Multi-Level Online Learning and Reasoning for Self-Integrating Systems

Marius Pol
Independent researcher
Paris, France
mariuspol@messagebox.email

Ada Diaconescu
LCTI Lab, *Télécom Paris, IP Paris*,
Palaiseau, France
ada.diaconescu@telecom-paris.fr

***Abstract*—Self-improving and self-integrating systems (SISSY) often employ runtime models to represent their state and environment, and reason upon them to determine the required adaptation logic for reaching their goals. However, most model-based approaches rely on static modeling languages and cannot handle runtime uncertainty (e.g. dynamically integrated resources) that requires online language extensions. In previous work, we proposed an approach to extend the system's modeling language with new monitoring and action dimensions. However, the solution generates a high number of new language elements, slowing down the reasoning process for large systems. In this position paper, we propose a multi-level approach for extending the modeling language at runtime, and aim to provide *online learning* and *reasoning* at *multiple levels of abstraction*. Increasing the modeling abstraction decreases the number of concepts to reason about, hence improving scalability. We provide a preliminary validation of this proposal by detecting novel abstract dimensions from monitoring data from the smart home domain.**

***Index Terms*—self-integration, model-based adaptation, multi-level, online learning, on the fly reasoning, knowledge abstraction**

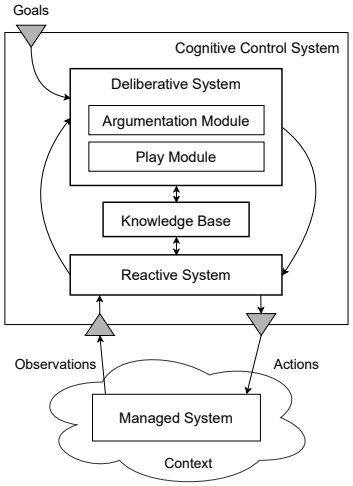

Fig. 1. Architectural Overview of the Cognitive Control System: the deliberative system provides reflective capabilities to deal with new situations; the reactive system handles change management in known situations.

I. Introduction

Self-adaptive systems often use runtime models of themselves and their environment to compute adaptation actions for meeting their goals when changes occur [1]. Such runtime models are synchronized with the managed system – i.e. via causal relations, both imposing and reflecting system changes dynamically [2]. However, traditionally, the modeling language (i.e. meta-model [3]) on which models are based upon is defined statically and offline. This limits system adaptations to those that can be defined via the predefined language.

Self-improving and self-integrating (SISSY) systems are open self-adaptive systems that must deal with unpredictable situations [4] (e.g. integrating new types of resources or systems at runtime). Hence, they require a dynamic and flexible language vocabulary to represent and reason about themselves, when undefined changes occur. In previous work [5], we proposed to learn new language elements, or 'concepts', at runtime, so as to extend the system's modeling language – and hence be able to represent and reason about unexpected situations. The proposed cognitive control system consisted of: a *deliberative* layer – for acquiring new knowledge in unexpected cases; and, a *reactive* layer – for managing the system in known cases (Figure 1). However, as the number of modeled concepts increases, determining adaptation solutions requires more-and-more computational resources, hence limiting the scalability of this initial approach.

To support modeling language extensions for large systems, we propose a solution based on *multi-level online learning* and *reasoning* (i.e. an end-to-end *self-aware* solution [6]). As in our initial solution, we replace traditional logical models, based on a fixed language, with geometric representations of monitoring data, that allow learning the underlying language online and computing the adaptation logic on the fly. We extend this approach to multi-dimensional representations, as a basis for conceptual abstraction. The learning process uses novelty-detection [7] to identify clusters in the monitoring data representations (Figure 2); and defines new language elements, i.e. 'concepts', based on these clusters. Multidimensional clusters represent abstract concepts; and unidimensional ones basic concepts. A *knowledge abstraction* method extends the modeling language with new vector-based representations, generated at runtime. Hence, the concepts of the modeling language are learned at increasingly higher abstraction levels. This decreases the number of concepts to reason about, hence improving system scalability. Reasoning determines adaptation

actions via on the fly computations on the learned concepts.

The paper's contributions are: a) online learning of multi-level concepts; b) proactive learning of correlations between actions and their effects; c) on the fly reasoning for reactive adaptations. We illustrate our proposal via examples from the smart home domain (sec. IV); and provide preliminary validation by generating abstract concepts based on the monitoring data obtained from a smart home dataset [8][1].

We present an overview of our approach and its theoretical background in (II). Then we focus on related work (III) and an illustration of our contributions within a smart home scenario (IV). We detail the knowledge abstraction process in (V) and the learning and reasoning processes in (VI). We present preliminary results in (VII) before concluding in (VIII).

II. Solution Overview & Background

A. Architectural Overview

This proposal extends our previous approach – i.e., a cognitive control system composed of a deliberative system and a reactive system, sharing a knowledge base (Figure 1). When the knowledge base already contains adaptation plans for a detected problem, the reactive system uses it to perform model-based adaptation. When the required adaptation knowledge is missing, the reactive system activates the deliberative system, whose argumentation module reasons upon the existing knowledge to compute new adaptation plans. When even the knowledge that supports reasoning is missing, the deliberative system activates the play module to learn it. Learning is therefore part of the reasoning process, to complete the existing model when adaptation knowledge is missing.

B. Online Modeling within Conceptual Spaces

The proposed learning process involves knowledge represented in *conceptual spaces* [9] – a cognitive knowledge representation framework where data is represented as vectors in geometrical spaces built on monitoring dimensions. These vectors denote the centroids of concepts, i.e. convex regions regrouping similar observations (Figure 2). Conceptual spaces are built on *domains*, which are composed of one or several dimensions. In the resulting geometric space, a distance measure may be defined, providing a similarity measure between observations. To select the relevant dimensions for representation, the conceptual spaces theory proposes weights, which intervene in the distance computation. The theory therefore offers *context-dependent*, *partial* representations.

In our proposal, every goal pursued has an associated conceptual space. Similar observations cluster into convex regions denoting concepts. At anyone time, the centroids of these regions depend on the data composing the concept, providing *dynamic* representations suitable for runtime operation. Low-level concepts are represented on domains containing single dimensions. We propose to increase the abstraction level of these representations by automatically determining domains with multiple dimensions. The knowledge abstraction method is based on mCANDIES – an online *novelty detection* method for SISSY systems [7]. Novelty detection determines clusters denoting concepts not yet modeled, and supports the generation of new concepts as weighted means of these clusters. Clusters contain concepts on multiple dimensions, therefore providing abstract representations (e.g. in a smart home system, an abstract cost goal composed of power and gas consumption).

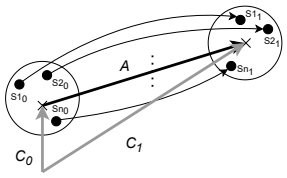

Fig. 2. Conceptual Spaces Representation: Black dots represent monitoring data. Gray vectors point to the middle of data clusters and represent concepts. Action A is represented as a category of changes from concept C_0 to C_1. Action A is determined via a contrast operation between concepts C_1 and C_0.

C. Modeling Language based on Predicates

The approach generates a logic-based *modeling language*, whose syntax is a set of logic symbols $\mathcal{L}$. A semantic mapping $\mathcal{M}$ provides meaning for syntax elements by relating logic symbols with conceptual representations, i.e. $\mathcal{M} : \mathcal{L} \rightarrow \mathcal{C}$. The play module adds conceptual representations to $\mathcal{C}$, and then, via a labeling function, generates symbols for these concepts, symbols which extend $\mathcal{L}$. The correspondence between symbols and concepts is included in $\mathcal{M}$. The actual symbols presented here are associated manually, for readability; providing them autonomously is out of the paper's scope.

To represent concepts generated by the play module on the symbolic level of representation, and enable computations in the deliberative system, we define logical predicates. A *predicate* is a function, denoted by a symbol, defined for variables monitored in the managed system, mapping onto binary values true or false. We associate predicates to concepts generated in our approach, predicates considered true when observations fall into the convex regions denoting these concepts (e.g predicate *ShowerWaterOn* for observations falling into the region denoted by the vector $S_{ShowerWaterOn}$ in Figure 3). In our approach, the *conceptual level of representation* is vector-based, while the *symbolic* one relies on predicates.

D. Online Reasoning

To reason for finding the adaptation logic when goals are no longer met, the possible actions and their effects also need to be modeled. The concepts introduced above represent system states. *Adaptation actions* are represented as categories of changes in conceptual spaces [10]. Figure 2 shows the representation of an action A, as a category of changes from a concept C_0 to a concept C_1. Data monitored on the domain of C_0 clusters to form the convex region denoting this concept (points $S1_0$, ..., Sn_0). Similarly, the concept C_1 regroups observations $S1_1$, ..., Sn_1. The action A represents a category

[1]https://data.mendeley.com/datasets/fcj2hmz5kb/3, last access August 2021

of changes from points represented by C_0 to points represented by C_1.

To generate the required symbolic knowledge based on the conceptual representations created by knowledge abstraction, we rely on a *contrast operation* in conceptual spaces [11]. This is a difference operation between concepts, and outputs a vector containing relevant dimensions that differentiate the input concepts (Figure 2). As the contrast operation is a vector, it has an associated concept, and hence a predicate. To model the actions and their effects, the contrast operation determines the logical symbols that represent relevant differences between concepts. As the system gains in experience, the generated knowledge becomes more abstract, therefore involving less computations for determining the adaptation knowledge.

The proposed reasoning capabilities extend both components of the previous cognitive control system: on the fly reasoning for the reactive system for fast reactions, and deliberative reasoning for computing adaptation logic based on existing knowledge. The *deliberative reasoning* capabilities support the three main reasoning processes observed in humans: deduction, induction, and abduction. *Deductive* reasoning occurs when determining the effects of actions. *Inductive* reasoning is accomplished in our case via the knowledge abstraction method. Here, induction may be considered as learning from clusters of observations, but also as learning from prior knowledge (e.g. refining a concept representing a cost goal by progressively adding more dimensions as they become relevant at runtime). *Abductive* reasoning determines the adaptation knowledge required to solve the targeted goals.

III. Related Work

Multi-layered control systems (e.g. three-layer reference model [12], Observer/Controller model [13]) propose a reflective layer for managing adaptation plans when required by changes in the environment. Our proposal builds on previous work implementing a reflective layer, a deliberative system with learning and reasoning capabilities at a low level of abstraction, offered by available sensors and actuators [5]. Self-awareness is an ongoing research challenge in the related fields of autonomic computing [14] and organic computing [15]. Multiple types of *knowledge representation* are available for managing adaptation knowledge [16]: model-based, goal-based, and utility-based. Our proposal is a learning system that generates adaptation knowledge autonomously.

Reinforcement learning may bridge the semantic gap between high-level goals and low-level actions. To cope with large-scale SISSY systems, *learning* is based on hybrid approaches (e.g. [17], [13]), which include design-time models updated at runtime to support adaptation in dynamic environments. While hybrid methods are based on knowledge represented on the symbolic level, we propose that systems build their own conceptual representations, and determine the symbolic representations at runtime. Moreover, due to the partial nature of the generated models, the proposed learning process focuses on knowledge relevant to the goals pursued. Learning classifier systems [18] also represent the knowledge

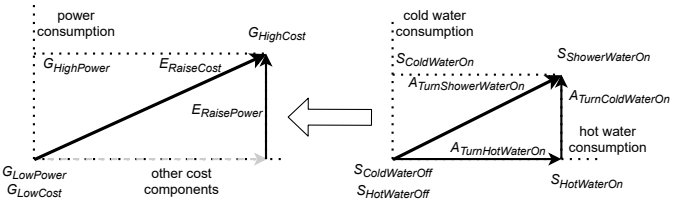

Fig. 3. Knowledge Abstraction Method. The power consumption component $E_{RaisePower}$ may be solved with the abstract action $A_{TurnShowerWaterOn}$, composed of $A_{TurnColdWaterOn}$ and $A_{TurnHotWaterOn}$. Black solid arrows designate abstract knowledge, while the empty arrow represents a learned correlation. G and S represent goals and current states. A and E represent actions and their effects.

geometrically, however the learning process focuses on one specific goal. Our approach also considers the priority of other goals pursued in determining solutions. While existing approaches are reactive, i.e. learning as new situations are experienced at runtime, our proposal takes a proactive approach for knowledge generation [19]. Namely, the play module generates knowledge proactively, during inactivity periods, to extend the existing modeling language with abstract representations. Modeling *causal relations* dynamically is essential for self-adaptive systems operating in uncertain environments [20]. Our current proposal only detects correlations, via novelty detection. Hence, future work will explore existing proposals [21] to determine actual causality.

The *reasoning* methods for runtime adaptation depend on the type of knowledge representation (e.g. model-based [22], goal-based [23], utility-based [24]). Our approach supports reasoning on all these representations. Based on the method to solve the semantic distance between high- and low-level representations, reasoning approaches for runtime adaptation may be top-down, bottom-up or hybrid [16]. Our approach is hybrid, with a bottom-up process for increasing the level of representation of conceptual knowledge, and a top-down reasoning process that focuses the learning process on knowledge relevant to the goals pursued.

IV. Contributions Overview in a Smart Home Scenario

In this section, we highlight our main contributions applied to a smart home scenario. The scenario focuses on a partial problem, relevant to our approach, where a smart home manager aims to lower power consumption as a way of achieving a cost goal. The power-consuming resources are not yet modeled. The scenario starts when the managed system needs to be adapted, as the goal pursued is $G_{LowCost}$, and the current state is $G_{HighCost}$ (Figure 3). To solve a goal, the deliberative system's reasoning aims to undo the actions that lead to the current unwanted state [5]. The effect $E_{RaiseCost}$ may be decomposed into a power consumption component, $E_{RaisePower}$, and other cost components, which we ignore here. We suppose the knowledge base contains no adaptation plans for solving the partial goal $G_{HighPower}$. Therefore, the deliberative system activates the play module to learn the missing knowledge.

A. Multi-Level Online Learning

To avoid a fixed modeling language, the play module represents the learned knowledge in conceptual spaces [9], via vector-based representations. For multi-level learning, the play module relies on the online novelty-detection capabilities in mCANDIES [7], to determine data clusters and create abstract concepts. In conceptual spaces, concepts represent system states, and actions represent categories of changes from one concept to another (see II-D and Figure 2). For example, action $A_{ColdWaterOn}$ represents changes from $S_{ColdWaterOff}$ to $S_{ColdWaterOn}$. The play module detects the clusters for concepts that represent system states before and after a goal's resolution, then generates the action representing this change. In our example, the play module analyses historical monitoring data and determines the clusters of concepts that solved $G_{HighPower}$ in the past: the cluster composed of $S_{ColdWaterOff}$ and $S_{HotWaterOff}$ in the initial state, and $S_{ColdWaterOn}$ and $S_{HotWaterOn}$ in the final state. The play module generates the abstract concepts $S_{ShowerWaterOff}$ and $S_{ShowerWaterOn}$ as weighted means of the clusters resulting from novelty detection. The generated concepts are added to the semantic domain of the modeling language $\mathcal{C}$, while the associated predicates extend the syntax $\mathcal{L}$. The play module takes a self-improving approach for learning, analyzing recent data to rapidly determine possible solutions, then considers older data looking for optimal solutions.

The abstract action $A_{TurnShowerWaterOn}$ is generated by the contrast operation, as the difference between $S_{ShowerWaterOn}$ and $S_{ShowerWaterOff}$. The meaning of the abstract action is observable in the figure, where the hot water component is more important than the cold water one. With the generated action $A_{TurnShowerWaterOn}$, the plan solving $G_{RaisePower}$ contains a single computation step, which increases the efficiency of the deliberative system when recomputing adaptation logic in this part of the state space, compared to a plan composed two low-level actions (i.e. $A_{TurnColdWaterOn}$ and $A_{TurnHotWaterOn}$). Therefore, the play module enables *multi-level online learning* for concepts (i.e. $S_{ShowerWaterOn}$), as well as *learning correlations* (i.e. plan containing $A_{TurnShowerWaterOn}$ to solve $G_{RaisePower}$).

B. Reactive and Deliberative Reasoning

Based on the concepts generated in the learning process, the cognitive control system may reason on the fly to determine adaptation plans via the contrast operation [11]. Along with raising the power consumption, another effect of $A_{TurnShowerWaterOn}$ is that the gas consumption also increases, as showering is accompanied by heating the bathroom. The play module generates the concept $G_{HighGas}$, and stores the correlation between it and $A_{TurnShowerWaterOn}$ for future use (not pictured in Figure 3). Hence, in future reasoning processes for goals that involve $G_{HighGas}$, the reactive system generates the required adaptation logic on the fly, via a rapid reaction, without having to activate the play module to generate it. During inactivity periods, the deliberative system searches for more computationally effective adaptation plans, exploiting the abstract concepts learned as above. We will provide examples for the three deliberative reasoning processes in VII.

V. Conceptual Knowledge Abstraction

In our proposal, abstract concepts are generated as weighted means of more basic concepts, represented on single dimensions. Clustering is performed only on dimensions relevant to a goal, associated to the same conceptual space, hence resulting in partial representations. The centroid of a convex region denoting a concept may be interpreted as a vector starting at the origin of the geometrical space. An abstract concept is computed as the sum of vectors denoting lower level concepts, and may therefore be viewed as a weighted mean of its components. The components to be considered for the knowledge abstraction process are determined on the dimensions where the most important changes between final and initial states are noticed.

Actions and effects are represented as changes on a single domain [25], which may contain multiple dimensions. The knowledge abstraction process extends the existing domains with new dimensions. The system learns to perform changes on additional dimensions, increasing the power of changes possible in the managed system with a single computation.

The proposed method does not aim to generate perfect abstract concepts immediately, and takes a self-improving approach to learning [26]. Concepts are created when the play module detects clusters in monitoring data, from a single observation. These concepts may be subjected to biases of experience and goals. Concepts that prove useful for computing the adaptation knowledge are stored and strengthened, while other concepts may be left out of the knowledge base. Hence, the modeling language refines as the reflective layer gains more experience, and is anchored in lower-level concepts, which evolve as their composing data evolves.

VI. Online Learning and On the Fly Reasoning

In this section, we focus on the symbolic level of knowledge representation and present the proposed learning and reasoning capabilities for SISSY systems.

A. Learning Abstract Concepts

Abstract concepts may be expressed in terms of their composing lower-level concepts, by projecting them onto their composing dimensions. Therefore, abstract conceptual representations maintain a connection to the underlying lower-level concepts. For example, the concept $S_{ShowerWaterOn}$ may be expressed as a weighted mean of $S_{ColdWaterOn}$ and $S_{HotWaterOn}$. When this knowledge is required for adaptation, the play module has access to this relation, as it is included in the conceptual knowledge representation.

Even though the play module stores all modeled concepts, these may not be included in the solution that started the original reasoning process in the deliberative system. Indeed, generated solutions may not be acceptable, e.g., as enacting them may generate conflicts with more important goals. Therefore, in the process of solving a single goal, multiple concepts may be modeled by the play module.

B. Reasoning via Concept Contrasting

The cognitive control system performs deliberative reasoning when adaptation logic is present in the knowledge base. The argumentation module in the deliberative system performs reasoning on the symbolic level of representation. This paper focuses on extending the modeling language at multiple levels of abstraction for reactive reasoning capabilities; more details about the deliberative system are presented in [5].

Goals are solved on the conceptual level of representation, by decomposing the effects required to achieve them from the current state. The deliberative system computes the effect required for adaptation with the contrast operation, as the vector from the current state to the goal state (e.g. $E_{RaiseCost}$ in Figure 3). It then decomposes the result, and determines possible solutions based on the available knowledge. When only low-level knowledge is available, the goal decomposition contains detailed low-level effects, which require many computations. As the knowledge abstraction process generates more abstract representations, returned solutions involve less reasoning. The decomposition that requires minimum computation to solve a goal is determined as the shortest path in a directed acyclic graph generated for the effects which have solutions in the knowledge base. The actions solving the initial goal represent the output of the reasoning process.

An example of this operation in shown in Figure 3. During the reasoning process (e.g. while adapting to $G_{HighCost}$), the contrast operation computes the required effect as the difference between the goal pursued on the power consumption dimension, $G_{HighPower}$, and the current state, $G_{LowPower}$. The resulting difference is represented as the effect $E_{RaisePower} \in \mathcal{C}$, and $RaisePower \in \mathcal{L}$. The play module is activated to learn the adaptation knowledge required to obtain this effect (as in IV-A), and extends the modeling language with the results, i.e. the semantic domain becomes $\mathcal{C} \cup \{C_{ShowerWaterOn}\}$, and the generated predicate is added to the syntax $\mathcal{L} \cup \{ShowerWaterOn\}$.

As goal representations are available at both symbolic and conceptual levels, our proposal supports model- and goal-based reasoning. Goals are expressed as conjunctions of concepts represented at a low abstraction level, to guide the knowledge generation process while the system is gaining experience. Following the knowledge abstraction method, the reflective layer incrementally builds its abstract representations of targeted goals. As the deliberative system considers goal priorities when generating the required adaptation knowledge, the cognitive control system layer should support utility-based reasoning.

VII. Results from a Smart Home Scenario

The proposed reflective layer is implemented in Python. The implementation contains a novelty detection component mCANDIES [7] [2]. The play module uses this to determine the effects of actions (forward), to identify possible actions leading to goals (backward), and to perform the knowledge abstraction method. For representing monitoring data in conceptual spaces, we rely on an existing implementation of conceptual spaces [27] [3]. For decomposing vectors during reactive reasoning, we included the *networkx* Python package for handling directed acyclic graphs [4].

In the performed experiments, the play module carries-out knowledge abstraction to learn from the ContextAct@A4H dataset [8] [5]. The dataset contains over 300 variables monitored over the months of July and November in a smart home with a single occupant. The only supposition we make about the data is that the action dimensions (i.e. dimensions available for control in managed systems) are given.

We suppose as known the conceptual representations on the power consumption dimension (*Water_Heater_Consumption* [6]), where there are two defined concepts, the current state and the goal pursued, i.e. $\mathcal{C} = \{ G_{HighPower}, G_{LowPower} \}$. During the reasoning process, the play module starts looking for possible actions that achieve $G_{HighPower}$ from $G_{LowPower}$ (cf. IV). Based on historical monitoring data, the play module determines the times t when the goal was achieved in the past, i.e. when the values measured on the power consumption dimension change from $G_{LowPower}$ to $G_{HighPower}$. The learning method performs online novelty detection at times t determined as above.

The play module determines possible causes of system states (i.e. abduction). Starting from the state where the goal pursued is met, our implementation plays in reverse the historical data to determine the possible actions that achieve the goal, waiting for a novel cluster to be detected in order to generate the concepts representing these actions. For example, the result of this backwards reasoning process is the cluster of concepts $S_{ColdWaterOn}$ and $S_{HotWaterOn}$, determined respectively on the *Cold_Water_Shower_Consumption* and *Hot_Water_Shower_Consumption* dimensions. The play module then generates the $S_{ShowerWaterOn}$ abstract concept. The weights of the dimensions in the generated concept are set as the magnitudes of the composing vectors. The concept for shower water is composed of 0.020 cold water and 0.073 hot water. Another concept discovered in the dataset is hands washing water on. This concept is based on the same low-level concepts as $S_{ShowerWaterOn}$, however, it is composed of equally important cold and hot water concepts. Having generated $S_{ShowerWaterOn}$, the adaptation plan for solving $G_{HighPower}$ contains only one action $A_{TurnShowerWaterOn}$. With our previous proposal, the plan would have been composed of two actions, $A_{TurnColdWaterOn}$, and $A_{TurnHotWaterOn}$.

The play module also generates the effects of these actions (i.e. deduction), to determine the consequences of the found solutions in the managed system. The play module updates the novelty detection component with data available after the times t, i.e. after the execution of actions, and represents the observed effects (e.g. $G_{HighGas}$, determined on the *Gas_Consumption* dimension). Learning $G_{HighGas}$ concept provides reactive rea-

[2]https://novelty-detection.net/p/ndnet, last access August 2021

[3]https://github.com/lbechberger/ConceptualSpaces, last access August 2021

[4]https://networkx.org/, last access August 2021

[5]https://data.mendeley.com/datasets/fcj2hmz5kb/3, last access August 2021

[6]The dimension names in the ContextAct@A4H dataset are in French.

soning capabilities for future adaptations. During both of these processes (i.e. deduction and abduction), the play module generates abstract concepts (i.e. induction) based on lower-level concepts.

VIII. Conclusions and Future Work

We proposed a reflective layer for managing adaptation knowledge, which provides multi-level online learning and on the fly reasoning capabilities for SISSY systems. Learning involves a slow conceptual knowledge abstraction process, and a fast symbolic knowledge generation process driven by the goals pursued. The reactive reasoning capabilities occur on the fly, based on conceptual knowledge whose level of abstraction is continuously increased by the proposed learning method.

As future work, we are interested in researching the communication possibilities offered by the modeling language generated by the proposed approach. Conceptual representations have the same meaning in conceptual spaces built on the same dimensions [28], and can therefore be shared between interacting SISSY systems. During self-integration, self-adaptive systems may grasp new concepts by contrasting their concepts with concepts available in the integrating systems.

We focused here on a reflective layer to manage adaptation knowledge in single self-integrating system. To extend these communication capabilities in a collaboration context, we envision a multi-level approach for the composition of system of systems [29]. Individual systems self-manage their adaptation knowledge by generating models driven by their own goals. They are also part of larger systems, where they may provide goals to other systems, via collaboration based on these communication capabilities. We envision a top-down approach, with central control for the goals pursued, and a self-organizing approach for collaboration with other systems.

References

[1] C. Krupitzer, F. M. Roth, S. VanSyckel, G. Schiele, and C. Becker, "A survey on engineering approaches for self-adaptive systems," *Pervasive and Mobile Computing*, vol. 17, pp. 184–206, Feb. 2015.

[2] T. Vogel, S. Neumann, S. Hildebrandt, H. Giese, and B. Becker, "Model-Driven Architectural Monitoring and Adaptation for Autonomic Systems," *Proc. 6th Int. Conf. Auton. Comput. - ICAC 09*, p. 67, 2009.

[3] G. Lehmann, M. Blumendorf, F. Trollmann, and S. Albayrak, "Meta-modeling Runtime Models," in *Models in Software Engineering*, J. Dingel and A. Solberg, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, vol. 6627, pp. 209–223.

[4] K. Bellman, S. Tomforde, and R. P. Wurtz, "Interwoven Systems: Self-Improving Systems Integration," in *2014 IEEE Eighth International Conference on Self-Adaptive and Self-Organizing Systems Workshops*. Imperial College, London, UK: IEEE, Sep. 2014, pp. 123–127.

[5] M. Pol and A. Diaconescu, "A Cognitive Control System for Managing Runtime Uncertainty in Self-Integrating Autonomic Systems," in *2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS 2020)*, 2020, p. 6.

[6] S. Kounev, P. Lewis, K. L. Bellman, N. Bencomo, J. Camara, A. Diaconescu, L. Esterle, K. Geihs, H. Giese, S. Götz, P. Inverardi, J. O. Kephart, and A. Zisman, "The Notion of Self-aware Computing," in *Self-Aware Computing Systems*. Cham: Springer International Publishing, 2017, pp. 3–16.

[7] C. Gruhl, B. Sick, and S. Tomforde, "Novelty detection in continuously changing environments," *Future Generation Computer Systems*, vol. 114, pp. 138–154, Jan. 2021.

[8] P. Lago, F. Lang, C. Roncancio, C. Jiménez-Guarín, R. Mateescu, and N. Bonnefond, "The ContextAct@A4H Real-Life Dataset of Daily-Living Activities," in *Modeling and Using Context*. Cham: Springer International Publishing, 2017, vol. 10257, pp. 175–188.

[9] P. Gärdenfors, *Conceptual Spaces: The Geometry of Thought*. MIT Press, 2000.

[10] P. Gärdenfors, "Representing actions and functional properties in conceptual spaces," in *Volume 1 Embodiment*. De Gruyter Mouton, 2008, p. 29.

[11] J.-L. Dessalles, "From Conceptual Spaces to Predicates," in *Applications of Conceptual Spaces*, F. Zenker and P. Gärdenfors, Eds. Cham: Springer International Publishing, 2015, pp. 17–31.

[12] J. Kramer and J. Magee, "Self-Managed Systems: An Architectural Challenge," in *Future of Software Engineering (FOSE '07)*. Minneapolis, MN, USA: IEEE, May 2007, pp. 259–268.

[13] S. Tomforde, H. Prothmann, J. Branke, J. Hähner, M. Mnif, C. Müller-Schloer, U. Richter, and H. Schmeck, "Observation and Control of Organic Systems," in *Organic Computing — A Paradigm Shift for Complex Systems*, C. Müller-Schloer, H. Schmeck, and T. Ungerer, Eds. Basel: Springer Basel, 2011, pp. 325–338.

[14] Petr Jan Horn, "Autonomic Computing: IBM's Perspective on the State of Information Technology," 2001.

[15] C. Müller-Schloer, "Organic computing: On the feasibility of controlled emergence," in *Proceedings of the 2nd IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis - CODES+ISSS '04*. Stockholm, Sweden: ACM Press, 2004, p. 2.

[16] P. Lalanda, *Autonomic Computing: Principles, Design and Implementation*. New York: Springer, 2013.

[17] G. Tesauro, N. Jong, R. Das, and M. Bennani, "A Hybrid Reinforcement Learning Approach to Autonomic Resource Allocation," in *2006 IEEE International Conference on Autonomic Computing*. Dublin, Ireland: IEEE, 2006, pp. 65–73.

[18] M. V. Butz, *Rule-Based Evolutionary Online Learning Systems: A Principled Approach to LCS Analysis and Design*, ser. Studies in Fuzziness and Soft Computing. Berlin: Springer, 2006, no. v. 191.

[19] A. Stein, S. Tomforde, A. Diaconescu, J. Hahner, and C. Muller-Schloer, "A Concept for Proactive Knowledge Construction in Self-Learning Autonomous Systems," in *2018 IEEE 3rd International Workshops on Foundations and Applications of Self* Systems (FAS*W)*. Trento: IEEE, Sep. 2018, pp. 204–213.

[20] M. Lippi, S. Mariani, and F. Zambonelli, "Developing a "Sense of Agency" in IoT Systems: Preliminary Experiments in a Smart Home Scenario," in *2021 IEEE PerCom Workshops*. Kassel, Germany: IEEE, Mar. 2021, pp. 44–49.

[21] K. Fadiga, E. Houzé, A. Diaconescu, and J.-L. Dessalles, "To do or not to do: Finding causal relations in smart homes," *ArXiv210510058 Cs*, May 2021.

[22] G. Blair, N. Bencomo, and R. B. France, "Models@ run.time," *Computer*, vol. 42, no. 10, pp. 22–27, Oct. 2009.

[23] N. Bencomo, J. Whittle, P. Sawyer, A. Finkelstein, and E. Letier, "Requirements reflection: Requirements as runtime entities," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - ICSE '10*, vol. 2. Cape Town, South Africa: ACM Press, 2010, p. 199.

[24] G. Tesauro, D. M. Chess, W. E. Walsh, R. Das, A. Segal, I. Whalley, J. O. Kephart, and S. R. White, "A Multi-Agent Systems Approach to Autonomic Computing," *N. Y.*, p. 8, 2004.

[25] M. Warglien, P. Gärdenfors, and M. Westera, "Event structure, conceptual spaces and the semantics of verbs," *Theor. Linguist.*, vol. 38, no. 3-4, Jan. 2012.

[26] K. Bellman, J. Botev, A. Diaconescu, L. Esterle, C. Gruhl, C. Landauer, P. R. Lewis, A. Stein, S. Tomforde, and R. P. Wurtz, "Self-Improving System Integration - Status and Challenges after Five Years of SISSY," in *2018 IEEE 3rd International Workshops on Foundations and Applications of Self* Systems (FAS*W)*. Trento: IEEE, Sep. 2018, pp. 160–167.

[27] L. Bechberger and K.-U. Kuhnberger, "A Comprehensive Implementation of Conceptual Spaces," *ArXiv Prepr. ArXiv170705165*, p. 14, 2017.

[28] L. Steels and T. Belpaeme, "Coordinating perceptually grounded categories through language: A case study for colour," *Behav. Brain Sci.*, vol. 28, no. 4, pp. 469–489, Aug. 2005.

[29] A. Diaconescu, "Goal-oriented Holonic Systems," in *Organic Computing: Technical Systems for Survival in the Real World*, C. Muller-Schloer and S. Tomforde, Eds. Springer Intl. Pub., 2017, p. 54.