

Modelling and Simulation of Adaptive Multi-Agent Systems with Stochastic Nets-within-Nets

Michael Köhler-Bußmeier¹ ^a and Lorenzo Capra² ^b

¹University of Applied Sciences Hamburg, Berliner Tor 7, D-20099 Hamburg, Germany

²Dipartimento di Informatica, Università degli Studi di Milano, Via Celoria 18, Milan, Italy

Keywords: Adaptive Systems, Multi-Agent Systems, Stochastic Petri Nets, Nets-within-Nets, Simulation.

Abstract: This study centers on self-adapting multi-agent systems modeled utilizing the SONAR framework. Our key focus is on forecasting the costs and benefits of adaptation during execution within the MAPE loop (monitor, analyse, plan, and execute). Analyzing these adaptation processes is intricate due to SONAR enabling second-order activities, such as structural adaptation involving agent interaction protocols or the organizational network itself. We forecast these dynamic processes using a stochastic run-time model (e.g., the environment has a stochastic representation). Since SONAR is conceptualized with HORNETS (a nets-within-nets formalism), we necessitate “probabilistic” HORNETS. To illustrate our approach’s effectiveness, we showcase a small case study of a self-modifying MAS organization and provide an analysis of adaptation dynamics.

1 INTRODUCTION

This research focuses on self-adaptive systems, particularly multi-agent systems (MAS) (Weiß, 1999) and their analysis (cf. (Capra and Köhler-Bußmeier, 2024)). Our application domain, that is, MAS for cyberphysical systems (Leitao and Karnouskos, 2015), allows *structural* modifications at run-time. Therefore, we use a specification framework, called SONAR (Köhler-Bußmeier et al., 2009), which enables the modification, addition, and deletion of components (e.g., agent interaction protocols) using a MAPE-Loop (monitor, analyse, plan, and execute) (Weyns, 2020). On the one hand, self-modification is powerful. However, it makes analysis, such as predicting adaption costs and benefits, challenging. It becomes even more demanding when performed at runtime (*analysis@run.time*) as part of the MAPE loop (Donckt et al., 2018).


The SONAR MAPE-Loop is specified as a Petri net, which has organizations and interaction protocols as tokens. Since protocols are Petri nets, we also use Nets-within-nets (Valk, 2003), which allow Petri nets to be tokens of other nets. The MAS organization is specified using HORNETS (Köhler-Bußmeier, 2009), which are a specialization of algebraic Petri


nets (Reisig, 1991) with net operators and nesting.

This research seeks to facilitate the examination of self-adaptive systems using a digital twin of the entire system. To achieve a functional model suitable for real-time analysis, the twin model integrates stochastic (probabilistic) components to abstract, for instance, the agents’ decision-making processes. Consequently, our twin model adopts HORNETS enhanced with probabilistic parameters.

Using runtime models for the analysis and planning of self-adaptation is an increasingly popular research domain; cf. (Simeoni et al., 2004; Bencomo et al., 2014; Calinescu et al., 2012; Bencomo et al., 2019). In this context, we illustrate our method for run-time analysis through a basic example where the MAS is able to make structural changes.

This article has the following structure: In Section 2, we present the formalism SONAR and its semantics, the SONAR-MAPE loop. We will explain how structural modifications of the MAS are carried out at run-time through second-order teamwork. In Section 3 we will present the digital twin model that is used during the analysis and planning phase of the MAPE-Loop to predict the benefit of the adaptations. We will shortly explain the underlying Petri net formalism named HORNETS. Section 4 presents the case study – implemented using RENEW (Kummer et al., 2004). The results of the analysis are given in Section 5. The work ends with a conclusion.

^a  <https://orcid.org/0000-0002-3074-4145>

^b  <https://orcid.org/0000-0002-1029-1169>

2 MAS-ORGANIZATIONS AND THE SONAR-MAPE-LOOP

Multi-agent systems (MAS) constitute a network of interacting agents. Unlike general distributed systems, MAS assumes that agents are autonomous entities, each pursuing its own goals. Traditional concepts in multi-agent systems, such as reasoning, coordination, negotiation, etc., are considered bottom-up approaches (Weiß, 1999). To structure MAS as a whole, these concepts are complemented by top-down concepts such as roles, hierarchies, positions, etc. All these concepts are encompassed by that of an *organization* (Org-MAS) (Dignum and Padget, 2013).

More than a decade ago, the SONAR framework (Köhler-Bußmeier et al., 2009) was introduced, a formalism based on Petri nets to specify and analyze Organizational Multi-Agent Systems (Org-MAS). In addition, a SONAR model is employed to create an execution engine. The SONAR engine establishes a general *execution loop* called the SONAR MAPE loop (Köhler-Bußmeier and Sudeikat, 2024), where a *task* initiates the *formation* of a team. This team then formulates a team *plan* (through negotiation), which is carried out in a distributed fashion by the team agents: organization \rightarrow Team \rightarrow Plan

As a special feature, the execution of the team plan may involve *transformation* statements that modify the SONAR-model at run-time. This enables *cooperative, self-organized adaption* of the organization:

```

organization
→ Team
→ Plan
↔ Transformation at run-time
→ new organization
→ ...

```

The formalism of SONAR (Köhler-Bußmeier et al., 2009) defines a notion of well-formedness (which we will not introduce here) to guarantee that each task may be handled by at least one team, etc. Furthermore, transformations in well-formed organizations preserve well-formedness.

3 THE DIGITAL TWIN MODEL USING HORNETS

The MAPE-Loop is nested within itself as it includes a digital twin. During the planning step, a model of the entire loop is used to predict the benefits of potential adjustments. We have opted for the Nets-within-Nets paradigm (Valk, 2003) to represent this recursive structure, employing a Petri net formalism where

the tokens themselves are Petri nets. Nets-within-nets can be seen as the Petri net perspective on contextual change, in contrast to the process algebra view of the Ambient Calculus (Cardelli et al., 1999) or the π -calculus (Milner et al., 1992). Here, we have selected HORNETS (Köhler-Bußmeier, 2009) since the organization model SONAR is based on Petri nets. These nets become net-tokens in our MAPE-Loop. Especially, we make use of the ability of HORNETS to perform algebraic operations on these net-tokens. Our Sonar-MAPE-Loop utilizes this functionality to adjust the structure of the organization net or the work-flows (modeled as net-tokens) during runtime.

3.1 Hornets

In the following, we give a rather informal introduction into HORNETS –for more details cf. (Köhler-Bußmeier, 2009; Köhler-Bußmeier, 2014a).

We will illustrate the main concepts of HORNETS using the example given in Figure 1. For simplicity, the set of net types only contains the workflow net: $K = \{\text{WFN}\}$. We have one operator \parallel for the parallel composition: $\parallel \in \Sigma_{\text{WFN}^2, \text{WFN}}$. The operator is interpreted by I as the usual AND operation between the workflow nets. We have the universe with three object nets: $\mathcal{U}_{\text{WFN}} = \{N_1, N_2, N_3\}$. All places of the system-net have the same type. The structure of the system net \hat{N} and the object nets is given in the usual way, as shown in Fig. 1. This EHORNET does not use communication channels, that is, all events occur autonomously. In the initial marking, we consider a EHORNET with two nets N_1 and N_2 as tokens (as shown on the left): $\mu_0 = \hat{p}[N_1, v] + \hat{q}[N_2, s]$.

To model a run-time adaption, we combine N_1 and N_2 resulting in the net $N_3 = (N_1 \parallel N_2)$. This modification is modelled by system net transition \hat{r} of the HORNET. The net token generated on \hat{r} has the structure of N_3 and its marking is obtained as a transfer from the token on v in N_1 and the token on s in N_2 into N_3 :

$$\hat{p}[N_1, v] + \hat{q}[N_2, s] \xrightarrow{\theta_1} \hat{r}[(N_1 \parallel N_2), s + v]$$

This transfer is possible since all the places of N_1 and N_2 are also places in N_3 and the tokens can be transferred in an obvious way.

Assume a fixed many-sorted predicate logic $\Gamma = (\Sigma, X, E, \Psi)$ with operators Σ , variables X , equations E , and predicates Ψ .

Definition 1. An elementary HORNET (EHORNET) is a tuple $EH = (\hat{N}, \mathcal{U}, I, k, \Theta, \mu_0)$ such that:

1. \hat{N} is an algebraic net, called the system net.
2. (\mathcal{U}, I) is a finite net-theory for the logic Γ .

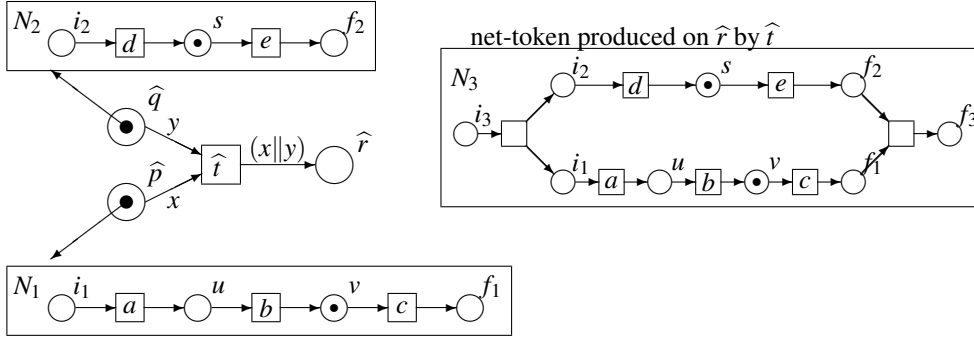


Figure 1: Modification of the Net-Token's Structure.

3. $k : \hat{P} \rightarrow K$ is the typing of the system-net places.
4. Θ is the set of nested events.
5. $\mu_0 \in \mathcal{M}_H$ is the initial marking.

We refer to (Köhler-Bußmeier, 2009; Köhler-Bußmeier, 2014a) for the formal definition of the firing rule.

The reachability graph $RG(EH) = (V, E, \mu_0)$ is defined in the usual way. The nested markings are the nodes, that is, $V = \mathcal{M}_H$, and the firing rule $\mu \xrightarrow{\theta} \mu'$ generates the edges $(\mu, \theta, \mu') \in E$.

For stochastic HORNETS, we equip the model with a rate function $\Lambda : \Theta \rightarrow \mathbb{R}^{>0}$ that assigns firing rates $\Lambda(\theta)$ to events $\theta \in \Theta$.

For stochastic Petri nets (SPN) (Marsan, 1990), we normalize over all transitions enabled in a given marking to obtain probabilities from these rates. The same idea is applied to the nested events of EHORNETS in the following. Let $En(\mu) := \{\theta \in \Theta \mid \mu \xrightarrow{\theta}\}$ be the set of all events enabled in the nested marking μ . Then, the probability of firing $\theta \in En(\mu)$ is proportional to its rate $\Lambda(\theta) \in \mathbb{R}^{>0}$. For EHORNETS we define the firing probability as:

$$Pr_{\mu}(\theta) := \frac{\Lambda(\theta)}{\sum_{\theta \in En(\mu)} \Lambda(\theta)} \quad (1)$$

For an arc (μ, θ, μ') in the reachability graph $RG(EH) = (V, E, \mu_0)$ we define its probability as:

$$Pr((\mu, \theta, \mu')) := Pr_{\mu}(\theta) \quad (2)$$

These probabilities turn the reachability graph into a (discrete) Markov chain.

3.2 The Digital Twin of the SONAR-MAPE-Loop

The Digital Twin of our SONAR-MAPE-Loop is used during the planning stage of the MAPE-Loop. The twin model allows us to evaluate different adaptation options based on their costs and advantages. Since

planning must come before ongoing activities, the digital twin version is more simplified. The digital twin of our SONAR-MAPE-Loop is illustrated in Fig. 2. In contrast to the original loop (Köhler-Bußmeier and Sudeikat, 2024), we utilize the following abstractions:

- We incorporate environmental assumptions using a stochastic model, specifically a distribution $P[\text{TASKS} = p_0]$ to generate initial tasks p_0 .
- We replace the decision logic of organizational position agents (OPAs), which resolves the conflicts between the team formation operators op , with a distribution $P[\text{ACT}_p = op]$.
- In the workflow nets (WFN) (Aalst, 1997), which specifies the agent interaction protocols, we simulate the decision logic for xor choices, again, with a stochastic distribution $P[\text{XOR}_p = \text{choice}_k]$.
- We also incorporate the agents' decision logic and their learning capabilities (i.e., parts that are external from the organization's viewpoint) using a stochastic model.

For more details on the SONAR-MAPE-Loop, the reader is referred to (Köhler-Bußmeier and Sudeikat, 2024).

4 THE CASE STUDY: SELF-MODIFYING ORGANIZATIONS

In our case study, we consider a scenario involving co-learning, which means that learning occurs at both the individual agent level and the organizational level. For simplicity, we will focus on the latter. The scenario is based on the well-known *battle of sexes* from game theory, in which two agents must choose between two actions, labeled a and b . They receive a positive reward if they choose the same action, and

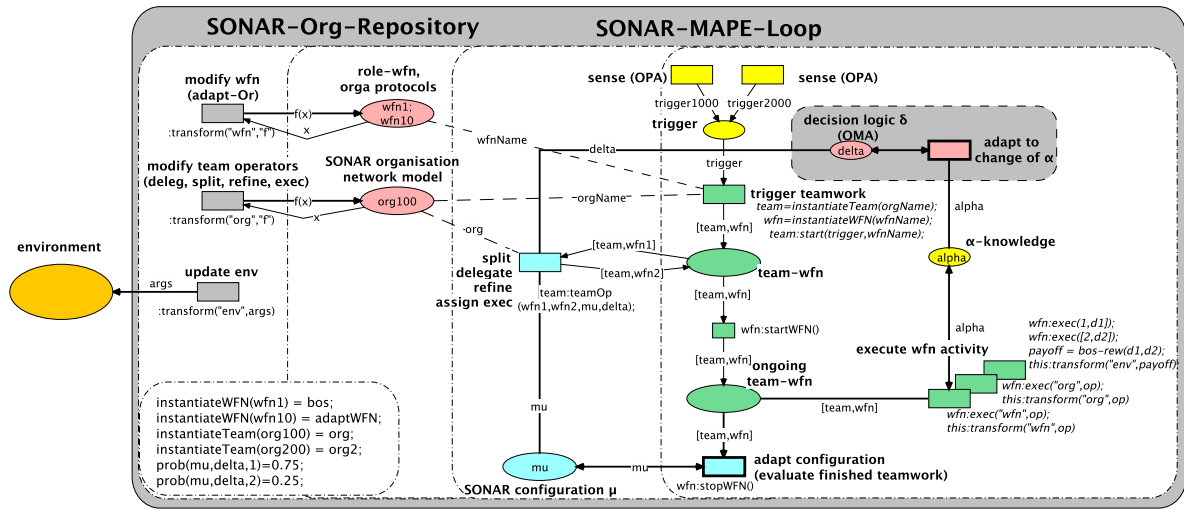


Figure 2: The SONAR-Mape-Loop (adapted from (Köhler-Bußmeier and Sudeikat, 2024)).

zero otherwise. In this game, the first agent prefers the action a , while the second prefers b . If we assume that the reward for the preferred outcome is three times higher than for the other, then the payoff matrix below specifies the game:

Here, we have two Nash equilibria for pure strategies: (a, a) and (b, b) . It can be shown that this game has an equilibrium *unique* for *mixed strategy*, where both agents choose their preferred action $p = 75\%$ of the time.

	a	b
a	(3, 1)	(0, 0)
b	(0, 0)	(1, 3)

The situation described is known as a coordination game because agents would benefit from coordinating their actions in advance. In this scenario, the social welfare (which is the sum of the individual payoffs) is $3 + 1 = 1 + 3 = 4$ in both cases. However, in uncoordinated games, a mixed strategy is the best choice, and agents only coordinate in $0.75 \cdot 0.25 + 0.75 \cdot 0.25 = 0.375$ of the cases, leading to an expected payoff of $(0.75 \cdot 0.25 + 0.75 \cdot 0.25) \cdot 4 = 1.5$. The ratio $\frac{3+1}{1.5} = 2.66\dots$, called *price-of-anarchy*, measures the need for an external coordination mechanism.

For MAS, this mechanism is termed an *organization* (Dignum and Padget, 2013). The SONAR-Org-MAS of this scenario is depicted in Fig. 3. The primary purpose of the SONAR-Org-Model is to assemble a team in response to certain triggers. It determines the workflow net used for the response and the agents involved and sets constraints on the agents' decisions, represented by parameters μ and δ . The Org-Model has three organizational agents O_0 , O_1 , and O_2 . Manages two tasks: The first task triggers the

team formation process for the workflow net (shown in Fig. 4) representing the battle-of-sexes interaction P . The protocol P defines the interaction of two roles R_1 and R_2 . The model specifies that R_1 is assigned to O_1 and R_2 to O_2 .

The choice between options a and b depends on the probability, $prob$, which is calculated as $c \cdot p_{org} + (1 - c) \cdot p_{agent}$, where p_{agent} represents the agents' decision logic and p_{org} represents the organizational constraint. For simplicity, set p_{agent} and p_{org} both to 75%, which represents the probability of choosing option a based on the Nash equilibrium. Therefore, we obtain $prob = 75\%$, regardless of the value of the organizational impact c .

The right side of the organization handles a task that triggers the adaptation. The second task in Fig. 3 initiates the team formation for a second-order WFN (cf. Fig. 5), which alters the role fragment so that the second agent always chooses option a (that is, we force the second agent to deviate from the optimal mixed strategy). This structural modification is formalized in the upper block (modify WFN) of the WFN. Furthermore, this second-order WFN extends the original organization model to incorporate this modification, as formalized in the lower block (modify organization model). It is worth noticing that we need a protocol here to synchronize the necessary elementary transformation steps. The shaded area of the organization net in Fig. 3 shows the nodes that will be added by the second order protocol. Since each agent block receives new nodes, the second-order protocol designates three roles to the agents O_0 , O_1 , and O_2 .

We use the syntax of the Java-based Petri net tool RENEW (Kummer et al., 2004) to execute the scenario. The sources are available at <https://github.com/>

Team Workflow (2nd order Protocol)

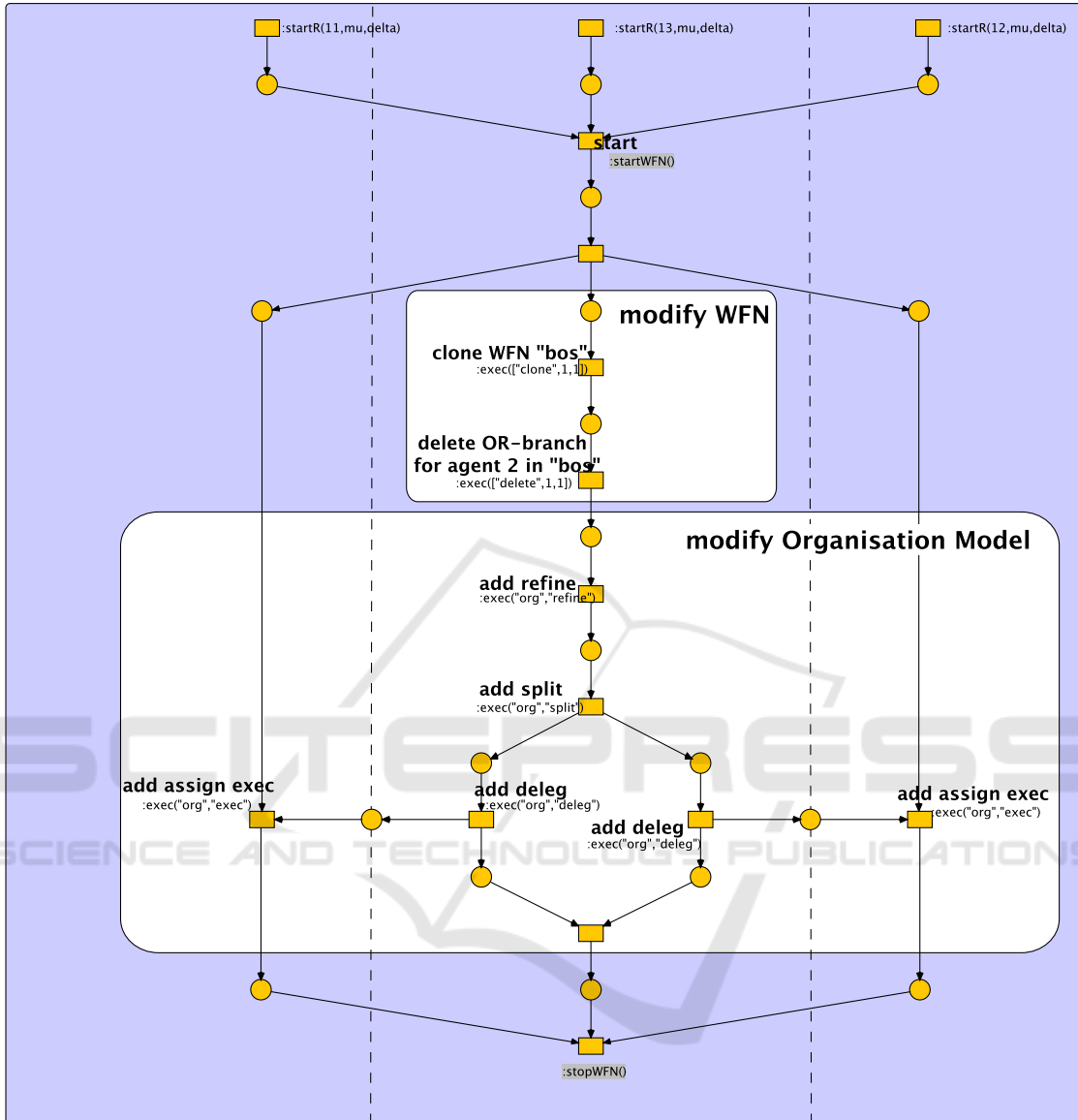


Figure 5: Second-order WFN: Team-based modification of the workflow given in Fig. 4 and the organization given in Fig. 3 .

Table 1: Simulation Results (left) and Expected Values (right).

outcome	before and ...	after adaptation	before (expected)	after (expected)
(a, a)	20%	78%	18,75%	75%
(a, b) or (b, a)	62%	22%	62.50%	25%
(b, b)	18%	0%	18,75%	0%

(b, b). In 62.5% of times, the agents will choose opposite options.

After the adaptation has taken place, we expect that in $0.75 \cdot 1$ of interactions, the agents will play (a, a) and in $0.25 \cdot 1$ of interactions, the agents will play (b, a). This leads to the expected social welfare

of $0.75 \cdot (3 + 1) + 0.25 \cdot (0 + 0) = 3$. So, the new price of anarchy is $\frac{3+1}{3} = 1.33\dots$, which is a substantial improvement from the initial value of $\frac{3+1}{1.5} = 2.66\dots$

Note that restricting the agents' options is not always beneficial: Removing option a for the second agent results in a welfare of $0.25 \cdot (3 + 1) + 0.75 \cdot$

$(0 + 0) = 1$. This leads to a new price-of-anarchy of $\frac{3+1}{1} = 4$, which is worse. The costs of this transformation would be the same.

Although anticipated and illustrated by a straightforward example, these results demonstrate the ability to forecast the costs and benefits of structural changes in complex organizations through the quantitative modeling and simulation of distributed agents using a formal digital-twin approach.

6 CONCLUSION

Our research explores self-modifying systems within the context of Org-MAS and MAPE-Loop. We use the formalism HORNETS for the twin model to predict the cost-benefit ratio of adaptation. Since the model includes stochastic elements, we require an appropriate analysis tool. In this study, we examine a simple case study using *the battle of sexes* game to demonstrate that the quantitative analysis of adaption using HORNETS is feasible and beneficial for the designer.

In our current research, we are focusing on converting elementary, two-levels, HORNETS into stochastic Symmetric Petri Nets (SSN) (that can be assessed within the GREATSPN framework (Amparore et al., 2016)) by mimicking net-tokens through an emulator (Camilli and Capra, 2021). Given the lower level of abstraction in SSN, we also aim to describe the semantics of HORNETS employing an advanced algebraic framework, such as Maude (Clavel et al., 2007). This extends the approach we applied to Elementary Object Systems (EOS) (Capra and Köhler-Bußmeier, 2023b; Capra and Köhler-Bußmeier, 2023a).

In general, such a translation is quite complex due to the nested nature of Nets-within-Nets. (Note, that in general there is no equivalent unnested nets, since the reachability problem is undecidable for Nets-within-Nets (Köhler-Bußmeier, 2014b), while it is decidable for place transition nets; so we need at least colored tokens and algebraic inscriptions.) However, we observed that our SONAR-MAPE-Loop has an interesting structural property, that is, the system-net does not combine or distribute net-tokens of the same net type, i.e., we have no forks or joins, which are usually the source of complexity for HORNETS (Köhler-Bußmeier, 2017).

For a similar subclass of EOS, called Generalized State Machines (GSM), we have already shown in (Köhler-Bußmeier, 2014b) that this structural subclass can be translated into an equivalent unnested Petri net. In ongoing work, we extend this idea to obtain an equivalent unnested algebraic Petri net from a

fork/join-free EHORNET. This translation would allow symbolic methods to rely on established and efficient analysis methods.

REFERENCES

- Aalst, W. v. d. (1997). Verification of workflow nets. In Azeme, P. and Balbo, G., editors, *Application and theory of Petri nets*, volume 1248 of *Lecture Notes in Computer Science*, pages 407–426, Berlin Heidelberg New York. Springer-Verlag.
- Amparore, E. G., Balbo, G., Beccuti, M., Donatelli, S., and Franceschinis, G. (2016). 30 years of GreatSPN. In *Principles of Performance and Reliability Modeling and Evaluation*, pages 227–254. Springer.
- Bencomo, N., France, R., Cheng, B., and Aßmann, U., editors (2014). *Models@run.time: foundations, applications, and roadmaps*. Lecture Notes in Computer Science. Springer, Germany.
- Bencomo, N., Götz, S., and Song, H. (2019). Models@run.time: a guided tour of the state of the art and research challenges. *Software & Systems Modeling*, 18(5):3049–3082.
- Calinescu, R., Ghezzi, C., Kwiatkowska, M., and Miranda, R. (2012). Self-adaptive software needs quantitative verification at runtime. *Commun. ACM*, 55(9):69–77.
- Camilli, M. and Capra, L. (2021). Formal specification and verification of decentralized self-adaptive systems using symmetric nets. *Discrete Event Dynamic Systems*, 31(4):609–657.
- Capra, L. and Köhler-Bußmeier, M. (2023a). A Maude formalization of object nets. In Batista, T., Bureš, T., Raibulet, C., and Muccini, H., editors, *Software Architecture. ECSA 2022 Tracks and Workshops*, volume 13928 of *Lecture Notes in Computer Science*, pages 246–261. Springer-Verlag.
- Capra, L. and Köhler-Bußmeier, M. (2023b). Maude specification of nets-within-nets: A formal model of adaptable distributed systems. In Hong, J. and Lanperne, M., editors, *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing, SAC '23*, page 188–191. ACM.
- Capra, L. and Köhler-Bußmeier, M. (2024). Modular rewritable Petri nets: an efficient model for dynamic distributed systems. *Theoretical Computer Science*, 990:114397.
- Cardelli, L., Gordon, A. D., and Ghelli, G. (1999). Mobility types for mobile ambients. In *Proceedings of the Conference on Automata, Languages, and Programming (ICALP'99)*, volume 1644 of *Lecture Notes in Computer Science*, pages 230–239. Springer-Verlag.
- Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., and Talcott, C. L., editors (2007). *All About Maude - A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic*, volume 4350 of *Lecture Notes in Computer Science*. Springer-Verlag.

- Dignum, V. and Padget, J. (2013). Multiagent organizations. In Weiss, G., editor, *Multiagent Systems, 2nd ed.*, Intelligent Robotics; Autonomous Agents Series, pages 51–98. MIT Press.
- Donckt, J. V. D., Weyns, D., Iftikhar, M. U., and Buqar, S. S. (2018). Effective decision making in self-adaptive systems using cost-benefit analysis at runtime and online learning of adaptation spaces. In *Best papers of ENASE'18*. Springer-Verlag.
- Köhler-Bußmeier, M. (2009). Hornets: Nets within nets combined with net algebra. In Wolf, K. and Franceschinis, G., editors, *International Conference on Application and Theory of Petri Nets (ICATPN'2009)*, volume 5606 of *Lecture Notes in Computer Science*, pages 243–262. Springer-Verlag.
- Köhler-Bußmeier, M. (2014a). On the complexity of the reachability problem for safe, elementary Hornets. *Fundamenta Informaticae*, 129:101–116. Dedicated to the Memory of Professor Manfred Kudlek.
- Köhler-Bußmeier, M. (2014b). A survey on decidability results for elementary object systems. *Fundamenta Informaticae*, 130(1):99–123.
- Köhler-Bußmeier, M. (2017). Restricting Hornets to support adaptive systems. In van der Aalst, W. and Best, E., editors, *PETRI NETS 2017*, Lecture Notes in Computer Science. Springer-Verlag.
- Köhler-Bußmeier, M. and Sudeikat, J. (2024). Studying the micro-macro-dynamics in MAPE-like adaption processes. In *16th International Symposium on Intelligent Distributed Computing (IDC'23)*. Springer-Verlag.
- Köhler-Bußmeier, M., Wester-Ebbinghaus, M., and Moldt, D. (2009). A formal model for organisational structures behind process-aware information systems. *Transactions on Petri Nets and Other Models of Concurrency. Special Issue on Concurrency in Process-Aware Information Systems*, 5460:98–114.
- Kummer, O., Wienberg, F., Duvigneau, M., Schumacher, J., Köhler, M., Moldt, D., Rölke, H., and Valk, R. (2004). An extensible editor and simulation engine for Petri nets: Renew. In Cortadella, J. and Reisig, W., editors, *International Conference on Application and Theory of Petri Nets 2004*, volume 3099 of *Lecture Notes in Computer Science*, pages 484 – 493. Springer-Verlag.
- Leitao, P. and Karnouskos, S. (2015). *Industrial Agents: Emerging Applications of Software Agents in Industry*. Elsevier Science Publishers B. V., Amsterdam, The Netherlands, 1st edition.
- Marsan, M. A. (1990). *Stochastic Petri nets: an elementary introduction*, pages 1–29. Springer-Verlag, Berlin, Heidelberg.
- Milner, R., Parrow, J., and Walker, D. (1992). A calculus of mobile processes, parts 1-2. *Information and computation*, 100(1):1–77.
- Reisig, W. (1991). Petri nets and algebraic specifications. *Theoretical Computer Science*, 80:1–34.
- Simeoni, M., Balsamo, S., Inverardi, P., and Di Marco, A. (2004). Model-based performance prediction in software development: A survey. *IEEE Transactions on Software Engineering*, 30:295–310.
- Valk, R. (2003). Object Petri nets: Using the nets-within-nets paradigm. In Desel, J., Reisig, W., and Rozenberg, G., editors, *Advanced Course on Petri Nets 2003*, volume 3098 of *Lecture Notes in Computer Science*, pages 819–848. Springer-Verlag.
- Weiß, G., editor (1999). *Multiagent systems: A modern approach to Distributed Artificial Intelligence*. MIT Press.
- Weyns, D. (2020). *An Introduction to Self-Adaptive Systems: A Contemporary Software Engineering Perspective*. John Wiley & Sons Ltd.