A HOLISTIC SELF-ADAPTIVE SOFTWARE MODEL

Shatha Alfar and Said Ghoul

Faculty of Information Technology, Research Laboratory on Bio-inspired Software Engineering Philadelphia University, Amman, Jordan

ABSTRACT

The recent self-adaptive software systematic literature reviews stated clearly the following insufficiencies: (1) the need for a holistic self-adaptive software model to integrate its different aspects (2) The limitation of adaptations to context changes (3) The absence of a general and complete adaptations' picture allowing its understandability, maintainability, evaluation, reuse, and variability. (4) The need for an explicit and a detailed link with resources, and (5) a usual limitation to known events.

In order to metigate these insufficiencies, this paper is proposing a holistic model that integrates the operating, adaptations, and adaptations' manager aspects. The proposed model covers all possible adaptations: operating (dealing with software functions failures), lifecycle (handling adaptations required by some software lifecycle steps), and context (facing context changes events). The presented work introduces the concept of software adaptations process integrating the specifications of all the above kind of adaptations. In fact, this work shows an explicit trace to its pure bio-inspired origin.

An application of the proposed approach on a "car industry case study" demonstrated its feasibility in comparison with similar works that proved its meaningful added value and its promising research perspectives.

KEYWORDS

Self-adaptive software, bio-inspired approach, adaptation events, immune system.

ACM Computing Classification System categories and concepts
- *Software and its engineering ~ Software design engineering*

1. INTRODUCTION

Self-Adaptive Systems (SAS) have been introduced as a solution to deal with environmental changes in computer systems [1, 2, 3]. These systems generally adapt themselves to their related changing environment, using accurate configurations that are already available during the running time of the software. They could adapt themselves without any external response.

Artificial immune systems propose both innate (predefined) and adaptive (by learning) immunity. These kinds of systems, which are inspired by nature can easily handle any software fault identification and detection. Despite of the important scale of the research in nature-inspired self-adaptation software, the artificial immune models face the insufficiency resulted from missing a holistic self-adaptive software model integrating its different aspects.

DOI: 10.5121/ijsea.2021.12301

To overcome this insufficiency, this paper proposes a holistic model that integrates the operating, adaptations, and adaptations manager aspects. The proposed model covers all possible adaptations: operating (dealing with software functions failures), lifecycle (handling adaptations required by some software lifecycle steps), and context (facing context changes events). The presented work introduces the concept of software adaptations process integrating the specifications of all the above kind of adaptations. In fact, this works show an explicit trace to its pure bio-inspired origin.

An application of the proposed approach on a car industry case study demonstrated its feasibility in comparison with similar works that proved its meaningful added value and promising future in research.

The remainder of this paper is structured as follows: Section 2 presents similar works and outline evidences about motivations of this work. Section 3 introduces a running industrial example that is used throughout the paper to illustrate the new introduced concepts. Section 4 presents the proposed approach. Section 5 evaluates the added value compared with previous related approaches. Section 6 terminates with a conclusion.

2. RELATED WORKS

Self-Adaptive Systems (SAS) generally adapt themselves to their related changing environment[1, 2, 3]. Several Self-Adaptive works relied on conventional approaches dealing with the conventional whole lifecycle [2], requirements [4], design [1], modeling [1], development and evaluation [3, 5]. Others works relied on nature-inspired approaches [6, 7, 8] based on immunity models [7, 8]. These last works dealt with Artificial Immune Systems (AIS) with the capabilities of self-configuration, self-adaptation and self learning. AIS propose both innate (predefined) and adaptive (by learning) immunity. These kinds of systems can easily handle any software fault identification and detection [9, 10].

It is clear, from the above research works and in recent SAS systematic literature reviews [11, 12, 13], that the following challenges are valuable to deal with: (1) the need to a holistic self-adaptive software model integrating its different aspects (2) The limitation of adaptations to context changes (3) The absence of a general and complete adaptations picture allowing its understandability, maintainability, evaluation, and variability. Reasoning about these adaptations for various purposes will be facilitated. (4) The need to explicit and detailed link with resources the self-adaptive software is inspired from, and (5) a usual limitation to known events.

To overcome these insufficiencies, this paper proposes a holistic model that integrates the operating, adaptations, and adaptations manager aspects.

3. RUNNING EXAMPLE

Moro et al. presented a running example that was obtained from a real demonstrator from the automotive industry [14]. Based on this inspiration, a running case study example was developed and will be used throughout this paper to illustrate the introduced concepts. This paper particularly introduces an aspect-based software model integrating its operating, adaptations, and manager aspects throughout its lifecycle. This model focuses on software operating, lifecycle, and context adaptations programs and processes.

The System_Car is a Software Product line [15] following up a car throughout its lifecycle (Figure 1a). The System_Car adapts itself to each new reached state according to preplanned

lifecycle adaptation programs specific to each state. At each state, the System_Car might be adapted according to Random events (Figure 1.b) requiring its operating or context adaptations (changes or failures in its provided functions or changes in its environment). Thus, when reaching a state the Car should firstly operate under its corresponding lifecycle adapted configuration (New Configuration LAE) and then might adapt itself to a new configuration (New configuration OAE or CAE) at each Random event requiring such adaptation.

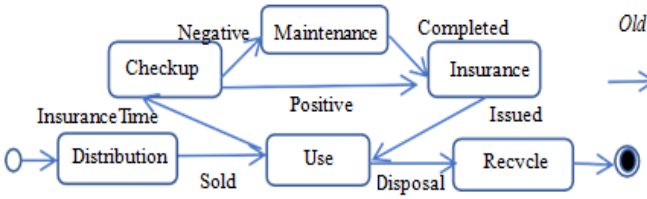

Figure 1a. A UML Car lifecycle.

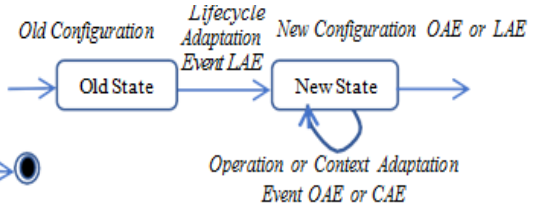

Figure 1b. Kind of adaptations

The System_Car is simplified to:

- Emergency calls compulsory sub-system that - in case of an accident- sets off an automated distress call that includes the car's satellite position and some relevant additional data. The car supports two call systems: one for Europe (eCall), and one for Russia (ERAGlonass). In Europe, GPS or Glonass can only be used with the English language.
- Operating, lifecycle, and context sensors that generate events leading to System_Car reconfiguration
- Constraints ensuring configuration integrity and coherence.

4. A HOLISTIC SELF-ADAPTIVE SOFTWARE MODEL

4.1. Adaptation Process

The idea of using concepts of natural Immune system [16] was inspired by the human body immunity system and how antibodies are produced against antigens (virus or bacteria). The human immune system deals with antigens by employing two systems: Innate Immune System (IIS) and Adaptive Immune System (AIS).

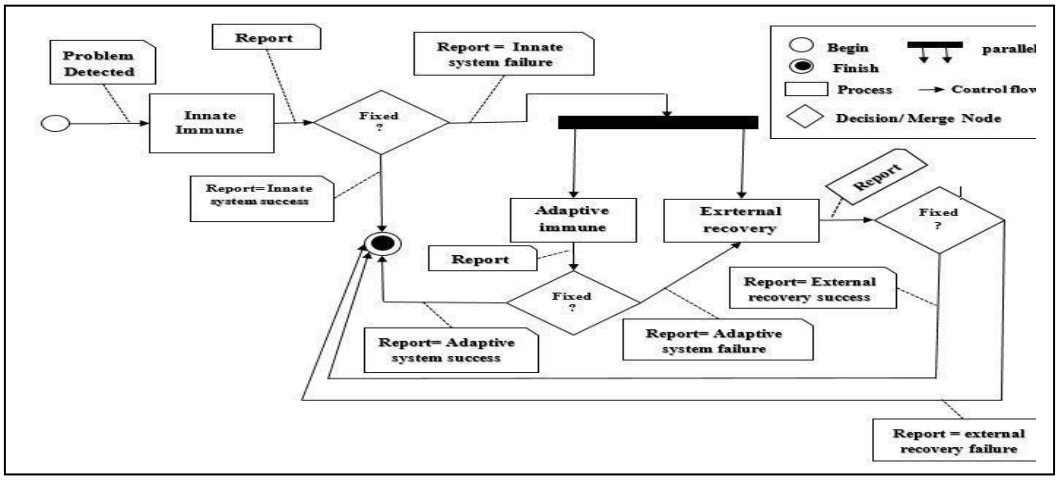

Figure 2a. Proposed general model of artificial immune system using UML

Figure 2a shows a bio-inspired artificial immune system model proposed for dealing with the self- adaptation process of software throughout its lifecycle. It is the kernel of the self-adaptations' manager.

The artificial IIS aims to detect a problem (operating, lifecycle, or context events) and then looks for a corresponding adaptation. If the problem is recognized, then it will be fixed. Otherwise, the problem will be moved out and the AIS step will be activated. The artificial AIS models the process for handling failed situations in the IIS. It consults an expert that suggests the most suitable solutions. The evaluator will then evaluate a given solution, from the set of candidate solutions, and its level of success. If this solution is suitable, then the problem will be fixed and the description of the problem with its successful solution will be saved for the future needs. Otherwise, the problem will be considered as a failure situation. In parallel with the advising task, the AIS might also look for an external (i.e. human) action.

The proposed self-adaptive software model introduces the software adaptation process concept, a general and complete adaptations picture, allowing its understandability, maintainability, evaluation, reuse, and variability. Reasoning for these integrated adaptations for various purposes will be facilitated. The adaptations are not only defined as separated and independent reactions to context events, but all the possible adaptations (each one is defined by an adaptation program) are integrated in an adaptations process, giving a complete adaptations picture and controlling their coherences and integrity. The figure 2b presents a lifecycle adaptations process for the System_Car and the figure 2c presents its context adaptations process. Where as the figure 2d presents an integrated adaptations process, combining all System_Car adaptations processes: Operating, lifecycle, and context within the whole System_Car lifecycle.

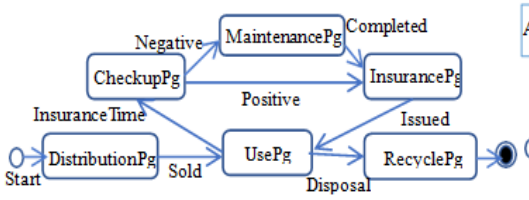

Figure 2b. A Car lifecycle adaptation process. DistributionPg, UsePg, .. are adaptation programs for the corresponding state.

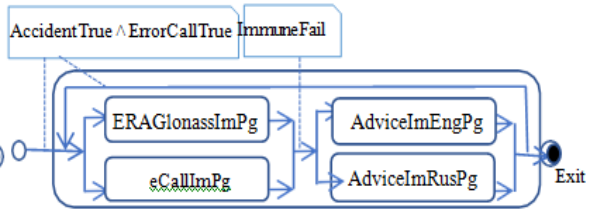

Figure 2c. A Car context adaptation process . ERAGlonassImPg, eCallImPg, .. are adaptation programs for the corresponding context events.

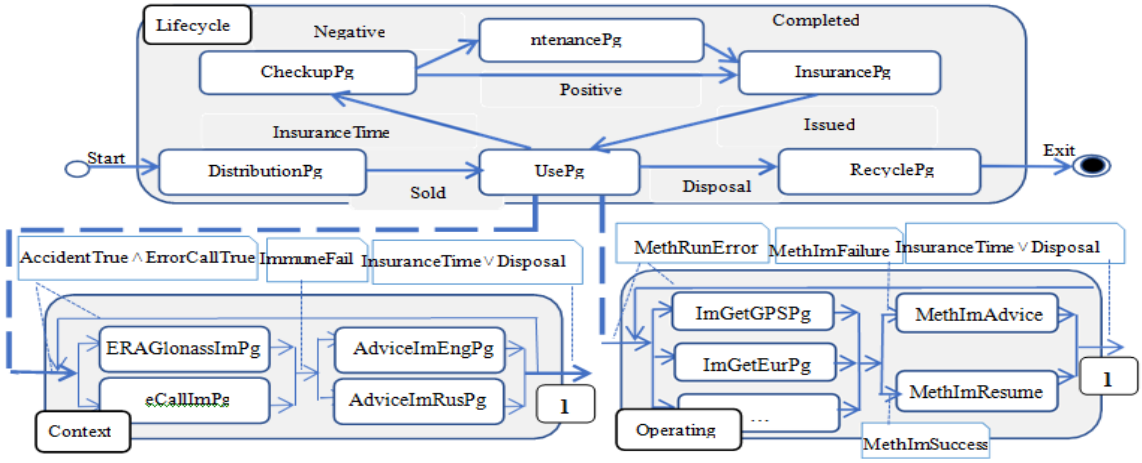

Figure 2d. A System_Car Integrated adaptations process: Operating, lifecycle, and context.

1 → continue to checkup Pg or to RecyclePg according to the generated event

4.2. An Aspect-Based Model

The proposed model presents software as an object having three aspects: Operation, Adaptations, and Adaptations manager (Figure 3a).

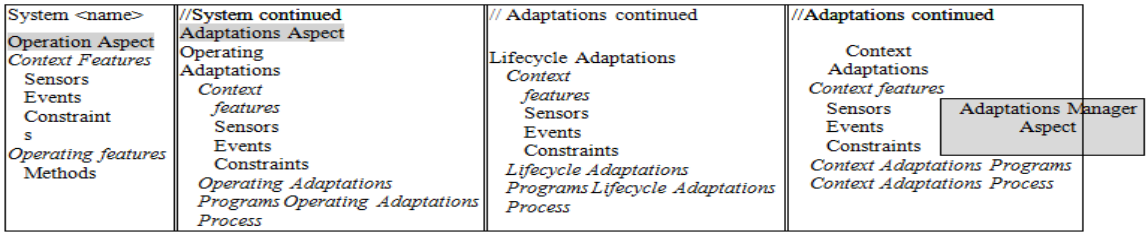

Figure 3a. A holistic software model integrating its operation, adaptations, and manager aspects

Operation Aspect
This aspect defines the normal operating way of software that supports provided functions. It is composed of context and operating features. The Context features define the context the software is operating

in, its sensors, generated events, and imposed integrity and coherence constraints. The operating features define the functions that are provided by the software (Figure 3b).

Adaptations Aspect
This aspect defines the adaptations way of software. It is composed of Operating, Context, and Lifecycle Adaptations aspects.

The operating/context adaptations are operating/context aware and dynamic adaptations (Figures 3c and 3e). They are handled through the whole immunity system (IIS and AIS). These aspects are composed of Context features, Operating/Context Adaptations Programs, and Operating/Context Adaptations Processes. The Context features define the software is operating/context adapting in: its operating/context sensors, operating/context

```
System Car
Operation Aspect
  Context Features
    Sensors // generate events from the context
      Location: {Europe, Russia}, Communication: {Call, eCall, ERAGlonass}; Diagnostic: {Text Error}, Language :
      {English,Russian}, CarState: {distribution, use, checkup, maintenance, insurance, recycle}
    Events
      eCall: ERAGlonass;
    Constraints //Properies which must be always verified by the systems
      Call → eCall ∨ ERAGlonass;
      eCall → Communication.GetGPS() ^ Location.GetEur() ^ Language.GetEnglish()∧ Diagnostic.GetEnglish()
      ERAGlonass → Communication.GetGlonass() ∧Location.GetRus()∧ Language.GetRussian()∧Diagnostic.GetRusian
      ()
  Operation features
    Methods //functions provided by the system
    Location.GetGPS(); Location.GetGlonass(); Location.GetEur(),
    Location.GetRus(). Diagnostic.GetEnglish ();Diagnostic.GetRusian ();
    Communication.GetGPS();Communication.GetGlonass();
    Language.GetEnglish (); Language.GetRusian()
    CarState.Distribution();
    CarState.Use();CarState.Checkup();CarState.Maintenance();CarState.Recycle();
    Sensor.Enable();Sensor.Disable();
```

Figure 3b. A System_Car operation aspect.

generated events and operating/context imposed integrity and coherence constraints. The Operating/Context Adaptations Programs define adaptations programs that are required at any operating/context event requiring adaptation, and Operating/Context Adaptations Process (Figures 2c and 2d) defines the whole software operating/context adaptations process by composing the adaptations programs of each operating/context relevant event. The model introduced two new concepts defined as operating/context events: Resume and advice. Resume might be generated by both IIS and AIS when a software is correctly reconfigured and should be resumed and Advice might be generated by only AIS at failure of IIS.

The lifecycle adaptations are context free and static adaptations (Figure 3d). They are handled through the IIS. This aspect is composed of Context features, Lifecycle Adaptations Programs, and Lifecycle Adaptation Process. The Context features define the context the software is lifecycle adapting in: its lifecycle sensors, lifecycle generated events, and lifecycle imposed integrity and coherence constraints. The Lifecycle Adaptations Programs define adaptations programs that are required at any lifecycle step and Lifecycle Adaptations Process (Figure 2b) defines the whole software lifecycle adaptations process by composing the adaptation programs of each step.

Adaptations Manager Aspect

Each configuration includes its own self-adaptation manager whereas it is a separated subsystem in current works [16, 17, 18]. This manager integrates, at the run time, the operation, lifecycle, and context adaptations aspects according to the events occurrence (Figure 4). At the initial state (generation of a configuration instance), only the Self-Adaptation Manager is activated. It processes

following the MAPE functionality and deals with lifecycle and context events. The MAPE loop is processed as follows:
(M): monitors the lifecycle /operating/ context: by detecting events generated from sensors.
(A): analyzes the known data for change: by identifying the corresponding lifecycle/ operating/context adaptations programs.
(P): plans the adaptations process: by selecting the corresponding actions in the lifecycle / operating/context Adaptations processes.
(E): execution of the adaptation: by the Self-Adaptation Manager.

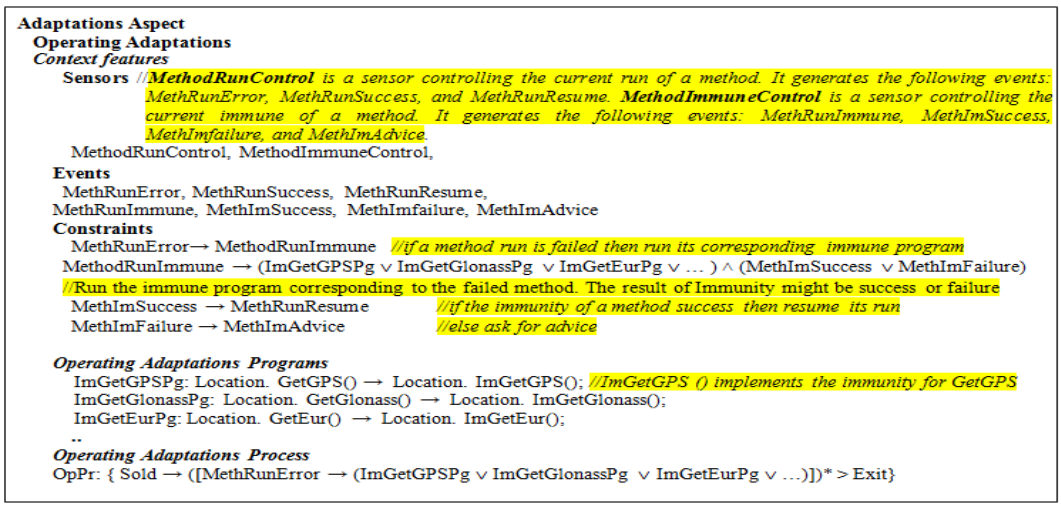

Figure 3c. A System_Car Operating adaptations

Adaptations Aspect
Lifecycle Adaptations
Context features
Sensors
// Share CarState: {distribution, use, checkup, maintenance, insurance, recycle}
CarManagement: { Sold, InsuranceTime, Disposal, Negative, Positive, Completed, Issued}
Events
Sold, InsuranceTime, Disposal, Negative, Positive, Completed, Issued
Constraints
Distribution∧ Sold→ Use
Use ∧ InsuranceTime → Checkup
…

Lifecycle Adaptations Programs
DistributionPg: CarState.Distribution() ^ Sensor.Enable(CarState) ^ (CarState = Distribution);
…

Lifecycle Adaptations Process
LCPr: {[Start→DistributionPg] >

Figure 3d. A System_Car Lifecycle adaptation aspect

Context Adaptations
Context features
Sensors
Accident: {true, false}, ErrorCall: {true, false}
Events
AccidentTrue, ErrorCallTrue, ErrorCallFalse, Advice, ImmuneSuccess, ImmuneFail, Resume
Constraints
AccidentTrue → Call *// eCall or ERAGlonass will be lauched.*
Call → ErrorCallTrue ∨ ErrorCallFalse
ErrorcallTrue → CallImmune *//if a run time error occurs then the corresponding immune program is activated*
CallImmune ∧ ImmuneSuccess → Resume *//Else the corresponding method is resumed*
CallImmune ∧ ImmuneFail → Advice *//if the immunity failed then advice will be required*

Context Adaptations Programs
ecallImPg: CallImmune ∧ Language-GetEnglish() → ERAGlonass
ERAGlonassImPg: CallImmune ∧ Language-GetRussian () → eCall
AdviceImEngPg: Advice ∧ Language-GetEnglish () → Diagnostic-GetEnglish ()
AdviceRusEngPg: Advice ∧ Language-GetRussian () → Diagnostic-GetRusian ()

Context Adaptations Process
LCPr: { Sold → ([AccidentTrue ∧ ErrorCallTrue → (ecallImPg ∨ ERAGlonassImPg)] >
[ImmuneFail → (AdviceImEngPg ∨ AdviceRusEngPg)])*> Exit}

Figure 3e. A System_Car Context adaptation aspect

The manager enforces the following rules on its associated software configuration (Figure 4).

Initial state: Self-Adaptation Manager Activation
R1. At configuration constructor
- Enable Self-Adaptation Manager.
- Enable all sensors.

Loop on Lifecycle/operating/context events
R2. Sensors generate corresponding events. R3. Occurred events are processed in parallel.
R4. Self-Adaptation Manager will check the event either it is a lifecycle/ operating/context event.
R5. The process of Lifecycle event (by IIS) will deal with the event regarding to the MAPE functionality as presented before.

R6. The process of operating/context event (by ISS and eventually by AIS) will deal with the event regarding to the MAPE functionality as presented before. It Might be enforced by Advice and Resume.
R7. Repeat R2 – R6 until the occurrence of the destructor event.
Final state
R8. At configuration destructor: The Self-Adaptation manager will stop.

5. EVALUATION

The presented work in this paper is a continuation of that presented in [6] which, inspired by genetics, introduced preplanned software adaptations. These adaptations are defined at the software design and triggered at its run time. They present expected and timed changes due to lifecycle steps. The recent literature reviews [16, 17. 18] present several meaningful aspects of a huge number of current approaches to self-adaptive and self-awareness software. All of the presented approaches deal only with context changes (lifecycle and operating case were missing) according to <event, action> individual rules handled by MAPE-K loops [11, 20]. In general, only known events (certain and uncertain) are handled. Rarely the unknown events are approached [12] and they are generally supported by self-learning techniques.

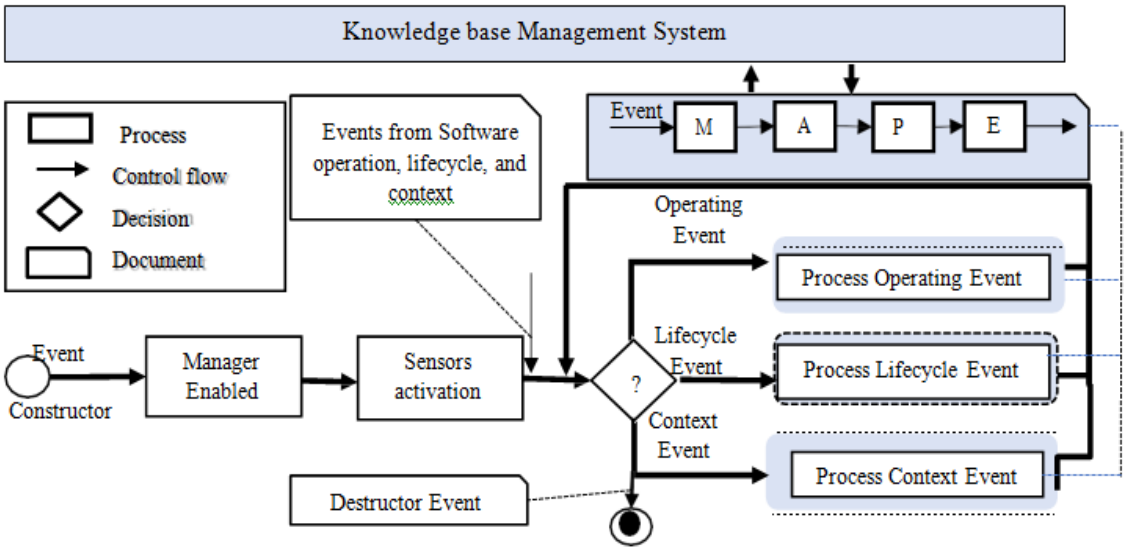

Figure 4. Self-adaptive software Manager Process

As a conclusion to their systematic literature review [11], Elhabbash et al. stated the following: (1) "A holistic self-aware system model that can consolidate various aspects of computational self-awareness based on the various and different inspirations of self-awareness is an open challenge that is worthy of future research". (2) "The majority of studies named only their source of inspiration, without a clear description about how self-awareness in software engineering is inspired by nature science".

In fact, unlike similar works, this paper proposes a holistic aspect-based software model which integrates the software operating, its adaptations (lifecycle, operating, and context), and associated manager. Through the newly introduced concept of adaptation process, this manager (based on MAPE-K) completely controls the software and integrated adaptations throughout the whole software lifecycle. The bio-inspiration of the model was clearly justified as required in

challenge 2 [11]. The adaptation based on unknown events is modeled through the advice and external recovery concepts, which are not yet completely modeled.

The complexity and quality of adaptations, in this work, are handled at the level of Adaptation Programs and Adaptation Processes of operating, lifecycle, and context adaptations. The Self-adaptive software Manager Process is only a triggering engine.

6. CONCLUSION

The recent literature reviews clearly show: (1) the need for a holistic self-adaptive software model integrating its different aspects. This paper proposes a holistic model that integrates the operating, adaptations, and adaptations manager aspects. (2) The limitation of adaptations to context changes. The proposed model covers all possible adaptations: operating (failures of software provided functions), lifecycle (predefined adaptations required in some software lifecycle steps), and context (adaptations required in response to context changes events). (3) The absence of a general and complete adaptations picture allowing its understandability, maintainability, evaluation, reuse, and variability. Reasoning for these adaptations for various purposes will be facilitated. The presented work introduces the concept of software adaptations process integrating the specifications of all the above kind of adaptations. (4) The need to an explicit and detailed link with resources the self-adaptive software is inspired from. In fact, this work shows explicit trace to its pure bio-inspired origins (operating, lifecycle, and context adaptations through the Innate and Adaptive Immune components). (5) A usual limitation to known events. The modeling of advice and external recovery dealing with unknown events by the Immune Adaptive manager component is in progress and constitutes an open research issue.

The presented work is a continuation of that presented in [6] which, inspired by genetics, introduced preplanned software adaptations. These adaptations are defined at the software design and triggered at its run time. They present expected and timed changes due to lifecycle steps. However, this work adds functional and environment adaptation and integrates them in a holistic model.

REFERENCES

[1] M. Hachicha et al. Translation of UML Models for Self-adaptive Systems into Event-B Specifications Advances in Intelligent Systems and Computing, 941, pp. 421-430, 2020

[2] A. Elhabbash et al. Self-awareness in Software Engineering: A systematic literature review. ACM Transactions on Autonomous and Adaptive Systems. 14(2),5, 2019

[3] O. De Sousa et al. Quality evaluation of self-adaptive systems: Challenges and opportunities, ACM International Conference Proceeding Series, pp. 213-218, 2019

[4] D. Rossi, Dynamic high-level requirements in self-adaptive systems. Proceedings of the ACM Symposium on Applied Computing. pp. 128-137, 2018

[5] M. Hachicha, Modelling, specifying and verifying self-adaptive systems instantiating MAPE patterns. International Journal of Computer Applications in Technology, 57(1), pp. 28-44, 2018

[6] E. Naffar and S. Ghoul. A Genetic Framework Model for Self-adaptive Software, Journal of Software Engineering, 11 (3): 255-265, 2017

[7] A. Mostafa, A cognitive adaptive artificial immunity algorithm for database intrusion detection systems. Journal of Theoretical and Applied Information Technology 97(16), pp. 4387-4400, 2019

[8] P . Saurabh, Immunity inspired cooperative agent based security system. International Arab Journal of Information Technology,15(2), pp. 289-295. 2018

[9] Saurabh Praneet and Verma Bhupendra, (2016). An efficient proactive artificial immune system based anomaly detection and prevention system, Expert Systems with Applications, Elsevier, vol. 60, pp. 311–320. 2016

[10] Banerjee Soumya, (2017). An Artificial Immune System Approach to Automated Program Verification: Towards a Theory of Undecidability in Biological Computing, PeerJ Preprints, 2017
[11] A. Elhabbash et al. Self-Awareness in Software Engineering: A Systematic Literature Review. ACM Transactions on Autonomous and Adaptive Systems (TAAS) Volume 14 Issue 2, December 2019
[12] C. Krupitzer et al. Comparison of Approaches for developing Self-adaptive Systems. In Proceedings of . ACM, New York, NY, USA, 14 pages. 2018
[13] M. Roger and A. Schmidt Self-adaptive Systems Driven by Runtime Models: A Systematic Literature Review of Approaches. International Conference on Software Engineering and Knowledge Engineering, SEKE, pp. 248-253. 2017
[14] J. Mauro et al. Context-aware reconfiguration in evolving software product lines, Science of Computer Programming 163 (2018) 139–159
[15] J . Horcas, Software product line engineering: A practical experience ACM International Conference Proceeding Series A, 2019
[16] L. Sompayrac,. How the Immune System Works, Immune System-anatomy & histology. Wiley. 2016
[17] S.Maksuti et al. Self-adaptation applied to MQTT via a generic autonomic management framework. Proceedings of the IEEE International Conference on Industrial Technology, 2019-February,8754937, pp. 1179-1185, 2019
[18] J. Oukharijane et al. Towards a new adaptation engine for self-adaptation of BPMN processes instances, ENASE 2019 - Proceedings of the 14th International Conference on Evaluation of Novel Approaches to Software Engineering, pp. 218-225, 2019
[19] E . Lee et al. Self-Adaptive Framework Based on MAPE Loop for Internet of Things. Sensors. Jul 7;19(13), 2019.
[20] M. Hachicha. Modelling, specifying and verifying self-adaptive systems instantiating MAPE patterns. Int. J. Computer Applications in Technology, Vol. 57, No. 1, 2018