Journal of Knowledge Information Technology and Systems(JKITS), Vol. 14, No. 3, pp. 267~274, June 2019
https://doi.org/10.34163/jkits.2019.14.3.006

Journal of Knowledge Information Technology and Systems

ISSN 1975-7700

http://www.kkits.or.kr

A Study on Workflow Processes for Self-Adaptive Software System Approaches

Jeong-Sig Kim[1], Jinhong Kim[*2]

[1]*Department of Computer and Mobile Convergence, Gyeonggi University of Science and Technology*
[2]*Department of Information Communication Engineering, Seoil University*

ABSTRACT

The Self-Adaptive Software Systems (SASS) is composed to a complicated process. It depends on some factors that can change during the system operation lifetime. The adaptive software system composes of the adaptation engine and the adaptable software. This engine implements adaptation logic by using a policy engine and by providing of middleware. The advantage of this approach consists on the fact that adaptation engine can be reused and each adaptation processes can be applied to various applications. For this reason, it is necessary to define the processes for providing as Self-Adaptive System (SASS) the capability of generating for run-time the process that manage to their adaptation. Accordingly, in our research paper, we present a framework for the automatic generation of processes for SASS based on the use of workflows, model-oriented and artificial intelligent planning techniques. Our approach can be relevant to different application domains, enhance the scalability related with the generation of adaptation plans, and enables the usage of each planning techniques. To evaluating our approach, we have proposed a prototype for generating for run-time the workflows.

KEYWORDS: Self-adaptive software system, System operation lifetime, Model-oriented, Artificial intelligent planning, Scalability

ARTICLE INFO: Received 21 May 2019, Revised 5 June 2019, Accepted 7 June 2019.

[*]Corresponding author is with the Department of Information Communication Engineering, Seoil University, 28, Yongmasan-ro 90-gil, Jungnang-gu, Seoul, 023192, Korea
E-mail address: jinhkm@seoil.ac.kr

1. Introduction

Self-adaptive Software was offered as a "Self- Adaptive Software evaluates its own behavior and changes behavior when the evaluation indicates that it is not accomplishing what the software is intended to do, or when better functionality or performance is possible." By their software systems, there are various ways of carrying out their purpose and their effective changes will be done for runtime [1]. The software should have some functionality in order to estimate its performance and behavior. It can reconfigure their operations to improve its performance [2][3]. We proposed how the adaptation is necessary to be applied. They introduced different approaches: *(1) Static/Dynamic Decision-Making.* This approach control how the decision process can be constructed and modified. Their decision process are coded of itself and it requires to recompilation for each systems components. By the dynamic decision making, policies and rules are defined extremely, so their modification performed for run-time to change behavior with functional requirements or non-functional requirements.

(2) External/Internal Adaptation. The adaptation approach is divided in two categories with respect to the adaptation mechanism and the application logic. Internal Approach is based on the programming language features, such as conditional expressions, parameterizations and exceptions [4]. This approach can be useful for local adaptation. And also it needs global information about this system. It can be realized by existing programming features or by having a new adaptation programming. External Approach deals with the this adaptation engine including adaptation processes. The adaptive software system composes of the adaptation engine and the adaptable software. This engine implements adaptation logic by using a policy engine and by providing of middleware. The advantage of this approach consists on the fact that adaptation engine can be reused and each adaptation processes can be applied to various applications. *(2) Making/Achieving Adaptation.* Firstly, this strategy consists of self-adaptivity into the system with developing phase [5-7]. Secondary, one is to achieve self-adaptivity with adaptive learning. In a point of view on software engineering is to adaptivity into the software system. Artificial intelligence and adaptive learning provide on the achieve adaptive behavior. Adaptive system design use a goal-oriented approach for their system requirement specification. Thus, it is expected that SASS should be able to provide adaptation plans for run-time, in order to deal with the variability and uncertainty involved in the adaptation effectively. Otherwise, self-adaptation process be consistent with the actual system by controling. Otherwise, it is hard to anticipate to all the possible contexts of adaptation for some types of systems. In our research paper, we propose to framework

that are able to dynamic generation of processes for runtime. This framework have a process and collection with program languages, mechanisms and techniques, and they could be supporting by computational infrastructure. Section 2 describes related works, Section 3 describes framework for process generation, and also Section 4 and 5 also describes WGP and WGP prototype implementation. Finally, Section 6 describes Conclusions.

2. Related Work

Dynamic software architectures and architecture-based adaptation framework developed to including an effort to characterize the style requirements [8-10]. They have a representative set of approaches, categorizing each by its primary focus. They focus on formalism and modeling, or mechanisms of adaptation of itself [11]. *(1) Distributed Adaptation.* Self-organizing systems in proposes an approach where self-managing units coordinate toward a common model, an architectural structure defined using the architectural formalism of Darwin. Each self-organizing component is responsible for managing its own adaptation with respect to the overall system [12][13]. To do this, each component maintains a copy of the architecture model of the entire system. *(2) Dynamic Architectures.* The K-Component model addresses the integrity and safety of dynamic software evolution, modeled as graph transformations of meta-models on architecture. It uses reflective programs called adaptation contracts to build adaptive applications, coordinated via a configuration manager [14]. These approaches assume that system implementations are generated from the architecture descriptions [15][16]. Our research approach, processes are represented through workflows that are dynamically generated.

3. Framework for Process Generation

In our research approach, processes are represented through workflows that are dynamically generated. Our framework provides the generation and execution of workflows in three phases: *strategic, tactical and operational*. By the *strategic phase*, AI planning get used to generate abstract workflows. An abstract workflows have set of tasks and data interdependencies among them without identifying the actual resources. At the *tactical phase*, an abstract workflows is mapped into a concrete workflows which identifies the actual resources associated with the tasks. It is important to abstract workflows that can be mapped into different concrete workflows by using different combinations of resources. At the *operational phase*, the concrete workflows is executed. <Figure 1> show a simplified view of our approach for the dynamic generation of workflows.

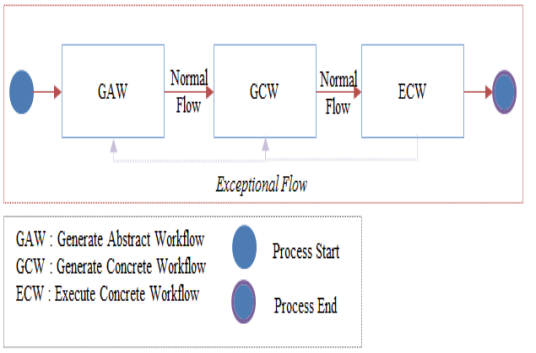

Figure 1. Workflows Generation

Our approach suggest to the context with three layers reference model for self-managed system by adopted. According to goals management layer, our strategic phase and abstract workflows are generated according to a particular objective. These workflows are used as a basis for generating the concrete workflows at the change management layer. Once a concrete workflows have been generated, it is executed at the component control layer, corresponding to the operational phase. In our approach, similar to the three layers reference model, if errors occur at a particular phase and that phase is not able to handle the error, these are propagated to the previous phase in which they should be dealt with. In case of a problem occurs during the execution of the concrete workflows, a new concrete workflows are generated at the tactical phase, without the need to generate a new abstract workflows. If it is not possible to generate a concrete workflows, the generation goes back to the strategic phase, where a new abstract workflows are generated. In the eventuality it is not possible to generate an abstract workflows, the generation finishes with an error.

4. WGP (Workflows Generation Process)

Our framework provide three different phases, each phases are associated with the generation of workflows, and the last phase is related to the workflows execution. In each phases, generation process is composed by each activities that some of these activities rely on the application domain. In addition, generation framework is used to the application domain. Accordingly, we have activities associated with the strategic and tactical phases in detail. Moreover, we focusing on the domain independent activities. *(1) Strategic Phase.* The main objective with this phase find the sequence of tasks that will compose to abstract workflows. AI planning techniques mean to their achieve. In order to generate a workflows, AI planner receives as input the goal, initial state set of available task templates. <Figure 2> shows an overview how workflows are generated by the strategy phase. By the AI planning, it is necessary to obtain the initial state and the goals associated with the workflows. In addition, they are represented by the *Obtain current state and Obtain workflows goals* activity. These activities are dependent of the application domain in which the generation framework is being used.

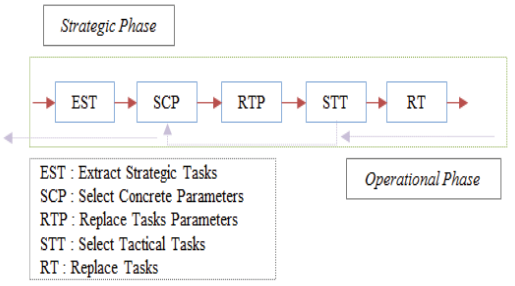

Figure 2. Activity Process Level

Translate into workflows activity receives the output of the planner and translates it into an abstract workflows in the format used for specifying workflows in the execution on platform. This translation involves the initialization of each action identified in the planning XML-based configuration language for plan. Their action are used for populating the associated workflows tasks. It is important to mention that the tasks that compose an abstract workflows are referred to as *strategic tasks*. And also, resources are associated with these tasks that referred to using logical names. After all, they are sufficient to identify the actual resources at the next phase.

5. WGP Prototype Implementation

The infrastructure for supporting the defined configuration process has been implemented based on the architecture shown in <Figure 3>. In our prototype, architectural models showed using Eclipse Framework models. In order to run our experiments, we have implemented a simplified execution our platform, in which components must be blocked. Blocked components are not considered when a new selected by configuration. And all architectural elements provide two different types of interfaces, application with configuration services interfaces.

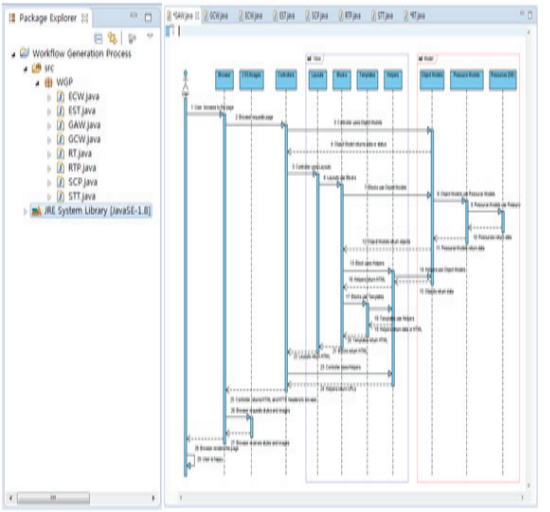

Figure 3. WGP Prototype by Eclipse Environment

Thus, The Modeling Workflows Engine (MWE) is a declarative configurable generator engine. Our research provide simple XML-based configuration language with all kinds of generator workflows. A generator workflows consists of a number, so-called workflows components. They are executed sequentially in a single Java Virtual Machine. A workflows component represents a part of a generator process. The size of the generated workflows depends on the number of components and connections in the selected

configuration. Every component of a configuration must be blocked before being connected.

6. Conclusions

Our research paper has proposed a framework for the automatic generation of processes for SAS systems, with AI planning and model transformation. Our framework could be applied to each different application domains by supporting the use of the most suitable generation techniques by application domain. Although, our proposed approach for the dynamic generation of process, self-adaptive software systems has produced quite promising implementation, we have identified a couple of limitations that could enhance overall effectiveness of the approach. For approach to our future research, we would like to simplify the reuse of the framework. This research could be achieved by using meta transformation languages for translating domain specific models into pre- and post-conditions. Moreover, it is to incorporate ideas from adaptive software product into our framework for dealing with the variability of processes.

References

[1] J. H. Kim, and S-C. Kim, *Toward hybrid model for architecture-oriented semantic schema of self-adaptive system*, GPC 2013, Vol. 7861, pp. 832-837, LNCS

[2] D. Perry, and A. Wolf, *Foundations for the study of software architecture.* ACM SIGSOFT Software Engineering Notes, Vol. 17, No. 4, pp. 40-52, 1992.

[3] E. Dashofy, A. Hoek, and R. Taylor, *Towards architecture-based self-healing systems.* In:Workshop on Self-Healing Systems (WOSS 2002), No. 1, pp. 18-19 2002.

[4] J. Kephart, and D. Chess, *The vision of autonomic computing.* IEEE Computer, Vol. 4, pp. 41-51, Jan. 2003.

[5] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J-M. Loingtier, and J. Irwin, *Aspect-oriented programming*. In: Akşit, M., Matsuoka, S. (eds.) ECOOP 1997. LNCS, Vol. 1241, pp. 220-242. Springer, Heidelberg, 1997.

[6] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, *Failure diagnosis using discrete-event models.* IEEE Trans. Cont. Syst. Tech. Vol. 4 No. 2, pp. 105-126, 1996.

[7] J. H. Kim, and E. S. Lee, *Semantic web recommender system based personalization service for user xquery pattern.* In: Deng, X., Ye, Y. (eds.) WINE 2005. LNCS, Vol. 3828, pp. 848-857. Springer, Heidelberg, 2005.

[8] J. H. Kim, and S. C. Kim, *Design of architectural smart vehicle middleware.* INFORMATION: An International Interdisciplinary Journal (III), Vol. 11, No. 4, ISSN pp. 1343-4500. 2013.

[9] N. Pessemier, L. Seinturier, T. Coupaye, L. Duchien, *A model for developing component-Based and aspect-Oriented systems.* In: Löwe, W., Südholt, M. (eds.)

SC 2006. LNCS, Vol. 4089, pp. 259-274. Springer, Heidelberg, 2006.

[10] M. U. Khan, R. Reichle, and K. Geihs, *Applying architectural constraints in the modelling of self-adaptive component-based applications.* In: ECOOP Workshop on Model Driven Software Adaptation (M-ADAPT), Berlin, Germany, Jul./Aug. 2007,

[11] G. Kiczales, and M. Mezini, Aspect-oriented programming and modular reasoning. In: 27th Int. Conference on Software Engineering (ICSE), St. Louis, MO, USA, pp. 49-58. ACM, May 2005.

[12] R. Alur, C. Courcoubetis, T. A. Henzinger, and P-H. Ho, *Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems.* In: Grossman, R.L., Ravn, A.P., Rischel, H., Nerode, A. (eds.) HS 1991 and HS 1992. LNCS, Vol. 736, pp. 209-229. Springer, Heidelberg, 1993.

[13] Y. Wang, and J. Mylopoulos. *Self-repair through reconfiguration: A requirements engineering approach.* In Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering, pp. 257-268. IEEE Computer Society, 2009.

[14] F Dalpiaz, R Ali, Y Asnar, V Bryl, and P Giorgini. *Applying tropos to socio-technical system design and runtime configuration.* In Proceeding of the Italian workshop Dagli OGGETTI agli AGENTI (WOA08), 2008.

[15] V. Bryl, P. Giorgini, and J. Mylopoulos. *Designing sociotechnical systems: from stakeholder goals to social networks*. Requir. Eng., Vol. 14, No. 1, pp. 47-70, 2009.

[16] S. Edelkamp, and J. Hoffmann, *Pddl2.2: The language for the classical part of the 4th international planning competition.* Technical Report 195, Jan. 2004.

자가 적응형 소프트웨어 시스템을 위한 워크플로우 프로세스에 관한 연구

김정식[1], 김진홍[2]

[1]*경기과학기술대학교 컴퓨터모바일융합학과 조교수*

[2]*서일대학교 정보통신공학과 조교수*

요 약

자가 적응형 소프트웨어 시스템은 시스템 운영 생명시간동안 변화되는 일부 요소들에 의존하는 복잡한 프로세스로 구성되어 있다. 적응형 소프트웨어 시스템은 적응형 엔진과 적응형 소프트웨어로 구성되어 있다. 이러한 엔진은 미들웨어에 의해 적응형 로직으로 구현된다. 이러한 접근의 장점은 적응형 엔진이 재상요될 수 있으며, 각각의 적응형 프로세스들은 다양한 애플리케이션에 적용가능 할 수 있는 장점이 있다. 이러한 이유로, 자가적응 소프트웨어 시스템은 자체적 적응을 수행하기 위해 프로세스가 동작하는 동안 자가 적응 시스템의 성능을 제공하기 위한 메커니즘 정의가 필요하다. 따라서, 본 논문에서는 자가 적응 소프트웨어 시스템 기반 워크플로우를 이용하기 위해 자동적 프로세스 세대들을 위한 프레임워크와 모델 지향, 그리고 인공 지능 계획 기술들을 제안하고자 한다. 우리의 제안방법은 서로 다른 응용 도메인에 적용할 수 있으며, 적응 계획 세대와 상호 협력할 수 있는 확장성을 증가시키고, 서로 다른 계획 기술에 대한 사용이 가능하도록 하는데 있다. 본 논문의 평가를 위해, 워크플로우의 런타임 동안 프로세스 세대의 프로토타입을 개발하고자 한다.

Acknowledgments

This work was supported by the Research Fund of Gyeonggi University of Science and Technology in 2019.

Jeong-Sig Kim he is Assistant Professor of Department of Computer and Mobile Convergence, at the Gyeonggi University of Science and Technology, Gyeonggi-do, and Korea. He respectively received his Ph.D. degrees in Electrical, Electrical and Computer Engineering from Sungkyunkwan University, Korea, in 2007. His research interests include smart vehicular network, smart platform, software engineering, wireless sensor networks, scalable reliable communication protocols, mobile computing, network security protocols, proxy caching systems.

E-mail address: arius70@gtec.ac.kr

Jin Hong Kim, he is Assistant Professor of Department of Information Communication Engineering at the Seoil University, Seoul, and Korea. He respectively received his Ph.D. degrees in Electrical, Electrical and Computer from Sungkyunkwan University, Korea, in 2006. Dr. Kim was Information System Research Center at the University of New South Wales in Australia. Moreover, He acted as an Engineer expert on behalf of Vodafone Company. He also served or currently serving as a reviewer and Technical Program Committee for many important Journals, Conferences, Symposiums, Workshop in Computer Communications Networks area. His research interests include smart vehicular network, smart platform, software engineering, wireless sensor networks, scalable reliable communication protocols, mobile computing, network security protocols, proxy caching systems. He is a life member of the KKITS.

E-mail address: jinhkm@seoil.ac.kr