ISSN 2278-3091
Volume 10, No.3, May - June 2021
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse311032021.pdf
https://doi.org/10.30534/ijatcse/2021/311032021

Domain Specific Model Driven Approach for Adaptive Systems

Magagi Ali Bachir[1*], Jellouli Ismail[1], El Garouani Said[2], Amjad Souad[1]
[1]Abdelmalek Essaadi University, Morroco
[2]Sidi Mohamed Ben Abdellah University, Morocco
*magagi.bachir1@gmail.com

ABSTRACT

An adaptive system is any system that can self-conform according to changes that occur in his environment. Self-adaptation includes self-reconfiguration, self-restructuring, self-repair, self-optimization or all at the same time. The realization of this kind of systems, in spite of the efforts made, suffers from a deficiency of engineering approaches. One of the most promising techniques in this quest is model-driven engineering. In the model-driven engineering paradigm, the model is the backbone of the systems engineering process. In this paper, we outline a model-based approach that offers a way to explicitly design self-adapting standard systems. We define it based on the UML profiling technique which allows to specify models for the most application domain frameworks. Through this profile we clearly define the components involved in the management of adaptation of systems, as well as the relationships between them. We present, for practical validation, an example application based on the approach.

Key words: Adaptive system, Model Driven Approach, Model Driven Engineering, UML Profile, Meta-Modeling, MAPE-K Control Loop.

1. INTRODUCTION

Computer systems are becoming more and more invasive in everyday life[1]. They allow us to easily and quickly perform some essential tasks. But these systems are very often called to operate in unstable (changing) environments. They can thus be confronted with problems related to these environments. Hence the need to equip these systems with the capacity to conform to them by modifying their structures or their configurations, one speaks about adaptation or self-adaptation. A self-adaptive system is a system that can change its own structure or configuration to respond to an unpredictable change in its environment. These changes may include a lack of resources required by the system, overuse of the system, or a threat to its operation or security. What will force in the first case the system to do without the missing resource and to continue to work, is what is called self-organization. In the second case the system will be forced to improve its performance to meet the very high demand that it faces, this is called self-optimization. In the case of the threat the system must try to make corrections to ensure the continuity of the service which it is supposed to render, this action is what is called self-repair. All these actions are included in what is called the self-adaptation of the systems.

To help solve the problem of adaptation, some researchers in computer science and more particularly in model-driven engineering [2] have focused their research on this axis. As a result, many approaches have been proposed, among which EUREMA [3],PLASMA [4],MVC-IMASAM[5] to name a few. These current methods do not allow to take into account all aspects of adaptation. Most are interested in a few very particular aspects, and the more general methods offer a very high complexity making difficult their handling, which leaves this field of research very interesting and still explorable.

The remainder of this document is structured as follows:

We will explain in the first parts the contextualization of our work, the fundamentals of the approach. We will present the technique used in section 3. In section 4 we will present the basic concept of the approach. The section 5 will present the global architecture of the approach. The section of Result (section 6) will present the definition of the UML profile[6] to control the approach and a concrete example to show how the approach can be used. In the section 7 we will present the previous similar works. The section 8 came to present a general conclusion of the work.

2. CONTEXTUALIZATION

Many of the model-driven engineering work has shown the inadequacy of UML [7]language in the design of certain types of information systems. Self-adaptive systems because of their complexities are no exception; many researchers are studying this type of systems and have therefore proposed many methods for the realization of these kinds of systems. Despite this effort, the proposed methods do not cover the need in its entirety, and need to be improved or even surpassed. It is in this sense that we undertake this work, we try to propose not only an extension of the UML language but also to try fill the gaps left by these different proposed methods, in order to allow a more efficient design and an

easier realization of self-adaptive systems, the result must offer an easy method to handle.

We propose a method based on models that will take into account all the specifications necessary for the design of self-adaptive systems, it must cover self-repair, self-optimization, self-organization .... It allows the explicit description of the adaptation mechanism of the modeled system. All the aspects modeled must be clearly explicit for a better reading and a better management of the models. The concept of adaptation that we will explain later, is based on the principle MAPE-K proposed by IBM (Figure 1).

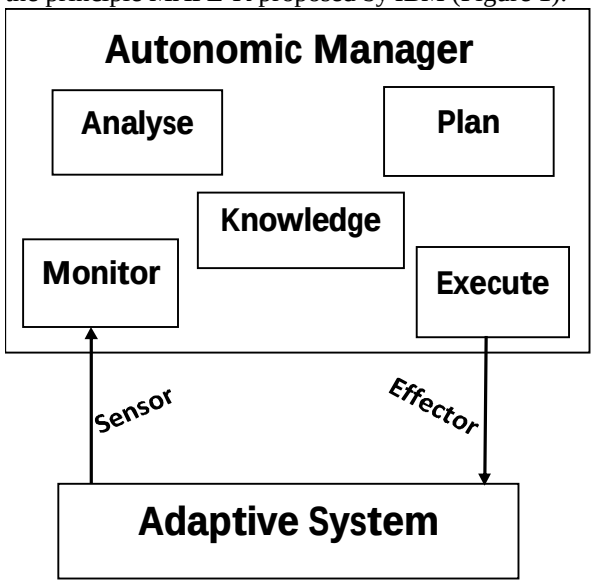

Figure 1: The MAPE-K model

The MAPE-K function in a very structured way, the sections are interrelated but each of them ensure a well-defined role. They can be presented as follow:

Monitor: it permanently supervises what is happening in the system environment. It collects information and detects any problem that needs to be analyzed.
Analyzer: it deals with the anomalies detected by the monitor by examining them in order to identify their natures and their implications.
Plan: it plans the action to take to react to the alterations that occur in the environment.
Execution: it uses effectors to make changes to the behavior or structure of the managed system.
Knowledge: It stores the data initially recorded or acquired following the actions of the monitor.

3. TECHNIQUE USED

Like the work done in [8] to set up our approach we used some parts of the model-driven engineering. Since we propose a method that is similar to a modeling language, we also use the technique presented by [2] that shows the procedure to determine a modeling language. Two aspects must be determined to have a real language: the abstract syntax and the concrete syntax.

Abstract syntax: This part describes all the valid models so usable to model any system. It also describes how the models will be structured [2] and the different interactions that may exist between them.

Concrete syntax: It describes how the models will be presented graphically or textually in the modeling editor. It also presents the semantics of models. In other words, it specifies the meaning of each model in the design.

To do this, we'll define a UML profile, which will mirror our view of the control loop. A profile is a set of stereotypes, giving classes additional specifications.

4. CONCEPT

Our approach clearly appears as a modeling language. So according to the methodology presented by [2] for the implementation of a new modeling language, we try to gradually determine the abstract syntax and the concrete syntax.

As mentioned above, the adaptation mechanism of our approach is inspired by IBM's MAPE-K principle. This mechanism can be divided into three phases: Acquisition, Analysis, and Action.

Acquisition: it mainly consists in senses the environment and its different fluctuations. Following this permanent monitoring it retrieves the changing parameters in the environment of the system and informs the adaptive system by sending it the details so that they can be analyzed.

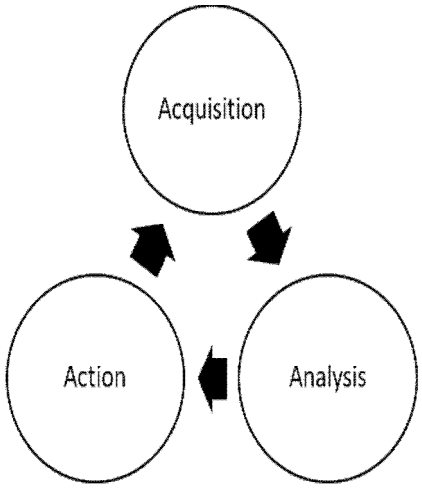

Figure 2: Our vision from MAPE-K

Analysis: it consists in intercepting the changes sent by the acquisition part. It reads the details, analyzes them to extract relevant information. Then an alert will be made to indicate that a change has occurred in the system environment and the parametric details related to it.

Action: based on the report received from the analyze part, it decides what actions to take to stabilize the state of the system so that it continues to operate. it can consist of a replacement of the system components or a simple modification of certain parameters.

5. THE ARCHITECTURE OF THE APPROACH

We will present here the global architecture of the approach. This approach offers a clear separation between the entities of the self-adaptive system and the instance that takes care of its self-adaptation. We thus distinguish a part that we can name models of the system which then represents the system itself and a part adaptation models which contains the necessary elements to ensure the adaptation of the system. The sensor is represented by the Listner model and the effector by the ConfigurationFile model as shown in the following figure.

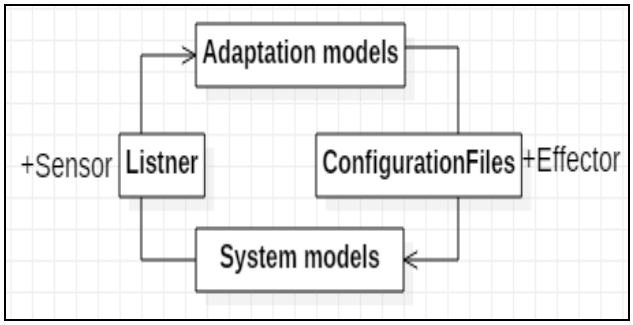

Figure 3: Global architecture of the approach

6. RESULTS

6.1. Definition of the profile

In this part we will try to present the UML profile defined for the approach. We will present the diagram showing all the elements of the profile in the image of the control loop. We summarize the different stereotypes presented in the UML profile, while explaining their meanings.

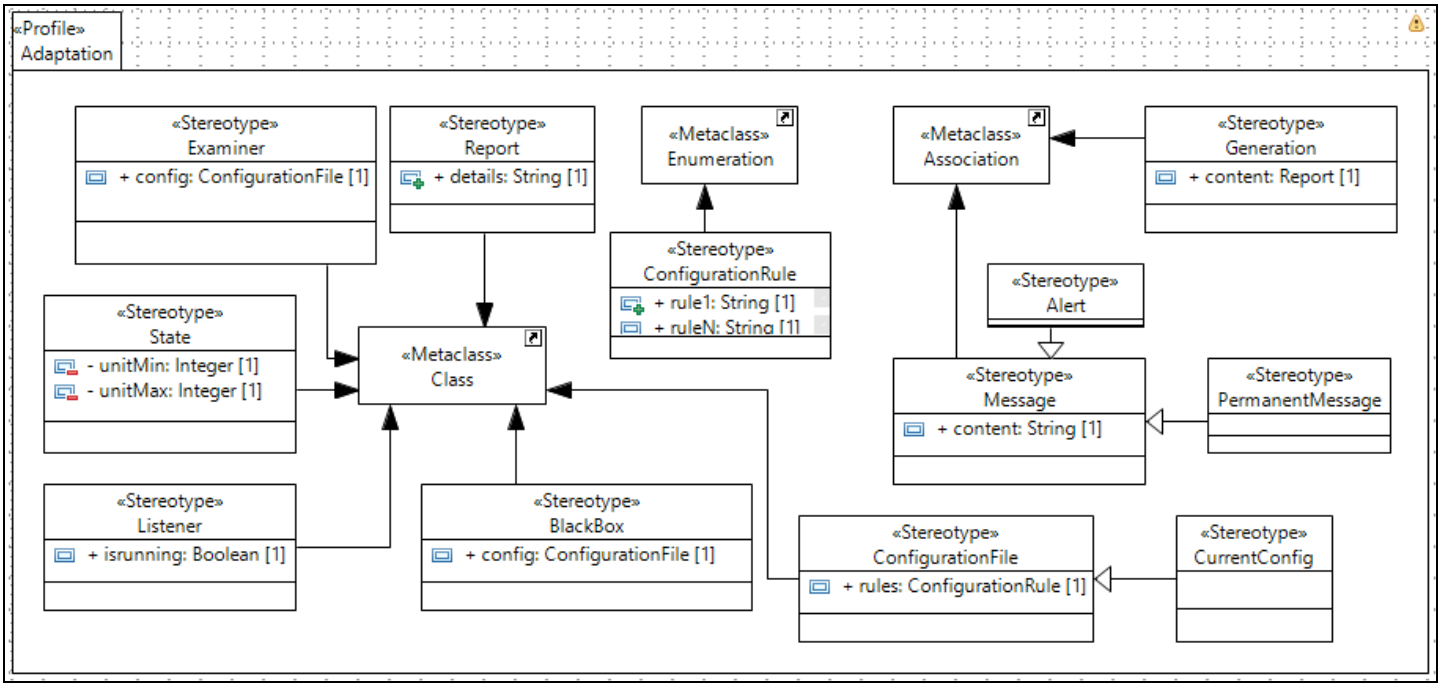

Figure 4: The profile diagram

The following table presents the different stereotypes, their types as well as their roles and meanings.

TABLE I: Stereotypes and descriptions

Stereotype	Comment
Classes	
Examiner	it is a model whose role is to analyze the details of the changes that have occurred in the system environment to decide on the action to be taken for the need of adaptation.
Report	it is a model whose role is to contain the details of the changes that have occurred in the system environment to convey it from the analysis part to the action part.
State	it is a model responsible for managing the state of the system environment and monitoring its change.
Listener	it is a model responsible for listening to variations in the state of the environment.

BlackBox	it is a model responsible for containing the various configuration files intended to control the system.
ConfigurationFile	it is a model responsible for containing a certain number of configurations which can be used to control the system.
CurrentConfig	it is the model responsible for containing the parameters of current control of the system.
Associations	
Message	It expresses a simple message occasionally sent from one model to another.
Generation	it indicates that the destination model is a model generated by the source model.
Alert	This is a type of message, prepared, in case the state of the system changes, by the dynamic model Listener. It is sent to the Examiner model which will study it and detect the symptom in order to prepare the Report for a next reconfiguration of the system. It must contain all the details about the invariant changes and their non-compliance with the current system configuration.
PermanentMessage	It describes a permanent interaction between two models. It must therefore be maintained throughout the life of the system in design.

Enumerations	
ConfigurationRule	it is a model which represents a very precise piloting rule in a ConfigurationFile.

6.2. Application example

We propose here (figure 5) the design of the adaptation mechanism of a video call application. We know that if the connection goes off during the call everything will stop (rule 1). In case of lack of connectivity the application must prioritize the audio and pause the video. As expressed by the rule 2 configuration rule. Each configuration rule consists of two parts: Condition and action.0 <speed <= 11,2kbps: pauseFV.start, this rule indicates that if the speed of the connection is not no higher than the 11.2kbps it is necessary to stop the video stream.

The Observation model, perceived here as an object, representing the listener constantly checks the speed of the connection through Connectivity. In case of lack of speed, we change the property to true and the Observation model will send an alert to Analysis which, after reading, will return a confirmation of receipt. ConfigurationRule rules that describe how the system responds to environmental changes to stabilize. **CurrentConfig** is made up of the drawn rules of type **ConfigurationRule** that can be derived from one or more Configuration Files.

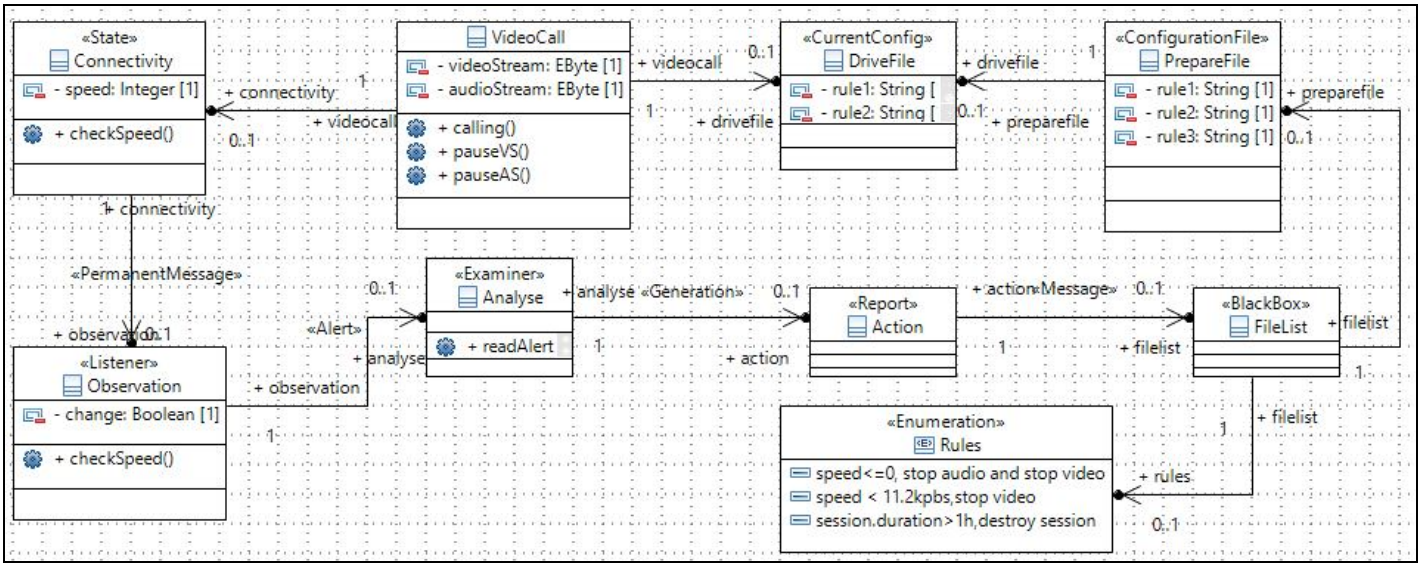

Figure 5: Video call adaptation using the approach

7. RELATED WORKS

The approach presented in this paper clearly contributes to solving the problem of self-adaptive systems realization, a very important problem that is attracting more and more attention from researchers. This approach is very special as it offers a level of abstraction high enough to make his apprehension, by the developers, very easy. The formal aspect

that this approach offers is one of its greatest advantages that, the developer can insert, in a very simple and expressive way, all the specification in terms of adaptation needs of the system to be designed, which greatly reduces the technical effort.

Nevertheless, this approach has a link, directly or indirectly, with other works more or less general than this, which we were more or less useful for the development of this approach.

Thus [3] proposes EUREMA, a formal approach based on models and mega-models [9] to reduce the task of designing and implementing self-adaptive systems. His approach proposes a control loop design framework based on the MAPE-K [10] principle proposed by IBM. It assumes that any aspect of adaptation, whether it is self-repair, self-organization, self-optimization ...., Is managed by a control loop. In this approach a control loop is a set of dynamic executable models running the system that will be encapsulated in another system called mega-model. Mega-models are run at system execution and system environment feature, and control models are generated based on adaptation requirements. In case of need of implementation EUREMA uses the stereotype principle to represent the models.

As for [11], it proposes a UML-based architecture for an expressive design of system self-adaptation needs. This approach is inspired by the MAPE-K principle and therefore determines three model packages, each package with a specific role. The Monitor package has specific models for perpetual monitoring of the state of the environment. The adaptation package contains the specific models for analysis and decision making to respond to environmental damage. When using the context adaptation package, it contains the templates needed to represent the system environment. This approach has set up a meta-model that will be inherited, by the stereotype principle in the specific implementation phase, and that will allow developers to check the validity of their models. For each proposed model, Adapt Cases allows the OCL specification accompaniment to define the constraints on the different models of the design.

We can also refer to one of our previous work[5] in which we presented a specification tool. This allows the system designer to be able to explicitly express the adaptation needs for the system he is designing. It offers a number of tools to represent all of the necessary components in an adaptive system.

8. CONCLUSION

We have shown in this article that our approach allows a rather elaborate design of self-adapting systems. It makes it possible to express clearly and with precision all the adaptation needs of the system during the design. We can express all the possible reactions of the system to respond to changes in its environment. We have also provided an example of an adaptation case to illustrate the approach. But since we are working in the field of model-based engineering, where models are not just for design, but are at the very heart of the system development process, there is room to improve this. approach. We plan to provide a model transformation grammar, based on our context, which will allow us to convert our base models to other formal or textual models. We can then implement this grammar with ad-hoc languages such as ATLAS Transformation Language Presentation (ALT)[12] or OMG's proposal, query view transformation (QVT)[13] to transform our models into expressions of certain programming languages.

REFERENCES

[1] J. M. Toms, Z. S. W. N, M. R. Thanka, and E. B. Edwin, “Innovative Agricultural Information System with User Friendly Digital Assistance for Farmers,” *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 10, no. 2, pp. 719–727, 2021.

[2] M. Brambilla, J. Cabot, and M. Wimmer, *Model-Driven Software Engineering in Practice*, vol. 1, no. 1. 2012.

[3] T. Vogel and H. Giese, “18 Model-Driven Engineering of Self-Adaptive Software with EUREMA,” vol. 8, no. 4, 2014.

[4] H. Tajalli, J. Garcia, G. Edwards, and N. Medvidovic, “PLASMA: A plan-based layered architecture for software model-driven adaptation,” *ASE’10 - Proc. IEEE/ACM Int. Conf. Autom. Softw. Eng.*, pp. 467–476, 2010.

[5] M. A. Bachir, I. Jellouli, E. G. Said, and A. Souad, “MVC-IMASAM: Model-View-Controller inspired modeling approach for system adaptation management,” in *2020 6th IEEE Congress on Information Science and Technology (CiSt)*, 2020, pp. 127–132.

[6] S. Walderhaug, E. Stav, and M. Mikalsen, “Experiences from Model-Driven development of homecare services: UML profiles and domain models,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5421, no. September, pp. 199–212, 2009.

[7] R. S. Bashir, S. P. Lee, S. U. R. Khan, V. Chang, and S. Farid, “UML models consistency management: Guidelines for software quality manager,” *Int. J. Inf. Manage.*, vol. 36, no. 6, pp. 883–899, 2016.

[8] M. Luckey and G. Engels, “High-quality specification of self-adaptive software systems,” *ICSE Work. Softw. Eng. Adapt. Self-Managing Syst.*, pp. 143–152, 2013.

[9] T. Vogel, A. Seibel, and H. Giese, “Toward Megamodels at Runtime.”

[10] E. Rutten, N. Marchand, D. Simon, E. Rutten, N. Marchand, and D. Simon, “Computing Feedback Control as MAPE-K loop in Autonomic Computing,” 2015.

[11] M. Luckey, B. Nagel, C. Gerth, G. Engels, and W. Straße, “Adapt Cases : Extending Use Cases for Adaptive Systems,” pp. 30–39, 2011.

[12] F. Jouault, “The Atlas Transformation Language (

ATL) ATL project Transforming models with ATL Operational Context of ATL ATL Transformation Example : Class to Relational."

[13] A. Kraas, "Realizing model simplifications with QVT operational mappings," *CEUR Workshop Proc.*, vol. 1285, pp. 53–62, 2014.