New Graphical Software Tool for Creating Cause-Effect Graph Specifications

Ehlimana Krupalija, Šeila Bećirović, Irfan Prazina, Emir Cogo, and Ingmar Bešić

Original scientific article

Abstract— **Cause-effect graphing is a commonly used black-box technique with many applications in practice. It is important to be able to create accurate cause-effect graph specifications from system requirements before converting them to test case tables used for black-box testing. In this paper, a new graphical software tool for creating cause-effect graph specifications is presented. The tool uses standardized graphical notation for describing different types of nodes, logical relations and constraints, resulting in a visual representation of the desired cause-effect graph which can be exported for later usage and imported in the tool. The purpose of this work is to make the cause-effect graph specification process easier for users in order to solve some of the problems which arise due to the insufficient amount of understanding of cause-effect graph elements. The proposed tool was successfully used for creating cause-effect graph specifications for small, medium and large graphs. It was also successfully used for performing different types of tasks by users without any prior knowledge of the functionalities of the tool, indicating that the tool is easy to use, helpful and intuitive. The results indicate that the usage of standardized notation is easier to understand than non-standardized approaches from other tools.**

Index terms—**software quality, black-box testing, cause-effect graph, graphical software tool.**

I. Introduction

Cause-effect graphing (CEG) was introduced as a black-box testing technique in 1970 as a new way of generating test cases for a given software product functionality [1]. It was later adopted as a standard black-box testing method in works such as [2], [3], [4] and [5]. Cause-effect graph specifications contain three different types of elements: nodes (causes, intermediates and effects), logical (Boolean) relations and dependency constraints. After creating the cause-effect graph specification from system requirements, different algorithms based on back-propagation of effect values through the graph such as [4], [6] and [7] can be used for deriving test case tables. These test case tables are then used for black-box testing the desired system.

The main difficulty in the usage of cause-effect graphs arises due to the large dimensionality of test cases. Each node can be in one of two states – active or inactive, which is why the total number of test cases is $2^{\text{number of causes}}$. It is very important to correctly specify all logical relations and constraints between causes and effects of the graph before generating the test case table by using the available algorithms. Usage of constraints reduces the feasible test case subset size, making the test case selection process easier and less costly [8]. Unfortunately, the definition of the most commonly used back-propagation algorithm for deriving test case tables from [4] contains many inconsistencies. Additionally, different types of constraints are often misinterpreted by users, which is why their usage is often omitted (e.g. MSK constrains in [9] and [10]). All of these problems are pointed out and explained in detail in [11]. Test-case-description-related problems (e.g. incomprehensible, abstract and poorly documented test cases) also pose a critical problem [12], which is why it is very important to correctly derive cause-effect graph element descriptions from system requirements. The lack of verification of the conformance of the cause-effect graph with the specification of the desired system leads to many problems with test case derivation from cause-effect graph specifications. For this reason, the usage of machine learning methods [13] and natural language processing algorithms [14] has recently been proposed for automatically converting system requirements to cause-effect graph specifications.

Another factor which increases the amount of errors in the cause-effect graph specification process is the misuse of graphical notation for representing cause-effect graph elements. Different types of notation are present in the available cause-effect graphing works and the lack of standardization between different approaches makes this process more error-prone. Some non-standard approaches regarding the notation used for depicting different logical relations are similar to the standardized notation such as [15], whereas in other cases such as [16] the notation is very different and may be hard to understand. Additionally, omitting intermediate nodes can lead to specifications which do not conform to system requirements or result in incorrect test case tables, as noted in [11]. This problem is also addressed in [14], which accentuates the importance of the conformance of cause-effect graph specifications and formal system requirements.

A limited number of software tools for aiding the process of creating cause-effect graph specifications is available. Most of these tools are not open-source and are not available for free usage. Only one available tool [17] contains graphical elements, whereas other tools such as [18] and [7] focus on the application of different CEG algorithms for deriving test case tables from cause-effect graph specifications provided by the user. The

Manuscript received July 26, 2022; revised October 10, 2022. Date of publication November 29, 2022. Date of current version November 29, 2022. The associate editor prof. Tihana Galinac Grbac has been coordinating the review of this manuscript and approved it for publication.

Authors are with the Department of Computer Science and Informatics, Faculty of Electrical Engineering, University of Sarajevo, Bosnia and Herzegovina (emails: ekrupalija1@etf.unsa.ba, sbecirovic1@etf.unsa.ba, iprazina1@etf.unsa.ba, ec15261@etf.unsa.ba, ingmar.besic@etf.unsa.ba).

Digital Object Identifier (DOI): 10.24138/jcomss-2022-0076

aforementioned problems with the cause-effect graph specification process accentuate the necessity of a new software tool for making this process easier for domain experts and end users.

In this paper, a new graphical software tool for creating cause-effect graph specifications is introduced. The tool is open-source and can be used for free. It is designed for defining all cause-effect graph elements, including different node types, logical relations and constraints. The purpose of the tool is to reduce the amount of errors in the cause-effect graph specification process. The generated cause-effect graphs can then be used for deriving test case tables for black-box testing different types of systems. The tool currently does not support any algorithms for test case table derivation, however it was designed in such a way that the introduction of such functionalities can be easily incorporated into the tool in the future. The expected contributions of the work include:

- Introduction of a new software tool which uses standardized graphical notation for creating cause-effect graph specifications. The tool can also be used for exporting the specifications to *.txt* files and importing these files for later reuse and modification. Usage of this tool can help in reducing the time for generating black-box tests for a given system based on system requirements and offer output that conforms to the standard accepted notation present in the available literature.
- Analysis of the scalability of the graphical-tool-based approach for the specification of cause-effect graphs. Using a variety of small, medium and large graphs presented in related work, the proposed tool was evaluated in order to confirm that it can be successfully used for creating both simple and complex cause-effect graph specifications.
- The comparison of available software tools to the newly introduced graphical software tool in terms of usage of graphical elements and usability. Usability of the newly proposed tool and its comparison to other available tools were evaluated by conducting surveys for user-based evaluation.

This work is structured as follows. In section II, cause-effect graphs are introduced and their main elements are explained in detail. Background and related work are also discussed in this section. In section III, the graphical user interface and functionalities of the proposed software tool are explained. The evaluation of the graphical tool by using different examples from the relevant literature and other available tools is summarized in section IV. Two surveys were conducted for determining the usability of the new software tool and for comparison with other existing approaches. Section V contains the explanation of internal and external threats to validity of the conducted study, as well as its limitations. In section VI, the overall analysis of the software tool is conducted with its comparison to earlier approaches. Recommendations for further research and possible future enhancements of the tool are also given in this section.

II. Preliminaries and Related work

This section contains the necessary preliminaries for understanding the cause-effect graph specification process. In Section II.A. all types of cause-effect graph elements are defined: different types of nodes, logical relations and dependency constraints, which will be used in the proposed graphical software tool. Section II.B. contains the explanation of all related work in the field of research, including the wide usage of cause-effect graphs for different applications, available types of graphical notation and a systematic review of existing CEG software tools.

A. *Cause-effect Graph Specification Elements*

Cause-effect graph specification and its elements are defined and explained in many works such as [1], [2] and [4]. Every cause-effect graph contains three different types of elements – graph nodes, logical relations and dependency constraints. All defined graph nodes, regardless of their type, can be in one of two states – active (1) or inactive (0). Cause-effect graphs can contain three different types of nodes:

1) *Causes*, which are used to describe different variables or events which result in the activation of effects in the system. Cause nodes are always placed on the left side of the graph. Causes are denoted as C_i (where $i > 0$ represents the number of the node). Every cause-effect graph must have at least one cause node.

2) *Effects*, which are used to describe different variables or events which are triggered by the causes of the system. Effect nodes are always placed on the right side of the graph. Effects are denoted as E_i (where $i > 0$ represents the number of the effect node). Every cause-effect graph must have at least one effect node.

3) *Intermediates*, which are used as helpers for capturing different logical relations between cause nodes. The purpose of these nodes is to reuse the effects of logical relations as causes for other logical relations. Intermediate nodes are always placed between the causes and effects of the graph. Intermediates are denoted as I_i (where $i > 0$ represents the number of the intermediate node). Usage of intermediates is optional.

Six different logical relations can be defined between graph causes, intermediates and effects. Due to the initially intended purpose of cause-effect graphs for testing hardware logical circuits in [1], truth tables of logical relations are very similar to those of logical gates. In addition to the four standard logical relations: DIR (*direct*), NOT (*negation*), AND (*conjunction*) and OR (*disjunction*), two additional relations (which are omitted in some works such as [4] and [19]) can be used: NAND (*Peirce arrow*) and NOR (*Sheffer stroke*). The direct and negation logical relations are unary, meaning they have exactly one cause and one effect node, whereas all other logical relations are *n*-ary, meaning they have $n > 1$ causes and one effect node. The truth table for all six logical relations is shown in Table I, where nodes (labeled as $\boldsymbol{N_1}$ and $\boldsymbol{N_2}$) represent causes and the operation result represents the resulting effect value (value of 0 means that the effect is not activated and vice versa). Only $\boldsymbol{N_1}$ values are used for determining the results of the direct and negation relations, which are unary.

Five different dependency constraints can be defined between graph causes or effects: EXC (*exclusion*), INC (*inclusion*), REQ (*required*), MSK (*masking*) and EXC Δ INC (*one and only one – exclusive inclusion*). All constraints except for MSK are defined on causes, whereas the MSK constraint is defined on effects. Intermediates cannot be used in constraints. EXC, INC and EXC Δ INC constraints are *n*-ary, meaning they have $n > 1$ causes, whereas REQ and MSK constraints are binary, meaning they are defined on exactly two nodes. The truth table for all five constraints is shown in Table II, where depending on the type of the constraint, nodes $\boldsymbol{N_1}$ and $\boldsymbol{N_2}$ can either represent causes or effects, whereas the operation result represents the resulting test case feasibility (value of 0 means that the test case is not feasible and vice versa).

TABLE I
TRUTH TABLE FOR ALL LOGICAL RELATIONS

Cause values		Resulting effect value					
N_1	N_2	DIR	NOT	AND	OR	NAND	NOR
0	0	0	1	0	0	1	1
0	1	0	1	0	1	1	0
1	0	1	0	0	1	1	0
1	1	1	0	1	1	0	0

TABLE II
TRUTH TABLE FOR ALL CONSTRAINTS

Node values		Resulting test case feasibility				
N_1	N_2	EXC	INC	REQ	MSK	EXC Δ INC
0	0	1	0	1	1	0
0	1	1	1	1	1	1
1	0	1	1	0	1	1
1	1	0	1	1	0	0

B. Related Work

Cause-effect graphs have been applied to many problems used in many areas such as telecommunication distributed systems [20], quantum programming [10], knowledge assessment [21], automatic college placement process [22] and high-speed safety-critical railway systems [9]. Due to their similarity to digital circuits [4], cause-effect graphs can be applied to a variety of problems that require logical relation modeling, in order to determine which test cases are feasible so the overall number of tests can be reduced. Cause-effect graphs have also been combined with other techniques, such as pairwise testing [23], UML model transformations [24] [25] and Boolean differentiation [26] [7].

The wide usage of cause-effect graphs as a black-box testing technique accentuates the importance of the process of correctly defining cause-effect graph specifications, which is imperative in order to correctly derive black-box test cases for the desired system. In some of the aforementioned works, constraint usage is omitted, which results in more feasible test cases than what is truly defined by system requirements. For example, in [9] the MSK constraint is not used between effect nodes. However, the effects are simultaneously exclusive and the existence of this constraint is necessary for removing infeasible test cases. In [22], the test case table contains test cases which do not conform to the presented cause-effect graph specification (e.g. when C6 is active, E2 is not set to active although these two nodes are connected by using the DIR logical relation). This indicates that cause-effect graph elements are used incorrectly or their usage is omitted in multiple available works, which results in incorrect feasible test case table specifications.

Different works use different types of graphical notation for describing cause-effect graphs, sometimes introducing elements which are hard to understand as they do not conform to any introduced standard. Due to these inconsistencies, it is important to analyze the standard accepted CEG graphical notation and its variations. In cases where multiple different proposed notations exist and are widely used, such as in case of entity relationship diagrams (ERDs), methods for conversion between different notations and detailed descriptions of differences between accepted standards are necessary [27]. However, in cause-effect graphs there is only one general standard and notations which do not conform to this standard are present only in a small number of available works. The initial graphical notation for representing cause-effect graph elements was introduced in [1]. This notation is very simple and includes the usage of different types of letters for representing logical relations and constraints (e.g. A for conjunction and O for disjunction). Full lines are used for connecting nodes by logical relations and dashed lines are used for connecting nodes by constraints.

Although the usage of full and dashed lines was adopted by all later works, many changes were made in the graphical notation for representing logical relations. Most works [2] [3] [4] [7] use graphical elements instead of letters to represent logical relations (e.g. wavy line for negation, arch and the symbol "∨" for disjunction) and this type of notation is considered as the general standard. However, some approaches use graphical notation which significantly differs from the standardized notation. For example, in [21] arrow tips are used for connecting nodes by logical relations, although arrow tips are standardly used only for representing the direction of the REQ and MSK constraints. In [15], logical relations are represented through the usage of bounding boxes and different symbols (e.g. AND instead of A for conjunction). A novel graphical notation with many differences to the general standard was proposed in [16]. This notation is more suited for the requirement elicitation process and introduces many new types of cause-effect graph elements such as membership and interactions. However, CEG representations created by the usage of this novel approach cannot be directly used for deriving test case tables and an algorithm for performing this conversion has yet to be introduced.

The development of software tools for aiding the process of cause-effect graph specification began with the introduction of TELDAP (Test Library Design Automation Program) in the initial work that introduced cause-effect graphs as a black-box testing technique [1]. TELDAP was an APL/360 program capable of processing cause-effect graph specifications and converting them to test case tables. TELDAP is outdated and no longer supported, so it will be omitted by this study. BenderRBT [17] is a commercial software tool for creating cause-effect graph specifications through a graphical user interface. This tool is not free for usage and does not use standard graphical notation. In fact, the graphical notation used in this tool does not conform to any of the previously mentioned works (e.g. no works use the symbol "=" for describing the DIR relation, the definition of the MSK constraint does not conform to the definition present in the standard literature, etc.).

Problems arising due to a low number of available software tools for creating CEG specifications were reported as far back as 1997 in [28]. Since then, several new CEG software tools have been introduced. Cause-effect graph software testing tool (CEGSTT) [18] is another tool for creating cause-effect graph specifications. It is not open-source nor available for free usage. This tool does not contain graphical elements, nor does it generate the graphical description of a cause-effect graph. Test generator for cause-effect graphs (TOUCH) [7] is a recently proposed software tool and it also does not contain graphical elements for cause-effect graph specifications. However, this tool is open-source [29] and cross-platform unlike BenderRBT and CEGSTT, which only run on the Microsoft Windows operating system. TOUCH supports test case table derivation by using many proposed algorithms based on Boolean experssions, whereas BenderRBT and CEGSTT use only the common Myers' backward-propagation algorithm for this purpose. Korean requirement analyzer for cause-effect graphs (KRA-CE) [30] is another recently proposed software tool which automatically converts system requirement descriptions to cause-effect graph specifications and test case tables. However, this approach is localized to the Korean language and does not contain graphical elements of cause-effect graph specifications. The tool is cross-platform but it is currently not open-source.

Table III contains a comparison of the previously described currently available CEG software tools. The number of available tools is limited and only one tool is open-source and can be used for free. Only one tool, which is commercial and not available for free usage, contains graphical and non-standardized elements of cause-effect graphs. Additionally, works which focus on applying the cause-effect graphing technique on different types of problems contain inconsistencies regarding notation usage and conformance to system requirements. All of the aforementioned drawbacks serve as the main motivation for the development of the new software tool proposed in this work. This tool, named ETF-RI-CEG, is an open-source graphical software tool. The primary goal of this tool is to enable users to create cause-effect graphs by using standardized graphical notation. Test case derivation is a functionality that is not yet supported. However, the import/export feature present in this tool makes it possible to save and reuse cause-effect graph specifications. In this way, new, upgraded versions of the tool in the future can be used for generating test case tables for the previously created cause-effect graphs and their graphical specifications.

TABLE III
COMPARISON OF AVAILABLE CEG SOFTWARE TOOLS

Tool	Type of specification	Graphical notation	Test case derivation	Cross-platform	Availability	Import/ export feature
BenderRBT [17]	Graphical	Custom	Myers' algorithm	No	Commercial	.CEG
CEGSTT [18]	UI	-	Myers' algorithm	No	Not open-source	Unknown
TOUCH [7] [29]	UI	-	multiple algorithms	Yes	Open-source	.Graphml
KRA-CE [30]	Textual	-	C3tree algorithm	Yes	Not open-source	Unknown
ETF-RI-CEG	Graphical	Standard	No	No	Open-source	.Txt

III. THE PROPOSED GRAPHICAL SOFTWARE TOOL

The proposed graphical software tool is named ETF-RI-CEG. The tool is a desktop application developed by using the .NET 5 framework and the C# programming language in the Windows Forms application template type. .NET 5 version of the framework is cross-platform and supported on Microsoft Windows, Mac OS and Linux operating systems. However, in order to be able to run the tool on Mac OS or Linux, Wine needs to be installed as the Windows Forms template execution is not yet supported. Only the installation of .NET 5 Runtime is required to be able to run the tool on Microsoft Windows. The software tool is open-source and available for free usage. It can be accessed online on GitHub[1]. The user manual of the tool contains specifications on how to build and use the application.

When the proposed tool is first opened, it contains an empty panel and options to add graph nodes, logical relations and constraints. Multiple operations can be performed with graph nodes in the proposed tool. New nodes can be added to the graph, existing nodes can be moved on the graph or deleted from the graph entirely. Adding and moving nodes is done by using the drag-and-drop operation on the panel, whereas node removal is performed by using the list of existing nodes and the *Delete* button. New nodes are always assigned the lowest unclaimed number for its type (e.g. if nodes C_1 and C_3 are defined in the graph, the new node will be denoted as C_2). After deleting a node from the graph, all logical relations and constraints that contain this node are also deleted.

Multiple operations can be performed with logical relations and constraints of the graph. New relations and constraints can be added to the graph or deleted from the graph entirely. All operations on logical relations and constraints are done in the same way. The graphical representation of logical relations and constraints after they are added to the graph in the graphical tool is shown in Fig. 1 (all types of logical relations) and Fig. 2 (all types of constraints). This representation uses the standardized graphical notation from [4] and [7].

Adding and removing logical relations and constraints is done in the same way for all different types in the graphical software tool. Adding a new logical relation or constraint is more complex than adding nodes, because all nodes which are part of the desired logical relation or constraint need to be selected. To make the selection process easier for the user, every selected node is marked with a black box. Removing existing logical relations and constraints in the proposed tool

[1] ETF-RI-CEG can be accessed by using the following link: https://github.com/ehlymana/ETF-RI-CEG-Graphical

works in the same way as removing an existing node, by selecting the desired logical relation or constraint from the corresponding list and clicking on the *Delete* button.

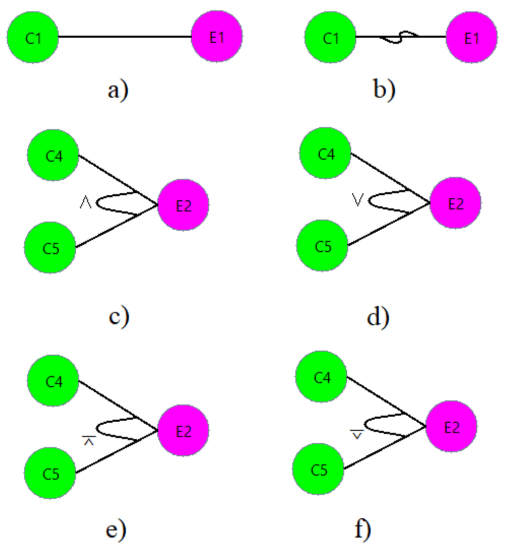


Fig. 1. Representation of logical relations in the graphical software tool: a) DIR relation, b) NOT relation, c) AND relation, d) OR relation, e) NAND relation, f) NOR relation

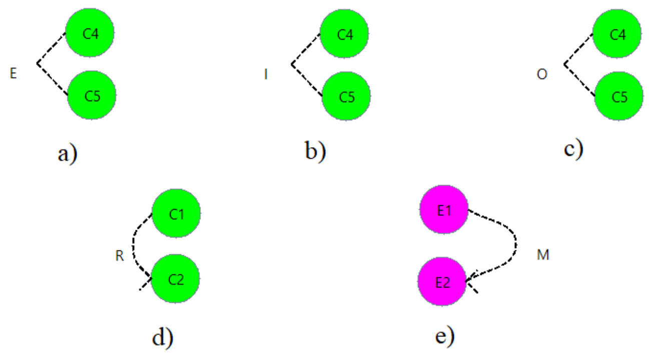
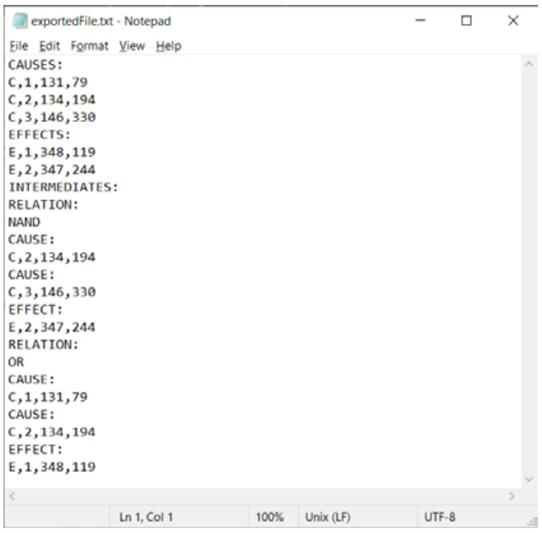


Fig. 2. Representation of constraints in the graphical software tool: a) EXC constraint, b) INC constraint, c) EXC Δ INC constraint, d) REQ constraint, e) MSK constraint

After defining the desired cause-effect graph, it can be saved for later usage by using the *Import/Export* option. The exported *.txt* file contains the structure of the graph (graph nodes, their locations, logical relations and constraints) created in the graphical software tool, as shown in Fig. 3. The contents of the file are easily readable and can be used for importing the graph for later usage in the proposed tool. When importing an existing graph file, the user is prompted to choose the desired exported file that contains the graph definition, after which the graph is shown on the panel of the tool, where it can be modified.



```
CAUSES:
C,1,131,79
C,2,134,194
C,3,146,330
EFFECTS:
E,1,348,119
E,2,347,244
INTERMEDIATES:
RELATION:
NAND
CAUSE:
C,2,134,194
CAUSE:
C,3,146,330
EFFECT:
E,2,347,244
RELATION:
OR
CAUSE:
C,1,131,79
CAUSE:
C,2,134,194
EFFECT:
E,1,348,119
```

Fig. 3. Contents of the *.txt* file generated by using the Export feature in the graphical software tool, which show the structure of the graph (graph nodes, logical relations and constraints)

IV. Evaluation

Evaluation of the proposed graphical tool was done by using three different approaches. First, multiple cause-effect graphs of different sizes from the standard literature were used for checking whether ETF-RI-CEG is scalable to graph size during the CEG specification process. The newly proposed tool was also compared to the only other currently available software tool which contains graphical elements, BenderRBT. Example graphs and a user survey were used for this purpose. User-based evaluation was also used for determining the overall usability of ETF-RI-CEG and multiple usability metrics were calculated based on the gathered information from multiple users.

A. Evaluation of Software Tool Scalability

In order to validate that the newly introduced graphical software tool can be applied for successfully creating CEG definitions, the following examples from the relevant literature were used:

1) A small cause-effect graph ($n = 6$ cause and effect nodes) from [4] which is shown in Fig. 4. This representation uses the standard accepted graphical notation and contains causes, effects, intermediates and different types of logical relations, as well as a single constraint. Fig. 5 shows the definition of the same cause-effect graph in the proposed tool. It is visible that the two graphs are nearly identical due to the fact that the same notation is used for representing the graph. The only difference can be seen in the naming convention for nodes – the original representation uses numbers associated with the system requirements, whereas the proposed tool explicitly defines the type and number for every node shown on the graph.

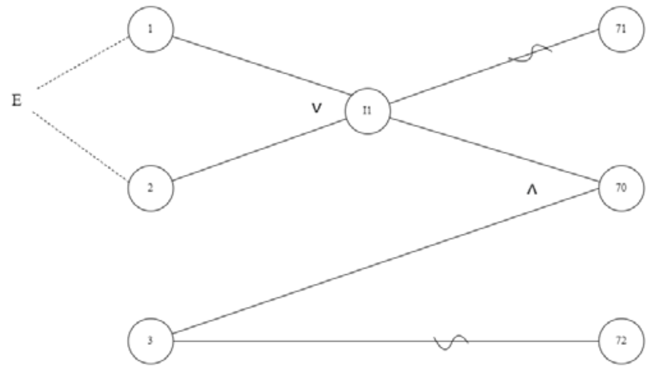


Fig. 4. Example cause-effect graph from [4] which contains three cause nodes, three effect nodes, one intermediate node, four logical relations and one constraint

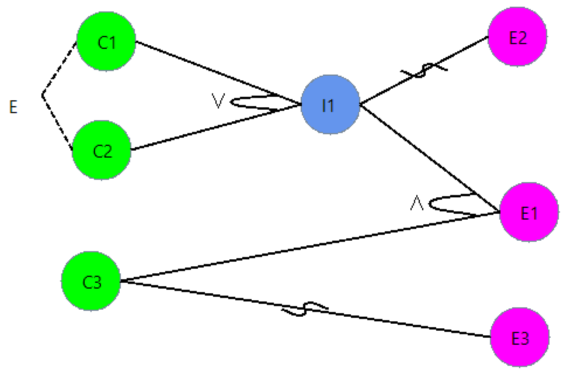


Fig. 5. The output of the proposed graphical software tool for the cause-effect graph defined in Fig. 4

2) A medium cause-effect graph (*n = 10* cause and effect nodes) from the original work that introduced the cause-effect graphing technique [1] which is shown in Fig. 6. This graph uses graphical notation proposed in the original work (A represents the AND relation, O represents the OR relation, etc.). Fig. 7 shows the definition of the same cause-effect graph in the proposed graphical software tool, where the standardized graphical notation is used to represent logical relations and one intermediate node is visible on the graph. There are many differences to the original representation including separate numbering for different types of nodes and the usage of improved graphical notation for representing logical relations. The most important difference to the original representation is the replacement of the EXC constraint originally defined on nodes C_1 and I_1. Using intermediates in constraints is not directly allowed in the proposed tool (because constraints can only be defined on cause nodes). This relation was replaced by putting an EXC constraint on both of the cause nodes that result in the activation of the intermediate node I_1 to capture the desired relation.

3) A large cause-effect graph (*n = 25* cause and effect nodes) from [4] which is commonly used for comparison due to its large size, which is shown in Fig. 8. Fig. 9 shows the definition of the same cause-effect graph in the proposed tool. Some of the differences to the original representation have already been mentioned in previous examples, including changes regarding node numbering. New intermediate nodes were added to the graph in order to capture the NOT logical relation prior to the usage of the desired nodes in binary logical relations, because unary and binary logical relations cannot be combined. The main difference to the original representation is the replacement of the negated REQ constraint originally defined on nodes C_{17} and C_8. The usage of this constraint is not directly allowed in the proposed tool (because constraint values cannot be negated). This relation was replaced by using the EXC Δ INC constraint (which renders two out of four combinations C_{17}-C_8: 0-0 and 1-1 infeasible) combined with the REQ constraint (which renders the third combination C_{17}-C_8: 0-1 infeasible, leaving only the desired combination C_{17}-C_8: 1-0 feasible).

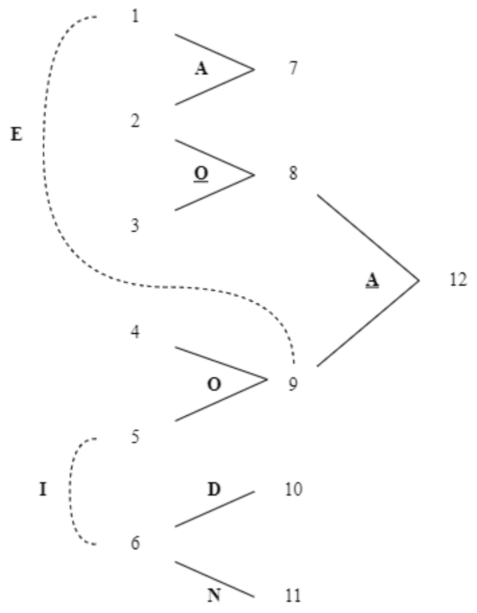


Fig. 6. Example cause-effect graph from [1] which contains six cause nodes, five effect nodes, one intermediate node, six logical relations and two constraints

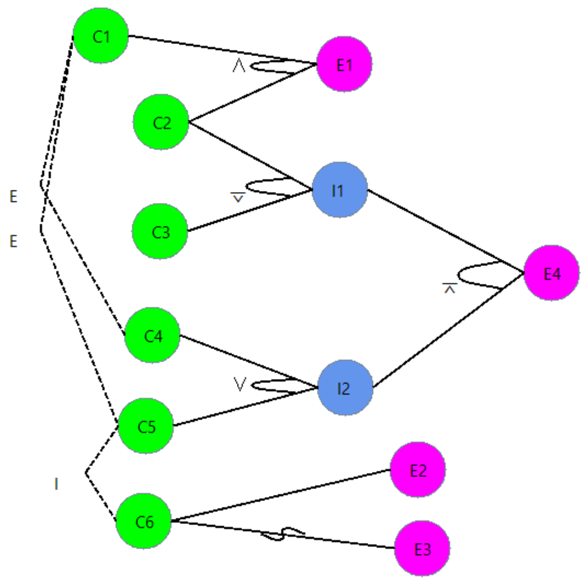


Fig. 7. The output of the proposed graphical software tool for the cause-effect graph defined in Fig. 6

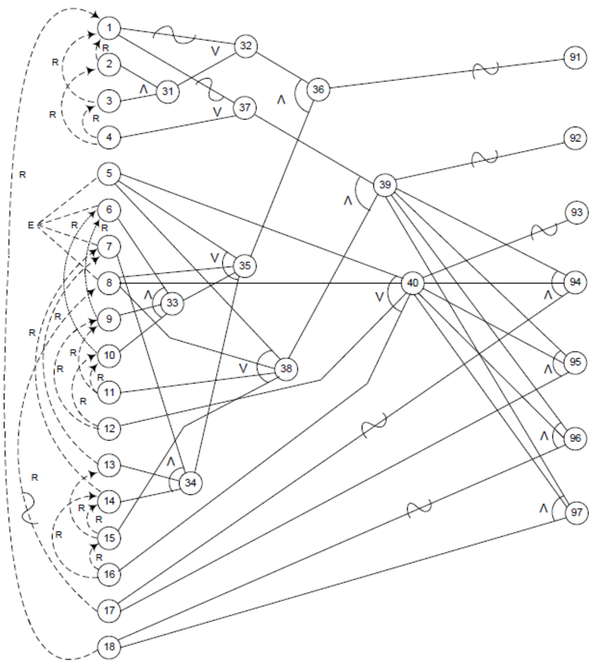


Fig. 8. A very large cause-effect graph from [4] which contains eighteen cause nodes, seven effect nodes, thirteen intermediate nodes, twenty logical relations and twenty-four constraints

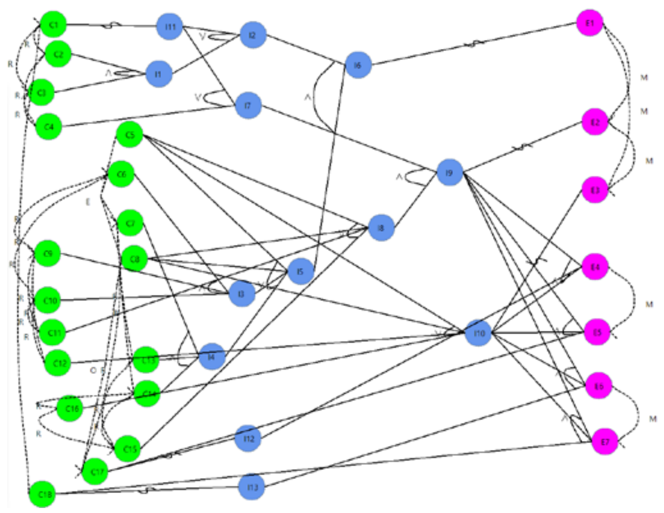


Fig. 9. The output of the proposed graphical software tool for the cause-effect graph defined in Fig. 8

B. Comparison to Other Available Tools

In order to compare the understandability of the proposed graphical software tool with existing approaches and tools, the only available graphical tool – BenderRBT [17] [31] was used. This tool enables users to graphically define cause-effect graphs, but it does not use standardized graphical notation. An example graph from [17] created by using this tool is shown in Fig. 10, whereas the definition of the same cause-effect graph in the proposed tool is shown in Fig. 11. There are many differences between the representations created by BenderRBT and the proposed tool. In BenderRBT, nodes are not numbered but instead contain descriptions related to system requirements. Standardized graphical notation is not used to represent logical relations or constraints.

Some differences which needed to be made have already been mentioned, such as combining the usage of unary and binary relations. The main difference to the original representation is the replacement of the MSK constraint originally defined on causes (which is forbidden, because the MSK constraint can be applied only to effects). It was additionally unclear what the meaning of the negated MSK constraint present in Fig. 10 was. As explained by [31], negation on the MSK relation is used in BenderRBT to represent the subject of the MSK relation, whereas the other links are used to represent the objects of the MSK relation. This does not conform to the standard definition of the MSK relation explained in Section II. Instead, it represents the REQ relation between the subject and the objects of the MSK relation. Therefore, the identified REQ relations were added to the graph as a replacement for the incorrectly applied MSK relation.

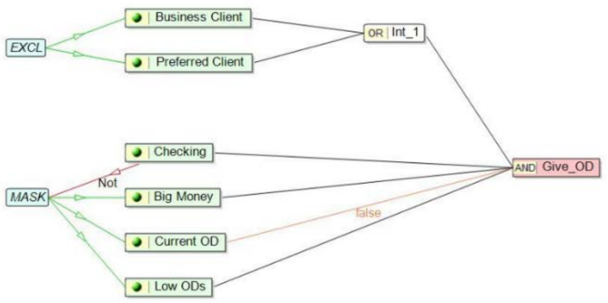


Fig. 10. A cause-effect graph from [17] which contains six cause nodes, one effect node, one intermediate node, two logical relations and two constraints

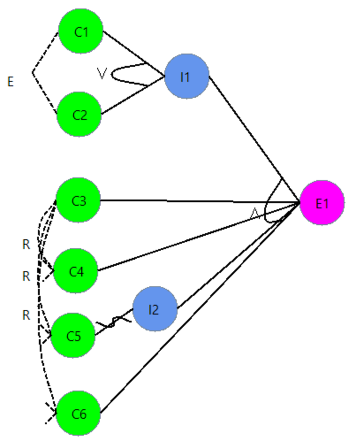


Fig. 11. The output of the proposed graphical software tool for the cause-effect graph defined in Fig. 10

The presented example and the achieved results by comparing the output of BenderRBT with the output of ETF-RI-CEG could not be generalized, because this example might not be representative of all BenderRBT outputs. In order to improve the objectivity of the results, a user survey[2] was conducted. The survey contained multiple examples which can

[2] The user survey can be accessed by using the following link: https://forms.gle/ypcHEP5YHkWC28FD6

be found in the BenderRBT user manual [31] and their representations created in ETF-RI-CEG. The survey focused on comparing the different representations and getting information from the users on which representation is easier to understand, how complex they perceive the cause-effect graphs, whether they can correctly identify results of logical relations and constraints, etc. The survey was completed by 59 BSc and MSc students of Faculty of Electrical Engineering, University of Sarajevo. Their background knowledge of software testing and cause-effect graphs is summarized in Table IV. Most participants had little to no experience with software testing and more than a third of participants had not heard of cause-effect graphs before.

TABLE IV
STRUCTURE OF SURVEY PARTICIPANTS

Q1: How many years of experience in software testing do you have?			
< 1 year	1-2 years	2-3 years	>3 years
35.59%	32.20%	25.42%	6.79%
Q2: Are you familiar with cause-effect graphs?			
Yes		No	
62.70%		37.30%	

Participants were shown three different cause-effect graph specifications of varying complexities from BenderRBT and their equivalent representations from ETF-RI-CEG. They were also asked to choose the tool with a more intuitive UI. The achieved results are shown in Fig. 12. The results show that in all cases, a larger number of users chose cause-effect graphs generated by the proposed tool as their preferred choice. It is also visible that more users found the user interface of the proposed tool more intuitive than the user interface of BenderRBT.

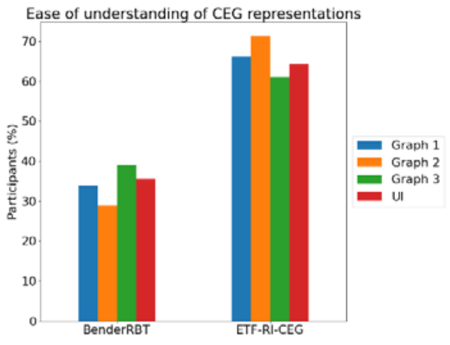


Fig. 12. The results of survey regarding user preference of CEG specifications generated by BenderRBT and ETF-RI-CEG software tools

The users' knowledge of constraints and ability to correctly identify cause-effect graph elements was also targeted by the survey. They were asked which types of nodes the REQ and MSK constraints are applied on, after which they were shown two CEG specifications – one containing the MSK constraint created by BenderRBT, and one containing the REQ constraint created by ETF-RI-CEG. The results are shown in Table V. A very low number of users correctly understood the REQ and MSK relations (28.8% and 27.1% respectfully), whereas an even lower number of users correctly identified test cases conforming to CEG specifications from BenderRBT and ETF-RI-CEG (20.3% and 16.9% respectfully). Additionally, only 20.3% of users were able to correctly identify that the two CEG representations generated by BenderRBT and ETF-RI-CEG from Fig. 10 and Fig. 11 were equivalent.

TABLE V
SURVEY RESULTS – KNOWLEDGE OF CONSTRAINT APPLICATION

Q1: Which type of nodes can the MSK constraint be applied to?			
Causes	Effects	Intermediates	I don't know
6.8%	27.1%	1.7%	64.4%
Identification of test cases from CEG MSK specification in BenderRBT			
Correct answer		Incorrect answers	
20.3%		79.7%	
Q3: Which type of nodes can the REQ constraint be applied to?			
Causes	Effects	Intermediates	I don't know
28.8%	5.1%	5.1%	61.0%
Identification of test cases from CEG REQ specification in ETF-RI-CEG			
Correct answer		Incorrect answers	
16.9%		83.1%	

The users were also shown a cause-effect graph of high complexity and its representations in BenderRBT and ETF-RI-CEG. They were asked to identify the number of all types of nodes contained in the graph and to rate the complexity of the graph representations from 1 to 10. The cause-effect graph representations generated by BenderRBT and ETF-RI-CEG achieved an average complexity rating of 7.41 and 5.62 respectfully. The exact number of nodes (causes, intermediates and effects) from representations generated by BenderRBT and ETF-RI-CEG were correctly identified by 25% and 59.38% of users respectfully. 57.6% of users answered that CEG nodes with textual descriptions made it easier to understand system requirements, whereas 49.2% of users answered that CEG nodes without textual descriptions made it harder to understand system requirements.

Other available tools (CEGSTT, TOUCH and KRA-CE) do not contain graphical elements of cause-effect graphs, as their primary purpose is the application of algorithms which are meant to convert cause-effect graph specifications to test case tables. Nodes and relations in these tools are instead defined by using the provided user interface or textual files. For this reason, the newly proposed tool could not be compared to these tools, as there is no graphical output for comparison.

C. *Evaluation of Usability*

User-based evaluation of the newly proposed tool was conducted by creating a remote usability survey[3] [32]. The survey contained directions on how to install ETF-RI-CEG and 12 tasks which the users needed to complete by using all functionalities of the proposed tool. The questions contained in the survey were based on the following user experience factors [33]: ease of use, efficiency, helpfulness, intuitive operation, learnability and simplicity. The survey was completed by 45 BSc and MSc students of Faculty of Electrical Engineering,

[3] The user survey can be accessed by using the following link: https://forms.gle/wbjFbfdf4LVkVrie7

University of Sarajevo. The achieved usability metric values as defined in [32] are summarized in Table VI. The participants were able to successfully complete 10.91 tasks out of 12 on average with an overall average success rate of 90.92%, overall average task accuracy of 74.81%, overall average efficiency of 0.285, overall average error rate of 3.96 and overall average critical statement ratio of 11.14.

TABLE VI
USABILITY METRICS OF ETF-RI-CEG STUDY

Metric	ETF-RI-CEG functionalities		
Success rate	Creating CEG elements	Modifying CEG elements	Export/Import feature
0-33% tasks	2.2%	2.2%	0%
33%-66% tasks	17.8%	11.1%	20%
66%+ tasks	80%	86.7%	80%
Average success rate	87.83%	94.67%	93.33%
Task accuracy	57.8%	86.7%	80%
Error rate	1.37	6.5	4.0
Efficiency	0.142	0.375	0.339
Average task duration	4.07 minutes	2.31 minutes	2.36 minutes
Critical statement ratio	6.5	5.43	21.5

Different types of mistakes which the users of ETF-RI-CEG reported are shown in Fig. 13. 43.8% of users did not report making any mistakes, whereas 50% of users who reported mistakes listed working with logical relations as the most difficult. Mistakes while using some functionalities such as element removal or the export/import feature were not reported by any users of the tool.

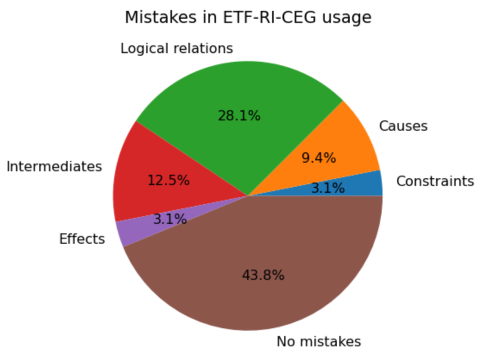


Fig. 13. Types of mistakes reported by users evaluating ETF-RI-CEG software tool

Users were asked to grade the proposed tool based on three user experience factors: ease of usage, intuitivity and helpfulness. The achieved results are shown in Fig. 14, where it can be seen that the proposed tool achieved results higher than 90% for all three factors of usability. Users were also asked to choose which functionalities they found easy and hard to use. The easiest functionalities identified by users were moving nodes (86.7%), adding nodes (84.4%) and the import feature (84.4%), whereas the hardest functionalities identified by users were adding complex logical relations (44.4%) and adding constraints (71.1%).

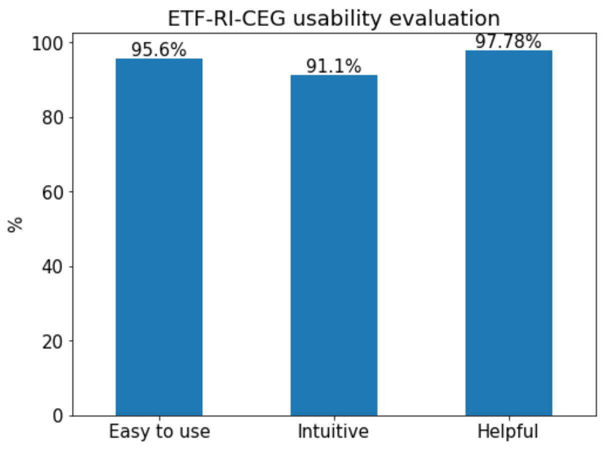


Fig. 14. Evaluation of different user experience factors by users of ETF-RI-CEG

D. *Result Analysis*

The achieved results when comparing the usage of notation in the standardized literature to results achieved by using the proposed tool from Section IV.A. indicate that standardized graphical notation and outputs of different logical relations and constraints are often violated by adding elements which are not defined in the available literature. This results in confusing CEG definitions with unclear relation outputs. Examples of these violations can be seen in Fig. 6, where a constraint is used on an intermediate node (although constraints cannot be defined on intermediate nodes), in Fig. 8, where unary relations (NOT) are combined with binary relations (AND) without the usage of intermediate nodes, where negated REQ relation is used (although the REQ relation cannot be negated), as well as in Fig. 10, where the negated MSK relation is used (although MSK relation cannot be negated) and where the MSK relation is used on cause nodes (instead of on effect nodes) in a completely misleading and confusing way, since the relation that wanted to be captured between the causes was actually the REQ relation.

The newly proposed tool solves the aforementioned problems by not allowing the user to manually specify anything that is not supported by the standardized literature. This makes it impossible to use an intermediate node in constraints or negate constraints whilst creating a cause-effect graph specification. These violations are not necessary and can be remodeled by using supported relation types, as demonstrated in all example graphs used for evaluation of the proposed tool. The user survey showed that most users do not understand the meaning and usage of the REQ and MSK constraints, which makes the necessity of the usage of standardized notation with explicitly defined truth tables even higher. This results in standardized and easily understandable cause-effect graphs that conform to standardized graphical notation and do not contain elements that are difficult to understand due to their inconsistencies with truth tables for logical relations and constraints.

Results of the user survey of ETF-RI-CEG when compared to the only other available software tool with the usage of graphical elements, BenderRBT, indicate that CEG specifications generated by the proposed tool are easier to understand. The textual descriptions present in BenderRBT representations did not make it significantly easier to understand system requirements, and equivalent representations from BenderRBT were rated as more complex than representations from ETF-RI-CEG. Usability metrics of the proposed tool indicate that most users were able to use the tool successfully and that difficulties in this process occurred when adding complex logical relations and constraints. It is important to note, however, that most users had little to no experience with cause-effect graphs and software testing in general. They were not taught how to use the tool or offered an user manual and they were still able to achieve an average success rate of 90.92%. Over 90% of users rated ETF-RI-CEG as easy to use, intuitive and helpful which indicates that the proposed tool will achieve similar results in the future, especially due to the fact that it will be open-source and free for usage.

V. Threats to Validity and Limitations

The evaluation of the newly proposed graphical software tool was conducted by using a large number of available cause-effect graph examples from the standard literature. A total of 16 graphs were created successfully with an average size of N = 9.4 nodes. The import/export feature of the tool enables other users to achieve the same results and access the successfully created cause-effect graph specifications. Additionally, ETF-RI-CEG is open-source and the exported generated cause-effect graph specifications are available for free usage. Therefore, the internal validity of this study has been achieved successfully because the experiment can be repeated and same results can be achieved without any external variables influencing the outcome.

The surveys conducted for evaluating the usability of the newly proposed tool and for comparing this tool with BenderRBT features expose threats to the external validity of the study. Both surveys were conducted in the form of a remote usability survey, which might result to results which are significantly biased. The participants of the surveys were mainly BSc and MSc students of Computer Science and Informatics, who had little to no experience in software testing. The overall number of participants was 45 and 59, which might not be a large enough sample for determining accurate results. Around 60% of participants were familiar with cause-effect graphs, although biased selection of subjects for taking the survey was avoided in a similar way as described in [12]. No training was provided to users before taking the survey, which might have affected the achieved results and increased the overall time required for completing tasks in the proposed graphical software tool.

The structure of participants poses another threat to external validity of the study. No domain or industrial experts were included in the study due to their unavailability, which might pose a problem in cases where open-source and industrial development have many differences, as reported in [34]. If BenderRBT was commonly used among domain experts in the software development industry, the usability of this tool would far outperform the usability of the newly proposed tool as the users would be more familiar with its interface. These claims cannot be proven or discarded due to the unavailability of this type of data. However, both surveys were conducted by using multiple consolidated factors from [33] for generating more objective results. A similar problem with the structure of participants and limited generalizability of the achieved results was reported in [35], however the structure of study subjects had a low impact on the achieved results due to the nature of the study itself and its human-centered concepts. A similar conclusion can also be derived for this study, because although the subjects of the study had some prior experience in software testing, they had not used neither ETF-RI-CEG nor BenderRBT before. This may lead to more objective results than if the tools were evaluated by domain experts who have had prior experience with BenderRBT, as the familiarity with any of the evaluated tools can affect the objectivity of the results and lead to different usability metric values.

The main limitation of this study is the unavailability of existing CEG software tools. Only one tool (TOUCH) is open-source and can be freely accessed, whereas no other existing tool could be successfully acquired by the authors of this paper. Due to this drawback, the properties of other existing tools could not be evaluated or validated (e.g. being cross-platform, the file format supported by the import/export feature), which is why the information from user manuals provided by the authors of the tools was used as the only relevant source. If BenderRBT had been available, a third survey requiring users to perform the entirely same tasks as in the study focusing on ETF-RI-CEG could have been conducted. These results could have then been compared and a more objective comparison between the usability of these two tools could have been made.

VI. Conclusion

Cause-effect graphing is a popular black-box testing technique widely used for creating test case tables necessary for executing black-box tests on the desired system. Problems with creating test cases for a given system often arise due to the insufficient amount of knowledge of cause-effect graph elements. This leads to the improper usage of logical relations and constraints while creating cause-effect graph specifications from system requirements. The standardized graphical notation is also often violated, resulting in specifications which are difficult to understand as they do not conform to truth table definitions and introduce elements without a sufficient amount of explanation or methods for their conversion to the standardized notation. A small number of software tools has been proposed for aiding the process of creating CEG specifications, however the available tools are mainly focused on the application of algorithms for deriving test cases instead of defining graphical cause-effect graph elements and most tools are not available for free usage. The only available graphical tool is commercial and does not use standardized graphical notation.

In this paper, a new graphical software tool for creating cause-effect graph definitions was presented. The tool aims to overcome the existing difficulties of creating cause-effect graph specifications which arise due to a poor understanding of logical relations and constraints. It also aims to provide a new and intuitive user interface that can help users create cause-effect graph definitions in a fast and efficient way while using standardized graphical notation without allowing the usage of any non-standardized elements in cause-effect graphs.

Several approaches have been used to evaluate the newly proposed software tool. The comparison between the graphical representation of the graphs created by using the proposed graphical software tool and the originally proposed representations shows that the new tool is scalable and can be used to successfully create specifications of cause-effect graphs of different sizes. The only differences in the graphical representations were a result of the improper usage of standardized graphical notation in the original works.

The results obtained by using the proposed tool when compared to the only other available graphical software tool show that the usage of standardized graphical notation creates specifications that are easier to understand than results obtained by using non-standardized graphical notation. This was verified by conducting a user survey, which showed that usage of textual descriptions of system requirements did not significantly improve the readability of cause-effect graph specifications. Users rated CEG specifications generated by using the proposed tool as less complex than the equivalent specifications generated by using the other available graphical tool. Another user survey which was conducted to evaluate the usability of the proposed software tool showed that most users found the proposed tool as helpful, easy and intuitive to use. High values of usability metrics indicate that the newly proposed software tool offers an intuitive and easily understandable output for users who can use truth tables as help when choosing the desired logical relations and constraints for cause-effect graph specifications. In this way, standardized graphical notation and explicit definitions can be used rather than non-standardized approaches.

The proposed graphical software tool was developed in the form of a desktop application. However, most tools are nowadays cloud-based and allow online collaboration between multiple users and an easily accessible user interface. Creating a web-based version of the proposed tool would remove the necessity of installation of prerequisites and the application itself, making the software tool available to more users and on multiple devices. Due to this, a web-application version of the software tool should be created by using the latest technologies, in order to make the tool fully cross-platform and widely used.

The output of the graphical software tool is the visual representation of the defined cause-effect graph, as well as an exported *.txt* file. This exported representation can potentially be used for reusing the graph definition in order to create black-box test case tables. This needs to be further explored for upgrading the graphical software tool with a new feature – automatically converting the graph definition into the desired test case table. The usability of the proposed graphical software tool would in this way be further improved and made comparable with other available tools, which already contain implementations of algorithms for test case table generating process from cause-effect graph specifications.

References

[1] W. R. Elmendorf, "Automated design of program test libraries," IBM Technical Report TR 00.2809, 1970.

[2] S. L. Pfleeger and J. M. Atlee, Software Engineering: Theory and Practice, 4th ed., New Jersey: Pearson Higher Education, 2010.

[3] M. E. Khan, "Different approaches to black box testing technique for finding errors," *International Journal of Software Engineering & Applications,* vol. 2, no. 4, pp. 31-40, 2011. doi: 10.5121/ijsea.2011.2404.

[4] G. J. Myers, T. Badgett and C. Sandler, The Art of Software Testing, 3rd ed., New Jersey: John Wiley & Sons, Inc., 2012, pp. 61-80.

[5] N. Anwar and S. Kar, "Review paper on various software testing techniques & strategies," *Global Journal of Computer Science and Technology,* vol. 19, no. 2, pp. 43-49, 2019. doi: 10.34257/GJCSTCVOL19IS2PG43.

[6] P. R. Srivastava, P. Patel and S. Chatrola, "Cause effect graph to decision table generation," *SIGSOFT Software Engineering Notes,* vol. 34, no. 2, 2009. doi: 10.1145/1507195.1507216.

[7] D. K. Ufuktepe, T. Ayav and F. Belli, "Test input generation from cause-effect graphs," *Software Quality Journal,* vol. 29, pp. 733-782, 2021. doi: 10.1007/s11219-021-09560-3.

[8] S. Singhal, N. Jatana, B. Suri, S. Misra and L. Fernandez-Sanz, "Systematic literature review on test case selection and prioritization: A tertiary study," *Applied Sciences,* vol. 11, no. 24, 2021. doi: 10.3390/app112412121.

[9] L. Dou and W.-D. Yang, "Design of test case for ATP speed monitoring function based on cause-effect graph," in *2019 CAA Symposium on Fault Detection, Supervision and Safety for Technical Processes (SAFEPROCESS),* Xiamen, 2019. doi: 10.1109/SAFEPROCESS45799.2019.9213325.

[10] N. Oldfield, T. Yue and S. Ali, "Investigating quantum cause-effect graphs," in *2022 IEEE/ACM 3rd International Workshop on Quantum Software Engineering (Q-SE),* Pittsburgh, 2022. doi: 10.1145/3528230.3529186.

[11] K. Nursimulu and R. L. Probert, "Cause-effect graphing analysis and validation of requirements," in *CASCON '95: Proceedings of the 1995 conference of the Centre for Advanced Studies on Collaborative research,* Toronto, 1995. doi: 10.5555/781915.781961.

[12] K. Juhnke, M. Tichy and F. Houdek, "Challenges concerning test case specifications in automotive software testing: Assessment of frequency and criticality," *Software Quality Journal,* vol. 29, pp. 39-100, 2021. doi: 10.1007/s11219-020-09523-0.

[13] B. Vogel-Heuser, V. Karaseva, J. Folmer and I. Kirchen, "Operator knowledge inclusion in data-mining approaches for product quality assurance using cause-effect graphs," *International Federation of Automatic Control (IFAC) PapersOnLine*, vol. 50, no. 1, pp. 1358-1365, 2017. doi: 10.1016/j.ifacol.2017.08.233.

[14] W. S. Jang and R. Y. C. Kim, "Automatic generation mechanism of cause-effect graph with informal requirement specification based on the Korean language," *Applied Sciences,* vol. 11, no. 24, 2021. doi: 10.3390/app112411775.

[15] J. Lal and S. Singh, "From cause to effect: An empirical study of cause-effect graphing testing techniques and its test measurement: A review," *International Journal of Computer Science and Technology,* vol. 3, no. 3, pp. 89-92, 2012.

[16] F. Huang and C. Smidts, "Causal mechanism graph - A new notation for capturing cause-effect knowledge in software dependability," *Reliability Engineering & System Safety,* vol. 158, pp. 196-212, 2017. doi: 10.1016/j.ress.2016.08.020.

[17] "Bender RBT (previously SoftTest/CaliberRBT)," BenderRBT Inc., [Online]. Available: https://benderrbt.com/bendersoftware.htm#rbt. [Accessed 4 June 2022].

[18] B. Bekiroglu, "A cause-effect graph software testing tool," *European Journal of Computer Science and Information Technology,* vol. 5, no. 4, pp. 11-24, 2017.

[19] S. Agrawal, R. Venkatesh, U. Shrotri, A. Zare and S. Verma, "Scaling test case generation for expressive decision tables," in *2020 IEEE 13th*

International Conference on Software Testing, Validation and Verification (ICST), Porto, 2020. doi: 10.1109/ICST46399.2020.00044.

[20] K. Nursimulu, *Cause-effect validation of requirements for distributed systems*, Ottawa: University of Ottawa, 1994. doi: 10.20381/ruor-16800.

[21] N. Gavrilović and L. Lazić, "Knowledge assessment using cause-effect graphing methods," in *The Seventh International Conference on eLearning (eLearning-2016)*, Belgrade, 2016.

[22] D. Jagli, T. Mamatha, S. Mahalingam and N. Ojha, "The application of cause effect graph for the college placement process," *International Journal of Software Engineering & Applications (IJSEA)*, vol. 3, no. 6, pp. 77-85, 2012. doi: 10.5121/ijsea.2012.3606.

[23] I. Chung, "Modeling pairwise test generation from cause-effect graphs as a Boolean satisfiability problem," *International Journal of Contents*, vol. 10, no. 3, pp. 41-46, 2014. doi: 10.5392/IJoC.2014.10.3.041.

[24] S. Weißleder and D. Sokenou, "Cause-effect graphs for test models based on UML and OCL," *SoftwareTechnik-Trends*, vol. 28, 2008.

[25] H. S. Son, Y. B. Park and R. Y. C. Kim, "Test case generation from cause-effect graph based on model transformation," in *2014 International Conference on Information Science & Applications (ICISA)*, Seoul, 2014. doi: 10.1109/ICISA.2014.6847468.

[26] T. Ayav and F. Belli, "Boolean differentiation for formalizing Myers' cause-effect graph testing technique," in *2015 IEEE International Conference on Software Quality, Reliability and Security - Companion*, Vancouver, 2015. doi: 10.1109/QRS-C.2015.31.

[27] I.-Y. Song, M. Evans and E. Park, "A comparative analysis of entity-relationship diagrams," *Journal of Computer and Software Engineering*, vol. 3, no. 4, pp. 427-459, 1995.

[28] A. Paradkar, K. C. Tai and M. A. Vouk, "Specification-based testing using cause-effect graphs," *Annals of Software Engineering*, vol. 4, no. 1, pp. 133-157, 1997. doi: 10.1023/A:1018979130614.

[29] D. K. Ufuktepe, "TOUCH: Test generator for cause effect graphs," [Online]. Available: https://github.com/denizkavzak/TOUCH. [Accessed 3 August 2022].

[30] W. S. Jang and Y. C. Kim, "Automatic cause-effect graph tool with informal Korean requirement specifications," *Applied Sciences*, vol. 12, 2022. doi: 10.3390/app12189310.

[31] *The BenderRBT Cause-Effect Graphing User Manual*, 3rd ed., Queensbury: Bender RBT Inc., 2006.

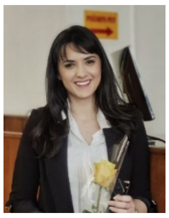
[32] M. Freiberg and J. Baumeister, "A survey on usability evaluation techniques and an analysis of their actual application," University of Würzburg, Würzburg, 2008.

[33] A. Hinderks, D. Winter, M. Schrepp and J. Thomaschewski, "Applicability of user experience and usability questionnaires," *Journal of Universal Computer Science*, vol. 25, no. 13, pp. 1717-1735, 2019.

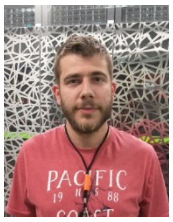
[34] M. Ulan, W. Löwe, M. Ericsson and A. Wingkvist, "Copula-based software metrics aggregation," *Software Quality Journal*, vol. 29, pp. 863-899, 2021. doi: 10.1007/s11219-021-09568-9.

[35] C. Burnay, S. Bouraga, J. Gillain and I. J. Jureta, "What lies behind requirements? A quality assessment of statement grounds in requirements elicitation," *Software Quality Journal*, vol. 28, pp. 1615-1643, 2020. doi: 10.1007/s11219-020-09521-2.

Ehlimana Krupalija received her B.Sc. and M.Sc. degrees in 2018 and 2020 at the Department of Computer Science and Informatics at the Faculty of Electrical Engineering of the University of Sarajevo. She is currently a teaching assistant and Ph.D. candidate at the Department of Computer Science and Informatics of the Faculty of Electrical Engineering, University of Sarajevo, Bosnia and Herzegovina. Her research interests include software quality, real-time systems, parallelization and optimization techniques.

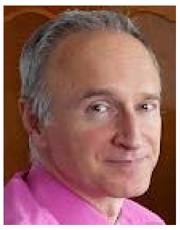


Šeila Bećirović received her B.Sc. and M.Sc. degrees in 2017 and 2019 at the Department of Computer Science and Informatics at the Faculty of Electrical Engineering of the University of Sarajevo. She is currently a teaching assistant and Ph.D. candidate at the Department of Computer Science and Informatics of the Faculty of Electrical Engineering, University of Sarajevo, Bosnia and Herzegovina. Her research interests include computer networks and security, mobile application development and operational research.



Irfan Prazina received his B.Sc. and M.Sc. degrees in 2013 and 2015 at the Department of Computer Science and Informatics at the Faculty of Electrical Engineering of the University of Sarajevo. He is currently a senior teaching assistant and Ph.D. candidate at the Department of Computer Science and Informatics of the Faculty of Electrical Engineering, University of Sarajevo, Bosnia and Herzegovina. His research interests include web technologies, software testing and mobile application development.

Emir Cogo received his B.Sc. and M.Sc. degrees in 2011 and 2013 at the Department of Computer Science and Informatics at the Faculty of Electrical Engineering of the University of Sarajevo. He is currently a senior teaching assistant and Ph.D. candidate at the Department of Computer Science and Informatics of the Faculty of Electrical Engineering, University of Sarajevo, Bosnia and Herzegovina. His research interests include game development, computer graphics and procedural modeling.



Ingmar Bešić graduated with distinction in 2000 at the Department of Computer Science and Informatics of Faculty of Electrical Engineering of the University of Sarajevo. He received his M.Sc. degree in Software Engineering in 2004 from the Keble College at the University of Oxford. In 2016 he received his Ph.D. degree at the Faculty of Electrical Engineering of the University of Sarajevo. His research interests include computer vision, real-time systems, software engineering, artificial intelligence, bioinformatics, computer assisted design and manufacturing and 3D scanning. He is currently an associate professor at the Department of Computer Science and Informatics of the Faculty of Electrical Engineering, University of Sarajevo, Bosnia and Herzegovina.