JEPIN
(Jurnal Edukasi dan Penelitian Informatika)
ISSN(e): 2548-9364 / ISSN(p) : 2460-0741

Vol. 8
No. 3
Desember
2022

GoEliTool for Software Requirements Elicitation using Goal-Oriented Approach

Rosa Delima[#1], Joko Purwadi[#2]

[#]*Program Studi Informatikaen, Universitas Kristen Duta Wacana*
Jl. Dr. Wahidin Sudirohusodo No. 5 – 25, Yogyakarta, Indonesia.
[1]rosadelimaplg@staff.ukdw.ac.id
[2]jokop@staff.ukdw.ac.id

***Abstract*— Requirements engineering (RE) is an essential initial stage in software engineering. The RE process begins with the elicitation stage. This stage collects all user requirements that must be fulfilled by the system which will be developed. A goal-oriented approach is an effective approach used to automate the RE process. The development of goal-oriented input document standards is one of the important issues that has not been widely studied. Therefore, this study developed a goal-oriented input document standard for the requirements elicitation process. A tool is developed based on the form of the input document that has been generated. The development of standard forms of input documents begins with literature study and data collection, analysis, design of standard forms of documents, tool design, tool development, and testing. At the analysis stage, a logical framework and element structure is formulated in a goal-oriented approach. Furthermore, the standard form of input documents is developed. The standard form of the document becomes a guideline for developing tools to process data requirements from elicitation results. Tool testing is carried out using black-box testing. The test results show that the tool can work according to the planned function. The trial of the use of the tools was carried out using five requirements datasets. The results of testing and using the tool through the requirements dataset show that GoEliTools can be used to record data on the requirements of several users for the development of an information system.**

Keywords*— *Requirements Elicitation, AREM, Goal-oriented, black-box testing, Software Engineering

I. INTRODUCTION

Requirements engineering (RE) is an important initial stage in software engineering. Requirements engineering includes several jobs starting with requirements elicitation/requirements collection from system users, analysis, specification, validation, and requirements management [1][2]. In the previous research, a model was developed to automate the requirements engineering process called AREM (Automatic Requirements Engineering Model). AREM can automate three stages of the RE process, such as analysis, specification, and validation of requirements [3]. For the elicitation stage, AREM can handle the process semi-automatically, where the input process of the requirements data still has to be done by users, such as a systems analyst.

Elicitation is the stage for requirement collection on the RE. This stage will receive input from stakeholders/users. This study uses a goal-oriented approach to elicit users' requirements. The approach focuses on stakeholders' goals for the Software Engineering process. Focus on goals supports a participatory approach that can make it easier for stakeholders to define their needs.

GORE (Goal-oriented requirements engineering) has been a widely researched approach in the last three decades. The application of GORE has been carried out at various stages in RE, starting from the elicitation, analysis, specification, and validation stages of requirements. The GORE approach was developed by Robinson [4], who published his research in 1989. In his research, Robinson used a goal approach to integrate specifications and handle conflicting requirements from multiple users [4]. The term GORE was introduced by Axel Van Lamsweerde [5].

A lot of GORE research has been done over the last 30 years. The research includes the development of a goal-oriented modeling language, developing models on GORE, developing functions and methods, and developing a framework for GORE [6][7].

Based on the literature study conducted, it is known that GORE is an effective method used for the RE process [6], but there are still several issues that must be resolved to be able to automate the entire RE process. One issue is the development of standard input documents for a goal-oriented approach. Based on a filtered literature study, two templates were developed by previous researchers in the requirements engineering process, namely [8] and [9].

In their research, Sarmiento et al. developed a restricted form of Natural Language (RNL) to generate test scenarios automatically. In developing the form, Sarmiento et al. did not specifically use the goal model approach; the restricted form they produced had similarities with several goal-oriented elements such as goals, resources, and actors [8].

Meanwhile, research [9] developed a template for requirements analysis on Data Mart. The template has two parts. The first part is document meta-data consisting of title, summary, actor, business process, update date, author,

actor, and process. Meta-data is the part of the document that contains information about the defined data source. The second part defines goals for requirements analysis, where an indicator is determined for each goal, consisting of a label, formula, and target value [9].

The two studies above do not specifically use the Goal-Oriented approach, and based on the results of the literature review, no input document templates have been found that have been specifically prepared using the GORE approach. Therefore, in this study, a form of input document was developed for goal-oriented data requirements. Based on the input document, the tool is designed for requirement data input and supports the elicitation stage of the RE. The tool is named GoEliTools (Goal-Oriented Elicitation Tools). GoEliTools supports the requirement elicitation process and prepares the requirements data to be analyzed automatically by AREM.

This paper is organized into four parts, starting with the introduction, then research methods in part II, and results and discussion in part III. The article ends with conclusions.

II. Research Methods

This research method consists of five stages: literature review and data collection, requirement analysis, design, coding, testing, and tool application. The stages of the research can be seen in Figure 1.

Figure 1. Stages in the research method

A. Literature Review and Data Collection

A literature review and data collection were conducted to obtain a rationale for the goal-oriented approach to RE. One of the goal-oriented approaches for the RE process is GORE (Goal-Oriented Requirements Engineering). GORE is a model for the requirements engineering process that focuses on using goals for elicitation, elaboration, refinement, specification, analysis, negotiation, documentation, and modification of requirements [5].

Goals are prescriptive statements expected to be fulfilled in the system to be developed through the cooperation of agents (e.g., humans, devices, and software) in a specified domain [10]. Goals are formulated with reference to the components that must be developed in the system. Goal formulation can be carried out at several levels of abstraction, from the highest level, a strategic issue, to the lowest level, related to technical issues. Goal formulation is carried out for functional purposes and non-functional purposes. Functional purposes are related to services that are met by the system. Meanwhile, non-functional purposes are related to service quality, including safety, security, accuracy, and performance [11].

Applying the GORE approach provides many advantages in the analysis, specification, and validation processes. This approach simplifies the process of defining requirements by users [12]. The Goal Oriented Modeling (GOM) approach can be used to support the definition of functional and non-functional requirements in the system [13], [14]. The research in this area in the last three decades can be seen in research [6].

B. Requirements Analysis

A Goal-oriented approach is an approach with a goal as the main focus. An organization prepares the goal to be a reference for achievement targets. The goal can be achieved through the outputs obtained from business activities in an organization. All work/activities can be done with the availability of resources in the organization. Based on this thought, a logical framework for the GORE approach was drawn up, as shown in Figure 2.

Figure 2. A logical framework of a goal-oriented approach

The arrangement of standard forms of input documents in GORE is derived from the logical framework that has been compiled. In Figure 2, it can be seen that the goal is the top element in the model, while the results/outputs become features of the system. The goal and features become goal elements. In the GORE approach, goals are divided into hard goals and soft goals. Hard goals are goals whose fulfillment can be done through verification techniques. In contrast, soft goals are goals whose fulfillment is carried out specifically depending on the goal's conditions and level of importance.[5]. A hard goal is a goal from a stakeholder that must be fulfilled in the system development [15]. The goal is achieved through a set of activities or tasks in the GORE approach. Each activity has a set of procedures to do the activities. Activities and procedures need resources for their implementation. The main resource in the model is the implementing actors. Actors consist of two types, namely human actors and agents. The agent actor has behavior that supports the operation or implementation of the procedure. Elements and relationships between elements in the model can be seen in Figure 3.

Figure 3. Elements and relationships between elements in the model.

The development of standard input documents is based on the elements and relationships between elements in the developed model. The document consists of four parts, namely 1) the identity of the stakeholders and the project; 2) the identification of system goals and features; 3) the identification of activities/tasks to support goals, and 4) the identification of operational procedures to achieve goals [16], [17]. The structure of the requirements document can be seen in Table I.

C. Tool Design

The tool design is defined as system architecture, data entry flow diagrams, use case diagrams, and database designs. GoEliTools consists of two main parts: the requirements data entry feature and requirements data extraction. The system architecture can be seen in Figure 4, where the tool will receive input in the form of data requirements from stakeholders, and data entry and storage will be carried out into the requirements database. Then, the object extraction process and the extraction of relations between objects on the data requirements will be carried out. The tool will provide output from data requirements that have been structured according to the elements of a goal-oriented approach.

TABLE I
THE STRUCTURE OF INPUT DOCUMENTS.

User Requirements Input Document	
Part 1: *Stakeholder* and Project Identity	Part 3: Activity Identification
1.1. Stakeholder Identity Stakeholder ID Stakeholder Name Stakeholder Role Data entry date 1.2. Project Identity Project ID Project Name Project Description	3.1. Activity Identification Activities Fulfilled Goal/Features 3.2. Activity Structure Activities Sub-activity Sub-sub-activity Necessary Resources
Part 2: Goal Identification and Features	Part 4: Operational Procedures Identification
1.1. Goal identification *Goal* *Sub-goal* *Sub-sub-goal* *Goal Type* 1.2. System Feature Identification System Features *Fulfilled Sub-goal*	4.1. Operational Procedures Identification Procedure Name Supported Activity Necessary Resources 4.2. Procedure Structure Procedure Name Detailed Procedure Pre-condition Post-condition
User Requirements Input Document	
Goal Type on the Feature	Formula Resources Used

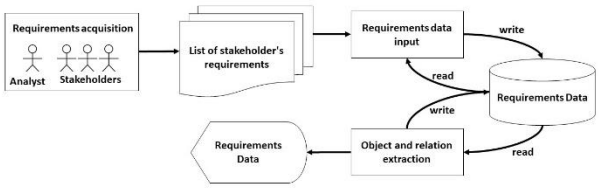


Figure 4. System architecture.

The requirement data entry feature consists of seven parts: project data entry, stakeholder data, goals or features, activities, activity resources, activity procedures, and detailed procedures. The flow chart for the required data entry feature can be seen in Figure 5, where the tool begins with logging into the system, followed by entering the required data. Suppose the required data is taken from a new project. In that case, the data entry process can be carried out sequentially, starting from project data to entering detailed procedure data for operational activities on the system to be developed. If the project requires data already in the system, the user can make changes to the data.

The second design is in the form of a Use Case diagram. This diagram shows the interaction between the user and the system. The requirements data entry tool has two types of users: analysts or analyst assistants and system admins. Analysts can perform various required data entry activities while the system admin is responsible for user management. The use case diagram of the system can be seen in Figure 6.

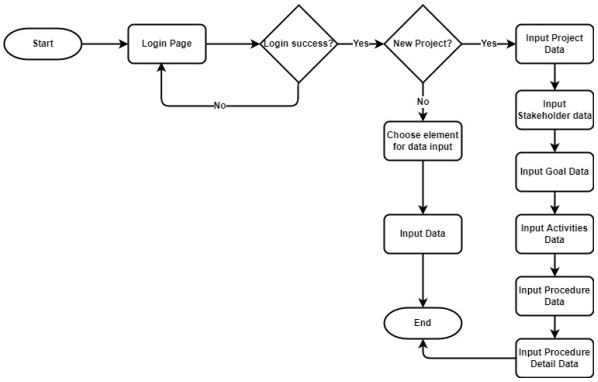


Figure 5. The flow diagram of the use of the tool

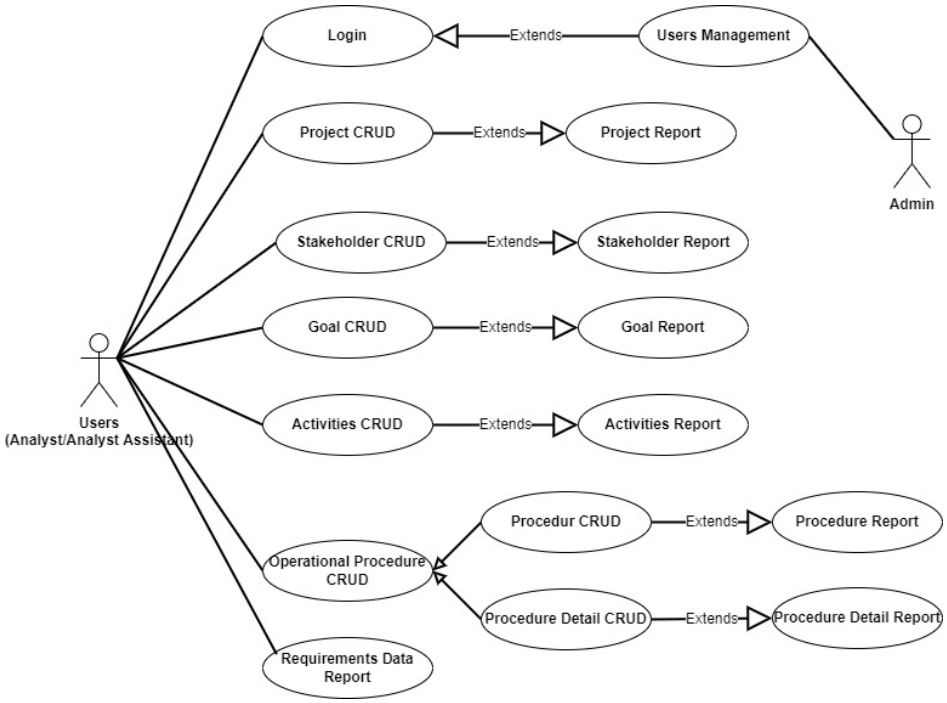


Figure 6. Use Case Diagram GoEliTools.

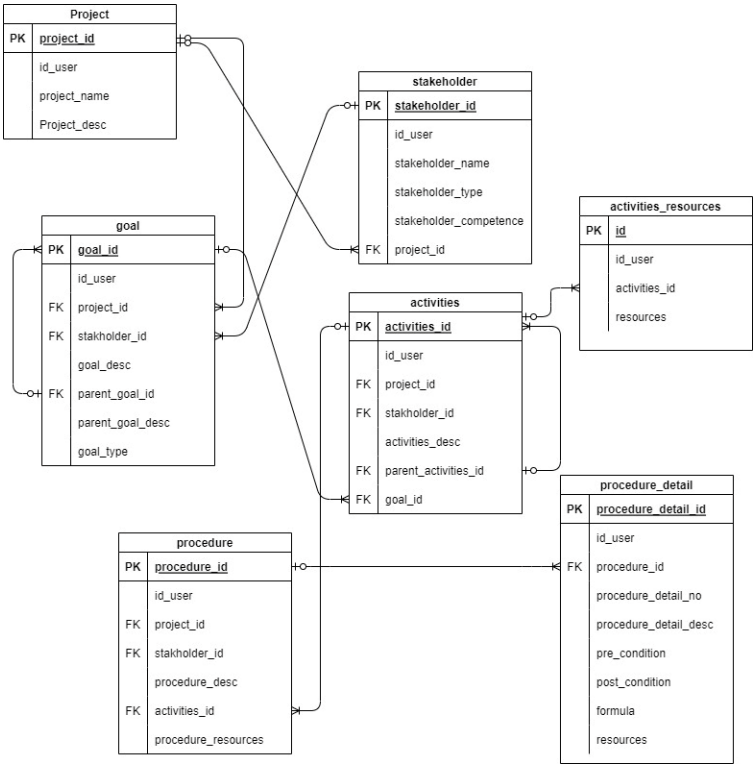


Figure 7. GoEliTools Database Design.

The database design of the tool can be seen in Figure 7, where the database consists of seven tables, namely the project table, stakeholders, goals, activities, activity resources, procedures, and detailed procedures. The relationship between tables in the database tool can be seen in Figure 7.

The object and relation extraction features are intended to obtain requirements data ready for analysis as the next stage of RE. The extraction process begins with pre-processing the requirements data and continues with case folding, stopword removal, and tokenization. The stopword removal and tokenization process are carried out using the literary library in PHP.

The object extraction and relationships between objects in the requirements data are done using a rule-based approach. There are nine rules used to extract object

relations in GoEliTools. The nine rules can be seen in table 2.

TABLE II
EXTRACTING OBJECT AND RELATIONS RULES [7].

Rule	Notation
1	*g(x) ^ g(y) ^ p(x,y) → rgoals(y,x)*
2	*g(x) ^ t(y) ^ p(x,y) → rtaskgoal(y,x)*
3	*t(x) ^ t(y) ^ p(x,y) → rtasks(y,x)*
4	*t(x) ^ o(y) ^ p(x,y) → roperationaltask(y,x)*
5	*o(x) ^ o(y) ^ p(x,y) → roperationals(y,x)*
6	*t(x) ^ a(y,x) → rtaskactor(y,x)*
7	*o(x) ^ a(y,x) → roperationalactor(y,x)*
8	*t(x) ^ r(y,x) → rtaskresource(y,x)*
9	*o(x) ^ r(y,x) → roperationalresource(y,x)*

Table II is a rule defined using predicate logic notation. Where the predicate g(x) states x is a goal, t(x) states x is a task, o(x) states x is operational, a(y,x) states y is an actor, and x, r(y,x) states y is the resource of x, and p(x,y) states that x is the parent of y. Logical notation rgoals(y,x) is a relationship formed between goals, in which y and x are both goals. Predicate rtaskgoal(y,x) is a relation between tasks and goals where y is a task and x is a goal. Predicate rtasks(y,x) is a relation between tasks where y and x are both tasks. Predicate roperationals(y,x) is a relation between operations and tasks where y is operational, and x is a task. Predicate roperationals(y,x) is a relation between operations where y and x are both operational. Predicate rtaskactor(y,x) is a relation between task and actor where y is the actor, and x is the task. Predicate roperationalactor(y,x) is a relation between operational and actor where y is the actor, and x is operational. Predicate rtaskresource(y,x) is a relation between task and resource where y is the resource and x is the task, and predicate operationalresource(y,x) is a relation between operational and resource where y is the resource and x is the operational [7].

As an example, rule 1: g(x) ^ g(y) ^ p(x,y) → rgoal(y,x) is a condition when x and y are goals and x is the parent of y, then the relation is rgoal(y,x) which means that y forms relations between goals with x [7].

D. Coding

Program coding is done using the PHP programming language with the CodeIgniter framework. The database implementation uses MySql.

E. Testing and Implementation

Black-box testing is performed to test the tool. Black-box testing is carried out using a test case to ensure that the tool runs according to the defined business processes and workflows. The application of the tool is carried out by using five data sets of user requirements for the development of the five information systems.

III. RESULT AND DISCUSSION

A. Results

GoEliTools is built based on the input document form in Table I. There are seven data entry features: project data entry, stakeholder data, goals or features, activities, activity resources, activity procedures, and detailed procedures.

The project data entry feature is used to record software development project data. This feature can record, repair, delete, and print project data. The stakeholder data entry feature records all stakeholders involved in defining system requirements. Like the project data entry feature, this feature can record, correct, delete, and print stakeholder data. Stakeholder data entry features and stakeholder data display can be seen in Figures 10 and 11. The project data entry feature and stakeholder data display can be seen in Figures 8 and 9.

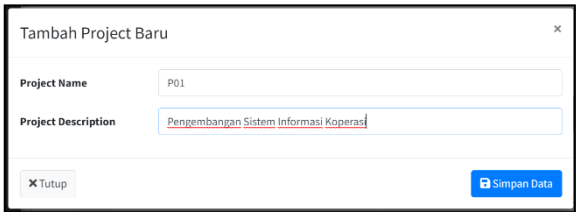


Figure 8. Project Data entry features

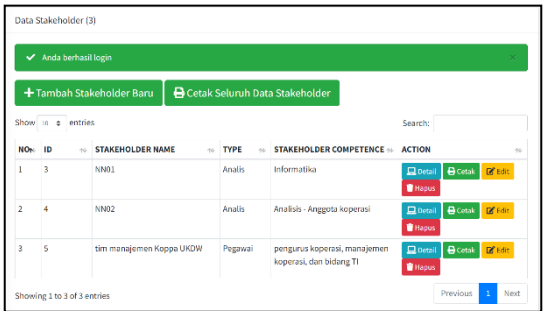


Figure 9. Stakeholder Data Feature

The goal/feature data entry feature is a feature for storing goal and feature data. The goal is the first element that must be defined in a goal-oriented approach. Meanwhile, features are goal elements that will become features in the system to be developed. Features are derivatives of goals; thus, each feature embodies one or more goals.

Likewise, a goal can be realized through one or more features. The relationship between goals and features is like the relationship between parent and child, where the goal is the parent, and the feature is the child. The goal data entry feature or goal data feature and display can be seen in Figures 10 dan 11.

The activity data entry feature is a feature to enter activity data or business processes in the organization where the system will be developed. The activity has relationships with previously defined features. The activity is the child of the system features to be developed. In data entry, the analyst must define the relationship between activities and system features to be developed.

The activity resources feature is a feature to complete the resources needed to run an activity. Resources can be actors, documents, and other resources. The activity data entry features and report for the activity list can be seen in Figures 12 and 13.

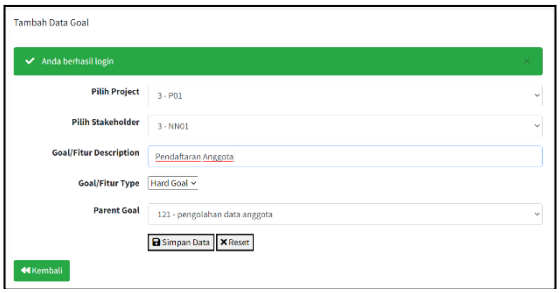


Figure 10. Goal/Feature Data entry page

Figure 11. Goal/Feature Data Display

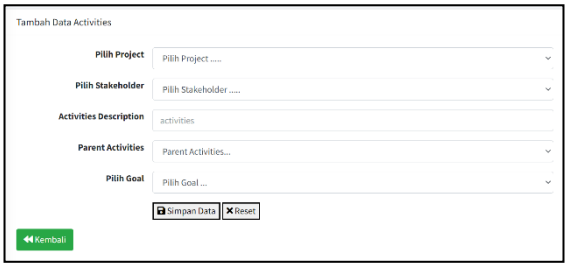


Figure 12. Activity Data entry page.

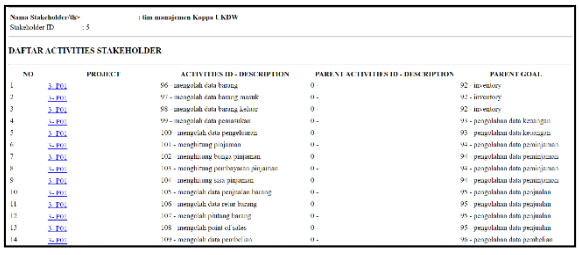


Figure 13. Activity List Report of each stakeholder.

The activity procedure feature is a feature to enter procedures related to the activities being carried out. Meanwhile, the detailed procedure data entry feature functions to detail each previously defined procedure. The interfaces for these two features can be seen in Figures 14 and 15.

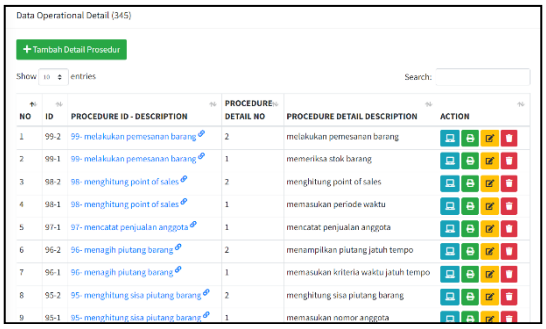


Figure 14. Detailed Procedure List Feature

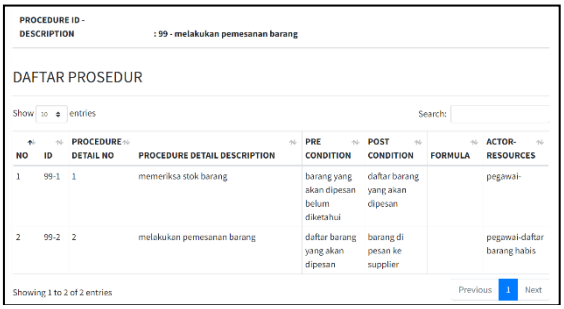
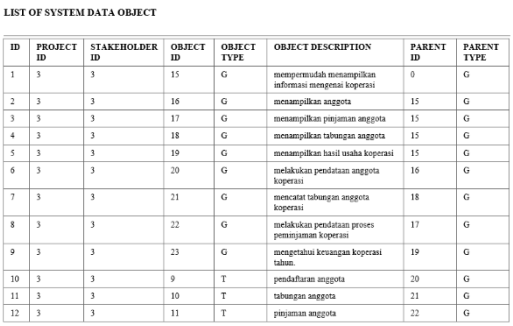


Figure 15. Detailed List Report for Each Procedure

The second feature in GoEliTools is object extraction and relationships between objects on the requirements data. In the goal-oriented approach, there are five main objects: goals, activities/tasks, procedures/operations, actors, and resources. The object is extracted based on the data table where the object is stored. The goal object is extracted from the goal table, the task object is extracted from the activity data, the operational object is extracted from the procedure table, the actor and resource object are extracted from the activities resources table, and the resources attribute is in the procedure table, and the resources attribute from the detailed procedure table. The extraction process is carried out sequentially, starting from the goals, activities, procedures, detailed procedures, and resources table. The results of object extraction can be seen in Figure 16.



LIST OF SYSTEM DATA OBJECT

ID	PROJECT ID	STAKEHOLDER ID	OBJECT ID	OBJECT TYPE	OBJECT DESCRIPTION	PARENT ID	PARENT TYPE
1	3	3	15	G	mempermudah menampilkan informasi mengenai koperasi	0	G
2	3	3	16	G	menampilkan anggota	15	G
3	3	3	17	G	menampilkan pinjaman anggota	15	G
4	3	3	18	G	menampilkan tabungan anggota	15	G
5	3	3	19	G	menampilkan hasil usaha koperasi	15	G
6	3	3	20	G	melakukan pendataan anggota koperasi	16	G
7	3	3	21	G	mencatat tabungan anggota koperasi	18	G
8	3	3	22	G	melakukan pendataan proses peminjaman koperasi	17	G
9	3	3	23	G	mengetahui keuangan koperasi tahun	19	G
10	3	3	9	T	pendaftaran anggota	20	G
11	3	3	10	T	tabungan anggota	21	G
12	3	3	11	T	pinjaman anggota	22	G

Figure 16. Object Extraction Results

The relations between objects are extracted based on the rules defined in Table II. The rules in Table II are translated into an object extraction algorithm which can be seen in algorithm 1 [7]. The results of the object relation extraction can be seen in Figure 17.

ID	PREDICATE	TERM1	TERM2	KODE TERM1	KODE TERM2	KETERANGAN
8	rgoal	mengetahui keuangan koperasi tahun.	menampilkan hasil usaha koperasi	23	19	rgoal(gchild,gparent)
9	rtaskgoal	pendaftaran anggota	melakukan pendataan anggota koperasi	9	20	rtaskgoal(task,goal)
10	rtaskgoal	tabungan anggota	mencatat tabungan anggota koperasi	10	21	rtaskgoal(task,goal)
11	rtaskgoal	pinjaman anggota	melakukan pendataan proses peminjaman koperasi	11	22	rtaskgoal(task,goal)
12	rtaskgoal	melakukan perhitungan keuntungan koperasi	mengetahui keuangan koperasi tahun.	12	23	rtaskgoal(task,goal)
13	rtask	mengisi formulir keanggotaan	pendaftaran anggota	13	9	rtask(tchild,tparent)
14	rtask	menyetor simpanan pokok wajib	pendaftaran anggota	14	9	rtask(tchild,tparent)
15	rtask	mencatat simpanan anggota bulan	tabungan anggota	16	10	rtask(tchild,tparent)
16	rtask	mencatat pinjaman anggota	pinjaman anggota	17	11	rtask(tchild,tparent)

Figure 17. Object Relation Extraction Results

Algorithm 1: **Object and relation extraction**

Object_relation_extraction() function
Input: table object
Output: table object_relation, requirements_data
// define root
Root[i] ← $\sigma_{parent\_id\ is\ null}(object)$
Predicte[i] ←*"rgoal"*
Insert_into_object_relation[i] ← *root[i] + predicate[i]*

//extract the other object
object[i] ← $\sigma_{parent\_id\ is\ not\ null}(object)$
for each rule in dom do
  match(object.object_type[i],object.parent_type[i])
  if match = True
    predicate[i] ← *relation_type*
  else
    predicate[i] ← θ
insert_into_object_relation[i] ← *object[i]+predicate[i]*

// create requirements_data
obj_relation[i] ← $\sigma(object\_relation)$
Insert_into_requirements_data[i] ← *obj_relation[i]*

B. Discussion on Tool Testing Results

Black-box testing is a functional test guided by the system's input/output behavior. In particular, the system receives external input so that the given by the system in response to the input will be used to verify the system's behavior without any assumptions of behavior other than the expected output [18][19]. Complete black-box testing should include verifying all possible inputs and outputs the system gives. Therefore, it can be said, in theory, it is impossible to do complete black-box testing [18] [19].

In this study, random method and cause-effect graphing techniques were applied. The random method is a black box testing technique that generates random test cases. Cause-effect graphing is a technique that relates the input (cause) and output (effect) [18]. The relationship between cause-effect is described in the form of a graph or can also be represented in a decision table [20]. Examples of applying the cause-effect graphing technique to the project data addition feature can be seen in Tables III to V. Table III lists causes, inputs, or actions given to input project data features. Table IV is a list of effects, reactions, or outputs given by the system to inputs, and Table V is a cause-effect graph represented as a decision table. If there is more than one cause, the relationship between the causes is "AND".

Test cases for testing are generated for all-cause conditions that have been defined. Black box testing for all test cases shows that the tool can be successful or in accordance with the test design. This means that the tool can work according to the planned conditions.

TABLE III
LIST OF CAUSES ON THE PROJECT DATA ENTRY FEATURE

Cause	Cause Description
C1	Fill in the correct username and password
C2	Choose the Sign In button
C3	Fill in the incorrect username or password
C4	Choose the project data menu
C5	Choose add a new project
C6	Choose print all project data
C7	Fill in the project name and project description data
C8	Clear the project name or project description data
C9	Choose to save button
C10	Choose close button
C11	Choose detail button
C12	Choose the edit project button
C13	Change project name or project description
C14	Choose reset button
C15	Choose back button
C16	Choose delete button
C17	Choose ok button
C18	Choose cancel button

TABLE IV
LIST OF EFFECTS ON THE PROJECT DATA ENTRY FEATURE

Effect	Effect Description
E1	Login succeeded
E2	Login failed
E3	Project page display
E4	Project data entry display
E5	Project report page display
E6	Inputted project data saved in the database
E7	Message Alert: project name or project description must be filled
E8	The project data entry page closed
E9	Detailed project page display
E10	Edit data project page display
E11	Save changing data
E12	Returns the original project name and project description data fields
E13	Close edit project data page
E14	Confirmation of action page display
E15	The record data project will be cleared
E16	Close confirmation of the action page

TABLE V
TABEL KEPUTUSAN CAUSE-EFFECT DECISION TABLE

	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	E10	E11	E12	E13	E14	E15	E16
C1	X																
C2	X	X															
C3		X															
C4			X														
C5				X													
C6					X												
C7						X											
C8							X										
C9						X	X					X					
C10								X									
C11									X								
C12										X	X						
C13												X	X				
C14													X				
C15														X			
	E1	**E2**	**E3**	**E4**	**E5**	**E6**	**E7**	**E8**	**E9**	**E10**	**E10**	**E11**	**E12**	**E13**	**E14**	**E15**	**E16**
C16															X		
C17																X	
C18																	X
E1			X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
E3				X	X				X	X					X		
E4						X	X	X									
E9											X						
E10												X	X	X			

In addition to testing, the tool has also been used to input data requirements sourced from five information system projects. The five projects are the Cooperative Information System project, the Personnel Information System, the Lecture Information System, the Financial Dashboard System, and the Human Resources Information System. The results of the application of the tool show that GoEliTools can be used to record data for software development needs. Table VI is a description of the requirements data with many stakeholders used for testing the application of the tool, and Table VII is the result of testing data entry and requirements data extraction process on the tool.

Evaluation of the application of the tool resulted in a description of the advantages and disadvantages of GoEliTools.

The tool has advantages, such as 1) tool already covers all elements of the GORE approach; 2) the tool has a data entry menu that matches the order of data entry; 3) the tool is built on a web-based so that they are easy to develop further; 4) data entry has facilitated parent and child relationships on model elements; and 5) the tool has been equipped with features of edit, delete, display data, and print reports that are quite varied; 6) tool can do object extraction and object relation correctly, and 7) tool can display joint information from all the stakeholders' requirements of a system development project.

The disadvantages of the tool include: 1) the data entry form does not accommodate data entry with one parent and many children (one to many); 2) the tool has not filtered the requirements data of each stakeholder; 3) the tool cannot accommodate data entry for more than one project; 4) some forms of reports are not yet available on the tool, so it is necessary to develop report features according to user's requirements, and 5) it is necessary to have consistency in naming objects and features to make it easier for users to remember and get used to using a tool.

TABLE VI
REQUIREMENTS DATA FOR TESTING

Num	Project Code	Project Name	Number of Stakeholders
1	P1	Cooperative information system	4
2	P2	Staffing system	4
3	P3	Lecture support system	4
4	P4	Financial dashboard system	8
5	P5	Human Resource Management System	4

TABLE VII
THE RESULT OF TESTING DATA ENTRY AND DATA EXTRACTION

Project code	Number of requirements data records				Number of Objects
	Goal	*Task*	*Op*	*Op dt*	
P1	107	158	176	345	1,126
P2	133	79	79	107	451
P3	333	555	400	954	3,709
P4	167	321	341	641	2,073
P5	385	704	548	1,430	4,581
Total	1125	1817	1544	3477	11,940

IV. CONCLUSION

In this study, the tool was successfully developed for the elicitation stage of the RE process. The tool is named GoEliTools. GoEliTools were created using a goal-oriented approach. The tool has two main features entry feature and object extraction features. The data entry feature has seven sub-features: project data processing, stakeholders, goals, activities, activity resources, procedures, and detailed procedures. The trial was conducted using the black-box method with cause-effect graphing techniques and random methods for generating test cases. The tool has also been used in five sets of data requirements from five information systems development projects. The test results show that the tool can work according to the planned inputs and outputs.

For further development, improvements can be made to GoEliTools by fixing all system deficiencies described in

the system deficiency analysis section. In addition, it is also necessary to test the tool's usability to know its level of usability and ease of use.

ACKNOWLEDGMENT

We want to thank the Institute for Research and Community Service Duta Wacana Christian University (LPPM UKDW), which has funded the research and publication of this research.

REFERENCES

[1] I. Sommerville, *Software Engineering Ninth Edition*, Ninth Edit. United States of America: Addison Wesley, 2011. doi: 10.1136/bmj.1.5802.756-b.

[2] R. S. Pressman and B. R. Maxim, *Software Engineering A Practitioner's Approach*, Eighth Edi. New York: Mc Graw Hill Education, 2015.

[3] R. Delima, R. Wardoyo, and K. Mustofa, "Automatic Requirements Engineering Model using Goal-Oriented Modelling with Text Pre-Processing Technique," in *Sixth International Conference on Informatics and Computing (ICIC)*, 2021, pp. 1–8. doi: 10.1109/icic54025.2021.9632980.

[4] N. Robinson, "Integrating Multiple Domain Specifications Goals," in *Proceedings of the 5th international workshop on Software specification and design (IWSSD) '89*, 1989, pp. 219–226.

[5] A. Van Lamsweerde, "Goal-oriented requirements engineering: A guided tour," in *Proceedings of the IEEE International Conference on Requirements Engineering*, 2001, pp. 249–261.

[6] 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.

[7] R. Delima, "Model Semi-Otomatis untuk Rekayasa Kebutuhan Perangkat Lunak," Universitas Gadjah Mada, Yogyakarta, 2022.

[8] E. Sarmiento, J. C. S. P. Leite, E. Almentero, and G. S. Alzamora, "Test Scenario Generation from Natural Language Requirements Descriptions based on Petri-Nets," *Electron. Notes Theor. Comput. Sci.*, vol. 329, pp. 123–148, 2016, doi: 10.1016/j.entcs.2016.12.008.

[9] F. Bargui, H. Ben-abdallah, and J. Feki, "A natural language-based approach for a semi-automatic data mart design and ETL generation," *J. Decis. Syst.*, vol. 25, no. 4, pp. 1–36, 2016, doi: 10.1080/12460125.2016.1158066.

[10] 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.

[11] A. Van Lamsweerde, "Goal-oriented requirements enginering: A roundtrip from research to practice," in *Proceedings of the IEEE International Conference on Requirements Engineering*, 2004, no. September, pp. 4–7.

[12] C. M. Nguyen, R. Sebastiani, P. Giorgini, and J. Mylopoulos, "Multi-objective reasoning with constrained goal models," *Requir. Eng.*, vol. 23, no. 2, pp. 189–225, 2018, doi: 10.1007/s00766-016-0263-5.

[13] J. Horkoff, N. A. Maiden, and D. Asboth, "Creative goal modeling for innovative requirements," *Inf. Softw. Technol.*, pp. 1–16, 2018, doi: 10.1016/j.infsof.2018.09.005.

[14] C. Kalloniatis, "Incorporating privacy in the design of cloud-based systems : a conceptual meta-model," *Inf. Comput. Secur.*, vol. 25 No.5, pp. 614–633, 2017, doi: 10.1108/ICS-06-2016-0044.

[15] W. Wirasta, H. L. Soemitro, and B. Hendradjaya, "Utilization of AHP Method in Elicitation Process for Goal Oriented Implementation using KAOS modelling," in *Proceedings of 2016 International Conference on Data and Software Engineering, ICoDSE 2016*, 2017, pp. 1–6. doi: 10.1109/ICODSE.2016.7936144.

[16] P. Giorgini, B. W. Eds, I. Conference, and D. Hutchison, "Advanced Information Systems Engineering," in *31st International Conference, CAiSE 2019 Rome*, 2019, vol. 932. doi: 10.1007/3-540-59498-1.

[17] R. Matulevičius and P. Heymans, "Visually Effective Goal Models using KAOS," in *FUNDP-PRECISE*, 2007, no. November. doi: 10.1007/978-3-540-76292-8.

[18] F. Lonetti and E. Marchetti, "Emerging Software Testing Technologies - Chapter Three," in *Advances in Computers*, 1st ed., vol. 108, Elsevier Inc., 2018, pp. 91–143. doi: 10.1016/bs.adcom.2017.11.003.

[19] P. S. Ganney, S. Pisharody, and E. Claridge, "Software Engineering," in *Clinical Engineering: A Handbook for Clinical and Biomedical Engineers-Second Edition*, Elsevier Ltd, 2020, pp. 131–168.

[20] GeeksforGeeks, "Software Engineering - Black box testing," *GeeksforGeeks*, 2022. https://www.geeksforgeeks.org/software-engineering-black-box-testing/