Requirements Conflict Detection and Resolution in AREM Using Intelligence System Approach

Rosa Delima[1], Retantyo Wardoyo[2], Khabib Mustofa[3]

[1]*Departement of Informatics, Universitas Kristen Duta Wacana, Indonesia*
[2,3]*Departement of Computer Science & Electronics, Universitas Gadjah Mada, Indonesia*
[1]rosadelima@staff.ukdw.ac.id, [2]rw@ugm.ac.id, [3]khabib@ugm.ac.id

Abstract - Requirements engineering (RE) is the process of defining user requirements that are used as the main reference in the system development process. The quality of the RE results is measured based on the consistency and completeness of the requirements. The collection of requirements from multiple stakeholders can cause requirements conflict and have an impact on the inconsistency and incompleteness of the resulting requirements model. In this study, a method for automatic conflict detection and resolution in the Automatic Requirements Engineering Model (AREM) was developed. AREM is a model that automates the process of elicitation, analysis, validation, and requirements specification. The requirement conflict detection method was developed using an intelligent agent approach combined with a Weighted Product approach. Meanwhile, Conflict resolution is made automatically using a rule-based model and clustering method. Testing the ability of the method to detect and resolve conflicting requirements was carried out through five data sets of requirements from five system development projects. Based on the test results, it is known that the system is able to produce a set of objects that have conflicts in the data requirements. For conflict resolution, experiments were conducted with five conflict resolution scenarios. The experimental results show that the method is able to resolve conflicts by producing the highest completeness value, but the results of conflict resolution also produce a number of soft goals. The success of the method in detecting and resolving conflicts in the model is able to overcome the problem of inconsistencies and incompleteness in the requirements model.

Keywords: Requirements engineering; AREM; conflict detection; conflict resolution; intelligence agent; weight product; rule-based; K-means clustering

I. INTRODUCTION

Requirements Engineering (RE) is an important stage in the development of software. RE consists of several stages, such as elicitation, analysis, specification, validation, and requirement management [1]. Automation of several stages of the RE can increase the effectiveness of the RE. Automatic Requirements Engineering Model (AREM) is a model that automates the elicitation, analysis, specification, and validation stages on RE. AREM was developed by researchers with the aim of increasing the effectiveness of the RE process by minimizing the role of system analysis or requirements engineer on the RE.

AREM was developed by integrating several approaches, namely Goal-Oriented Requirements Engineering (GORE), intelligent agents, text processing, decision support systems, rules-based, and intelligent systems approach. The model input is in the form of data requirements from each stakeholder. To obtain complete requirements data, it is necessary to collect requirements from various stakeholders. Stakeholders include organizational management and end-users. Differences in requirements between stakeholders are very likely to occur because each stakeholder has a definition of the requirements that may be different or even contradictory to the requirement of other stakeholders.

The GORE approach is used as a standard form of input for the model. In GORE, there are several elements, namely goal, task, and operational [2]. The goal is the main element that represents the requirements of the stakeholders. Conflicting goals between stakeholders will affect the consistency and completeness of the model [3-4]. Therefore, AREM must be able to detect conflicts between elements in the data requirements and followed by requirements conflict resolution.

Conflict detection is an essential part of the validation model [5]. In this study, a novel method was developed to detect conflict of requirements in AREM. The method was developed by integrating the intelligence agent and weighted product approaches. The intelligence agent approach is used to carry out the process of meeting the needs based on the behaviour of the agent from a number of test cases, while the weighted product is used to determine the fulfillment of goals with or relation. This research produces a set of goals or model elements that conflict with other elements. Based on the results of conflict detection in the model elements, an algorithm for conflict handling has been developed so that the model can produce consistent and complete requirements.

Conflict resolution is automatically carried out by using the integration of the clustering method to cluster the level of conflict and a rule-based approach to conflict resolution. The success of conflict handling is measured by the completeness value generated by the model.

This paper consists of four parts, beginning with an introduction which consists of background and related work. The second section contains the method development process and continues with experiments and results. The last section contains the conclusion of the research.

Conflict of requirements is a problem that arises in multiple stakeholders [6]. Conflict of requirements occurs when there is an inconsistency between one requirement and another. A mild form of conflict is in the form of goal divergence. A set of goal statements $(G_1, G_2, \ldots., G_n)$ is declared to be divergent in the domain (Dom) if it can be found that the boundary condition (B) is true, where the goals cannot be met simultaneously. It can be said that the requirements become logically inconsistent if the boundary condition (B) is true and the fulfillment of the goal is false. The logical notation for this condition can be seen in (1) [2].

$$\{G_1, G_2, \ldots \ldots, G_n, B, Dom\} \vDash False \tag{1}$$

There are many causes of the conflict of requirements, such as there are many system's requirements defined, a complex system will have complex requirements as well. Therefore, a conflict of requirements occurs, and requirements from multiple stakeholders are highly at risk for conflicts [4].

Based on the literature study conducted by [4], it is known that many studies have been conducted to detect and resolve conflicts in RE. There are three conflict detection techniques, such as manual, automatic, and general framework. The manual technique performs conflict detection and handling manually or directly by requirements engineers. Meanwhile, in automatic techniques, tools were developed to detect and handle conflicts automatically. The general framework performs conflict detection using special techniques that cannot be categorized into manual or fully automated. In research [4], a review of 22 papers was conducted. They categorized them as follows: eleven papers proposed manual technique, nine papers proposed automatic technique, and two papers proposed a general framework for conflict detection.

Detection of conflicts between objects in RE, particularly the GORE approach, is known as divergence analysis [2]. This process is very important, which will have an impact on the quality of requirements generated by the model. In the GORE study, this area has not received much attention [7]; some studies emphasize the use of obstacles for risk analysis and conflict detection [8-13]. Some others use a formal approach to detect conflicts [14-16].

In this research, we have closeness with the research conducted by [15]. In research [15], they used a formal temporal logic approach to calculate goal conflict and then applied a string model counting technique to estimate the possibility of conflict. In this study, the conflict detection process was carried out using an agent approach to detect conflicts from a number of test cases. Then, the model will calculate the number of conflicts for each pair of objects so that a number of objects that experience conflict exceed the specified threshold value.

Conflict resolution is an attempt to deal with conflicts that occur. In RE, conflict resolution is closely related to meeting the requirements defined by stakeholders. Conflict results in incompleteness in the fulfillment of requirements [9-10]. Therefore, conflict resolution must be carried out in order to obtain good quality requirements.

In the GORE approach, completeness is notated as in (2), where all goals (G) are met through the fulfillment of requirements (Req) in the estimated environment (Asm) and the specified domain (Dom) [2].

$$\{Req, \text{Asm}, Dom\} \vDash \text{G} \tag{2}$$

There are many studies on resolving conflicting requirements, including Rehman et.al. [17], who reviewed and applied communication and negotiation in conflict resolution. The same thing was also done by Maysoon and Djamal [3]; they reviewed a number of methods which were categorized into two conflict resolution techniques, namely negotiation between stakeholders and prioritizing requirements. Both of these techniques resolve conflicts manually by prioritizing the role of analysis systems in conflict resolution. For automatic conflict resolution, there are studies that apply rule-based and genetic algorithms methods [3,18].

In GORE, most of the studies apply the obstacle approach to conflict resolution [8-10]. This study, it contributes through the resolution of conflicting requirements by applying the rule-based method with K-means clustering.

II. METHOD

The proposed method to detect conflict in AREM uses the intelligence agent approach. The agent architecture consists of an environment that provides input for agent behaviour which will then be stored in the agent-behaviour data. The analysis process is carried out automatically through input from inference triggers in the form of tasks/operations run by agents. Then, the analysis process will be carried out through fact data to provide results in the form of status to triggers and input values for elements in the model. After completing the divergence analysis, then conflict detection is carried out; if a conflict is found, a list of pairs of elements experiencing conflict will be generated, and then conflict handling will be carried out. In the final stage, the agent will provide input back to the environment in the form of requirements data that have been refined and are ready to be used in the next sub-model. The architecture of the agent used can be seen in Fig. 1.

The requirements conflict detection stage is carried out through four stages, such as generating test cases, generating agent behaviour, divergence analysis, calculating object pair conflicts, and producing an output in the form of a list of element pairs that experience conflicts in the model. The stages of requirements conflict detection can be seen in Fig. 2.

A. Generate Test Cases for Agent Behaviour

A test case is a collection of objects and relations between objects that are formed in the requirements elicitation and requirement refinement stages. In the GORE approach, the requirements data are in the form of a goal tree, where the root of the tree is the main goal. Each branch is a representation of each element of the requirements data. Fig. 3 is a goal tree that represents system requirements.

In Fig. 3, it can be seen that the requirements data in the model consist of goal, task, operational, and resources (actor and resources). The relation to the element is in the form of the contribution that the element gives to the element above it. The lowest element is a set of facts that form the basis for tracking the fulfillment of goals on the model. In the agent approach (Fig. 1), the environment will provide input for agent behaviour. Test case generation is the generation of input values for agent behaviour. Input is generated randomly to determine the value of the behaviour agent. There are two possible values of the fact on the agent, such as true (T) and false (F).

B. Divergence Analysis

After the fact values are known, then a divergence analysis is carried out. The analysis was carried out using the forward chaining algorithm [19], starting from the facts (bottom of the tree) to the root (main goal). The forward-chaining algorithm applied can be seen in Fig. 4 [19].

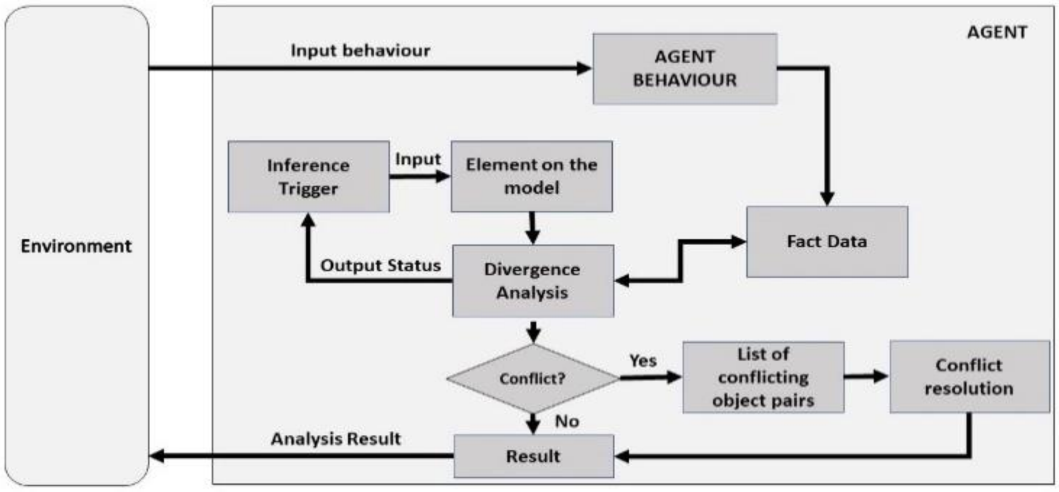

Fig 1. Agent architecture in AREM

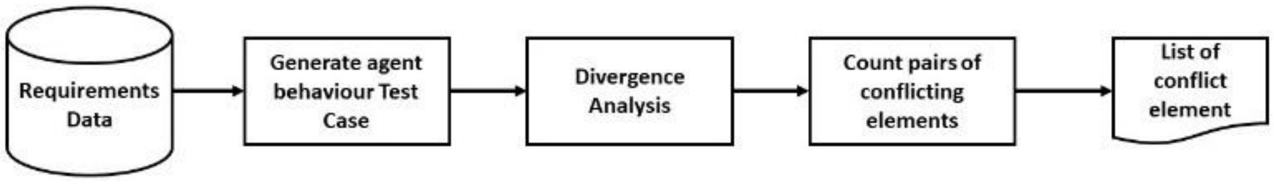
Requirements Data
Generate agent behaviour Test Case
Divergence Analysis
Count pairs of conflicting elements
List of conflict element

Fig 2. Conflict detection stages on model

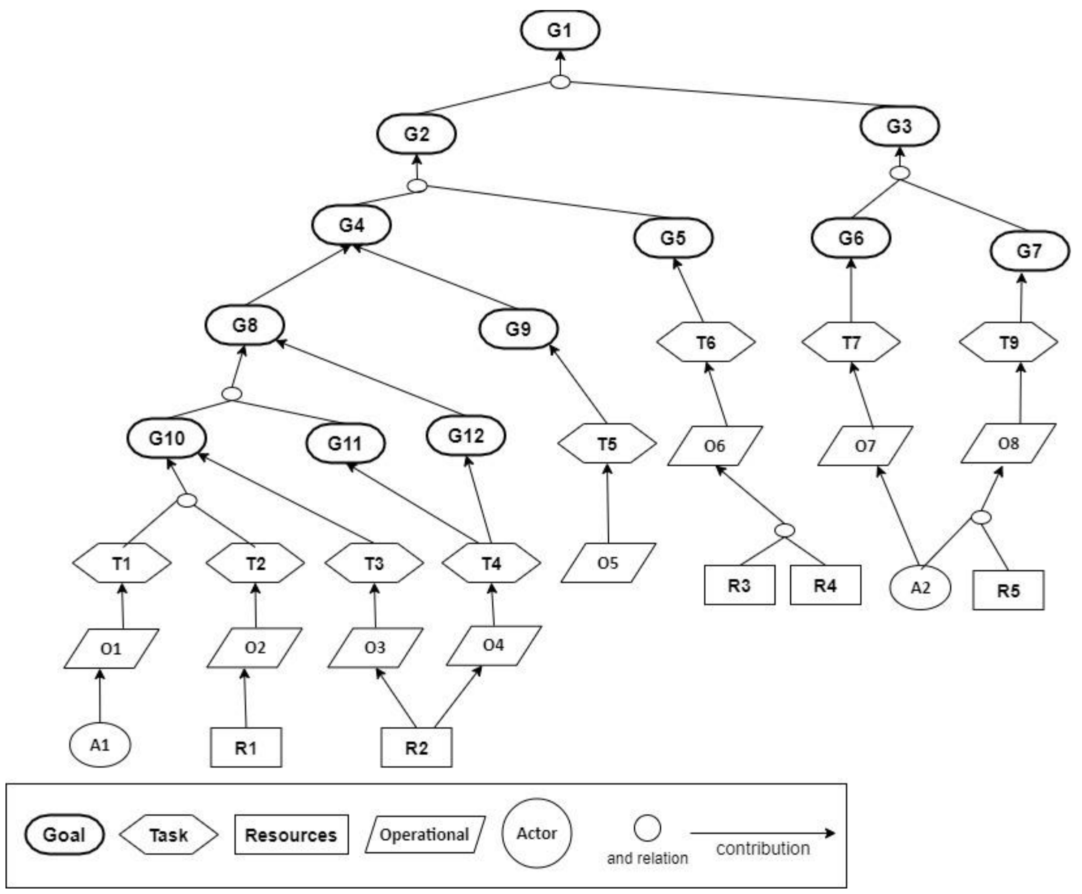

Fig 3. Representation of requirements data in tree shapes

```
function Forward-chaining(Dom, α) returns a substitution or false
inputs: Dom, the domain, a set of object dan relation between object definite
clauses α, the query, an atomic object

while true do
    new ← { } // The set of new object inferred on each iteration
    for each rule in Dom do
      if ($p_1 \wedge \ldots \wedge p_n \Rightarrow q$)← rule with AND relation then
          for each θ such that
            $SUBST(\theta, p_1 \wedge \ldots \wedge p_n) = SUBST(\theta, p'_1 \wedge \ldots \wedge p'_n)$
            for some $p'_1, \ldots, p'_n$ in Dom
      elseif ($p_1 \vee \ldots \vee p_n \Rightarrow q$)← rule with OR relation then
          for each θ such that $SUBST(\theta, p_1 \vee \ldots \vee p_n) =$
                                  $SUBST(\theta, \max(p'_1 value, \ldots, p'_n value))$
            for some $p'_1, \ldots, p'_n$ in Dom
      $q' \leftarrow SUBST(\theta, q)$
      if q' does not unify with some sentence already in Dom or new then
                add q' to new
      $\varphi \leftarrow UNIFY(q', \alpha)$
      if $\varphi$ is not failure then return $\varphi$
    if new = { } then return false
add new to Dom
```

Fig. 4 Forward chaining algorithm

In the divergence analysis process, there are two forms of rules that describe the relationship between objects such as AND and OR. As seen in figure 3, goal G_8 gets contribution from G_{10}, G_{11}, and G_{12} with the notation $(G_{10} \wedge G_{11}) \vee G_{12} \Longrightarrow G_8$. The inference process for rules with OR relation is carried out using a Weighted Product (WP) decision support approach. WP is a method for dealing with multiple-criteria decision-making problems [20]. In WP, the priority value of each object in the relation will be calculated. The stages of calculating the priority value of objects in WP can be seen in Fig. 5.

In this research, there are three criteria in the WP method such as number of child nodes (C_1), the number of nodes associated with object down to operational element (C_2), and the number of nodes and resources needed to the value of the object (C_3). On WP, there are two kinds of criteria as profit criterion (C_1 and C_2) and cost criterion (C_3). Then the weight value is determined for each criterion with the provisions of 4, 3, and 2 for each C_1, C_2, and C_3.

The next step is to normalize the weight (W_i) and calculate the weight vector (S_i). The formula for weight normalization can be seen in (3), where normalization is formulated by dividing the weight of the criteria by *i* (W_i) by the number of weights criteria from 1 to n. The calculation of the weight vector can be seen in (4), where X_{ij} is the value for alternative/object *i* and criteria *j*. The results of the vector calculation will produce an object priority order whose value will be used as a reference value for the parent node.

$$W_i = \frac{w_i}{\sum_{i=1}^{n} w_i} \quad (3)$$

$$S_i = \prod_{j=1}^{n} X_{ij}^{W_j} \quad (4)$$

C. Conflict Count

After conducting a divergence analysis for all cases that have been generated, the conflict value calculation of each object in the model will be calculated. The conflict value calculation is done by generating all objects in the model and pairing one object with another object. Therefore, data pairs will be formed for all objects ($P_{(i,j)}$) owned by the model. The conflict value calculation is conducted by calculating the number of cases where node pair $P_{(i,j)}$, value *i* (v_i) is not equal with value *j* (v_j) divided by the number of all test cases generated (*n*). The formulation of conflict_value can be seen in (5).

$$Conflict\_value\_P_{(i,j)} = \frac{number\_of\_case\_where \;\; v_i \neq v_j}{n} \quad (5)$$

The final result of the requirement conflict detection model is a list of object pairs that experience conflict with the conflict_value exceeding the set threshold.

D. Conflict Resolution

Conflict resolution is carried out through two stages, namely the stage of determining groups/clusters from conflicts and handling conflicts. The type of conflict handling for each object is based on two variables, such as the type of object and the magnitude of the conflict value on the object. The types of objects are divided into three types of objects, such as goals, tasks, and operations. Each object is assigned a value of 1, 2, and 3 for goals, tasks, and operations. Meanwhile, the handling of conflict values is done by clustering conflict values.

Conflict values are clustered into three clusters, such as cluster 1 for objects with small conflict values, cluster 2 for objects with average or moderate conflict values, cluster 3 for objects with large conflict values.

The clustering process is carried out using the K-Means approach through the following stages [21] :

- Early initiation. At this stage, the number of clusters is determined, followed by calculating the distance between the node and the initial centroid.
- Repeat until there is no change in the centroid of the data. This stage begins with determining the data cluster according to the closest distance between the node and the centroid. Next, the new centroid is recalculated.
- Perform final cluster data storage. This stage is carried out if the clustering has been convergent. The final result of the cluster becomes the cluster value of the conflict node.

Cluster value generation is carried out using a quarter approach to statistics, namely quarter 1, quarter 2 (median), and quarter 3 [22]. A quaternary approach is an approach that calculates the three midpoints of the data. The equation to determine the quarter can be seen in (6), where *i* is the *i* quarter and n is the number of data

$$Q_i = \frac{i \times (n+1)}{4} \quad (6)$$

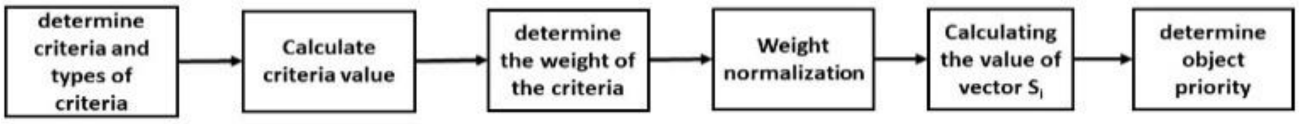
determine criteria and types of criteria
Calculate criteria value
determine the weight of the criteria
Weight normalization
Calculating the value of vector S_i
determine object priority

Fig. 5 Stages for weighted product method

To determine the distance of nodes in each cluster, the Euclidean distance approach is used as in (7), where the value of the cluster (c_i) corresponds to the value of the data in Q1, Q2, and Q3 and x_i is the value of the i node and c_i is the value of the i cluster.

$$D(x_i, c_i) = \sqrt{(x_i - c_1)^2} \tag{7}$$

Conflict handling is carried out based on the type and cluster group of objects. There are three types of conflict handling. The first conflict handling is to make the object a soft element. The soft element is a requirements element in the model that has a lower level of fulfillment than the hard element. This means that the soft elements will be fulfilled after all the hard elements are met. The second conflict handling is to make an OR relation on the object experiencing the conflict. This handling is done through refinement resolution on the model, and the third conflict handling is not handling the conflict, meaning that the conflict will be ignored. The choice of the type of conflict resolution is carried out based on the rules that have been defined. The list of applied rules can be seen in table VIII, in section IV.

III. RESULTS AND DISCUSSION

The experiment was carried out using five system development projects, such as a cooperative information system, a staffing system, a lecture support system, and a financial dashboard system. The requirement elicitation is carried out using standard inputs developed with the GORE approach. Table I is a description of the number of objects in the project used as the experimental data.

A. Data Preparation

In running the algorithm for conflict detection, input data in the form of requirements data is necessary. Based on the requirements data, agent_behaviour data will be generated to support the analysis process. The result of the conflict detection process is a list of object pairs that experience conflict. Referring to the requirements data in the form of a tree in fig. 3, then the requirements data, agent_behaviour data, and the results of the requirements data can be formed, as can be seen in Tables II to IV.

B. Requirements Conflict Detection Process

The first step is to generate test cases. In the experiment, 100 test cases were generated for each project. Then, the agent behaviour is generated for each test case. Generating agent behaviour is carried out on all objects, which are the initial facts on the requirements data. The initial fact is the node in the requirement tree which is at the bottom, above the actors and resources. For example, in Fig. 3, there are fifteen nodes that are initial facts, namely resource objects (*A1, A2, R1, R2, R3, R4, R5*) and operational objects (*O1* to *O8*). Generating True and False values in each case for agent behaviour is done by generating random values.

The next step is a divergence analysis for each test case. The analysis is carried out by conducting a bottom-up search on the requirements tree. The search is carried out based on the rules on the requirements data, which is a representation of the requirement tree. If the rule has an OR relation, the determination of the goal value will be based on the value of the node that has the highest priority. For example, in Fig. 3, there are two rules that have an OR relation, $(\mathrm{G}_{10} \wedge \mathrm{G}_{11}) \vee \mathrm{G}_{12} \Longrightarrow \mathrm{G}_8$ and $(\mathrm{T}_1 \wedge \mathrm{T}_2) \vee \mathrm{T}_3 \Longrightarrow \mathrm{G}_{10}$. To determine the value of G_8 priority values are calculated $(\mathrm{G}_{10} \wedge \mathrm{G}_{11})$ and G_{12}. The calculation of the priority value uses the WP method with stages such as Fig. 4. Based on C_1, C_2, and C_3 which have been defined, the criteria values as in Table V are obtained.

Then normalization of the weights is carried out using equation 3. It is known that if $W = (4,3,2)$ then the normalization results are $W_1 = 0.44$, $W_2 = 0.33$, and $W_3 = 0.22$. Based on the criteria and weight values, the vector S_i uses equation 4, and the obtained result is $S_1 = (4^{0.44})(8^{0.33})(11^{-0.22}) = 2.16$ and $S_1 = (1^{0.44})(2^{0.33})(3^{-0.22}) = 0.99$. Based on vector calculations, it is known that the priority value of the node $(\mathrm{G}_{10} \wedge \mathrm{G}_{11})$ is more than G_{12} therefore the value of the node G_8 will be determined by the value of the node $(\mathrm{G}_{10} \wedge \mathrm{G}_{11})$.

TABLE I
DATASET OF EXPERIMENTS

Project Code	Project Name	The number of requirements data objects			
		Goal	Task	Op	Res
P1	Cooperative information system	62	71	195	140
P2	Staffing system	83	34	26	26
P3	Lecture support system	145	119	174	230
P4	Financial dashboard system	43	40	65	66
P5	Human Resource Management System	218	218	448	470

TABLE II
EXAMPLE OF REQUIREMENTS DATA

Child Object	Parent Object	Child Type	Parent Type
G2 ^ G3	G1	Goal	Goal
G4 ^ G5	G2	Goal	Goal
T1 ^ T2	G10	Task	Goal
T3	G10	Task	Goal

TABLE III
EXAMPLE OF AGENT-BEHAVIOUR DATA

Object	Object Pre-value	Object Post-value
A1	True	True
R1	True	True
O1	True	False
O2	True	False

TABLE IV
EXAMPLE OF RESULT DATA

Object 1	Object 2	Conflict Value
G2	G3	0.56
G2	G4	0.75
G2	T1	0.60
G3	T1	0.90

TABLE V
EXAMPLE OF CRITERIA VALUES

Node	C1	C2	C3
G10 ^ G11	4	8	11
G12	1	2	3

The final step in conflict detection is to calculate the conflict value for each node pair. For example, from the results of the divergence analysis, data is obtained as in Table VI, it is known that there are six test cases and five of the value test cases nodes G_5 and G_6 have different values, so based on equation 4, the conflict value for the pair of nodes is 0.83.

C. Conflict Detection Result and Conflict resolution

The experiment was carried out by generating 100 test cases for each project. The experimental results show that the number of pairs of nodes in each project is strongly influenced by the number of objects in the requirements data. It can be seen in table VII that the P3 project has the highest number of objects, so this project will definitely produce the highest number of pair nodes. Likewise, with data processing time. The more objects, the longer the time needed to perform divergence analysis.

Evaluation of the test results is carried out based on the threshold value of the conflict value. We used a threshold of 0.5 for conflict value. This threshold value means that the model will only handle 50% of conflicting nodes. This is based on the consideration, the more handling nodes that experience conflict, the more effort is needed for conflict handling, but too few nodes can lead to a lack of effectiveness in handling conflicts.

After obtaining a list of nodes experiencing conflict, then conflict resolution is carried out. Conflict resolution is carried out based on the established rules. In this experiment, four conflict resolution scenarios were created. The rules for the four scenarios can be seen in Table VIII. The types of objects (Table 8) are divided into three types of objects, such as goals, tasks, and operations. Each object is assigned a value of 1, 2, and 3 for goals, tasks, and operations. Meanwhile, Conflict values are clustered into three values, such as 1 for objects with small conflict values, 2 for objects with average or moderate conflict values, and 3 for objects with large conflict values. There are three ways to conflict resolution; the first conflict handling is to make the object a soft element assigned by conflict resolution = 1; The second conflict resolution is to make an OR relation on the object experiencing the conflict, assigned by conflict resolution = 2, and the third conflict resolution is letting the conflict persist, assigned by conflict resolution = 3. The four scenarios have different conflict handling characteristics. The first scenario is intended to determine whether the type of object and cluster of conflict does not affect the effectiveness of conflict resolution. The second scenario is intended to measure whether conflict resolution can avoid the formation of soft goals/soft elements. Scenario three is intended to determine the effectiveness of conflict management if only two conflict clusters are determined, and scenario four is intended to measure the effectiveness of conflict management based on cluster groups.

TABLE VI
EXAMPLE OF DIVERGENCE ANALYSIS RESULTS

Case Num.	Object 1	Object 1 Value	Object 2	Object 2 Value
1	G5	True	G6	False
2	G5	False	G6	False
3	G5	True	G6	False
4	G5	False	G6	True
5	G5	True	G6	False
6	G5	False	G6	True

D. Completeness Measure

After handling the conflict, the completeness (Comp.) measurement is carried out on the model. Completeness is a measurement of meeting the requirements of all goals in the model. The value of completeness is measured by [0,1], where 0 means that there is no requirement that can be met by the model, and 1 means that all requirements can be met by the model. Table IX shows the level of completeness that is resulted after conflict resolution on the four tested scenarios.

Based on the data in Table IX, it can be concluded that the way of handling conflict through changing elements into soft elements is able to increase the completeness value to the highest level. Based on the four scenarios carried out, the fourth scenario has the least number of soft goals, namely an average of 52.4 or 47%. This scenario gives the best results from the four scenarios tested for the highest completeness value. However, if soft goals are important, then the second scenario is the best choice, but the completeness value only reaches 0.44. The completeness value of this score will reach the highest value if the conditions of all facts or all elements of tasks and operations can be met or valued true.

IV. CONCLUSION

In this study, a novel method was developed to detect and resolve requirements conflict in AREM. The method was developed using an intelligence agent approach combined with a weighted product for conflict detection and rule-based integrated with K-means clustering for conflict resolution. The input of the model is in the form of requirement data developed using the GORE approach. The data is in the form of a tree of requirements and is represented in the form of rules. Intelligent agents perform conflict detection through agent behaviour that is generated automatically through test cases. Conflict detection is carried out through divergence analysis for each case. This process produces a set of conflicting node pairs based on the specified threshold value.

TABLE VII
NUMBER OF OBJECTS AND TIME ANALYSIS OF EACH TESTED DATA

Project Code	The Number of Objects	Total Number of Pair Node	Time Analysis for 100 cases (in a second)
P1	468	17020	497
P2	169	3741	153
P3	668	30876	577
P4	214	2775	129
P5	1.354	101.025	1.044

TABLE VIII
CONFLICT RESOLUTION RULES

Rule ID	If	Then
Scenario 1		
1	All Type of object and all conflict_cluster	Conflict resolution = 1
Scenario 2		
1	Type of object = 1 and conflict_cluster = 1	Conflict resolution = 3
2	Type of object = 1 and conflict_cluster = 2	Conflict resolution = 2
3	Type of object = 1 and conflict_cluster = 3	Conflict resolution = 2
4	Type of object = 2 or Type of object = 3	Conflict resolution = 1
Scenario 3		
1	Type of object = 1 and conflict_cluster = 1	Conflict resolution = 3
2	Type of object = 1 and conflict_cluster = 2	Conflict resolution = 1
3	Type of object = 1 and conflict_cluster = 3	Conflict resolution = 1
4	Type of object = 2 or Type of object = 3	Conflict resolution = 1
Scenario 4		
1	Type of object = 1 and conflict_cluster = 1	Conflict resolution = 3
2	Type of object = 1 and conflict_cluster = 2	Conflict resolution = 2
3	Type of object = 1 and conflict_cluster = 3	Conflict resolution = 1
4	Type of object = 2 or Type of object = 3	Conflict resolution = 1

TABLE XI
COMPLETENESS MEASURE AFTER CONFLICT RESOLUTION

Project Code	Comp. Baseline	Number of Goals	Completeness value after conflict resolution							
			Scenario 1		Scenario 2		Scenario 3		Scenario 4	
			Comp	Soft Goal	Comp.	Soft Goal	Comp	Soft Goal	Comp.	Soft Goal
P1	0.01	62	1	56	1	0	1	43	1	33
P2	0.01	83	1	71	0.03	0	1	29	1	22
P3	0.01	145	1	129	0.03	0	1	125	1	69
P4	0.01	43	1	35	1	0	1	35	1	25
P5	0,01	218	1	212	0,13	0	1	179	1	113
Average	**0.01**	**110.2**	**1**	**100.6**	**0.44**	**0**	**1**	**82.2**	**1**	**52.4**
Average Percent Soft goal				**90%**		**0**		**75%**		**47%**

The method was tested on a dataset consisting of five software projects. The test results show that the method is able to detect the conflict of requirements in the five datasets by using a threshold value of 0.5. By handling the conflict, the model can increase the completeness value to the highest value. This means that the conflict detection and resolution method has succeeded in identifying a number of nodes that cause conflicts in requirements that have an impact on the quality of the model. The method is able to overcome the problem of inconsistency and incompleteness on AREM.

ACKNOWLEDGEMENT

The authors thank the Republic of Indonesia Ministry of Research and Higher Education through Universitas Gadjah Mada, which has funded this research with contract numbers: 2271/UN1/DITLIT/DIT-LIT/PT/2021. The authors also thank The Faculty of Information Technology Universitas Kristen Duta Wacana for providing facilities and funding for publishing this article.

REFERENCES

[1] I. Sommerville, *Software Engineering Ninth Edition*, Ninth Edit. United States of America: Addison Wesley, 2011.

[2] A. van Lamsweerde, *Requirements Engineering From System Goals to UML Models to Software Specifications*. Wiley, 2009.

[3] M. Aldekhail and D. Ziani, "Intelligent Method for Software Requirement Conflicts Identification and Removal: Proposed Framework and Analysis," *IJCSNS Int. J. Comput. Sci. Netw. Secur.*, vol. 17, no. 12, pp. 91–98, 2017.

[4] M. Aldekhail, A. Chikh, and D. Ziani, "Software Requirements Conflict Identification: Review and Recommendations," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 10, pp. 326–335, 2016, doi: 10.14569/ijacsa.2016.071044.

[5] W. Guo, L. Zhang, and X. Lian, "Automatically detecting the conflicts between software requirements based on finer semantic analysis," *arXiv*, vol. abs/2103.0, 2021, [Online]. Available: http://arxiv.org/abs/2103.02255

[6] A. Van Lamsweerde, R. Darimont, and E. Letier, "Managing conflicts in goal-driven requirements engineering," *IEEE Trans. Softw. Eng.*, vol. 24, no. 11, pp. 908–926, 1998, doi: 10.1109/32.730542.

[7] R. Delima, R. Wardoyo, and K. Mustofa, "Goal-Oriented Requirements Engineering: State of the Art and Research Trend," *JUITA J. Inform.*, vol. 9, no. 1, pp. 105–114, 2021, doi: 10.30595/juita.v9i1.9827.

[8] A. van Lamsweerde and E. Letier, "Integrating obstacles in Goal-Driven Requirements Engineering," in *Proceedings - International Conference on Software Engineering*, 1998, no. April, pp. 53–62.

[9] A. Van Lamsweerde, "Handling obstacles in goal-oriented requirements engineering," *IEEE Trans. Softw. Eng.*, vol. 26, no. 10, pp. 978–1005, 2000, doi: 10.1109/32.879820.

[10] D. Alrajeh, J. Kramer, A. Van Lamsweerde, A. Russo, and S. Uchitel, "Generating obstacle conditions for requirements completeness," in *Proceedings - International Conference on Software Engineering*, 2012, pp. 705–715. doi: 10.1109/ICSE.2012.6227147.

[11] A. Cailliau and A. van Lamsweerde, "Assessing requirements-related risks through probabilistic goals and obstacles," *Requir. Eng.*, vol. 18, no. 2, pp. 129–146, 2013, doi: 10.1007/s00766-013-0168-5.

[12] S. Zardari, R. Bahsoon, and A. Ekárt, "Cloud adoption: Prioritizing obstacles and obstacles resolution tactics using AHP," *Proc. ACM Symp. Appl. Comput.*, pp. 1013–1020, 2014, doi: 10.1145/2554850.2555067.

[13] A. Cailliau and A. Van Lamsweerde, "Runtime Monitoring and Resolution of Probabilistic Obstacles to System Goals," *Proc. - 2017 IEEE/ACM 12th Int. Symp. Softw. Eng. Adapt. Self-Managing Syst. SEAMS 2017*, pp. 1–11, 2017, doi: 10.1109/SEAMS.2017.5.

[14] R. Degiovanni, N. Ricci, D. Alrajehy, P. Castro, and N. Aguirre, "Goal-conflict detection based on temporal satisfiability checking," *ASE 2016 - Proc. 31st*

IEEE/ACM Int. Conf. Autom. Softw. Eng., pp. 507–518, 2016, doi: 10.1145/2970276.2970349.

[15] R. Degiovanni, P. Castro, M. Arroyo, M. Ruiz, N. Aguirre, and M. Frias, "Goal-conflict likelihood assessment based on model counting," *Proc. - Int. Conf. Softw. Eng.*, pp. 1125–1135, 2018, doi: 10.1145/3180155.3180261.

[16] R. Degiovanni, G. Regis, F. Molina, and N. Aguirre, "A genetic algorithm for goal-conflict identification," *ASE 2018 - Proc. 33rd ACM/IEEE Int. Conf. Autom. Softw. Eng.*, pp. 520–531, 2018, doi: 10.1145/3238147.3238220.

[17] M. B. Rehman, H. M. E. I. Dafallaa, N. Ahmad, I. Ahmad, M. Rashid, and R. Khan, "Requirement elicitation: Requirements conflict resolution and communication model for Telecommunication Sector," in *Proceedings of the 2nd International Conference on ICT for Digital, Smart, and Sustainable Development, ICIDSSD 2020*, 2021, pp. 105–122. doi: 10.4108/eai.27-2-2020.2303293.

[18] Q. Khan, M. A. Khan, Q. Javaid, I. Ullah, K. Ullah, and M. Fawad, "A Rule Based Genetic Algorithm Technique for Conflicts Resolution in Requirements Engineering," vol. 13, no. 11, pp. 8427–8433, 2016, doi: 10.1166/jctn.2016.5993.

[19] S. Russell and P. Norvig, *Artificial Intelligence A Modern Approach Fourth Edition*, Fourth. Pearson Series in Artificial Intelligence, 2020.

[20] A. Kolios, V. Mytilinou, E. Lozano-Minguez, and K. Salonitis, "A comparative study of multiple-criteria decision-making methods under stochastic inputs," *Energies*, vol. 9, no. 7, pp. 1–21, 2016, doi: 10.3390/en9070566.

[21] R. Cahyanto, A. R. Chrismanto, and D. Sebastian, "Pengelompokan Komentar Dataset Sentipol dengan Modified K-Means Clustering," *J. Tek. Inform. dan Sist. Inf.*, vol. 6, no. 3, pp. 531–540, 2020, doi: 10.28932/jutisi.v6i3.3006.

[22] P. S. Mann, "Introductory Statistics, 8th Ed," *John Wiley Sons, Inc.*, vol. 8, no. 11, p. 736, 2012.