KLIK: Kajian Ilmiah Informatika dan Komputer
ISSN 2723-3898 (Media Online)
Vol 3, No 6, Juni 2023, Hal 893-900
DOI 10.30865/klik.v3i6.912
https://djournals.com/klik

Dialectical Assurance Case using Structured Assurance Case Metamodel Notation

Nungki Selviandro*, Ryan Adeputra Sutopo, Gia Septiana Wulandari

Faculty of Informatics, School of Computing, Telkom University, Bandung, Indonesia
Email: [1]nselviandro@telkomuniversity.ac.id, [2]ryanade71@gmail.com, [3]giaseptiana@telkomuniversity.ac.id
Correspondence Author Email: nselviandro@telkomuniversity.ac.id

Abstract−An assurance case is a document created to enable the exchange of information related to system safety and security among various system stakeholders, including suppliers, acquirers, operators, and regulators. It is essential to review such assurance cases to ensure their validity. The Structured Assurance Case Metamodel (SACM) is one of the languages that can be used to communicate assurance cases, and it offers various variants and expressive notations to facilitate this task. While creating assurance cases, it is necessary to consider dialectical notation with CounterArguments and CounterEvidence to explain situations where an assurance case has an incorrect value or can be countered. SACM classifies this notation as dialectical, and it is useful for reviewing and developing assurance cases. However, despite the usefulness of this approach, there is limited research on its effectiveness. Therefore, this study aims to explain dialectical notation in the SACM assurance case language and develop existing SACM applications. The application was developed to provide users who are already familiar with assurance cases, especially SACM, with a practical way of introducing dialectical notation. By doing this, we hope to expand the use of SACM and improve its effectiveness in communicating assurance cases among system stakeholders.

Keywords: Assurance Case; Structured Assurance Case Metamodel; Dialectical Approach

1. INTRODUCTION

An assurance case is a vital document that plays a critical role in ensuring system safety and security by facilitating the exchange of information between various system stakeholders. These stakeholders include suppliers, acquirers, operators, regulators, and other interested parties. The Structured Assurance Case Metamodel (SACM) is a language used to communicate assurance cases. SACM has various notations, including the Structured Assurance Case Metamodel Notation (SACMN), which is a versatile and expressive notation used to communicate different types of systems. SACMN notation has several types designed to meet the needs of communicating various systems. For example, core notation is used to communicate the most critical aspects of a system, while modular notation is used to communicate interfaces and dependencies between different system components. Pattern notation is used to capture recurring patterns in system design, while dialectical notation is used to review assurance cases. The dialectical SACMN notation is an essential tool used to review assurance cases to ensure that the claims made in the assurance case are valid and that there are no counters in the assurance case system. The dialectical notation consists of counter-inference and counter-evidence, which are used to carry out the assurance case review process. However, despite its importance, there is low usage of assurance case reviews that use the SACMN modelling language due to a lack of research explanations of the concept of dialectical assurance cases.

The creation of a dialectical notation for the assurance case language, Structured Assurance Case Metamodel, is a necessary requirement. This notation draws upon the assurance case language Goal Structuring Notation (GSN) and the journal Goal Structuring Notation Community Standard Version 3 [17]. The dialectical notation is designed to provide a more structured and detailed framework for composing and communicating arguments related to the security, reliability, and feasibility aspects of a system within the context of an assurance case. With this notation, practitioners and stakeholders can better understand and analyze the arguments within an assurance case, as well as trace relationships between claims, evidence, and assumptions. The use of dialectical notation helps to depict arguments in a structured manner, using elements such as claims, subclaims, evidence, assumptions, and reasoning that connects them. This notation also makes it easier to detect and rectify errors or deficiencies in reasoning, thereby enhancing the reliability and quality of the assurance case. By referencing the journal [13], the development of dialectical notation can adopt and expand upon the principles and conventions established in previous standard versions. Consequently, creating dialectical notation for the assurance case language Structured Assurance Case Metamodel will significantly improve the representation and analysis of assurance cases, as well as enhance understanding and confidence in the existing arguments.

The Structured Assurance Case Metamodel requires a dialectical notation due to its lack of detail refers to the journal [8]. A journal explains the challenges in representing complex assurance arguments and how the current notation fails to capture intricate relationships. This leads to incomprehensibility and inadequate analysis. A dialectical notation tailored to the Metamodel is proposed to enhance precision, clarity, and coherence. It facilitates systematic analysis and identification of gaps, weaknesses, and inconsistencies. The creation of dialectical notation becomes imperative based on insights from the journal's analysis and implementation of a web-based graphic editor. This opportunity to improve the overall quality and effectiveness of assurance case development and evaluation processes is crucial.

Therefore, this study aims to introduce the dialectic concept using SACMN and develop tool support for constructing assurance cases, especially the dialectic feature. After the introduction, the study will analyze the perception of the assurance case users on the dialectic concept using SACMN. This study is crucial in promoting the usage of SACMN notation in assurance cases and ensuring that the claims made in assurance cases are valid and defensible.

2. RESEARCH METHODOLOGY

2.1 Research Related to Dialectic Assurance Case (GSN Dialectic)

Goal Structuring Notation (GSN) is a graphical argument notation that can be used to document explicitly the elements and structure of an argument and the argument's relationship to evidence [9]. The notations contained in the GSN have various types, divided into core, arguments, modular, confidence, and dialectic. This dialectic notation is a process in its simplest form for providing the truth. A dialectic notation will be used as a dialectic process to provide a framework that questions, tests, and makes the assurance case free from the identification of doubts and errors that make it ambiguous or do not match the needs of the system function in the assurance case being created. The notation used for the dialectic process in the GSN modelling language is as follows.

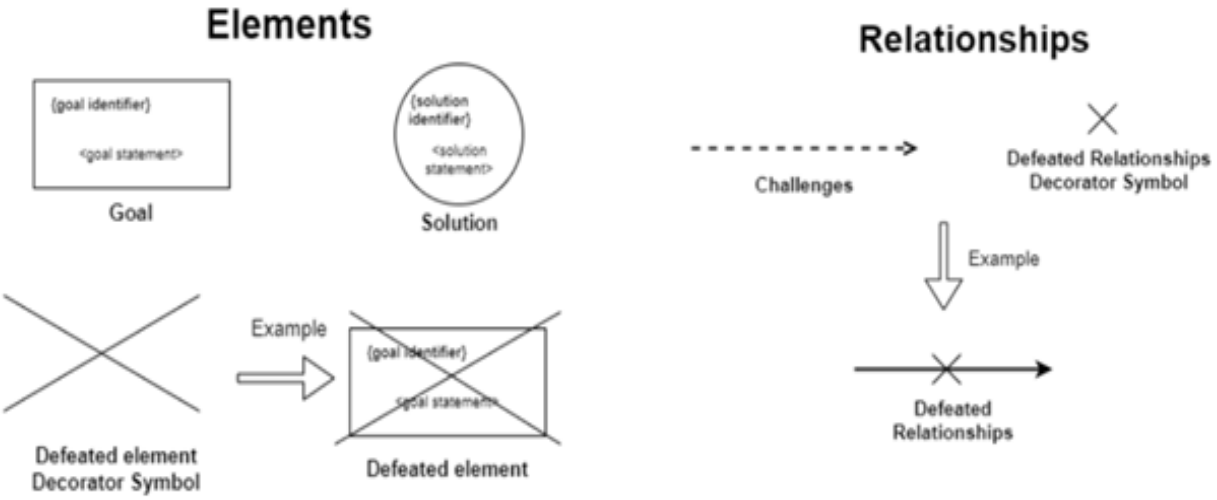

Figure 1. Various notation from GSN dialectic notation

From **Figure 1.**, it can be explained that the various notations that belong to the dialectic process of the GSN modelling language are as follows,

a. Goal element, a core element of GSN to be used in a dialectic context to assert a challenge to part of the argument.
b. Solution element, a core element of GSN to be used to present a reference to evidence that asserts a challenge to part of the argument.
c. Defeated element, is a symbol by giving a cross ('**X**') on the element in the GSN indicates that the element has been defeated.
d. Challenges relationship, represented by a dashed line with an open arrowhead, used as a challenge to any GSN entity.
e. Defeated relationship, is a symbol by giving a cross ('**X**') in the middle of the GSN relationship indicates that the relationship has been defeated.

After discovering the various notations used for the dialectic process in the GSN language, the use of this notation can be used for any goal structure that has been made in the assurance case. After the elements or relationships have been indicated as defeated, a counter will be made to the notation. Here are the elements used to counter the defeated goal or solution argument.

Figure 2. Supported elements for GSN dialectic notation

Counter Evidence will use a form of notation like a solution and Evidenced Counter Argument will use a form of notation like a solution. These two notations can support a process to challenge any element that indicates defeat. The following is an example of an assurance case that has a dialectic process in it.

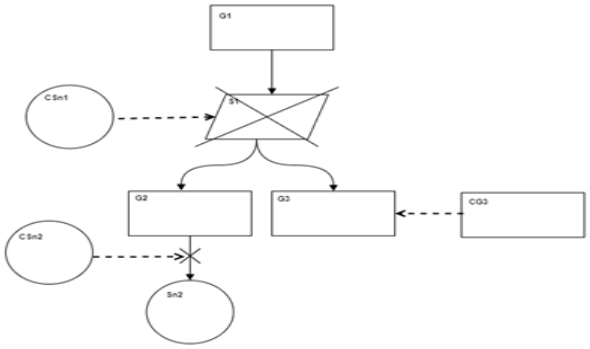

Figure 3. Example of using the GSN dialectic notation

From **Figure 3.**, it can be explained that G1 is a goal that has S1 as a solution that has been indicated defeated. Because S1 is indicated to have been defeated, counter evidence is needed for S1 by making CSn1 as counter evidence. From S1, we have 2 goals, namely G2 and G3, with G3 being challenged by CG3 to question the validity of G3 and need more justification to prove G3 is valid. G2 has a relationship that is indicated to be defeated to Sn2, therefore G2 has counter evidence (via solution) with index CSn2.

2.2 Dialetical assurance case using Structured Assurance Case Metamodel Notation (SACMN)

SACMN is one of the three languages available for modelling assurance cases. SACMN provides the specification and abstract syntax for the development and representation of assurance cases [3]. SACMN is a specification that defines a metamodel for representing structured assurance cases. This assurance case modelling language was published by the Object Management Group (OMG) with the aim of increasing the standardization and validity of various notations for assurance cases. The notation on SACMN is divided into nodes and relations by dividing into sub-types of notations such as core, modular, pattern, and dialectic. This type of notation has a different function and is used according to needs when creating assurance cases with the SACMN modelling language. An explanation of the types of SACMN notation is as follows.

a. Core notations, the notations used as the basic notation in SACMN which explains fundamentally a system with Claim nodes as contextual or evidential information and with ArtifactReference nodes as a reference for supporting evidence of the Claim. A claim and an ArtifactReference also require further supporting arguments to make the claim more valid.
b. Modular notations, the notations used for grouping various notation elements, and can be used as reference from other notation modules as element justification. This modular notation is interconnected with one another as a reference and the main module described in an assurance case.
c. Pattern notations, reusable notation such as 'Templates', with the aim of helping to visually optimize the information level when iterative argument structures are required.
d. Dialectic notations, the notations used for further review of the validity of the assurance case that has been made. Starting from the validity of the claim that has been made, is it valid and there is no counter that can be done. If the claim has a counter or is invalid, the Claim will change to a DefeatedClaim node and a CounterArgument node will be made as a further argument regarding the counter claim, and by using the CounterEvidence relation that connects the two nodes.

For further information about the types of notations, you can read the journal [3]. Related to the purpose of this study, the discussion on dialectical notation is continued. The dialectical notation has various notations which will be listed below.

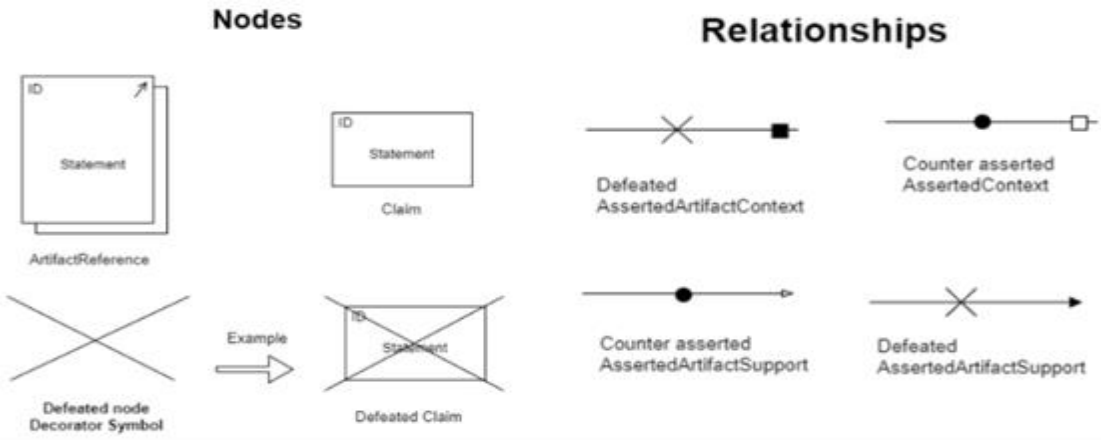

Figure 4. Various notation from SACM dialectic notation

From **Figure 4**, various SACMN dialectical notations can be explained as follows,

a. ArtifactReference node, an argument used as evidence reference or contextual information. In dialectical notation, this node is reused as a CounterArgument which will be used as a reference for an invalid or defeat assurance case arrangement.
b. Claim node, a structured argument that contained a true or false statement.
c. Defeated node decorator symbol, a sign that uses the symbol ('X') and symbolizes that the node has an invalid value or description and is worth defeat.
d. Defeated Claim node, an argument that has been made in the Claim has an argument that is invalid or has an incorrect value. This node requires a CounterAgument/CounterEvidence as a reference for the justification of this Claim.
e. Defeated AssertedArtifactContext relation, relations indicates that the inference is defeated by CounterEvidence. This relation can be used between Claim as the start and ArtifactReference as the destination.
f. Counter asserted AssertedContext, relations indicate that the inference counters its declared purposes.
g. Counter asserted AssertedArtifactSupport, relations indicate that the inference counters its declared purposes, this relation can be used between ArtifactReference as the start and Claim as the destination.
h. Defeated AssertedArtifactSupport, relations indicate that the inference is defeated by CounterEvidence.

The SACMN notation listed on this type of dialectical is used to review the assurance cases that have been made, with the aim of re-checking their validity. The following is an example of using notation in the assurance case.

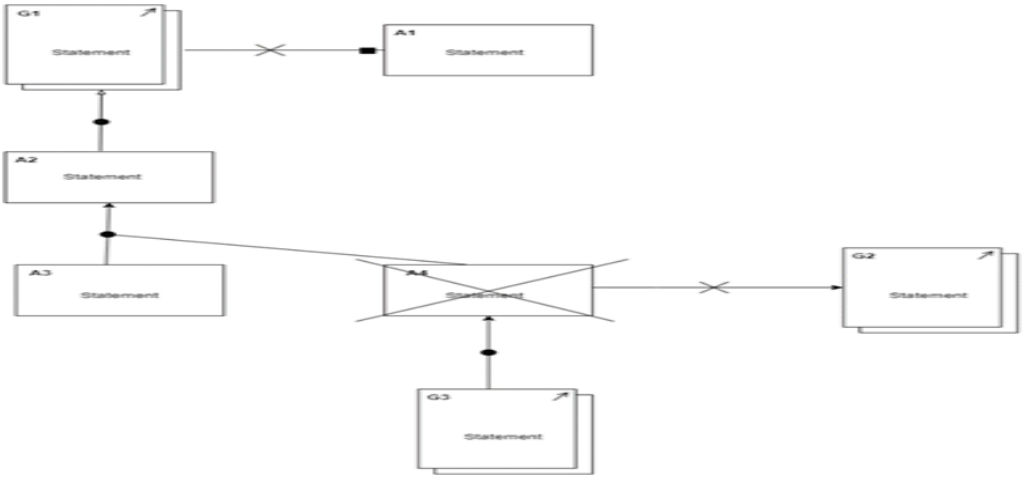

Figure 5. Example of using the SACM dialectic notation

From **Figure 5.**, it can be explained that ArtifactInference G1 was defeated by a counterevidence described as claim A1. ArtifactInference G2 has a CounterArgument that can be used in Claim A2. Claim A2 has branches of Claim A3 and Claim A4 as support for the CounterArgument that has been given. Claim A4 has a reference that supports A4, namely ArtifactReference G3, but after reviewing it, Claim A4 is declared to have an incorrect and inappropriate validation value, therefore Claim A4 becomes a DefeatedClaim, and is assisted by a CounterArgument from ArtifactReference G2.

2.3 Tool Support Development Process

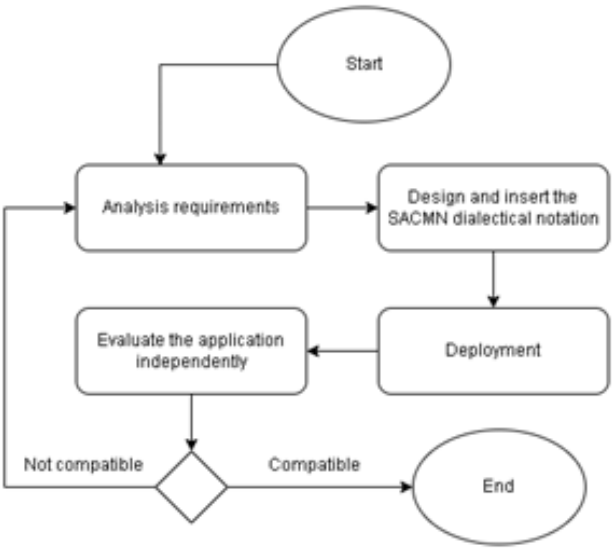

Figure 6. Application development flow

The introduction of dialectical notation from SACMN requires an application that is used as an introduction medium. Existing applications already have the required core notation and require further development. Further information can be seen in the journal [4]. The flow of the application development is shown in the following figure.

Figure describes the flow of application development from applications that are already available to the dialectical notation feature successfully used in the application. The following is an explanation of each flow listed in the figure.

a. **Analysis requirements** – the first step in application development is to analyze the requirements that are already available in the application [4], then adjust to the latest requirements. The new requirements needed are the new notation and the appropriate relations in the SACMN dialectical notation. Notation size adjustment to make it easier to use and does not require re-measurement of the notation.
b. **Design and insert the SACMN dialectical notation** – Designing the notation that has been described in the figure , adjusting the size of the notation to be used in the application, so that the appearance of the notation in the application before use and after use remains in accordance with the certainty of the notation.
c. **Deployment** – at this step, coding will be carried out on the existing application so that the notation that has been successfully inserted into the application can be used and can be related to other notations.
d. **Evaluate the application independently** – after the application has been successfully made in accordance with all the previous steps, testing will be carried out independently to find out the deficiencies that exist. The deficiencies will be revised as soon as possible according to the required application requirements.

2.4 Stages of Application Testing

After the application development is carried out and it is in accordance with the application requirements. In the next stage, testing will be carried out on users who are familiar with SACMN, more precisely users who have tested the development of previous tools [4]. Tests will be carried out on 26 people who are related to informatics majors and are familiar with SACMN. The following are the steps of scenario testing for testing in this research.

Table 1. Testing scenario

No	Test Scenario Step
1	Explanation of the SACMN dialectical notation to the user
2	A question-and-answer session with users regarding the purpose and essence of this research
3	The user will be asked to re-create the assurance case which already has the SACMN dialectical notation
4	The user will be given a questionnaire using the google form platform

After the user does the testing, by testing the components of the Method Evaluation Model (MEM) which have aspects of perceived usefulness that test the perception by the user for the usefulness of this dialectical notation. After that, the perceived ease of use is also tested to know the perception of the user of the ease of use after using the application of this dialectical notation. The last aspect tested by MEM is the intention to use, this aspect aims to find out the feelings of the user's intention to use the application more often or not.

Questions on the questionnaire will relate to aspects of MEM which will be used as a further assessment. The assessment method will be used to assess whether the results of the questionnaire are valid or not using Cronbach's alpha method. After Cronbach's alpha results are obtained and classified as good results or above 0.7 points, the results of the questionnaire will be assessed by finding the mean value of each MEM aspect to get the main result value from this research

3. RESULT AND DISCUSSION

After testing the user and the user has answered the questions on the questionnaire, the results obtained will be assessed using Cronbach's alpha method. The following is a rating table for Cronbach's alpha method to determine the validity of the questionnaire results obtained.

Table 2. Rating of cronbach's alpha

Cronbach's Alpha	Internal consistency
$0.9 \leq \alpha$	Excellent
$0.8 \leq \alpha < 0.9$	Good
$0.7 \leq \alpha < 0.8$	Acceptable
$0.6 \leq \alpha < 0.7$	Questionable
$0.5 \leq \alpha < 0.6$	poor
$\alpha < 0.5$	Unacceptable

This rating table is used as a reference for evaluating this method. The questionnaire given to the user contained 4 questions about perceived usefulness, 3 questions about perceived ease of use, and 3 questions about the intention to use. The following is Cronbach's alpha formula used to calculate the results of this questionnaire.

$$\partial = \left[\frac{k}{(k-1)}\right]\left[1 - \frac{\sum \sigma^2 b}{\sigma^2 t}\right] \quad (1)$$

Description:

∂ = Cronbach's Alpha coefficient
k = Number of question items
$\sum \sigma^2 b$ = Sum of the item variances
$\sigma^2 t$ = Variances of Total Score

This formula is used to calculate all questions from the questionnaire by combining all the questions into one table. After becoming one table, it is calculated by determining the number of questions, the number of question variances, and the variance of the total number of questions. The following shows the results of the recap of the questionnaire conducted in this research.

Table 3. The results of the questionnaire

No	Perceived Usefulness				Perceived Ease of Use			Intention to Use		
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1	1	2	1	1	2	2	1	2	1	2
2	1	2	2	1	2	1	2	1	2	1
3	1	3	2	2	2	1	2	2	2	3
4	1	1	2	2	2	2	2	3	3	2
5	2	2	2	2	2	1	2	2	2	3
6	2	3	2	4	2	2	3	2	3	2
7	2	1	3	2	2	2	2	3	3	1
8	1	2	1	2	2	1	2	2	1	2
9	3	1	3	3	2	2	3	4	2	3
10	2	2	2	1	1	2	1	3	1	2
11	1	1	2	1	1	2	1	2	2	1
12	1	1	1	2	1	1	2	3	2	3
13	2	2	2	1	1	2	2	1	1	1
14	2	1	2	3	2	2	1	2	2	2
15	1	1	2	1	1	2	1	2	1	2
16	2	2	2	2	2	3	3	1	2	3
17	2	1	3	1	1	2	2	2	3	2
18	1	2	1	1	2	2	1	1	2	1
19	2	2	2	1	2	2	1	1	1	2
20	2	3	2	3	3	3	2	2	2	2
21	1	1	2	3	1	2	1	1	2	1
22	2	2	2	1	3	1	2	2	2	2
23	2	1	2	2	1	2	1	1	3	2
24	1	2	2	1	3	2	2	2	2	2
25	1	2	2	2	1	3	1	3	3	1
26	2	3	2	2	3	2	2	2	3	2

After sorting up the results from the questionnaire, we will calculate each component in Cronbach's alpha. The following is a description of the calculation of each component,

a. Number of question items: 10
b. Sum of the item variances: 4,269230769
c. Variances of Total Score: 12,18

After calculating each component, Cronbach's alpha from this questionnaire got 0.721653825 points and it can be concluded that this questionnaire is classified as "acceptable" and the results can be used as a reference for calculating the final results of the research. After getting the values from Cronbach's alpha, we will calculate the mean of each aspect in MEM, the following shows the results of the analysis of each aspect in MEM with the required variables.

Table 4. Result of each variable in the questionnaire

Components	Perceived Usefulness				Perceived Ease of Use			Intention to Use		
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q 10
Modus	2	2	2	1	2	2	2	2	2	2
Min	3	3	3	4	3	3	3	4	3	3
Max	1	1	1	1	1	1	1	1	1	1
Mean	1,77				1,80			1,98		

Median		2				2			2	
Standard Deviation	0,57	0,71	0,52	0,84	0,69	0,58	0,66	0,8	0,72	0,68

From the table, several variables are calculated for all aspects in MEM, e.g., mean, and median. These two examples of variables are used as a rating for the results of the questionnaire. The results obtained from aspects of perceived usefulness get 1.77 points, perceived ease of use get 1.80 points, and intention to use get 1,98 points. The mean results obtained from each aspect refer to the choices in the questions in the questionnaire that use a linear scale which symbolizes 1 as "Very appropriate" and 5 as "very not appropriate". So, it can be concluded that the results of each aspect belong to the values 1 and 2 which symbolize "very appropriate" and "appropriate". The mean result of each aspect will be rounded down and get a value of 2 from each aspect and it can be classified that each aspect in MEM represents "appropriate".

4. CONCLUSION

From this research, the application that has been developed to be able to use the SACMN dialectical notation has been completed and tested on several users who are familiar with SACMN. After the user uses the SACMN application, we will provide a questionnaire to the users with the aim of testing 3 perception aspects of MEM. The results obtained from this questionnaire will be processed and checked for validity using Cronbach's alpha method. By giving a total of 10 questions on the questionnaire, the results of Cronbach's alpha get 0.721653825 points and can be classified into "acceptable" results. The results of Cronbach's alpha calculation will be followed by finding the mean value of each aspect of the MEM tested in the questionnaire. The mean value obtained in the three aspects gets a value of 2 after rounding down the value, and can be categorized as "appropriate" according to the value of the questions given in the questionnaire (linear scale rating). From this research, it is hoped that there will be further development of existing applications to make them easier to use. Wider questionnaire data collection to make it easier and introduce to more people about assurance case, more precisely SACMN.

REFERENCES

[1] I. Habli, R. Alexander, and R. Hawkins, "Safety Cases: An Impending Crisis?," Jun. 2021.

[2] O. Jaradat, I. Šljivo, I. Habli, and R. Hawkins, "Challenges of Safety Assurance for Industry 4.0," Jun. 2017, pp. 103–106. doi: 10.1109/EDCC.2017.21.

[3] C.-L. Lin, W. Shen, and R. Hawkins, "Support for safety case generation via model transformation," ACM SIGBED Review, vol. 14, pp. 44–52, Jun. 2017, doi: 10.1145/3076125.3076130.

[4] R. Mokhtar, S. Othman, and R. Ramlan, "Modelling Quality Assurance System Process Using UML Notation," Ingénierie des systèmes d information, vol. 27, pp. 705–716, Jun. 2022, doi: 10.18280/isi.270503.

[5] M. Chelouati, A. Boussif, J. Beugin, and E.-K. El-Miloudi, "Graphical safety assurance case using Goal Structuring Notation (GSN)– challenges, opportunities and a framework for autonomous trains," Reliab Eng Syst Saf, vol. 230, p. 108933, Jun. 2022, doi: 10.1016/j.ress.2022.108933.

[6] A. Deeb, T. Garner, and D. Kuchekar, "Building the Assurance Case for Recovery of Autonomous Underwater Vehicles from Offshore Platforms," Jun. 2023. doi: 10.4043/32185-MS.

[7] N. Fung, S. Kokaly, A. Sandro, and M. Chechik, "Assurance Case Property Checking with MMINT-A and OCL," 2022, pp. 351–360. doi: 10.1007/978-3-030-82083-1_30.

[8] N. Selviandro, "Assurance Case Pattern using SACM Notation," Jun. 2021, pp. 494–499. doi: 10.1109/ICoICT52021.2021.9527483.

[9] S. Ramakrishna, H. Jin, A. Dubey, and A. Ramamurthy, "Automating Pattern Selection for Assurance Case Development for Cyber-Physical Systems," 2022, pp. 82–96. doi: 10.1007/978-3-031-14835-4_6.

[10] C. Weir, A. Rashid, and J. Noble, "Challenging software developers: Dialectic as a foundation for security assurance techniques," J Cybersecur, vol. 6, no. 1, 2021, doi: 10.1093/CYBSEC/TYAA007.

[11] F. U. Muram and M. A. Javed, "ATTEST: Automating the review and update of assurance case arguments," Journal of Systems Architecture, vol. 134, Jan. 2023, doi: 10.1016/j.sysarc.2022.102781.

[12] "Garantia de qualidade do ensino superior: o caso dos Estados Unidos," Estudos em Avaliação Educacional, Sep. 2022, doi: 10.18222/eae.v33.9022_port.

[13] A. Gambarotto, "Teleology and mechanism: a dialectical approach," Synthese, vol. 201, no. 5, May 2023, doi: 10.1007/s11229-023-04137-y.

[14] P. J. Graydon and C. M. Holloway, "An investigation of proposed techniques for quantifying confidence in assurance arguments," Saf Sci, vol. 92, pp. 53–65, 2017, doi: 10.1016/j.ssci.2016.09.014.

[15] P. O. Antonino, M. Trapp, and P. Barbosa, Computer Safety, Reliability, and Security (incl WAISE), vol. 9337, no. August 2016. 2018. [Online]. Available: http://link.springer.com/10.1007/978-3-319-24255-2

[16] A. Gacek, J. Backes, D. Cofer, K. Slind, R. Collins, and M. Whalen, "Resolute: An assurance case language for architecture models," HILT 2014 - Proceedings of the ACM Conference on High Integrity Language Technology, pp. 19–27, 2014, doi: 10.1145/2663171.2663177.

[17] The Assurance Case Working Group, "Goal Structuring Notation Community Standard Version 3," pp. 1–130, 2021, [Online]. Available: https://scsc.uk/r141C:1?t=1

[18] R. A. Sutopo, N. Selviandro, and G. S. Wulandari, "Analysis and Implementation of Web-based Graphic Editor for Structured Assurance Case Metamodel Notation".

[19] N. Selviandro, "Assurance Case Pattern using SACM Notation," 2021 9th International Conference on Information and

Communication Technology, ICoICT 2021, pp. 494–499, 2021, doi: 10.1109/ICoICT52021.2021.9527483.
[20] R. Wei, T. P. Kelly, X. Dai, S. Zhao, and R. Hawkins, "Model based system assurance using the structured assurance case metamodel," Journal of Systems and Software, vol. 154, pp. 211–233, 2019, doi: 10.1016/j.jss.2019.05.013.
[21] A. L. Springer, "White Rose Research Online URL for this paper : Version : Accepted Version Ibrahim orcid . org / 0000-0003-2736-8238 (2020) A Visual Notation for the Representation of Assurance Cases using SACM . In : International Symposium on Model-Based Safety and A," 2020.