . . . . .	131
C.2	Dynamic context $DC_2$ : changes in the environment during the execution of the RT topology are introduced to increment the cost and to reduce the performance of the system: $P(MC_{t+1} = \text{True}, MP_{t+1} = \text{True}   NFR_t, RT_t)$ . . . . .	138
C.3	Dynamic context $DC_3$ : changes in the environment during the execution of the topologies MST and RT are introduced to increment the cost and to reduce the reliability and the performance of the system: $P(MR_{t+1} = \text{True}   NFR_t, MST_t)$ and $P(MP_{t+1} = \text{True}, MC_{t+1} = \text{True}   NFR_t, RT_t)$ . . . . .	143
C.4	Dynamic context $DC_4$ : changes on the environment during the execution of the MST topology are introduced to increment the cost and to reduce the reliability and the performance of the system: $P(NFR_{t+1} = \text{True}   NFR_t, MST_t)$ . . . . .	149
C.5	Dynamic context $DC_5$ : changes on the environment during the execution of the RT topology are introduced to increment the cost and to reduce the reliability and the performance of the system: $P(NFR_{t+1} = \text{True}   NFR_t, RT_t)$ . . . . .	154

- C.6 Dynamic context  $DC_6$ : changes on the environment during the execution of the topologies MST and RT are introduced to increment the cost and to reduce the reliability and the performance of the system:

$$P(NFR_{t+1} = \text{True} | NFR_t, MST_t) \text{ and } P(NFR_{t+1} = \text{True} | NFR_t, RT_t) \quad . . . 159$$

## **D RDM SAS under different Service Level Agreements (SLAs) 164**

- D.1 SLAs: less strict scenario . . . . . 164
- D.2 SLAs: stricter scenario . . . . . 165

# List of Figures

1.1	Research challenges and contributions . . . . .	16
1.2	Overview of the contributions . . . . .	19
2.1	MAPE-K feedback loop . . . . .	24
2.2	Runtime models in an extended MAPE-K feedback loop . . . . .	25
2.3	General POMDP: runtime model for sequential decision-making . . .	29
3.1	Goal models for decision-making under uncertainty and requirements representation . . . . .	37
3.2	Specification of preferences at design-time . . . . .	38
3.3	Runtime reassessment and update of preferences . . . . .	39
4.1	Goal Model - RDM Case Study (i* notation) . . . . .	43
4.2	POMDP requirements-aware model - RDM Case Study . . . . .	45
4.3	RE-STORM architecture: runtime models and managed system . . . .	57
4.4	DESPOT Belief Tree with 2 sampled scenarios marked with green and red dots (The DESPOT tree is overlaid on a standard belief tree) . . . .	60
5.1	RDM example - POMDP representation . . . . .	67
5.2	RDM Example- Structural Assessment Network (SAN) . . . . .	69
5.3	POM - general structure to compare NFRs . . . . .	71
5.4	POM - example of NFRs comparison . . . . .	71
5.5	Relatives weights of $alt \in Alt$ in relation to MC . . . . .	72
5.6	P-CNP abstractions for weights propagation at runtime . . . . .	73
5.7	ARRoW process: design-time and runtime behaviour . . . . .	74
5.8	$POM_{NFRs}$ Example Case 1 . . . . .	80
5.9	$POM_{NFRs}$ Example Case 2 . . . . .	80

6.1	Implementation and evaluation of RE-STORM and ARRoW: inputs and output . . . . .	89
6.2	Dynamic context - specification template . . . . .	94
6.3	Minimization of Cost (MC) - satisficement level under stable conditions	96
6.4	Maximization of Reliability (MR) - satisficement level under stable conditions . . . . .	96
6.5	Maximization of Performance (MP) - satisficement level under stable conditions . . . . .	96
6.6	Chosen topology under stable conditions . . . . .	97
6.7	Dynamic contexts $DC_1$ to $DC_6$ - average satisficement levels of MR . .	106
6.8	Dynamic contexts $DC_1$ to $DC_6$ - consolidated view of the average satisficement levels NFRs . . . . .	107
6.9	Dynamic contexts $DC_1$ to $DC_6$ - consolidated view of preferred topologies	108
6.10	Simulation tool: architecture of implementation supported by RE-STORM and ARRoW . . . . .	114
C.1	$DC_1$ - Chosen topology before the update of reward values $R(s,a)$ . . .	132
C.2	$DC_1$ - Minimization of Cost: satisficement level before the update of reward values $R(s,a)$ . . . . .	133
C.3	$DC_1$ - Maximization of Reliability: satisficement level before the update of reward values $R(s,a)$ . . . . .	133
C.4	$DC_1$ - Maximization of Performance: satisficement level before the update of reward values $R(s,a)$ . . . . .	133
C.5	$DC_1$ - Minimization of Cost: satisficement level after the update of reward values $R(s,a)$ . . . . .	134
C.6	$DC_1$ - Maximization of Reliability: satisficement level after the update of reward values $R(s,a)$ . . . . .	135
C.7	$DC_1$ - Maximization of Performance: satisficement level after the update of reward values $R(s,a)$ . . . . .	135
C.8	$DC_1$ - Chosen topology after the update of reward values $R(s,a)$ . . . .	135
C.9	$DC_1$ - Average satisficement level for MC . . . . .	136
C.10	$DC_1$ - Average satisficement level for MR . . . . .	137

C.11 DC <sub>1</sub> - Average satisficement level for MP . . . . .	137
C.12 DC <sub>2</sub> - Chosen topology before the update of reward values $R(s,a)$ . . .	139
C.13 DC <sub>2</sub> - Minimization of Cost: satisficement level before the update of reward values $R(s,a)$ . . . . .	139
C.14 DC <sub>2</sub> - Maximization of Reliability: satisficement level before the up- date of reward values $R(s,a)$ . . . . .	140
C.15 DC <sub>2</sub> - Maximization of Performance: satisficement level before the update of reward values $R(s,a)$ . . . . .	140
C.16 DC <sub>2</sub> - Minimization of Cost: satisficement level after the update of reward values $R(s,a)$ . . . . .	140
C.17 DC <sub>2</sub> - Maximization of Reliability: satisficement level after the update of reward values $R(s,a)$ . . . . .	141
C.18 DC <sub>2</sub> - Maximization of Performance: satisficement level after the up- date of reward values $R(s,a)$ . . . . .	141
C.19 DC <sub>2</sub> - Chosen topology after the update of reward values $R(s,a)$ . . . .	141
C.20 DC <sub>2</sub> - Average satisficement level for MC . . . . .	142
C.21 DC <sub>2</sub> - Average satisficement level for MR . . . . .	142
C.22 DC <sub>2</sub> - Average satisficement level for MP . . . . .	143
C.23 DC <sub>3</sub> - Chosen topology before the update of reward values $R(s,a)$ . . .	144
C.24 DC <sub>3</sub> - Minimization of Cost: satisficement level before the update of reward values $R(s,a)$ . . . . .	144
C.25 DC <sub>3</sub> - Maximization of Reliability: satisficement level before the up- date of reward values $R(s,a)$ . . . . .	145
C.26 DC <sub>3</sub> - Maximization of Performance: satisficement level before the update of reward values $R(s,a)$ . . . . .	145
C.27 DC <sub>3</sub> - Minimization of Cost: satisficement level after the update of reward values $R(s,a)$ . . . . .	145
C.28 DC <sub>3</sub> - Maximization of Reliability: satisficement level after the update of reward values $R(s,a)$ . . . . .	146
C.29 DC <sub>3</sub> - Maximization of Performance: satisficement level after the up- date of reward values $R(s,a)$ . . . . .	146
C.30 DC <sub>3</sub> - Chosen topology after the update of reward values $R(s,a)$ . . . .	147



C.31 DC <sub>3</sub> - Average satisficement level for MC . . . . .	147
C.32 DC <sub>3</sub> - Average satisficement level for MR . . . . .	148
C.33 DC <sub>3</sub> - Average satisficement level for MP . . . . .	148
C.34 DC <sub>4</sub> - Chosen topology before the update of reward values $R(s,a)$ . . .	149
C.35 DC <sub>4</sub> - Minimization of Cost: satisficement level before the update of reward values $R(s,a)$ . . . . .	150
C.36 DC <sub>4</sub> - Maximization of Reliability: satisficement level before the up- date of reward values $R(s,a)$ . . . . .	150
C.37 DC <sub>4</sub> - Maximization of Performance: satisficement level before the update of reward values $R(s,a)$ . . . . .	150
C.38 DC <sub>4</sub> - Minimization of Cost: satisficement level after the update of reward values $R(s,a)$ . . . . .	151
C.39 DC <sub>4</sub> - Maximization of Reliability: satisficement level after the update of reward values $R(s,a)$ . . . . .	151
C.40 DC <sub>4</sub> - Maximization of Performance: satisficement level after the up- date of reward values $R(s,a)$ . . . . .	151
C.41 DC <sub>4</sub> - Chosen topology after the update of reward values $R(s,a)$ . . . .	152
C.42 DC <sub>4</sub> - Average satisficement level for MC . . . . .	152
C.43 DC <sub>4</sub> - Average satisficement level for MR . . . . .	153
C.44 DC <sub>4</sub> - Average satisficement level for MP . . . . .	153
C.45 DC <sub>5</sub> - Chosen topology before the update of reward values $R(s,a)$ . . .	154
C.46 DC <sub>5</sub> - Minimization of Cost: satisficement level before the update of reward values $R(s,a)$ . . . . .	155
C.47 DC <sub>5</sub> - Maximization of Reliability: satisficement level before the up- date of reward values $R(s,a)$ . . . . .	155
C.48 DC <sub>5</sub> - Maximization of Performance: satisficement level before the update of reward values $R(s,a)$ . . . . .	155
C.49 DC <sub>5</sub> - Minimization of Cost: satisficement level after the update of reward values $R(s,a)$ . . . . .	156
C.50 DC <sub>5</sub> - Maximization of Reliability: satisficement level after the update of reward values $R(s,a)$ . . . . .	156

C.51 DC <sub>5</sub> - Maximization of Performance: satisficement level after the up- date of reward values $R(s,a)$ . . . . .	156
C.52 DC <sub>5</sub> - Chosen topology after the update of reward values $R(s,a)$ . . . .	157
C.53 DC <sub>5</sub> - Average satisficement level for MC . . . . .	157
C.54 DC <sub>5</sub> - Average satisficement level for MR . . . . .	158
C.55 DC <sub>5</sub> - Average satisficement level for MP . . . . .	158
C.56 DC <sub>6</sub> - Chosen topology before the update of reward values $R(s,a)$ . . .	159
C.57 DC <sub>6</sub> - Minimization of Cost: satisficement level before the update of reward values $R(s,a)$ . . . . .	160
C.58 DC <sub>6</sub> - Maximization of Reliability: satisficement level before the up- date of reward values $R(s,a)$ . . . . .	160
C.59 DC <sub>6</sub> - Maximization of Performance: satisficement level before the update of reward values $R(s,a)$ . . . . .	160
C.60 DC <sub>6</sub> - Minimization of Cost: satisficement level after the update of reward values $R(s,a)$ . . . . .	161
C.61 DC <sub>6</sub> - Maximization of Reliability: satisficement level after the update of reward values $R(s,a)$ . . . . .	161
C.62 DC <sub>6</sub> - Maximization of Performance: satisficement level after the up- date of reward values $R(s,a)$ . . . . .	161
C.63 DC <sub>6</sub> - Chosen topology after the update of reward values $R(s,a)$ . . . .	162
C.64 DC <sub>6</sub> - Average satisficement level for MC . . . . .	162
C.65 DC <sub>6</sub> - Average satisficement level for MR . . . . .	163
C.66 DC <sub>6</sub> - Average satisficement level for MP . . . . .	163
D.1 Less strict scenario - Minimization of Cost: $P(MC = \text{True}) \geq 0.7$ . . . .	164
D.2 Less strict scenario - Maximization of Reliability: $P(MR = \text{True}) \geq 0.8$	165
D.3 Less strict scenario - Maximization of Performance: SLA $P(MP = \text{True}) \geq$ <b>0.6)</b> . . . . .	165
D.4 SLAs example 01 - Minimization of Cost: $P(MC = \text{True}) \geq 0.8$ . . . .	166
D.5 SLAs example 01 - Maximization of Reliability: $P(MR = \text{True}) \geq 0.95$	166
D.6 SLAs example 01 - Maximization of Performance: $P(MP = \text{True}) \geq 0.85$	166

D.7	SLAs example 01 - Minimization of Cost: satisficement level after the update of reward values $R(s,a)$ . . . . .	167
D.8	SLAs example 01 - Maximization of Reliability: satisficement level af- ter the update of reward values $R(s,a)$ . . . . .	167
D.9	SLAs example 01 - Maximization of Performance: satisficement level after the update of reward values $R(s,a)$ . . . . .	167
D.10	SLAs example 02 - Minimization of Cost: $P(MC = \text{True} \geq 0.8)$ . . . . .	168
D.11	SLAs example 02 - Maximization of Reliability: $P(MR = \text{True} \geq 0.99)$	168
D.12	SLAs example 02 - Maximization of Performance: $P(MP = \text{True} \geq 0.85)$	169
D.13	SLAs example 02 - Minimization of Cost: satisficement level after the update of reward values $R(s,a)$ . . . . .	169
D.14	SLAs example 02 - Maximization of Reliability: satisficement level af- ter the update of reward values $R(s,a)$ . . . . .	170
D.15	SLAs example 02 - Maximization of Performance: satisficement level after the update of reward values $R(s,a)$ . . . . .	170

# List of Tables

4.1	RDM SAS - NFRs and states $s \in S$ . . . . .	47
4.2	RDM SAS - Reward values $R(s,a)$ . . . . .	49
4.3	RDM SAS - MON variables and observations . . . . .	51
4.4	CPT MC: $P(MC'   MC, MR, MP, a)$ . . . . .	53
4.5	CPT MR: $P(MR'   MC, MR, MP, a)$ . . . . .	53
4.6	CPT MP: $P(MP'   MC, MR, MP, a)$ . . . . .	54
4.7	CPT RBC: $P(RBC   MC', a)$ . . . . .	55
4.8	CPT ANL: $P(ANL   MR', a)$ . . . . .	55
4.9	CPT TTW: $P(TTW   MP', a)$ . . . . .	56
5.1	RDM example - Reward values $R(s,a)$ . . . . .	68
5.2	P-CNP scale of comparison . . . . .	70
5.3	RDM example - SLAs . . . . .	75
5.4	RDM example - Ranges for suitable satisficement . . . . .	76
5.5	RDM example - Ranges for poor satisficement . . . . .	76
5.6	RDM example - Updated reward values $R(s,a)$ . . . . .	81
6.1	RDM SAS - SLAs . . . . .	87
6.2	RDM SAS - Reward values $R(s,a)$ . . . . .	88
6.3	SLAs fulfillment based on the average satisficement levels of NFRs . . . . .	101
6.4	Dynamic contexts $DC_1$ and $DC_3$ - average satisficement levels of NFRs . . . . .	102
6.5	Dynamic contexts $DC_2$ and $DC_4$ - average satisficement levels of NFRs . . . . .	103
6.6	Dynamic context $DC_5$ - average satisficement levels of NFRs . . . . .	104
6.7	Dynamic context $DC_6$ - average satisficement levels of NFRs . . . . .	105
6.8	Comparison of RE-STORM and ARRoW to other approaches . . . . .	111

## Chapter 1

# Introduction

### 1.1 Motivation

Self-adaptive Systems (SASs) are expected to self-adapt to even unanticipated events based on incomplete information about themselves and their environment. Therefore, decision-making under uncertainty is paramount for self-adaptive systems [3]. Significant advances have been made in applying models to drive decision-making under uncertainty [6, 39, 18, 73, 94, 93]. Recent progress in machine learning, including Bayesian learning and inference [53], has been key to enable access to information at runtime, to dynamically keep the models of decision-making up-to-date during runtime [7, 44]. Moreover, SASs are subject to the satisficement of non-functional requirements (NFRs) which are usually competing among them, and collectively characterize how system's goals are to be satisfied by their trade-offs [56, 81]. Several authors have approached different issues related to uncertainty and NFRs trade-offs [24, 81, 23, 92, 73, 80, 17, 98, 79, 54]. However, critical challenges need to be further explored. One of the issues that needs further research is that current approaches that deal with uncertainty and the specification of NFRs and their preferences mainly do it at design-time [24, 81, 58, 57, 22]. This is an issue as initial specifications of NFRs and their preferences, settled for foreseen contexts at design-time, may not be suitable anymore when unexpected contexts arise during the system's execution.

On the other hand, recent progress on Partially Observable Markov Decision Processes (POMDPs) implementations [96] have shown promising results in the AI research community for decision-making under uncertain environments [41, 75, 55, 96,

82, 4]. In those research communities, different scalability issues [96] have been overcome for the decision-making planning and when models and assumptions change at runtime [55, 96]. However, these models scarce of support for specific definition representation for trade-off of NFRs as the environment changes.

Given the above, two main challenges have been identified in this research work (See Fig. 1.1):

(i) The need for new decision-making techniques driven by the current satisfaction levels of NFRs subject to uncertain environments based on new collected evidence about the need for changing and,

(ii) The need of new techniques to reassess and update initial preferences according to new and unforeseen contexts arisen at runtime.

## 1.2 Overall aim and Research questions

The overall aim of this dissertation is to improve the specification and runtime handling of the uncertainty related to the levels of satisfaction of the NFRs when new evidence is collected [37, 7, 8] to:

- Explicitly deal with the uncertainty associated with the current runtime context represented as probability distributions over the system's NFRs satisfaction
- Balance different conflicting NFRs
- Maintain the definition of uncertainty over time as new evidence arrives in a consistent way with the past using Bayesian learning
- Incorporate preferences (i.e. rewards and penalties) that properly address the current runtime context modelled

In this dissertation, we move towards this direction by addressing the following research questions:

**RQ1:** How can we represent the current state of NFRs and their evolution in model-based self-adaptive systems that are subject to uncertain environments?

**RQ2:** How can we improve the trade-off among NFRs in model-based self-adaptive systems that are subject to uncertain environments, by updating preferences and based on new evidence collected during execution?

**RQ3:** How can techniques for eliciting initial preferences about requirements, used at design-time, be applicable to runtime models for self-adaptive systems?

The research questions **RQ1** and **RQ2** are mainly related to the first challenge identified in the previous section. The research question **RQ3** is more related to the second challenge (See Fig. 1.1).

### 1.3 Contributions

This section describes the contributions developed to tackle the previously stated challenges in this work. A summary is shown in Fig. 1.1 and a detailed list of publications can be consulted in Appendix A.

Challenges	Proposed solution approaches	Dissemination / Publications	
(i) Runtime decision-making under uncertainty based on runtime evidence	RE-STORM	SEAMS '18 SASO '19	MODELS '19
(ii) Dynamic update of preferences	ARRoW	AIRE workshop '16 RE '16 (Poster) MRT workshop '16 CIBSE '17 RE NEXT '17 RE '17 (Poster) SAC '19	Plus an ongoing journal paper (in draft at the moment)

FIGURE 1.1: Research challenges and contributions

In order to address the first challenge, we have developed RE-STORM: Requirements Trade-off for self-adaptation based on Partially Observable Markov Decision Processes [9, 36, 37]. RE-STORM, a mathematical probabilistic framework

based on POMDPs, serves as a requirements-aware runtime model that can be updated with new learned evidence during execution to support reasoning and decision-making about partial satisficement of NFRs and their trade-off according to changes in the environment. RE-STORM is explained in detail in **Chapter 4**.

To address the second challenge, we have developed ARRoW:

Automatic Runtime Reappraisal of Weights [38, 67, 9], an approach to support the dynamic update of preferences/weights associated with the NFRs and adaptation actions in a SAS to therefore improve the levels of satisficement of NFRs when the current preferences do not agree with newly found contexts. To develop ARRoW, we have extended the Primitive Cognitive Network Process (P-CNP), a version of the Analytical Hierarchy Process (AHP), to therefore enable the handling and update of preferences during runtime. ARRoW works on top of RE-STORM and is supported by a set of runtime models, which contains the updated preferences that serve as input for the decision-making process in a SAS during execution. ARRoW is presented in detail in **Chapter 5**.

We have also evaluated the contributions presented above, applying them to at least a substantial case study. The contributions have been shared with the research community in the form of several conference publications [9, 38, 67, 36, 35, 37] which are presented in detail in **Chapter 7**. As future work, we have identified different challenges that even after our approaches have been applied, still remain unaddressed (See Chapter 7).

## 1.4 Research strategy

To reach the aim the research described here, the following approach has been applied:

1. To undertake a survey of the state of the art on approaches for decision-making under uncertainty in SASs, to therefore identify the research gaps.
2. To examine and evaluate the state of the art related to the specification and reassessment of initial assumptions in SASs. Specifically, the specification, reassessment and update of preferences over NFRs and adaptation actions in a



system. Publication [35] shows the initial results of this step.

3. To carry out an investigation on runtime models, MAPE-K feedback loops for self-adaptation and systems that use runtime models as a mean to cope with runtime uncertainty by exploiting new information available during the system's execution.
4. To investigate the complementary nature of the topics described in (1), (2) and (3) when developing model-based self-adaptive systems to cope with runtime uncertainty management, to study how they can be complemented between them.
5. To propose a structured technique to support the runtime and up-to-date representation of the levels of satisficement of the NFRs and the decision-making driven by them. This technique is complemented with the reassessment and update capabilities of initial assumptions in a SAS.

The research done aims to demonstrate the viability and the benefits of the approach using the proposed techniques to support decision-making under uncertainty driven by the current satisficement level of the NFRs in a system.

## 1.5 Overview of the work developed

Fig. 1.2 shows an overview of the work performed and presented in this dissertation. Row 1 identifies the **Challenges** that were addressed; row 2 shows the **Research Questions** related to the challenges, row 3 list the **Scientific Publications** generated.

Challenges	(i) Runtime decision-making under uncertainty based on runtime evidence	(ii) Dynamic update of preferences
Research Questions	RQ1 : NFRs representation RQ2 : NFRs trade-offs	RQ3: reassessment and update of preferences
Scientific Publications	<div>SEAMS '18</div> <div>SASO '19</div>	<div>CIBSE '17</div> <div>RE-NEXT '17</div> <div>SAC '19</div> <div>RE '16 Poster</div> <div>RE '17 Poster</div> <hr/> <div>MODELS '19</div> <div>Journal paper in preparation</div>

FIGURE 1.2: Overview of the contributions

For the first challenge, **runtime decision-making under uncertainty**, two research questions have been identified, i.e. RQ1 and RQ2, and 2 publications associated have been accepted: SEAMS'18 [37] and SASO'19 [38].

For the second challenge, **runtime update of preferences**, one research question has been identified, RQ3, and three associated publications have been accepted: CIBSE'17 [35], RE '17 [68], RE-NEXT'17 [67], SAC'19 [38]. In addition, the paper MODELS'19 [9]), consolidates the results in both challenges.

The work presented in this dissertation has also made contributions to (i) the Software Engineering at Aston (SEA) research group and (ii) the research group of the Systems Engineering department at Pontificia Universidad Javeriana (Bogota, Colombia) [12]. As a result of collaboration activities within the SEA research group, tools provided by the project: History-aware explanation capabilities in SAS [32, 33, 31] have been used in this work to support the implementation of time-aware queries over the history of a requirements-aware runtime model. Specific resultant behaviours due to the decision-making process of RE-STORM are reported in Chapter 6.

## 1.6 Dissertation outline

The remainder of this dissertation develops as follows:

Chapter 2, **Research Baseline**, presents the technical background and the fundamental approaches that are applied in the following parts of the dissertation, such as decision-making under uncertainty, models at runtime and sequential decision-making by using Partially Observable Markov Decision Processes (POMDPs).

Chapter 3, **State of the Art on decision-making under uncertainty**, discusses the state of the art in relation to the scope of this dissertation: decision-making under uncertain environments and runtime reassessment and update of preferences over NFRs and decision-making strategies. This chapter motivates the approaches presented.

Chapter 4, **RE-STORM**, presents a proposal for NFRs trade-offs and decision-making by using POMDPs as runtime models. In this chapter, “partial observability”, a intrinsic characteristic of a POMDP, is exploited to model at runtime the uncertainty about the satisficement levels of NFRs and to drive the decision-making in a SAS based on their current satisficement values.

Chapter 5, **ARRoW**, presents an approach for automatic runtime reappraisal of preferences by updating reward values  $R(s,a)$  in a POMDP runtime model. New preferences are obtained by using the Primitive Cognitive Network Process (P-CNP). Specifically, pairwise comparisons of the current satisficement levels of the NFRs and runtime models of P-CNP are used to derive the final preferences.

Chapter 6, **Evaluation**, presents the details of the evaluations performed to validate the proposed approaches, the measurements and the results obtained.

Finally, in Chapter 7, **Conclusions and Future Work** are presented. The concluding remarks are discussed along with a brief outline of the dissertation that highlights the contributions of this research and answers to the research questions. The future research agenda is also presented.

## Chapter 2

# Research Baseline

In this chapter the research baseline of the dissertation is presented. Different techniques and approaches, which serve as the starting point for this work, are explained. Firstly, uncertainty and models are presented. Secondly, the MAPE-K loop and its relationship with runtime models is explained. Thirdly, the types of systems from which we leverage our work are presented. Finally, Partially Observable Markov Decision Processes (POMDPs), decision-making under uncertainty and their relationship with the Reinforcement Learning (RL) approach are depicted.

## 2.1 Uncertainty

In self-adaptation, uncertainty can be defined as a system state of incomplete or inconsistent knowledge such that it is not possible for it to know which adaptation decision hold at a specific point [1]. Uncertainty may arise for example, due to missing or ambiguous requirements, erroneous assumptions, unpredictable behaviour in the execution environment, or incomplete information obtained by potentially imprecise or unreliable sensors in the monitoring infrastructure.

The different sources of uncertainty mentioned above, may occur and affect a software system, either at the requirements, design, or execution phases of the software life cycle. If a source of uncertainty at the requirements or design-time levels is not managed before the execution phase begins, then that source of uncertainty will be propagated throughout the execution of the system. Our proposal contributes an approach to dealing with uncertainty at runtime by using requirements-aware runtime models.

## 2.2 Models and uncertainty

A software system can successfully operate in multiple dynamic contexts by using runtime models which augment the information available at design-time with information monitored at runtime [39]. We use runtime models and its capacity for augmenting information available at design-time, as a means to cope with uncertainty. In this work, we take a general definition of model formalised in [39]:

*A **model** is a characterization with three main elements: **an original** the model refers to, **a purpose** that defines what the model should be used for, and **an abstraction function** that maps only purposeful and relevant characteristics of the original to the model.*

A model may refer to more than one original and any model is used as a representation of their originals to ease development or runtime activities.

### 2.2.1 Runtime models

A runtime model can be defined as a self-representation of its associated original (e.g. the running system) that addresses an aspect of it. In a system, aspects could be its structure, behaviour, or goals which can be manipulated at runtime for specific purposes [11]. The first and most common original of a runtime model is the system itself. The runtime model is causally-connected to the running system, meaning that a change in the runtime model triggers a corresponding change in the running system and vice versa. A runtime model captures relevant information of the running system for different purposes. In this work, one main purpose is the runtime and up-to-date representation of the current satisficement levels of the NFRs of the system to support the required decision-making, which is driven by the trade-off of the levels of satisficement of the NFRs.

### 2.2.2 Kinds of Runtime models

Based on their possible originals, different kinds of runtime models can be identified. Next, the runtime models related to our work are presented.

## System Models

A system is the most common original in a runtime model. In these models the runtime model provides an abstract view on the running system. The runtime model can be used to describe possible future configurations of the running system. Afterwards, to realise the causal connection, the update of the running system is triggered.

In our work, the system behaviour is controlled by the runtime model, based on its representation of the current satisficement levels of the NFRs. The current satisficement level of the NFRs is not directly observable. It is inferred by using probability distributions and Bayesian inference [53] based on monitored values from the system's context during its execution. Details on the computation of the current satisficement levels of the NFRs in a system are presented in Chapter 4.

## Context Models

The context of the system is the part of the environment that can be observed by the system [40]. This context can be an original of a runtime model. The runtime model represents characteristics of the context via the use of sensors. In this model, the causal connection implies that the runtime model is accordingly updated when the context changes as indicated by changing measurements of the sensors.

It is possible that relevant characteristics of the context cannot be observed directly. Therefore, a dedicated analysis is required to derive them indirectly from other observations. In this dissertation, observations collected from the system's context are used to infer the current state of the system, i.e. the current satisficement level of its NFRs.

## Requirement Models

The requirements of the system may be also subject of a runtime model [81]. In this case, either some form of (i) online representation of the requirements exists that is linked to the runtime model by a causal connection or (ii) changes of the requirements have to be manually reflected on to the runtime model. In both cases the runtime model carries information about the relevant requirements within the system

and therefore the system can, for example, check whether the current requirements are fulfilled or try to adjust its behaviour such that the fulfillment of the requirements increases. Our approach follows this behaviour. In our proposal, an online representation of the NFRs exist and the decision-making in a system is driven by the current satisficement levels of its NFRs.

In this section, three different runtime models related to our proposal have been presented. In practice, a single runtime model may refer multiple originals as in our case. Our implementation of a requirements-aware runtime model refers:

- the system itself to modify its behaviour and
- the context of the system to infer at runtime, the current satisficement levels of its NFRs.

### 2.3 MAPE-K loop and runtime models

Different kinds of software systems [17, 56] can be implemented via an autonomic manager that steers the adaptation with a feedback control loop known as the MAPE-K feedback loop [90, 51]. The MAPE-K loop shown in Fig. 2.1, emphasizes the role of feedback for autonomic computing [51].

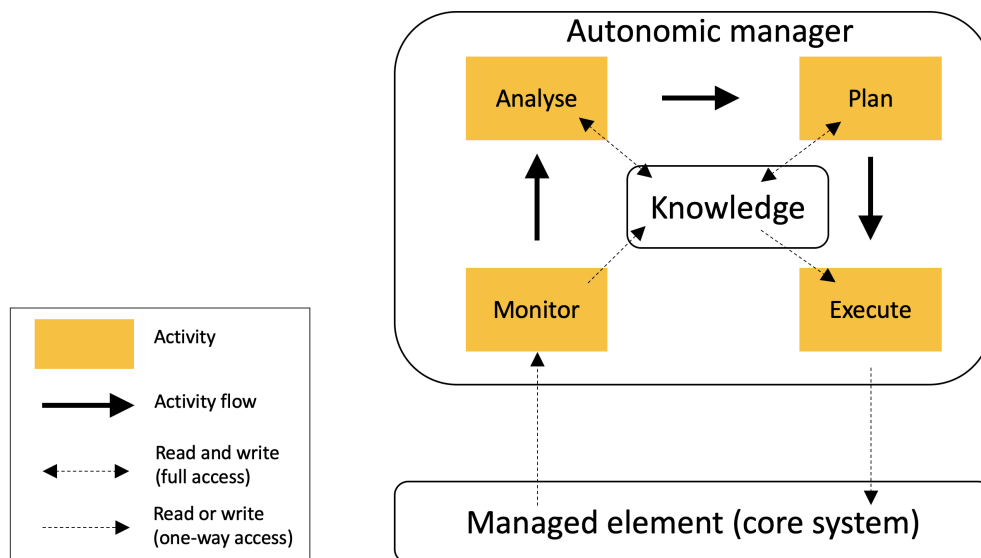


FIGURE 2.1: MAPE-K feedback loop

First, the system is split into a managed element (core system) and an autonomic manager (adaptation engine). Four key activities are defined in a MAPE-K feedback loop. They operate on the basis of a common knowledge base. **(M)** stands for the **monitoring** process of the managed elements. Monitoring is responsible for gathering raw data, such as measurements about the state of the managed system. It typically defines the frequency at which the data must be acquired. **(A)** represents the **analysis** of the monitored data, which could require filtering actions either because of noise on the monitored data or because it cannot be used directly from its raw monitored values. Both, monitored and analyzed data, are used to update the knowledge base of the MAPE-K loop. The knowledge base allows the system to use data and historic information to underpin the decision-making process to satisfy its goals [77, 54]. **(P)** stands for **planning** actions using the knowledge of the system held in the knowledge base. **(E)** represents the **execution** of the planned action. It consists of changing the value of the actuator(s) in a system at a frequency which is most often equivalent to the sampling frequency of the monitoring phase [77].

An extended MAPE-K loop that includes runtime models is depicted in Figure 2.2. This extended version uses an adaptation engine that takes into account, in addition to the core system, its context and requirements as part of the knowledge base.

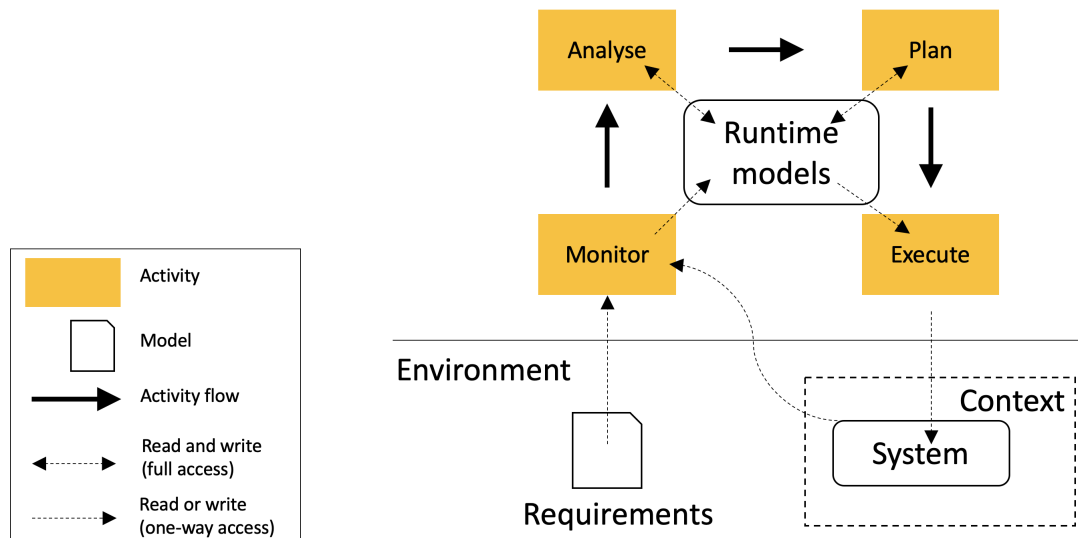


FIGURE 2.2: Runtime models in an extended MAPE-K feedback loop



In the extended MAPE-K loop, the monitoring activity is gathering measurements about the state of the system and its context, but additionally, this activity may recognize updates to the requirements. The analysis activity interprets the collected data and detects changes that might lead to adaptations. Afterwards the runtime models are updated accordingly. The planning activity employs the runtime models to reason about how the running system should adapt in response to changes. Finally, the execution activity uses the runtime models as basis to realize planned adaptations.

Our proposal uses a requirements-aware runtime model to interact with the core system and its context to deal with the runtime uncertainty regarding the current satisficement levels of the system's NFRs.

## 2.4 Types of systems using runtime models

Different types of systems take advantage of runtime models to control some aspects of their execution by using the MAPE-K loop. This section shows the types of systems related to our work.

### 2.4.1 Context-aware systems

Context-awareness [91] implies that a system is able to monitor its context. Context-aware systems select and apply adaptations depending on their context. Context-aware systems can leverage runtime models to represent the context and cover all processes of the MAPE-K loop and support adaptation.

In our approach, the context provides necessary sensory information required to infer and represent the current satisficement levels of the NFRs in a system. The analysis activity of the MAPE-K loop matches the perceived sensory information to the NFRs and updates the runtime models accordingly. The planning activity identifies the actions that the system should perform based on the updated satisficement levels of its NFRs (which are updated based on the recognized current system's context) and the execution activity applies these actions at runtime.

### 2.4.2 Requirements-Aware systems

Requirement-awareness enables a system to identify changes to its own requirements. Requirements-aware systems use runtime models to represent their requirements [25, 26], track their changes [10, 88] and trigger adaptations in the system behaviour in order to increase requirements satisfaction [5]. In these systems, requirements can be revised and reappraised over short periods of time. Modifications of requirements can be triggered due to different reasons, for example, by their varying satisfaction, changing market needs or changing final users preferences [40]. Our proposal leverages this definition.

In this dissertation, we use requirements-aware runtime models and the activities of the MAPE-K loop to support requirements-awareness and adaptation. The analysis activity uses the data collected from the system's context to update the runtime model and recompute the NFRs satisficement levels. The planning activity computes the adaptations to be performed by taking into account the current satisficement levels of the NFRs and assumptions as captured by the runtime models. One important assumption in our approach are the initial stakeholders' preferences about the NFRs and adaptation actions in a system, which may need to be updated at runtime, under some specific situations detected during the system's execution. Afterwards, the execution activity applies selected adaptation actions on the system.

### 2.4.3 Self-adaptive Systems (SASs)

Self-awareness [60] is the capability of a system to monitor itself. Additionally, a self-adaptive systems can also react to observed changes by applying proper adaptations to themselves. At present, the term self-adaptive systems is used in a very broad sense and it can include self-awareness, context-awareness as well as requirements-awareness [40]. Our approach leverages this broad definition to use a requirements-aware runtime model to drive the decision-making process based on the current satisficement level of the NFRs in a SAS.

## 2.5 SASs and runtime uncertainty management

Any source of uncertainty that is not managed before the SAS is deployed must therefore be addressed by the SAS at runtime. Runtime uncertainty occurs primarily from interactions between a SAS and its unpredictable environment. The execution environment can introduce events or conditions that a SAS might be unable to interpret or handle because they were unforeseeable at design-time. In this work, our concern is related to the management of runtime uncertainty and their impact on the current satisfaction levels of the NFRs in a SAS.

### 2.5.1 Probability theory and uncertainty management in SASs

Probability theory is the branch of mathematics concerned with the study of random phenomena. Probability is the measure of the likeliness that an event will occur, and is quantified as a number in the real interval  $[0, 1]$  (where 0 indicates impossibility and 1 certainty). Within probability theory, information relative to the frequencies of past outcomes can be used to derive probabilities that represent the likelihood of possible outcomes for future events. This interpretation of probability, which is related to statistical and Bayesian inference [53], may be employed by SASs:

- to estimate future contexts and system's behavior for optimizing the system's operation.
- to deal with runtime uncertainty through reasoning under partial observability.

Our approach uses Bayesian theory to infer at runtime the satisfaction levels of the NFRs in a SAS based on observations obtained from the current system's context. We realize this proposal by using Partially Observable Markov Decision Processes (POMDPs).

## 2.6 POMDPs and decision-making under uncertainty

Our proposal represents a novel use of POMDPs in the research area of self-adaptation and requirements management. POMDP foundations along with concepts of partial observation and policies are explained below.

### 2.6.1 POMDPs

They provide a principled approach to model sequential decision-making problems and make rational decisions in the face of uncertainty within changing environments [55, 71]. Fig. 2.3 shows a general POMDP with its main elements. A POMDP can be specified as a 6-tuple  $(S, A, Z, T, O, R)$ , where:

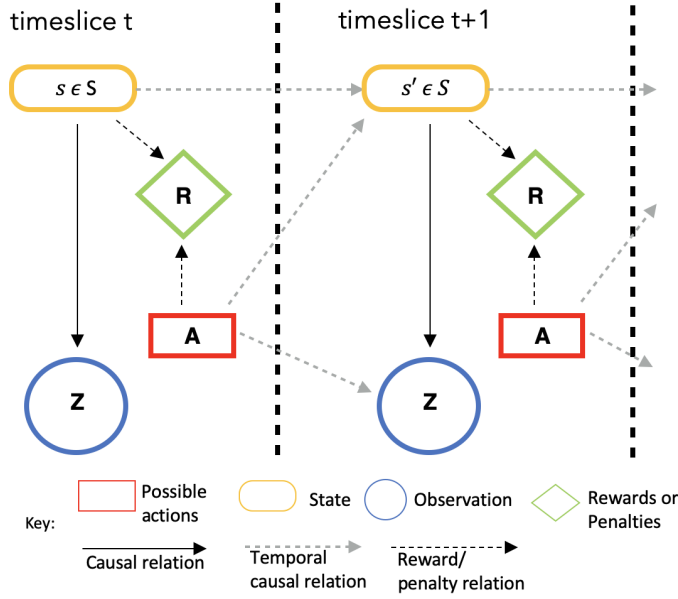


FIGURE 2.3: General POMDP: runtime model for sequential decision-making

- $S$ ,  $A$  and  $Z$  represents the system's state space, action space and observation space, respectively [4]. At each time slice, the system takes action  $a \in A$  to move from a state  $s \in S$  (Fig. 2.3, time slice  $t$ ) to  $s' \in S$  (Fig. 2.3, time slice  $t + 1$ ) and then receives an observation  $z \in Z$ .
- $S$  represents the state space, i.e. a set of distinct states  $s \in S$  the system could reach. The number of states may be finite, countably infinite or continuous. We will focus on discrete models with a finite number of states.
- $A$  represents the action space. A system seeks to influence its state by executing actions from the set  $A$ . The system's goal is to choose actions in such a way that desirable states  $s \in S$  are visited more frequently.
- $T: S \times A \times S \rightarrow [0,1]$  is the transition function. A POMDP allows action effects, which are subject to uncertainty, to be modelled. This implies that the system has a certain probability of making a transition to any state  $s \in S$  as a result of an action

execution. The stochastic nature of action effects is captured by the transition function  $T$ . Specifically, it describes a conditional probability function  $T(s, a, s') = P(s' | s, a)$  where at each time slice, the system takes action  $a \in A$  to move from a state  $s \in S$  to  $s' \in S$  and then receives an observation  $z \in Z$ . This transition function exhibits the Markov property [53], which says that the probability of transition to some state  $s$  at the next timeslice  $t + 1$  depends only on the state  $s$  and the action  $a$  at the current time slice  $t$ .

- $Z$  represents the observation space. After executing an action, the system gets an observation  $z \in Z$ . Observations corresponds to features of the environment directly perceptible by system's monitorables. In contrast, states correspond to features which are not directly observable by the system.
- $R: S \times A \rightarrow \mathbb{R}$  is the reward function and represents the preferences of the system. The system gets a reward  $R(s, a)$  for taking action  $a \in A$  at time  $t$  to arrive to the new state  $s \in S$  at time  $t + 1$ .
- $O: S \times A \times Z \rightarrow [0, 1]$  is the observation function. It describes the conditional probability function  $O(s', a, z) = P(z | s', a)$  of observing  $z \in Z$  when action  $a$  is performed and the resulting state is  $s'$ . This function models noisy sensors observations, which represent only partial information to the system since the same observation may be experienced in different states. Observations  $z \in Z$  corresponds to features of the environment directly perceptible by system's monitorables. In contrast, states  $s \in S$  correspond to features which are not directly observable by the system. In real contexts, a system should be able to use its observations  $z \in Z$  to infer its current state  $s \in S$ . We reach this goal by using Bayesian inference as is depicted below.

**Partial information on the state  $S$  and Bayesian Inference.** In a partially observable system, the system's states are not directly observable. Instead, a **belief** over possible states is maintained. Let  $b_{t-1}$  be the belief at time  $t - 1$ . If the system takes action  $a_{t-1}$  and receives observation  $z_t$  at time  $t$ , then Bayes' rule, i.e. Bayesian inference is applied to obtain the new belief  $b_t$ :

$$b_t(s') = \eta O(s', a, z) \sum_{s \in S} T(s, a, s') b_{t-1}(s) \quad (2.1)$$

where  $\eta$  is a normalizing constant [96]. A **belief**  $b$  is a probability distribution that represents the system knowledge about its current state. The next step is to choose an action based on that belief, i.e. to use a policy.

**A POMDP policy  $\pi$ .** A policy  $\pi$  defines the system strategy for all possible situations it can find [75]. In terms of a POMDP, a policy  $\pi$  defines a mapping that specifies the action  $a = \pi(b)$  at belief  $b$ . The goal is to maximise the expected value  $E$ , i.e. the possible amount of reward earned under the current belief  $b$  as is shown below:

$$V_{\pi}(b) = E\left(\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(b_t)) | b_0 = b\right) \quad (2.2)$$

The constant  $\gamma \in [0, 1)$  is the discount factor, which express preferences for immediate rewards over future ones. POMDPs provide reasoning and decision-making over time, using partial knowledge (i.e. beliefs) of the states  $s \in S$  of a running system based on runtime evidence (i.e., observations  $z \in Z$ ).

*In our proposal, the belief  $b$  about the system's state  $s \in S$ , represents the current satisficement level of the NFRs in a SAS. The policy  $\pi$ , in the expression  $a = \pi(b)$ , defines the action taken by a SAS at the current satisficement level of its NFRs.*

## 2.7 RE-STORM: a Reinforcement Learning POMDP approach

POMDP is a Reinforcement Learning (RL) technique. As such, it focuses on learning a policy  $\pi$  by interacting with the environment. The learner (e.g. the planning activity in a MAPE-K feedback loop in our case) should discover which actions yield the most reward by testing them [89]. In this work, the learner is represented as a requirements-aware runtime model, used as an online planner for decision-making and supported by our POMDP implementation.

A POMDP, uses a reward function  $R(s,a)$  that captures the immediate or short-term consequences of executing actions. To capture the long-term consequences and

compute the expected value  $E$  in Equation 2.2, we apply the Bellman's principle of optimality by following the planning activity described in Chapter 4, section 4.4.2. Specifically, during the system's execution and per each time slice, we learn a policy  $\pi$  for the current state of the system, i.e. the current belief  $b$  about the satisficement level of its NFRs.

The environment to interact with, for discovering the action that yields the most reward, is produced by the POMDP implementation of our approach, the DESPOT algorithm [96]. By using online POMDP planning, we perform online training at each time slice to determine an approximated optimal policy  $\pi$  under the current belief  $b$  about the state of the system. Afterwards, an action  $a=\pi(b)$  is performed. Details on online POMDP planning within a MAPE-K feedback loop are presented in Chapter 4.

*In RE-STORM, we use the RL approach to learn a new policy  $\pi$ , as the system runs and per each time slice.*

## 2.8 Summary

In this chapter, we have presented four main concepts related to our proposal: *uncertainty, runtime models, MAPE-K feedback loop and Partially Observable Markov Decision Processes (POMDPs)*. A POMDP is a discrete time stochastic process where the decision-maker can not directly observe the underlying state of the system. Instead, it maintains a probability distribution over the possible states. POMDPs allow our proposal to support decision-making with incomplete information.

We use runtime models as a mean to manage uncertainty during the system's execution, through augmenting the information available at design-time with information monitored at runtime. Different types of runtime models related to our work have been depicted. System, context and requirements runtime models leverage our proposals: RE-STORM and ARRoW. They both support a requirements-aware runtime model which enable decision-making under uncertainty, based on the current satisficement levels of the NFRs in a SAS.

The interaction between a runtime model and a referred system, has been framed within a *MAPE-K feedback loop*, where the core system, its context and requirements, are part of the Knowledge Base, which offers support to the monitoring, analysis and planning activities in the feedback loop.

Different types of systems take advantages of the use of runtime models. Our proposal is oriented to support the execution of *Self Adaptive Systems* (SASs) in their broad definition, i.e. SAS which may include self-awareness, context-awareness and requirements-awareness capacities.

In this work, POMDPs enable the implementation of a requirements-aware runtime model to support decision-making under uncertainty, with partial information about the current satisficement levels of the NFRs in a SAS. POMDPs integrated within a MAPE-K feedback loop, take advantage of Reinforcement Learning (RL) techniques to “learn” per each time slice, which action produce the most reward, considering the long term effect of the executing actions. Details on this approach are presented in Chapter 4.

Next, the State of the Art on decision-making under uncertainty is presented in Chapter 3.



## Chapter 3

# State of the art on Decision-Making under Uncertainty

This chapter presents the state of the art related to this dissertation: decision-making under uncertain environments, driven by the current state of the NFRs in a SAS. The results have been organised and classified using four criteria:

- $C_1$  Techniques for decision-making under uncertainty
- $C_2$  Requirements representation for decision-making
- $C_3$  Specification of preferences at design-time
- $C_4$  Runtime update of preferences

The criteria  $C_i$  take into account the two main challenges identified in Chapter 1:

- Decision-making under uncertain environments
- Update of preferences over NFRs and adaptation actions at runtime

The criteria ( $C_1$ ) Techniques for decision-making under uncertainty and ( $C_2$ ) Requirements representation for decision-making allow us to study and structure the state of the art with respect to the first challenge. The criteria ( $C_3$ ) Specification of preferences at design-time and ( $C_4$ ) Runtime update of preferences, allow to structure the state of the art with respect to the second challenge. The approaches identified following this classification, departure but also extend the techniques identified in [35]. They presented us with opportunities and the motivation to implement our proposals: RE-STORM and ARRoW, depicted in Chapters 4 and 5 respectively.

### 3.1 Techniques for decision-making under uncertainty ( $C_1$ )

Different approaches supported by runtime models show progress on dealing with decision-making under uncertainty in SASs [39]. Some approaches have scalability issues, that is, it is not yet possible to easily apply them to real domain problems. For example, In [86], Song et al. present scalability issues associated with the number of constraints. Authors in [34] suffers from scalability problems with respect to the size of its configuration space. Bencomo et al. [8] use the mathematical model provided by Dynamic Decision Networks (DDNs). However, it presents scalability issues related to its planning horizon.

Other approaches tend to be reactive, adapting in response to changes without anticipating what the subsequent adaptations needs will be. For instance, Peng et al. [70] use a PID controller to dynamically adjust the trade-off among NFRs as part of its decision-making process. They choose adaptation actions based on sensor data related only to the current system's state. They are not considering the possible future evolution of the satisficement level of the NFRs. This may result on attractive short-term actions with undesirable longer-term consequences.

Sousa et al. [87] propose the modelling of quality of service trade-offs based on utility theory. A utility function maps the possible quality levels to a normalized utility space  $U [0,1]$ , where the user is "happy" with utility values close to 1, and "unhappy" with utility values close to 0. A Solver determines the tactic that produce the highest utility. The Solver is invoked by the application before carrying out each unit of work. Elahi et al. [22] present a trade-off analysis algorithm that takes pairwise comparisons of alternative solutions to determine the best solution among several alternatives. Valid satisfaction levels that the requirements may have are enumerated with respect to the relative rankings of alternatives. To determine the best alternative for each possible goal satisfaction level, the algorithm decides the optimum alternative by using a heuristic method.

In [28], Filieri et al. introduce a mathematical framework for run-time probabilistic model checking under uncertainty. The approach generates a set of verification conditions at design-time that can be evaluated at runtime as soon as environmental

changes occur. Changes in the environment are represented as changes on the probability distributions of a Discrete Time Markov Chain (DTMC) and requirements are represented in Probabilistic Computation Tree Logic (PCTL). However, this proposal ignores requirements evolution, e.g. evolution of preferences over NFRs. Sensitivity Analysis, a tool of probabilistic model checking, is used at runtime to support self-adaptation by prioritizing an adaptation action among multiple possible alternatives. Moreno et al. in [61] present an approach that construct a Markov Decision Process (MDP) to take optimal decisions. At design-time, most of the MDP is constructed by using formal methods. The approach considers at design-time the many possible system's states, and combinations of tactics. At runtime, the adaptation decision is made by solving the MDP through stochastic dynamic programming. In [15] Camara et al. introduce a technique based on probabilistic model checking of stochastic multi-player games (SMG) that enables decision-making under uncertainty. The authors present a case study where equal importance is assumed for the NFRs. The PRISM language for Stochastic Multiplayer Games and Markov Decision Processes (MDPs) are used to build a PRISM SMG model, where the reasoning about strategies of adaptation is performed. They also assume that there is no uncertainty on the impact of the adaptation actions on NFRs, i.e. the impact is deterministic. The approaches in [28, 61, 15] scarce of explicit representation of preferences for NFRs and adaptation actions. Additionally, initial assumptions settled for foreseen contexts at design-time, can not be updated at runtime when new and possible unexpected contexts arise during the system's execution.

## 3.2 Requirements representation for decision-making ( $C_2$ )

Fig. 3.1 shows different approaches that use goal models for decision-making under uncertainty and requirements representation. A goal-oriented model is a framework for capturing relationships between goals. Goal models have been used to model requirements and decision-making of SASs [7]. Next, several model-based approaches to represent requirements and their relationships are presented.

Based on goal models			Based on goal models leveraged by Markov Decision Processes (MDPs)	
Garcia-Galan et al. [34]	Song et al. [84]	Peng et al. [68]	Filieri et al. [28]	Bencomo et al. [8]
Sousa [85]	Letier et al. [56]	Liaskos et al. [57]	Camara et al. [15]	Moreno et al. [60]
	Elahi et al. [22]	Bowers et al. [13]		

FIGURE 3.1: Goal models for decision-making under uncertainty and requirements representation

Song et.al. [86] presents an approach where adaptation goals represent functional and NFRs. Structural runtime models are transformed into a Constraint Satisfaction Problem (CSP) [43] to keep uptodate the current system's context and configuration to facilitate posterior reasoning. Letier et.al. [57] propose the representation of system's goals as part of an Architecture Decision Model (ADM). The system's goals include NFRs such as performance and reliability. They are partitioned into two main categories: G+ (goals to be maximized) and G- (goals to be minimized). In [34], Garcia-Galan et al. propose a preference-based analysis method to identify service configurations that maximize tenants' satisfaction at runtime. The approach involves the creation of an Extended Feature Model (EFM) and a Semantic Ontology of User Preferences (SOUP) model [34]. The EFM is a variability model and represents functional requirements and NFRs. In [58], Liaskos et al. present a goal model extension to support requirements. The goals are classified as either preference goals or mandatory. The preference goals are used to evaluate alternative ways to achieve mandatory goals. Weights are assigned to preference goals using a quantitative requirement prioritization scheme: Analytic Hierarchy Process (AHP) [78]. Peng et al. [70] propose the use of a goal model, with NFRs and functional requirements. They also represent their contribution relationships, and its quantitatives expected satisfaction. Sousa et al. [87] propose an approach that represents NFRs as quality attributes. A model represents the trade-off among the NFRs and supports the coordination of the resources usage in a software application. Bencomo et al. [8] use random variables in a Dynamic Decision Networks (DDNs) to represent the NFRs in

a SAS. Bowers et al. [13] present Providentia, an approach to automatically optimize the selection of functional requirements and their corresponding weights to satisfy NFRs objectives. In Providentia, a goal model represents both: functional and NFRs.

In [28], Filieri et al. use Discrete Time Markov Chain (DTMC) and Probabilistic Computation Tree Logic (PCTL) to represent requirements. Authors in [61] present an approach that constructs a Markov Decision Process (MDP) to support system goals (e.g. NFRs) and taking optimal decisions. In [15], Camara et al. introduce a technique based on probabilistic model checking of stochastic multi-player games (SMG) and MDPs to represent NFRs (e.g. security and user experience).

The approaches above use different types of runtime goal models to represent requirements. Some of them [28, 61, 15] use sequential decision-making frameworks (e.g. MDPs) and assume full observability of the current state, e.g. NFRs of the system.

### 3.3 Specification of preferences at design-time ( $C_3$ )

In this section we explore different model-based approaches to determine up to what extent initial preferences about NFRs and adaptation actions in a SAS are specified at design-time. A summary is shown in Fig. 3.2.

Stakeholders	Stakeholders and MCDA methods	No representation	Simulator
Garcia-Galan et al. [34]	Letier et al. [56]	Filieri et al. [28]	Bowers et al. [13]
Bencomo et al. [8]	Liaskos et al. [57]	Moreno et al. [60]	
Song et al. [84]	Elahi et al. [22]	Camara et al. [15]	
Peng et al. [68]			
Sousa [85]			

FIGURE 3.2: Specification of preferences at design-time

The most widespread method for preference elicitation about NFRs and adaptation actions is by asking system's stakeholders. Bencomo et al. [8], Sousa et al. [87] and Song et al. [86] elicit initial preferences with information from system's stakeholders. Peng et al. [70] from a range of values between 1 and 10, initially set the preferences of the NFRs to 5. Authors in [57, 22, 58] complement this approach by using multi criteria decision analysis methods (MCDA), e.g. Analytic Hierarchy Process (AHP). Elahi et. al. [22] incorporate Stakeholders' preferences by using a MCDA-based method: the Even Swaps Method [42]. This method determines preferences based on an ordinal scale avoiding the elicitation of numerical weights. Authors in [34] define the system's preferences by using initial configurations defined by stakeholders.

Other approaches [28, 61, 15] determine at design-time possible requirements violations, verification conditions and uncertainty functions that should be evaluated at runtime based on monitored data collected from the environment. These approaches scarce of explicit representation of preferences for NFRs and architectural actions or assume equal importance for them [15].

In [13], Bowers et al. use an executable simulation of a SAS to initially determine weights (a.k.a. preferences) of functional requirements to satisfy NFRs objectives. These weights are used for decision-making during the system's execution but they can not be updated at runtime.

### 3.4 Reassessment and update of preferences at runtime ( $C_4$ )

Initial specifications and assumptions about a SAS may be defined at design-time. At runtime, such design assumptions can prove to be wrong or not valid anymore [7].

Not autonomous Update of preferences		Autonomous Update of preferences
<div>Song et al. [84]</div> <div>Sousa et al. [85]</div>		<div>Peng et al. [68]</div>

FIGURE 3.3: Runtime reassessment and update of preferences

Reassessment and update of preferences is a field that needs more exploration. Different approaches [58, 57, 74, 13, 15] work with initial design-time preferences and do not support preference update. Bencomo et al. [8] identify at runtime the need of preferences reassessment.

Some other approaches admit preferences update but not in an autonomous way as they require user intervention. In [86], if users do not agree with the final solution, they can revise the configuration values. The new users' preferences elicited allow tuning the weights of existing goals or the generation of new ones. Authors in [87] use a user interface for manipulating thresholds on preferences at runtime. Peng et. al. [70] show first results of autonomous preference updating by monitoring the environment at runtime and using a preference tuning algorithm.

### 3.5 Summary

In this chapter we have described several approaches for decision-making in SASs. Different research gaps have been identified, which represent opportunities for the implementation of our approaches. A summary is presented as follows.

- *Criteria (C<sub>1</sub>) Techniques for decision-making under uncertainty and (C<sub>2</sub>) Requirements representation for decision-making.* Several authors [86, 70, 34, 22, 87] use runtime goal models with reactive control decision-making [4], i.e. they ignore prediction uncertainty during decision-making which often results in sub-optimal decisions over the long term. Some others [28, 61, 15] use more refined runtime models, e.g. Markov Decision Processes (MDPs) to represent requirements and their evolution over time. They assume full observability of the current state of the system as part of their decision-making process. A better runtime representation of requirements that can deal with modelling the nature associated to NFRs and therefore drive decision-making, is needed.
- *Criteria (C<sub>3</sub>) Specification of preferences at design-time and (C<sub>4</sub>) Runtime update of preferences.* A common finding among the studied approaches with the exception of [70, 86, 87] up to some extent, is that at design-time, initial preferences (based on information provided by system's experts or learned from foreseen

possible contexts by using simulators [13]) may be specified. However, these approaches do not allow the update of preferences when new scenarios, not previously foreseen, may arise at runtime. Also, scalability issues are an important barrier to make feasible the overcoming of these challenges.

Our proposals represents a contribution regarding these matters and are presented in the next chapters 4 and 5. Next, in chapter 4, based on the tailored version of a state-of-the-art POMDP implementation, we present RE-STORM, a technique that uses a requirements-aware runtime model for decision-making in SASs and which tackles some of the research issues identified before. In RE-STORM the decision-making process takes into account the future evolution of the satisficement levels of the NFRs and the actions taken in the system, i.e. the approach is able to do projections on the future during decision-making. It is also possible in RE-STORM, to update initial preferences in the requirements-aware runtime model. In chapter 5, ARRoW, an approach to supports the runtime update of preferences over NFRs and adaptation decisions is presented.



## Chapter 4

# RE-STORM: Requirements Trade-offs for Self-adaptation using POMDPs

### 4.1 Overview

This chapter presents RE-STORM, a formal analysis technique, based on Partially Observable Markov Decision Processes (POMDPs) to support decision-making driven by the current satisficement level of NFRs in a self-adaptive system (SAS) [36, 37] and evidence collected at runtime. The trade-offs between NFRs (i.e. the qualities of a system) are embodied as a POMDP in the context of the MAPE-K loop. Based on evidence about events monitored at runtime and using Bayesian learning, the levels of satisficement of the NFRs are inferred and updated during execution using runtime models which reside in the Knowledge Base (K) [9]. Specifically, we have casted the decision-making problem of a SAS and the trade-off of the NFRs, such as reliability and performance, in terms of a POMDP decision problem. The approach involves:

- The specification of the NFRs in terms of a POMDP model [71, 14].
- An architecture that leverages the different activities of the MAPE-K loop to support decision-making under uncertainty based on the current satisficement level of the NFRs in a SAS [36, 9].

Next, the details of the approach are explained.

## 4.2 Case study: Remote data mirroring (RDM) system

As an example to demonstrate the specification of NFRs and the architecture to support our approach, let us consider the case of the Remote Data Mirroring (RDM) self-adaptive system (SAS) [73]. The RDM SAS is composed of data servers and network links. It must replicate and distribute data in an efficient manner by minimizing consumed bandwidth and providing assurance that distributed data is not lost or corrupted [47, 73]. The RDM can be configured by using two different topologies: *Minimum Spanning Tree* (MST) and *Redundant Topology* (RT). These two possible configurations allow the system to selectively activate and deactivate network links to change its overall topology at runtime [29].

Fig. 4.1 shows a goal-oriented requirements model of the RDM SAS, represented in iStar (i\*) notation [97].

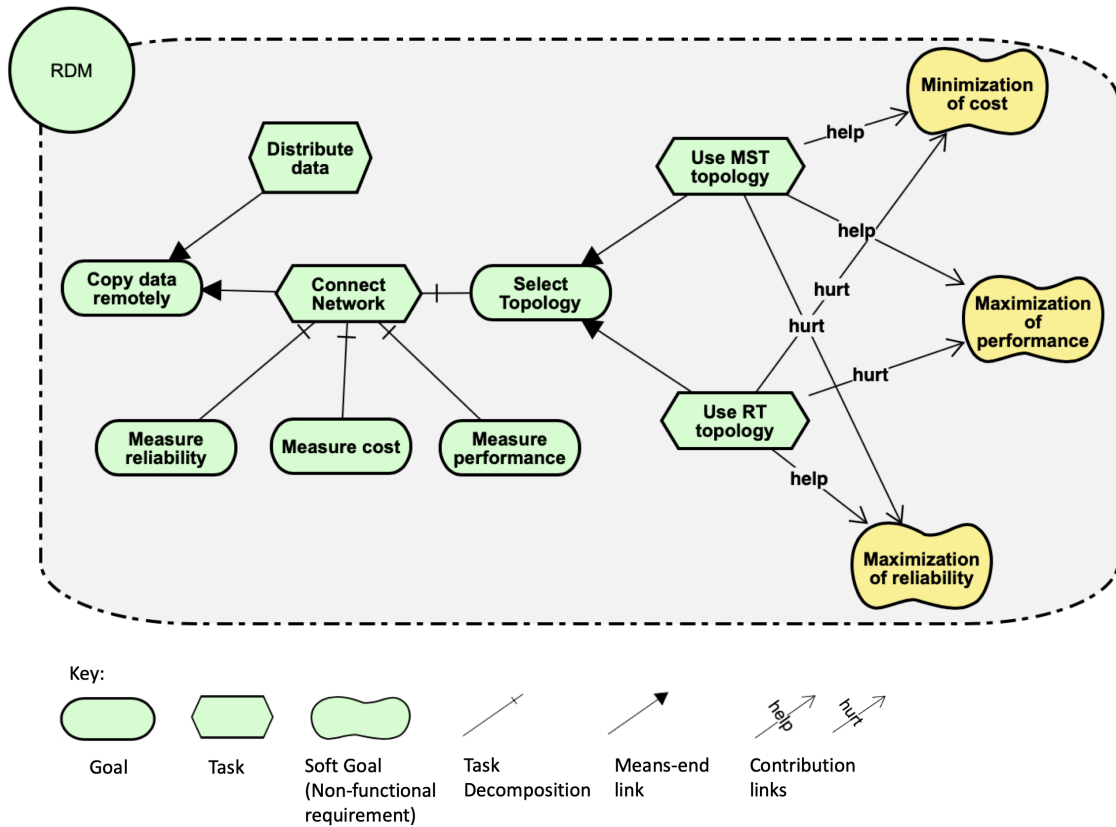


FIGURE 4.1: Goal Model - RDM Case Study (i\* notation)

In these models, a node represents a goal and an edge represents a specific type of refinement. While goals that represent required functional properties can be evaluated in an absolute manner, a special category of goals called soft goals can only

be satisfied [21] to a certain degree. Soft goals typically represent non-functional requirements (e.g. *Maximization of Reliability* (MR) and *Minimization of Cost* (MC)) that constrain how functionality should be delivered to stakeholders. For example, the goal model in Fig. 4.1 captures the following NFRs for the RDM SAS: *Minimization of Cost* (MC), *Maximization of Reliability* (MR), and *Maximization of Performance* (MP). In order to satisfy these NFRs, the RDM must achieve functional goals such as constructing a connected network and distributing data. These functional goals can be achieved through alternative goal realization strategies, modelled as tasks in iStar (i\*), that include constructing different network topologies, such as a MST or a RT topology.

The RDM SAS self-adapts by reconfiguring itself at runtime according to changes in the environment, which may include either delayed or dropped messages and network link failures. Each network link in the RDM system involves an operational cost, which is primarily measured in terms of inter-site network traffic [47]. Each link also has a measurable throughput, latency, and loss rate. The reliability, performance and cost of the RDM SAS are determined by these metrics according to the following trade-off: RT topologies offer a higher level of reliability than MST topologies (See Fig. 4.1, the “help” contribution link between RT topology and *Maximization of Reliability*). However, the costs of maintaining a more reliable RT topology may be prohibitive in some contexts (See Fig. 4.1, the “hurt” contribution link between RT topology and *Minimization of Cost*). The performance, given the greater number of data redundancy, is also reduced. Both configurations (i.e. MST and RT topologies) provide their own levels of reliability, performance and cost which are taken into account while estimating at runtime the levels of satisfaction of the NFRs observed, i.e. MC, MR and MP.

Next, the details on the representation of these NFRs as part of a POMDP model are presented.

### 4.3 NFRs decision-making as a POMDP

The use of equivalences between NFRs levels of satisfaction and the states in a POMDPs [37], which underpins RE-STORM, allow the specification and trade-offs

of NFRs in terms of the states of a POMDP model. These equivalences are depicted in this work as a set of mapping rules. The use of the mapping rules allow a POMDP to support the decision-making process driven by the current satisficement levels of the NFRs in a SAS. These rules are described in detail in sections 4.3.2 and 4.3.3.

Fig. 4.2 shows a POMDP representation of the RDM SAS and its NFRs with the entire set of relationships among the POMDP's elements.

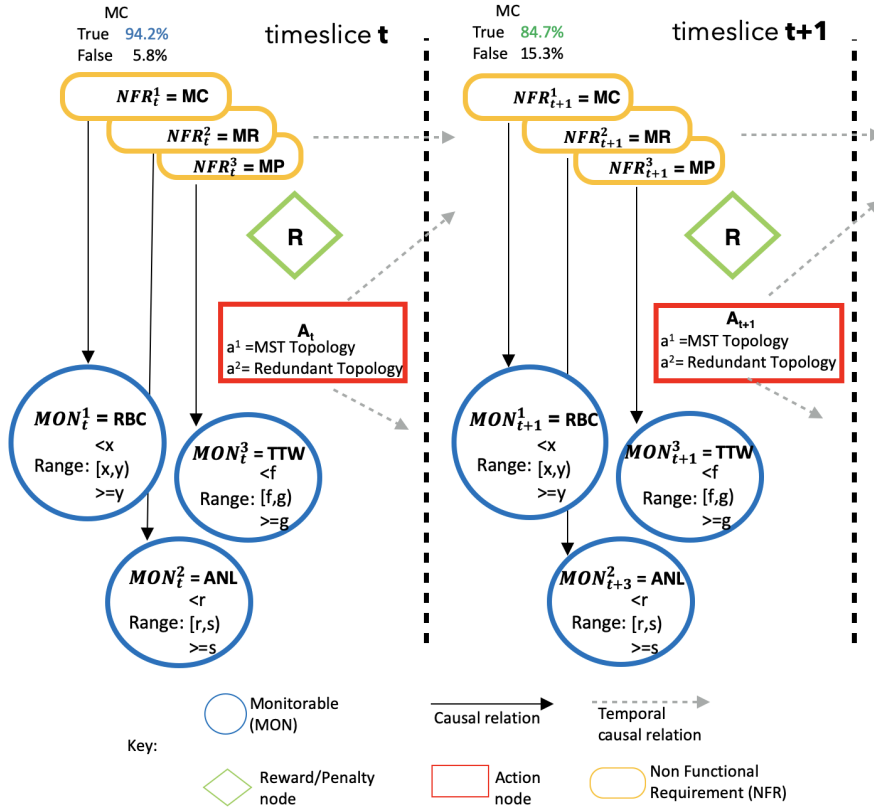


FIGURE 4.2: POMDP requirements-aware model - RDM Case Study

Note how whilst in Fig. 4.1, a high level static view of the impacts of the topologies RT and MST over the NFRs is presented (i.e. the impacts are considered constant over time), in Fig. 4.2, the impacts are dynamic and based on a temporal causal relation (specified by gray dotted arrows), which represents the probability distributions of the transition function in a POMDP.

### 4.3.1 RDM SAS - POMDP representation

In our proposal, the states  $s \in S$  in a POMDP represent the levels of satisficement of the NFRs of the RDM SAS. Note in Fig. 4.2 that the NFRs  $NFR_t^1$ ,  $NFR_t^2$ , and

$NFR_i^3$  are not directly observable. Instead, observations are obtained by using monitoring variables (called MON variables). Three MON variables are specified in the RDM SAS: *Ranges of Bandwidth Consumption* (RBC) (i.e.,  $RBC < x$ ,  $RBC \in [x, y)$  and  $RBC \geq y$ ), *Active Network Links* (ANL) (i.e.,  $ANL < r$ ,  $ANL \in [r, s)$  and  $ANL \geq s$ ) and *Total Time for Writing* (TTW) (i.e.,  $TTW < f$ ,  $TTW \in [f, g)$  and  $TTW \geq g$ ). TTW is a performance measure, which is the sum of the times to write each copy of data on each remote site [47].

In the RDM SAS, the pair values  $(x, y)$ ;  $(r, s)$ ; and  $(f, g)$  represent range boundaries for the MON variables RBC, ANL and TTW respectively. In the case of RBC and TTW, the lower the monitored values, the greater the satisficement level of MC and MP respectively. Conversely, in the case of ANL, the higher the monitored values, the greater the satisficement level of MR.

The MON variables are influenced by their related NFRs and actions (See Fig. 4.2, causal and temporal relations). It is also possible to determine the effects of the opposite influence. By applying Bayesian Inference [53], we can identify how the state of a NFR is influenced by the current monitored values of its related MON variable. This principle, together with the transition and observation functions of a POMDP are used in Equation (2.1) to infer the beliefs, i.e. the probability distributions about the current levels of satisficement of the NFRs in the RDM SAS.

**Definition 1.** *In our approach, a POMDP is a requirements-aware model that serves as a runtime model to support decision-making driven by the trade-off of the current levels of satisficement of NFRs in a SAS.*

Next, we explain how POMDPs offer a frame for modelling NFRs trade-offs to support the decision-making process when new evidence is collected from the environment.

#### 4.3.2 From Requirements to POMDPs: a requirements-aware model

Different mapping rules to represent NFRs in terms of states  $s \in S$ , rewards  $R(s, a)$  and observations  $z \in Z$  in a POMDP are explained as follows.

- a) **NFRs as POMDP states  $s \in S$ .** Each state  $s \in S$  in a POMDP is represented by a set of possible values (T=True or F=False) of the NFRs from  $NFR^1$  to  $NFR^n$ , where “ $n$ ” is the number of NFRs in the model. The values True and False are not directly observable. They have associated a probability distribution. Given the identified relationship between NFRs and states in a POMDP, we formalize the following mapping rule:

**Mapping Rule 1.** A state  $s \in S$  represents a set of values (combinations of True or False) of the NFRs  $NFR^1, \dots, NFR^n$ . The values of the NFRs are not directly observable. They have associated a belief, i.e. a probability distribution.

In the RDM SAS, three NFRs were specified: *Minimization of Cost* (MC), *Maximization of Reliability* (MR) and *Maximization of Performance* (MP). The related states  $s \in S$  of a POMDP are shown in Table 4.1.

TABLE 4.1: RDM SAS - NFRs and states  $s \in S$

State (S)	NFR <sup>1</sup> =MC	NFR <sup>2</sup> =MR	NFR <sup>3</sup> =MP	Belief / Probability Distribution P(S)
$s_1$	T	T	T	$P(s_1)$
$s_2$	T	T	F	$P(s_2)$
$s_3$	T	F	T	$P(s_3)$
$s_4$	T	F	F	$P(s_4)$
$s_5$	F	T	T	$P(s_5)$
$s_6$	F	T	F	$P(s_6)$
$s_7$	F	F	T	$P(s_7)$
$s_8$	F	F	F	$P(s_8)$

The number of states  $s \in S$  is represented as  $|S|$  and depends on the number of NFRs,  $|NFR|$ , and the number of values, i.e.  $|NFR_{sv}|$ , for each NFR.  $|S|$  is computed as follows:

$$|S| = |NFR_{sv}|^{|NFR|} \quad (4.1)$$

The case study presented in fig. 4.2 contains 3 NFRs. Therefore, the number of possible states  $|S|$  is  $(2)^3=8$ .

- b) **NFRs, their probabilities of satisficement.** NFRs are the qualities to be satisfied in a system [21]. Satisficement and satisficing are portmanteau words of satisfy and suffice, and refer to decision-making strategies that seek to meet an acceptability threshold. This is because measuring the satisfaction of NFRs is challenging; it may not be possible to conclude that a NFR is fully satisfied. Instead, they can be labelled as sufficiently satisficed [21]. Probabilistic approaches have been used to model the lack of crispness about the satisfiability nature of NFRs [8, 27, 24].

In our approach, theoretically each NFR has two possible values: True and False, however they are not directly observable. Therefore, we use Equation (2.1) to infer the current probability distribution about these values. The probability distribution represent the satisficement level of the NFR.

*In RE-STORM, the satisficement level of each  $NFR^i$  is represented by the probability distribution  $P(NFR^i=True)$  and  $P(NFR^i=False)$ .*

For instance, in Fig. 4.2, the probability distribution for the satisficement level of MC at time slice  $t + 1$ , given the previous adaptation action  $A_t$  and the previous  $NFR_t$  states (i.e.  $NFR_t^1$ ,  $NFR_t^2$  and  $NFR_t^3$ ), is represented by:

$$P(MC_{t+1}=True|NFR_t, A_t) \text{ and } P(MC_{t+1}=False|NFR_t, A_t)$$

Probabilities are excellent tools to model the levels of satisficement based on observations. The higher the probability  $P(MC_{t+1}=True|NFR_t, A_t)$  computed by Equation (2.1), the higher the belief that MC is being satisfied.

*In RE-STORM, for each time slice, we use these probabilities (i.e. beliefs) to choose the best self-adaptation action  $a \in A$ .*

- c) **NFRs, preferences and the reward function  $R(s,a)$  in a POMDP.** A reward function assigns a cardinal scale to each 2-tuple: state and action of the system [45], indicating its desirability. Following Mapping Rule 1, the state “ $s$ ” in a reward function  $R(s,a)$  is represented by a set of possible values of the NFRs in a SAS.

Table 4.2 shows the reward values  $R(s,a)$  related to our case study.

TABLE 4.2: RDM SAS - Reward values  $R(s,a)$

Reward values $R(s,a)$	Action (A)	State (S)		
		NFR <sup>1</sup> =MC	NFR <sup>2</sup> =MR	NFR <sup>3</sup> =MP
$r_1=0.0919$	$a^1=MST$	T	T	T
$r_2=0.0943$	$a^1=MST$	T	T	F
$r_3=0.0896$	$a^1=MST$	T	F	T
$r_4=0.0377$	$a^1=MST$	T	F	F
$r_5=0.1014$	$a^1=MST$	F	T	T
$r_6=0.0660$	$a^1=MST$	F	T	F
$r_7=0.0306$	$a^1=MST$	F	F	T
$r_8=0.0023$	$a^1=MST$	F	F	F
$r_9=0.1014$	$a^2=RT$	T	T	T
$r_{10}=0.0731$	$a^2=RT$	T	T	F
$r_{11}=0.0660$	$a^2=RT$	T	F	T
$r_{12}=0.0613$	$a^2=RT$	T	F	F
$r_{13}=0.0660$	$a^2=RT$	F	T	T
$r_{14}=0.0377$	$a^2=RT$	F	T	F
$r_{15}=0.0542$	$a^2=RT$	F	F	T
$r_{16}=0.0259$	$a^2=RT$	F	F	F

For example, we observe that when the levels of satisficement of the NFRs MC and MP are the only ones below their expected minimum values, i.e. MC=False, MR=True and MP=False (See rows  $r_6$  and  $r_{14}$  in Table 4.2) the specification suggested by experts favours the topology MST (row  $r_6=0.0660$ ) over the topology RT (row  $r_{14}=0.0377$ ). This suggests that under this specific context (i.e. when MP and MC are considered not being satisficed), the *Minimum Spanning Tree topology* (MST) as an adaptation action would be preferred over a *Redundant topology* (RT) action that would offer more reliability. All other possible reward values  $R(s,a)$  in Table 4.2, favour the topologies MST or RT, based on information provided by the system’s experts at design-time.



*At design-time, the **reward values**  $R(s,a)$  are initially defined. However, under some specific situations detected at the running system, they may need to be updated at run-time.*

The reward values  $R(s,a)$  (a.k.a. preferences or weights in RE-STORM) are used by a POMDP to choose an optimal action after applying the Bellman's principle of optimality [36]. Given the above, we present the following mapping rule:

**Mapping Rule 2.** *The reward values  $R(s,a)$  represent the preferences over the execution of an action  $a \in A$  at time  $t$ , that as a result, produces a new state  $s$ , i.e. a new set of satisficement levels of the NFRs (based on Mapping Rule 1) at time  $t+1$ .*

The number of possible reward values  $R(s,a)$  is represented by  $|R|$  and depends on: the number of NFRs represented by  $|NFR|$ , the number of state values  $|NFR_{sv}|$  for each NFR and the number of actions  $|A|$ .  $|R|$  is computed using the equation 4.2 as follows:

$$|R| = |NFR_{sv}|^{|NFR|} * |A| \quad (4.2)$$

The RDM case study contains 3 NFRs, 2 possible values for each NFR and 2 possible actions (i.e. MST and RT), therefore, the number of reward values  $|R|$  is  $(2)^{3*2}=16$ .

- d) **MON variables and POMDP observations  $z \in Z$ .** MON variables provide the observations required to infer at runtime the current satisficement levels of the NFRs in a SAS. Each observation  $z \in Z$  in a POMDP, represents a set of possible observation values obtained from  $MON^1$  to  $MON^k$ , where  $k$  is the number of MON variables.

**Mapping Rule 3.** *An observation  $z \in Z$  represents a set of observation values of the MON variables  $MON^1, ..., MON^k$ . MON variables are variables of the environment that*

affect the levels of satisficement of the NFRs in a SAS and support their inference by using Equation (2.1).

The RDM SAS involves three MON variables, RBC, ANL and TTW. Each of them has three possible values representing ranges of monitored observations. The observations  $z$  associated with these MON variables are shown in Table 4.3.

TABLE 4.3: RDM SAS - MON variables and observations

Observations (Z)	MON <sup>1</sup> =RBC	MON <sup>2</sup> =ANL	MON <sup>3</sup> =TTW
$z_1$	<x	<r	<f
$z_2$	<x	<r	in f_g
$z_3$	<x	<r	>=g
$z_4$	<x	in r_s	<r
$z_5$	<x	in r_s	in r_s
$z_6$	<x	in r_s	>=s
$z_7$	<x	>=z	<r
$z_8$	<x	>=z	in r_s
$z_9$	<x	>=z	>=s
...	...	...	...
$z_{25}$	>=y	>=z	<r
$z_{26}$	>=y	>=z	in r_s
$z_{27}$	>=y	>=z	>=s

The MON variables RBC, ANL, and TTW are associated to the NFRs MC, MR and MP respectively (See Fig. 4.2). The number of observations  $z \in Z$  is represented as  $|Z|$  and depends on the number of MON variables  $|MON|$  and the number of possible values  $|MON_{ov}|$  for each MON variable. The number of observations  $|Z|$  is computed as follows:

$$|Z| = \prod_{i=1}^{|MON|} |MON_{ov}| \quad (4.3)$$

In the case of the RDM SAS, we have 3 MON variables and 3 possible range values for each one, therefore the number of observations  $|Z|$  is  $3*3*3=27$ .

### 4.3.3 From requirements to POMDPs: NFRs inference evolution

In RE-STORM, per each time slice during the system's execution, the current satisficement levels of the NFRs is inferred by using Equation (2.1). This inference considers the transition and observation functions in a POMDP and represents a

time-based relationship among POMDP elements (NFRs inference evolution). The mapping rules to represent NFRs and MON variables in terms of the transition and observation functions respectively are presented as follows.

- a) **NFRs and the POMDP transition function.** In a POMDP, the transition function  $T(s, a, s') = P(s' | s, a)$  represents the probability of the system making a transition from state  $s$  to state  $s'$  when action  $a$  is executed under the current state  $s$ . The following mapping rule allows the representation of the transition function in a POMDP with respect to the state values of the NFRs in a SAS.

**Mapping rule 4.** *The transition function  $T(s, a, s') = P(s' | s, a)$ , represents a system taking an action 'a' under the current satisficement level of its related NFRs to update the next time slice with a new satisficement level that better meets its NFRs.*

Based on the previous rule, the transition function is derived as a function of the NFRs:

$$T(s, a, s') = P(NFR^{(1)} \dots NFR^{(n)} | NFR^{(1)} \dots NFR^{(n)}, A) \quad (4.4)$$

where  $NFR^{(i)}$  and  $NFR'^{(i)}$  represent the NFR "i" at the time slices  $t$  and  $t+1$  respectively,  $\forall_i \in [1, n]$ . Using Bayes' theorem [69, 53], the transition model can be factored as a product of conditional distributions. Let us apply this concept on the RDM SAS, where two possible actions exist: *Minimum Spanning Tree* (MST) and *Redundant topology* (RT). These actions affect the NFRs *Minimization of Cost* (MC), *Maximization of Reliability* (MR) and *Maximization of Performance* (MP). Therefore, the factored transition model for this example is as follows:

$$\begin{aligned} T(s, a, s') &= P(MC' | MC, MR, MP, a) \\ &P(MR' | MC, MR, MP, a) P(MP' | MC, MR, MP, a) \end{aligned} \quad (4.5)$$

As it is observed in Equation (4.5) and visually in Fig. 4.2,  $MC'$ ,  $MR'$  and  $MP'$  are influenced by both, the previous action  $a \in A$ , and the previous state of MC, MR and MP, i.e. they are interdependent. The conditional probability tables (CPTs)

for the NFRs of the RDM SAS are shown in Tables 4.4, 4.5 and 4.6.

TABLE 4.4: CPT MC:  $P(MC' | MC, MR, MP, a)$

NFR <sup>1</sup> : Minimization of Cost (MC)					
Action $A_{t-1}$	$MC_{t-1}$	$MR_{t-1}$	$MP_{t-1}$	$P(MC_t=T)$	$P(MC_t=F)$
$a^1_{MST}$ Topology	T	T	T	0.9	0.1
$a^1_{MST}$ Topology	T	T	F	0.88	0.12
$a^1_{MST}$ Topology	T	F	T	0.92	0.08
$a^1_{MST}$ Topology	T	F	F	0.9	0.1
$a^1_{MST}$ Topology	F	T	T	0.85	0.15
$a^1_{MST}$ Topology	F	T	F	0.83	0.17
$a^1_{MST}$ Topology	F	F	T	0.87	0.13
$a^1_{MST}$ Topology	F	F	F	0.85	0.15
$a^2_{RT}$ Topology	T	T	T	0.86	0.14
$a^2_{RT}$ Topology	T	T	F	0.84	0.16
$a^2_{RT}$ Topology	T	F	T	0.88	0.12
$a^2_{RT}$ Topology	T	F	F	0.86	0.14
$a^2_{RT}$ Topology	F	T	T	0.73	0.27
$a^2_{RT}$ Topology	F	T	F	0.71	0.29
$a^2_{RT}$ Topology	F	F	T	0.75	0.25
$a^2_{RT}$ Topology	F	F	F	0.73	0.27

Key:  $MC_{t-1}$  = MC at time t -1       $MC_t$  = MC at time t  
 $A_{t-1}$  = set of actions  $a^k$  at time t-1,  $\forall k \in [1,2]$

TABLE 4.5: CPT MR:  $P(MR' | MC, MR, MP, a)$

NFR <sup>1</sup> : Maximization of Reliability (MR)					
Action $A_{t-1}$	$MC_{t-1}$	$MR_{t-1}$	$MP_{t-1}$	$P(MR_t=T)$	$P(MR_t=F)$
$a^1_{MST}$ Topology	T	T	T	0.91	0.09
$a^1_{MST}$ Topology	T	T	F	0.93	0.07
$a^1_{MST}$ Topology	T	F	T	0.89	0.11
$a^1_{MST}$ Topology	T	F	F	0.91	0.09
$a^1_{MST}$ Topology	F	T	T	0.93	0.07
$a^1_{MST}$ Topology	F	T	F	0.95	0.05
$a^1_{MST}$ Topology	F	F	T	0.91	0.09
$a^1_{MST}$ Topology	F	F	F	0.93	0.07
$a^2_{RT}$ Topology	T	T	T	0.95	0.05
$a^2_{RT}$ Topology	T	T	F	0.97	0.03
$a^2_{RT}$ Topology	T	F	T	0.93	0.07
$a^2_{RT}$ Topology	T	F	F	0.95	0.05
$a^2_{RT}$ Topology	F	T	T	0.97	0.03
$a^2_{RT}$ Topology	F	T	F	0.99	0.01
$a^2_{RT}$ Topology	F	F	T	0.95	0.05
$a^2_{RT}$ Topology	F	F	F	0.97	0.03

Key:  $MR_{t-1}$  = MR at time t -1       $MR_t$  = MR at time t  
 $A_{t-1}$  = set of actions  $a^k$  at time t-1,  $\forall k \in [1,2]$

TABLE 4.6: CPT MP:  $P(MP' | MC, MR, MP, a)$

NFR <sup>1</sup> : Maximization of Performance (MP)					
Action $A_{t-1}$	MC $t-1$	MR $t-1$	MP $t-1$	$P(MP_t=T)$	$P(MP_t=F)$
$a^1_{MST}$ Topology	T	T	T	0.9	0.1
$a^1_{MST}$ Topology	T	T	F	0.85	0.15
$a^1_{MST}$ Topology	T	F	T	0.92	0.08
$a^1_{MST}$ Topology	T	F	F	0.87	0.13
$a^1_{MST}$ Topology	F	T	T	0.88	0.12
$a^1_{MST}$ Topology	F	T	F	0.83	0.17
$a^1_{MST}$ Topology	F	F	T	0.9	0.1
$a^1_{MST}$ Topology	F	F	F	0.85	0.15
$a^2_{RT}$ Topology	T	T	T	0.82	0.18
$a^2_{RT}$ Topology	T	T	F	0.75	0.25
$a^2_{RT}$ Topology	T	F	T	0.84	0.16
$a^2_{RT}$ Topology	T	F	F	0.77	0.23
$a^2_{RT}$ Topology	F	T	T	0.8	0.2
$a^2_{RT}$ Topology	F	T	F	0.73	0.27
$a^2_{RT}$ Topology	F	F	T	0.82	0.18
$a^2_{RT}$ Topology	F	F	F	0.75	0.25

Key:  $MP_{t-1}$  = MP at time  $t-1$        $MP_t$  = MP at time  $t$   
 $A_{t-1}$  = set of actions  $a^k$  at time  $t-1$ ,  $\forall k \in [1,2]$

The transition function  $T(s, a, s')$ , i.e. the probabilities in Tables 4.4, 4.5, and 4.6, are used during the system execution in Equation (2.1), to compute the current probability distribution (i.e. belief) about the satisficement level of the NFRs in the RDM SAS.

- b) **Monitoring (MON) variables and the POMDP observation function.** MON variables allow to infer at runtime the current satisficement levels of the NFRs in a system. The next mapping rule allows to represent the observation function  $O(s', a, z) = P(z|s', a)$  of a POMDP in relation to the values of the MON variables in a SAS.

**Mapping rule 5.** The observation function  $O(s', a, z) = P(z|s', a)$ , represents a system that gets observations  $z$  from its monitoring variables, under the current state of their related NFRs and after taking action ' $a$ ' in the previous time slice.

The observation function derived from Mapping Rule 5, is represented as follows:

$$\begin{aligned} O(z, a, s') &= P(MON^1 | NFR^1 \dots NFR^n, A) \\ &P(MON^2 | NFR^2 \dots NFR^n, A) \dots \\ &P(MON^l | NFR^n \dots NFR^n, A) \end{aligned} \quad (4.6)$$

In the case of the RDM SAS, three NFRs: *Minimization of cost* (MC), *Maximization of Reliability* (MR) and *Maximization of Performance* (MP) exist, that affect the MON variables *Ranges of Bandwidth Consumption* (RBC), *Active Network Links* (ANL) and *Total Time for Writing* (TTW) respectively. The factored observation function for this example is:

$$\begin{aligned} O(z, a, s') &= P(RBC | MC', a) P(ANL | MR', a) \\ &P(TTW | MP', a) \end{aligned} \quad (4.7)$$

Similar to the transition function, the observation function also requires initial conditional probabilities for the MON variables. The conditional probabilities for the MON variables above are shown in Tables 4.7, 4.8, and 4.9.

TABLE 4.7: CPT RBC:  $P(RBC | MC', a)$

OBS <sup>1</sup> : Ranges of Bandwidth Consumption (RBC)				
Action $A_{t-1}$	$MC_t$	$P(RBC_{t+1} < x)$	$P(RBC_{t+1} \text{ in } [x, y])$	$P(RBC_{t+1} >= y)$
$a^1$ =MST Topology (MST)	T	0.8	0.15	0.05
$a^1$ _MST Topology (MST)	F	0.72	0.18	0.1
$a^2$ _Redundant Topology (RT)	T	0.78	0.16	0.06
$a^2$ _Redundant Topology (RT)	F	0.68	0.2	0.12

TABLE 4.8: CPT ANL:  $P(ANL | MR', a)$

OBS <sup>2</sup> : Active Network Links (ANL)				
Action $A_{t-1}$	$MR_t$	$P(ANL_{t+1} < r)$	$P(ANL_{t+1} \text{ in } [r, s])$	$P(ANL_{t+1} >= s)$
$a^1$ =MST Topology (MST)	T	0.06	0.16	0.78
$a^1$ _MST Topology (MST)	F	0.12	0.2	0.68
$a^2$ _Redundant Topology (RT)	T	0.05	0.15	0.8
$a^2$ _Redundant Topology (RT)	F	0.1	0.18	0.72

TABLE 4.9: CPT TTW:  $P(TTW|MP',a)$ 

OBS <sup>3</sup> : Total Time for Writing (TTW)				
Action $A_{t-1}$	$MP_t$	$P(TTW_{t+1} < f)$	$P(TTW_{t+1} \text{ in } [f, g])$	$P(TTW_{t+1} \geq g)$
$a^1$ =MST Topology (MST)	T	0.83	0.13	0.04
$a^1$ _MST Topology (MST)	F	0.67	0.23	0.1
$a^2$ _Redundant Topology (RT)	T	0.8	0.15	0.05
$a^2$ _Redundant Topology (RT)	F	0.63	0.25	0.12

The observation function  $O(s', a, z) = P(z|s', a)$ , i.e. the probabilities in Tables 4.7, 4.8, and 4.9, are an input in equation (2.1), to update the current belief about the satisficement levels of the NFRs in a SAS.

In this section, we have specified the required mapping rules to allow the specification and trade-off of NFRs in terms of a runtime POMDP model. Next, the details of the decision-making process of RE-STORM are presented.

#### 4.4 MAPE-K loop and POMDPs for decision-making in self-adaptation

The runtime behaviour of RE-STORM is based on a POMDP requirements-aware model within a feedback control loop (See Fig. 4.3 with the runtime architecture of this proposal). In this section, the different activities of the MAPE-K loop [51] and the details of the decision-making driven by the satisficement levels of the NFRs in the RDM SAS, are presented as follows.

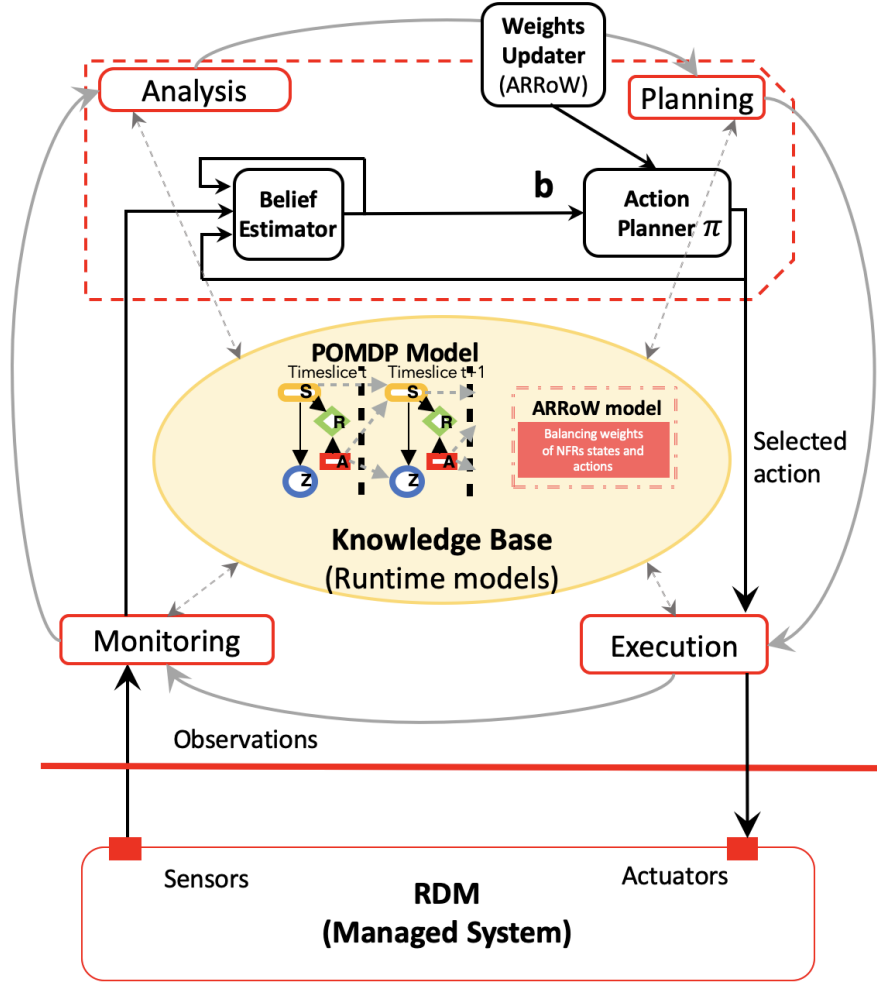


FIGURE 4.3: RE-STORM architecture: runtime models and managed system

#### 4.4.1 RE-STORM: MAPE-K loop activities

- a) *Monitoring*. In this activity the necessary data from the context is collected. In the RDM SAS the MON variables *Ranges of Bandwidth Consumption* (RBC), *Active Network Links* (ANL) and *Total Time for Writing* (TTW) are monitored during the system execution. The observed values for each MON variable constitute the evidence to compute later in Equation (2.1) the current belief about the satisficement level of the NFRs in the RDM system.
- b) *Analysis*. In requirements-aware systems, this activity uses the data about the context to update the requirements model or recompute the requirements satisfaction [39]. Any required data transformation to enable data to be used at the next stage should be performed at this step. Therefore, the component labeled



Belief Estimator in Fig. 4.3 is responsible for updating the belief about the satisficement level of the NFRs based on the previous belief  $b$ , the last action, and the last observation. This update is performed by using Equation (2.1) and the result is a new belief  $b$ , which represents the most probable satisficement level of the NFRs in a system given past experiences. The new belief will be the input for the planning activity. It is also recorded in the Knowledge Base as part of the POMDP runtime model.

- c) *Planning*. The component labeled Action Planner in Fig. 4.3 is the policy  $\pi$  responsible for generating actions, as a function of the current belief  $b$  about the satisficement levels of the NFRs in the system. We use online POMDP planning [75] to choose the best action. Online POMDP planning is a technique that interleaves planning with plan execution: at each time slice, the system searches for an optimal action  $a \in A$  at the current belief  $b$ . It then executes the chosen action immediately [96]. Details on this activity are presented in section 4.4.2.
- d) *Execution*. Once an action has been selected it is executed by the system. As a result, the system reaches a new state  $s'$  with probability  $T(s, a, s') = P(s' | s, a)$  and receives an observation  $z \in Z$  with probability  $O(s', a, z) = P(z | s', a)$ . It also receives a real number, i.e. a reward value  $R(s, a)$ , which represents the reward or penalty for arriving to the new state  $s'$ . Then, the MAPE-K loop starts again.
- e) *Knowledge Base*. Inside the Knowledge Base of an autonomic system are one or more models that support the Monitoring, Analysis, Planning and Execution activities [16]. In our proposal, two runtime models (See Fig. 4.3), are kept in the Knowledge Base: (i) a POMDP runtime model, which contains the current beliefs about the satisficement level of the NFRs, along with the reward values  $R(s, a)$  and (ii) the ARRoW model [38], a weights updater used to update reward values  $R(s, a)$  in a POMDP when requirements about the satisficement levels of the NFRs, i.e. Service Level Agreements (SLAs), are violated during the system's execution. Next, the planning activity for decision-making driven by NFRs is explained in detail.

#### 4.4.2 RE-STORM: Details on the online planning activity using POMDPs

During planning in RE-STORM, we depart from the belief  $b_0$  about the current state of the system. Further, and based on the “memoryless property” of a stochastic Markov process [53, 75], we have that the future state of the system depends only upon its current state, not on the past. Our proposal uses this assumption to project future evolutions of the satisficement levels of the NFRs in a SAS from the current belief  $b_0$ . Specifically, we use the Determinized Sparse Partially Observable Tree (DESPOT) algorithm [2, 96] as the planner of our proposal.

**Proactive self-adaptation.** One desirable capability of autonomic self-adaptive systems is anticipation, which is defined as being able to anticipate to some extent, needs, behaviours to be able to manage itself in a proactive way [64]. Proactive self-adaptation implies predictions of how the environment is going to evolve in the relatively near future. The approach makes decisions under the uncertainty implied by those predictions. RE-STORM provides a decision-making under uncertainty that is proactive and self-adaptive. Its implementation is based on the DESPOT tree in order to select the adaptation decisions. Next, relevant details of the approach are presented. The next two steps are related to the capabilities of proactive self-adaptation offered by RE-STORM during planning.

- a) **Build a DESPOT tree to project future evolutions of NFRs.** The action planner module of our approach, considers future evolutions of the satisficement level of the NFRs to decide the next action  $a \in A$ , i.e. to reason about long-term effects of immediate actions [96]. An example of this behaviour is presented in Chapter 6. The future evolutions of the state of the system are represented by the following DESPOT tree.

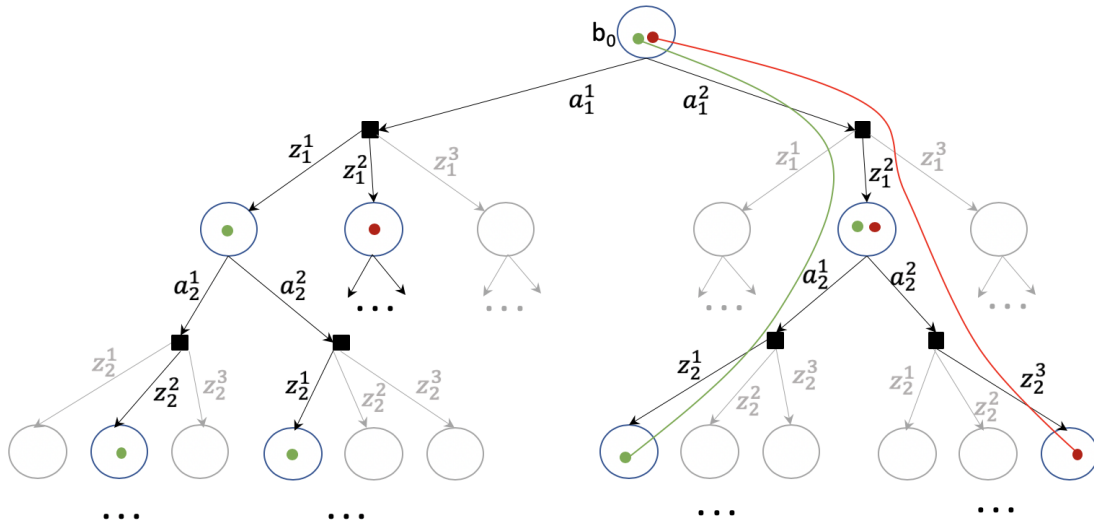


FIGURE 4.4: DESPOT Belief Tree with 2 sampled scenarios marked with green and red dots (The DESPOT tree is overlaid on a standard belief tree)

DESPOT builds per each time slice, a sparse approximation of a standard belief tree: a DESPOT tree (See Fig. 4.4), by using a simulation model [4]. The root node of the tree is the belief  $b_0$  which represents the belief about the current state of the running system. Each edge in the tree represents an action observation pair. If a child node  $b_t$  is connected to its parent  $b_{t-1}$  by an edge  $(a_t, z_t)$ , then  $b_t = \tau(b_{t-1}, a_t, z_t)$  according to Equation (2.1). In RE-STORM, each node in a DESPOT tree represents a belief about the level of satisficement of the NFRs. The DESPOT tree represents the neighborhood of the current belief  $b_0$ .

- b) **Select an optimal action**  $a \in A$ . The Bellman's principle of optimality [76] is shown in equation (4.8). It is applied over a DESPOT tree to choose the best action:

$$V(b) = \max_{a \in A} \left\{ \sum_{s \in S} b(s) R(s, a) + \gamma \sum_{z \in Z} p(z|b, a) V^*(r(b, a, z)) \right\} \quad (4.8)$$

The algorithm searches the tree with root at the current belief  $b_0$ . Specifically, the action planner module (See Fig. 4.3) uses lookahead search [84] to approximate the optimal discounted reward value  $V^*(b_0)$  [83, 55, 2]. The search is guided by a

lower bound  $l(b_0)$  and an upper bound  $\mu(b_0)$  on the approximated optimal discounted reward value  $V^*(b_0)$ . The explorations continue, until the gap between the bounds  $\mu(b_0)$  and  $l(b_0)$  reaches a target level  $\epsilon_0$  or the allocated planning time  $T_{max}$  runs out. Equation (4.8) recursively computes over the tree the maximum value of action branches and the average value of observation branches [55, 75, 4, 2]. The result is an approximately optimal policy for the current belief  $b_0$  [2, 96].

*A policy is learned by interacting with the environment as it was stated in Chapter 2, section 2.7. (In this case the environment is represented by the DESPOT tree).*

The system then executes the first action of the policy  $\pi(b_0)$ , i.e. the action with the highest discounted reward value  $V^*(b_0)$ .

**Definition 2.** The belief  $b_0$  represents the current satisficement levels of the system's NFRs. The selection of the optimal action  $a \in A$  made in equation (4.8), is based on the current and projected beliefs about the satisficement levels of the NFRs in a SAS.

The algorithm 1, which is shown below, provides a high-level view of the process to build and search a DESPOT tree. Two additional algorithms (Algorithms 2 and 3), complement this process by implementing specific behaviours for exploration and backup of belief nodes  $b$ , to determine an optimal action, which is derived from the DESPOT tree.

In more detail, the algorithm 1 constructs and searches a DESPOT tree incrementally. Initially, it contains only a root node with belief  $b_0$  about the current satisficement level of the NFRs in a system. The tree also contains the initial upper and lower bounds associated to the belief  $b_0$  (lines 4–5). The algorithm performs explorations using algorithm 2, to expand the DESPOT tree and to reduce the gap  $\epsilon(b_0)$  between the bounds  $\mu(b_0)$  and  $l(b_0)$  at the root node  $b_0$ . Each exploration aims at choosing and expanding a promising leaf node (line 8) and adds its child nodes into the tree until the current leaf node is not heuristically promising [96]. Then, the

---

**Algorithm 1** Algorithm for building and searching a DESPOT tree D in each time slice

---

```

1: Parameter(s):
2: - Initial belief  $b_0$  about the current satisfaction level of the NFRs
3: Runtime execution:
4: -Use the belief  $b_0$  to create the root node of a new DESPOT tree D
5: -Initialize upper and lower bounds:  $\mu(b_0)$  and  $l(b_0)$ 
6: -Initialize  $\epsilon(b_0) \leftarrow \mu(b_0) - l(b_0)$ 
7: while  $\epsilon(b_0) > \epsilon_0$  and running time is less than  $T_{max}$  do
8:    $b \leftarrow \text{Explore}(D, b_0)$  ▷ (Explore a promising path)
9:    $\text{Backup}(D, b)$  ▷ (Backup on bounds at each node b)
10: end while
11: Return: DESPOT tree with an approximated optimal policy  $\pi^*(b_0)$ 

```

---

algorithm traces the path back to the root and performs backup using Algorithm 3 on the upper and lower bounds at each node along the way to the root node (line 9). The explorations continue until the gap between the bounds  $\mu(b_0)$  and  $l(b_0)$  reaches a target level  $\epsilon_0$  ( $\epsilon_0 \geq 0$ ) or the planning time  $T_{max}$  finishes (line 7).

---

**Algorithm 2** Algorithm to expand the branches of a DESPOT tree D

---

```

1: Parameter(s):
2: -A DESPOT tree D and the current belief b
3: Runtime execution:
4: - $D_h \leftarrow$  DESPOT tree height
5: - $\Delta(b) \leftarrow$  current height of the belief b
6: while  $\Delta(b) \leq D_h$  and  $E(b) > 0$  do
7:   if b is a leaf node in D then
8:     Expand b one level deeper. Insert each new child  $b'$  of b into D,
9:     and initialize  $\mu(b')$  and  $l(b')$ 
10:     $a^* \leftarrow \arg \max_{a \in A} \mu(b, a)$ 
11:     $z^* \leftarrow \arg \max_{z \in Z_{b, a^*}} E(\tau(b, a^*, z))$ 
12:     $b \leftarrow \tau(b, a^*, z^*)$ 
13:   end if
14: end while
15: Return: An expanded DESPOT tree

```

---

In Algorithm 2, the exploration to expand the DESPOT tree starts at the root node  $b_0$ . At each node  $b$  along the exploration path, the best action branch  $a^*$ , according

to the upper bound  $\mu(b)$ , is selected (line 10). Afterwards, the observation branch  $z$  that leads to a child node  $b' = \tau(b, a^*, z)$  maximizing the excess uncertainty  $E(b')$ , is selected (line 11). The excess uncertainty  $E(b')$  measures the difference between the current gap at  $b'$  and the expected gap at  $b'$  if the target gap  $\epsilon(b_0)$  at  $b_0$  is satisfied. The exploration strategy seeks to reduce the excess uncertainty in a greedy manner [96, 41]. The exploration at a node  $b$  is terminated under the following conditions (line 6). First,  $\Delta(b) \leq D_h$ , i.e. the maximum tree height is exceeded, and second,  $E(b) > 0$ , indicating that when the expected gap at  $b$  is reached, further exploration from  $b$  onwards may be unproductive. When the exploration terminates, Algorithm 3 is performed.

---

**Algorithm 3** Algorithm to perform backup on the bounds of each node  $b$  using Bellman's principle

---

- 1: Parameter(s):
  - 2: -A DESPOT tree  $D$  and the current belief  $b$
  - 3: Runtime execution:
  - 4: **for** each node  $x$  on the path from  $b$  to the root  $D$  **do**
  - 5:     Perform backup on  $\mu(x)$  and  $l(x)$
  - 6: **end for**
- 

In Algorithm 3, the path back to the root node is traced, and the backup is performed on the upper and lower bounds at each node  $b$  along the way, using the Bellman's principle of optimality [96]. Specifically, the bounds  $\mu(b)$  and  $l(b)$  at each node  $b$  are recomputed.

#### 4.4.3 RE-STORM: Approximated optimal policy

In RE-STORM, a DESPOT tree represents the possible future evolutions of a belief  $b$  about the current satisficement level of the NFRs in a SAS. An approximated optimal policy  $\pi$  is derived from a DESPOT tree, for each time slice during the system's execution. The approach to derive an approximated policy  $\pi$  of size  $|\pi|$ , is deeply related to the process of building a DESPOT tree. It is built through a set of *randomly sampled scenarios* [96] from the root node of the tree, i.e., the belief  $b_0$  (See Fig. 4.4 ) about the current satisficement level of the NFRs in a SAS. A scenario is an *abstract*

*simulation trajectory* with some start state  $s_0$ . A scenario for a belief  $b$ , is a random sequence  $\Phi = (s_0, \phi_1, \phi_2, \dots)$ , in which the start state  $s_0$  is sampled according to  $b$  and each  $\phi_i$  is a real number sampled uniformly and independently from the range  $[0,1]$ . A DESPOT tree is defined constructively by applying a simulation model [96, 4] to all possible action sequences under  $K$  sampled scenarios. Fig. 4.4 shows how a node  $b$  in a DESPOT tree branches into  $|A|$  action edges, and each action edge further branches into a subset of  $|Z|$  observation edges.

While a standard belief tree of height  $D_h$  has  $|A|^D |Z|^D$  nodes, a corresponding DESPOT tree has  $|A|^D K$  nodes for  $|A| \geq 2$ , because of reduced observation branching under the sampled scenarios, as observed in Fig. 4.4.

An approximated optimal policy  $\pi$  is derived by choosing  $K$  to be equal or greater than  $\lceil \pi \ln(|\pi||A||Z|) \rceil$  [96, 4]. Since a DESPOT tree has size  $|A|^D K$ , the choice of  $K$  allows to trade off computation cost and approximation accuracy. Specifically, the *default values* provided by the DESPOT algorithm for domain independent POMDP problems have been used in this implementation, i.e.  $K = 500$ , DESPOT tree height  $D_h = 90$ , and max planning time  $T_{max} = 1$  second for each time slice.

## 4.5 Summary

In this chapter, RE-STORM, a proposal for decision-making driven by the current satisficement levels of the NFRs in a SAS has been presented. Within the context of the MAPE-K loop, the analysis activity uses Bayesian inference to compute the current belief  $b$  about the satisficement levels of the NFRs. During the planning activity, the Bellman's principle of optimality formalised in Equation (4.8) is applied to compute a policy  $\pi$  that yields an optimal action  $a = \pi(b)$ .

At design-time, initial preferences about NFRs and adaptation actions are specified and correspond to the reward values  $R(s,a)$  in a POMDP (Based on mapping Rule 2). These preferences, which are used in Equation (4.8), under some unforeseen situations that arise at runtime, may be not suitable and affect the Service Level Agreements (SLAs) of the system. Specifically, the thresholds of the levels of satisficement of the NFRs in a SAS may be violated. To overcome this situation, ARRoW,

a proposal to reassess and update initial preferences during the system's execution is presented next, in Chapter 5.



## Chapter 5

# ARRoW: Automatic Runtime Reappraisal of Weights for Self-Adaptation

### 5.1 Overview

This chapter presents ARRoW (Automatic Runtime Reappraisal of Weights), a technique to support the dynamic update of preferences (a.k.a. weights) associated with the NFRs and adaptation actions in SASs. ARRoW takes into account the current levels of satisficement that NFRs can reach during the system's operation in order to update the initial preferences when needed. To develop ARRoW, the Primitive Cognitive Network Process (P-CNP) has been extended to enable the handling and update of weights during runtime. P-CNP is a version of the Analytical Hierarchy Process (AHP). Specifically, the approach involves the following:

- The specification of NFRs and design decisions to deduce the corresponding weights as a P-CNP problem.
- The use of P-CNP at runtime.

The extension made to P-CNP and an example to illustrate the application of ARRoW is presented next.

## 5.2 A motivating example

As an example to facilitate the explanation of the ARRoW process, a reduced version of the Remote Data Mirroring (RDM) SAS presented in Chapter 4 has been considered (See Fig. 5.1). In this version, two NFRs: *Minimization of Cost* (MC) and *Maximization of Reliability* (MR) are taken into account.

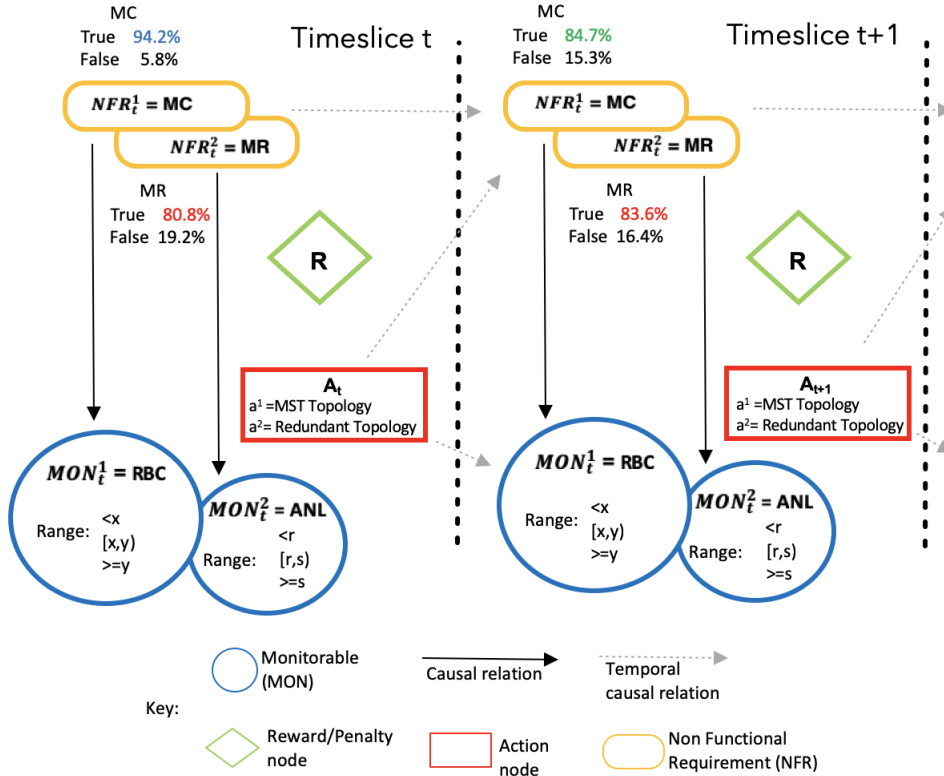


FIGURE 5.1: RDM example - POMDP representation

As explained in Chapter 4, in SASs, the decision-making process requires the runtime quantification and trade-off of multiple non-functional requirements (NFRs) and the cost-benefit analysis for alternative solution strategies [58]. Preferences associated with NFRs and adaptation actions, are used during the decision-making process in self-adaptation [65].

*In our proposal RE-STORM, these preferences coincide with the reward values  $R(s,a)$  in a POMDP.*

Table 5.1 shows the reward values  $R(s,a)$  (i.e. preferences) for the RDM example.

TABLE 5.1: RDM example - Reward values  $R(s,a)$

Reward $R(s,a)$	Action (A)	State (S)	State (S) as NFRs		Alternatives alt $\in$ Alt
			NFR <sup>1</sup> =MC	NFR <sup>2</sup> =MR	
$r_1=0.3140$	$a^1=MST$	$s_1$	T	T	alt <sub>1</sub>
$r_2=0.1487$	$a^1=MST$	$s_2$	T	F	alt <sub>2</sub>
$r_3=0.0826$	$a^1=MST$	$s_3$	F	T	alt <sub>3</sub>
$r_4=0.0165$	$a^1=MST$	$s_4$	F	F	alt <sub>4</sub>
$r_5=0.3388$	$a^2=RT$	$s_1$	T	T	alt <sub>5</sub>
$r_6=0.0330$	$a^2=RT$	$s_2$	T	F	alt <sub>6</sub>
$r_7=0.0661$	$a^2=RT$	$s_3$	F	T	alt <sub>7</sub>
$r_8=0$	$a^2=RT$	$s_4$	F	F	alt <sub>8</sub>

In self-adaptation, the initial preferences specified at design-time may be not suitable for some specific situations found at runtime. We tackle this challenge using ARRoW, the technique presented in this Chapter.

When using ARRoW, new preferences, i.e. new reward values  $R(s,a)$ , are calculated. They are used as an input in Equation (4.8) for the decision-making process based on RE-STORM during execution. Next, we explain how the P-CNP framework has been extended to enable dynamic reappraisal and update of weights in SASs by using ARRoW.

### 5.3 P-CNP for reassessing NFR weights in Self-adaptation

This section overviews the P-CNP framework to derive the weights among several competing alternatives by using a comparison scale and paired differential comparisons [102]. We briefly explain how these concepts have been associated with the concepts of runtime decision-making by a SAS, underpinned by POMDPs, to dynamically reappraise weights based on the results shown in [38, 67, 66]. Under this context, the use of P-CNP involves the following steps:

#### 5.3.1 Step 1: Problem definition

In this step, a decision problem is formulated as a Structural Assessment Network (SAN) [101]. A SAN is a model with a goal (i.e. a functional requirement in our case), a criteria structure (i.e. the system's NFRs) and a set of alternatives to achieve

the goal based on the identified criteria. Fig. 5.2 shows an example of a SAN model for the case of the RDM SAS.

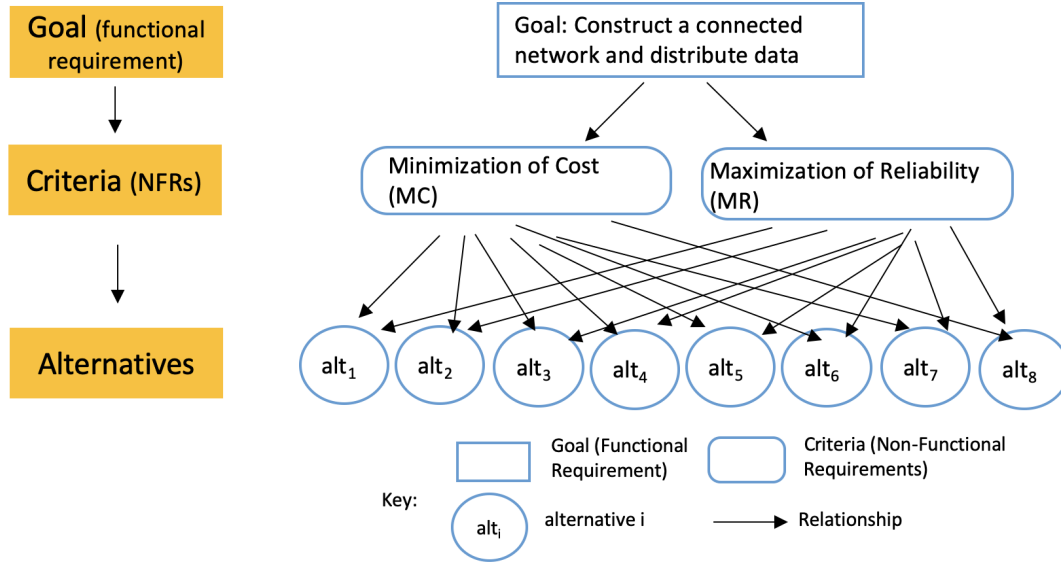


FIGURE 5.2: RDM Example- Structural Assessment Network (SAN)

In our example, the goal is to construct a connected network to distribute data in the RDM. Decisions taken by the system under the current runtime context, have an impact on the satisficement levels of the NFRs *Minimization of Cost* (MC) and *Maximization of Reliability* (MR). These NFRs correspond to the criteria in the SAN model. To meet its goal, the RDM SAS has a set of alternatives  $alt \in Alt$ , represented by the possible NFRs' states to be reached after taking a self-adaptation action MST or RT topology.

*In ARRoW, an alternative  $alt \in Alt$  represents a set of possible NFRs' states to be reached after taking a self-adaptation action.*

For example, the alternative  $alt_1$  in Table 5.1, represents that after performing action  $a^1 = MST$ , the reached NFRs' states are  $MC = True$  and  $MR = True$ . Each alternative  $alt \in Alt$  should have a preference or numeric weight associated, i.e. a reward value  $R(s,a)$  in a POMDP as shown in Table 5.1. Once formulated a SAN problem as the shown in Fig. 5.2, the next step is to assign weights to the criteria identified in the model as explained next.

### 5.3.2 Step 2: Weights assignment to NFRs

The comparison and prioritization of the NFRs, is performed by using (i) the comparison scale shown in Table 5.2 and (ii) pairwise differential comparisons [102]. Different from the original use of P-CNP which is performed manually, in this work the comparisons and prioritizations are made autonomously by the SAS at runtime. Two basic concepts to perform this step are presented next.

TABLE 5.2: P-CNP scale of comparison

Numerical Form ( $v_{ij}$ )	Verbal Form
0	Object $v_i$ and Object $v_j$ are <b>equal</b>
1	Object $v_i$ is <b>slightly</b> over Object $v_j$
2	Object $v_i$ is <b>moderately</b> over Object $v_j$
3	Object $v_i$ is <b>fairly</b> over Object $v_j$
4	Object $v_i$ is <b>highly</b> over Object $v_j$
5	Object $v_i$ is <b>strongly</b> over Object $v_j$
6	Object $v_i$ is <b>significantly</b> over Object $v_j$
7	Object $v_i$ is <b>outstandingly</b> over Object $v_j$
8	Object $v_i$ is <b>absolutely</b> over Object $v_j$
-1	Object $v_j$ is <b>slightly</b> over Object $v_i$
...	...
-8	Object $v_j$ is <b>absolutely</b> over Object $v_i$

- a) **Pairwise differential comparisons.** In P-CNP, the comparison of two objects is represented by the difference between them. To make pairwise differential comparisons, a single number is drawn from the scale shown in Table 5.2 to represent dominance in view of the semantic form  $v_{ij} = v_i - v_j$ .

The knowledge representation of pairwise comparisons is divided as two parts: syntactic form and semantic form. The syntactic form (a.k.a. verbal form) is the comparison sentence using linguistic words for comparison. The semantic form (a.k.a. numerical form) is the mathematical description or numerical representation of the syntactic form.

For example, with respect to a comparison, it can be said the following syntactic form: alternative A is “*moderately*” more preferable than alternative B. Note in Table 5.2 the different syntactic forms that can be used. The equivalent semantic form for this example is:  $v_{ij} = \text{value judgement of A} - \text{value judgement of B} = 2$ .

To compare at runtime the satisficement levels of the NFRs, the semantic form is

presented as follows:  $nfr_{ij} = nfr_i - nfr_j$ .

- b) **Pairwise opposite matrix (POM)**. A POM is a quadratic matrix that represents the relative importance between each pair of compared elements [99]. Fig. 5.3 shows a generic POM, where  $nfr_{ij} = nfr_i - nfr_j$  represents the differential pairwise comparison between the values of the compared NFRs in the row “i” and the column “j” in the POM.

$$\begin{bmatrix} nfr_{ij} \end{bmatrix} = \begin{matrix} & \begin{matrix} nfr_1 & nfr_2 & \dots & nfr_n \end{matrix} \\ \begin{matrix} nfr_1 \\ nfr_2 \\ \dots \\ nfr_n \end{matrix} & \begin{bmatrix} 0 & nfr_1 - nfr_2 & \dots & nfr_1 - nfr_n \\ nfr_2 - nfr_1 & 0 & \dots & nfr_2 - nfr_n \\ \dots & \dots & \dots & \dots \\ nfr_n - nfr_1 & nfr_n - nfr_2 & \dots & 0 \end{bmatrix} \end{matrix} \quad \begin{matrix} \text{Weights of} \\ \text{NFRs} \\ - \\ - \\ \dots \\ - \\ \hline 1.0000 \end{matrix}$$

FIGURE 5.3: POM - general structure to compare NFRs

When  $i = j$ , then  $nfr_{ij} = 0$ . The values  $nfr_{ij}$  correspond to the values of a measurement scale schema as is shown in Table 5.2. Fig. 5.4 shows an example of a POM after the comparison between two NFRs: *Minimization of Cost* (MC) and *Maximization of Reliability* (MR) of the RDM SAS.

$$\text{POM}_{\text{NFRs}} = \begin{bmatrix} \text{NFR}_{ij} \end{bmatrix} = \begin{matrix} & \begin{matrix} \text{MC} & \text{MR} \end{matrix} \\ \begin{matrix} \text{MC} \\ \text{MR} \end{matrix} & \begin{bmatrix} 0 & 3 \\ -3 & 0 \end{bmatrix} \end{matrix} \quad \begin{matrix} \text{Weights of} \\ \text{NFRs} \\ 0.5938 \\ 0.4062 \\ \hline 1.0000 \end{matrix}$$

FIGURE 5.4: POM - example of NFRs comparison

The value 3 in the POM above, represents that MC is “fairly over” (See Verbal Form in Table 5.2) than MR. The Row Average plus the Normal Utility (RAU) prioritization method [99] is used to derive weights from a POM:

$$w_i = \left( \frac{1}{p} \sum_{j=1}^p v_{ij} \right) + \kappa \quad (5.1)$$

In Equation (5.1),  $p$  represents the number of compared elements. Kappa ( $\kappa$ ) represents the perception by people of the difference values of paired objects in different scenarios.  $\kappa$  is greater than 0 and, by default, is the highest value of the comparison scale [99]. The results presented in this dissertation were obtained by using the default ( $\kappa$ ).

### 5.3.3 Step 3: Comparison of alternatives $\text{alt} \in \text{Alt}$ with respect to NFRs

This comparison is performed by using paired differential comparisons between the alternatives  $\text{alt} \in \text{Alt}$ , with respect to each NFR identified as part of the criteria in Step 1. Fig. 5.5 shows the comparison between alternatives with respect to the NFR *Minimization of Cost* (MC).

		alt <sub>1</sub>	alt <sub>2</sub>	alt <sub>3</sub>	alt <sub>4</sub>	alt <sub>5</sub>	alt <sub>6</sub>	alt <sub>7</sub>	alt <sub>8</sub>	Weights of alternatives
POM <sub>Alt-MC</sub> =	alt <sub>1</sub>	0	0	3	3	1	1	5	5	0.1602
	alt <sub>2</sub>	0	0	3	3	1	1	5	5	0.1602
	alt <sub>3</sub>	-3	-3	0	0	-2	-2	2	2	0.1133
	alt <sub>4</sub>	-3	-3	0	0	-3	-3	2	2	0.1094
	alt <sub>5</sub>	-1	-1	2	3	0	0	3	3	0.1426
	alt <sub>6</sub>	-1	-1	2	3	0	0	3	3	0.1426
	alt <sub>7</sub>	-5	-5	-2	-2	-3	-3	0	0	0.0859
	alt <sub>8</sub>	-5	-5	-2	-2	-3	-3	0	0	0.0859

FIGURE 5.5: Relatives weights of  $\text{alt} \in \text{Alt}$  in relation to MC

In the example above, we compare the alternatives  $\text{alt} \in \text{Alt}$  previously shown in Table 5.1. For instance, we observe that the comparison between  $\text{alt}_1$  and  $\text{alt}_7$  is equal to 5. The value 5 (See the scale of comparison in Table 5.2) indicates that  $\text{alt}_1$  is “strongly over” (i.e. more preferable) than  $\text{alt}_7$ . The interpretation of this result means that from the point of view of *Minimization of Cost* (MC), it is more preferable to perform action  $a = \text{MST}$  to then reaches the NFRs’ states  $\text{MC} = \text{True}$  and  $\text{MR} = \text{True}$ , than perform action  $a = \text{RT}$  and reaches the NFRs’ states  $\text{MC} = \text{False}$  and  $\text{MR} = \text{True}$ .

As a result of this step we obtain a set of relative weights for the alternatives  $\text{alt} \in \text{Alt}$  in relation to each NFR. These relative weights are independent of the step 2 of P-CNP and only need to be calculated once at design-time. The obtained POMs are called  $\text{POM}_{\text{Alt}-\{\text{NFR}\}}$ , where  $\{\text{NFR}\}$  is the specific NFR in relation to which

the alternatives  $alt \in Alt$  were compared. In the RDM example, two POMs were obtained:  $POM_{Alt-MC}$  and  $POM_{Alt-MR}$ .

### 5.3.4 Step 4: Information fusion and global weights

The global weights for each alternative  $alt \in Alt$  are computed by using the weighted arithmetic mean [99]:

$$alt_i = \sum_{j=1}^n w_j * p_{ij}, i = 1, \dots, m \quad (5.2)$$

$w_j$  is a NFR weight (obtained at step 2),  $p_{ij}$  is a relative weight of the alternative  $alt_i$  with respect to  $nfr_j$  (obtained in step 3), and  $m$  is the number of alternatives  $alt \in Alt$ . As a result, a final matrix, called Decision Matrix Alt-Fusion ( $DM_{Alt-Fusion}$ ) is derived (See Fig. 5.6). The final weights of alternatives  $alt \in Alt$  represented by a  $DM_{Alt-Fusion}$  are the updated reward values  $R(s,a)$  in the underlying POMDP, which will be used by RE-STORM in equation (4.8) during the decision-making of the running SAS.

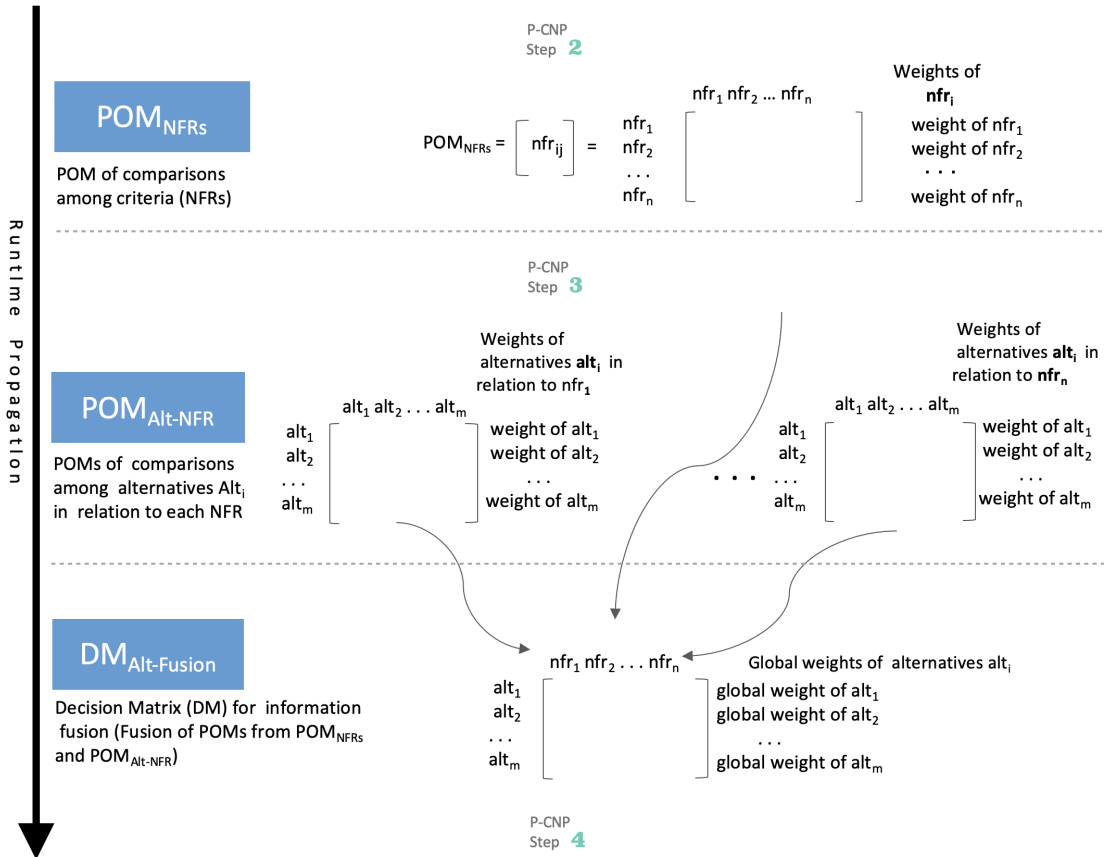


FIGURE 5.6: P-CNP abstractions for weights propagation at runtime



## 5.4 Automatic Runtime Reappraisal of Weights: ARRoW

ARRoW is a technique created to be performed during the execution of a SAS. However, the fulfillment of some pre-conditions is required before its operation. In this section, the initial setup of ARRoW, which is performed at design-time, is presented. Afterwards, the details of its runtime process is explained through the RDM example. Fig. 5.7, shows a summary of the ARRoW processes in both contexts: design-time and runtime.

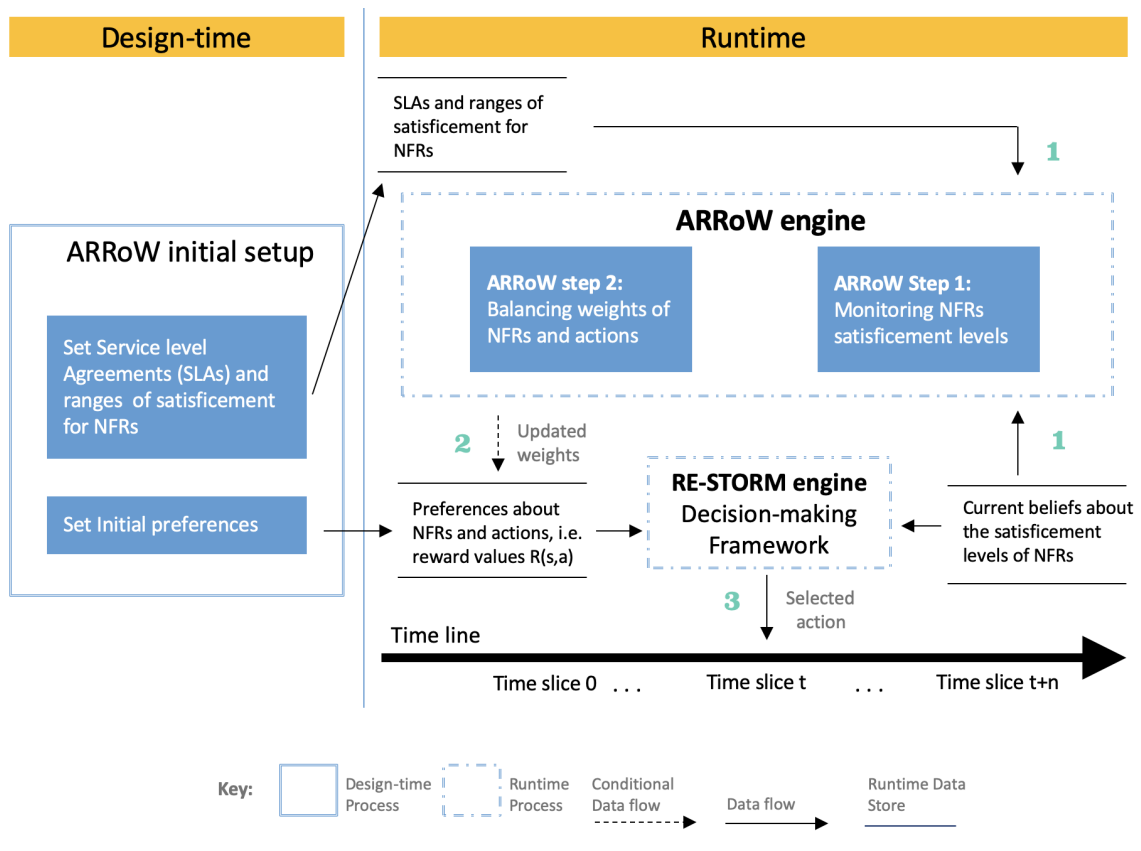


FIGURE 5.7: ARRoW process: design-time and runtime behaviour

### 5.4.1 ARRoW Initial Setup

The initial setting of ARRoW before runtime assumes:

- Service Level Agreements (SLAs) for the satisficement levels of the NFRs
- An underlying decision-making framework

Next, we briefly describe the content and purpose of these elements.

- a) **Service level agreements (SLAs) for NFRs.** Different approaches are used to determine the Service Level Agreements (SLAs) in a system, for example they can be learned or provided by experts. In the case of the RDM SAS, the SLAs for the NFRs *Minimization of Cost* (MC) and *Maximization of Reliability* (MR) were defined on the basis of information provided by system's experts. The SLAs represent the satisficement threshold for each NFR. Any value below the threshold of a NFR is in a zone of *poor satisficement*. In contrast, any value equal or greater than the threshold is seen in a zone of *suitable satisficement*. The identified SLAs are:  $P(MC=True \geq 0.7)$  and  $P(MR=True \geq 0.9)$ . The specification of the SLAs within the zones of satisficement (poor and suitable) is highlighted in Table 5.3.

TABLE 5.3: RDM example - SLAs

Non Functional Requirements (NFRs)	Zone of poor satisficement (under the threshold)	Zone of suitable satisficement (over the threshold)
Minimization of Cost (MC)	[0.00 , 0.70] ←---	[0.70 , 1.00]
Maximization of Reliability (MR)	[0.00 , 0.90] ←---	[0.90 , 1.00]

Note in Fig. 5.7, how the SLAs specified at design-time, represent an input of the ARRoW process at runtime.

*Ranges of levels of satisficement of NFRs.* In order to perform the comparisons among the satisficement levels of the NFRs at runtime (following the Step 2 of P-CNP described in section 5.3.2), the zones of *suitable satisficement* and *poor satisficement* for each NFR are divided in ranges. In ARRoW, each numeric factor of the P-CNP scale of comparison shown in Table 5.2, is associated with a range of satisficement.

In the case of the RDM example, the ranges of satisficement for the suitable zone of the NFRs MC and MR are shown in Table 5.4. Table 5.5 shows the equivalent ranges for the poor zone of the NFRs MC and MR.

TABLE 5.4: RDM example - Ranges for suitable satisficement

Factors of the P-CNP scale of comparison	Zone of suitable satisficement	
	Minimization of Cost (MC)	Mazimization of Reliability (MR)
1	[ <b>0.7000</b> , 0.7375)	[ <b>0.9000</b> , 0.9125)
2	[0.7375 , 0.7750)	[0.9125 , 0.9250)
3	[0.7750 , 0.8125)	[0.9250 , 0.9375)
4	[0.8125 , 0.8500)	[0.9375 , 0.9500)
5	[0.8500 , 0.8875)	[0.9500 , 0.9625)
6	[0.8875 , 0.9250)	[0.9625 , 0.9750)
7	[0.9250 , 0.9625)	[0.9750 , 0.9875)
8	[0.9625 , <b>1.0000</b> ]	[0.9875 , <b>1.0000</b> ]

TABLE 5.5: RDM example - Ranges for poor satisficement

Factors of the P-CNP scale of comparison	Zone of poor satisficement	
	Minimization of Cost (MC)	Mazimization of Reliability (MR)
1	[ <b>0.0000</b> , 0.0875)	[ <b>0.0000</b> , 0.1125)
2	[0.0875 , 0.1750)	[0.1125 , 0.2250)
3	[0.1750 , 0.2625)	[0.2250 , 0.3375)
4	[0.2625 , 0.3500)	[0.3375 , 0.4500)
5	[0.3500 , 0.4375)	[0.4500 , 0.5625)
6	[0.4375 , 0.5250)	[0.5625 , 0.6750)
7	[0.5250 , 0.6125)	[0.6750 , 0.7875)
8	[0.6125 , <b>0.7000</b> ]	[0.7875 , <b>0.9000</b> ]

The ranges above together the SLAs specified in this section, are an input of the ARRoW process at runtime as is depicted in Fig. 5.7. They enable ARRoW to perform runtime pairwise differential comparisons between the satisficement levels of the NFRs to therefore assign them weights, i.e. a specific importance. The computed weights represent the result of Step 2 in P-CNP.

Details on the use of the ranges in Tables 5.4 and 5.5 are presented in section 5.4.2, the ARRoW runtime process.

b) **An underlying runtime decision-making framework.** ARRoW relies on RE-STORM, as its underlying runtime decision-making framework. RE-STORM involves a requirements-aware runtime model based on Partially Observable Markov Decision Processes (POMDPs) which offers ARRoW access to:

- The current belief about the satisficement levels of the NFRs in a SAS

- The current reward values  $R(s,a)$  which in this work represent the current preferences (a.k.a. weights) over the NFRs and adaptation actions in a system.

*Initial reward values  $R(s,a)$ .* In the decision-making process leveraged by RE-STORM, the system gets a reward value  $R(s,a)$  after performing an action “a” and achieving a new state “s”, i.e. a new satisficement level for its NFRs. For the RDM case study, similar to the SLAs, *initial* reward values  $R(s,a)$  were also defined on the basis of information provided by system’s experts. Note in Fig. 5.7, how these preferences are an input for the ARRoW process at runtime.

The reward values  $R(s,a)$  are used in Equation (4.8) for the selection of self-adaptation actions during the online planning activity of RE-STORM. They may need to be updated at runtime, when it is detected that the current values are not suitable anymore for a better decision-making according to the current system’s context. Details on this process are presented in the next section.

#### 5.4.2 The ARRoW runtime process

ARRoW is mainly applied during the execution of the system as shown in Fig. 5.7. The time is handled based on time slices. In Algorithm 4, a high level view of the algorithm that summarizes the complete process to update weights, i.e., rewards values  $R(s,a)$  in a POMDP, is shown. The steps of the ARRoW process at runtime are summarized as follows:

##### Step 1: Monitoring NFR satisficement to detect poor levels

This step is performed at runtime for each time slice. If at least one NFR has its satisficement level, i.e. probability  $P(\text{NFR}=\text{True})$ , labeled below its specified threshold; then the following step will hold. Otherwise, the SAS continues using its current weights.

---

**Algorithm 4** Automatic Runtime Reappraisal of Weights (ARRoW) per each time slice

---

```

1: Pre conditions:
2: -Set thresholds and ranges of satisficement for each NFR.
3: -Set initial reward values  $R(s,a)$ .
4: Runtime execution:
5: while the system is running do
6:   Per each time slice, monitor NFRs satisficement (Step 1)
7:   if at least one NFR is under its threshold then
8:     -Compare satisficement levels of NFRs and
9:     assign new weights to NFRs  $\rightarrow$   $POM_{NFRs}$  updated (Step 2.1)
10:    -Using  $POM_{NFRs}$  and  $POM_{Alt-NFRs}$  derive new reward values  $R(s,a) \rightarrow$ 
11:     $DM_{Alt-Fusion}$  updated (Step 2.2)
12:   end if
13:   Perform POMDP decision-making using possibly updated reward values
14:    $R(s,a)$ 
15: end while

```

---

**Step 2: Balancing weights of NFRs and actions**

Step 2.1 - Compare and assign new weights to NFRs. First, pairwise differential comparisons among the satisficement levels of the NFRs are performed to update their weights at runtime. The step departures from an empty POM  $POM_{NFRs}$  as the shown in Fig. 5.3. The needed comparisons are performed to fill up the matrix and derive updated weights for each NFR (Algorithm 4, step 2.1).

At runtime and per each pairwise comparison of the NFRs  $nfr_i$  and  $nfr_j$ , where “i” is a row and “j” is a column in the POM, the ARRoW engine faces different situations which are presented as cases as follows:

- Case 1: probability  $P(nfr_i=True) < nfr_{i\_threshold}$  and probability  $P(nfr_j=True) \geq nfr_{j\_threshold}$ . In this case,  $nfr_i$  has its satisficement level under the threshold. On the other hand, the satisficement level of  $nfr_j$  is greater or equal to the threshold. Under this context the ARRoW engine performs the following rule:

*More importance is assigned to  $nfr_i$ . The P-CNP numeric factor associated to the current range of satisficement of  $nfr_j$  is selected as result.*

- Case 2: probability  $P(nfr_i=True) \geq nfr_{i\_threshold}$  and probability  $P(nfr_j=True) < nfr_{j\_threshold}$ . This is the opposite of Case 1. Therefore:

*More importance is assigned to  $nfr_j$ . The P-CNP numeric factor associated to the current range of satisficement of  $nfr_i$  is multiplied by -1 and selected as result.*

- Case 3: probability  $P(nfr_i=True) \geq nfr_{i\_threshold}$  and probability  $P(nfr_j=True) \geq nfr_{j\_threshold}$ . In this case, both NFRs have their satisficements levels greater or equal to their thresholds. Therefore:

*More importance is assigned to the NFR with lower satisficement. The subtraction of the P-CNP numeric factors associated to the current ranges of satisficement for  $nfr_i$  and  $nfr_j$  is selected as result.*

- Case 4: probability  $P(nfr_i=True) < nfr_{i\_threshold}$  and probability  $P(nfr_j=True) < nfr_{j\_threshold}$ . In this case both NFRs have their satisficement levels under their thresholds. The ARRoW engine applies the same rule as in Case 3 but the P-CNP numeric factors for both NFRs are taken from their zone of poor satisficement (See Table 5.5).

For better understanding, examples of the cases mentioned above are shown as follows.

- Case 1 example: probability  $P(MC=True)=0.62$  and probability  $P(MR=True)=0.94$ . We observe that  $nfr_i$ , MC, has a satisficement level below its threshold (See Table 5.5). Conversely, MR (i.e.,  $nfr_j$ ) has a level of satisficement over its threshold, within the range [93.75%, 95.50%]. Therefore, a higher priority is given to MC by choosing the P-CNP numeric factor associated to the range [93.75% , 95.50%], i.e., the value 4. A POM representing this comparison is shown in Fig. 5.8.

		MC	MR	Weights of NFRs
$POM_{NFRs} = \begin{bmatrix} nfr_{ij} \end{bmatrix} =$	MC	0	4	0.6250
	MR	-4	0	0.3750
				<hr/> 1.0000

FIGURE 5.8:  $POM_{NFRs}$  Example Case 1

- Case 2 example: probability  $P(MC=True)=0.83$  and probability  $P(MR=True)=0.84$ . Whilst  $nfr_i$ , i.e. MC, has a satisficement levels over its threshold, within the range [81.25% , 85.00%] (See Table 5.4), the  $nfr_j$  MR has a satisficement level under its threshold. A higher importance is assigned to MR by choosing the P-CNP numeric factor associated to the range [81.25% , 85.00%], i.e., 4, and multiplying it by -1. A POM representing this comparison is shown in Fig. 5.9.

		MC	MR	Weights of NFRs
$POM_{NFRs} = \begin{bmatrix} nfr_{ij} \end{bmatrix} =$	MC	0	-4	0.3750
	MR	-4	0	0.6250
				<hr/> 1.0000

FIGURE 5.9:  $POM_{NFRs}$  Example Case 2

- Case 3 example: probability  $P(MC=True)=0.74$  and probability  $P(MR=True)=0.97$ . MC has a satisficement level within the range [73.75% , 75.50%] and MR within the range [96.25% , 97.50%]. Under this context, a higher importance is assigned to MC by subtracting their related P-CNP numeric factors (See Table 5.4), i.e.,  $6 - 2 = 4$ . This result also corresponds to the POM shown in Fig. 5.8. An equivalent approach is applied for Case 4.

Once the pairwise comparisons between the NFRs are completed, their new weights are derived using the Equation (5.1).

Step 2.2 - Derive new reward values  $R(s,a)$ . The ARRoW engine performs the steps 3 and 4 of P-CNP specified in section 5.3, to compute the updated reward values

$R(s,a)$  in a POMDP. Table 5.6 shows an example of updated weights for the RDM case study.

TABLE 5.6: RDM example - Updated reward values  $R(s,a)$

Reward $R(s,a)$	Action (A)	State (S)	State (S) as NFRs		Alternatives alt $\in$ Alt
			NFR <sup>1</sup> =MC	NFR <sup>2</sup> =MR	
$r_1=0.1757$	$a^1=MST$	$s_1$	T	T	alt <sub>1</sub>
$r_2=0.1679$	$a^1=MST$	$s_2$	T	F	alt <sub>2</sub>
$r_3=0.1093$	$a^1=MST$	$s_3$	F	T	alt <sub>3</sub>
$r_4=0.1002$	$a^1=MST$	$s_4$	F	F	alt <sub>4</sub>
$r_5=0.1523$	$a^2=RT$	$s_1$	T	T	alt <sub>5</sub>
$r_6=0.1484$	$a^2=RT$	$s_2$	T	F	alt <sub>6</sub>
$r_7=0.0768$	$a^2=RT$	$s_3$	F	T	alt <sub>7</sub>
$r_8=0.0690$	$a^2=RT$	$s_4$	F	F	alt <sub>8</sub>

The updated weights are further used in Equation (4.8) by RE-STORM to choose the optimal action  $a \in A$  in the given time slice.

## 5.5 Summary

We have presented ARRoW in this chapter, a technique to improve the decision-making in SASs by reappraising and updating unsuitable weights to leverage a better-informed decision-making.

A main contribution is the mapping from the specification of NFRs to a P-CNP problem to enable the runtime leverage of weights. The NFRs specification includes the identification of a threshold of satisficement for each NFR to identify unacceptable poor zones and suitable zones of satisficement that will support the identification of situations where the dynamic reassessment of weights is needed.

We have also shown how the P-CNP approach has been extended to be used at runtime by allowing the runtime propagation of elements of P-CNP POM matrices and the calculated weights.

ARRoW relies on an underlying Bayesian-based temporal framework, RE-STORM, presented in Chapter 4. Crucially, ARRoW leverages the Bayesian inference process underneath, which on the other hand, provides the mechanism to get access to evidence about the levels of satisficement of the NFRs to inform ARRoW.



Next, the evaluation and benefits of using RE-STORM and ARRoW, as opposed to ignoring the impact of new evidence over the preferences in a SAS, is presented in the following Chapter 6.

## Chapter 6

# Evaluation

This chapter presents a set of experiments to evaluate our proposals RE-STORM and ARRoW, using the RDM SAS case study explored earlier. The overall objective of the RDM is to distribute data among remote servers in an efficient manner by satisficing the NFRs of the system [73]. We have configured the RDM SAS with initial preferences about their NFRs and adaptation actions. The initial configurations of the RDM SAS are described in section 6.2.

In these experiments the validity of the initial preferences for new situations found at runtime is studied. Several dynamic contexts are taken into consideration. Such contexts were randomly created to affect the RDM SAS during its execution. Let us revisit the three (3) Research Questions of this dissertation, which were stated in Chapter 1:

- **RQ1:** How can we represent the current state of NFRs and their evolution in model-based self-adaptive systems that are subject to uncertain environments?
- **RQ2:** How can we improve the trade-off among NFRs in model-based self-adaptive systems that are subject to uncertain environments, by updating preferences and based on new evidence collected during execution ?
- **RQ3:** How can techniques for eliciting initial preferences about requirements, used at design-time, be applicable to runtime models for self-adaptive systems?

With the experiments shown, we aim to demonstrate the validity of the following hypothesis to therefore answer the three research questions of this dissertation:

**H:** *“Dynamic changes in the context, from the managed system observed at runtime, require the reassessment and update of current reward values  $R(s,a)$  of the POMDP runtime model to improve accordingly the trade-off levels of the satisficement of the NFRs for the new environmental conditions.”*

The first part of the hypothesis: *“Dynamic changes in the context, from the managed system observed at runtime, allow the reassessment and update of current reward values  $R(s,a)$  ...”*, allows us to answer the research question **RQ3** proposed in Chapter 1, section 1.2.

The second part of the hypothesis: *“... POMDP runtime model to improve accordingly the trade-off levels of the satisficement of the NFRs for the new environmental conditions.”*, allows us to answer the research questions **RQ1** and **RQ2**.

The initial setup and the details of the experiments for decision-making under uncertainty subject to dynamic environments are presented as follows.

## 6.1 Infrastructure used during the evaluation

The behaviour of the RDM SAS is implemented with the simulation model [4] provided by the DESPOT toolkit [96, 2, 4]. DESPOT has been selected due to its availability as open source and the results obtained on solving POMDPs for real settings [4]. The RDM network and its behaviour is based on the substantial case study presented in [29] and the more general specifications and expert-based knowledge presented in [47]. The case study RDM SAS of this dissertation comprises 25 RDM servers with 300 physical network links. Each link can be activated or deactivated. While active, each link can be used to transfer data between RDM servers. For the RDM configuration the minimum number of Active Network Links (ANL) expected is equal to 24 (i.e., number of *RDM servers* - 1 = 25 - 1 = 24) [29].

The RDM application has been simulated over 1000+ time slices (simulations of 1000, 2000 and 3000 thousand time slices have been performed). During each

simulation, periods of dynamic perturbances have been randomly inserted, which are detailed in section 6.2.1, Step 3.

## 6.2 Initial setup of experiments with the RDM SAS

The sources of uncertainty, and initial requirements specifications used in the evaluation are formalised below.

### 6.2.1 Identification of sources of uncertainty and their treatment

Expert judgement is an important source of information for the specification of the system [63]. System's experts and requirements engineers should identify sources of uncertainty that can affect the SAS at runtime to then initialise the system accordingly. There are sources of uncertainty to be considered in a SAS during the whole life cycle of the software system as shown in the taxonomy presented in [1]. In this dissertation, our main concerns are related to the uncertainty produced during the system's execution. Within this classification presented in [1], RE-STORM deals specifically with the runtime uncertainty due to unpredictable environments [1]. Two main POMDP elements deal with the sources of uncertainty provoked by these unpredictable environments: the transition function  $T(s, a, s')$  and the observation function  $O(s', a, z)$ . They both help model the uncertainty and require to be initialised in the RDM SAS. The steps carried out for their specification are presented as follows.

- a) **Step 1: Specification of the transition function  $T(s, a, s')$ .** The transition function  $T(s, a, s') = P(s' | s, a)$ , represents the probability that the system has of making a transition from  $s$  to a state  $s' \in S$  as a result of the execution of an action  $a$ , which represents that the action effects in the new state of the system are subject to uncertainty. Several techniques to initialise a model-based SAS can be applied to capture the experts' knowledge into the probability distributions. These include, for example, prior distributions for Bayesian Analysis [48, 62]. For instance, the RDM network parameters are initialized according to the known probability that certain network links will fail at any given point during

the system's execution as described in [29] and [47]. The results of this step are the specification of the conditional probabilities of the transition function  $T(s, a, s')$  shown in Tables 4.4, 4.5 and 4.6.

- b) **Step 2: Specification of the observation function  $O(s', a, z)$ .** The observation function  $O(s', a, z) = P(z|s', a)$ , represents the probability of observing  $z \in Z$  when the action  $a$  is performed and the current state is  $s'$ . The observation function models the uncertainty related to the stochastic nature of noisy sensors observations. The results of this step are the specification of the conditional probabilities of the observation function  $O(s', a, z)$  shown in Tables 4.7, 4.8 and 4.9.
- c) **Step 3: Configuration of dynamic changes in the environment.** This step is related to the nature of the experiment that uses a simulator. Therefore, the impacts of the topologies MST and RT on the levels of satisficement of the NFRs of the RDM SAS, which change at runtime, were simulated using DESPOT. To do that the uncertainty modelled by the transition function  $T(s, a, s')$  was randomly modified to introduce the changes required explained as follows:

*The probability distributions of the transition function  $T(s, a, s') = P(s' | s, a)$ , were randomly changed at runtime. They increment the uncertainty to be managed by our proposals: RE-STORM and ARRoW, during the simulation of the RDM SAS.*

Specifically, the new impacts were conceived to make the SAS to show periods of deteriorated satisficement of its NFRs and evaluate RE-STORM. Accordingly, the probability taken from Equation (4.4):  $P(NFR^{(1)} \dots NFR^{(n)} = \text{True} | NFR^{(1)} \dots NFR^{(n)}, A)$ , in the transition function  $P(s'|s,a)$ , will be lower than if these periods had not been simulated.

The different dynamic contexts have been simulated by randomly choosing the duration of each period of deteriorated satisficement, which were between 5 and 15 time slices based on the data provided by [47]. A maximum deviation of 12% from the current transition function  $P(s'|s,a)$  was targetted. Details on the dynamic contexts simulated during the system's execution, are described in section

6.4, , but before it is explained the setting of the initial preferences and the SLAs.

## 6.2.2 Setting of the initial preferences and Service Level Agreements (SLAs) for NFRs

As part of the initial setting of ARRoW (depicted in section 5.4.1), Service Level Agreements (SLAs) for the satisficement levels of the NFRs and initial preferences, i.e. reward values  $R(s,a)$ , require to be specified.

- a) **Service Level Agreements (SLAs).** The identified SLAs for the NFRs *Minimization of Cost* (MC), *Maximization of Reliability* (MR) and *Maximization of Performance* (MP) were:

$$P(MC=True) \geq 0.7 ,$$

$$P(MR=True) \geq 0.9 , \text{ and}$$

$$P(MP=True) \geq 0.75$$

The zones of poor and suitable satisficement for each NFR are highlighted in Table 6.1.

TABLE 6.1: RDM SAS - SLAs

Non Functional Requirements (NFRs)	Zone of poor satisficement (under the threshold)	Zone of suitable satisficement (over the threshold)
Minimization of Cost (MC)	[0.00 , <b>0.70</b> ) $\leftarrow$	[0.70 , 1.00]
Maximization of Reliability (MR)	[0.00 , <b>0.90</b> ) $\leftarrow$	[0.90 , 1.00]
Maximization of Performance (MP)	[0.00 , <b>0.75</b> ) $\leftarrow$	[0.75 , 1.00]

- b) **Reward values  $R(s,a)$ .** The initial preferences of the RDM SAS are shown in the column “Reward values  $R(s,a)$ ” of Table 6.2.

TABLE 6.2: RDM SAS - Reward values  $R(s,a)$ 

Reward values $R(s,a)$	Action (A)	State (S)		
		NFR <sup>1</sup> =MC	NFR <sup>2</sup> =MR	NFR <sup>3</sup> =MP
$r_1=0.0919$	$a^1=MST$	T	T	T
$r_2=0.0943$	$a^1=MST$	T	T	F
$r_3=0.0896$	$a^1=MST$	T	F	T
$r_4=0.0377$	$a^1=MST$	T	F	F
$r_5=0.1014$	$a^1=MST$	F	T	T
$r_6=0.0660$	$a^1=MST$	F	T	F
$r_7=0.0306$	$a^1=MST$	F	F	T
$r_8=0.0023$	$a^1=MST$	F	F	F
$r_9=0.1014$	$a^2=RT$	T	T	T
$r_{10}=0.0731$	$a^2=RT$	T	T	F
$r_{11}=0.0660$	$a^2=RT$	T	F	T
$r_{12}=0.0613$	$a^2=RT$	T	F	F
$r_{13}=0.0660$	$a^2=RT$	F	T	T
$r_{14}=0.0377$	$a^2=RT$	F	T	F
$r_{15}=0.0542$	$a^2=RT$	F	F	T
$r_{16}=0.0259$	$a^2=RT$	F	F	F

They are used by RE-STORM for the selection of self-adaptation actions during the online planning activity specified in section 4.4.2. During the execution of the experiments, the preferences above may need to be updated at runtime by ARRoW, to therefore favour the use of a topology which better contributes to improve the satisficement level of NFRs when their SLAs are detected to be unacceptable.

### 6.3 Implementation and evaluation of RE-STORM and ARRoW

In this section, details on the implementation and evaluation of our proposals RE-STORM and ARRoW are provided to facilitate their adoption on other application domains as a general framework to support decision-making in SASs subject to uncertainty, and driven by the satisficement levels of their NFRs. Fig. 6.10 shows a high level view of the configuration at design-time required by both approaches.

RE-STORM and ARRoW represent a model-based approach to support decision-making in SASs [96, 4, 55]. As such, the initial representation of the *SAS environment* should be specified at design-time to be used during the system's execution. Next, the configuration files of this environment are briefly described:

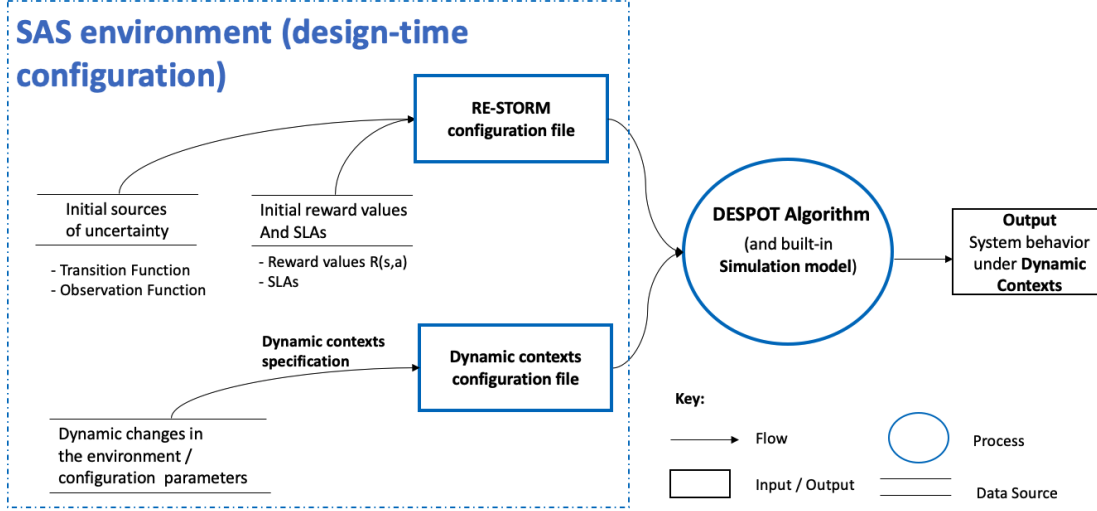


FIGURE 6.1: Implementation and evaluation of RE-STORM and ARRoW: inputs and output

### 6.3.1 Configuration file of RE-STORM

At the current stage, it supports the behaviour of the SAS under the *stable conditions* reported in section 6.4.2. In a future version, it will complement the RE-STORM Simulation Tool (ST) described in section 6.9. Next, the main parameters of the configuration file are presented:

- Transition function.** It is a set of real values between  $[0,1]$ . They represent the probability distributions of the transition function  $T(s, a, s') = P(s' | s, a)$  in a POMDP. Details on their specification in terms of the NFRs of a SAS have been presented in sections 4.3.3 and 6.2.1 (step "1"). The specific values assigned to this parameter for the RDM SAS have been presented in the Tables 4.4, 4.5 and 4.6.
- Observation function.** It is a set of real values between  $[0,1]$ . They represent the probability distributions of the observation function  $O(s', a, z) = P(z | s', a)$  in a POMDP. Details on their specification in terms of the MON variables of a



SAS have been presented in the sections 4.3.3 and 6.2.1 (step “2”). The specific values assigned to this parameter for the RDM SAS have been presented in Tables 4.7, 4.8 and 4.9.

- **Reward valuers  $R(s,a)$ .** It is a set of real value between  $[0,1]$ . They represent the initial preferences of the system’s stakeholders, i.e the obtained reward  $R(s,a)$  after taking action  $a \in A$  at time  $t$ , to arrive to the new state  $s \in S$  at time  $t + 1$ . Details on their specification have been presented in sections 4.3.2 and 6.2.2. The values assigned to this parameter for the RDM SAS have been presented in Table 6.2.
- **Thresholds for the levels of satisficement of NFRs.** They represent the *service level agreements* (SLAs) to be monitored during the system’s execution. They are used by the ARRoW approach to trigger the need of update of reward values  $R(s,a)$  in a POMDP. The default values used for the NFRs *Minimization of Cost* (MC), *Maximization of Reliability* (MR) and *Maximization of Performance* (MP) are  $[0.7, 0.9, 0.75]$ . Different thresholds have also been evaluated and reported in section 6.6.

In the next section, the configuration file to enable the behaviour of the dynamic contexts  $DC_1$  to  $DC_6$  is described.

### 6.3.2 Configuration file of the dynamic contexts $DC_i$ to $DC_6$

This configuration file supports further simulation of the SAS under different dynamic environments. Next, its main parameters are presented:

- **Dynamic context  $DC_i$  to be activated.** It is an integer value (between 0 and 5) which represent the dynamic context  $DC_i$  to be activated.
- **Noise factor.** It is a real value between  $[0,1]$  which represent the probability of activation of the selected dynamic context  $DC_i$ . The default value used during this evaluation is 0.5.
- **Deviation range of the selected dynamic context  $DC_i$ .** It is a range of values  $[lowerBound, upperBound]$ , from which a *real number* is randomly selected to

decrease specific probability values (See Tables 4.4, 4.5 and 4.6), accordingly to the dynamic context selected.

For example, for the range **[0.1, 0.15]** and the dynamic context  $DC_1$ :  $P(MR_{t+1} = \text{True} | \text{NFR}_t, \text{MST}_t)$ , a random value between 10% and 15% will be selected to reduce the current positive impact of the topology  $\text{MST}_t$  over the reliability of the system (i.e.  $\text{MR}_{t+1} = \text{True}$ ).

- **Length of the selected dynamic context  $DC_i$ .** It is a range of values [lowerBound, upperBound], from which an *integer number* is randomly selected to specify the number of timeslices that the selected dynamic context is performed. The default range of values used during this evaluation is **[5, 15]**.
- **Flag to update reward values  $R(s,a)$ .** It is an integer value (0 or 1) to determine if the ARRoW approach is performed when a NFR is detected below its threshold of satisficement (i.e. below its SLA) during the execution of the dynamic context selected. For example, the behaviour reported for each dynamic context  $DC_i$  in Appendix C, section “*Behaviour before reassessment of reward values  $R(s,a)$* ”, has been obtained by using the **flag value 0**. Conversely, the behaviour reported under the section “*Behaviour after reassessment of reward values  $R(s,a)$* ”, has been obtained with the **flag value 1**.
- **Thresholds for the levels of satisficement of NFRs.** They represent the *service level agreements* (SLAs) to be monitored during the system’s execution. They are used by the ARRoW approach to trigger the need of update of reward values  $R(s,a)$  in a POMDP. The default values used for the NFRs *Minimization of Cost* (MC), *Maximization of Reliability* (MR) and *Maximization of Performance* (MP) are **[0.7, 0.9, 0.75]**. Different thresholds have also been evaluated and reported in section 6.6.

Our proposal has been successfully applied on other application domains [65, 12]. As in those cases, the implementation and evaluation of any new application domain supported by RE-STORM and ARRoW can be managed through the update of the parameters in the configuration files described above. Additionally, further customization of the SAS environment could be reached by updating the simulation

model (See Fig. 6.1) provided by the DESPOT algorithm [96]. Details on its use are presented as follows.

### 6.3.3 DESPOT simulation model

The DESPOT simulation model [96, 4] has been implemented in C++ and it is included in the DESPOT toolkit<sup>1</sup>. It is used by RE-STORM to produce an approximated optimal policy to support the decision-making process (See section 4.4.3). One advantage of the use of a simulation model is that it comes with the flexibility of integrating the stakeholders' domain knowledge into the representation of the *SAS environment*. Therefore, a simulation model do not necessarily require an explicit representation of the POMDP model. For example, the probability distributions shown in Tables 4.4, 4.5 and 4.6, reflect an explicit representation of the transition function  $T(s, a, s') = P(s' | s, a)$  for the RDM SAS, i.e. each possible transition is represented with an explicit probability distribution. Contrarily, a non-explicit representation based on the stakeholders' domain knowledge, can help to simplify this representation. Listing 6.1 shows an excerpt of non-explicit representation for the RDM SAS. For example, based on the stakeholders' knowledge of the system, a non-explicit representation could be the assumption that under the execution of the MST topology (See Listing 6.1, line 7) the state values of the NFRs MC, MR and MP, stochastically depend on a random factor previously determined at design-time (See listing 6.1, lines 8, 9, and 10). This not explicit representation, would allow to simplify the transition function  $T(s, a, s')$ , from the representation  $P(s' | s, a)$  to a reduced version  $P(s' | a)$ , based on specific domain knowledge of the system. One additional advantage of this representation is that it support the specification of large problems, e.g. a system with a state space of size  $10^{56}$  [96], where an explicit representation for each possible state of the system would be unviable. The use of simulation models (a.k.a. generative models) in the Artificial Intelligence field, have enabled POMDP solvers to produce approximated optimal solutions for real problems with large state and observation spaces [41, 55, 96, 4].

<sup>1</sup>DESPOT toolkit git repository: <https://github.com/AdaCompNUS/despot>

LISTING 6.1: DESPOT simulation model - Excerpt of the Step function

```

1  . . .
2  bool Rdm::Step(State& s, double random_num, int action, double& reward,
3  OBS_TYPE& obs) const {
4
5  RdmState& state = static_cast<RdmState&>(s);
6
7  if (action == MST){
8  state.mc_satisfaction=random_num<=0.6 ? True : False;
9  state.mp_satisfaction=random_num<=0.6 ? True : False;
10 state.mr_satisfaction=random_num<=0.2 ? True : False;
11 }
12 . . .

```

In the next section, details on the experiments performed to evaluate RE-STORM and ARRoW are presented.

## 6.4 Experiments for decision-making under uncertainty related to dynamic changes in the environment

In Chapter 4, the formalisation of Mapping Rule 2 established the link between the preferences in a SAS and the reward values  $R(s,a)$  in a POMDP. In this section, the evaluation of the need of reassessment for the current reward values  $R(s,a)$  due to dynamic changes in the environment, and the way how RE-STORM and ARRoW act accordingly is presented below. Specifically, we showcase:

- A template for the specification of dynamic contexts affecting the behaviour of the RDM SAS.
- The behaviour of the RDM SAS under stable conditions.
- The behaviour of the RDM SAS under different dynamic contexts.

### 6.4.1 Template for the specification of dynamic contexts

A Dynamic Context (DC) represents situations where unexpected changes are produced in the environment during the system's execution (e.g. unexpected data packets loss in the RDM network). We have implemented different dynamic context by randomly changing the transition function  $T(s, a, s') = P(s' | s, a)$  as it was stated in section 6.2.1, Step 3. Fig. 6.2 overviews the template used for the specification of the Dynamic Contexts  $DC_i$  applied over the RDM SAS during its execution.

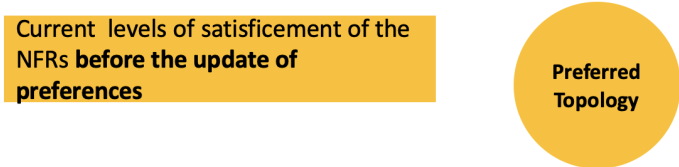
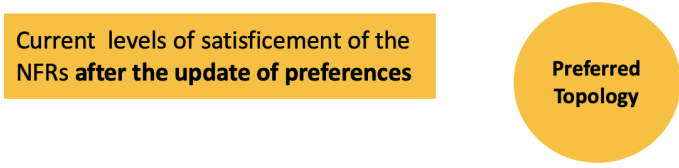
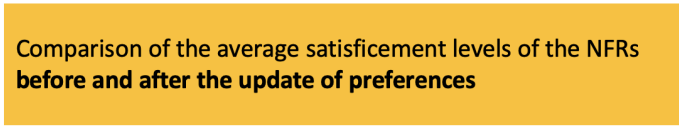
<b>DC<sub>i</sub> name</b>	<i>Title and probability representation of the changes in the environment</i>
<b>DC<sub>i</sub> specification</b>	<i>a) Dynamic context description</i>
	<i>b) Behaviour before reassessment of reward values <math>R(s,a)</math></i> 
	<i>c) Behaviour after reassessment and update of reward values <math>R(s,a)</math></i> 
	<i>d) Average satisficement level of NFRs before and after update of reward values <math>R(s,a)</math></i> 

FIGURE 6.2: Dynamic context - specification template

The template is composed by two main parts. The first part specifies the name of the context. It presents a title describing the changes introduced to affect the behaviour of the RDM SAS. The name also includes the conditional probability that represent the specific changes performed over the current transition function  $T(s, a, s') = P(s' | s, a)$ .

The second part of the template presents the details of the specification of the dynamic context, which are described below:

- a) *Dynamic context description.* The changes produced in the environment during the system's execution, in terms of the application domain, i.e. a Remote Data Mirroring (RDM) network, are explained.
- b) *Behaviour before reassessment of reward values  $R(s,a)$ .* The RDM SAS continues its execution using the initial reward values  $R(s,a)$  specified in section 6.2.2. The decision-making provided by RE-STORM uses this initial setting. The effects of the dynamic context on (i) the satisficement levels of the NFRs and (ii) the selected topologies (MST or RT) are reported.
- c) *Behaviour after reassessment and update of reward values  $R(s,a)$ .* The initial reward values  $R(s,a)$  have been updated by ARRoW and the decision-making process provided by RE-STORM uses the updated values. The effects of the dynamic context and the updated reward values  $R(s,a)$  on (i) the satisficement levels of the NFRs and (ii) the selected topologies (MST or RT) are reported.
- d) *Average satisficement levels of NFRs before and after update of reward values  $R(s,a)$ .* A comparison of the average satisficement levels before and after the update of reward values  $R(s,a)$  for each NFR, is presented.

#### 6.4.2 RDM SAS under stable conditions

Figs. 6.3, 6.4 and 6.5 show the behaviour of the RDM SAS using the setup specified in section 6.2. Under this configuration, as explained below, the satisficement levels of the NFRs *Minimization of Cost* (MC), *Maximization of Reliability* (MR) and *Maximization of Performance* (MP) are in general over their Service Level Agreements (SLAs). This behaviour is taken as that shown by the RDM SAS in *stable conditions*.

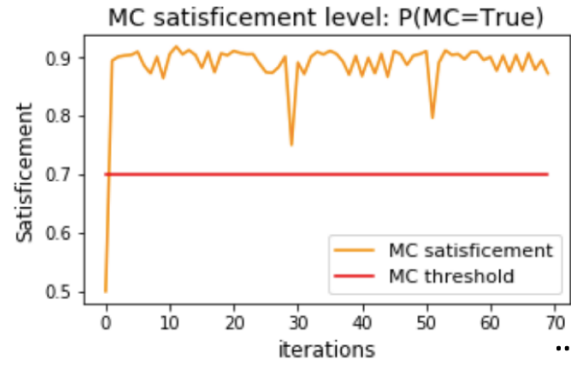


FIGURE 6.3: Minimization of Cost (MC) - satisficement level under stable conditions

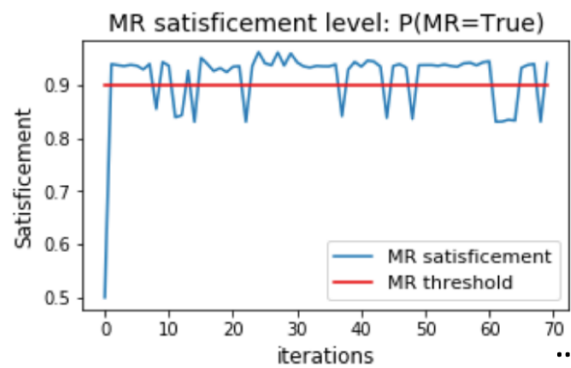


FIGURE 6.4: Maximization of Reliability (MR) - satisficement level under stable conditions

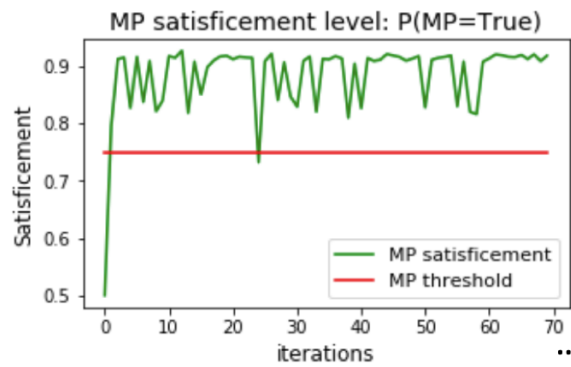


FIGURE 6.5: Maximization of Performance (MP) - satisficement level under stable conditions

Specifically, under *stable conditions*, the satisficement levels of MC and MP mostly meets the SLAs previously identified in section 6.2.2, i.e. they are over their threshold. The satisficement level of MR shows some values below its threshold, but this behaviour is considered to be part of a tolerance level in the system.

It is also observed that the preferred configuration is to use a *Minimum Spanning Tree Topology* (MST) (See Fig. 6.6), which has a positive impact on the satisfaction levels of the NFRs MC and MP (e.g. by saving inter-site network traffic) in comparison to the use of a *Redundant Topology* (RT), as was depicted in section 4.2.

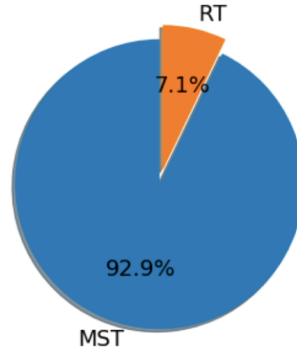


FIGURE 6.6: Chosen topology under stable conditions

*Under stable conditions, the current preferences of the RDM SAS, favour the use of the MST Topology and the satisfaction levels of the NFRs are in general over their satisfaction thresholds.*

### 6.4.3 RDM SAS dynamic contexts

Six different dynamic contexts were defined to be used for further simulations of the RDM SAS. In order to evaluate the approach, each of those dynamic contexts served to simulate changes in the environment to therefore trigger the need for reassessment of stakeholders' preferences ( i.e. the reward values  $R(s,a)$  in the POMDP requirements-aware model). The behaviour shown by the RDM SAS in *stable conditions* serves as the context which, the rest of Dynamic Contexts are compared against. The nature of the changes related to the six (6) different dynamic contexts are presented next:

- **Dynamic Context DC<sub>1</sub>.** Changes in the environment during the execution of the MST topology are introduced to reduce the reliability of the system:  $P(MR_{t+1} = \text{True} | \text{NFR}_t, \text{MST}_t)$ .



Dynamic context  $DC_1$  description. A period of consecutive and unexpected data packet loss during the execution of the MST Topology is generating a reduction on the reliability of the system. A MST topology is designed to connect all remote sites in an RDM SAS by the identification of a minimum spanning tree on the network of possible links among each remote site. Data packet loss may represent link failures in a RDM system, which may be caused, for example, by problems with the equipment (e.g. failures in a switch or router or power failures [47]).

- **Dynamic context  $DC_2$ .** Changes in the environment during the execution of the RT topology are introduced to increment the cost and to reduce the performance of the system:  $P(MC_{t+1} = \text{True}, MP_{t+1} = \text{True} | NFR_t, RT_t)$ .

Dynamic context  $DC_2$  description. Unexpected data packet loss during the execution of the RT Topology, are generating an unusual rate of data forwarding, which would increase the bandwidth consumption (i.e. cost) and would reduce the system's performance. In the RDM SAS, the cost for inter-site links communication is a function of the data sent over them. Therefore, a *Redundant Topology* (RT), which involves a bigger number of inter-site network links than a *Minimum Spanning Tree Topology* (MST), is more expensive. Costs increase as the number of network links increases and a reduction on the system's performance<sup>2</sup> could also be expected.

- **Dynamic context  $DC_3$ .** Simultaneous occurrence of the dynamic context  $DC_1$  and  $DC_2$ .
- **Dynamic context  $DC_4$ .** Changes on the environment during the execution of the MST topology are introduced to increment the cost and to reduce the reliability and the performance of the system:  $P(NFRs_{t+1} = \text{True} | NFR_t, MST_t)$ .

---

<sup>2</sup>The performance in these systems is measured as the total time to perform the write of data, which is the sum of the response times of the writes of each copy of data on each remote site [47].

Dynamic context DC<sub>4</sub> description. Unexpected data packet loss during the execution of the MST Topology generates an unusual reduction on the reliability of the system (i.e. DC<sub>1</sub> context behaviour). In contrast to DC<sub>1</sub>, we also observe the increment in bandwidth consumption and the reduction of the system's performance (NFRs MC and MP respectively).

- **Dynamic context DC<sub>5</sub>.** Changes on the environment during the execution of the RT topology are introduced to increment the cost and to reduce the reliability and the performance of the system:  $P(\text{NFR}_{t+1} = \text{True} | \text{NFR}_t, \text{RT}_t)$ .

Dynamic context DC<sub>5</sub> description. Unexpected data packet loss during the execution of the RT Topology is generating an increment in bandwidth consumption and reduction of the system performance (i.e. DC<sub>2</sub> context behaviour), but also a reduction on the positive impact of the RT topology over the reliability of the system.

- **Dynamic context DC<sub>6</sub>.** Simultaneous occurrence of the dynamic context DC<sub>4</sub> and DC<sub>5</sub>.

Dynamic context DC<sub>6</sub> description. This context represents an unusual scenario explicitly used to evaluate our approach under extreme detrimental conditions. A case like this would be usually related to a significant site failure [47, 50], where both repeated and multiple concurrent failures are expected [47] as in the previous two dynamic contexts but all at the same time. A full-scale site failure may be caused by a power outage affecting all the buildings on different campuses, an earthquake or flood affecting buildings within several metropolitan areas. Under this context, the worst-case data loss [50] may occur in different sites (RDM nodes), i.e. a site can be destroyed or inoperative before the full backup of information is shipped offsite. Site failure disasters are usually modelled with a failure rate of once per year [50]. The main goal

of this dynamic context is to study the behaviour of the RDM SAS using RESTORM in situations where it may be difficult to meet the SLAs regardless the adaptation action selected.

Each of these contexts are fully specified in the Appendix C. The template presented in section 6.4.1 has been used to show the information of each dynamic context. Specifically, details in terms of the application domain and their impacts on the current satisficement levels of the NFRs in the RDM SAS are shown in the appendix. A summary of their behaviour and main findings are presented next.

## 6.5 Results of experiments

Initial configurations and preferences under the *stable conditions* presented in section 6.4.2, favour the use of the MST topology in the RDM SAS. In general, during the experiments and under these conditions, the satisficement levels of the NFRs agreed with their SLAs (See Figs. 6.3, 6.4 and 6.5).

On the other hand, unexpected dynamic contexts detected at runtime (dynamic contexts  $DC_1$  to  $DC_6$ ), have produced negative impacts on the satisficement levels of the NFRs, causing that the initial assumptions do not fit anymore. A summary of the impacts of the dynamic contexts  $DC_i$  on the average satisficement levels of the NFRs in the RDM SAS is shown in Table 6.3.

TABLE 6.3: SLAs fulfillment based on the average satisficement levels of NFRs

Dynamic Context $DC_i$	Average satisficement level of NFRs before update of reward values $R(s,a)$			Average satisficement level of NFRs after update of reward values $R(s,a)$		
	MC	MR	MP	MC	MR	MP
$DC_1$	✓	✗	✓	✓-	✓	✓-
$DC_2$	✓	✓	✓	✓-	✓+	✓-
$DC_3$	✓	✗	✓	✓-	✓	✓-
$DC_4$	✓	✓	✓	✓-	✓+	✓-
$DC_5$	✓	✓	✓	✓-	✓-	✓-
$DC_6$	✓	✗	✓	✓-	✗+	✗

Key:	✓ Satisficement level over the threshold	✗ Satisficement level under the threshold	✓+ Increment of the satisficement level
		✗+ Increment of the satisficement level but still under the threshold	✓- Reduction of the satisficement level

In each dynamic context depicted, we show whether the average satisficement levels of the NFRs, agree with their SLAs before and after the update of reward values  $R(s,a)$  (See Table 6.3, columns 2 and 3 respectively). Next, the results of each dynamic context are described.

### 6.5.1 Summary and review of results of each of the 6 dynamic context

In the dynamic contexts  $DC_1$  and  $DC_3$ , the simulation of perturbances on the environment produced an important reduction on the reliability of the system. In both contexts, the reliability was under its satisficement threshold (See Table 6.4, column 2).

TABLE 6.4: Dynamic contexts  $DC_1$  and  $DC_3$  - average satisficement levels of NFRs

Dynamic Context $DC_i$	Average satisficement level of NFRs before update of reward values $R(s,a)$			Average satisficement level of NFRs after update of reward values $R(s,a)$		
	MC	MR	MP	MC	MR	MP
$DC_1$	✓	✗	✓	✓ <sub>-</sub>	✓	✓ <sub>-</sub>
	Preferred topology : MST (78.6%)			Preferred topology : RT (75.7%)		
	Changes introduced in $DC_1$ : $P( MR_{t+1} = \text{True} \mid NFR_t, MST_t )$					
$DC_3$	✓	✗	✓	✓ <sub>-</sub>	✓	✓ <sub>-</sub>
	Preferred topology : MST (100.0%)			Preferred topology : RT (72.9%)		
	Changes introduced in $DC_3$ : $P( MR_{t+1} = \text{True} \mid NFR_t, MST_t )$ and $P( MC_{t+1} = \text{True} \text{ and } MP_{t+1} = \text{True} \mid NFR_t, RT_t )$					

Key:	✓ Satisficement level over the threshold	✗ Satisficement level under the threshold	✓ <sub>+</sub> Increment of the satisficement level
		✗ <sub>+</sub> Increment of the satisficement level but still under the threshold	✓ <sub>-</sub> Reduction of the satisficement level

Initial preferences in  $DC_1$  and  $DC_3$  were not suitable anymore. Despite the new detected contexts, the preferred topology continued to be MST. After the reassessment performed by ARRoW, the reward values  $R(s,a)$  were updated at runtime, and the reliability of the system is taken by the decision-making process of RE-STORM, to levels where its average level of satisficement address the required SLA (See Table 6.4, column 3). A slight reduction on the performance and cost were also observed due to the trade-off among NFRs, but this reduction does not imply any risk to continue meeting the SLAs of cost and performance: both NFRs were over their thresholds.

The dynamic contexts  $DC_2$  and  $DC_4$  represent situations where despite dynamic changes at the environment, the current RDM configuration and preferences still kept the average level of satisficement of the NFRs over their thresholds (See Table 6.5, column 2).

TABLE 6.5: Dynamic contexts  $DC_2$  and  $DC_4$  - average satisficement levels of NFRs

Dynamic Context $DC_i$	Average satisfaction level of NFRs before update of reward values $R(s,a)$			Average satisfaction level of NFRs after update of reward values $R(s,a)$		
	MC	MR	MP	MC	MR	MP
$DC_2$	✓	✓	✓	✓ <sub>-</sub>	✓ <sub>+</sub>	✓ <sub>-</sub>
	<b>Preferred topology :</b> MST (98.6%)			<b>Preferred topology :</b> MST (68.6%)		
	<i>Changes introduced in <math>DC_2</math> :</i> $P( MC_{t+1} = \text{True}, MP_{t+1} = \text{True}, \mid NFR_t, RT_t )$					
$DC_4$	✓	✓	✓	✓ <sub>-</sub>	✓ <sub>+</sub>	✓ <sub>-</sub>
	<b>Preferred topology :</b> RT (61.4%)			<b>Preferred topology :</b> RT (71.4%)		
	<i>Changes introduced in <math>DC_4</math> :</i> $P( NFR_{t+1} = \text{True} \mid NFR_t, MST_t )$					

Key:

✓

Satisfaction level over the threshold

✗

Satisfaction level under the threshold

✗<sub>+</sub>

Increment of the satisfaction level but still under the threshold

✓<sub>+</sub>

Increment of the satisfaction level

✓<sub>-</sub>

Reduction of the satisfaction level

Table 6.5 shows that on average, the levels of satisficement of the NFRs were always meeting their SLAs. However, the reward values  $R(s,a)$  were updated in  $DC_2$  and  $DC_4$ , when in specific time slices during the system's execution, the satisficement levels of the NFRs were below their thresholds. The final result was a slight improvement on the reliability of the system (See Table 6.5, column 3). The trade-off effects over the cost and performance, also produced a reduction on their satisficement levels but still they were meeting their SLAs.

The dynamic context  $DC_5$  represents environments where the current impact of the RT topology over the satisficement levels of the NFRs is less favourable, i.e. the probability  $P(NFR_{t+1} = \text{True} | NFR_t, RT_t)$  has been reduced with respect to the *stable conditions* of the RDM SAS.

Under *stable conditions*, the RT topology has a perceived positive impact on the reliability of the system stated by the domain experts, and a less favourable impact on the cost and performance. However, in  $DC_5$ , the positive impact on the reliability

of the system has been reduced, while the negative impacts on the cost and performance have been incremented to take into account the changes introduced by this dynamic context.

TABLE 6.6: Dynamic context  $DC_5$  - average satisficement levels of NFRs

Dynamic Context $DC_i$	Average satisficement level of NFRs before update of reward values $R(s,a)$			Average satisficement level of NFRs after update of reward values $R(s,a)$		
	MC	MR	MP	MC	MR	MP
$DC_5$	✓	✓	✓	✓-	✓-	✓-
	Preferred topology : MST (100.0%)			Preferred topology : MST (62.9%)		
	Changes introduced in $DC_5$ : $P(NFR_{t+1} = \text{True} \mid NFR_t, RT_t)$					

Key:

✓

Satisficement level over the threshold

✗

Satisficement level under the threshold

✗+

Increment of the satisficement level but still under the threshold

✓+

Increment of the satisficement level

✓-

Reduction of the satisficement level

Accordingly, when the update of reward values  $R(s,a)$  is performed and the RT topology is used, it can be seen that RT topology is used 37.1% (see Table 6.6, 100% - 62.9% = 37.1%). The final result is a trade-off with a slight reduction on the average satisficement of the reliability, cost and performance of the system. Note, however, that the satisficement levels of cost, reliability and performance still meet their SLAs (See Table 6.6, column 3).

Finally, in the dynamic context  $DC_6$ , the most hostile environment designed for the experiments of this evaluation was studied.  $DC_6$  reflects the negative impacts of the dynamic contexts  $DC_4$  and  $DC_5$  simultaneously. Under  $DC_6$ , when applying RE-STORM with no re-assessment of weights, it was found that regardless the adaptation action selected by the RDM SAS, the environment showed a trade-off behaviour with a tendency to increment the cost (MC) while reducing the reliability (MR) and performance (MP) levels. The levels of satisficement of all the NFRs in  $DC_6$  were lower than those shown in the experiments with the *stable conditions*. When repeating the experiments of  $DC_6$  but using RE-STORM to update the reward





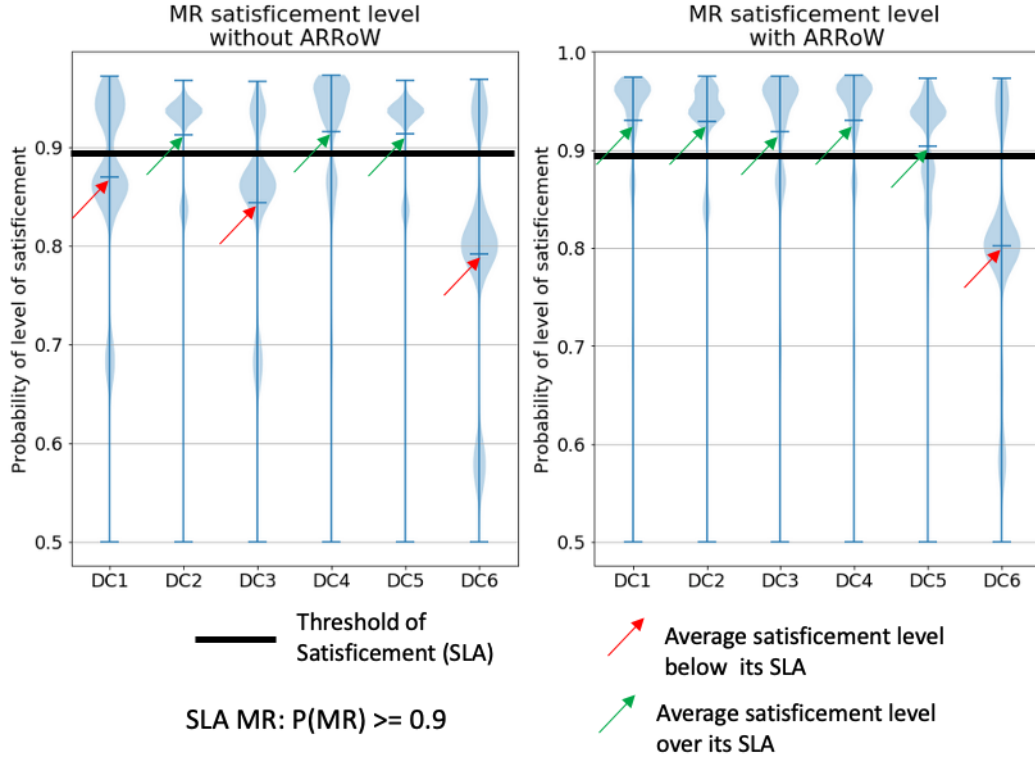


FIGURE 6.7: Dynamic contexts  $DC_1$  to  $DC_6$  - average satisfaction levels of MR

Under extreme hostile contexts, e.g. dynamic context  $DC_6$  or contexts with virtually unreachable SLAs under those conditions (See details in Appendix D, section D.2 stricter scenario), a better trade-off of the NFRs is still offered by RE-STORM. To determine how suitable is the approach under those specific contexts, is an opportunity for further evaluation of system's stakeholders through the self-explanation capacities of the system. Regarding this matter, it is part of the collaborative work within the Software Engineering at Aston (SEA<sup>3</sup>) research group, towards the improvement of the reflective capabilities in SAS. At the current stage, the RDM SAS is supporting the implementation of an externally-guided history-aware decision-making approach with explanation capabilities [33, 32]. Specifically, we argue that external entities (e.g. human stakeholders) will be able to evaluate and update the parameters of the POMDP requirements-aware runtime model, based on live explanations of the SAS behaviour, to therefore improve the self-adaptation and trade-off of the NFRs in a SAS.

<sup>3</sup>Software Engineering at Aston (SEA) research group <https://cs.aston.ac.uk/sea/>

### 6.5.2 Aggregated view of results

Figs. 6.8 and 6.9 synthesize the NFRs satisficement levels and preferred topology of the RDM SAS under the dynamic contexts  $DC_1$  to  $DC_6$ . It can be observed that when the decision-making process is applied under *new detected contexts* that were not foreseen in advance, and the *Weights Updater module* of RE-STORM was not used to update the reward values  $R(s,a)$  (See Fig. 6.8), then the satisficement levels of the cost (MC) and performance (MP) of the system met its Service Level Agreements (SLAs), but the satisficement level of the reliability of the system (MR) was always below its required SLA. Under this scenario, the preferred adaptation action is the MST topology (See Fig. 6.9a), which is favoured by unmatched initial reward values  $R(s,a)$ , i.e. initial preferences over the NFRs and adaptation actions in the system. On the other hand, when the reward values  $R(s,a)$  are updated accordingly to the new detected contexts, the satisficement level of MR is improved and taken to a value that meets its SLA. As a trade-off, a reduction on the satisficement levels of the cost (MC) and performance (MP) of the system is also observed, but still continues meeting their SLAs.

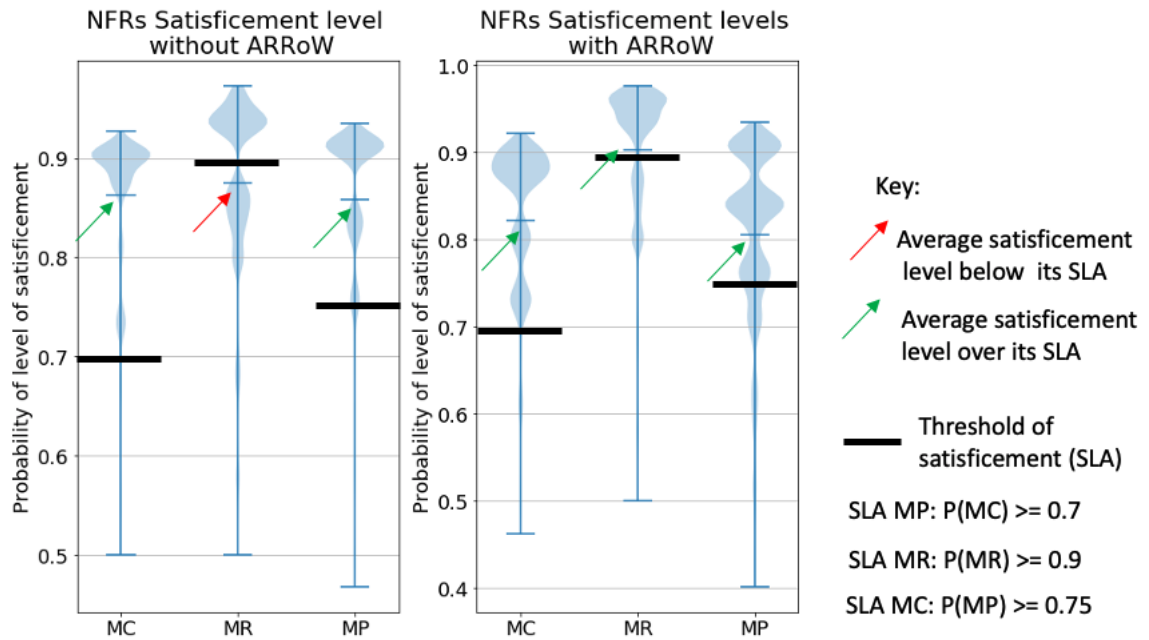


FIGURE 6.8: Dynamic contexts  $DC_1$  to  $DC_6$  - consolidated view of the average satisficement levels NFRs

As is shown in Fig. 6.8, the reassessment and update of preferences, as well as the decision-making driven by the current satisficement levels of the NFRs provided by

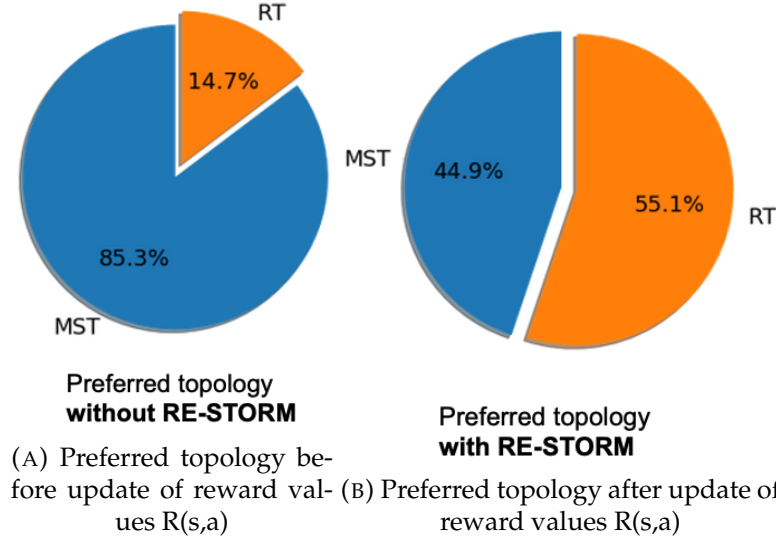


FIGURE 6.9: Dynamic contexts  $DC_1$  to  $DC_6$  - consolidated view of preferred topologies

our proposal, have improved the general performance and trade-off of the NFRs in the RDM SAS. RE-STORM provides the infrastructure to support the reassessment and update of the current reward values  $R(s,a)$  when dynamic context not previously foreseen are observed during the system's execution. An approximated optimal policy, to choose the best adaptation action under the current runtime context, is provided by the approach to improve the trade-off and the satisficement levels of the NFRs in a SAS.

## 6.6 Further analysis: behaviour of the RDM SAS under different Service Level Agreements (SLAs)

We have shown in section 6.5 how, under different dynamic contexts (from  $DC_1$  to  $DC_6$ ), the approaches RE-STORM and ARRoW have the following trade-off:

*The NFR with the lowest satisficement level with respect to its SLA has been improved by updating the current reward values  $R(s,a)$  and according to the new runtime context detected. Further, an acceptable reduction of the satisficement level of the other NFRs has been observed as a consequence of the trade-off performed. This reduction is usually affordable by meeting the stablished SLAs.*

A NFR within a poor zone of satisficement (i.e. with respect to its SLA) is taken to a suitable zone (i.e. over its threshold) with an acceptable reduction on the satisficement level of the other NFRs affected, but still over their minimum threshold of satisficement. This is the case of the dynamic contexts  $DC_1$  and  $DC_3$  reported in Table 6.3. Under these scenarios, a natural question arises:

*How would the RDM behave, supported by RE-STORM and ARRoW, using different SLAs as those used here for the experiments?*

The answer to this question is twofold. On the one hand, if the SLAs, different from the established in section 6.2.2, were less strict. For example by keeping the SLA related to the cost of the system as  $P(MC = \text{True}) \geq 0.7$ , but reducing the required reliability from  $P(MR = \text{True}) \geq 0.9$  to  $P(MR = \text{True}) \geq 0.8$  and the performance from  $P(MP = \text{True}) \geq 0.75$  to  $P(MP = \text{True}) \geq 0.6$ . Under this scenario, the execution of ARRoW to update the reward values  $R(s,a)$  would not be necessary. The satisficement levels of the NFRs MC, MR and MP and their SLAs under a less strict scenario are presented in Appendix D, section D.1

On the other hand, if the SLAs were stricter, the expectation would be that RE-STORM and ARRoW would have performed the trade-off identified at the beginning of this section, i.e. they should improve the lowest satisficement level for a NFR, even if it is not possible to take it to a suitable zone of satisficement over its threshold. Details on the behaviour of RE-STORM and ARRoW to improve the satisficement levels of the NFRs under a stricter scenario are presented in Appendix D, section D.2.

Additional tests considering different SLAs have been performed. For example, tests where the satisficement levels of the full set of NFRs are below their thresholds have been carried out as well. The obtained behaviour after the update of the reward values  $R(s,a)$  is equivalent to the presented above and detailed in Appendix D, i.e. RE-STORM and ARRoW, perform the trade-off of the NFRs independently of the SLAs in use, as stated at the beginning of this section.

## 6.7 Time-aware queries for internal evaluations of the RDM SAS

As part of our collaboration within the Software Engineering at Aston (SEA) research group, we used the time-aware query language developed in the project “History-aware explanation capabilities in SASs” [31, 33, 32]. It was used for our internal evaluations during the simulation of the RDM SAS and the implementation of our approaches RE-STORM and ARRoW.

Authors in [31], proposed temporal graph databases as a useful representation to trace models to support self-explanation in SASs. One approach to feed a temporal graph database is the following: a system runs as normal, while capturing logs in a machine-parseable form. After the system has finished its execution, its history is converted into a temporal graph database conforming to a reusable trace metamodel. We can then study its history with a time-aware query language [31]. A time-aware query is based on a time-aware query language and enables the access to the information recorded of the traces of execution of a system.

The experiments and the analysis presented in this chapter have been supported by different time-aware queries to help to explain why the system took a decision and why it is showing the current behavior. The queries were performed over logs that represent 1000 timeslices of execution of the system and have been presented in [31, 33, 32].

An example of time-aware query, implemented to detect situations where RE-STORM considers the behaviour described in section 4.4.2, i.e. “the long-term effects of immediate actions”, can be consulted in Appendix B. In the next section, the author contrast his contribution against related research work.

## 6.8 Comparison with related work

In recent years, different approaches for decision-making under uncertainty and driven by their NFRs have been proposed. Table 6.8 shows a summary of the work, which have already been depicted in Chapter 3. A comparison against the proposals RE-STORM and ARRoW is also presented to therefore complement the evaluation.

TABLE 6.8: Comparison of RE-STORM and ARRoW to other approaches

	Level of achievement				
	Scalability issues	"Long term effect" control in decision-making	NFRs representation (Partially Observable)	Preferences specification	Update of preferences (at runtime)
<b>Decision-making approaches</b>					
Garcia-Galan et al. [34]	✗	✗	✗	✓	✗
Song et al. [86]	✗	✗	✗	✓	✓
Letier et al. [57]	✓-	✗	✗	✓	✗
Elahi et al. [22]	✓-	✗	✗	✓	✗
Liaskos et al. [58]	✓-	✗	✗	✓	-
Peng et al. [70]	✓-	✗	✗	✓	✓
Sousa et al. [87]	✓-	✗	✗	✓	✓
Filieri et al. [28]	✓-	✓	✗	✗	✗
Bencomo et al. [7]	✗	✓	✗	✓	✗
Camara et al. [15]	✓-	✓	✗	✗	✗
Bowers et al. [13]	✓-	✓	✗	✓	✗
Moreno et al. [61]	✓-	✓	✗	✗	✗
<b>RE-STORM &amp; ARRoW</b>	✓-	✓	✓	✓	✓
Key: ✓ The approach overcomes the criterion    ✗ The approach does not fulfill the criterion    - The approach does not provide information    ✓- The approach is in process to fulfill the criterion					

The criteria for comparison of RE-STORM and ARRoW with respect to the other approaches are listed below:

- Scalability issues
- "Long term effects" control in decision-making under uncertainty
- NFRs representation (Partially Observable)
- Preferences specification at design-time
- Runtime reassessment and update of preferences

The column *Scalability issues* in Table 6.8, shows that the authors in [34, 86, 7] are still dealing with scalability issues, mainly related to the size of the state and action spaces [34, 86] and the planning horizon [7] in their implementations. Other approaches [70, 57, 22, 58, 28, 13, 61, 15, 87] do not present scalability problems.

However, all of them have presented initial implementations mostly related to a single application domain that still requires further exploration. Therefore, including RE-STORM and ARRoW, we decided to classify them as “in process to fulfill the criterion”. Specifically, for the case of RE-STORM, it uses a state-of-the-art POMDP implementation (the DESPOT algorithm) [4] that relies on sampling and approximation techniques to overcome the two main curses of POMDPs: the curse of “History” (i.e. beliefs grows exponentially with the planning horizon) and the curse of “Dimensionality” (i.e. belief grows exponentially with the number of states) [96]. Accordingly, RE-STORM overcomes previous scalability issues related to the planning horizon [7] and can work with a big number of NFRs (e.g. the DESPOT algorithm has been tested with a state spaces  $s \in S$  of size  $\approx 10^{56}$  [96]).

The next column, *Long term effect control*, allows us to classify the evaluated approaches in two main categories, those that use reactive control decision-making [34, 86, 57, 22, 58, 87] and their decision-making process relies on techniques restricted to the current state of the system, and those that takes into consideration the long term effects of their immediate actions [28, 8, 15, 61, 13] and uses sequential decision-making approaches such as Markov Decision Processes (MDPs) and Partially Observable MDPs. In our case, RE-STORM supports a decision-making process based on the Bellman’s principle of optimality. It considers the current state of the system but also projects future evolutions of the satisficement levels of the NFRs, as it was stated in Chapter 4, section 4.4.2. This capacity, enable RE-STORM to overcome, the well known potential problem in reactive approaches: i.e. choosing attractive short term actions with perhaps undesirable longer-term consequences [61].

The column *NFRs representation* shows how the approaches analysed assume full observability of the current state of the NFRs. Whether they use a general goal model [34, 86, 57, 22, 58, 87] or an implementation of a Markov Decision Process (MDP) [28, 8, 15, 61, 13], the approaches are not able to model the uncertainty related to the satisfiability of the NFRs in a system [8]. These approaches assume that the satisficement levels of the NFRs are fully observable at every time step. This assumption often does not hold in reality, as for example noisy sensors may limit the system’s perceptual abilities. Instead, with RE-STORM we obtain observations associated to the

NFRs. In a POMDP, the current observation alone is insufficient for choosing optimal actions. Therefore, the system's history, i.e. its past observations and actions, is encoded into a belief  $b$  [49]. In RE-STORM, beliefs (probability distributions) represent the current satisfaction levels of the NFRs in a SASs, which are used in the decision-making process presented in section 4.4.2.

In column *Preferences specification*, it is observed that most of the approaches present an explicit specification of preferences at design-time. These approaches range from preferences provided by system's stakeholders [7, 86, 87] to preferences determined by using a simulator [13]. In contrast, other approaches [28, 15, 61] scarce of explicit representation of preferences or assume the same weight, even if violation conditions for their NFRs are determined. In our case, RE-STORM uses a requirements-aware runtime model based on POMDPs to keep the initial preferences provided by system's experts. Then, later at runtime, and based on the current satisfaction level of the NFRs in a SAS, initial preferences could be reassessed and updated by ARRoW.

Finally, the column *Update of preferences* shows that most approaches do not update preferences at runtime, with the exception of the approaches in [86, 70, 87]. Moreover, the update of preferences in [86, 87] is not autonomous.

Different from the above, in our case, initial assumptions at design-time (i.e. preferences) may be updated during the system's execution to improve the decision-making according to new contexts, which may not have been foreseen. Specifically, RE-STORM supports runtime preferences reassessment and update by using ARRoW [38]. However, based on the modularity of the architecture in RE-STORM, another updaters can be used [9].

ARRoW is supported by a Multi Criteria decision-making technique: Primitive Cognitive Network Process (P-CNP). P-CNP represents an improved approach of the Analytic Hierarchy Process (AHP) in the aspects of paired interval scale and the corresponding mathematical development [99, 100, 102, 101]. P-CNP overcomes the drawbacks of knowledge representation with pairwise reciprocal matrices in AHP [78], i.e. it provides a more precise and natural representation of stakeholders' perception of paired comparisons [101].



ARRoW assigns higher importance (i.e. preference) to a NFR when a requirements violation is detected at runtime, i.e. its satisficement level is below a preestablished threshold. Both, the ARRoW model and the POMDP model, reside in the Knowledge Base of the MAPE-K architecture as runtime models.

## 6.9 Towards a simulation tool supported by RE-STORM and ARRoW

AS part of our research agenda, and the research collaborations reported in section 7.2, the approaches RE-STORM and ARRoW are intended to be available as a simulation tool (ST) in a near future. The simulation tool provides a well-defined and general abstract API, to enable users to quickly implement new elements in the POMDP requirements-aware model (See Fig. 6.10).

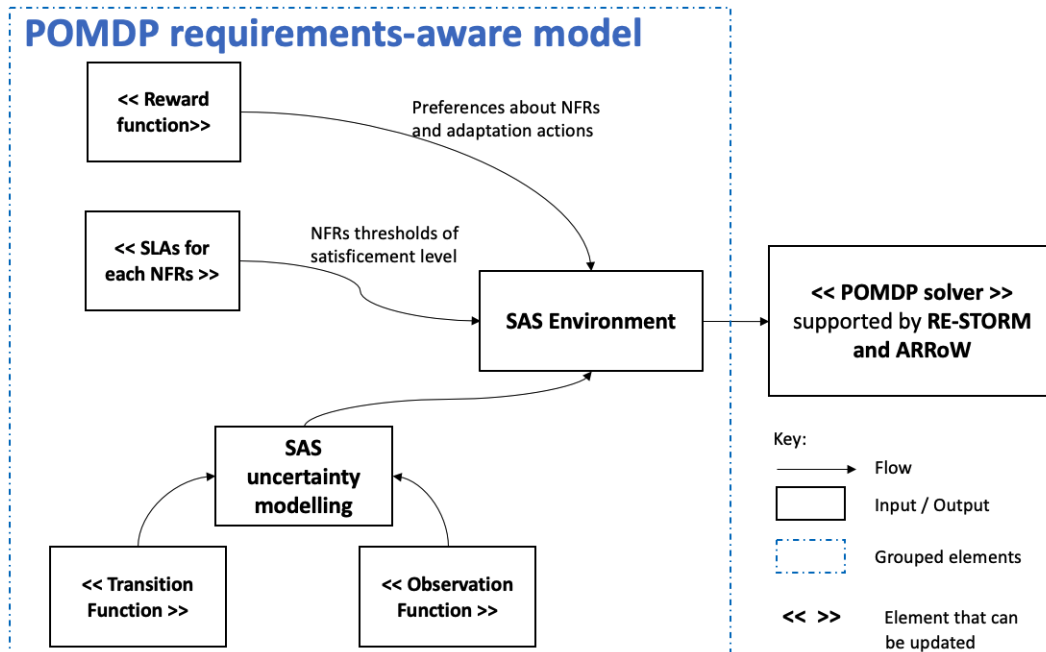


FIGURE 6.10: Simulation tool: architecture of implementation supported by RE-STORM and ARRoW

The simulation tool provides a default implementation of a POMDP requirements-aware model and a POMDP solver (based on the DESPOT algorithm [96]). Furthermore, the future improved version ideally, should provide an easy-to-use architecture based on interfaces that, in conjunction with user-friendly configuration files, would allow users to specify the **SAS environment** of a standard domain problem

for decision-making under partial observability with no additional coding effort. Fig. 6.10 shows different elements that would be updated by extending the simulation tool abstract API. For example, at the current stage, the POMDP solver element is implemented by the DESPOT algorithm [96, 4]. A possible new «*POMDP solver*» with support for a different algorithm could be implemented to enrich the set of POMDP solvers provided by the simulation tool. We envision the new elements will be implemented as shared libraries that are loaded dynamically during runtime. They will be used to implement:

- the SLAs for each NFR,
- the transition function,
- the observation function, and
- the reward function of the model

The elements implemented as shared libraries will support their exchange at runtime. Following the above, the simulation tool will provide better support for dynamic changing environments. For example, when the quality of a sensor deteriorates, one new «*Observation Function*» could replace its current implementation, while the ST is running.

## 6.10 Summary

In this chapter, we have presented the evaluation of the approaches: RE-STORM and ARRoW, during the simulation of the RDM SAS.

It was highlighted that the main concern in this work is the management of the uncertainty produced during the system's execution. To support modelling the runtime uncertainty, two main POMDP elements have been studied: the transition and the observation functions. During the execution of the experiments, six different dynamic contexts, from  $DC_1$  to  $DC_6$ , were introduced by randomly changing the transition function  $T(s, a, s') = P(s' | s, a)$ . These contexts, supported the evaluation of the following hypothesis, which was proposed at the first part of this chapter:

**H:** “Dynamic changes in the context, from the managed system observed at runtime, require the reassessment and update of current reward values  $R(s,a)$  of the POMDP runtime model to improve accordingly the trade-off levels of the satisficement of the NFRs for the new environmental conditions.”

Based on the evaluation discussed, the author argues how the application of the approaches RE-STORM and ARRoW in the RDM SAS gives enough evidence to support the hypothesis above.

First, RE-STORM is our contribution towards filling the lack of a *time-history wise scalable platform* to support NFRs specification and runtime trade-off of the satisficement levels of NFRs to drive the decision-making in SASs. The experiments described in section 6.4, show how RE-STORM deals with uncertainty in an explicit way and updates its definition over time as new evidence arrives, keeping consistency with the *history of the execution*. Specifically, RE-STORM uses Equation (2.1) to update the probabilities of the satisficement level for each NFR (i.e., beliefs), based on the previous belief  $b$ , the last action, and the last observation. The *history of the execution* (the past observations and actions) is encoded into a belief  $b$  in a POMDP. The POMDP implementation in RE-STORM is the DESPOT algorithm [96]. The experimental results obtained using the RDM case study in section 6.4 and other domains [65, 12] show that the adaptation actions made about the trade-off of the satisficement levels of the NFRs are soundness and compatible with other experiences [8, 7, 1]. Further, RE-STORM presents a scalable solution which respect to previous seminal work in [8, 7].

Second, the experiments described in section 6.4, and the analysis performed in sections 6.5 and 6.6, also show that different dynamic contexts, not previously foreseen, may negatively affect the satisficement level of a SAS. Specifically, the results show that when the decision-making process provided by RE-STORM, continues using the initial reward values  $R(s,a)$  under a *new detected context*, the satisficement level of the NFRs can be drastically reduced (See summary in Table 6.3), even below their required thresholds, due to unmatching initial reward values  $R(s,a)$ . In contrast, when the reward values  $R(s,a)$  are updated by ARRoW, the decision-making process provided by RE-STORM, improves the NFR with the lowest satisficement

level, taken it to a suitable zone satisficement, (depending on the SLA stablished) by leveraging access to *new runtime evidence*. As a trade-off, a slight reduction in the satisficement levels of the other NFRs, depending on their SLAs, can also be observed.

Finally, the experiments have shown that the reassessment and update of the reward values  $R(s,a)$  and the decision-making driven by the current satisficement levels of the NFRs provided by RE-STORM, improve the general performance of the RDM SAS by favouring a better trade-off of the satisficement levels of the NFRs, according to the *new runtime contexts detected*. Based on these findings, the hypothesis **H** is accepted as valid.

The author has also contrasted his research contributions against related research work. The next chapter presents the conclusions of the thesis taking into consideration the extent to which the work meets the research questions and the aim of this dissertation.

## Chapter 7

# Conclusions and Future Research

## Agenda

This chapter sum ups the results presented and concludes the dissertation. The main motivation for the implementation of the proposed approaches RE-STORM and AR-RoW has been the challenges related to the need of new techniques for decision-making under uncertainty and the dynamic update of preferences.

This chapter presents answers to the research questions of the dissertation in section 7.1. The chapter also presents research collaborations which are part of our ongoing work, in section 7.2. Finally, the chapter explores how the research can be developed further in section 7.3.

### 7.1 Research Questions

The three (3) research questions of the dissertation have been addressed in Chapters 4 and 5, and the evaluations described in Chapter 6. We present answers to these research questions in the following paragraphs:

- **RQ1:** *How can we represent the current state of NFRs and their evolution in model-based self-adaptive systems that are subject to uncertain environments?*

To address **RQ1**, we have developed RE-STORM, an approach that support decision-making under uncertainty driven by the current state (i.e. current level of satisficement) of the NFRs in a SAS. To do this, first, the formalization of several mapping rules (shown in section 4.3.2) allowed us to stablish the equivalence between the levels of satisficement of the NFRs in a SAS and the

states  $s \in S$  in a POMDP. The result is that a POMDP can provide a decision-making mechanism for a SAS. Further, the POMDP can act as a requirements-aware runtime model [9] in the context of the MAPE-K loop. The extended version of the POMDP presented enclosed by RE-STORM, maintains the current state of the NFRs and its evolution as is described as follows:

- Representation of the current state of the NFRs. By definition, the current state of the system in a POMDP is not directly observable as it is the case of the levels of satisficement of the NFRs in a SAS. Instead, the observations are obtained from the current runtime context, to therefore infer the satisficement levels of the NFRs in a system. RE-STORM uses Bayesian Inference and the mathematical background provided by POMDPs to represent the uncertainty of the current system's context as probability distributions (i.e. beliefs) over the levels of satisficement of the NFRs in a system.
- Representation of the evolution of the state of the NFRs. In RE-STORM, the current belief  $b_0$  about the satisficement levels of the NFRs, is constantly updated over time during the system's execution. Per each time slice, after executing an action  $a \in A$ , new evidence in the form of new observations  $z \in Z$ , are collected from the environment. Afterwards, Equation (2.1), shown in section 2.6.1, is applied to compute the new satisficement levels of the NFRs (i.e. the new belief  $b_0$ ). This is part of the continuous behaviour of the system determined by the transition function  $P(s' | s, a)$  and the observation function  $P(z | s', a)$  in the POMDP. Based on the “memoryless property” of any stochastic Markov Process described in section 4.4.2, the new belief  $b_0$  should be consistent with the past execution of the system. In a POMDP, the belief summarizes the previous experience of the system [49].

*With RE-STORM, the use of a requirements-aware runtime model based on the mathematical background of POMDPs allow for modelling the runtime uncertainty of the environment and its evolution using probability distributions (i.e. beliefs) over the*

*satisficement levels of the NFRs in a system.*

The above is the base to answer the research question **RQ2**.

- **RQ2:** *How can we improve the trade-off among NFRs in model-based self-adaptive systems that are subject to uncertain environments, by updating preferences and based on new evidence collected during execution?*

To address **RQ2**, let's focus on the fact that RE-STORM uses the Bellman's principle of optimality described in section 4.4. During the system's execution, per each time slice, the current belief  $b_0$  is used as the input of Equation (4.8), which (i) projects to the future the possible evolution of the satisficement level of the NFRs, makes the trade-off among them and (ii) computes an approximated optimal policy  $a = \pi(b_0)$  to select the best adaptation action  $a \in A$  in a SAS. The runtime architecture to support this behaviour has been depicted in section 4.4, and is leveraged by a POMDP requirements-aware runtime model framed within a MAPE-K feedback loop.

In Chapter 6 we have shown how RE-STORM improves the trade-offs by the use of ARRoW, our proposal for the reassessment and update of preferences over NFRs and adaptation actions. Specifically, ARRoW corresponds to the weights updater module of the runtime architecture of RE-STORM (See Fig. 4.3), and is performed when a NFR is below its SLA. Afterwards, RE-STORM continue with the execution of Equation (4.8).

The results show that when ARRoW updates unmatching preferences to the current runtime context detected, RE-STORM improves the trade-off and the satisficement levels of the NFRs in a SAS according to their Service Level Agreements (SLAs).

- **RQ3:** *How can techniques for eliciting initial preferences about requirements, used at design-time, be applicable to runtime models for self-adaptive systems?*

To address **RQ3**, we have developed ARRoW. In Chapter 5, we have shown how the Primitive Cognitive Network Process (P-CNP), a technique to prioritize alternatives based on a given criteria, was extended to be executed at runtime and support the specification of NFR states and adaptation actions as a P-CNP problem. The new extended version relies on P-CNP runtime abstractions: Pairwise Opposite Matrices (POMs), for the propagation of pairwise comparisons of the satisficement levels of the NFRs to therefore compute updated reward values  $R(s,a)$  (i.e. preferences) in a POMDP. The results show that the updated preferences provided by ARRoW, which match the current runtime context, allows RE-STORM to improve the trade-off and the decision-making in a SAS.

## 7.2 Exploited Research Collaborations

Our results, have currently fostered a collaboration within the Software Engineering at Aston (SEA) research group, investigating History-Aware explanation capabilities in SASs. Partial results of this collaboration have been presented in the publications “Reflecting on the past and the present with temporal graph-based models” [31], “Back to the Past: Towards History-Aware Self-Adaptation with Explanation Capabilities” [33] and “Querying and annotating model histories with time-aware patterns” [32].

In this collaboration, the RDM SAS is currently used as the case study for the evaluation of different levels of reflective capacities in SASs [33]. Four different levels have been defined: Forensic self-explanation (**Level 1**), Live Self-explanation (**Level 2**), Externally-Guided and History-Aware decision-making with explanation capabilities (**Level 3**) and Autonomous History-Aware decision-making with explanation/reasoning capabilities (**Level 4**). At the current stage, an extension over the simulation of the RDM SAS has been implemented to support the Level 3. External entities (in this case, human stakeholders) based on live explanations of the system’s behaviour, will be able to update at runtime the parameters of the system.



Other collaboration have been established with the research group of the Systems Engineering department at Pontificia Universidad Javeriana (Bogota, Colombia). Partial results of this collaboration have been presented in the publication “Improving the Decision-Making Support in Context-Aware Applications: The Case of an Adaptive Virtual Education Learning Management System” [12] and a journal paper is under preparation.

### 7.3 Future Research Agenda

The contributions presented in this dissertation can certainly be developed further. Specifically, new areas of research that would allow researchers to further work to eventually produce more useful knowledge have been identified. In this section the author proposes different areas of research to carry out additional work.

#### 7.3.1 Optimization of parameters for self-adaptation

- *Learning the preferences.* At the current stage, ARRoW uses P-CNP as a technique to dynamically update at runtime the reward values  $R(s,a)$  in a POMDP to therefore, improve the decision-making process and the satisficement levels of the NFRs in a SAS by using new evidence monitored at runtime. We believe that a next step is the computation of optimal or at least better preferences, that can produce better satisficement levels of the NFRs to follow the Service Level Agreements (SLAs). Recent progress in the machine learning technique Inverse Reinforcement Learning (IRL) [52, 95], offers the potential to learn preferences from the interaction with simulated or real environments. IRL represents the challenge of modeling preferences in a SAS to overcome a manual specification of its *reward function*  $R(s,a)$  [30].

In this proposal two natural steps would be involved. First, initial preferences provided by system’s stakeholders can be the departure point to learn an initial reward function  $R(s,a)$  suitable for what has been identified as a context of *stable conditions* in section 6.4.2. Next step would be to generalize the learned reward function  $R(s,a)$  to *different dynamic contexts* such as those showcased in section 6.4. Initial results from researchers in the machine learning field

[20], show that the reward function can be a transferable representation of the behaviour in a SAS to a different context, since it compactly represents the agent's objectives and preferences.

- *Learning the transition function*  $T(s, a, s') = P(s' | s, a)$  *and the observation function*  $O(s', a, z) = P(z | s', a)$ . Initial conditional probabilities (probability distributions) provided by system's stakeholders for modelling the uncertainty during the system's execution such as those described in section 6.2.1, can be a departure point to learn or infer the transition and observation functions to therefore obtain a more accurate representation of the runtime environment and improve the decision-making process. Techniques for learning causality with data [59] and approaches for learning POMDP models [19] are alternatives to be further explored in this area.

### 7.3.2 Optimization of the decision-making process

- *Continuous policy learning*. RE-STORM builds a belief tree where the root node is the current belief  $b$  of the SAS for each time slice during the system's execution. The belief tree represents projections to the future of the satisficement levels of the NFRs, adaptation actions and observations obtained from the environment. Over this structure, RE-STORM learns a policy  $a = \pi(b)$  for the current belief  $b$  of the SAS. This process is repeated for each time slice, i.e. a new policy  $a = \pi(b)$  is always "learned from scratch" on each time slice.

We believe that a natural following next step is the computation of a policy under *continuous learning* during the system's execution, as oppose to having to calculate them from scratch on each time slice. The use of machine learning techniques, e.g. Reinforcement Learning (RL) approaches [46] or neural network architectures for planning under partial observability [72] are possible paths to be explored. Initial results depicted in [72], show the feasibility of an architecture that combines deep reinforcement learning and approximate POMDP planning. These results prompt us to suggest that a policy could be learned by connecting a POMDP model with an algorithm that solves the

model (i.e. a POMDP solver such as DESPOT [96]), but embedding the solution structure of planning in a neural network learning architecture.

- *History-aware and self-explanation.* It is part of the ongoing research collaboration about explanation capabilities exposed by a SAS to improve the decision-making process by including the human in the loop, and promoting the understanding of the system by end users [31, 33, 32]. As part of our collaborations presented in section 7.2, we are working on experiments where findings [33, 32] detected in the runtime behaviour of the system and supported by time-aware queries [32], provide additional information to stakeholders for better understanding of the decision-making exposed by a SAS. These findings may trigger the need to eventually update at runtime parameters in a requirements-aware runtime model.

## Appendix A

# Publications

This appendix contains the titles and venues of the publications that the PhD research has produced so far:

### A.1 Conferences

1. *"RE-PREF: Support for REassessment of PReferences of Non-functional Requirements for Better Decision-making in Self-adaptive Systems"*, Luis Garcia-Paucar, Nelly Bencomo, Poster in 24th International Requirements Engineering Conference (RE '16), Beijing, China, September, 2016
2. *"Survey on Preferences of Quality Attributes in the Decision-making for Self-Adaptive Systems: the Bad, the Good and the Ugly"*, Luis Garcia-Paucar, Nelly Bencomo, in 20th IberoAmerican Conference on Software Engineering Steering (CibSE '17), Buenos Aires, Argentina, May, 2017
3. *"Juggling Preferences in a World of Uncertainty"*, Luis Garcia-Paucar, Nelly Bencomo, Kevin Yuen, in 25th IEEE International Requirements Engineering Conference (RE '17), Lisbon, Portugal, September, 2017
4. *"ARRoW: Tool Support for Automatic Runtime Reappraisal of Weights"*, Luis Garcia-Paucar, Nelly Bencomo, Poster in 25th IEEE International Requirements Engineering Conference (RE '17), Lisbon, Portugal, September, 2017
5. *"RE-STORM: Mapping the Decision-Making Problem and Non-Functional Requirements Trade-off to Partially Observable Markov Decision Processes"*,

Luis Garcia-Paucar, Nelly Bencomo, in 13th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '18), Gothenburg, Sweden, May, 2018

6. *"ARRoW: Automatic Runtime Reappraisal of Weights for Self-Adaptation"*, Luis Garcia-Paucar, Nelly Bencomo, in 34th ACM/SIGAPP Symposium on Applied Computing (SAC '19), Limassol, Cyprus, April, 2019
7. *"Knowledge Base K Models to Support Trade-offs for Self-adaptation using Markov Processes"*, Luis Garcia-Paucar, Nelly Bencomo, in 13th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO '19), Umea, Sweden, June, 2019
8. *"RaM: Causally-connected and Requirements-aware Runtime Models using Bayesian Learning"*, Nelly Bencomo, Luis Garcia-Paucar, in IEEE / ACM 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS '19), Munich, Germany, September, 2019

## A.2 Workshops

1. *"The Reassessment of Preference Non-Functional Requirements for Better Informed Decision-making in Self-Adaptation"*, Luis Garcia-Paucar, Nelly Bencomo, in Third International Workshop on Artificial Intelligence for Requirements Engineering (AIRE '16), Beijing, China, September, 2016
2. *"Runtime Models Based on Dynamic Decision Networks: Enhancing the Decision-making in the Domain of Ambient Assisted Living Applications"*, Luis Garcia-Paucar, Nelly Bencomo, Kevin Yuen, in 11th International Workshop on Models@run.time (MRT '16), Saint Malo, France, October, 2016

The author of this dissertation is the principal author of the publications enumerated above, with the exception of the publication A1.8. Additionally, the author's effort has contributed to the publication of the following papers as providing an application case study and evaluation support as well as being co-author:

1. *"Reflecting on the past and the present with temporal graph-based models"*, Antonio Garcia-Dominguez, Nelly Bencomo, Luis Garcia-Paucar, in 13th International Workshop on Models@run.time (MRT '18), Copenhagen, Denmark, October, 2018
2. *"Towards History-Aware Self-Adaptation with Explanation Capabilities"*, Antonio Garcia-Dominguez, Nelly Bencomo, Juan Marcelo Parra-Ullauri, Luis Garcia-Paucar, in IEEE 4th International Workshops Foundations and Applications of Self\* Systems (FAS\*W), Umea, Sweden, June, 2019
3. *"Querying and annotating model histories with time-aware patterns"*, Antonio Garcia-Dominguez, Nelly Bencomo, Juan Marcelo Parra-Ullauri, Luis Garcia-Paucar, in IEEE / ACM 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS '19), Munich, Germany, September, 2019

## Appendix B

# Example of time-aware query for the RDM SAS

In section 6.7, the use of time-aware queries is reported. Next, details on the implementation of a time-aware query to detect situations where an apparently bad decision in the current time slice, produced an increment on the satisficement level of a NFR in the long term, are presented.

### B.1 Time-aware query to show proactiveness in self-adaptation

It was stated in Chapter 4 that RE-STORM considers future evolutions of the satisficement levels of the NFRs (i.e. projections into the future) to decide the next action  $a \in A$ , and to therefore reason about long-term effects of immediate actions [85]. As a consequence of this behaviour, it is possible that the RDM SAS may decide to make a decision that is apparently bad in the short-term but good in the long-term, i.e. deceiving at first while improving the behaviour in the long term. Situations as such [32] may require an explanation. A scenario to clarify this behaviour is presented as follows.

- a) **Scenario illustrating proactiveness in self-adaptation.** We illustrate RDM's proactiveness with cases where a seemingly "bad" decision by the SAS turned out to be a good one in the long run.

---

**Algorithm 5** Query to detect proactive adaptation: the long term effects of immediate actions.  $L$  is the current runtime log,  $T$  the set of timeslices in  $L$ ,  $S^{\text{NFR}}(t)$  the satisficement of the NFR at timeslice  $t$ , and  $\alpha^{\text{NFR}}$  the threshold for the NFR.

---

```

1: Result = {}
2:  $T_B = \{t \in T \mid S^{\text{NFR}}(t) < \alpha^{\text{NFR}}\}$ 
3: for each  $t_b \in T_B$  do
4:   if  $S^{\text{NFR}}(t_b + 1) < S^{\text{NFR}}(t_b) \wedge$ 
        $\exists n \in \mathbb{N}_{>0}, \forall j \in [1, n] \mid$ 
        $S^{\text{NFR}}(t_b + j + 1) > S^{\text{NFR}}(t_b + j)$  then
5:     Add  $(t_b, n)$  to Result
6:   end if
7: end for
8: Result: Sequences showing proactive adaptation.
```

---

Algorithm 5 represents a query to find a timeslice during the system's execution, where the satisficement level of a NFR is below its threshold (e.g.  $\text{MR} \geq 0.9$ ), and the action suggested by the SAS under this context results in a further reduction on the next timeslice. However, as the action is further continued in the following timeslices, the satisficement gradually increases until reaching and exceeding its threshold. This is an example of the type of reasoning used in the RDM case study based on RE-STORM. The RDM SAS can predict in an uncertain environment what is the likely impact of the current action in the future.

#### b) Query results.

Listing B.1 shows an excerpt of the examples found by the query.

One of the detected sequences started at timeslice 138, when the RDM SAS decided to use the *Minimum Spanning Tree* (MST) topology. As an immediate consequence, a reduction on the satisficement level of the NFR *Maximization of Reliability* (MR) is observed: from 0.8943 (timeslice 138) to 0.8525 (timeslice 139). However, the satisficement grew during the following timeslices, until exceeding its threshold in timeslice 141.

A similar situation was shown in timeslice 324, among others. This shows that



LISTING B.1: Excerpt of output from Algorithm 5 about long term effect of immediate actions.

```

[[138, Minimum Spanning Tree Topology, Maximization of Reliability ,
  0.894321707807189, [[139, Minimum Spanning Tree Topology,
    Maximization of Reliability , 0.852577860667983], [140, Minimum
    Spanning Tree Topology, Maximization of Reliability ,
    0.897735592250711], [141, Minimum Spanning Tree Topology,
    Maximization of Reliability , 0.928494865846856]]],
[324, Minimum Spanning Tree Topology, Maximization of Reliability ,
  0.861605674968342, [[325, Minimum Spanning Tree Topology,
    Maximization of Reliability , 0.8466796875], [326, Minimum Spanning
    Tree Topology, Maximization of Reliability , 0.856691253951577], [327,
    Minimum Spanning Tree Topology, Maximization of Reliability ,
    0.925433890656174]]], ...]

```

decisions with apparently immediate negative effects, in the long term, are producing the expected increase of the satisficement level of the NFRs.

## Appendix C

# Dynamic Contexts to represent unexpected changes in the environment

This appendix contains details on the specification of the dynamic contexts  $DC_i$ , randomly applied over the RDM SAS to evaluate our approaches during the system's execution. Each context represents a new scenario not previously foreseen to therefore trigger the need for reassessment of stakeholders' preferences.

### C.1 Dynamic Context $DC_1$ : changes in the environment during the execution of the MST topology are introduced to reduce the reliability of the system: $P(MR_{t+1} = \text{True} | NFR_t, MST_t)$

- a) **Dynamic context  $DC_1$  description.** Let us revisit the description of the dynamic context  $DC_1$ , which was stated in section 6.4.3: *"A period of consecutive and unexpected data packet loss during the execution of the MST Topology is generating a reduction on the reliability of the system. A MST topology is designed to connect all remote sites in an RDM SAS by the identification of a minimum spanning tree on the network of possible links among each remote site. Data packet loss may represent link failures in a RDM system, which may be caused, for example, by problems with the equipment (e.g.*

*failures in a switch or router or power failures [47])."*

Next, the behaviour of the RDM SAS under the execution of the dynamic context  $DC_1$  is depicted.

- b) **Behaviour before reassessment of reward values  $R(s,a)$ .** Despite the new detected conditions in context, and based on the initial RDM configuration, i.e the initial stakeholders' preferences, the most selected topology continues to be MST (See Fig. C.1).

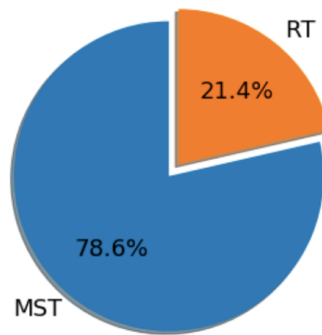


FIGURE C.1:  $DC_1$  - Chosen topology before the update of reward values  $R(s,a)$

It is also observed in Figs. C.2, C.3 and C.4, that this configuration reduces the satisficement level of the NFRs. In special, it clearly produces a poor satisficement level on the reliability of the system. Under the current dynamic context  $DC_1$ , the reward values  $R(s,a)$  (See Table 6.2), are not suitable anymore as they continue favouring the use of a topology that does not contribute to improve the satisficement level of MR, which is mainly under its tolerance threshold.

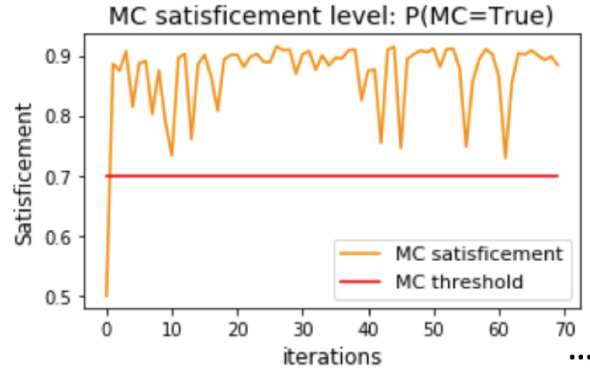


FIGURE C.2:  $DC_1$  - Minimization of Cost: satisficement level before the update of reward values  $R(s,a)$

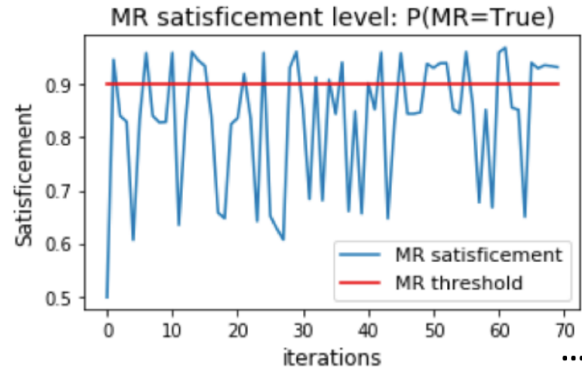


FIGURE C.3:  $DC_1$  - Maximization of Reliability: satisficement level before the update of reward values  $R(s,a)$

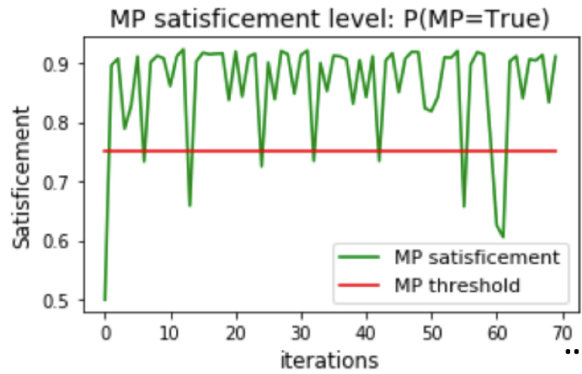


FIGURE C.4:  $DC_1$  - Maximization of Performance: satisficement level before the update of reward values  $R(s,a)$

Specifically, data packet loss are producing in the system more states  $MR_{t+1}=\text{False}$  when the MST topology is performed. For these states, the current reward values  $R(s,a)$  slightly favour the use of the RT topology. Therefore, in comparison to normal conditions, a slight increment on the use of the RT topology is observed

in Fig. C.1, but it is not enough to improve the reliability of the system under the dynamic context  $DC_1$ .

The reward values  $R(s,a)$  should be eventually reassessed and updated dynamically to assign higher importance to NFRs with poor satisficement levels (e.g. MR, the reliability of the system) and to improve the selection of the adaptation action  $a \in A$  in the online planning activity of RE-STORM.

- c) **Behaviour after reassessment and update of reward values  $R(s,a)$ .** The thresholds (i.e. SLAs) identified for each NFR are monitored. If the satisficement of any NFR is detected below its threshold; the reassessment and possible update of reward values  $R(s,a)$  is carried out by the *weights updater* module included in the planning phase of the MAPE-K Loop presented in section 4.4.1.

Examples of possible implementations for updating preferences about NFRs in a SAS are found in [86, 70, 67]. Our approach uses the ARRoW model (Automatic Runtime Reappraisal of Weights) [67] given its ability to update rewards values  $R(s,a)$  at runtime [38].

The updated reward values  $R(s,a)$  constitute an *additional input* to the *action planner* module in RE-STORM as it was depicted in Fig. 4.3. The latter will select the best self-adaptation action  $a \in A$  based on the additional information.

Figs. C.5, C.6 and C.7, show a sample of the new satisficement levels for the NFRs MC, MR and MP after updating the reward values  $R(s,a)$ . Given that the reliability of the system was below its satisficement threshold, a higher reward value was assigned by ARRoW to MR (i.e. stronger preference for MR).

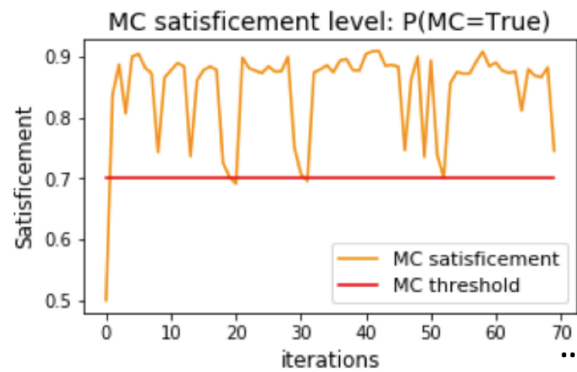


FIGURE C.5:  $DC_1$  - Minimization of Cost: satisficement level after the update of reward values  $R(s,a)$

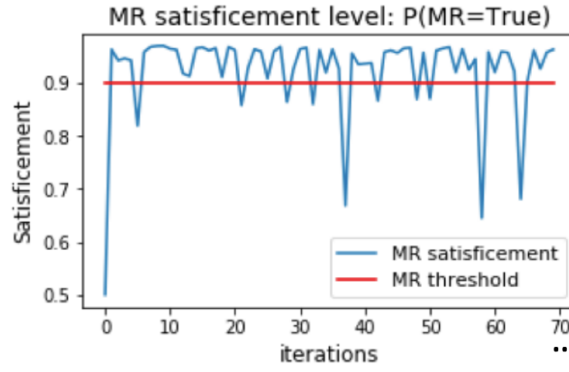


FIGURE C.6:  $DC_1$  - Maximization of Reliability: satisficement level after the update of reward values  $R(s,a)$

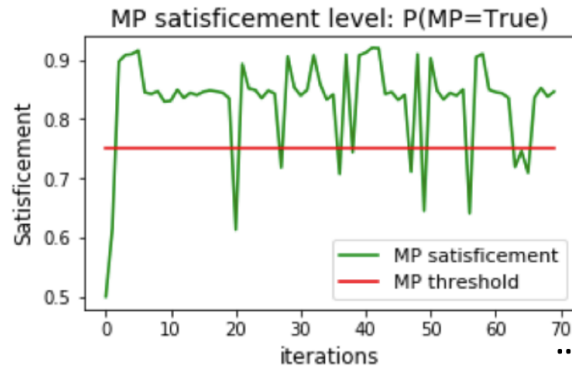


FIGURE C.7:  $DC_1$  - Maximization of Performance: satisficement level after the update of reward values  $R(s,a)$

The satisficement level of MR (the reliability of the system) improves as a result of the better informed decision-making provided by RE-STORM. Note in Fig. C.8, how after updating the reward values  $R(s,a)$ , the most preferred topology by the decision-making process is RT Topology.

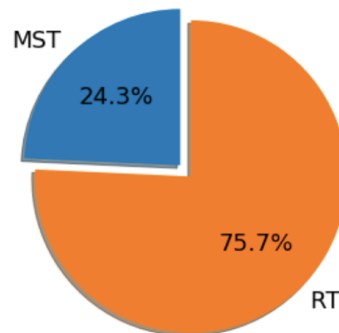


FIGURE C.8:  $DC_1$  - Chosen topology after the update of reward values  $R(s,a)$

It is also observed in Figs. C.5 and C.7, that the trade-off among NFRs provoques

a slight reduction in the satisficement levels of cost and performance in comparison to contexts where the reward values  $R(s,a)$  are not updated.

- d) **Average satisficement levels of NFRs before and after the update of reward values  $R(s,a)$ .** Figs. C.9, C.10 and C.11 show the average of the levels of satisficement of MC, MR and MP after 5 rounds during the first 1000+ time slices. According to the legend, given a round **[round-n]**, **na** indicates that ARRoW was not used, while **A** indicates that ARRoW was used to update weights at runtime.

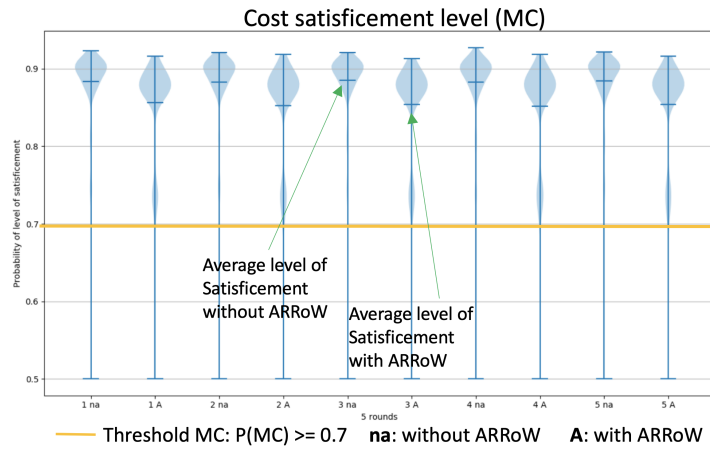


FIGURE C.9:  $DC_1$  - Average satisficement level for MC

The results in Fig. C.10 (specifically columns **[round-n] na**) show that during each round, and when the weights are not updated, the average satisficement level of reliability is below the required threshold, i.e. it is within a poor zone of satisficement due to unmatching initial reward values  $R(s,a)$  for the new dynamic context  $DC_1$ . Meanwhile, cost and performance are within a suitable zone of satisficement levels (See Figs. C.9 and C.11, columns **[round-n] na**).

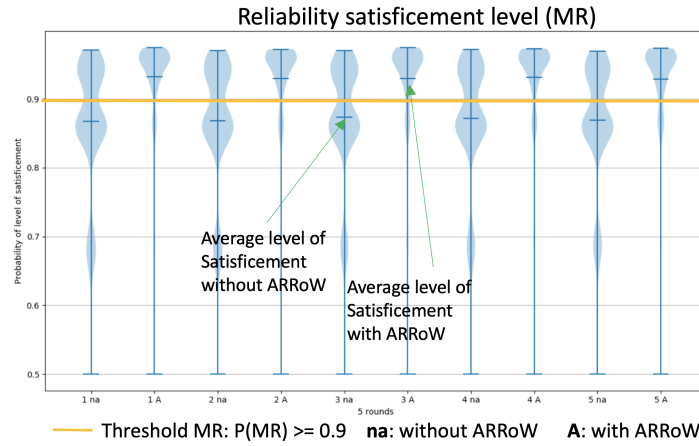


FIGURE C.10:  $DC_1$  - Average satisficement level for MR

On the other hand, when the weights are updated by ARRoW (See Fig. C.10, columns **[round-n] A**), the satisficement level of reliability is improved by leveraging access to new runtime evidence. As a trade-off, a reduction in the satisficement levels of cost and performance was also observed. However, still these levels were in a suitable zone (See Figs. C.9 and C.11, columns **[round-n] A**).

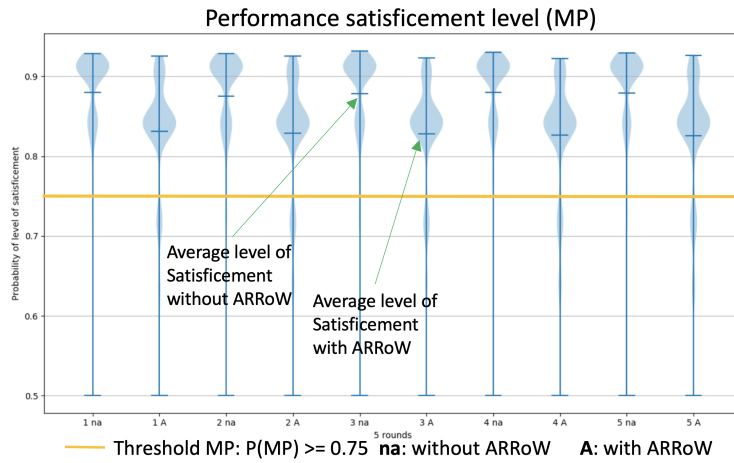


FIGURE C.11:  $DC_1$  - Average satisficement level for MP

In the dynamic context  $DC_1$ , the update of reward values  $R(s,a)$  contribute to improve the average satisficement level of the reliability in the RDM SAS. All the NFRs meet their SLAs.



**C.2 Dynamic context  $DC_2$ : changes in the environment during the execution of the RT topology are introduced to increment the cost and to reduce the performance of the system:  $P(MC_{t+1} = \text{True}, MP_{t+1} = \text{True} | NFR_t, RT_t)$ .**

- a) **Dynamic context  $DC_2$  description.** The description of this dynamic context has been stated in section 6.4.3: *“Unexpected data packet loss during the execution of the RT Topology, are generating an unusual rate of data forwarding, which would increase the bandwidth consumption (i.e. cost) and would reduce the system’s performance. In the RDM SAS, the cost for inter-site links communication is a function of the data sent over them. Therefore, a Redundant Topology (RT), which involves a bigger number of inter-site network links than a Minimum Spanning Tree Topology (MST), is more expensive. Costs increase as the number of network links increases and a reduction on the system’s performance<sup>1</sup> could also be expected.”*

Next, the behaviour of the RDM SAS under the execution of the dynamic context  $DC_2$  is depicted.

- b) **Behaviour before reassessment of reward values  $R(s,a)$ .** Given the new detected conditions in the dynamic context  $DC_2$ , and based on the initial RDM configuration, i.e the initial stakeholders’ preferences, it is observed that the MST topology is used even more than before (See Fig. C.12).

---

<sup>1</sup>The performance in these systems is measured as the total time to perform the write of data, which is the sum of the response times of the writes of each copy of data on each remote site [47].

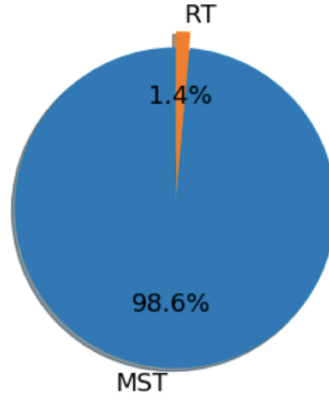


FIGURE C.12:  $DC_2$  - Chosen topology before the update of reward values  $R(s,a)$

Data packet loss and unusual data forwarding produce more state values  $MC_{t+1} = \text{False}$  and  $MP_{t+1} = \text{False}$ . For these states the current reward values  $R(s,a)$  favour the use of MST topology (See Table 6.2). Therefore, during the planning activity, MST topology is more selected by RE-STORM. This behaviour (i.e. the use of MST topology) allows to “avoid” the negative effects of the dynamic context  $DC_2$ , where the impact of RT topology, over the cost and performance of the system, is less favourable. By using more MST topology, the satisficement levels of cost and performance are improved, but with a slight reduction on the reliability of the system (See Figs. C.13, C.14 and C.15).

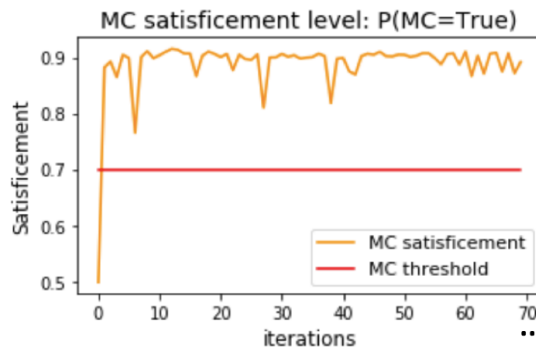


FIGURE C.13:  $DC_2$  - Minimization of Cost: satisficement level before the update of reward values  $R(s,a)$

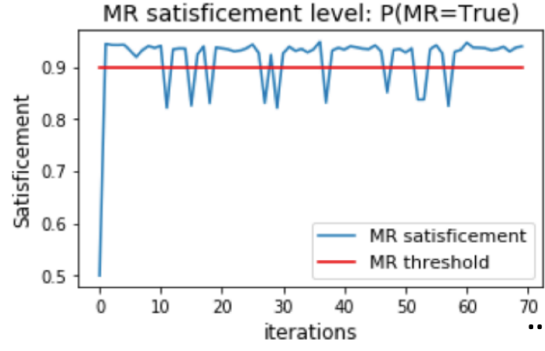


FIGURE C.14:  $DC_2$  - Maximization of Reliability: satisficement level before the update of reward values  $R(s,a)$

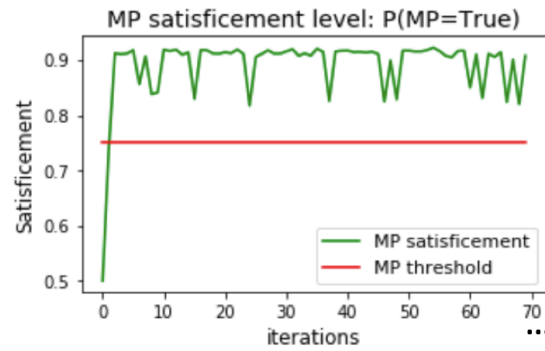


FIGURE C.15:  $DC_2$  - Maximization of Performance: satisficement level before the update of reward values  $R(s,a)$

c) **Behaviour after reassessment and update of reward values  $R(s,a)$ .** Figs. C.16, C.17 and C.18, show a sampled pattern of the new satisficement levels for the NFRs MC, MR and MP after updating the reward values  $R(s,a)$ .

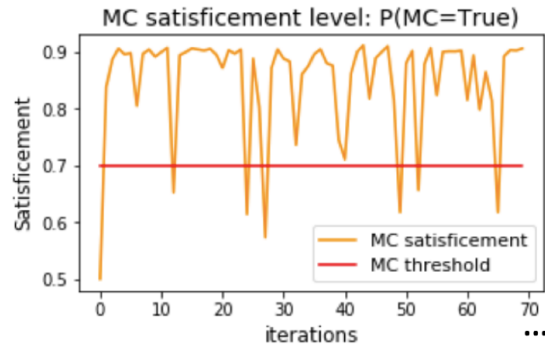


FIGURE C.16:  $DC_2$  - Minimization of Cost: satisficement level after the update of reward values  $R(s,a)$

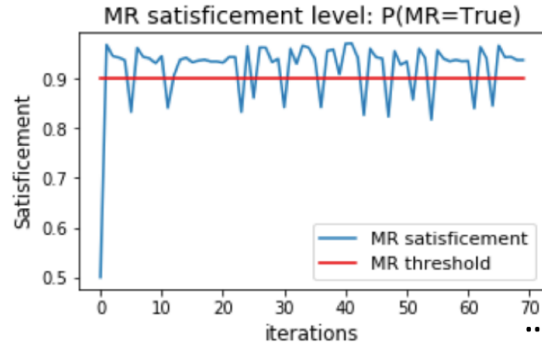


FIGURE C.17:  $DC_2$  - Maximization of Reliability: satisficement level after the update of reward values  $R(s,a)$

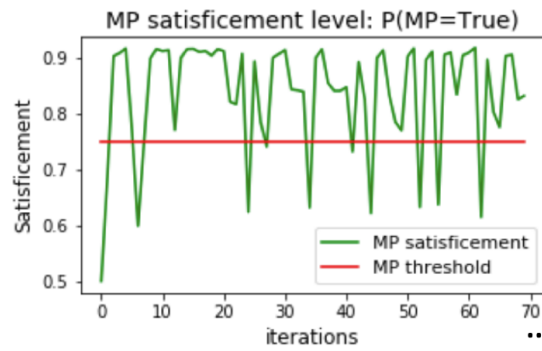


FIGURE C.18:  $DC_2$  - Maximization of Performance: satisficement level after the update of reward values  $R(s,a)$

It is observed in Figs. C.16, C.17 and C.18, that the reliability is slightly improved but a more considerable reduction on the cost and performance satisficement levels is also observed in comparisson to the contexts where reward values  $R(s,a)$  are not updated. After the update, RT topology is more used than before (See Fig. C.19).

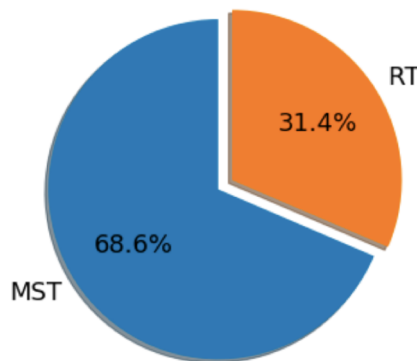


FIGURE C.19:  $DC_2$  - Chosen topology after the update of reward values  $R(s,a)$

- d) **Average satisficement levels of NFRs before and after the update of reward values  $R(s,a)$ .** Figs. C.20, C.21 and C.22 respectively show the average of the satisficement levels of MR, MC and MP after 5 rounds during the first 1000+ time slices.

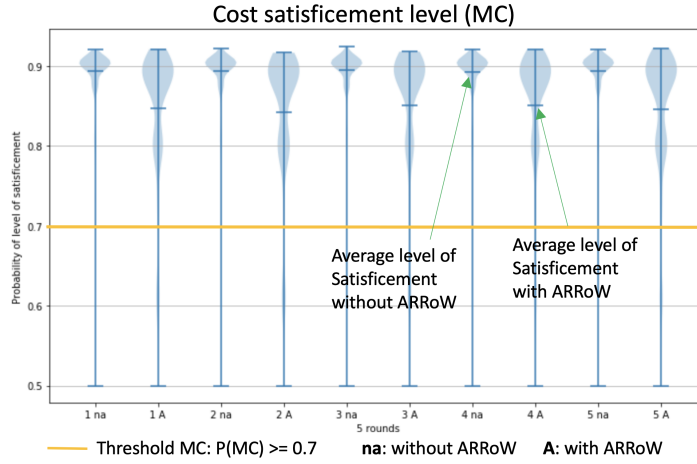


FIGURE C.20: DC<sub>2</sub> - Average satisficement level for MC

The results in Fig. C.21 (specifically columns [round-n] na) show that during each round, and when the weights are not updated by using ARRoW, the average satisficement level of the reliability of the system is slightly over the required threshold. Meanwhile, the cost and performance are within a suitable zone of satisficement level (See Figs. C.20 and C.22, columns [round-n] na).

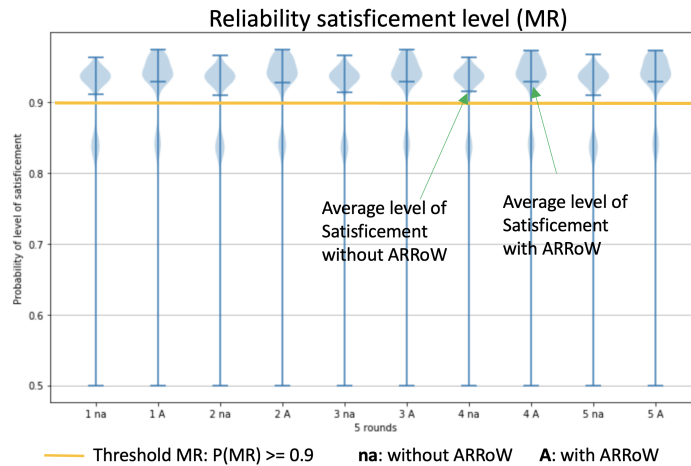


FIGURE C.21: DC<sub>2</sub> - Average satisficement level for MR

On the other hand, when the weights are updated, the satisficement level of reliability (i.e. MR) is slightly improved by leveraging access to new runtime evidence (See Fig. C.21, columns [round-n] A). As a trade-off, a reduction on

the satisficement levels of cost and performance is also observed but still within a suitable zone (See Figs. C.20 and C.22, columns [round-n] A).

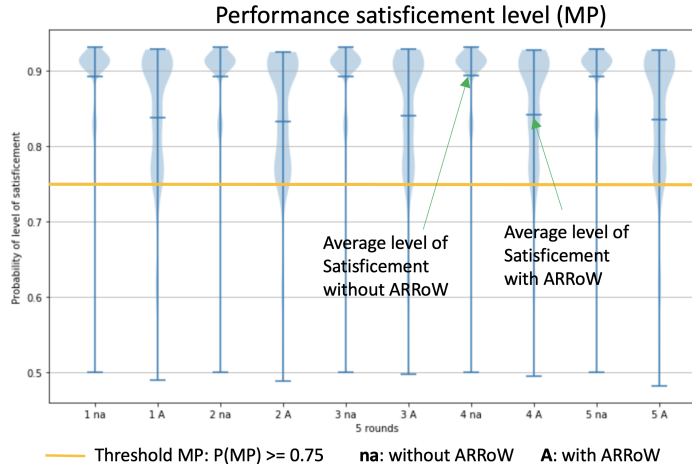


FIGURE C.22: DC<sub>2</sub> - Average satisficement level for MP

In the dynamic context DC<sub>2</sub>, the behaviour after updating the reward values  $R(s,a)$  represents an opportunity for further evaluation by system's stakeholders. There is a slight improvement on the average satisficement of the reliability of the system. However, this result could not be good enough to justify a rewards update, given the reduction on the satisficement levels of the performance and cost when the update is performed (even if they are still, on average, over their thresholds).

**C.3 Dynamic context DC<sub>3</sub>: changes in the environment during the execution of the topologies MST and RT are introduced to increment the cost and to reduce the reliability and the performance of the system:  $P(MR_{t+1} = \text{True} | \text{NFR}_t, \text{MST}_t)$  and  $P(MP_{t+1} = \text{True}, MC_{t+1} = \text{True} | \text{NFR}_t, \text{RT}_t)$**

- a) **Dynamic context DC<sub>3</sub> description.** The situations described in the dynamic contexts DC<sub>1</sub> and DC<sub>2</sub>.

Next, the behaviour of the RDM SAS under the execution of the dynamic context  $DC_3$  is depicted.

- b) **Behaviour before reassessment of reward values  $R(s,a)$ .** Given the new dynamic context  $DC_3$ , and based on the initial RDM configuration, MST topology is the only topology to be used (See Fig. C.23).

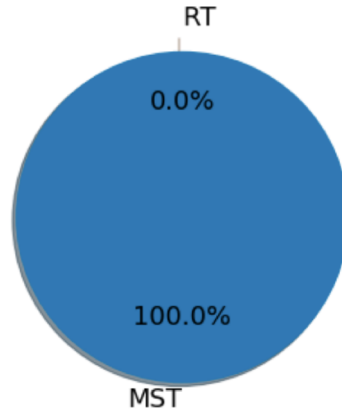


FIGURE C.23:  $DC_3$  - Chosen topology before the update of reward values  $R(s,a)$

Data packet loss and unusual rates of data forwarding favour the occurrence of (i) more state values  $MC_{t+1}=\text{False}$  and  $MP_{t+1}=\text{False}$  when  $RT_t$  topology is used, and (ii) more state values  $MR_{t+1}=\text{False}$  when  $MST_t$  topology is used. Under these states, the current reward values  $R(s,a)$  strongly favours the use of MST topology. Therefore, during the planning activity of RE-STORM, the only selected topology is MST. This behaviour improves the performance and cost of the system, but considerably reduce its reliability (See Figs. C.24, C.25 and C.26).

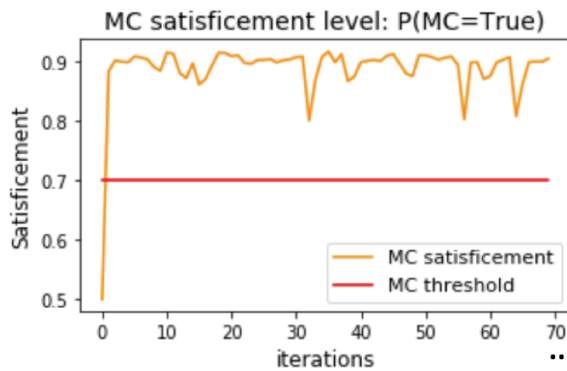


FIGURE C.24:  $DC_3$  - Minimization of Cost: satisfaction level before the update of reward values  $R(s,a)$

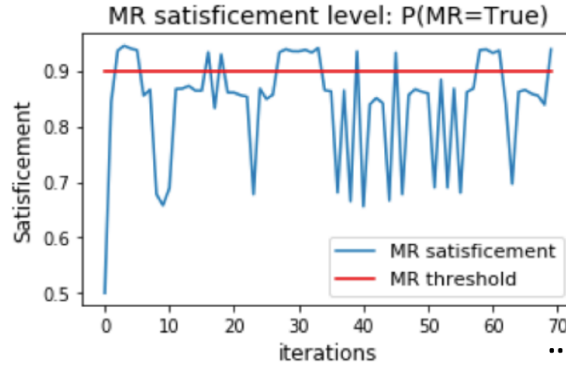


FIGURE C.25:  $DC_3$  - Maximization of Reliability: satisficement level before the update of reward values  $R(s,a)$

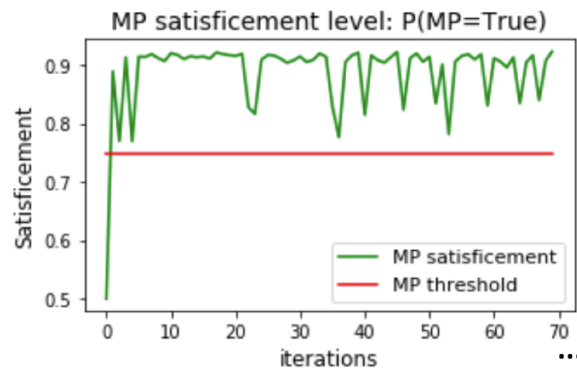


FIGURE C.26:  $DC_3$  - Maximization of Performance: satisficement level before the update of reward values  $R(s,a)$

c) **Behaviour after reassessment and update of reward values  $R(s,a)$ .** Figs. C.27, C.28 and C.29 show the satisficement levels of the NFRs after updating the reward values  $R(s,a)$ .

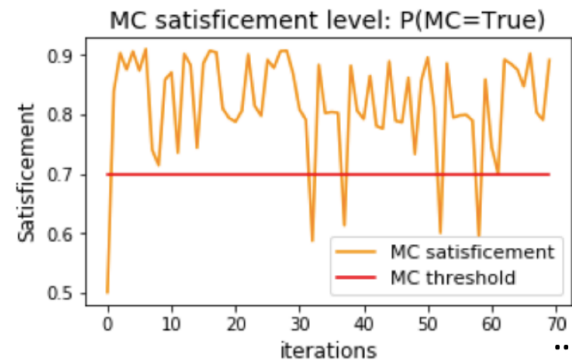


FIGURE C.27:  $DC_3$  - Minimization of Cost: satisficement level after the update of reward values  $R(s,a)$



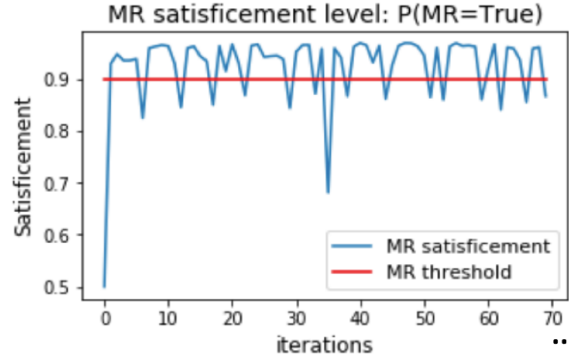


FIGURE C.28:  $DC_3$  - Maximization of Reliability: satisficement level after the update of reward values  $R(s,a)$

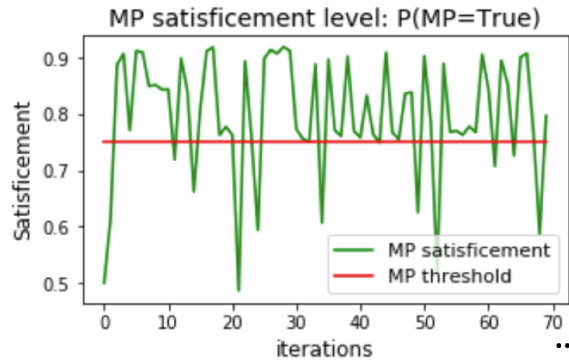


FIGURE C.29:  $DC_3$  - Maximization of Performance: satisficement level after the update of reward values  $R(s,a)$

A higher reward value was assigned to the reliability of the system (i.e. stronger preference for MR) given it was under its threshold. It is observed in Fig. C.28 that the reliability improves in comparison to the satisficement level shown in Fig. C.25. A reduction on the satisficement levels of cost and performance is also observed in comparisson to contexts where the reward values  $R(s,a)$  are not updated. Fig. C.30 also shows that after updating the reward values  $R(s,a)$ , the RT topology is used again during the system execution.

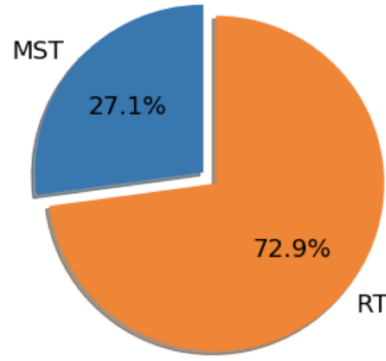


FIGURE C.30: DC<sub>3</sub> - Chosen topology after the update of reward values  $R(s,a)$

d) **Average satisficement levels of NFRs before and after the update of reward values  $R(s,a)$ .** Figs. C.32, C.31 and C.33 respectively show the average of the levels satisficement of MR, MC and MP after 5 rounds of execution during the first 1000+ time slices.

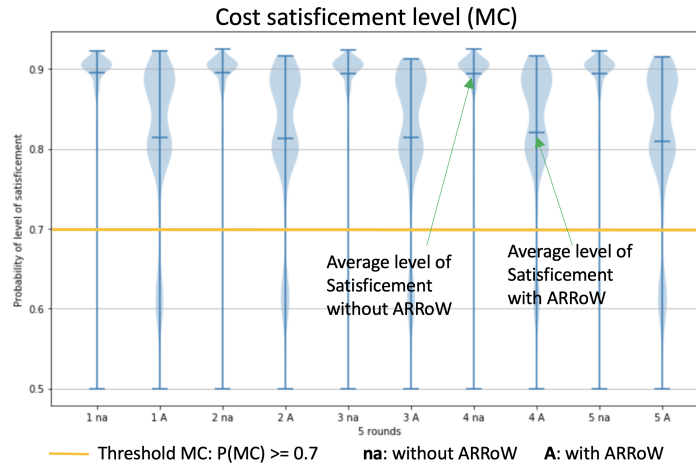


FIGURE C.31: DC<sub>3</sub> - Average satisficement level for MC

The results in Fig. C.32 (specifically columns **[round-n] na**) show that during each round, and when the weights are not updated, the average satisficement level of reliability is below the required threshold. Meanwhile, cost and performance are within a suitable zone of satisficement levels (See Figs. C.31 and C.33, columns **[round-n] na**).

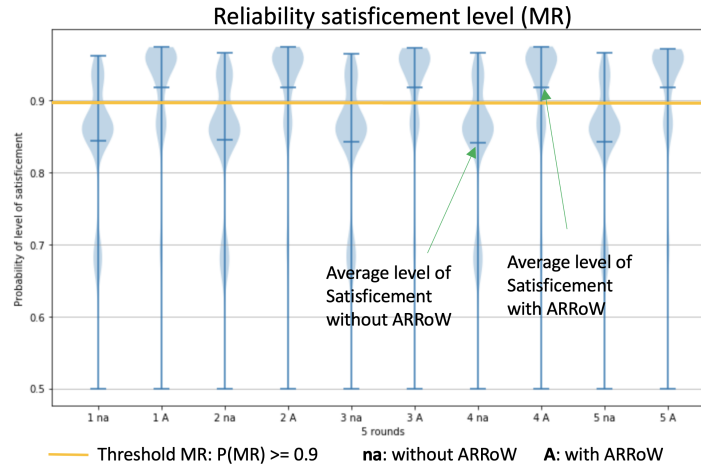


FIGURE C.32: DC<sub>3</sub> - Average satisficement level for MR

On the other hand, when the weights were updated (See Fig. C.32, columns [round-n] A), the satisficement level of reliability was improved. As a trade-off, a reduction in the satisficement level of cost and performance was observed, but in general still these levels were in a suitable zone (See Figs. C.31 and C.33, columns [round-n] A).

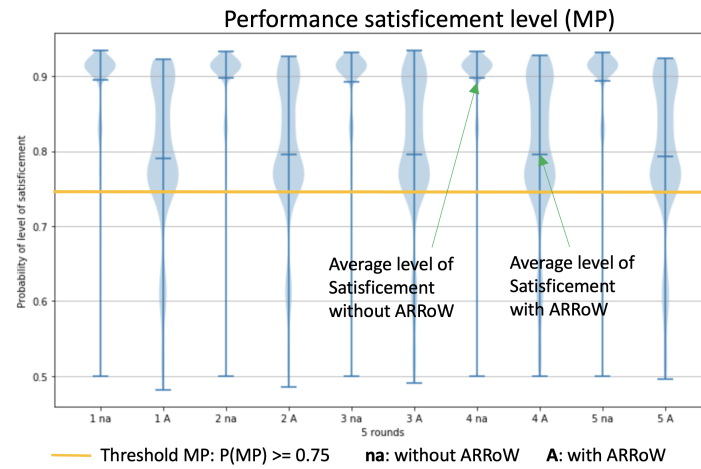


FIGURE C.33: DC<sub>3</sub> - Average satisficement level for MP

*In the dynamic context DC<sub>3</sub>, the update of reward values  $R(s,a)$  contributes to improve the reliability of the RDM SAS. As a trade-off there is some reduction on the satisficement levels of cost and performance but still on average over their thresholds.*

#### C.4 Dynamic context $DC_4$ : changes on the environment during the execution of the MST topology are introduced to increment the cost and to reduce the reliability and the performance of the system: $P(NFRs_{t+1} = \text{True} | NFR_t, MST_t)$

- a) **Dynamic context  $DC_4$  description.** Let us revisit the description of the dynamic context  $DC_4$ , which has been stated in section 6.4.3: “Unexpected data packet loss during the execution of the MST Topology is generating an unusual reduction on the reliability of the system (i.e.  $DC_1$  context behaviour), but also the increase in bandwidth consumption and the reduction of the system’s performance (NFRs MC and MP respectively).”

Next, the behaviour of the RDM SAS under the execution of the dynamic context  $DC_4$  is depicted.

- b) **Behaviour before reassessment of reward values  $R(s,a)$ .** Given the new dynamic context detected in part a), and based on the initial RDM configuration, it is observed that the RT topology is more used than before (See Fig. C.34).

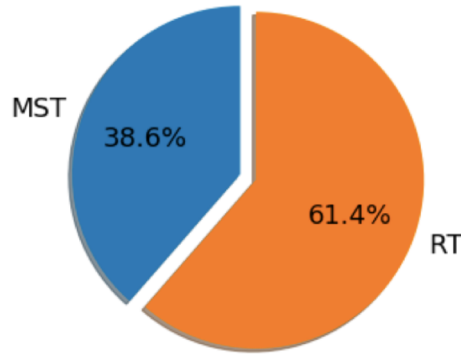


FIGURE C.34:  $DC_4$  - Chosen topology before the update of reward values  $R(s,a)$

Data packet loss favours the occurrence of more state values  $MC_{t+1} = \text{False}$ ,  $MR_{t+1} = \text{False}$  and  $MP_{t+1} = \text{False}$  when  $MST_t$  topology is used. Under these states, the current reward values  $R(s,a)$  favour the use of RT topology. Therefore, during the planning activity after applying Equation (4.8) in RE-STORM,

the preferred topology is RT. This behaviour slightly reduces the performance and cost, and moderately improves the reliability of the system (See Figs. C.35, C.36 and C.37).

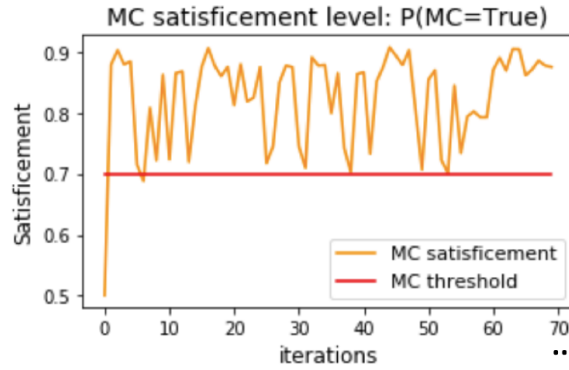


FIGURE C.35:  $DC_4$  - Minimization of Cost: satisficement level before the update of reward values  $R(s,a)$

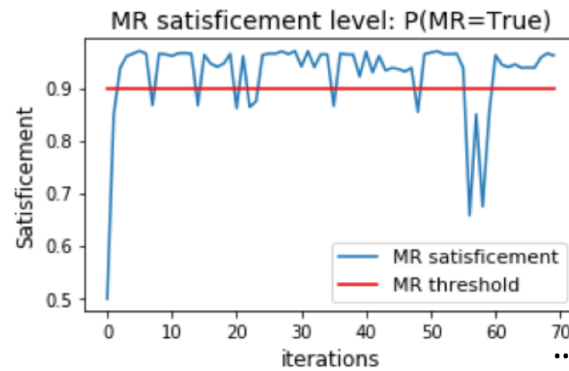


FIGURE C.36:  $DC_4$  - Maximization of Reliability: satisficement level before the update of reward values  $R(s,a)$

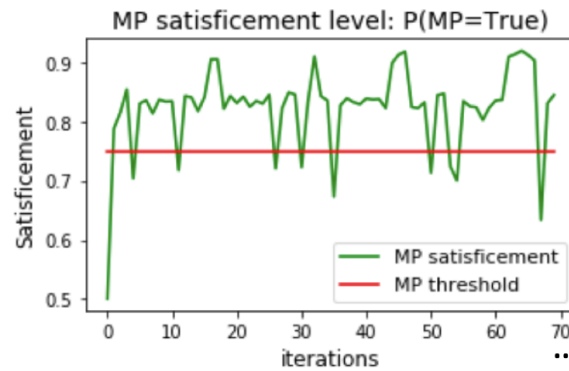


FIGURE C.37:  $DC_4$  - Maximization of Performance: satisficement level before the update of reward values  $R(s,a)$

c) **Behaviour after reassessment and update of reward values  $R(s,a)$ .** Figs. C.38, C.39 and C.40, show the satisficement levels of the NFRs in the RDM SAS after updating the reward values  $R(s,a)$ .

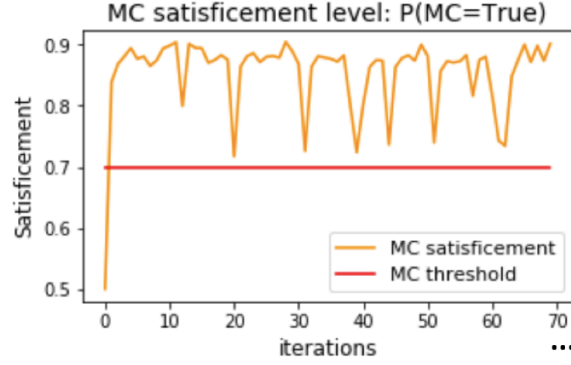


FIGURE C.38:  $DC_4$  - Minimization of Cost: satisficement level after the update of reward values  $R(s,a)$

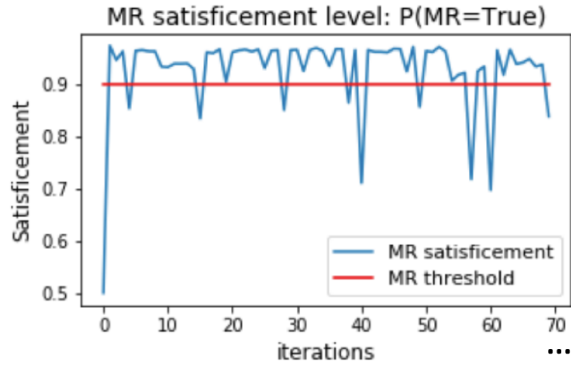


FIGURE C.39:  $DC_4$  - Maximization of Reliability: satisficement level after the update of reward values  $R(s,a)$



FIGURE C.40:  $DC_4$  - Maximization of Performance: satisficement level after the update of reward values  $R(s,a)$

It is observed in Figs. C.38, C.39 and C.40, that the satisficement level of the reliability of the system slightly improves. A slight reduction on the satisficement

of performance is also observed in comparison to contexts where the reward values  $R(s,a)$  are not updated. After updating them, the RT Topology is more used than before (See Fig. C.41).

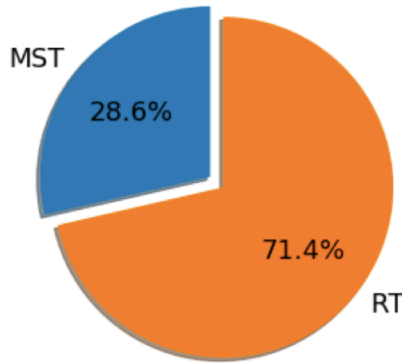


FIGURE C.41:  $DC_4$  - Chosen topology after the update of reward values  $R(s,a)$

d) **Average satisficement levels of NFRs before and after the update of reward values  $R(s,a)$ .** Figs. C.43, C.42 and C.44 respectively show the average of the levels satisficement of MR, MC and MP after 5 rounds of execution during the first 1000+ time slices.

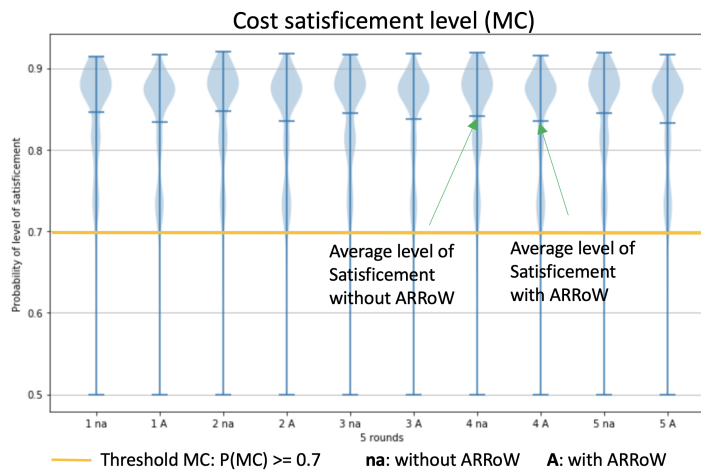


FIGURE C.42:  $DC_4$  - Average satisficement level for MC

The results in Fig. C.43 (specifically columns **[round-n] na**) show that during each round, and when the weights are not updated, the average satisficement level of the reliability of the system is already over its threshold. Meanwhile, the cost and performance are within a suitable zone of satisficement levels (See Figs. C.42 and C.44, columns **[round-n] na**).

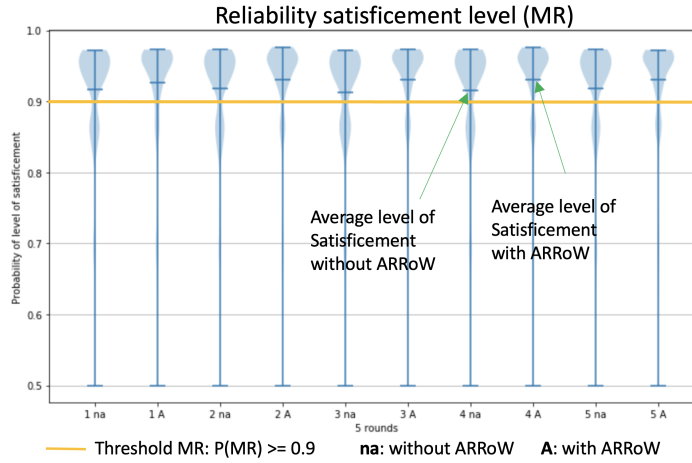


FIGURE C.43:  $DC_4$  - Average satisficement level for MR

On the other hand, when the weights were updated (See Fig. C.43, columns [round-n] A), the satisficement level of reliability was slightly improved. As a trade-off, a reduction in the satisficement levels of cost and performance was also observed but still with satisficement levels within a suitable zone (See Figs. C.42 and C.44, columns [round-n] A).

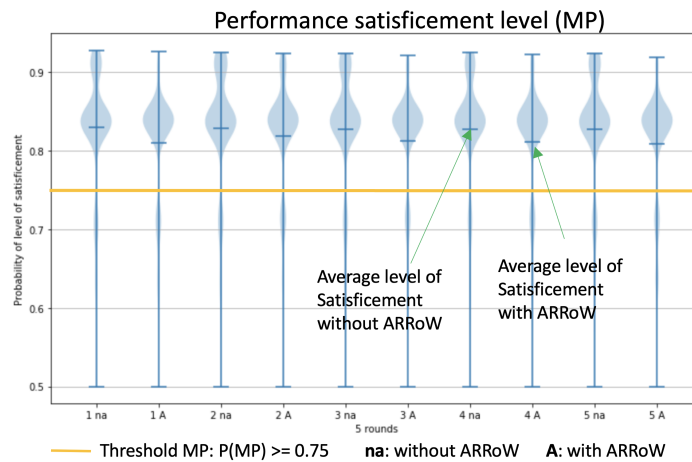


FIGURE C.44:  $DC_4$  - Average satisficement level for MP

In the dynamic context  $DC_4$ , the satisficement level of the NFRs is always over the thresholds (before and after the update of reward values  $R(s,a)$ ). Therefore, similar to  $DC_2$ , it may not be required to update them. A further evaluation by system's stakeholders may be needed to determine its suitability.



### C.5 Dynamic context DC<sub>5</sub>: changes on the environment during the execution of the RT topology are introduced to increment the cost and to reduce the reliability and the performance of the system: $P(NFR_{t+1} = \text{True} | NFR_t, RT_t)$

- a) **Dynamic context DC<sub>5</sub> description.** The description of this dynamic context has been stated in section 6.4.3: “Unexpected data packet loss during the execution of the RT Topology is generating an increment in bandwidth consumption and reduction of the system performance (i.e. DC<sub>2</sub> context behaviour), but also a reduction on the positive impact of the RT topology over the reliability of the system.”

Next, the behaviour of the RDM SAS under the execution of the dynamic context DC<sub>5</sub> is depicted.

- b) **Behaviour before reassessment of reward values  $R(s,a)$ .** Given the new dynamic context detected, and based on the initial RDM configuration, the MST topology is the only one to be used (See Fig. C.45).

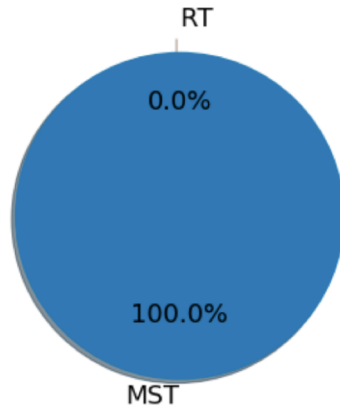


FIGURE C.45: DC<sub>5</sub> - Chosen topology before the update of reward values  $R(s,a)$

Data packet loss favours the occurrence of more state values  $MC_{t+1} = \text{False}$ ,  $MR_{t+1} = \text{False}$  and  $MP_{t+1} = \text{False}$  when  $RT_t$  topology is used. For these states the current reward values  $R(s,a)$  favour the use of the topology MST. Therefore, during the planning activity of RE-STORM, the selected topology is MST. As a

result of this behaviour the reliability of the system is below its threshold during several timeslices (See Figs. C.46, C.47 and C.48). Meanwhile, the cost and performance are on a suitable zone of satisficement, i.e. over their thresholds.

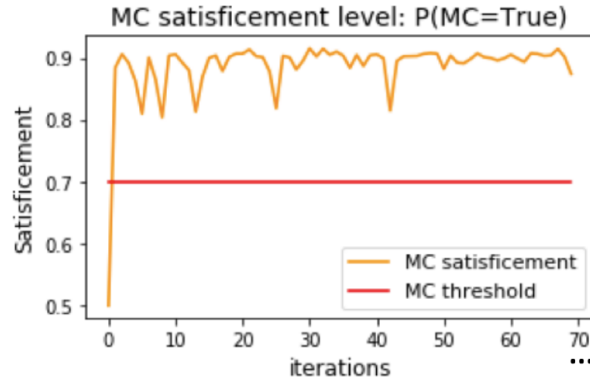


FIGURE C.46: DC<sub>5</sub> - Minimization of Cost: satisficement level before the update of reward values  $R(s,a)$

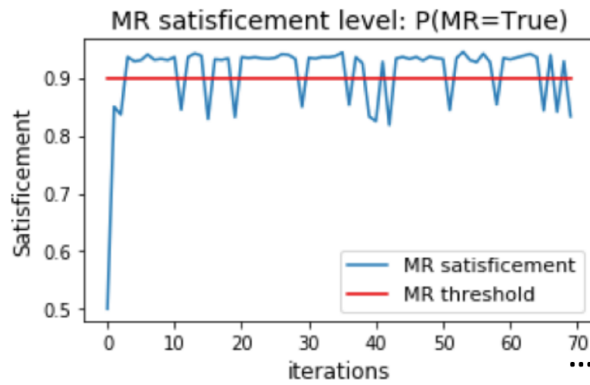


FIGURE C.47: DC<sub>5</sub> - Maximization of Reliability: satisficement level before the update of reward values  $R(s,a)$

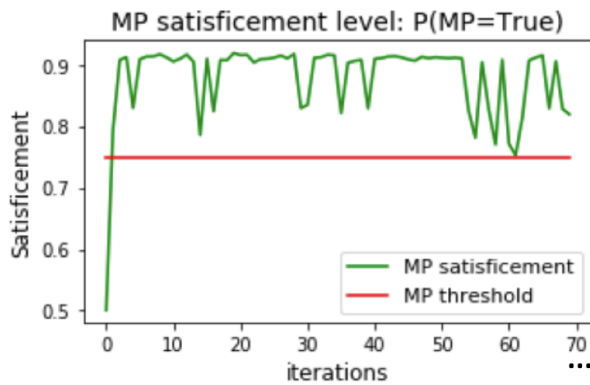


FIGURE C.48: DC<sub>5</sub> - Maximization of Performance: satisficement level before the update of reward values  $R(s,a)$

c) **Behaviour after reassessment and update of reward values  $R(s,a)$ .** Figs. C.49, C.50 and C.51, show the satisficement levels of the NFRs in the RDM SAS after updating the reward values  $R(s,a)$ .

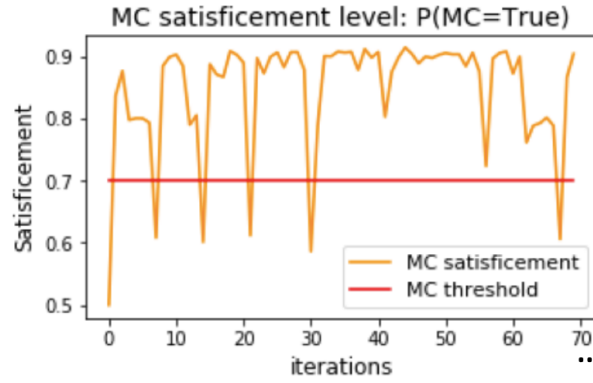


FIGURE C.49:  $DC_5$  - Minimization of Cost: satisficement level after the update of reward values  $R(s,a)$

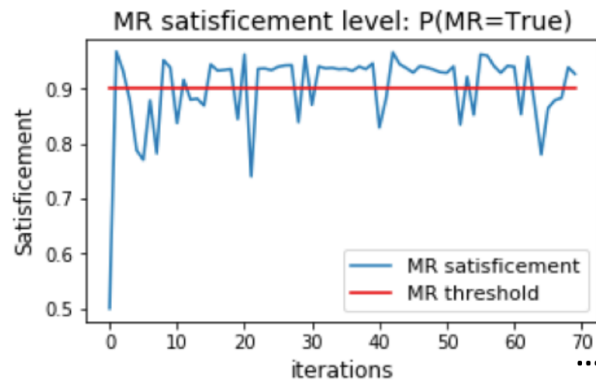


FIGURE C.50:  $DC_5$  - Maximization of Reliability: satisficement level after the update of reward values  $R(s,a)$

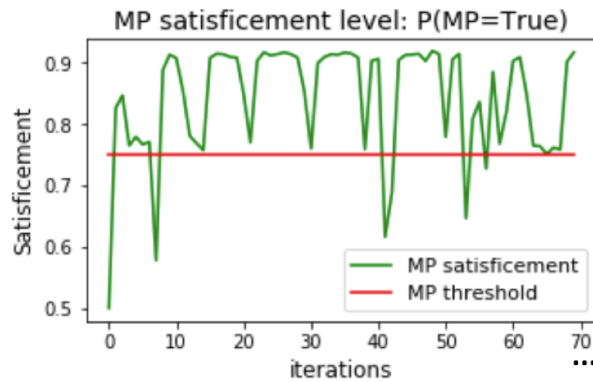


FIGURE C.51:  $DC_5$  - Maximization of Performance: satisficement level after the update of reward values  $R(s,a)$

Under the current context a higher reward value was assigned to the reliability of the system (i.e. stronger preference for MR). However, it is observed in Fig. C.50 that the satisficement level of reliability is *slightly reduced*. A slight reduction on the satisficement of cost and performance is also observed in Figs. C.50 and C.51, in comparisson to contexts where rewards  $R(s,a)$  are not updated. It is also observed that after updating the reward values  $R(s,a)$  the RT topology is used again (See Fig. C.52).

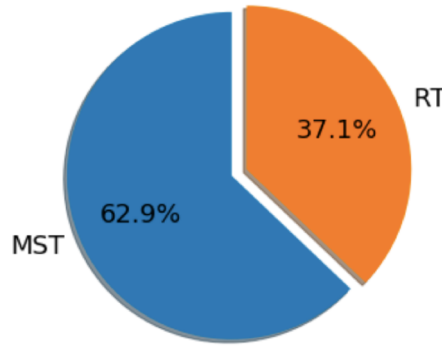


FIGURE C.52: DC<sub>5</sub> - Chosen topology after the update of reward values  $R(s,a)$

- d) **Average satisficement levels of NFRs before and after the update of reward values  $R(s,a)$ .** Figs. C.54, C.53 and C.55 respectively show the average of the levels satisficement of MR, MC and MP after 5 rounds of execution during the first 1000+ time slices.

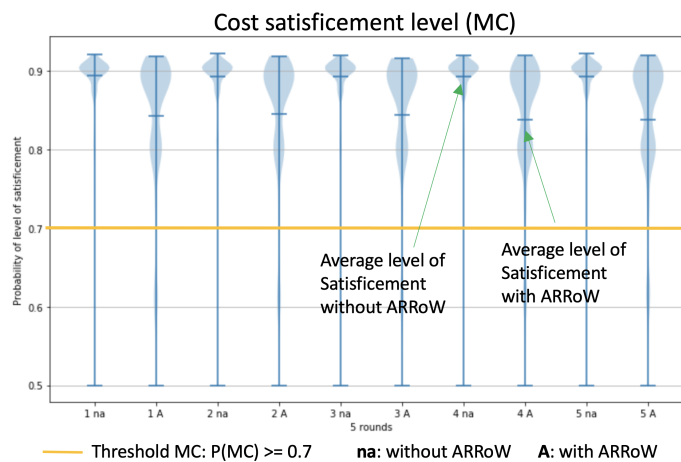


FIGURE C.53: DC<sub>5</sub> - Average satisficement level for MC

The results in Fig. C.54 (specifically columns [round-n] A) show that during

each round, when the reward values  $R(s,a)$  are updated to improve the reliability of the system, the effect is *the opposite*: a slight reduction on the average satisficement of the reliability of the system (but still over its threshold). Meanwhile, the cost and performance, show also a reduction on their average satisficement, but still within a suitable zone (See Figs. C.53 and C.55).

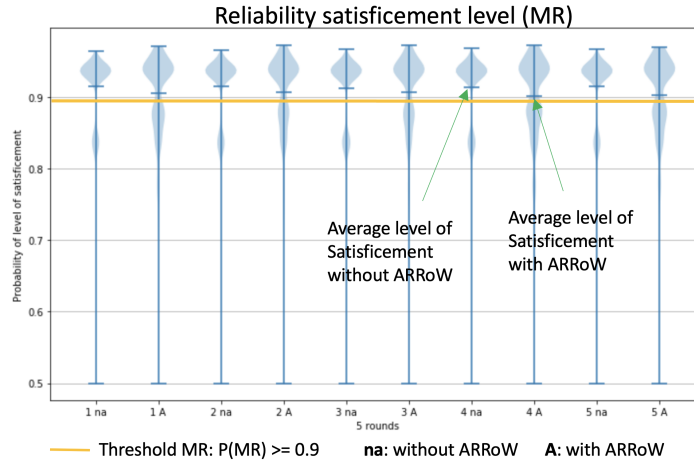


FIGURE C.54: DC<sub>5</sub> - Average satisficement level for MR

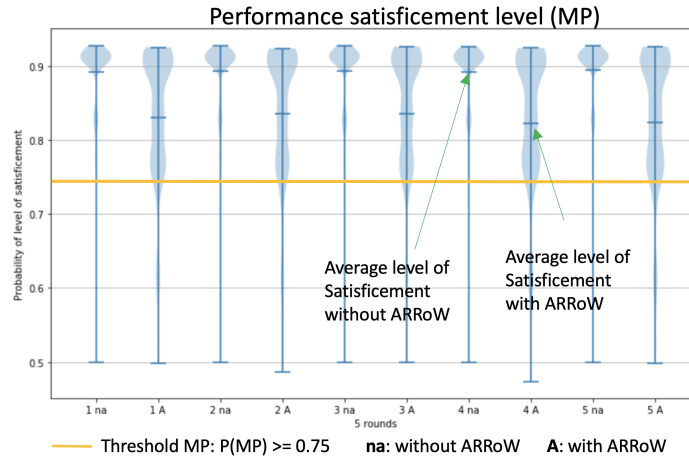


FIGURE C.55: DC<sub>5</sub> - Average satisficement level for MP

*In the dynamic context DC<sub>5</sub>, the positive impact of RT topology over the reliability of the system has been lost. Therefore, when the reward values  $R(s,a)$  are updated to assign more preference to reliability, when RT topology is performed, the result is counterproductive, a lower reliability. Under this context, when  $RT_t$  topology is used, states  $MR_{t+1}=False$  are more feasible than before.*

**C.6 Dynamic context  $DC_6$ : changes on the environment during the execution of the topologies MST and RT are introduced to increment the cost and to reduce the reliability and the performance of the system:  $P(NFR_{t+1} = \text{True} | NFR_t, MST_t)$  and  $P(NFR_{t+1} = \text{True} | NFR_t, RT_t)$**

- a) **Dynamic context  $DC_6$  description.** The situations described in the dynamic contexts  $DC_4$  and  $DC_5$ .

Next, the behaviour of the RDM SAS under the execution of the dynamic context  $DC_6$  is depicted.

- b) **Behaviour before reassessment of reward values  $R(s,a)$ .** Given the new detected context, and based on the initial RDM configuration, it is observed that the MST topology is the only one to be used (See Fig. C.56).

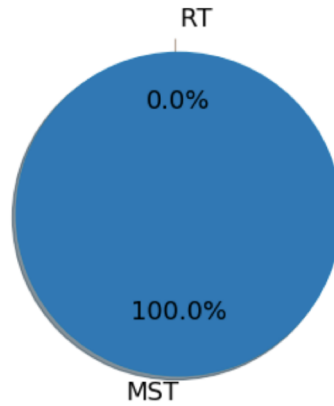


FIGURE C.56:  $DC_6$  - Chosen topology before the update of reward values  $R(s,a)$

Data packet loss are favouring the reduction of the satisficement level of the cost, reliability and performance regardless of the topology to be used. This reduction is observed in Figs. C.57, C.58 and C.59. In  $DC_6$ , either we use MST or RT topology, we obtain more state values  $MC_{t+1}=\text{False}$ ,  $MR_{t+1}=\text{False}$  and  $MP_{t+1}=\text{False}$ . For these states, the current reward values  $R(s,a)$  strongly favour the use of the

topology MST. Therefore, during the planning activity and after applying Equation (4.8), the selected topology is MST as it is observed in Fig. C.56.

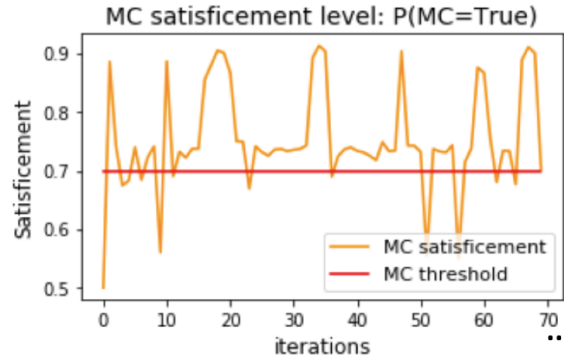


FIGURE C.57:  $DC_6$  - Minimization of Cost: satisficement level before the update of reward values  $R(s,a)$

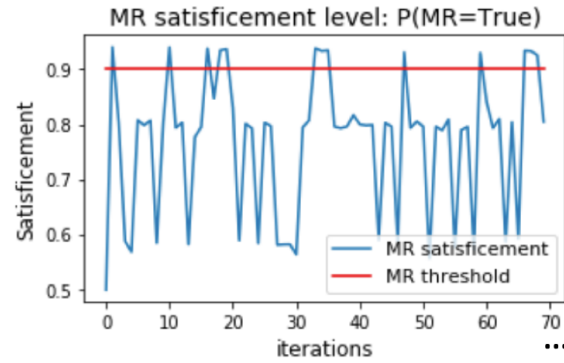


FIGURE C.58:  $DC_6$  - Maximization of Reliability: satisficement level before the update of reward values  $R(s,a)$

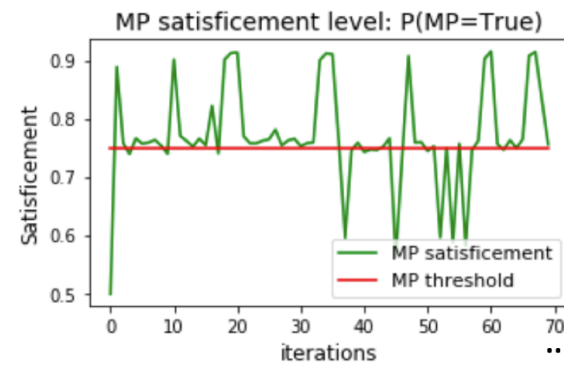


FIGURE C.59:  $DC_6$  - Maximization of Performance: satisficement level before the update of reward values  $R(s,a)$

- c) **Behaviour after reassessment and update of reward values  $R(s,a)$ .** Figs. C.60, C.61 and C.62, show a sampled pattern of the new satisficement levels for the NFRs MC, MR and MP after updating the reward values  $R(s,a)$ .

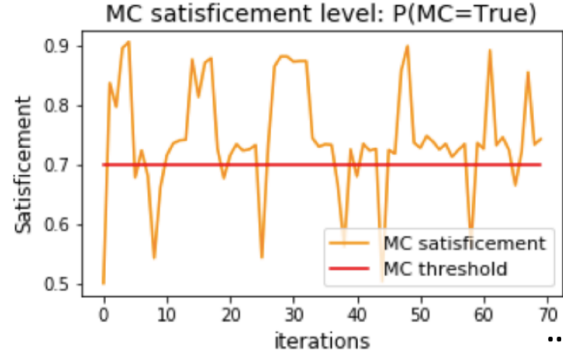


FIGURE C.60:  $DC_6$  - Minimization of Cost: satisficement level after the update of reward values  $R(s,a)$

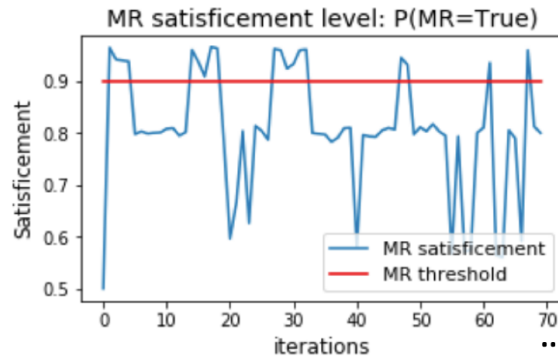


FIGURE C.61:  $DC_6$  - Maximization of Reliability: satisficement level after the update of reward values  $R(s,a)$

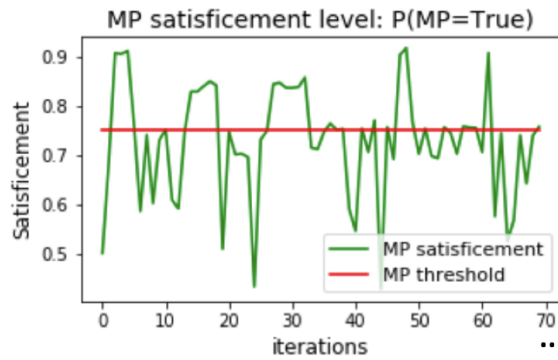


FIGURE C.62:  $DC_6$  - Maximization of Performance: satisficement level after the update of reward values  $R(s,a)$

It is observed in Fig. C.61 that the reliability is slightly incremented. A reduction on cost and performance is also observed in comparisson to contexts where rewards  $R(s,a)$  are not updated (See Figs. C.60 and C.62). After updating the rewards  $R(s,a)$ , the RT topology is used again (See Fig. C.63).



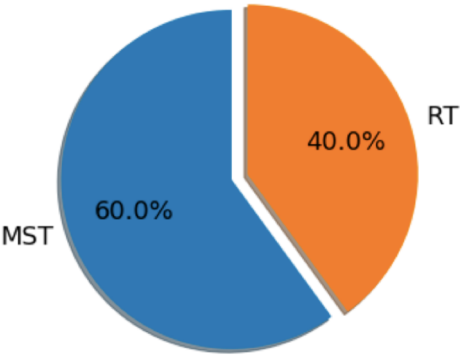


FIGURE C.63:  $DC_6$  - Chosen topology after the update of reward values  $R(s,a)$

d) **Average satisficement levels of NFRs before and after the update of reward values  $R(s,a)$ .** Figs. C.65, C.64 and C.66 respectively show the average of the levels satisficement of MR, MC and MP after 5 rounds during the first 1000+ time slices.

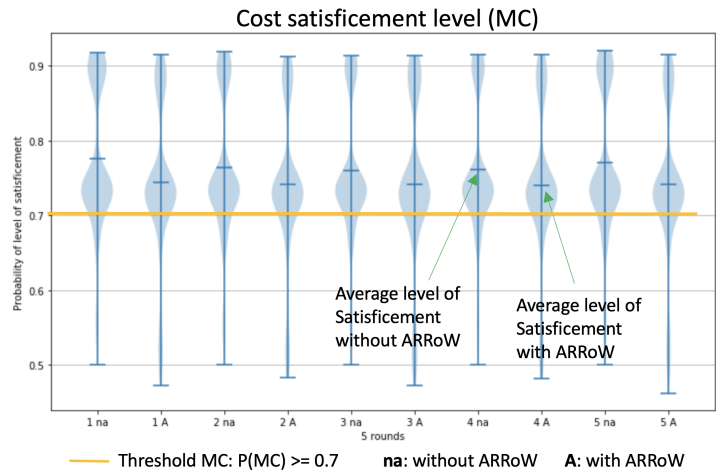


FIGURE C.64:  $DC_6$  - Average satisficement level for MC

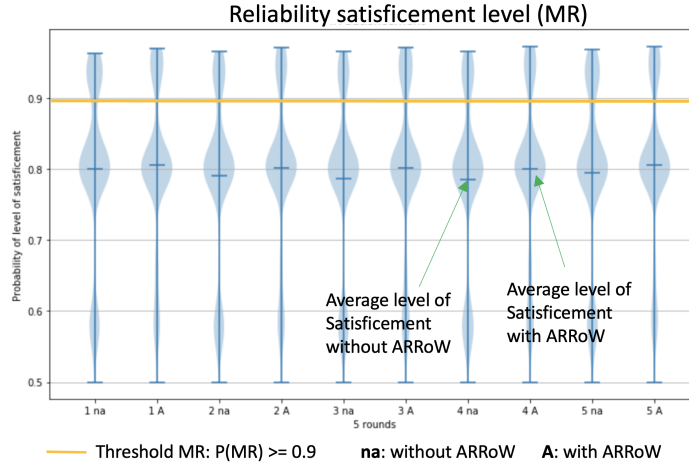


FIGURE C.65: DC<sub>6</sub> - Average satisficement level for MR

The results in Fig. C.65 (specifically columns [round-n] A) show that during each round, a slight increment on the reliability is obtained when the reward values  $R(s,a)$  are updated. However, the average satisficement is still below its threshold. Meanwhile, the cost and performance experiment a reduction on their satisficement levels (See Figs. C.64 and C.66).

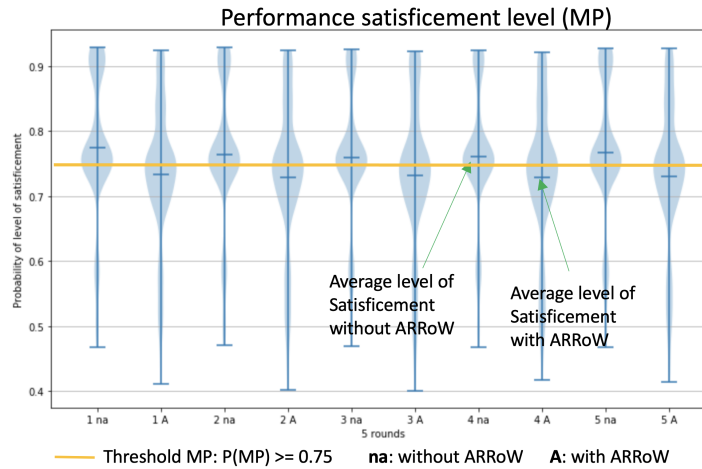


FIGURE C.66: DC<sub>6</sub> - Average satisficement level for MP

In the dynamic context DC<sub>6</sub>, the trade-off of NFRs is highlighted. Specifically, an increment on reliability and simultaneously a decrement on the satisficement levels of cost and performance is observed. However, under the current environmental conditions regardless the topology in use, it is not feasible to improve the reliability of the system to meet its SLA.

## Appendix D

# RDM SAS under different Service Level Agreements (SLAs)

In section 6.6, how the RDM SAS would behave using different SLAs has been reported. In this section, more details on the implemented scenarios to demonstrate the RDM SAS behaviour are presented.

### D.1 SLAs: less strict scenario

Under this scenario, the satisficement levels of the NFRs would always meet their SLAs as shown in Figs. D.1, D.2, D.3 and therefore ARRoW would not be called.

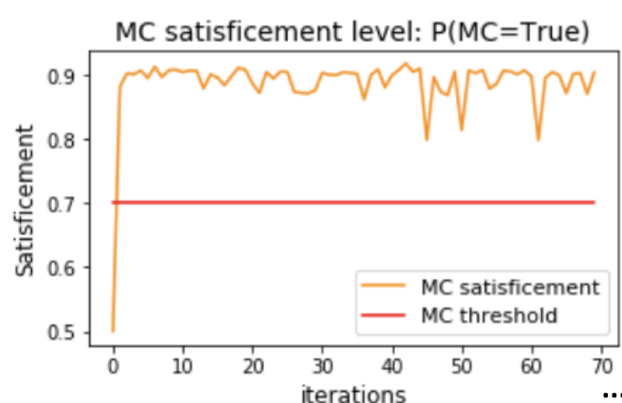


FIGURE D.1: Less strict scenario - Minimization of Cost:  $P(\text{MC} = \text{True}) \geq 0.7$

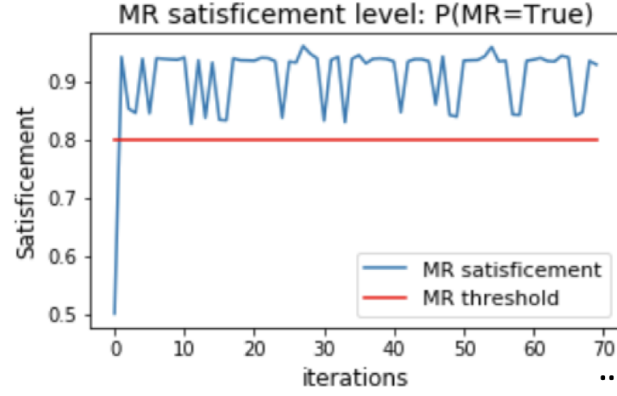


FIGURE D.2: Less strict scenario - Maximization of Reliability:  $P(MR = \text{True}) \geq 0.8$

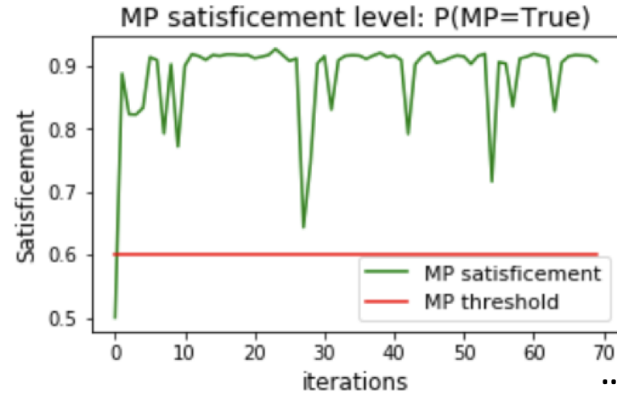


FIGURE D.3: Less strict scenario - Maximization of Performance: SLA  $P(MP = \text{True}) \geq 0.6$

## D.2 SLAs: stricter scenario

Under a stricter scenario, the thresholds for the satisficement level of the NFRs are incremented. In the following two examples the behaviour of ARRoW under a stricter scenario is described.

- a) **SLAs example 01:** In this example, the required satisficement level for the cost, reliability and performance of the system has been incremented. The new established SLAs are  $P(MC = \text{True}) \geq 0.80$ ,  $P(MR = \text{True}) \geq 0.95$  and  $P(MP = \text{True}) \geq 0.85$ .

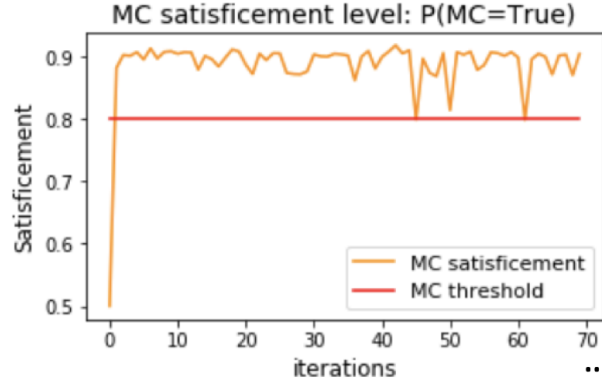


FIGURE D.4: SLAs example 01 - Minimization of Cost:  $P(\text{MC} = \text{True}) \geq 0.8$

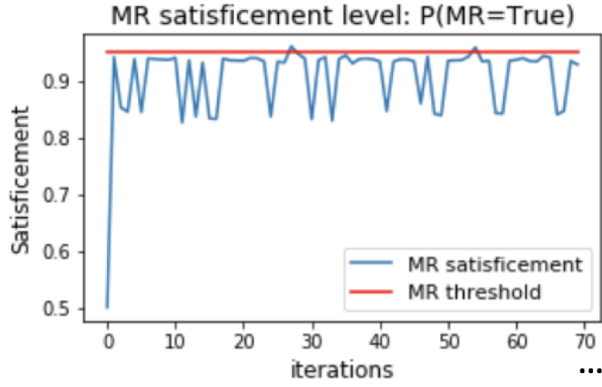


FIGURE D.5: SLAs example 01 - Maximization of Reliability:  $P(\text{MR} = \text{True}) \geq 0.95$

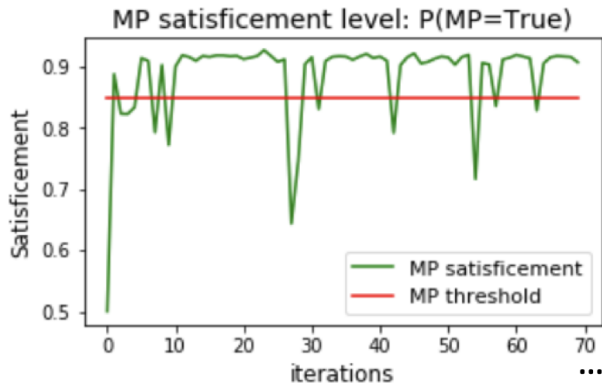


FIGURE D.6: SLAs example 01 - Maximization of Performance:  $P(\text{MP} = \text{True}) \geq 0.85$

It is observed in Figs. D.4, D.5 and D.6, that from the three NFRs under evaluation, the reliability of the system has the lowest satisficement level in relation to its SLA. Therefore, the reward values  $R(s,a)$  will be updated by ARRoW accordingly to this context. The decision-making provided by RE-STORM, taking into

account the new preferences, will improve the reliability but with a reduction on the cost and the performance of the system due to the trade-off performed. Figs. D.7, D.8 and D.9 show the new satisficement level of the NFRs after the update of the reward values  $R(s,a)$ .

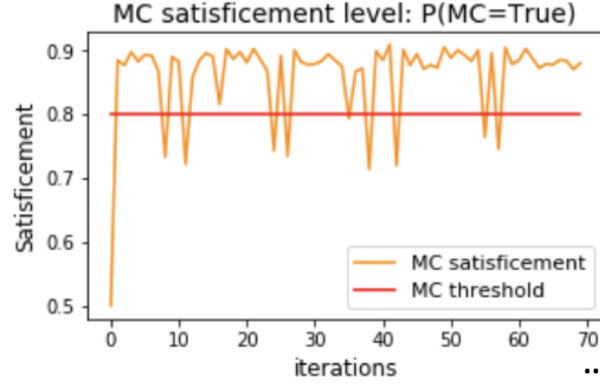


FIGURE D.7: SLAs example 01 - Minimization of Cost: satisficement level after the update of reward values  $R(s,a)$

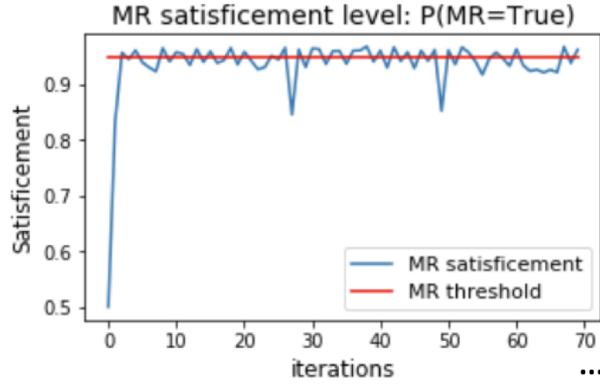


FIGURE D.8: SLAs example 01 - Maximization of Reliability: satisficement level after the update of reward values  $R(s,a)$

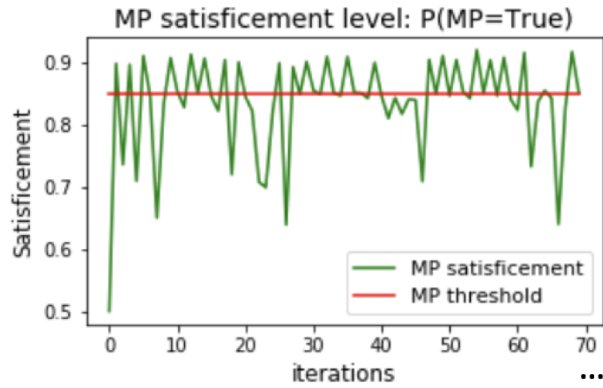


FIGURE D.9: SLAs example 01 - Maximization of Performance: satisficement level after the update of reward values  $R(s,a)$

The average satisficement level for the NFRs shown in Figs. D.7, D.8 and D.9, is  $MC = 0.8608$ ,  $MR = 0.9373$  and  $MP = 0.8354$ . Although, it is possible to observe in Fig. D.8 that the satisficement level of the reliability is *during some time slices* over its threshold, on average, it was not possible to meet its SLA.

*RE-STORM and ARRoW always improve the NFR with the lowest satisficement level with respect to its SLA. However, unusually high SLAs may be not reached.*

- b) **SLAs example 02:** Different from the previous example, an even higher satisficement level for the reliability of the system is required in this example. The new established SLAs are:  $P(MC = \text{True}) \geq 0.80$ ,  $P(MR = \text{True}) \geq 0.99$  and  $P(MP = \text{True}) \geq 0.85$ . Figs. D.10, D.11 and D.12, show the behaviour of the RDM SAS before the update of the reward values  $R(s,a)$ .

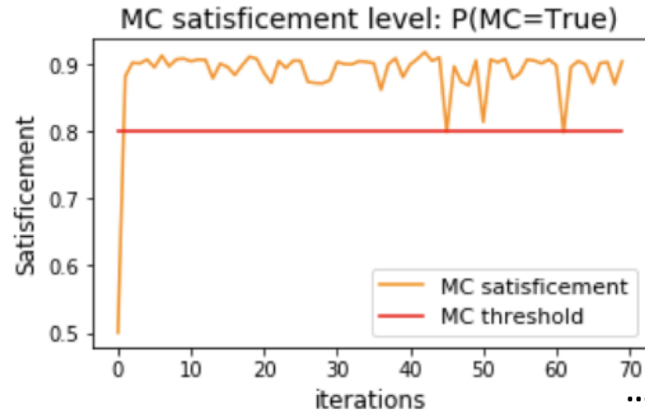


FIGURE D.10: SLAs example 02 - Minimization of Cost:  $P(MC = \text{True}) \geq 0.8$

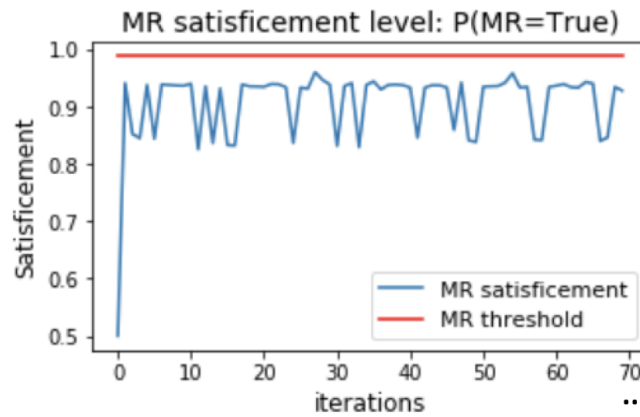


FIGURE D.11: SLAs example 02 - Maximization of Reliability:  $P(MR = \text{True}) \geq 0.99$

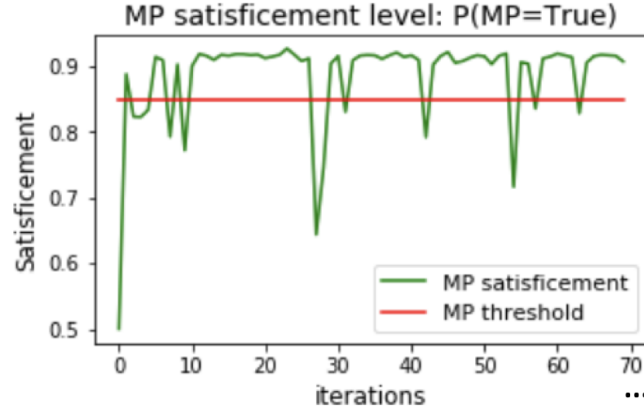


FIGURE D.12: SLAs example 02 - Maximization of Performance:  
 $P(MP = \text{True}) \geq 0.85$

In Fig. D.11, it is observed that the satisficement level of the reliability is always below its SLA. The reward values  $R(s,a)$  are updated by ARRoW accordingly to this context, i.e. a higher importance than the allocated in Example 01 will be assigned to the reliability of the system. Figs. D.13, D.14 and D.15 show the new satisficement level of the NFRs after the update of the reward values  $R(s,a)$ .

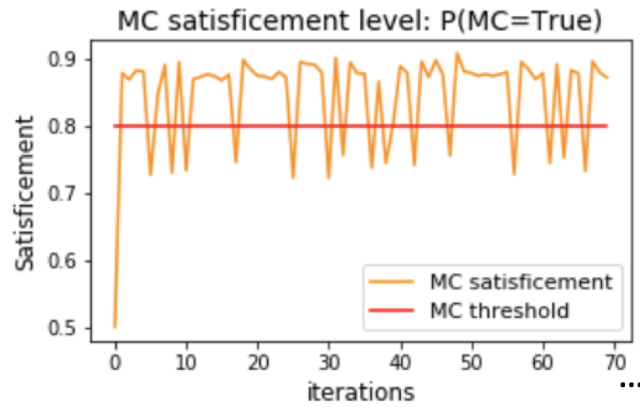


FIGURE D.13: SLAs example 02 - Minimization of Cost: satisficement  
 level after the update of reward values  $R(s,a)$



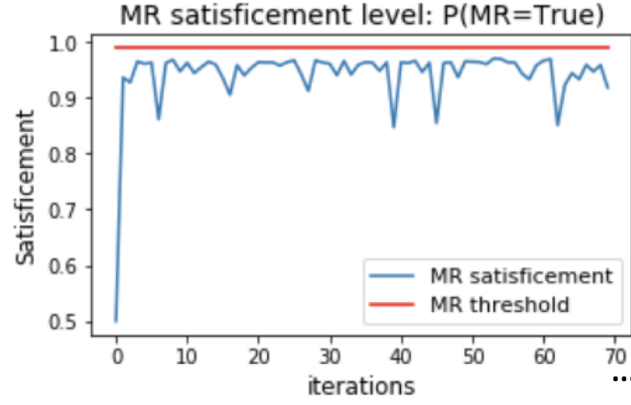


FIGURE D.14: SLAs example 02 - Maximization of Reliability: satisficement level after the update of reward values  $R(s,a)$

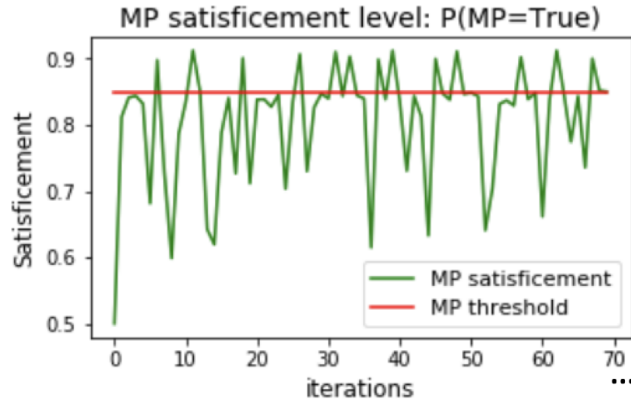


FIGURE D.15: SLAs example 02 - Maximization of Performance: satisficement level after the update of reward values  $R(s,a)$

The average satisficement level for the NFRs shown in Figs. D.13, D.14 and D.15, is MC = **0.8435**, MR = **0.9418** and MP = **0.8075**. In this case, regardless of the increment of the average satisficement level of the reliability of the system in comparison to the Example 01, its satisficement level is constantly under its SLA, as is shown in Fig. D.14.

# Bibliography

- [1] A. Jensen A. Ramirez and B. Cheng. "A taxonomy of uncertainty for dynamically adaptive system". In: *Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2012 ICSE Workshop*. (2012).
- [2] D. Hsu A. Somani N. Ye and W.S. Lee. "DESPOT: online POMDP planning with regularization". In: *In Advances in Neural Information Processing Systems (NIPS)* (2013).
- [3] Jesper Andersson et al. "Software Engineering for Self-Adaptive Systems". In: ed. by Betty H. Cheng et al. Berlin, Heidelberg: Springer-Verlag, 2009. Chap. Modeling Dimensions of Self-Adaptive Software Systems, pp. 27–47. ISBN: 978-3-642-02160-2. DOI: 10.1007/978-3-642-02161-9\_2.
- [4] Haoyu Bai et al. "Intention-aware online POMDP planning for autonomous driving in a crowd". In: *In Proc. IEEE Int. Conf. on Robotics & Automation* (2015).
- [5] Luciano Baresi and Liliana Pasquale. "Fuzzy Goals for Requirements-driven Adaptation". In: *18th International IEEE Requirements Engineering Conference, RE'10*, 2010.
- [6] Nelly Bencomo and Amel Belaggoun. "A world full of surprises: bayesian theory of surprise to quantify degrees of uncertainty". In: *ICSE*. 2014, pp. 460–463.
- [7] Nelly Bencomo and Amel Belaggoun. "Supporting Decision-making for Self-adaptive Systems: From Goal Models to Dynamic Decision Networks". In: *REFSQ - Best Paper Award*. 2013.
- [8] Nelly Bencomo, Amel Belaggoun, and Valerie Issarny. "Dynamic Decision Networks to Support Decision-making for Self-adaptive Systems". In: *(SEAMS)*. 2013.

- [9] Nelly Bencomo and Luis Garcia-Paucar. "RaM: Causally-connected Requirements-aware Models using Bayesian Inference". In: *IEEE/ACM 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS)*. Munich, Germany. (2019).
- [10] Nelly Bencomo et al. "Requirements reflection: requirements as runtime entities". In: *ICSE* (2). 2010, pp. 199–202.
- [11] Gordon Blair, Nelly Bencomo, and Robert France. "Guest Editor's Introduction: Models@run.time". In: *IEEE Software* (2009).
- [12] Jose Bocanegra et al. "Improving the Decision-Making Support in Context-Aware Applications: The Case of an Adaptive Virtual Education Learning Management System". In: *XXI Ibero-american Conference on Software Engineering* (2018).
- [13] Kate M. Bowers, Erik M. Fredericks, and Betty H. C. Cheng. "Automated Optimization of Weighted Non-functional Objectives in Self-adaptive Systems". In: *Search-Based Software Engineering*. Ed. by Thelma Elita Colanzi and Phil McMinn. Cham: Springer International Publishing, 2018, pp. 182–197.
- [14] Darius Braziunas. "POMDP solution methods". In: 2003.
- [15] Javier Cámara et al. "Reasoning about sensing uncertainty and its reduction in decision-making for self-adaptation". In: *Sci. Comput. Program.* 167 (2018), pp. 51–69.
- [16] Javier Camara et al. "Uncertainty in Self-Adaptive Systems: Categories, Management, and Perspectives". In: *Institute for Software Research. Carnegie Mellon University* (2017).
- [17] Betty H. Cheng and et al. "Software Engineering for Self-Adaptive Systems". In: Berlin, Heidelberg: Springer-Verlag, 2009. Chap. Software Engineering for Self-Adaptive Systems: A Research Roadmap, pp. 1–26. ISBN: 978-3-642-02160-2.
- [18] Betty H.C. Cheng et al. "Goal-Based Modeling Approach to Develop Requirements for Adaptive Systems with Environmental Uncertainty". In: *MODELS Conf.* 2009.

- 
- [19] Hamid R. Chinaei and Brahim Chaib-Draa. "Dialogue POMDP Components (Part I): Learning States and Observations". In: *Int. J. Speech Technol.* 17.4 (Dec. 2014), pp. 309–323. ISSN: 1381-2416. DOI: 10.1007/s10772-014-9244-6. URL: <http://dx.doi.org/10.1007/s10772-014-9244-6>.
- [20] Jaedeug Choi and Kee-Eung Kim. "Inverse Reinforcement Learning in Partially Observable Environments". In: *Journal of Machine Learning Research* 12 (July 2011), pp. 691–730. ISSN: 1532-4435. URL: <http://dl.acm.org/citation.cfm?id=1953048.2021028>.
- [21] Lawrence Chung et al. *Non-Functional Requirements in Software Engineering*. Vol. 5. Springer, 1999.
- [22] Golnaz Elahi and Eric Yu. "Requirements Trade-offs Analysis in the Absence of Quantitative Measures: A Heuristic Method". In: *Proceedings of the 2011 ACM Symposium on Applied Computing*. Taiwan, 2011. ISBN: 978-1-4503-0113-8. DOI: 10.1145/1982185.1982331.
- [23] Naeem Esfahani, Ehsan Kouroshfar, and Sam Malek. "Taming uncertainty in self-adaptive software". In: *Proc. of the 19th ACM SIGSOFT Symp FSE*. 11. 2011. DOI: 10.1145/2025113.2025147.
- [24] Naeem Esfahani and Sam Malek. "Uncertainty in Self-Adaptive Software Systems". In: *Software Engineering for Self-Adaptive Systems 2 (SEfSAS 2)*. Springer-Verlag, 2012.
- [25] M.S. Feather et al. "Reconciling system requirements and runtime behavior". In: *Workshop Software Specification and Design* (1998). DOI: 10.1109/IWSSD.1998.667919.
- [26] S. Fickas and M.S. Feather. "Requirements monitoring in dynamic environments". In: *RE Conf.* 1995.
- [27] Antonio Filieri, Carlo Ghezzi, and Giordano Tamburrelli. "A formal approach to adaptive software: continuous assurance of non-functional requirements". In: *Formal Asp. Comput.* 24.2 (2012), pp. 163–186. DOI: 10.1007/s00165-011-0207-2.

- [28] Antonio Filieri, Giordano Tamburrelli, and Carlo Ghezzi. "Supporting Self-Adaptation via Quantitative Verification and Sensitivity Analysis at Run Time". In: *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, VOL. 42, NO. 1 (2016).
- [29] Erik M. Fredericks. "MITIGATING UNCERTAINTY AT DESIGN TIME AND RUN TIME TO ADDRESS ASSURANCE FOR DYNAMICALLY ADAPTIVE SYSTEMS". In: *Michigan State University. PhD Thesis*. (2015).
- [30] Yang Gao et al. "A survey of inverse reinforcement learning techniques." In: *International Journal of Intelligent Computing and Cybernetics* 5.3 (2012), pp. 293–311. ISSN: 1756-378X. URL: <http://search.ebscohost.com/login.aspx?direct=true&db=edsemr&AN=edsemr.10.1108.17563781211255862&site=eds-live&authtype=ip,shib&custid=s9815128>.
- [31] Antonio Garcia-Dominguez, Nelly Bencomo, and Luis Garcia Paucar. "Reflecting on the past and the present with temporal graph-based models". In: *MRT 2018 : 13th International Workshop on Models@run.time. Copenhagen, Denmark, 14.-19. October 2018* (2018).
- [32] Antonio Garcia-Dominguez et al. "Querying and annotating model histories with time-aware patterns". In: *2019 IEEE / ACM 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS)*. Munich, Germany (2019).
- [33] Antonio Garcia-Dominguez et al. "Towards History-aware Self-adaptation with Explanation Capabilities". In: *2019 IEEE 4th International Workshops on Foundations and Applications of Self\* Systems (FAS\*W)*, Umea, Sweeden (2019).
- [34] Jesús García-Galán et al. "User-centric Adaptation of Multi-tenant Services: Preference-based Analysis for Service Reconfiguration". In: *SEAMS*. SEAMS 2014. Hyderabad, India, 2014. ISBN: 978-1-4503-2864-7. DOI: 10.1145/2593929.2593930.
- [35] Luis Garcia-Paucar and Nelly Bencomo. "A Survey on Preferences of Quality Attributes in the Decision-making for Self-Adaptive and Self-Managed Systems: the Bad, the Good and the Ugly." In: *XX Ibero-American Conference on Software Engineering - CIBSE*. Buenos Aires (2017).

- [36] Luis Garcia-Paucar and Nelly Bencomo. "Knowledge Base K Models to Support Trade-offs for Self-adaptation using Markov Processes". In: *13th IEEE Conference on Self-Adaptive and Self-Organizing Systems. Umea, Sweden* (2019).
- [37] Luis Garcia-Paucar and Nelly Bencomo. "RE-STORM: Mapping the Decision-Making Problem and Non-Functional Requirements Trade-off to Partially Observable Markov Decision Processes". In: *SEAMS*. 2018.
- [38] Luis Garcia-Paucar, Nelly Bencomo, and Kevin Fung Yuen. "ARRoW: Automatic Runtime Reappraisal of Weights for Self-Adaptation". In: *34th ACM/SI-GAPP Conference on Applied Computing. Limassol, Cyprus* (2019).
- [39] Holger Giese et al. "Living with Uncertainty in the Age of Runtime Models". In: *Models@run.time*. Vol. 8378. LNCS. Springer, 2014, pp. 47–100.
- [40] Holger Giese et al. "Living with Uncertainty in the Age of Runtime Models". English. In: *Models@run.time*. Ed. by Nelly Bencomo et al. Vol. 8378. LNCS. Springer International Publishing, 2014, pp. 47–100. ISBN: 978-3-319-08914-0. DOI: 10.1007/978-3-319-08915-7\_3.
- [41] D. Hsu H. Kurniawati and W.S. Lee. "SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces". In: *In Proc. Robotics: Science and Systems* (2008).
- [42] John S. Hammond, Ralph L. Keeney, and Howard Raiffa. "Even Swaps: A Rational Method for Making Trade-offs". In: *Harvard Business Review, March–April*, 137–150 (1998).
- [43] McMinn P.-De Souza J. & Yoo S. Harman M. "Search based software engineering: Techniques, taxonomy, tutorial." In: *Search, 2012*, 1–59. (2011).
- [44] Sara Hassan, Nelly Bencomo, and Rami Bahsoon. "Minimize Nasty Surprises with Better Informed Decision-Making in Self-Adaptive Systems". In: *10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. 2015.
- [45] E.J. Horvitz, J.S. Breese, and M. Henrion. "Decision theory in expert systems and artificial intelligence". In: *Int. Journal of Approximate Reasoning* 2, 247–302 (1988) (1998).

- [46] Maximilian Igl et al. "Deep Variational Reinforcement Learning for POMDPs". In: *Proceedings of Machine Learning Research* 80 (2018). Ed. by Jennifer Dy and Andreas Krause, pp. 2117–2126. URL: <http://proceedings.mlr.press/v80/igl18a.html>.
- [47] Minwen Ji, Alistair C Veitch, John Wilkes, et al. "Seneca: remote mirroring done write." In: *USENIX Annual Conference*. 2003, pp. 253–268.
- [48] Sindhu R. Johnson et al. "A valid and reliable belief elicitation method for Bayesian priors". In: *Journal of Clinical Epidemiology* (2010). ISSN: 0895-4356. DOI: <https://doi.org/10.1016/j.jclinepi.2009.08.005>.
- [49] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. "Planning and Acting in Partially Observable Stochastic Domains". In: *Artif. Intell.* 101.1-2 (May 1998), pp. 99–134. ISSN: 0004-3702. DOI: 10.1016/S0004-3702(98)00023-X. URL: [http://dx.doi.org/10.1016/S0004-3702\(98\)00023-X](http://dx.doi.org/10.1016/S0004-3702(98)00023-X).
- [50] Santos C.-Beyer D. Chase J. Wilkes J Keeton K. "Designing for disasters". In: *In Proceedings of the 3rd USENIX Conference on File and Storage Technologies*, pp. 59–62. *USENIX Association, Berkeley* (2004).
- [51] J. O. Kephart and D. M. Chess. "The vision of autonomic computing". In: *Computer* 36.1 (2003), pp. 41–50. DOI: 10.1109/MC.2003.1160055.
- [52] Daiko Kishikawa and Sachiyo Arai. "Comfortable Driving by using Deep Inverse Reinforcement Learning." In: *2019 IEEE International Conference on Agents (ICA), Agents (ICA), 2019 IEEE International Conference on* (2019), pp. 38–43. ISSN: 978-1-7281-4026-1. URL: <http://search.ebscohost.com/login.aspx?direct=true&db=edsee&AN=edsee.8929214&site=eds-live&authtype=ip,shib&custid=s9815128>.
- [53] Daphne Koller and N Friedman. "Probabilistic Graphical Models: Principles and Techniques". In: 2009.
- [54] Christian Krupitzer et al. "A survey on engineering approaches for self-adaptive systems". In: *Pervasive and Mobile Computing* 17, Part B (2015). ISSN: 1574-1192. DOI: <http://dx.doi.org/10.1016/j.pmcj.2014.09.009>.

- [55] H. Kurniawati and V. Yadav. "An Online POMDP Solver for Uncertainty Planning in Dynamic Environment". In: *Proc. Int. Symp. on Robotics Research* (2013).
- [56] Rogerio de Lemos et al. "Software Engineering for Self-Adaptive Systems: A second Research Roadmap". In: *Software Engineering for Self-Adaptive Systems*. Dagstuhl Seminar Proceedings 10431. Germany: Schloss Dagstuhl, 2011.
- [57] Emmanuel Letier, David Stefan, and Earl T. Barr. "Uncertainty, Risk, and Information Value in Software Requirements and Architecture". In: *Proceedings of ICSE*. ICSE 2014. Hyderabad, India: ACM, 2014, pp. 883–894. ISBN: 978-1-4503-2756-5. DOI: 10.1145/2568225.2568239.
- [58] Sotirios Liaskos et al. "Representing and Reasoning About Preferences in Requirements Engineering". In: *Requir. Eng.* (Sept. 2011), pp. 227–249. ISSN: 0947-3602. DOI: 10.1007/s00766-011-0129-9.
- [59] Sheng-Hsuan Lin and Mohammad Arfan Ikram. "On the relationship of machine learning with causal inference." In: *European Journal of Epidemiology* (2019), p. 1. ISSN: 0393-2990. URL: <http://search.ebscohost.com/login.aspx?direct=true&db=edssjs&AN=edssjs.EDD75808&site=eds-live&authtype=ip,shib&custid=s9815128>.
- [60] Pattie Maes. "Concepts and Experiments in Computational Reflection". In: *SIGPLAN Not.* 22.12 (Dec. 1987), pp. 147–155. ISSN: 0362-1340. DOI: 10.1145/38807.38821. URL: <http://doi.acm.org/10.1145/38807.38821>.
- [61] G. A. Moreno et al. "Efficient Decision-Making under Uncertainty for Proactive Self-Adaptation". In: *2016 IEEE International Conference on Autonomic Computing (ICAC)*. 2016, pp. 147–156. DOI: 10.1109/ICAC.2016.59.
- [62] David E. Morris, Jeremy E. Oakley, and John A. Crowe. "A web-based tool for eliciting probability distributions from experts". In: *Environmental Modelling & Software* (2014). ISSN: 1364-8152. DOI: <https://doi.org/10.1016/j.envsoft.2013.10.010>.
- [63] Anthony O'Hagan. "Probabilistic uncertainty specification: Overview, elaboration techniques and their application to a mechanistic model of carbon



- flux". In: *Environmental Modelling & Software* (2012). ISSN: 1364-8152. DOI: <https://doi.org/10.1016/j.envsoft.2011.03.003>.
- [64] Hariri S. Parashar M. "Autonomic Computing: An Overview." In: *Unconventional Programming Paradigms. UPP 2004. Lecture Notes in Computer Science, vol 3566. Springer, Berlin, Heidelberg* (2005.).
- [65] Luis H Garcia Paucar and Nelly Bencomo. "Runtime Models Based on Dynamic Decision Networks : Enhancing the Decision-making in the Domain of Ambient Assisted Living Applications". In: *MRT - Models@run.time at MOD-ELS 2016, Saint-Malo, France* (2016).
- [66] Luis H Garcia Paucar and Nelly Bencomo. "The Reassessment of Preferences of Non-Functional Requirements for Better Informed Decision-making in Self-Adaptation". In: *AIRE - 3rd International Workshop on Artificial Intelligence for Requirements Engineering. Beijing* (2016).
- [67] Luis H Garcia Paucar, Nelly Bencomo, and Kevin Kam Fung Yuen. "Juggling Preferences in a World of Uncertainty". In: *RE NEXT, Lisbon*. (2017).
- [68] Luis Hernán García Paucar and Nelly Bencomo. "ARRoW: Tool Support for Automatic Runtime Reappraisal of Weights". In: *25th IEEE International Requirements Engineering Conference, RE 2017, Lisbon, Portugal, September 4-8, 2017*. 2017, pp. 458–461. DOI: 10.1109/RE.2017.58.
- [69] Judeal Pearl. "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference." In: *Morgan Kaufmann* (1988).
- [70] X. Peng et al. "Self-Tuning of Software Systems Through Goal-based Feedback Loop Control". In: *Requirements Engineering Conference (RE)*. 2010, pp. 104–107. DOI: 10.1109/RE.2010.22.
- [71] Pascal Poupart. "Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov Decision Processes". In: *Thesis for the degree of Doctor of Philosophy. University of Toronto* (2005).
- [72] Florian Pusse and Matthias Klusch. "Hybrid Online POMDP Planning and Deep Reinforcement Learning for Safer Self-Driving Cars." In: *2019 IEEE Intelligent Vehicles Symposium (IV), Intelligent Vehicles Symposium (IV), 2019*

- IEEE* (2019), pp. 1013–1020. ISSN: 978-1-7281-0560-4. URL: <http://search.ebscohost.com/login.aspx?direct=true&db=edsee&AN=edsee.8814125&site=eds-live&authtype=ip,shib&custid=s9815128>.
- [73] Andres Ramirez et al. “RELAXing Claims: Coping With Uncertainty While Evaluating Assumptions at Run Time”. In: *MODELS* (2012).
- [74] Andres J. Ramirez and Betty H C Cheng. “Automatic derivation of utility functions for monitoring software requirements”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6981 LNCS (2011), pp. 501–516. ISSN: 03029743. DOI: 10.1007/978-3-642-24485-8\_37.
- [75] S Ross et al. “Online planning algorithms for POMDPs”. In: *J. Artificial Intelligence Research*, 32(1), 663–704 (2008).
- [76] Stuart J. Russell and Peter Norvig. *Artificial intelligence - a modern approach: the intelligent agent book*. Prentice Hall series in artificial intelligence. Prentice Hall, 1995, pp. I–XXVIII, 1–932. ISBN: 978-0-13-103805-9.
- [77] Eric Rutten, Nicolas Marchand, and Daniel Simon. “Feedback Control as MAPE-K Loop in Autonomic Computing”. In: (2017). Ed. by Rogério de Lemos et al., pp. 349–373.
- [78] T. Saaty. “Decision making with the analytic hierarchy process”. In: *Inter. Journal of Services Sciences*, (2008).
- [79] Maria Salama, Rami Bahsoon, and Nelly Bencomo. “Managing Trade-offs in Self-Adaptive Software Architectures: A Systematic Mapping Study”. In: *Managing trade-offs in adaptable software architectures*. Ed. by Ivan Mistrok et al. Elsevier, 2016.
- [80] Mazeiar Salehie and Ladan Tahvildari. “Self-adaptive Software: Landscape and Research Challenges”. In: *ACM Trans. Auton. Adapt. Syst.* 4.2 (May 2009). ISSN: 1556-4665. DOI: 10.1145/1516533.1516538.
- [81] Pete Sawyer et al. “Requirements-Aware Systems: A Research Agenda for RE for Self-adaptive Systems”. In: *Proceedings of RE. RE '10*. Washington, DC, USA: IEEE, 2010. ISBN: 978-0-7695-4162-4.

- [82] D Silver and J Veness. "Monte-Carlo planning in large POMDPs". In: *Advances in Neural Information Processing Systems (NIPS)* (2010).
- [83] David Silver and Joel Veness. "Monte-Carlo planning in large POMDPs". In: *Advances in neural information processing systems*. 2010, pp. 2164–2172.
- [84] T. Smith and R. Simmons. "Heuristic search value iteration for POMDPs." In: *In Proc. Conf. on Uncertainty in Artificial Intelligence*, pp. 520–527 (2004.).
- [85] Adhiraj Somani et al. "DESPOT: Online POMDP planning with regularization". In: *Advances in neural information processing systems*. 2013, pp. 1772–1780.
- [86] Hui Song et al. "Self-Adaptation with End-User Preference: Using Run-Time Models and Constraint Solving". In: *the Intrnl. Conference MODELS*. USA, 2013. DOI: 10.1007/978-3-642-41533-3\_34.
- [87] Joao Pedro Sousa et al. "User guidance of resource-adaptive systems". In: *In Proc. of International Conference on Software and Data Technologies*. 2008.
- [88] Vítor E. Silva Souza, Alexei Lapouchnian, and John Mylopoulos. "(Requirement) Evolution Requirements for Adaptive Systems". In: *Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. SEAMS '12. Zurich, Switzerland: IEEE Press, 2012, pp. 155–164. ISBN: 978-1-4673-1787-0. URL: <http://dl.acm.org/citation.cfm?id=2666795.2666820>.
- [89] Richard Sutton and Andrew Barto. "Reinforcement Learning, An Introduction". In: *The MIT Press* (2019).
- [90] W. E. Walsh et al. "Utility functions in autonomic systems". In: *Autonomic Computing, 2004. Proceedings. International Conference on*. 2004, pp. 70–77. DOI: 10.1109/ICAC.2004.1301349.
- [91] Mark Weiser. "The Computer for the 21st Century". In: *SIGMOBILE Mob. Comput. Commun. Rev.* 3.3 (July 1999), pp. 3–11. ISSN: 1559-1662. DOI: 10.1145/329124.329126. URL: <http://doi.acm.org/10.1145/329124.329126>.

- [92] Kristopher Welsh, Pete Sawyer, and Nelly Bencomo. "Towards requirements aware systems: Run-time resolution of design-time assumptions". In: *ASE*. 2011, pp. 560–563.
- [93] Jon Whittle et al. "RELAX: A language to address uncertainty in self-adaptive systems requirement". In: *Requirements Engineering* (2010). ISSN: 09473602. DOI: 10.1007/s00766-010-0101-0.
- [94] Jon Whittle et al. "RELAX: Incorporating Uncertainty into the Specification of Self-Adaptive Systems". In: *RE Conf.* 2009.
- [95] Long Xin et al. "Accelerated Inverse Reinforcement Learning with Randomly Pre-sampled Policies for Autonomous Driving Reward Design." In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC), Intelligent Transportation Systems Conference (ITSC), 2019 IEEE* (2019), pp. 2757–2764. ISSN: 978-1-5386-7024-8. URL: <http://search.ebscohost.com/login.aspx?direct=true&db=edsee&AN=edsee.8916952&site=eds-live&authtype=ip,shib&custid=s9815128>.
- [96] Nan Ye et al. "DESPOT: Online POMDP Planning with Regularization". In: *Journal of Artificial Intelligence Research* 58 (2017) 231-266 (2017).
- [97] E. Yu. "Towards modeling and reasoning support for early-phase requirements engineering". In: *RE Conf. USA*, 1997.
- [98] Eric Yuan, Naeem Esfahani, and Sam Malek. "A Systematic Survey of Self-Protecting Software Systems". In: *ACM Trans. Auton. Adapt. Syst.* 8.4 (Jan. 2014), 17:1–17:41. ISSN: 1556-4665. DOI: 10.1145/2555611.
- [99] K. K. F. Yuen. "Cognitive network process with fuzzy soft computing technique in collective decision aiding". In: *The Hong Kong Polytechnic University, PhD thesis* (2009).
- [100] K. K. F. Yuen. "Pairwise opposite matrix and its cognitive prioritization operators: comparisons with pairwise reciprocal matrix and analytic prioritization operators". In: *Journal of the Operational Research Society* (2012) 63, 322–338 (2012).

- 
- [101] K. K. F. Yuen. "The Primitive Cognitive Network Process in healthcare and medical decision making: Comparisons with the Analytic Hierarchy Process". In: *Applied Soft Computing Journal* 14.PART A (2014), pp. 109–119. ISSN: 15684946. DOI: 10.1016/j.asoc.2013.06.028. URL: <http://dx.doi.org/10.1016/j.asoc.2013.06.028>.
- [102] K. K. F. Yuen and Wei Wang. "Towards a ranking approach for sensor services using primitive cognitive network process". In: *4<sup>th</sup> Annual IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems, IEEE-CYBER 2014* (2014), pp. 344–348. DOI: 10.1109/CYBER.2014.6917487.