Designing Self-healing Software Using Role-Based Multi Agent System Approach

Falahah1*, Kridanto Surendro2, Wikan Danar Sunindyo3
1,2,3 School of Electrical and Information Engineering, Bandung Institute of Technology, Jl. Ganesha 10.
Bandung, Indonesia.

Email: *falahah@widyatama.ac.id

Abstract. Multi-Agent System (MAS) is one of the popular approaches on modeling behaviour and states in complex system, and has been widely implemented in various field, both on modelling computer-based system or social system condition. There are many approaches and ways on modeling MAS, such as Gaia, OORam, Tropos, and MaSE. In general, the approaches usually starting from defining agents then followed by defining the attributes and roles of each agent. The problem could arise when some agents can share the same roles, because the roles should be defined on each agent. This problem can be resolved by using Role-based MAS or RoMAS approach, that emphasized on defining the role then assigning agent into the defined roles. Using this approach, we can share the role among agents and the agents could build or designed separated from the roles. On this research, we designed the conceptual model for self-healing software that will be modeled using RoMAS. The model show organization of roles that has 3 main roles which are logger, controller and executor. We also identify the breakdown for each roles and it has 3 generic roles such as getter, comparator and setter. The generic roles have responsibilities to collect data, control the system and execute the recovery plan if the system fail. By defining the role, we can propose the architecture of self-healing software that can be independent from agents that would be assigned for executing the role.

1. Introduction

Self-healing software (SHS) is a software that has capability to “heal” the system by detecting and reacting on system fault or malfunction [1]. The software can examine the failure and execute the appropriate correction to recover the system functionality. It can be achieved by three main steps which are: detecting, diagnosing and recovering and it’s also fulfilled the MAPE-K framework as the self-healing attribute is part of self-* concept in autonomic computing [2]. Usually, self-healing software is an additional feature in a system and to be self-healed, the system should meet certain criteria such as it should be exposed their status so it can be monitored.

The research of self-healing kept arise since the concepts first published by IBM on 2000’s on their concept of autonomic computing [3]. There are many approaches has been proposed and vary types of self-healing software has been developed since then. The proposed of SH software that has been developed usually implemented on specific area such as on mobile application system, network system, web services system, and many more.

Along with the grow of computer, network and system, the needs of self-healing software still exist in many software environments, either in standalone software or in complex and distributed system such as in cloud or network-based application. The approach on self-healing behaviour can be categorized
into four types which are error handling, manage the flow of information, adjust problematic data, and retry [4]. The approach on self-healing software development is vary depends on the environment of SH software will be implemented. Some research area on SH development include implementation of SH in embedded system, operating system, network, mobile application and many more.

Implementation of multi agent approach on modelling and designing self-healing software has been proposed by some researchers such as [5][6][7][8]. One of this approach is role-based multi agent model. The first step on role-based multi agent modelling software is to define the role of the agent, then assign the role into the agents. On role-based multi agent approach, we can design the system more flexible because it resulted the system the role and we can assign agent separately from the system, the agents can be another software or human actor [9].

On this research we propose the conceptual model of self-healing software using role-based multi-agent approach. The proposed model will be used as a framework for preparing formal model and building the prototype of model in specific area implementation. We use role-based multi agent system (ROMAS) as an approach on identifying the role needed for defining self-healing software. On this paper we identify the role, role organization and role relation, but we did not assign the agents yet and meta-code for the agents.

2. Method
2.1. Multi Agent Software Development Approach
Generally, step in building agent-based software is similar with others software development method, which consist of analysis, design, build and deploy. The most important step on multi agent modelling is lying on modelling the problem space and design the solution, which using agents and its behaviour. Several method has been published and used such as Gaia, Tropos, MaSE, and Promotheus. Some of specific activities are not exist in Gaia method are improved by Blanes et.al, who add requirement identification step, which called RE4Gaia. RE4Gaia consists two step in modelling, which are modelling requirement and analysis requirement [10]. Modelling requirement consist of four things which are mission statement, functional refinement tree (FRT), requirement role model (RRM), and domain model (DM). Requirement analysis consist of describing the activity diagram, environment model and defining the organization rule.

The specific phase in RE4Gaia methodology in completing requirement identification techniques which is role identification or modelling role. The aim of role identification is to identify the possibilities of building role hierarchy. Role hierarchy exist on ROADMAP methodology which are extended on Gaia method before release of Gaia v.2 [11]. On ROADMAP method, the system can be considered as computer organization which are consist of role hierarchy and agent, as shown in figure 1.

![Hierarchy model of Roles and Agents on ROADMAP](image)

**Figure 1.** Hierarchy model of Roles and Agents on ROADMAP [11]

2.2. RoMAS
RoMAS (*Role-Based Multi Agent System*) is an approach on role-based multi agent system. The methodology consists of six steps which are [12]:

1. Defining use case
2. Identify the role from use case
3. Define the role organization
4. For each role, if there is no agent, then continue to step 5, if there is an agent, then:
   a. Assign role to the agent
   b. Define the dynamic attribute that connecting the role and the agent
   c. Continue to step 6
5. Assign the agent for the role, continue to step 4
6. Define the meta-code for the agent that already connected to the role

We can draw the step above as shown in figure 2.

![Figure 2. Steps in Modeling Multi Agent System using Role-based approach](image)

On RoMAS, actor is a role and is defined in beginning of modeling step, agent identification is assumed as domain for problem that needs to solve. RoMAS also defined the relationship between agent and role, through role organization and role space. Role organization is defined in conceptual level as a hierarchical structure of role with relation such as inheritance, aggregation, generalization and association. Role space consists of role instance that can be connected or disconnected from the agent when the system is running.

3. Results and Discussion
On this research, we will implement the RoMAS method to define the role-based model for self-healing software. It starts from defining the problem area, use case and followed by identify role from the use case. Generally, we can describe the state of the system according to self-healing software implementation as a state chart diagram as shown in figure 3.

We adopt the framework from RE4Gaia using RFT for identifying the role based that needed for self-healing software. The software will run to support certain goal in a sequential business process. On this conceptual model, we assume that the agents will collaborate to achieve the goal or goal-based agent. The agent will evaluate the recent state and compare it into the goal. Based on the evaluation result, the agent will choose some alternative action to keep the goal on the track. Searching and selecting the action is part of decision-making process which are internal agent’s capabilities. Goal-based approach is considered more practical when its implemented into business process support, because the parameter for supporting decision process can be expressed explicitly and easy to modify.
Figure 3. System state when SH software is implemented

Based on the requirement above, we can identify the functional of self-healing system and describe it into use case diagram as shown in figure 4. We can define 3 actors (representing the role) which are logger, controller and executor. In the picture, the roles are shown as rounded rectangle. Each role will perform certain process such as capture data, store data, detect, diagnose and recover, as a basic process for self-healing system.

Figure 4. Use Case Diagram Software Self-Healing

Detail properties of Roles can be identified by define the goal, attribute and responsibility. Figure 5 show the example of detail properties of logger. We identify the main goal of logger is data logging, and it can be fulfilled by identify sets of performance parameters. The parameters can be defined later based on key metrics for system performance. Example of key metrics can include: database connection, response time, endpoint success, error and exceptions, and much more.

The next step in RoMAS approach is construct role organization. Role organizations show the structure of role, interaction and communication between roles, also specialization, generalization, inheritance, and associations. On role-based approach, we have to define the specific role that execute
specific goal or task. Generally, there are three specialize roles that can exist in each role above, which are: getter, comparator, and setter. Getter is the role that has responsibility to read the information, comparator has responsibility to compare, analyse, classify and recommend, and setter has responsibility to send the alert and present the result. In each main role, for example, the logger role, we can define getter, comparator and setter, as well as for detector, diagnose and executor. For executor, we add the new role which is executor, that has responsibility to execute the recovery action if it is suitable to execute. Figure 6 show the role organization in self-healing software.

We define the goal, attribute, services and responsibility for each role as shown in Table 1. As we can see from the role organization diagram, there is three common roles that can be derived from logger, controller and executor, which are getter (that has responsibility to get data/information), comparator (compare between input into standard or existing library, and to analyse the match pattern), and setter (set the status, set the information, or record the status as the result of comparison or pattern analysis).

Description and properties of three common roles above is written in separated table (Table 2). Table 2 also show the map between task of each common role into higher role or its parent roles.

![Figure 5. Role Properties](image-url)
Figure 6. Roles Hierarchy Organization

Table 1. Role Properties

| Role   | Main Goal       | Description                                    | Attribute       | Service            | Responsibility         |
|--------|-----------------|-----------------------------------------------|-----------------|--------------------|------------------------|
| Logger | Data logging    | Logging data from targeted system             | Key_metric      | read(), classify(), store() | periodic_operation, continue_operation |
| Controller | Monitoring system | Monitoring the status of system health and performance | Key_metric, status | Monitor(), Analyse(), Report() | periodic_operation, continue_operation |
| Detector | Detect the failure | Detect the system failure based on key metric | Read(), Alert() | Read(), Alert() | periodic_operation, continue_operation |
| Diagnose | Diagnose the failure | Diagnosing the causes of failure system | Read(), Alert() | Read(), Alert() | Keep_alerting |
| Executor | Execute action | Execute recover action | Failure_model, recover_action | Match(), Recommend() | Running_action_plan |

Table 2. Common Roles in SH Software

| Role   | Services          | Description                                                                 | Logger Task in Role | Detector Task in Role | Diagnose Task in Role | Executor Task in Role |
|--------|-------------------|------------------------------------------------------------------------------|---------------------|-----------------------|------------------------|------------------------|
| Getter | Read()            | Role for reading information or data                                           | Reading data from the system targeted | Reading data from logger | Reading data from logger | Reading data from logger |
| Comparator | compare() | Compare between 2 conditions, compare data read from the targeted system with standard on key metric | Compare information from logger into match failure condition | Analyze the causes of failure | Classify the failure condition | Compare the failure condition with match common solution |
| Comparator | analyze() | Compare between 2 conditions, compare data read from the targeted system with standard on key metric | Compare information from logger into match failure condition | Analyze the causes of failure | Classify the failure condition | Compare the failure condition with match common solution |
| Comparator | classify() | Compare between 2 conditions, compare data read from the targeted system with standard on key metric | Compare information from logger into match failure condition | Analyze the causes of failure | Classify the failure condition | Compare the failure condition with match common solution |
| Comparator | recommend() | Compare between 2 conditions, compare data read from the targeted system with standard on key metric | Compare information from logger into match failure condition | Analyze the causes of failure | Classify the failure condition | Compare the failure condition with match common solution |
| Setter | alert(), record(), present(), report() | Set the data into storage, display or as input for next process | Set the status for each key metric, Set alert, Record system state, Set alert, Present system performance | Set alert, Record system status, Present system performance | Set alert, Record system status, Present system performance | Set alert, Record system status, Present system performance |


4. Conclusion

Based on result and discussion above, we can conclude that method RoMAS can be used as an approach for modelling self-healing system. The model consists of 3 main role which are logger, controller and executor and organization of model that show the relation between the role. The controller role can be breakdown into 2 main role which are detector and diagnose. The model will match with SH software that consist 3 steps: detect, diagnose and recover. By using role-based approach, we can identify common roles that can exist in each role, which are getter, comparator and setter. The common roles will have similar task on each higher role where the common role is assigned or called. For each role that had been identified, we define the properties of the role which are the goal, attribute, services and responsibility. By propose this model, it will help us to design the prototype for realising the model into real work software.

References

[1] Gorla, A. 2007, September. Towards design for self-healing. In Fourth international workshop on Software quality assurance: in conjunction with the 6th ESEC/FSE joint meeting (86-89). ACM.
[2] Ahmed, S., Ahamed, S. I., Sharmin, M., and Hasan, C. S. 2009. Self-healing for autonomic pervasive computing (285-307). Springer US.
[3] Koopman, P. 2003. Elements of the self-healing system problem space, Workshop on Software Architectures for dependable System, WADS 2003, 1–6.
[4] Borchardt, V. 2012. Recovery Oriented Computing in Distributed Systems.
[5] Zidan, A, El-Saadany, E., 2012, A Cooperative Multiagent Framework for Self-Healing Mechanisms in Distribution Systems, IEEE Transactions on Smart Grid 3(3), pp.1525-1539
[6] Bagheri, H, Architectural Approaches for Self-Healing Systems Based on Multi Agent Technologies
[7] Park J., Youn H., Lee E. 2005 A Multi-agent Based Context Aware Self-healing System. In: Gallagher M., Hogan J.P., Maire F. (eds) Intelligent Data Engineering and Automated Learning - IDEAL 2005. IDEAL 2005. Lecture Notes in Computer Science, 3578. Springer, Berlin, Heidelberg
[8] Bagheri, H., Torkamani, M. A., and Ghaffari, Z. 2013. Architectural Adan roaches for Self-Healing Systems Based on Multi Agent Technologies. International Journal of Electrical and Computer Engineering (IJECE), 3(6), pp.779-783.
[9] Xu, H., & Zhang, X. 2005, May. A Methodology for Role-Based Modeling of Open Multi-Agent Software Systems. In ICEIS (3), pp. 246-253.
[10] Blanes D., Insfran E., Abrahão S. 2009 RE4Gaia: A Requirements Modeling Approach for the Development of Multi-Agent Systems. In: Ślęzak D., Kim T., Kiumi A., Jiang T., Verner J., Abrahão S. (eds) Advances in Software Engineering. ASEA 2009. Communications in Computer and Information Science, vol 59. Springer, Berlin, Heidelberg
[11] Juan, Thomas, Adrian Pearce, and Leon Sterling. "ROADMAP: extending the gaia methodology for complex open systems." Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1. ACM, 2002.
[12] Yan, Q., et al. "ROMAS: A role-based modeling method for multi-agent system." Proceedings of International Conference on Active Media Technology. 2003.