Data-Driven Environment Modeling for Self-Adaptive Systems

Yong-Jun Shin, Young-Min Baek, Eunkyoung Jee, Doo-Hwan Bae
School of Computing
Korea Advanced Institute of Science and Technology (KAIST)
Daejeon, Republic of Korea
{yjshin, ymbaek, ekjee, bae}@se.kaist.ac.kr

***Abstract*—A self-adaptive system (SAS) is one that is capable of adapting its behavior or structure to the changing environment. Many analysis and design approaches have been proposed to develop SASs systematically based on knowledge of the environment in order to deal with various and dynamic conditions of the environment. However, most existing approaches require designers to have considerable knowledge of the environment without practical guidelines. In addition, many SAS modeling studies do not consider the environment as a major concern, although the main goal of an SAS is an adaptation to the environment. In this study, we propose a method of generating environment models for SASs, by considering the environment as a first-class entity of the modeling process. To guide the analysis and understanding of the SAS environment, we propose a metamodel that encompasses the characteristics of the SAS environment. Based on the metamodel, environment models can be generated using historical environment data, which can be collected from existing similar systems or open data. As a case study, we apply our method to a smart traffic control system with real data, while following an existing approach to developing requirements of an SAS. We show that our proposed method can practically help designers to generate environment models with concrete features that are necessary for SAS modeling by considering the environment as a major entity of the modeling process.**

***Index Terms*—self-adaptive system, environment modeling, data-driven analysis, smart traffic system, modeling methodology**

I. Introduction

A self-adaptive system (SAS) is a system that has the ability to adapt its behavior or structure to the environment [1], [2]. Because the environment or operating conditions of modern systems are becoming complex and diverse, self-adaptability has become an important concept. For example, an autonomous robot should change its behavior according to its environment, where various obstacles may exist, for safe driving to a target [3]. The robot should adapt its behavior to its environment. An adaptive software-defined network should also adapt its structure to attain a stable throughput and latency [4]. Active research is being conducted on how to develop SASs that provide stable services and results in various environments.

There are many studies on SAS in terms of modeling, verification, validation, or control theory [5]–[8]. Modeling approaches address the earlier phases of SAS development including requirements, architectures, or behaviors of SASs. Various modeling methodologies and frameworks have been proposed, and one of the widely used and intuitive modeling methodologies is enumerating the possible environmental changes and corresponding models of the system for each condition [9]. The MAPE feedback loop provides a reference model of SASs with explicit processes of adaptation mechanism [10]. In addition, frameworks to support the SAS development, such as the Rainbow or the FUSION framework, were proposed [11], [12].

Most analysis and design approaches for SASs require designers to have some knowledge of the system environment [13]–[17]. To follow those modeling approaches, designers need domain knowledge of the environment or expert's help. It makes sense that the approaches are based on knowledge of the environment, but it is difficult to follow the approaches when the designers' knowledge is not sufficient. There is a lack of practical guidelines of analyzing the environment to support the SAS modeling approaches. As the environment where the system should adapt becomes more complex with many dynamic environmental factors, a practical modeling approach, which considers understanding the environment as one of the major steps in the modeling process, is necessary.

The environment has not been regarded as a first-class entity of SAS modeling and the concepts of the environment of SAS have not been explicitly addressed. Because the main purpose of SAS is adaptation to uncertain environments, the environment of the SAS itself needs to be dealt with thoroughly. The concept and characteristics of the implicitly agreed SAS environment need to be clarified. A method of dealing with the environment practically from the perspective of designers throughout the modeling process is required.

In this study, we propose a method of generating environment models of SASs, considering the environment as a first-class entity of the modeling process. To guide the analysis and understanding of the SAS environment, we propose a metamodel that encompasses the characteristics of the SAS environment. Based on the metamodel, environment models can be generated using historical environment data, which can be collected from existing similar systems or open data. As a case study, we apply our method to a smart traffic control system with real data, while following an existing approach to developing the requirements of an SAS. We show that our proposed method practically helps designers to generate

TABLE I
CHARACTERISTICS OF THE ENVIRONMENT OF SAS MODELING APPROACHES

Characteristics		The environment in the SAS modeling approach literature	Authors
Changes in environment	Anticipated changes in environment	Possible states of changing environments are enumerated. The system is modeled to monitor the given states of the environment.	Morandini *et al.* [14]
		Environmental conditions and uncertainties are modeled by domain knowledge. The model of environmental uncertainties guides the requirement elicitation.	Cheng *et al.* [15]
		The guarded rule, which is "if condition then updates" form is a basic transition rule. The condition, which will be monitored, is given by domain knowledge.	Arcaini *et al.* [13]
		Different domains, which mean possible changes in the environment, are enumerated. Separate models for each domain are combined with verification of global invariants.	Zhang *et al.* [9]
		Various environment factors and their conditions are described by fuzzy rules. The fuzzy rules are modeled by an extended Petri net.	Ding *et al.* [17]
	Unanticipated changes in environment	Control engineering approaches are used to deal with unpredictable changes in the environment.	Patikirikorala *et al.* [18]
		A genetic algorithm is used to deal with unexpected environment conditions for generating a behavior model of the system.	Goldsby *et al.* [3]
		A knowledge-based framework is used to select features of a system to deal with environmental conditions, which are not anticipated at design time.	Elkhodary *et al.* [12]
Effect on system goal		The environment causes some faults and from the fault scenarios and recovery activities of faults, the system modeling is derived.	Morandini *et al.* [14]
		Environmental uncertainties that may violate goals of an SAS are used to refine the goal model of the SAS.	Cheng *et al.* [15]
		An environment model is used to generate a goal model in SAS. A failure model is derived from the relationships between the environment model and the goal model.	Morandini *et al.* [16]
		A monitoring component monitors violations of system goals caused by environmental changes. A violation of a goal triggers the adaptation of the SAS.	Elkhodari *et al.* [12]

environment models with concrete features that are necessary for SAS modeling by considering the environment as a major entity of the modeling process.

The remainder of this paper is organized as follows. Section 2 introduces the related works. Section 3 introduces our modeling approach with the metamodel of the SAS environment and the environment model generation process. Section 4 explains a case study, and Section 5 concludes this study including its contributions and future work.

II. Related work

A. Characteristics of the environment of SAS

As indicated in Table I, several studies have considered the concepts of the environment that an SAS has to adapt to. Although there is a lack of concerted efforts on explicit definitions or concrete characteristics of the environment of SASs, it is possible to identify concepts that have been used implicitly. Table I summarizes the characteristics of the environments covered in the SAS modeling literatures. The literatures shows that the environment changes its status or behavior dynamically. The changes may or may not be anticipated at the design time. The changes in the environment affect the goals of the SAS; hence, the system should monitor its unexpected changes and respond accordingly. The characteristics are managed differently in each modeling approach but are commonly agreed upon in the literatures.

One of the main characteristics of the environment is that the environment dynamically changes its states or behaviors. The SAS has to adapt its behavior to the dynamic environment. Some of the modeling approaches assume that the changes in the environment are known or expected [9], [13]–[15], [17]. It does not mean that designers can predict how the environment will change in real time, but it means that designers can possibly identify variations that the environment can have. The possible changes are used as guides to the SAS modeling, and the SAS can be expressed as a sum of subsystems that can transfer their current modes following the conditions of the environment [19]. Each subsystem behaves for each condition of the environment and the transitions between subsystems enable adaptation.

Another characteristic of the environment is that the changes in the environment may not be anticipated at design time. Therefore, some modeling approaches do not try to specify the possible changes in the environment but focus on adaptation mechanisms [3], [12], [18]. The most representative adaptation mechanism is the MAPE feedback loop, which repeats the process of monitoring and analyzing the environmental changes, planning an adaptive solution, and manipulating adaptation effectors [10]. The SAS is regarded as a combination of a managed subsystem and a managing subsystem that monitors the dynamic environment [20]. To provide an efficient adaptation mechanism, many frameworks were proposed [11], [12], [21], [22]. Their common agreements are that there are changes that cannot be found at design time and that the SAS should be modeled to be able to respond to the changes during runtime.

Another characteristic is that the changes in the environment affect the achievement of system goals. The SAS has to adapt to the dynamic environment so that it achieves the goal consistently. There is a framework that triggers the system's adaptation while monitoring the violation of system goals [12]. Other modeling techniques use the failure model of the system or violations of the goals and their recovery activities [14]–[16]. Those modeling approaches consider managing the violation of goals or effects on the goals caused by the environment, as a purpose of the SAS.

There is another study that concerns with the internal mechanisms of the environment as well as with observable factors of the environment [23]. The study makes the behavioral

model of the environment from observable factors using a genetic algorithm and shows an adaptation of a system to the environmental behavior model. Although it is not a mainstream of SAS modeling approaches, it can provide more fundamental adaptation to the target environment.

B. Environment model for modeling approaches

The characteristics of the environment are important to the SAS modeling. Analyzing and understanding the environment are an important part of SAS modeling. Morandini, *et al.* [16] explicitly describe the environment modeling in the goal and failure modeling processes of the SAS. The authors provide a metamodel of the environment. However, the metamodel is only used in the goal and failure modeling processes; therefore, it is difficult to use it to provide a general understanding of the environment for other SAS modeling approaches. Although environment modeling is considered one of the important elements in their approach, the process of deriving the environment model is vague.

The environment has not been treated as a major concern in the SAS modeling, but in the field of multi-agent system (MAS) engineering, there is an approach to treat the environment as a first-class entity of the modeling process [24]. The approach defines the environment of the MAS with the roles and effects on the MAS. The authors consider the environment as a new design dimension by showing the overall process of MAS engineering. They explain the roles of the environment on the requirement specification, analysis, and design of the MAS. Other studies in MAS engineering deal with the environment explicitly and include environment modeling in the system design process [25]–[28].

The environmental model used in the MAS studies does not focus on the characteristics of the SAS environment discussed in the above SAS modeling approaches. In the field of SASs, the environment is a trigger and a reason for the adaptation. However, despite its importance, the environment is not treated as important and explicit for SAS modeling. An SAS modeling approach that considers the environment as a major entity is necessary. In this study, we show how to model the environment to provide sufficient understanding of it for SAS modeling in a practical way.

III. ENVIRONMENT MODEL GENERATION FOR SAS MODELING

A. Overall approach and illustrative example

We propose a data-driven environment modeling method for SASs. To provide a practical method of analyzing the environment for SAS modeling, our method guides designers to generate environment models using historical data of the environment. This method provides processes to analyze and understand the environment, assuming that designers lack of the domain knowledge of the environment and has little data analysis capabilities. We also propose a metamodel of the environment so that designers can generate environment models that fully reflect the characteristics of the SAS environment. The generated environment model can be used for SAS modeling.

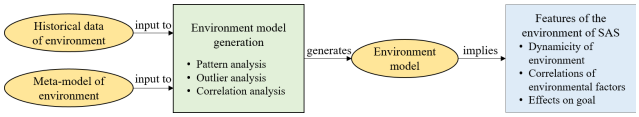

Fig. 1. Overview of environment modeling for SASs

Figure 1 shows the overall flow of our approach. Input historical data of the environment represents the environment's known histories such as raw data of sensors. The input data are used to generate an environment model based on the metamodel, which is another input of this method. The environment model generation process includes data analysis techniques such as pattern, outlier, and correlation analysis. The generated environment model represents some concrete features of the environment, such as dynamicity of the environment, correlations of environmental factors, and effects on goal. The extracted and concretized features are used to support designers in analyzing the environment so that they can practically follow the SAS modeling approaches based on such understanding.

The historical data of the environment are one of the inputs of our approach for environment model generation. An SAS designer may not be an expert in a domain of the environment. Actual data of the environment support the designer to analyze the environment and provide concrete domain knowledge so that the designer can practically generate the environment model. In addition, in the big data era, it becomes easier to acquire environmental data, which are related to the system of interest. This method fully utilizes this data-rich situation for SAS modeling.

The metamodel is another input. The metamodel is derived based on a conceptual model of the SAS environment. The metamodel guides the model generation process so that the generated environment model provides features to be used for SAS modeling. The metamodel represents changes in the environment, their effect on system goals, and the internal mechanism within the range known from the given data.

We illustrate the proposed approach by using a smart traffic control system, which adapts its traffic signal to the dynamic traffic conditions to reduce traffic jam, as an example. The smart traffic control system is one that controls traffic condition by changing signals or policies automatically for a safe and efficient traffic flow. Traffic control systems are one of the important infrastructure in smart cities and has been used as examples of SASs surrounded by dynamic traffic flow and conditions [13], [29], [30]. The traffic environment dynamically changes over time or by specific events. Thus, a dynamic and uncertain traffic should be carefully considered for the optimal reaction of a smart traffic control system.

A designer, who needs to develop an adaptive traffic control system, can analyze the traffic environment with traffic data, even if he/she is not a traffic expert. The metamodel helps the designer to generate a traffic environment model by guiding

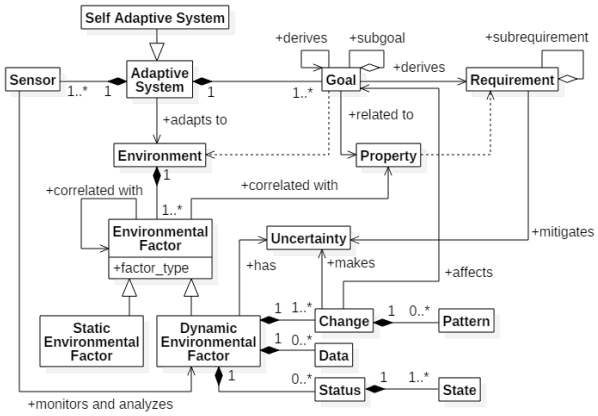

Fig. 2. A conceptual model of environment for SASs

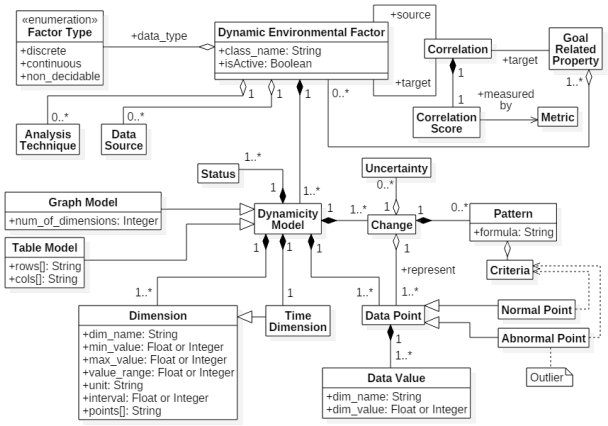

Fig. 3. A metamodel of environment for SASs

the required analysis of the traffic data. Because the metamodel reflects the features that are necessary for SAS modeling, the generated traffic environment model captures the features of the environment and also provides knowledge on the traffic environment for designing smart traffic system adaptive to various traffic conditions.

B. Metamodel of environment for SAS

We propose a conceptual model and a metamodel of SAS environments. The conceptual model represents a relationship between the SAS and the environment. Based on the conceptual model, we propose the metamodel for guiding the environment model generation.

Figure 2 shows a conceptual model of the environment for SASs. An *adaptive system* adapts its behavior to the surrounding *environment*. In particular, an SAS decides the adaptation autonomously or with minimal intervention [31]. To decide when and how to adapt to the environment, the SAS has one or more *sensor*s to monitor the changing environment. It is referred to as monitoring and analysis. The SAS has its *goal*s to achieve regardless of changes in the environment. A goal may also derive another goal or has a hierarchical relationship with subgoals. A goal is related to a *property* that allows verifying whether or not the goal is achieved.

The environment of an adaptive system consists of a number of *environmental factor*s. For example, the traffic environment can be conceptualized as a set of average speed and the number of cars on the roads of interest. An environmental factor may have a correlation with another factor, and it could be static or dynamic, but *dynamic environmental factor*s are dealt with in greater depth for SASs. A dynamic environmental factor changes its *status* and may have *pattern*ed *change*s such as daily traffic fluctuations. Changes in a dynamic environmental factor generate *uncertainty* and affect the SAS's goal.

Figure 3 shows a metamodel of the environment for the SAS that we propose based on the conceptual model. An environment model consists of a number of *dynamic environmental factor* models. A dynamic environmental factor model has its *factor type* as discrete, continuous, or non-decidable. For example, the average speed of cars on a road is a continuous factor, but the available/blocked state of a road is a discrete factor. Additionally, a dynamic environmental factor model has its optional information such as *analysis technique*s and *data source* of the environmental factor.

A *dynamicity model* is a submodel of the dynamic environmental factor that represents information about changes in the environmental factor. Based on the environmental data, it has *dimension*s of the data, such as the time when the traffic data was measured. Because *time* is the most dominant dimension of data, it is a default dimension of a dynamicity model. The environmental *data point*s with changing values represent the *change*s in the environmental factor and the changes may have a *pattern* with its *criteria*. If the environmental factor shows a patterned change, each data point can be categorized as a *normal* or *abnormal point*. Because abnormal points also provide insights regarding unexpected or uncertain changes in the environment, it should be carefully managed during SAS analysis and design. For example, if there is a patterned daily traffic fluctuation, there can be abnormal traffic situations when there is a festival in the city. The smart traffic control system should also adapt to such abnormal situation. In addition, the dynamicity of the environmental factor can be represented as a *graph* or *table model* to help the designer visually analyze the situation.

A dynamic environmental factor may have a *correlation* with another factor, and an explicit expression of this relationship implies internal mechanisms of the environment within the range of known historical data. For example, the correlation of the number of passing cars of two roads implies a flow of traffic. The correlations of environmental factors are analyzed and recorded in the dynamic environmental factor model. In addition, a factor model also represents its correlation to a *goal-related property* to express an effect of the environmental factor on the SAS's goal. A traffic of a road's effect on the traffic jam of an area can be imitated as a correlation between the number of cars passing the road and the average occupancy of the area.

The metamodel shows features of the SAS environment. The information on historical data, axes of data, changes

in environmental factors, coefficients between environmental factors, and relationships between the environment and the goal-related properties are shown in the metamodel. Some of the features are determined by the historical data, and the others are analyzed by data analyst. In the next section, we will explain how an environment model is generated using historical data based on the metamodel.

C. Environment model generation process

Prior to the environment model generation, the historical data of the environment should be acquired and organized for analysis. With the help of big data and trends to open and share them, we can acquire environmental information or parts of them. Acquiring historical data is not included this study. The input historical data should be given with specific environmental factors. The environmental data, which are organized as a set of environmental factors' data, become the basis of the environment model. The next step is to identify the features of each factor by data analysis.

In the metamodel, the factor type, analysis technique, data source, and dimension information are determined by the form of acquired data. The average speed of cars on a road is a continuous factor and its data source is an open traffic data repository. The analysis technique of a factor can be statistical methods or algorithms that will be explained later. Other features of the metamodel should be extracted or processed through data analysis. They can be obtained by simple data analysis techniques that are basically provided by popular data analysis libraries such as Python NumPy, SciPy or Pandas. The process of generating an environment model by data analysis consists of patterned change analysis, exceptional change analysis, correlation analysis between environmental factors, and correlation analysis between an environmental factor and a goal-related property.

The changes in each environmental factor and its patterns should be analyzed first. Regarding a data point as a status of the environmental factor, the changes or variations in the status can be analyzed as a pattern if they have a distinct dominant flow of changes. The pattern is expressed by a polynomial equation or state-based machine if the status of the factor is represented as a continuous value or a discrete value, respectively. Polynomial regression can be used to generate a pattern of continuous state changes as an equation. State-based model generation, such as the Markov-chain model, can be used to express discrete changes of environmental factors. For example, the pattern of change in the number of cars can be expressed as an equation by taking the time as the input and the number of cars as the output.

The next step is to identify abnormal data. Even though the changes are represented as a pattern, there may exist some outliers that do not match with the representations. Those exceptional data of the environment should be carefully considered in SAS modeling to prepare for unexpected situations. Abnormal data provide insights regarding the necessary recovery activities of an SAS to handle unexpected conditions. A specific threshold or outlier classification method such as using the z-score can be used to identify an outlier. The number of cars during a festival day could be identified as an outlier compared to the usual pattern.

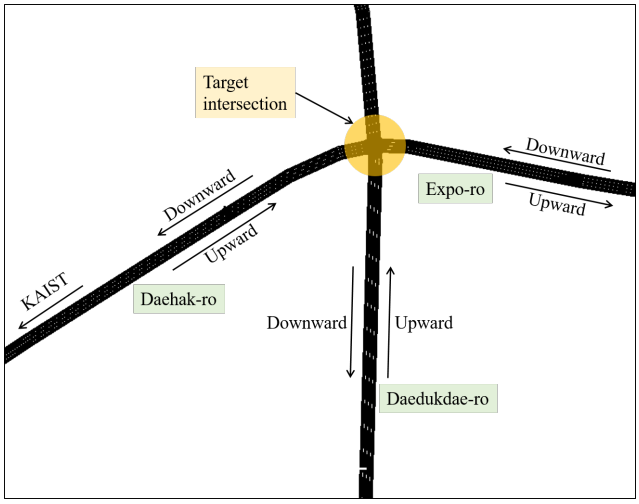

Fig. 4. A map around KAIST

After analyzing the changes in environmental factors, analyzing the correlations of the environmental factors is conducted. Typically, the correlation of two factors can be analyzed using the Pearson correlation coefficient. The correlation score of two factors have a value between 1 and 1, where 1 indicates a positive correlation, and 1 indicates a negative correlation. For example, if a correlation score of the number of cars on two roads is close to 1, the correlation can be used to predict the traffic change of one road by monitoring another road. In addition, in the SAS, an environmental factor is directly related to a monitoring component or a sensor. Therefore, the correlation of each environmental factor could determine the relationships between the values of the sensors in the SAS.

Finally, the effects on goal achievement or violation by environmental factors are analyzed using the same correlation analysis method of the previous step. The properties of SAS's goals are reflected indirectly in the monitored environmental factors [4], [8]. The correlation score between the environmental factors and the goal-related property identifies the effect of the environmental factor on the SAS's goal. For example, if a Pearson correlation score of a specific road traffic condition to an area of traffic flow is higher than that of the other roads, adapting to that road's traffic condition may have higher priority than that of the other roads.

With these steps, we can obtain concrete features of the environment. The features can support basic understanding of the environment and additional data analysis can be conducted to generate a more sophisticated environment model. The understanding and insights provided by the environment model that this method generates can explicitly support the SAS modeling. In the case study, we will show how our environment model generation method and the environment model assists an SAS modeling approach with a real-case scenario.

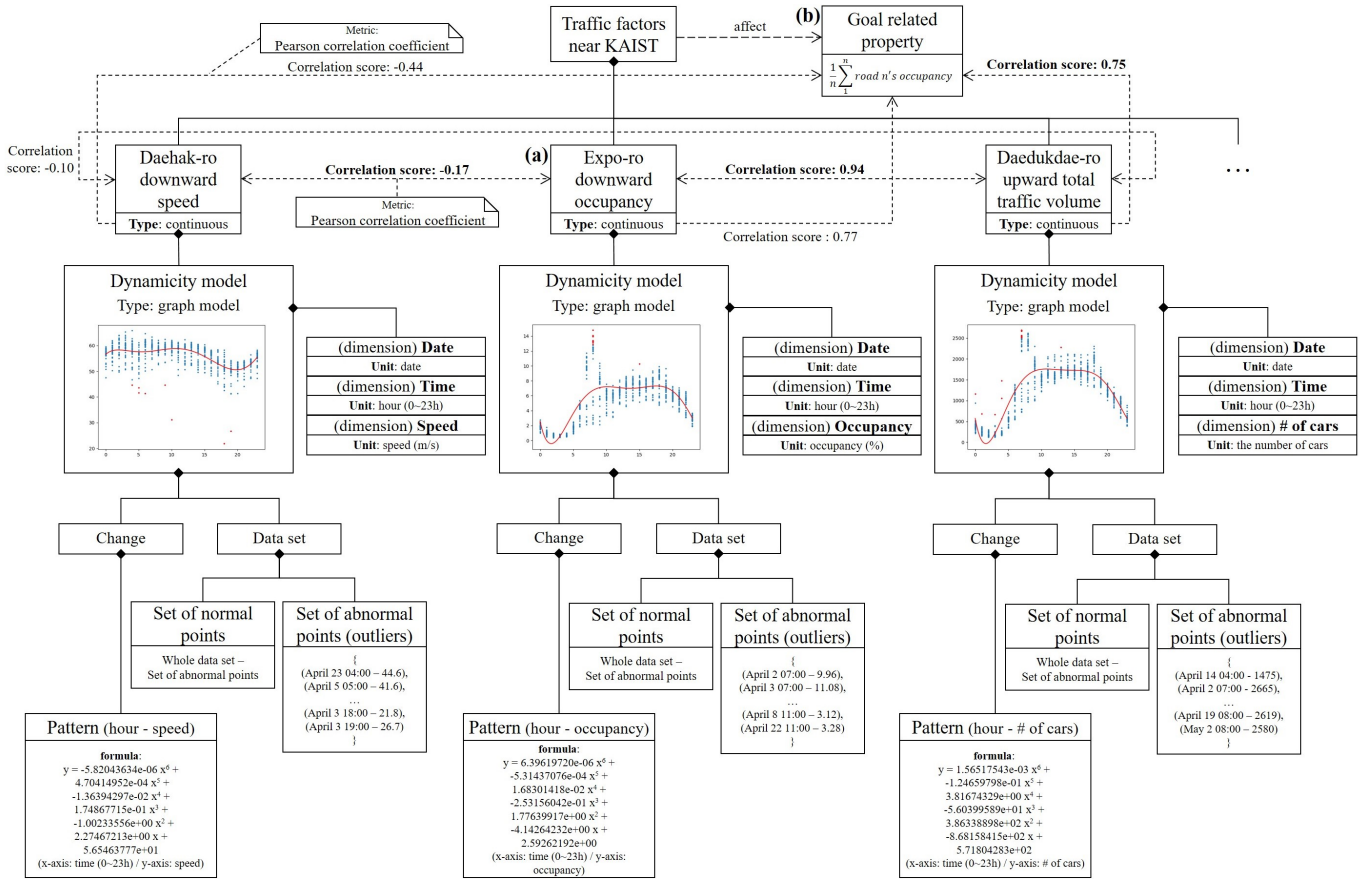

Fig. 5. A traffic environment model generated using the given historical traffic data

IV. Case Study: Smart Traffic Control System

A. Scenario: Daily traffic jam near Korea Advanced Institute of Science and Technology (KAIST)

To apply our approach to a realistic example, we show the smart traffic control system design in a case study. For a more detailed scenario, we adopt a specific target environment near KAIST, Daejeon, South Korea. Figure 4 shows the map around KAIST. The traffic of the area varies with time and causes a huge traffic jam every day. The smart traffic control system involved in this case study is a traffic signal controller of the target intersection of three roads, as shown in Figure 4, which needs to dynamically adapt to reduce the daily traffic jam of the area.

We acquired the historical traffic data of the target area from the Daejeon Transportation Data Warehouse system (DTDW 2.0), which is managed by Daejeon City[1]. We collected a month's worth of available data. The data set contains collected data from both directions (upward and downward) of the three roads (Daehak-ro, Expo-ro, and Daedukdae-ro), which are adjacent to the target intersection. For each road, the data set contains the average speed, the number of cars, and the average occupancy of the road. The acquired data set, which contains 18 factors (three factors of two directions of three roads), is used to generate the environment model.

B. Traffic environment model generation

The given historical data are composed of raw data of the 18 environmental factors. The format of the data is a set of tuples that contain a measuring date and time and a value, i.e., the average speed, the number of cars, or the average road occupancy. We generated the traffic environment model of the given historical data of the target intersection following our model generation method. All data analyses were performed using the basic functions of Python NumPy library[2]. A part of the generated environment model including only three factors is shown in Figure 5.

The changes in each traffic factor are analyzed and modeled by extracting patterned changes and their outliers. The patterns of changes were found by polynomial regression based on the given data set because the types of all traffic factors are continuous. In Figure 5, each traffic factor has information about its dynamicity as a data set of raw data, an extracted equation to express the pattern, and a graphical representation. The extracted pattern shows the changes in average speed, occupancy, and traffic volume with time, taking the x-axis as time and the y-axis as speed (m/s), occupancy (%), or the

[1]DTDW 2.0 - http://tportal.daejeon.go.kr

[2]NumPy - http://numpy.org

number of cars, respectively. Through the generated model, we can observe the changes in traffic condition around KAIST. The model shows the actual situation, which is repeated every day.

Data points that do not match the extracted patterns were classified as outliers. The outliers show the possibilities of changes that do not follow the analyzed and expected changes in the traffic factors. The outliers were classified according to the z-score, which is easy to use for outlier classification within a data set[3]. In a set of data points that have the same measuring time, if the absolute value of a data point z-score is greater than 2, it was classified as an outlier. These data are represented as a set of abnormal points (red dots in the graph model in Figure 5). Identified outliers do not belong to the main pattern of traffic factor change but can show possible changes that the traffic control system has to adapt.

Correlations between traffic environmental factors were analyzed. In Figure 5, the correlation between two factors is represented by a bidirectional arrow with a correlation score. The correlation was evaluated using the Pearson correlation coefficient. Factor (a) in Figure 5, Expo-ro downward occupancy shows a high positive correlation (0.94) with Daedukdae-ro upward total traffic volume, but shows a low negative correlation (-0.17) with Daehakro downward speed. Figure 6 shows a heat map of correlations for all traffic factors. The analyzed correlations between traffic factors provide insights into the mechanisms of the changes in the traffic environment.

Correlations between environmental factors and the goal of the target system are also analyzed. The goal-related property is expressed as the average of all road occupancies in element (b) in Figure 5. In Figure 5, the environment model shows that the Daedukdae-ro upward total traffic volume has a positive correlation (0.75) with the goal. Although it is not included in Figure 5, the Daedukdae-ro downward speed has the greatest effect on the goal among all the analyzed factors except for occupancies with high negative correlations (-0.89). It indicates that adapting to the Daedukdae-ro downward speed, which has the highest correlation with the goal, is relatively more important for the smart traffic control system of the target intersection.

We have shown how to generate a traffic environment model using given historical data. This traffic environment model can provide designers with specific information about the features of the traffic environment. Authors without expert knowledge on traffic were able to understand the situation of the target intersection by following the data analysis process of the environment model generation. Next, we show an example of how the generated environment model of traffic can be used in the SAS analysis and design methods.

C. Application to requirement modeling

We apply the traffic environment model to develop the requirements of the smart traffic control system because the

[3] $z = \frac{x-\mu}{\sigma}$, where μ is the mean of the population, and σ is the standard deviation of the population. A data point, whose absolute value of z-score is larger than a threshold, is regarded as an outlier.

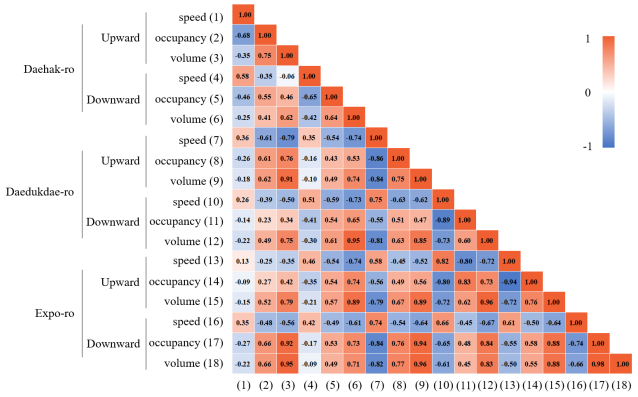

Fig. 6. Correlations between given traffic factors

requirement analysis is the foremost phase of the software development and it can show a conceptual perspective of the system. Cheng *et al.* proposed a goal-based modeling approach to develop the requirements of SAS [15]. The approach derives requirements using the uncertainty model of the environment. It is suitable for our case study because the uncertainty model can be derived from the dynamicity model of the traffic environment. In addition, the features of the traffic environment model helps to elicit requirements. Therefore, we apply the generated traffic environment model to the requirement modeling as a case study.

This is the summarized process of developing the requirements of the SAS proposed by Cheng [15].

- Step 0: Identify the high-level requirements and the environment
- Step 1: Derive requirement models
- Step 2: Identify the uncertainty factors of the environment
- Step 3: Refine requirements to mitigate the uncertainty factors
  - i No refinement
  - ii Add low-level subrequirements
  - iii RELAX requirements
  - iv Add high-level requirements

Steps 0 to 2 involve identifying the first set of requirements and uncertainties of the environment, which will be used to refine the requirements in Step 3. The uncertainty factors of the environment are possible changes or variations in the environment that can prevent a requirement or a goal from being met. Step 3 is a repeated process to refine requirements until there are no more requirements to be refined. Step 3(iii) is provided to make an unsatisfiable requirement more flexible that is not able to be refined by adding lower or higher requirements to deal with an uncertainty factor of the environment. Making the requirements more flexible is supported by RELAX, a requirement language for SAS [32].

In Cheng's study, a generated goal model by the process was shown and each element in the goal model specifies a requirement. In our study, we directly show the generated requirements by the process on the requirement diagram of SysML. SysML is a modeling language extending parts of UML [33] and adding a newly defined requirement diagram.

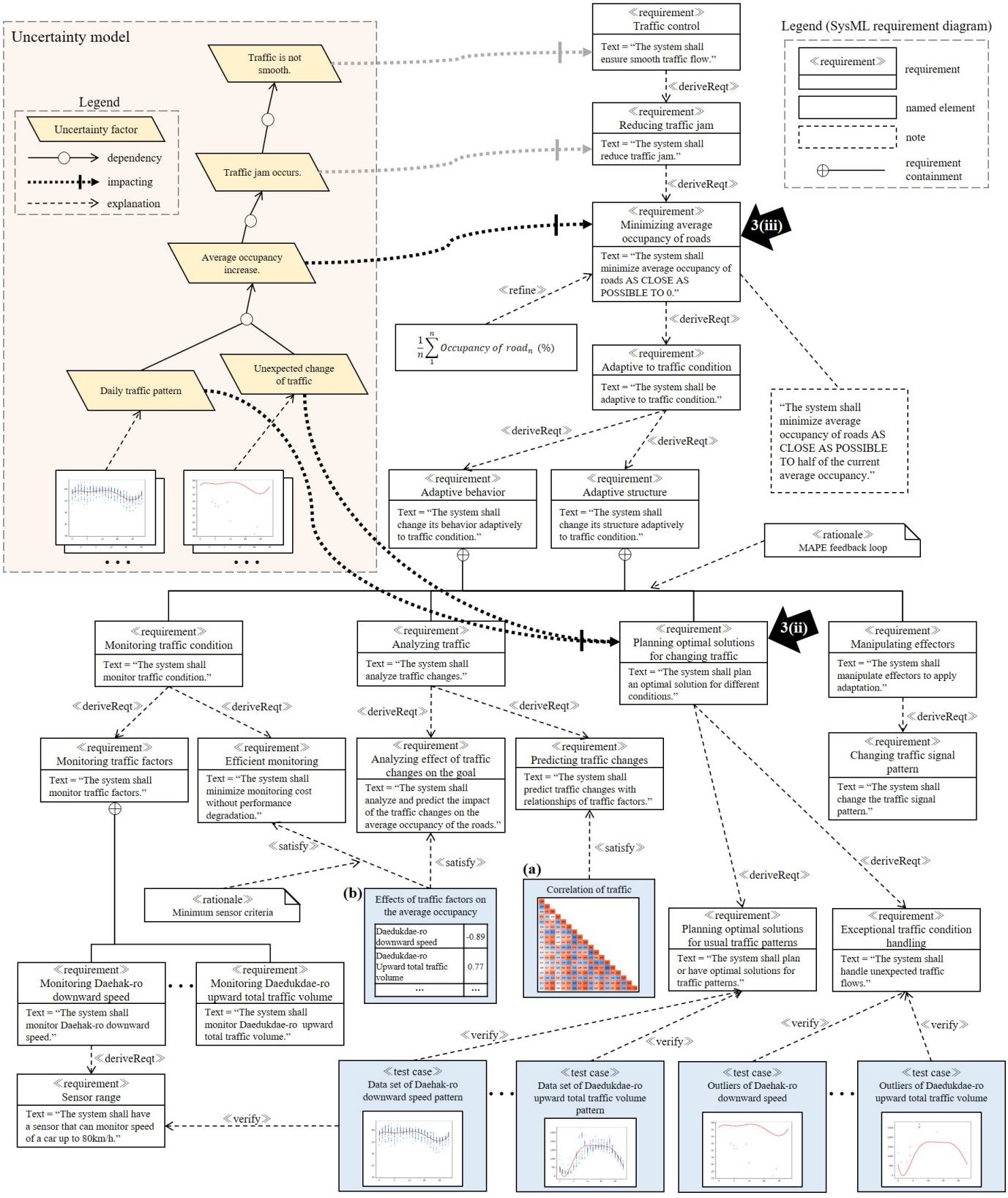

Fig. 7. An uncertainty model of the environment and a requirement diagram of the smart traffic control system

The requirement diagram of SysML supports defining requirements and relationships in a formal representation and specifying additional element, data, or rationale related to the requirements. Therefore, we show the result of following the requirement modeling approach using the requirement diagram of SysML.

Figure 7 shows the uncertainty model and the developed requirement diagram. The upper left diagram shows the uncertainty factors that prevent the satisfaction of the requirements of the smart traffic control system. Identifying the uncertainty factors required by this modeling approach can be easily achieved by referring to the dynamicity model in the generated traffic environment model. In addition, the generated environment model can provide concrete data such as the pattern of changes or data sets. The identified uncertainty factors are used to elicit requirements.

Step 3(iii) redefines a requirement that is not satisfiable due to the effect of an uncertainty factor. Step 3(iii) in Figure 7 makes the requirement more flexible with a relaxed value of the average occupancy of the current traffic, which is a specific information in the traffic environment model. Based on the generated environment model, the concrete understanding guided a flexible and realistic requirement.

Step 3(ii) is adding a subrequirement to mitigate uncertainty factors, which are not sufficiently dealt with by the current requirements. Step 3(ii) in Figure 7 shows that the requirement is not specific in addressing patterned and abnormal changes in the environment analyzed in the traffic environment model. Therefore, two subrequirements were derived based on the information from the traffic environment model. In addition, test cases that can be used for the verification of the generated requirements were extracted from the traffic environment model. The "test case" and "verify" are stereotypes of the SysML notation. We could define these requirements along with the Cheng's process and the knowledge of the traffic by the environment model.

Not limited to Cheng's process, features of the traffic environment model were used to elicit other requirements of the smart traffic control system. Element (a) in Figure 7 shows the analyzed correlations of traffic factors, which are used to satisfy a requirement to predict the traffic changes. Because the correlations between the environmental factors allow a glimpse of the internal mechanisms of the environmental change, it can be used to analyze the environment of the SAS. Element (b) in Figure 7 shows the analyzed effects of environmental factors on the goal-related property, which are used to satisfy a requirement related to analyzing the effect of traffic changes. They are also used to support prioritizing sensors for efficient monitoring of the smart traffic control system.

In the case study, we generated a traffic environment model for a smart traffic control system modeling. We collected open data of real traffic environment for the model generation. We generated a traffic environment model by using the acquired historical traffic data and demonstrated that our method can be a practical guideline for generating an environment model that provides concrete features of the SAS environment. The generated traffic environment model was applied to the requirement modeling. It was shown that the environment model can provide effective and concrete insights into the modeling process. Although this case study only showed the requirement elicitation phase, the environment model is expected to provide assistance to designers throughout the entire SAS design process.

V. Threats to validity

Our environment model generation method does not show specific input data types or required data analysis techniques. These can be additional decisions that the designers should make to apply this environment model generation method for SAS modeling. However, our method has a significance in showing the domain-general process of environment model generation. In this study, we proposed an environment metamodel considering the features that the environment model should have for SASs and provided guidelines for model generation based on the metamodel. We also showed the basic data analysis process required for the method.

This method helps designers who lack knowledge of the domain to have a minimum understanding of the environment that is essential for SAS modeling with the help of historical data. The types of data that can be acquired will differ according to the domain; thus, the restriction of data is not recommended. This method also assumes little knowledge of the designer's data analysis techniques. Therefore, only the minimum requirements for the data analysis process are shown. In addition, the functions that are basically provided by popular data analysis libraries are used in the case study. Because of the features of this method that suggest a domain-general and basic data-driven model generation process, we do not represent this method in the form of an environment model generation algorithm that can be automated. It is also important to provide baselines of the metamodel and a process that can be extended with additional domain knowledge or analysis techniques.

VI. Conclusion

An SAS modeling approach considering the environment as a first-class entity is necessary to attain deep understanding of the environment during the design of an SAS. We proposed a data-driven environment modeling method for SASs. The proposed method is based on a metamodel that provides concrete features to be used for SAS modeling. The environment model is generated using historical data of environment based on the data analysis guidelines and the metamodel. In the case study of the smart traffic control system, we showed that our method can guide designers to generate an environment model. The generated traffic environment model was used to elicit requirements of the smart traffic control system. We have shown that our approach can practically help designers to obtain the minimum understanding of the environment required for the SAS modeling with simple data analysis methods.

Our approach is based on the historical data of the environment related to the SAS, and acquiring data is becoming easier with big data and expansion of data publicity. In the future, a modeling methodology, which includes generating models of an SAS using the environment model, will be studied. A guideline to derive the behavioral or architectural model of an adaptive software from the environment model is necessary. In addition, following the modeling methodology, a domain-general tool to support environmental data-driven SAS modeling is necessary to reduce the burden of designers.

REFERENCES

[1] B. H. Cheng, R. de Lemos, H. Giese, P. Inverardi, J. Magee, J. Andersson, B. Becker, N. Bencomo, Y. Brun, B. Cukic *et al.*, "Software engineering for self-adaptive systems: A research roadmap," in *Software engineering for self-adaptive systems*. Springer, 2009, vol. 5525, pp. 1–26.

[2] R. De Lemos, H. Giese, H. A. Müller, M. Shaw, J. Andersson, M. Litoiu, B. Schmerl, G. Tamura, N. M. Villegas, T. Vogel *et al.*, "Software engineering for self-adaptive systems: A second research roadmap," in *Software Engineering for Self-Adaptive Systems II*. Springer, 2013, pp. 1–32.

[3] H. J. Goldsby and B. H. Cheng, "Avida-mde: a digital evolution approach to generating models of adaptive software behavior," in *Proceedings of the 10th annual conference on Genetic and evolutionary computation*. ACM, 2008, pp. 1751–1758.

[4] A. J. Ramirez and B. H. Cheng, "Evolving models at run time to address functional and non-functional adaptation requirements," in *Proceedings of the 4th International Workshop on Models at Runtime*, 2009.

[5] J. Andersson, R. De Lemos, S. Malek, and D. Weyns, "Modeling dimensions of self-adaptive software systems," in *Software engineering for self-adaptive systems*. Springer, 2009, pp. 27–47.

[6] N. Li, D. Bai, Z. Yang, and W. Jiao, "Verifying stochastic behaviors of decentralized self-adaptive systems: A formal modeling and simulation based approach," *arXiv preprint arXiv:1706.08271*, 2017.

[7] G. Tamura, N. M. Villegas, H. A. Müller, J. P. Sousa, B. Becker, G. Karsai, S. Mankovskii, M. Pezzè, W. Schäfer, L. Tahvildari *et al.*, "Towards practical runtime verification and validation of self-adaptive software systems," in *Software Engineering for Self-Adaptive Systems II*. Springer, 2013, pp. 108–132.

[8] A. Filieri, M. Maggio, K. Angelopoulos, N. D'Ippolito, I. Gerostathopoulos, A. B. Hempel, H. Hoffmann, P. Jamshidi, E. Kalyvianaki, C. Klein *et al.*, "Software engineering meets control theory," in *Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE Press, 2015, pp. 71–82.

[9] J. Zhang and B. H. Cheng, "Model-based development of dynamically adaptive software," in *Proceedings of the 28th international conference on Software engineering*. ACM, 2006, pp. 371–380.

[10] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, no. 1, pp. 41–50, 2003.

[11] D. Garlan, S.-W. Cheng, A.-C. Huang, B. Schmerl, and P. Steenkiste, "Rainbow: Architecture-based self adaptation with reusable infrastructure," *Computer Science Department*, p. 668, 2004.

[12] A. Elkhodary, N. Esfahani, and S. Malek, "Fusion: a framework for engineering self-tuning self-adaptive software systems," in *Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering*. ACM, 2010, pp. 7–16.

[13] P. Arcaini, E. Riccobene, and P. Scandurra, "Modeling and analyzing mape-k feedback loops for self-adaptation," in *Proceedings of the 10th international symposium on software engineering for adaptive and self-managing systems*. IEEE Press, 2015, pp. 13–23.

[14] M. Morandini, L. Penserini, and A. Perini, "Towards goal-oriented development of self-adaptive systems," in *Proceedings of the 2008 international workshop on Software engineering for adaptive and self-managing systems*. ACM, 2008, pp. 9–16.

[15] B. H. Cheng, P. Sawyer, N. Bencomo, and J. Whittle, "A goal-based modeling approach to develop requirements of an adaptive system with environmental uncertainty," in *International Conference on Model Driven Engineering Languages and Systems*. Springer, 2009, pp. 468–483.

[16] M. Morandini, L. Penserini, A. Perini, and A. Marchetto, "Engineering requirements for adaptive systems," *Requirements Engineering*, vol. 22, no. 1, pp. 77–103, 2017.

[17] Z. Ding, Y. Zhou, and M. Zhou, "Modeling self-adaptive software systems by fuzzy rules and petri nets," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 2, pp. 967–984, 2018.

[18] T. Patikirikorala, A. Colman, J. Han, and L. Wang, "A systematic survey on the design of self-adaptive software systems using control engineering approaches," in *Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2012 ICSE Workshop on*. IEEE, 2012, pp. 33–42.

[19] J. Zhang and B. H. Cheng, "Specifying adaptation semantics," *ACM SIGSOFT Software Engineering Notes*, vol. 30, no. 4, pp. 1–7, 2005.

[20] D. Weyns, B. Schmerl, V. Grassi, S. Malek, R. Mirandola, C. Prehofer, J. Wuttke, J. Andersson, H. Giese, and K. M. Göschka, "On patterns for decentralized control in self-adaptive systems," in *Software Engineering for Self-Adaptive Systems II*. Springer, 2013, pp. 76–107.

[21] J. Siljee, I. Bosloper, J. Nijhuis, and D. Hammer, "Dysoa: making service systems self-adaptive," in *International Conference on Service-Oriented Computing*. Springer, 2005, pp. 255–268.

[22] N. M. Villegas, G. Tamura, H. A. Müller, L. Duchien, and R. Casallas, "Dynamico: A reference model for governing control objectives and context relevance in self-adaptive software systems," in *Software Engineering for Self-Adaptive Systems II*. Springer, 2013, pp. 265–293.

[23] J. R. Williams, S. M. Poulding, R. F. Paige, and F. Polack, "Exploring the use of metaheuristic search to infer models of dynamic system behaviour." *MoDELS@ Run. time*, vol. 1079, pp. 76–88, 2013.

[24] A. Molesini, A. Omicini, and M. Viroli, "Environment in agent-oriented software engineering methodologies," *Multiagent and Grid Systems*, vol. 5, no. 1, pp. 37–57, 2009.

[25] D. Weyns, A. Omicini, and J. Odell, "Environment as a first class abstraction in multiagent systems," *Autonomous agents and multi-agent systems*, vol. 14, no. 1, pp. 5–30, 2007.

[26] C. Bernon, V. Camps, M.-P. Gleizes, and G. Picard, "Engineering adaptive multi-agent systems: The adelfe methodology," in *Agent-oriented methodologies*. IGI Global, 2005, pp. 172–202.

[27] A. Molesini, A. Omicini, E. Denti, and A. Ricci, "Soda: A roadmap to artefacts," in *International Workshop on Engineering Societies in the Agents World*. Springer, 2005, pp. 49–62.

[28] A. Molesini, A. Omicini, and M. Viroli, "Environment in agent-oriented software engineering methodologies," *Multiagent and Grid Systems*, vol. 5, no. 1, pp. 37–57, 2009.

[29] T. Tristono, S. D. Cahyono, and P. Utomo, "Model petri net of adaptive traffic lights and its collaboration with a special event," in *MATEC Web of Conferences*, vol. 147. EDP Sciences, 2018, p. 02005.

[30] M. U. Iftikhar and D. Weyns, "A case study on formal verification of self-adaptive behaviors in a decentralized system," *arXiv preprint arXiv:1208.4635*, 2012.

[31] F. D. Macías-Escrivá, R. Haber, R. Del Toro, and V. Hernandez, "Self-adaptive systems: A survey of current approaches, research challenges and applications," *Expert Systems with Applications*, vol. 40, no. 18, pp. 7267–7279, 2013.

[32] J. Whittle, P. Sawyer, N. Bencomo, B. H. Cheng, and J.-M. Bruel, "Relax: Incorporating uncertainty into the specification of self-adaptive systems," in *Requirements Engineering Conference, 2009. RE'09. 17th IEEE International*. IEEE, 2009, pp. 79–88.

[33] S. Friedenthal, A. Moore, and R. Steiner, *A practical guide to SysML: the systems modeling language*. Morgan Kaufmann, 2014.