Lydia Robert-Edia *et al*, International Journal of Computer Science and Mobile Applications,
Vol.10 Issue. 5, May- 2022, pg. 7-15 **ISSN: 2321-8363**
Impact Factor: 6.308
(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

SELF-ADAPTIVE MOBILE APPLICATIONS AND THEIR ARCHITECTURAL STABILITY

Lydia Robert-Edia[1]; Blessed U. Yahaya[2]
[1]Anredia, lidichamp@gmail.com
[2]Flutterwave, blessedyahaya5@gmail.com

DOI: 10.5281/zenodo.6599079

Abstract
Nowadays mobile applications are becoming more and more universal. Due to increasing hardware enhancement on regular basis, multiple versions of operating systems and device specifications make self-adaption an essential attribute for mobile application's architecture. Therefore, to cater for the dynamic environment, systems must dynamically adapt their architecture to fulfill functional and non-functional requirements. Stability of an architecture basically defines that how well an architecture will accommodate the changing requirements and adapt to the dynamic environment. Thus, behavioral stability of a self-adaptive system is a major concern while developing self-adaptive systems. To address this issue, we work out a methodical approach which can be used for the analysis and modeling of systems with enhance architectural stability; furthermore, this method emphasizes on the behavioral characteristics of a system during execution. For the proof of concept of our method, we applied this approach on the architecture of a mobile application. Our proposed model is focused to gain the stockholders' desire for architecture stability and other related attributes

Keywords*: Mobile applications; Architecture; Stability; Self-Adaption*

1. Introduction
For any system its architecture style/pattern is the essential part that affects its functionality throughout its operational life. Architecture of software needs to be designed in a manner that it conforms to maximum quality attributes following a certain Standard. The stability of an architecture defines that how well an architecture will accommodate the changing requirements and adapt to the dynamic environment. In this context, thus stability of a self-adaptive system is a major concern. By behavioral stability of a system we mean that the functional and non-functional features of a system must not impair in any way. In the early phases of software development, the choice of quality attributes (QAs) is of primitive importance for the software project. The stability of system should be considered as a quality attribute at the various phases of software life cycle. At the architecture level the stability attribute ensures that the changing requirements or the dynamic environment does not affect architecture of a system and it remains performing its intended behavior. Mobile applications are becoming more convenient source for number of services these days. On increasing trend of smart phones in society, the demand for mobile applications has raised to a considerable level. Other than that hardware enhancement on regular basis, multiple versions of operating systems and device specifications make self-adaption an essential attribute for mobile application's architecture. Self-adaptive architecture leads an application to cope with dynamic environment, user's changing requirements and performs its intended behavior without giving an

impression to application failure to its user. Self-adaptation in mobile applications has been investigated in the software engineering community. Frame-work [24] for self-adaptive applications is a detailed framework but it lacks the quality attribute of stability. The issue hence addressed in this paper is to address how to tackle architectural stability. As a behavioral feature and quality attributes that need to be tackled during the dynamic operation of mobile applications, so the system adapts according to the environment and varying requirements and guarantees that its intended behavior (functional and nonfunctional requirements) is intact. We examine a systematic method for analyzing and modeling stability of architecture which focuses on the dynamic behavioral characteristics. This analysis model encompasses stability as the primary feature to conform to the needs of the stakeholders.
In section 2, we describe the background that covers the definitions of some main concepts. Section 3 shed light of some salient of the related work. Section 4 covers the proposed solution and Section 5 describes the outcomes and benefits of the proposed solution. Section 6 and Section 7 encompasses the case study and stability analysis of mobile application. Section 8 concludes with some directions for the future work.

2. BACKGROUND

This segment covers some fundamental ideas utilized in this paper. According to ISO/IEC/IEEE Standards, Programming Architecture is "crucial association of a framework exemplified in its parts, their connections to each other, and to nature, and the standards controlling its plan and advancement" [1]. Quality property is a trademark or highlight of main elements of software; this concept also includes general term related to some quality variables, quality of sub-elements, or metric qualities [2].

The self-adaptive nature of a software enables it to monitor its own behaviour and control/change its behaviour it is unable to accomplish some routine behaviour, or when it can perform some function in a better way [3] [4]. Similarly, a self-adaptive framework has the capacity of modifying its behaviour powerfully, because of the changes in his operational/executional environment (e.g. workload) and to deal with the conflict/clashes to achieve its objectives (e.g. quality necessities) [5]. An Adaptive application is intended to keep running on a cell phone. Portable applications ordinarily help clients with the comparative offices to those got to on PCs [5].

Ever growing evolution of technology and changing needs of customers has made the concept of software self-adaptation very trendy and need of the hour. Creating self-adaptive software (SAS) that oversees varieties progressively, at the runtime in a quick and solid way is an extremely powerful answer to limit the cost of programming change. Ever changing user requirements or inconsistency in the number of available resources are two likely situations in which SAS can reconfigure and continuously optimize itself at runtime. An adaptive behavior can replace a normal behavior in an autonomous way.

In general, a SAS implements a reference architecture, consisting of two major subsystems [17]: the adaptation manager (AM), and the adaptable software (AS), as shown in Figure 1. The AM plays its role as an exogenous controller for the observation of variations in the operating states of SAS and selects some suitable adaptive behaviour.

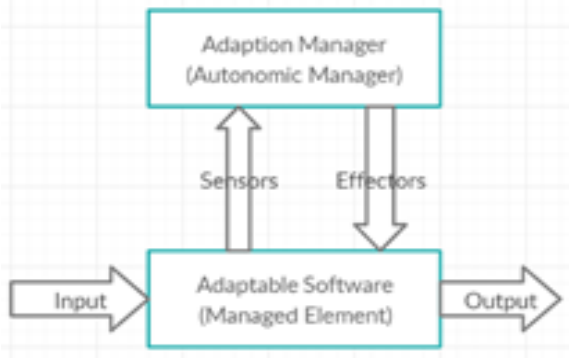

Figure 1: Self-Adaptive Software (SAS): Reference Architecture from [17]

3. RELATED WORK

This section briefly mentions the related technologies, work and frameworks while emphasizing on their main attributes. A lot of work has been done on dynamic and architecture-based adaptation frameworks, which includes the efforts to describe the requirements of self-healing systems [6, 7, 8].

The RINGA framework in [9] utilizes the idea of runtime verification for self-adaptive software by finite state machines. This framework comprises of two components, which are design and runtime. For designing the software SA-FSM (self-adaptive finite state machine) was proposed and its verification was done with an abstract algorithm A-FSM (abstracted finite state machine). The runtime verification was done with A-FSM with MAPE (monitoring, analysing, planning, and executing) loop implemented. [9].

The DiVA project in [10] studies the adaption at design and dynamic stages. DiVA emphases on adaptation at the architectural level and is grounded on a four-dimensional modelling method to self-adaptive software (SAS) [11]. FIOT [12] is a framework which can cope up with the evolution of IOT devices. This approach is based on machine learning algorithms and on Multi-Agent Systems (MAS).

SAFCA (Self-Adaptive Framework for Concurrency Architectures) is an architecture level and self-adaptive framework to increase the reliability and performance of software. In this approach the software can dynamically adapt different architectural style most suited to scenarios and observations at runtime [13].

AspectOpenCOM [14] is a reflection-based framework. This technology focuses on adaptation of already deployed aspects.

With the extensive use of mobile phones, various research works address the challenge of enhancing the quality of mobile applications by considering adaption in the operational context. In [15] the author proposed a system that manages itself when high level objectives are given to it by the administrators. In [16], researchers have used software architecture to enhance the quality of self-adaptive software. The reliability of the self-adaptive software can be increased through predicting outcome of every strategy at planning phase.

4. PROPOSED SOLUTION

Mobile Applications with the context awareness are able to check the operational environment; but it is in fact the adaptations that provide the primary mechanism for it. The adaptive behaviour or self-adaption of a system means its capability to change its behaviour dynamically using compositional adaptation defined within the framework.

Our paper focuses on self-adaption—for clarity we provide a stability analysis model for mobile applications that will be weighed out by an architectural stability model. The mobile application model identifies the

attribute precedence and the resources needed both these features are the major elements of adaptation (for removing or changing) for dynamic adaptation. In the software product line (SPL) domain, this adaptation to a changing context refers to member derivation. The feature priority level will be distinguished to accommodate the external feature adaptability.

Our approach uses a model for the mobile applications which are self-adaptive and context aware. This model is used for explaining SPL on the bases of feature selection based on multiple views. In general, models are used to expresses the domain related elements and their relationships, also the constraints which govern their structure as well as their behaviour. In our model along with the feature view, we also propose to include the runtime adaptability view. The specification of context requirements can also be used to enhance the selection of features at runtime effectively.

After the selection of the quality attributes, of interest, the next milestone is to develop a measurement plan which monitors the levels of the specific quality attributes, given the constraints of the specific phase. In addition to the mobile adaptability model for its stability, a stability architectural model will be applied to it as an architectural property. Application probabilistic relational model will be used to weigh out the stability variables that will later be quantitatively managed by Bayesian Network systems. The stability can be measured and evaluated by gauging the impact of using an adaptation strategy on the concerned quality attributes. This will help attaining the required run-time goals, and after that by assessing the architecture stability by using some probabilistic model. This includes two things: (i) dynamic quality of service (QoS) modelling at the architecture-level, and (ii) evaluating the effect and performance of the adaptation procedures. Moreover, it includes an approach for the verification based on the historical data for weighing the architecture stability. Our main focus is upon defining a framework that can handle runtime variability; this can lead to adaptation and application of a stability model for such self-adaptive applications.

Stability is one of the major adaptation properties that an adaptive system must possess. An unstable adaptation will result in indefinite repetition of the guiding action which might degrade the system instead of improving it. Stability depends on the quality attributes such as *performance*, *dependability*, *security* etc.

According to Figure 2, stability can also be verified by measuring the performance, dependability and security attributes at the runtime.

As far as the stability is concerned, the dynamic resource allocation system addresses performance and stability based on the dependability. Similarly, S.Parekh proposed the controller which addresses stability as one of the major adaptation property, by achieving service level objectives with the help of system's tuning parameters [18]. A basic control strategy (to influence the behaviour of the managed framework) was applied by them to execute a transfer function. Baresi and Guinea [19] likewise managed the stability by proposing a system which capable to recover from any undesired situation caused by any unwanted event in the system. The proposed system is created on service-oriented architecture (SOA). This would guarantee the availability of service (availability) and precision of service (reliability).

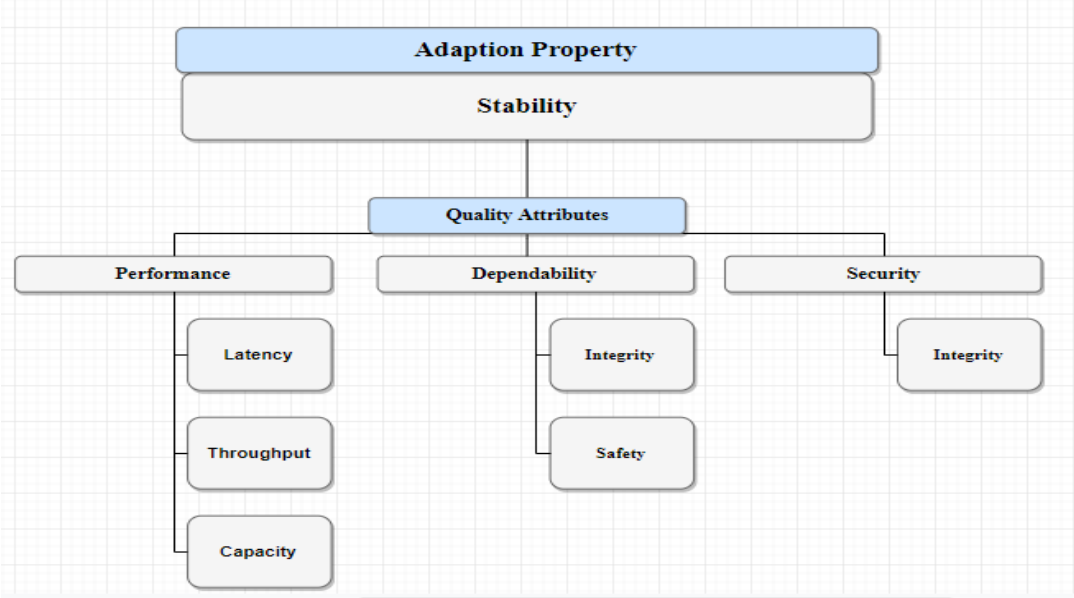

Figure 2: Mapping of the Adaption Property with the Quality Attributes

5. OUTCOME AND BENEFITS

To effectively engineer and to use the self-adaptive software systems; in the context of a modelled self-adaptive mobile application framework, we have taken an analysis model for handling architectural stability like a behavioural aspect for operations at runtime. This also ensures the seamless adaptation and providing constant desired services. This model analyses the stability depending on the architectural viewpoints. Stability viewpoints address the interest of the stakeholders related to the system's behaviour and the dimensions of stability which shows the type of architecture. In order to declare the desired behaviour which must be kept stable, the stability attributes are well-defined. The mathematical model provides the analysis based on quantitative terms for modelling the dynamic effects, correlation amongst stability attributes and analysing linked trade-offs. This approach is customized to simplify and streamline the process of adapting current or legacy software towards the runtime adaptively. The proposed approach is applied on self-adaptive mobile applications. The probabilistic model quantitatively captures the bearings and relations among the stability attributes.

The proposed approach will be used to conduct the runtime analysis, regarding the stability attributes. Such reasoning will improve the quality of adaptation to achieve the desired behaviour and supporting all the operations. Quality of services objectives are meant to be fulfilled with the help of the stable self-adaptive architecture. Adaption is then performed by keeping these objectives in view and by eradicating the unnecessary ones. Stability is an essential measure of SAS to guarantee efficiency and long-time use.

6. CASE STUDY: SELF-ADAPTIVE MOBILE APPLICATION ARCHITECTURE

We will use the approach presented by Maria Salama, Rami Bahsoon for Runtime Stability [20]. The planned approach is applied to test the stability during runtime; considering its adaptive behaviour under consideration too. Currently adaptive Mobile applications can be categorized on the basis of the following approaches

1. The system running the application handles its adaption.
2. The adaption is handled by the application itself.
3. The application and the system running it both handle the adaptation.

The focus in our paper is on the adaptation architecture managed by the application itself. The mobile applications need to operate under continuously changing conditions. These varying conditions could either be the available resources of the system the mobile application is operating on; or the bandwidth limitation of the wireless link. Many other such factors affecting the quality of the application must be adapted accordingly. For instance, in case the mobile system is running low on power the mobile application must adapt. The purpose of mobile application adaption is to provide the multi-tenant users with satisfying customer experience.

7. ANALYSIS OF STABILITY FOR SELF ADAPTIVE MOBILE APPLICATION ARCHITECTURES

In this analysis stability of architecture is [21] applied on case study from [22] [23]. Following steps are involved:

1. In accordance to the stability analysis model the two prominent categories of stability dimensions (the adaptation goal and adaptation property) stand out.
2. Next step is to pinpoint the stability stakeholders. The key stakeholders considered are the users, businesses and operational environments.
3. Define the main concerns for stability. The concerns of stability are different for every stakeholder. The user's concern is performance, businesses need security and the environments are to be made dependable.
4. After we are done identifying the classes for the stability dimensions, we need to derive viewpoints. Few viewpoints for stability (e.g. user, environmental, business and quality of adaptation) need to be identified and catered for. These represent the adaptation goals and its property dimension. Viewpoint *the quality of service* manages the requirements of end users**,** related to quality. Whereas aspects related to energy consumption are addressed by the environmental viewpoint.
5. Based on viewpoints, we define related attributes. Here, we have considered the performance and throughput. The green ability attribute, i.e., the energy consumed is used for environmental aspect. Operational cost based on the computational resources is defined for the economic constraints as shown in Figure 3. Settling time is referred to as the time required by the adaptation application to accomplish the adaptation goal properties [13].

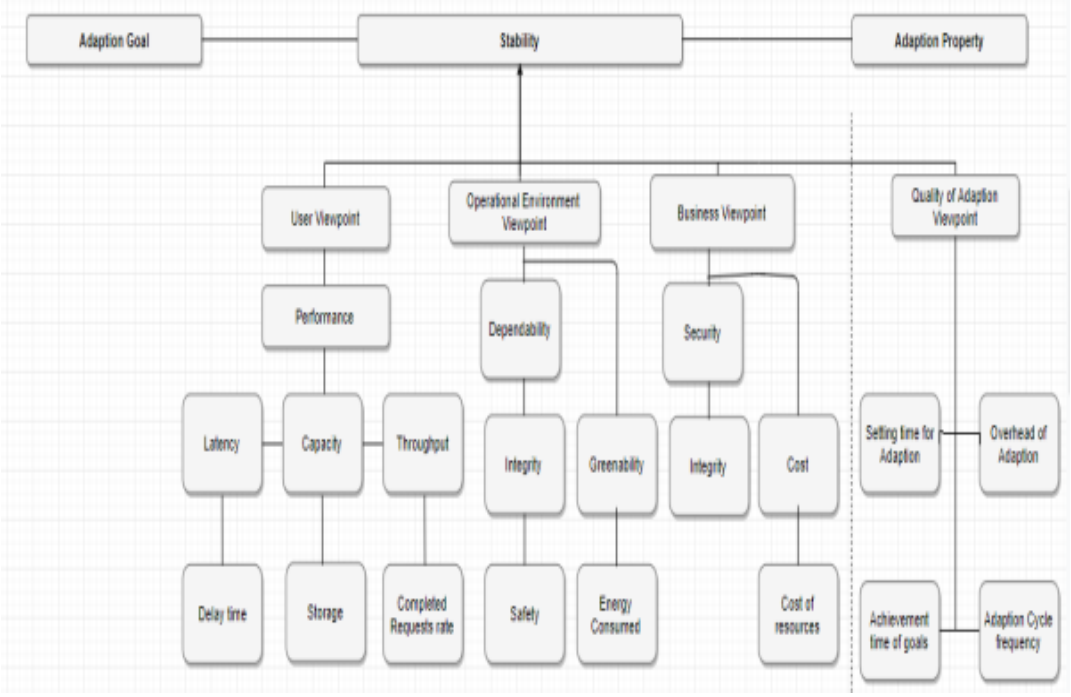

Figure 3: Stability Analysis of Case Study

8. CONCLUSION AND FUTURE WORK

This paper presents work based of a self-adaptive model for the mobile applications that was then outweighed by an architectural stability model for eliminating and altering features while adapting the mobile application dynamically. Feature priorities and the resources needed are identified by the mobile application model. For mobile applications which are context aware and self-adaptive, a model was used for describing feature selection-based SPL. After that a measurement plan was developed to monitor the levels of specific quality attributes. Then a stability architectural model was applied to it as architectural property.

Our future work will focus on building mobile applications that focus on ensuring seamless adaptation and providing constant desired services, which will cover mobile apps in a scenario when a mobile is running on low power. System discovers the status and apps adapt its behaviour accordingly while maintaining a permissible performance level and ensuring architectural stability meeting the required quality of services (QOS). Other than mobile applications, architectural stability can be applied to software defined radio while self-adapting system's behaviour according to the medium available while maintaining the quality of service.

REFERENCES

[1]. International Organization for Standardization and International Electro technical Commission (ISO/IEC), ISO/IEC/IEEE 24765:2010(E) Systems and software engineering Vocabulary, Report (2010).

[2]. Software Engineering Standards Committee of the IEEE Computer Society, IEEE standard for a software quality metrics methodology, (1998), Report IEEE Std 1061-1998, The Institute of Electrical and Electronics Engineers, Inc.

[3]. R. Laddaga, Self-adaptive software, Technical Report 98-12, DARPA BA (1997).
[4]. B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, J. Magee, J. Andersson, B. Becker, N. Bencomo, Y. Brun, B. Cukic, G. MarzoSerugendo, S. Dustdar, A. Finkelstein, C. Gacek, K. Geihs, V. Grassi, G. Karsai, H. M. Kienle, J. Kramer, M. Litoiu, S. Malek, R. Mirandola, H. M¨uller, S. Park, M. Shaw, M. Tichy, M. Tivoli, D. Weyns, J. Whittle, (2009), SOFTWARE ENGINEERING FOR SELF-ADAPTIVE SYSTEMS: A RESEARCH ROADMAP, SpringerVerlag, pp. 1–26.
[5]. P. Oreizy, M. M. Gorlick, R. N. Taylor, D. Heimbigner, G. Johnson, N. Medvidovic, A. Quilici, D. S. Rosenblum, A. L. Wolf, (1999) ,An architecture based approach to self-adaptive software, IEEE Intelligent Systems 14 (3),54–62.
[6]. C. Meng, (2001) "On evaluating self-adaptive software", Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 65–74.
[7]. Jeremy S. Bradbury, James R. Cordy, JuergenDingel, and Michel Wermelinger, (2004). A survey of self-management in dynamic software architecture specifications. In WOSS '04: Proc. of the 1st ACM SIGSOFT Workshop on Self-managed Systems, ACM, New York, pages 28– 33.
[8]. DebanjanGhosh, Raj Sharman, H. RaghavRao, and Shambhu Upadhyaya., (2007). Self-healing systems - survey and synthesis. Decis. Support Syst., 42(4):2164–2185.
[9]. Peyman Oreizy, Michael M. Gorlick, Richard N. Taylor, Dennis Heimbigner, Gregory Johnson, Nenadzedvidovic, Alex Quilici, David S. Rosenblum, and Alexander L. Wolf.,(1999). An architecture-based approach to self-adaptative software. IEEE Intelligent Systems, 14(3):54– 6.
[10]. Euijong Lee, Young-Gab Kim, Young-Duk Seo, Kwangsoo Seol, Doo-Kwon Baik (2018) RINGA: Design and verification of finite state machine for self-adaptive software at runtime, Information and Software Technology,Volume 93, Pages 200-222.
[11]. Brice Morin, Olivier Barais, Jean-Marc Jezequel, Franck Fleurey, and Arnor Solberg.,(2009). Models@ run.time to support dynamic adaptation. Computer, 42(10):44–51, 2009. 31, 41, 68, 92.
[12]. Franck Fleurey & Arnor Solberg.,(2009). A domain specific modeling language supporting specification, simulation and execution of dynamic adaptive systems. In Proceedings of the 12th International Conference on Model Driven Engineering Languages and Systems, pages 606–621, 2009.
[13]. Nathalia Moraes do Nascimento & Carlos José Pereirade Lucena, (2017) FIoT: An agent-based framework for self-adaptive and self-organizing applications based on the Internet of Things Information Sciences, Volume 378, Pages 161-176.
[14]. Chung- Horng Lung & Xu Zhang,(2016). Improving software performance and reliability in a distributed and concurrent environment with an architecture-based self-adaptive framework, Journal of Systems and Software, Volume 121, Pages 311-328.
[15]. Paul Grace, Bert Lagaisse, Eddy Truyen, and WouterJoosen,(2008). A reflective framework for finegrained adaptation of aspect-oriented compositions. In Proceedings of the 7th international conference on Software composition, pages 215–230.
[16]. Jeffrey O. Kephart & David M. Chess., (2003) The Vision of Autonomic Computing. IEEE Computer, 36(1).
[17]. João M. Franco, Francisco Correia, Raul Barbosa, Mário Zenha, (2016). Improving self-adaptation planning through software architecture-based stochastic modeling, Journal of Systems and software, Volume 115, Pages 42-60.
[18]. Mehdi Amoui Kalareh, Evolving Software Systems for Self-Adaptation, http://hdl.handle.net/10012/6643.
[19]. S. Parekh, N. Gandhi, J. Hellerstein, D. Tilbury, T. Jayram, and J. Bigus., (2002).Using control theory to achieve service level objectives in performance management. Real-Time Systems, 23:127–141.

[20]. Luciano Baresi and Sam Guinea. Self-Supervising BPEL Processes. , (2011). IEEE Trans. on Software Engineering, 37:247–263.

[21]. Maria Salama & Rami Bahsoon. ,(2017). Analyzing and Modeling runtime Architecture stability for Self Adaptive Software, 27 July. 15,16,17,18.

[22]. M. Salama, R. Bahsoon, A taxonomy for architectural stability, in:1345 Proceedings of 31st ACM/SIGAPP Symposium on Applied Computing (SAC), Software Architecture: Theory, Technology, and Applications.

[23]. T. Chen, F. Faniyi, R. Bahsoon, P. R. Lewis, X. Yao, L. Minku, L. Esterle, The handbook of engineering self-aware and self-expressive systems, Technical report, School of Computer Science, University of Birmingham.

[24]. M. Salama, R. Bahsoon, Quality-driven architectural patterns for self-aware cloud-based software, in: Proceedings of IEEE 8th International Conference on Cloud Computing (IEEE CLOUD), 2015, pp. 844–851.

Cite this paper as:
"Lydia Robert-Edia, & Blessed U. Yahaya. (2022). SELF-ADAPTIVE MOBILE APPLICATIONS AND THEIR ARCHITECTURAL STABILITY. *International Journal of Computer Science and Mobile Applications, 10(5)*, 7–15. https://doi.org/10.5281/zenodo.6599079"