Received 15 July 2023, accepted 23 August 2023, date of publication 30 August 2023, date of current version 5 September 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3310394

RESEARCH ARTICLE

The Automatic Definition of the Intuitive Linguistic Heuristics Set to Recognize the Elements of UML Analysis and Design Models in English

AYAD TAREQ IMAM
Faculty of Information Technology, Isra University, Amman 11622, Jordan
e-mail: alzobaydi_ayad@iu.edu.jo

ABSTRACT Elicitation of the elements of Unified Modelling Language (UML) analysis and design models from sentences written in scripted English is essential in the production of analysis and design models. The correct elicitation of these elements depends on the intuitive, manually defined set of linguistic heuristics, which is used to map a word in the sentence to its correct semantics in the domain of UML analysis and design models. This paper proposes a Genetic Algorithm-based classification rule discovery approach and a developed Enhanced Intuitive Linguistic Heuristics (EILH) dataset to automate the definition of the intuitive linguistic heuristics set to elicit five elements of UML analysis and design models from English sentences. These elements are the use case, the actor, the sender, the receiver, and the message. The automatically defined intuitive linguistic heuristics set was evaluated by developing an Artificial Neural Network (ANN) to recognize the elements of the UML analysis and design models using both manually defined and automatically defined sets. This comparison shows the superiority of the automatically defined set over the manually defined one.

INDEX TERMS Automatic programming, computer aided software engineering, decision support systems, feature extraction, knowledge discovery, requirements engineering, semantic search, semiotics.

I. INTRODUCTION

The identification or elicitation of the elements of the analysis and design models of an information system from sentences articulated in a natural language is considered an essential step in producing analysis and design models. For these models to be accurate and efficient, the elements that comprise them must be identified with great care, so that no element is lost, and the identified elements are correctly identified and categorized. The elicitation of elements from sentences written in natural language needs linguistic skills- precisely heuristics [1], [2], [3], [4].

The intuitive linguistic heuristics set for elicitation of the elements of the UML analysis and design models were manually introduced by Abbott [5] to determine the class, the object (instance of a class), and attribute semantics of various language constructs that are found in the requirements specification sentences, like *nouns, verbs, and adjectives*, in addition to the constraints, the operation, the inheritance, and the aggregation relationships [6].

However, since the requirements specification comes in a natural language form, ambiguity (in terms of multiple semantics of a language construct) is a natural feature found in the requirements. Moreover, Abbott's heuristics did not define more elements of the analysis model (other than the object, attribute, and class). Therefore, the linguistic heuristics set defined by Abbott became somehow an imprecise set resulting in a lengthy and repetitive identification process [6]. This is a true challenge for the developers, who should produce an accurate set of analysis model elements during

The associate editor coordinating the review of this manuscript and approving it for publication was Justin Zhang.

the identification process. Bruegge and Dutoit [6] proposed the meta-description of language constructs as an improved heuristics set to solve the above-mentioned criticisms. Examples of these meta-descriptions are the clarified terms used in the requirements description, real-world entities & activities, data resources, etc. However, these meta-descriptions seem hard and time-consuming to apply since they require an ongoing revision of the terms' meanings by both the developers and users. The use of verb classification and semantic role (thematic roles) as a heuristics set enhancing Abbott linguistics heuristics set was suggested by Imam and Alnsour [7], [8], whereas this suggested set of heuristics is simpler than the meta-description heuristics of Bruegge and Dutoit. Examples of semantic roles are agent, experiencer, instrument . . . etc. For more details about thematic roles, refer to Chapter 19 of Jurafsky and Martin [3].

This paper aims to automate the definition of a set of intuitive language heuristics to identify five elements of UML analysis and design models instead of the manual approach followed by Abbott, Bruegge and Dutoit [6] and Imam and Alnsour [7], [8] to define this set of heuristics. The proposed automatic approach was achieved by developing a new GA-based classification rule discovery approach and building the first Enhanced Intuitive Linguistic Heuristics (EILH) dataset intended for applications of automatic extraction of UML diagram elements from English-written sentences. In this paper, the UML modelling and analysis elements, which will be recognized based on the automatically defined EILH set, are the use case, actor, sender, receiver, and message.

Classification rules discovery is a data mining approach for extracting a set of rules from a labelled training dataset for discovering hidden relationships between different elements in a dataset. The symbolic method is the basic extraction method for classification rules. Despite its simplicity, symbolic technology produces a large number of classification rules, which in turn makes symbolic technology costly in terms of time. Recently, sort of Machine Learning (ML) techniques like decision tree, Artificial Neural Network (ANN), Support Vector Machine (SVM), and rough set in addition to Genetic Algorithm (GA), and Swarm Intelligence (SI) have been used for discovering classification rules. The techniques of both the symbolic approach and the ML approach are of different performance and complexity, and each one has its pros and cons [9], [10], [11]. However, in this paper, GA is used as a method for classification rule discovery.

The GA has been proposed as a classification method that can produce optimal classifiers and has been widely used in the field of producing understandable and interesting classification rules due to its powerful global search mechanism in a defined search space and its ability to manage attributes' interactions better than other greedy based induction algorithms. The work of Salma-Tuz-Jakirin et al. [12] to discover classification rules of Nursery, Car, Breast Cancer, Qualitative Bankruptcy, Iris, and Lenses datasets of the UCI machine learning repository dataset, the work of Al-Maqaleh and Shahbazkia [10] to discover classification rules of Adult, Heart disease, Iris, and Dermatology datasets of the UCI machine learning repository, the works of both Shi and Lei [13] and Pawar and Bichkar [14] for discovering classification rules for the intrusion detection systems; all represent cases prove the efficiency of using the GA as a method for classification rules discovery. Yet, no previous work was proposed to use the GA as a classification rule discoverer of the elements of the UML analysis and design models.

The structure of this paper encompasses 5 sections. Section II reports the previous related works and approaches to deal with the problem of the automatic extraction of the elements of the UML analysis and design model. The Proposed Classification Rule Discoverer is presented and explained in detail in Section III. The results and discussion are given in Section IV. Finally, Section V concludes the paper and reports the findings and suggests future work.

A. CLASSIFICATION RULE DISCOVERY

The discovery of classification rules is the aim of the predictive Data Mining (DM) approach, which encompasses dataset and ML algorithms to discover and define a classification (mapping) model between the predictor variables (termed as independent inputs) in the dataset and predicted class variables (termed as dependents outputs), even when that mapping function is very complex [15].

Classification rule discovery comes in one of two modes. The first mode is the complete classification, which is typically inappropriate to be applied to mining descriptive data due to the big size, hard understanding, and most importantly the unguaranteed overall classification accuracy of the resulting classification models. The second mode is partial classification (or nugget discovery), which aims to produce classification rules that have understandable, yet strong descriptions (properties, features, heuristics, or conditions) of a particular class [16]. This work concerns the partial classification mode of classification rules discovery.

The discovery of classification rules results in a classification rule(s) of a general form, which is ''If *a condition* THEN *conclusion*'', which encompasses the antecedent predicate that is used to specify the conditions (or descriptions of a class in a partial classification) and the consequent predicate that is used to define the concluded result (or a particular class name in partial classification) upon these conditions and it has the following form: $P \rightarrow Q$, where P is the antecedent and Q is the consequent [17]. Usually, the composing of the classification rule is achieved by manual and careful extraction of the antecedents and action of each production rule from a textual description written in a natural language [10], [12]. While all of them have the same generic structure, the condition part of the rule (antecedent) is different, giving the flexibility to form different expressions to be used with different computation types. Based on the type of the condition, the rules forms are [15]:

- The Classical propositional logic rules (C-rules), their condition type is of crisp value.
- The Fuzzy logic rules (F-rules), their condition type is real-valued resulting from membership functions.
- Classification rules (A-rules), their condition type is a belonging relationship of an element to a set
- Threshold rules (T-rules), their condition type is a belonging relationship of a subset to a whole set
- Prototype-based rules (P-rules), and their condition type is a similarity (or distance) measure between two cases.

The techniques that have been developed for achieving the partial classification rules discovery are the Modern Heuristic, Multi-Objective, and All-Rules Search [18]. Most rule-finding techniques work on the rule format. Using a more efficient rule-finding technique with highly flexible rule formats will result in more reliable class-determining rules being discovered, which means creating a large search space problem. To reduce the number of rules, while preserving the highly significant ones, two techniques can be used, which are the restrictions on the structure of the rules and the use of the minimum support and the minimum confidence thresholds, which are set by the user. Both methods exclude valuable classification rules and preserve weak ones, resulting in a poor class description [11], [19]. However, in this work, the Modern Heuristic technique is used among these three techniques as it is the best one for reducing the size of the search space.

Modern Heuristic technique utilizes optimization algorithms to discover the most interesting classification rules based on certain criteria like the GA, Particle Swarm Optimization (PSO), tabu search, and Ant Colony Optimization (ACO) optimization algorithms. Examples of utilizing these algorithms for discovering classification rules include the work of Azaryuon et al. [20], the work of Liu et al. [21], and the work of [22] that all utilized ACO to develop new versions of Ant-Miner, which is a classification rule discovery program. The modern heuristic approach makes the partial classification mode (of classification rules discovery) view the problem of finding strong descriptions of a class as an optimization problem. To realize this view, the proposed classification rules shape a search space for finding a class by using the conjunctive technique, which is optimized by selecting a subset of these rules based on a measurable evaluation like the fitness measure. The cons of a big number of production rules are opacity, inefficiency during execution, conflict resolution, and the absence of learning. Reducing the number of production rules by selecting the best-performing ones is a suitable solution for most of these problems except for the problem of the absence of learning [23]. A big number of production rules for the same action is usually happened due to the large diversity in the structures of the natural language and should be reduced by selecting the best ones among them- bearing in mind that manual selection is a difficult and complex task. The sets of rules resulting from the process of discovery of classification rules are valuable as long as their formulation is understandable, their results are perfectly accurate, and the number of such rules is not exceptionally large [15].

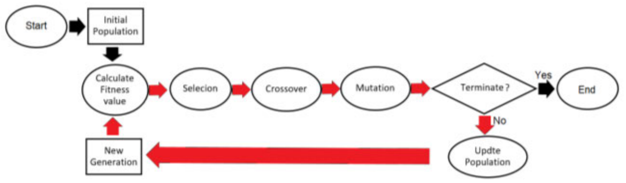


FIGURE 1. **The operations of a generic GA.**

The resulting set of classification rules from the discovery process creates a set of production rules, which is a subsystem of the Inference Engine (IE) computation model that imitates the human problem-solving ability. The production rule subsystem is of big interest because of the generalization of the 'if-then' structure that is used in most programming languages, and its simplicity since it is close to the natural language. Rule-based systems must reasonably reveal the hidden knowledge in the data, have justification(s) for the produced conclusions, show potential conflicts, and avoid unexpected conclusions produced by predictors in atypical conditions. Rules are used in sorts of computerized processing tasks including classification by using ML, regression by using statistics, and classification [24].

B. GENETIC ALGORITHM

It is a class of optimized search algorithms whose work takes inspiration from the natural gene system and was developed by John Holland et al. at the University of Michigan [11]. By using ranking values, GA performs a blind active search. As illustrated in Figure 1, a defined set of chromosomes, called the initial population, represents the domain of solutions that would be evolved by the GA's serial operations several times, under certain conditions, resulting in a generation of enhanced quality at each time. These operations are [25]:

- **Calculate the fitness value** for each chromosome in the population (generation)
- **The selection operation** picks the highest rank chromosomes, according to a user-defined fitness function, are selected; and the rest chromosomes are removed from the population.
- **The crossover operation** selects and pairs a couple(s) of chromosomes resulting from the selection operation, by partially swapping each couple's genes around one or more arbitrarily pointed crossing points to generate one offspring.
- **The mutation operation** selects a particular number of chromosomes and modifies the value of a randomly selected gene of a chromosome abruptly.
- **Update population**by adding newly generated offspring.

The representation (encoding) of the chromosomes, the choice of a fitness function, in addition to the GA parameter values are the three factors that affect the GA effectiveness to solve the problem. The type of application guides the determination of the GA factors' values [10].

Worth mentioning Genetic Programming (GP), which is an extended GA whose method of encoding chromosomes is achieved via using a parse tree to represent very complex chromosomes instead of using fixed length vectors as done in the GA [24]. This flexibility in the representation makes GP fit to derive more accurate or complex classification rules compared to GA, which is used to well derive simple classification rules [9].

II. PREVIOUS RELATED WORKS FOR EXTRACTING THE ELEMENTS OF ANALYSIS AND DESIGN MODELS

Thematic roles technique along with the SVM approach of ML proposed by Imam et al. [8] as a named entity recognition (NER) system to extract the system, actor, and use case elements from unstructured English descriptions of user requirements of the software. This work officially was the first specialized NER that targeted software requirements descriptions written in the English language, and the first use of SVM for specifying the semantics of words under the software requirements domain of discourse. The performance of this suggested NER was 72.1% for the F-measure, 76% for recall, and 76.2% for precision. The thematic roles technique was also used by Jebril et al. [26] who proposed a manual algorithmic approach to identify the actors and their use cases from a user requirements description written in the English language. These two works were the first-ever ones that used thematic roles for identifying the elements used in the analysis models of software requirements engineering.

The domain ontology technique (the ancestor of NER) is a classification approach that necessitates the definition of a specific domain semantic and has no need for linguistics semantics. Murtaza et al. [27] proposed the domain ontology technique as a Case-Based Reasoning (CBR) system to convert the collected requirements formed using textual natural language into a structured natural language. Moreover, More and Phalnikar [28] utilized the Domain ontology technique as a part of the RAPID tool that uses textual requirements to produce Unified Modelling Language (UML) diagrams.

ANN was proposed by Alhroob et al. [29] to automatically extract the actors and actions from a user requirements description, which is written in the English language. The distinguished point in this work is the way that ANN was used to get a definitive identification of the actors and actions using intuitive linguistic tags. The performance of this suggested ANN for the five test cases was ranging from 17 % to 63 % for precision, from 56 % to 100 % for recall, and from 29 % to 71 % for F-measure.

Key-Word-In-Context (KWIC) feature was used by Deeptimahanti et al. [30] to recognize particular elements in the requirement document as a part of the proposed UMGAR system for generating use-case diagrams. Worth mentioning that the mapping technique was used earlier by using older AI techniques like the work of Imam et al. [31]. In this work, which was part of the development of the Arabic Natural Language Interface to Robot, the keywords technique was used to recognize the semantics of an issued natural language-formed command (imperative) sentence to a robot.

The Part-Of-Speech (POS) tag technique used was proposed to support the parsing process to extract phrases, activities, etc. from an entered textual requirement written in the English language by Gulia and Choudhury [32], and proposed by Mohanan and Samuel [33] to support Semantic Business Vocabulary (SBV) and rules that process textual requirement and business designs written in the English language to produce object-oriented models.

Synthesizing rules based on previous examples is an approach proposed by Georgiades and Andreou [34], for formalizing certain elements and activities of the use case modeling process. This is not an analyzing process of current text written in natural language as other approaches follow.

The above-reported previous works and approaches show that no outdated or updated research works proposed the automation of the definition of intuitive linguistic heuristics for identifying the elements of UML analysis and design models by using the GA technique.

III. THE PROPOSED CLASSIFICATION RULE DISCOVERER

Figure 2 illustrates the proposed classification rules discoverer approach that has been followed to extract a set of classification rules to recognize the elements of the UML's analysis and design models of an information system in sentences written in English. The Elements that are targeted to be recognized by the discovered classification rules are the use case, actor, sender, receiver, and message. The proposed approach encompasses the development of a first-time defined EILH dataset, and the development of a GA-based classification rule discoverer as will be elaborated in the following subparagraphs.

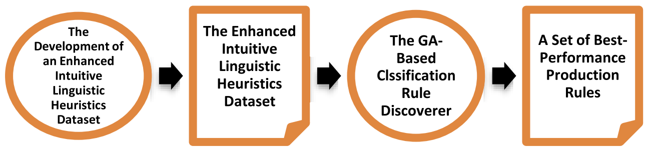

The Development of an Enhanced Intuitive Linguistic Heuristics Dataset
The Enhanced Intuitive Linguistic Heuristics Dataset
The GA-Based Clssification Rule Discoverer
A Set of Best-Performance Production Rules

FIGURE 2. **Flow diagram of the proposed classification rules discoverer approach.**

A. THE DEVELOPMENT OF AN ENHANCED INTUITIVE LINGUISTIC HEURISTICS DATASET

The use of ML approaches for information processing requires the use of a dataset. Many researchers and research centres produced datasets for all sorts of applications like intrusion detection systems, sentiment analysis, and others. Unfortunately, there is no dataset for the application of automatic extraction of the elements of UML diagrams from sentences written in the English language. This fact motivated the production of the EILH dataset to be used in this research work and similar future works. However, in this work, the

TABLE 1. Semantic contentions of an individual verb.

Frameset	Arg0	Arg1	Arg2	Arg3
accept.01	Acceptor	Thing accepted	Accepted from	Attribute

proposed modified GA used the EILH dataset firstly for distinguishing five elements of UML diagrams, and secondly to compose the 'if-then' clauses.

The number of English sentences (software requirement sentences and flow of event sentences) that were used to develop the EILH data set was 2330. The sentences were selected to cover the five elements and were divided into two separate groups as illustrated in Figure 3, namely:

- The training data set: 2/3 of the entire sentence (1400 sentences) to generate best-performance production rules.
- The test data set: 1/3 of the entire (930) sentences to measure the accuracy of the generated production rules.

Based on that, five data subsets have been made: the use case data subset of size 2025, the actor data subset of size 3223, the sender data subset of size 179, the receiver data subset of size 225, the message data subset of size 54, and 1206 for unclassified words. These data subsets were used as the initial five populations to be used five times by the modified GA to obtain the best-attributed description of each of the five elements of the UML diagrams.

The linguistic heuristics (features) had been extracted from English sentences by linguistically analyzing these sentences, which was achieved automatically by using Semantic Role Labelling Demo software. This software tool is an application hosted by the Cognitive Cognition group at the University of Illinois at Urbana-Champaign to perform tokenizing, lexical annotation, syntax annotation, and semantic annotation of each tokenized word by using two linguistic banks, which are [35]:

Proposition Bank (PropBank): considers the structure of the argument procedure to provide a full corpus with semantic roles, including roles that are traditionally seen as arguments and as assistants. It allows the determination of the actual frequency of changes in syntax to determine the problems and strategies involved in understanding natural language. This bank characterizes semantic roles on a verb-by-verb basis. As shown in Table 1, the semantic contentions of an individual verb are coded as Arg and are numbered starting with zero [36].

Noun Bank (NomBank): provides a case structure for instances of more than 5,000 nouns in the Penn Treebank II corpus. It forms a small part of wider attempts to add an annotation to some layers of the Penn Treebank II corpus. In this bank, the subject of a verb (the agent's thematic role) is coded as [A0], the object of a verb (theme or patient) as [A1], and the indirect object as [A2]. AM-LOC represents the location of the event, [VB] represents the action or event, and [SUP] represents the support verb [37].

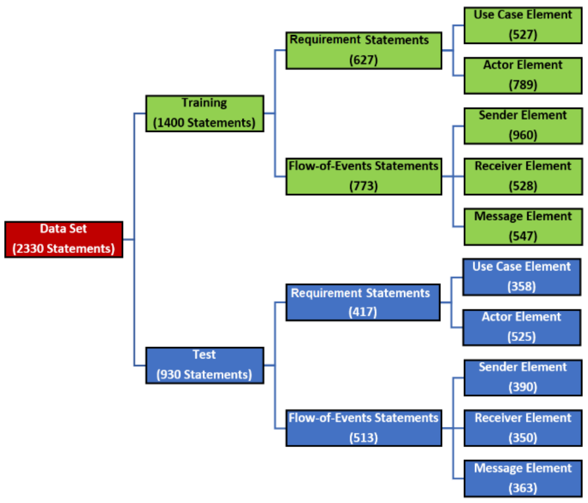
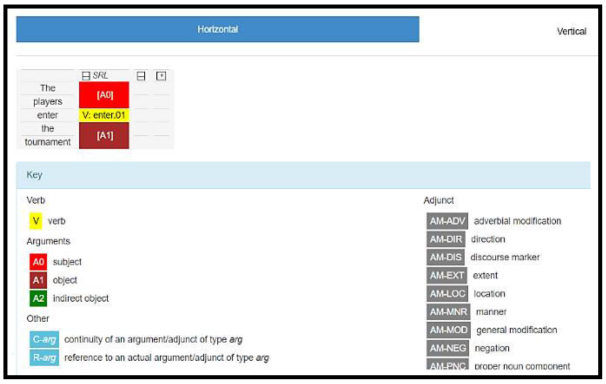


FIGURE 3. The hierarchy of the types of sentences used to develop EILH data set.



	SRL
The players	[A0]
enter	V: enter.01
the tournament	[A1]

Key

Verb

V	verb

Arguments

A0	subject
A1	object
A2	indirect object

Other

C-arg	continuity of an argument/adjunct of type *arg*
R-arg	reference to an actual argument/adjunct of type *arg*

Adjunct

AM-ADV	adverbial modification
AM-DIR	direction
AM-DIS	discourse marker
AM-EXT	extent
AM-LOC	location
AM-MNR	manner
AM-MOD	general modification
AM-NEG	negation
AM-PNC	proper noun component

FIGURE 4. Sample run of an SRL [35].

Tokenisation was the first process applied to each sentence, followed by other lexical and semantic processes. Note that all these processes are executed automatically once a sentence is given to SRL software as shown in Figure 4 illustrates for the sentence: 'The players enter the tournament. The resulting linguistic heuristics (features) of each word fall into three linguistic types, namely:

The source sentence: the source of the word either from the requirement sentence or from the flow of the event sentence.

Lexical features: the lexical type of the word either noun, verb, or preposition.

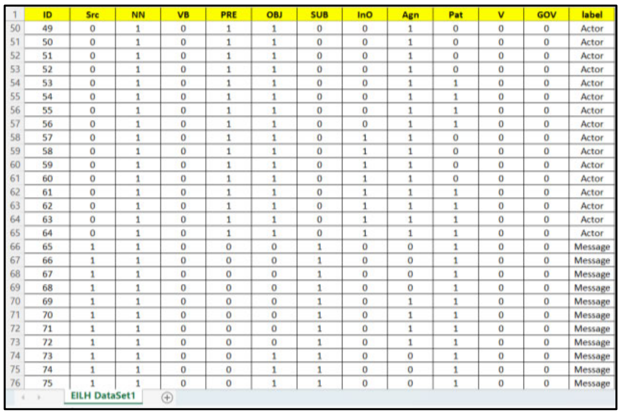
Syntax features: the arguments' identity of the verb, which could be an object, subject, and Indirect Object.

Semantic features: represent the thematic role of each word, which could be agent, patient, Action (or Event), and Governor.

The possible values of the extracted linguistic features were numerically represented, in which the feature was represented as 1 if its linguistic property exists in the word and represented as 0 otherwise. Table 2 illustrates the possible coding values for each feature in the EILH dataset.

TABLE 2. Codes used by the codify method.

Linguistic analysis level	Property	Code
SRC	Requirement or Flow of Event	0 or 1
Lexical Attributes	Noun (NN)	0 or 1
	Verb (VB)	0 or 1
	Preposition	0 or 1
Syntax Attributes	Object (OBJ)	0 or 1
	Subject (SUB)	0 or 1
	The indirect object (A2)	0 or 1
Semantic (Thematic Roles) Attributes	Agent (A0)	0 or 1
	Patient (A1)	0 or 1
	Action or event (V)	0 or 1
	Governor (GOV)	0 or 1



1	ID	Src	NN	VB	PRE	OBJ	SUB	InO	Agn	Pat	V	GOV	label
50	49	0	1	0	1	1	0	0	1	0	0	0	Actor
51	50	0	1	0	1	1	0	0	1	0	0	0	Actor
52	51	0	1	0	1	1	0	0	1	0	0	0	Actor
53	52	0	1	0	1	1	0	0	1	0	0	0	Actor
54	53	0	1	0	1	1	0	0	1	1	0	0	Actor
55	54	0	1	0	1	1	0	0	1	1	0	0	Actor
56	55	0	1	0	1	1	0	0	1	1	0	0	Actor
57	56	0	1	0	1	1	0	0	1	1	0	0	Actor
58	57	0	1	0	1	1	0	1	1	0	0	0	Actor
59	58	0	1	0	1	1	0	1	1	0	0	0	Actor
60	59	0	1	0	1	1	0	1	1	0	0	0	Actor
61	60	0	1	0	1	1	0	1	1	0	0	0	Actor
62	61	0	1	0	1	1	0	1	1	1	0	0	Actor
63	62	0	1	0	1	1	0	1	1	1	0	0	Actor
64	63	0	1	0	1	1	0	1	1	1	0	0	Actor
65	64	0	1	0	1	1	0	1	1	1	0	0	Actor
66	65	1	1	0	0	0	1	0	0	1	0	0	Message
67	66	1	1	0	0	0	1	0	0	1	0	0	Message
68	67	1	1	0	0	0	1	0	0	1	0	0	Message
69	68	1	1	0	0	0	1	0	0	1	0	0	Message
70	69	1	1	0	0	0	1	0	1	1	0	0	Message
71	70	1	1	0	0	0	1	0	1	1	0	0	Message
72	71	1	1	0	0	0	1	0	1	1	0	0	Message
73	72	1	1	0	0	0	1	0	1	1	0	0	Message
74	73	1	1	0	0	1	1	0	0	1	0	0	Message
75	74	1	1	0	0	1	1	0	0	1	0	0	Message
76	75	1	1	0	0	1	1	0	0	1	0	0	Message

EILH DataSet1

FIGURE 5. A screenshot of the EILH dataset file.

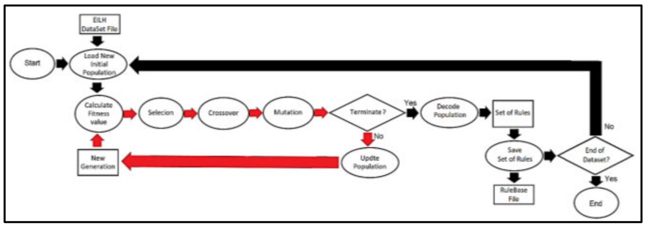


FIGURE 6. Flowchart of the GA-Based rule set generator.

To annotate each word (token) saved in the 'EILH DataSet' Excel file (of.CSV extension), the developed 'Codify' procedure was used to accomplish the task of features' coding. The pseudocode for this method is given in Algorithm 1.

Each sentence contains a different number of targeted elements depending on the way the sentence was formed (direct/indirect speech, for example. However, EILH consists of 2046 records, each record encompasses 11 features (linguistic heuristics) that are used for featuring an element of the UML analysis and design models, which are the use case, actor, sender, receiver, and message. These elements were used for labeling the features of the EILH dataset.

Finally, the EILH data set is currently under careful study and review to make it conform to the standards of a dataset that can be used with different ML techniques. The study includes the major properties of a dataset like being a balanced dataset, statistical studies, and many other processes. Once this study would complete, EILH will be published for public use. Figure 5 is a screenshot of the current version of EILH used in this research work.

Algorithm 1 Codify Procedure (Token, Initial Population)

```
Begin
  For each feature
    If the token's property exists, then set its feature
    field to 1
    Else set its feature field to 0
    End If
  End For
  Save record of features in its initial population
End.
```

Algorithm 2 FitnessValue Procedure (Chromosome, Integer [ ])

```
Begin
For each chromosome in the population
  D = The size of the training data set
  ncovers = No. of instances covered by R
  ncorrect = No. of instances correctly classified by R
  Cov = ncovers / D
  Acc = ncorrect / ncovers
  Chromosome's FitnessValue = Acc+ Cov
End For
End.
```

TABLE 3. Chromosome's structure.

SRC	NN	VB	PRE.	OBJ	SUB	A2	A0	A1	V	GOV

B. THE GA-BASED CLASSIFICATION RULE DISCOVERER

GA in this work was implemented with certain modifications to compose a set of "If-Then" production rules as a step for the final goal, which is the discovery of the features (antecedents of the rules) of each element of the UML analysis and design models.

As illustrated in Figure 6, the modifications included the repetition of the genetic process five times for the five EILH data subsets and the composing of a set of rules by a decoding step. The steps of the proposed modified GA, which has been implemented by using the C# programming language, are:

a) Load New Initial Population: This step aims to load a population from the EILH dataset file. A population is a set of chromosomes that takes their values from records in the EILH data subset. As illustrated in Table 3, each chromosome encompasses 10 genes that represent possible alternative values of the ancestors (or features as termed by ML) of a production rule that are used to recognize the elements of the UML analysis and design models.

Algorithm 3 Selection Procedure (Population)

```
Begin
    Sort the population in descending order based on the
    chromosomes' fitness values
    Delete the last chromosomes from the population whose
    fitness value is less than a threshold
End.
```

Algorithm 4 CrossOver Procedure (Chrom1, Chrom2)

```
Begin
        FrstHlfChro1 = Copy 1st Half of Chrom1
        ScndHlfChro1 = Copy 2nd Half of Chrom1
        FrstHlfChro2 = Copy 1st Half of Chrom2
        ScndHlfChro2 = Copy 2nd Half of Chromo2
        NChrom1 = Concat (FrstHlfChro1, ScndHlfChro2)
        NChrom2 = Concat (FrstHalfChro2,
        ScndHalfChro1)
        Add new NChrom1 and NChrom2 to the population
End
```

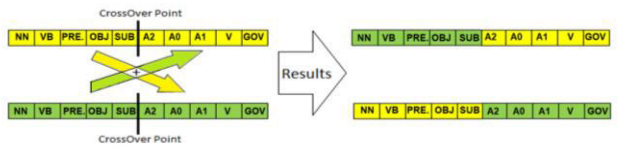

CrossOver Point
NN VB PRE. OBJ SUB A2 A0 A1 V GOV
NN VB PRE. OBJ SUB A2 A0 A1 V GOV
CrossOver Point
Results
NN VB PRE. OBJ SUB A2 A0 A1 V GOV
NN VB PRE. OBJ SUB A2 A0 A1 V GOV

FIGURE 7. Crossover in GA.

b) Calculate the Fitness Value: The fitness values are used to rank chromosomes in the population space to select the ones that will be reproduced and passed on to the new generation. There is no unique fitness function for the GA; instead, each problem would have its fitness function that reflects the best quality of the solution to be founded by the GA for that problem [25]. Considering the problem defined in this work, which is a classification problem, the fitness function was based on the accuracy and coverage of the discovered rule (R), as follows [10], [11]:

- *Coverage*: is the percentage of the classification of class (C) to the whole instances in the training dataset (D), covered by a classification rule. It is interpreted as:

$$\mathrm{Cov} = |C\wedge D|/D \tag{1}$$

- *Accuracy*: the percentage of the correct classification of a class (CC) to the whole classification of a class (C) in the training dataset (D), covered by a classification rule. It is interpreted as:

$$\mathrm{Acc}(R) = |CC\wedge C|/\ C \tag{2}$$

The chromosome's fitness value is the sum of conf (R) and cove (R). These formulas were used to develop the FitnessValue procedure as part of the implementation of the modified GA as shown in Algorithm 2.

Algorithm 5 Decoding Procedure (Chromosome)

```
Begin
    For all chromosomes in the population
    Antecedent=""
    For each gene in the chromosome
        If gene == 1 then
            concat the Antecedent + feature's name Based on
            Table 2 + word "and",
        EndIf
    EnfFor
    Remove the last "and" word from the Antecedent.
    Concat the word "IF" + Antecedent + with the word
    "Then" + population name (label or class)
    Write the composed sentence into Rulebase File
EndFor
End.
```

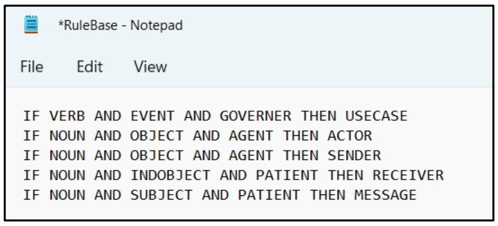
c) Selection: aims to choose the fittest chromosome and drop the weaker ones from the population based on the fitness value of each chromosome. The developed pseudocode for the selection process is illustrated by Algorithm 3.

d) Crossover: swapping between two chromosomes' genes at a randomly selected crossover point [25]. The single-point crossover method, which is the border between gene 5 and gene 6, to make a mutual exchange, has been used here. Crossover, which is illustrated in Figure 7, is used to realize the evolutionary property of the GA, which allows for improvements in the population.
The above description of the crossover process was implemented as a method called **CrossOver**. The pseudocode for the **CrossOver** method is given by Algorithm 4.

e) Mutation: involves internal chromosome swapping within genes. Usually, the mutation rate is kept low to avoid corrupting any improvements that have been made in the previous generations [25]. The mutation process had not been applied because the reduction of the complexity that may arise with the generation of the new population (new generation) was an aim.

f) Termination: This is the condition to terminate the evolution process of the GA. The termination condition could be one of three alternatives: reaching a predefined number of generations, reaching a predefined fitness value, or a lack of improvement in the population over a predefined number of iterations [25]. The number of generations (evolved populations) has been used and set to 100 as a termination condition.

g) Decode Population: The ultimately resulting population contains evolved chromosomes, where the best 4 chromosomes from each population (of the use case, actor, sender, receiver, or message) were selected as the best solution for their class. These chromosomes were decoded using the decoding method, which converts



```
IF VERB AND EVENT AND GOVERNER THEN USECASE
IF NOUN AND OBJECT AND AGENT THEN ACTOR
IF NOUN AND OBJECT AND AGENT THEN SENDER
IF NOUN AND INDOBJECT AND PATIENT THEN RECEIVER
IF NOUN AND SUBJECT AND PATIENT THEN MESSAGE
```

FIGURE 8. A screenshot of the textual form of the resulting classification rules.

TABLE 4. The automatically defined set of heuristics for each element.

	USE CASE	ACTOR	SENDER	RECEIVER	MESSAGE
SRC	0	0	1	1	1
Noun (NN)	0	1	1	1	1
Verb (VB)	1	0	0	0	0
Preposition (PRE.)	0	0	0	0	0
Object (OBJ)	0	1	1	0	0
Subject (SUB)	0	0	0	0	1
Indirect object (A2)	0	0	0	1	0
Agent (A0)	0	1	1	0	0
Patient (A1)	0	0	0	1	1
Action or event (V)	1	0	0	0	0
Governor (GOV)	1	0	0	0	0

the chromosomes (coded form) into an explicit textual form as a set of 'If-Then' sentences and saves them in a Rulebase file. Note that since the value of each feature is either 0 or 1, the relationship in a condition is limited to '=' or is a logical value (true or false), which simplifies the composing of the antecedents of the "IF-THEN" sentence. The Decoding procedure represents the actual composer of the set of production rules and its pseudocode is illustrated by Algorithm 5.

C. SET OF PRODUCTION RULES

This is the generated set of classification rules for recognizing five elements of UML analysis and design models from sentences written in the English natural language. This automatically discovered a set of heuristics (or features), which were saved in the 'RuleBase' text file encompassing lexical, syntax, and semantic (thematic roles) types resulting from natural language processing (NLP).

Figure 8 is a screenshot of the textual form of the resulting classification rules, and the automatically resulting set of heuristics for each element of these five elements is illustrated in Table 4.

IV. RESULTS AND DISCUSSION

In this paper, the proposed GA-based classification rules discovery approach was used to discover an optimized set of classification rules as a step to automate the definition of the enhanced intuitive linguistic heuristics set that would be used to recognize five elements of UML analysis and design models.

Recalling that the manually defined set of each element of the five elements of the UML analysis and design model is [1] and [26]:

- Use-case: It is a service (verb or predicate) that is performed by a system.
- Actor: It is a person or another system (noun and beneficiary of the use case) that undergoes interaction(s) with a system to gain benefit(s).
- Message: It is, a predicate or a verb name that sends from a sender to a receiver
- Sender: It is a noun representing the agent of a predicate.
- The receiver: It is a noun that represents the patient, theme, benefactors, or recipient of the message.

The automatically defined set of heuristics for each of these elements not only corresponds to the theoretical definitions of these elements but also details linguistic features not contained in the theoretical definitions. For example, according to Table 4, a use case is a verb, action or event, and governor, which is a detailed description of the above theoretical linguistic description of the use case. The same thing is noted about the other four elements.

The set of production rules, as well as the intuitive linguistic heuristics (the antecedent of each classification rule) that feature each element, had been assessed by using the test data part of the EILH dataset. The binomial approach has been used manually to assess both the resulting set of classification rules and automatically to assess the resulting set of heuristics because the binomial approach uses human performance as a comparison benchmark, and the difficulty of making a comparison with the performance of some previously reported related work that was manually developed because the quantitative evaluation information for those works is either missing or the evaluation approaches used are not the same for all works.

The assessment of a classification model is expressed by its accuracy, which is defined as the percentage of the number of cases that are correctly classified to the entire classified cases number, and it is described by the following formula [38]:

$$\text{Accuracy} = \text{No. of cases that correctly classified} \\ /\text{No. of entire classified cases} \qquad (3)$$

However, the reported shortcomings of this naive definition of accuracy include the mishandling of the different types of errors and the dependence of the evaluation of accuracy on the class distribution in the dataset [25]. The types of wrong classification results are distinguished using a two-dimensional confusion matrix technique, where the rows of the matrix represent the number of predicted classes resulting from the classification model, and the columns represent the number of actual classes reported by a human. While the use of a confusion matrix allows for better analysis of errors by identifying the types of errors, the increase in the size of the confusion matrix as more classes are involved in the classification poses a challenge and requires a solution that gives a stable size for the confusion matrix. This challenge is solved with the

TABLE 5. Binary values-based confusion matrix.

		Manually Classified Class	
		P	N
Predicted Class by the classification approach	T	TP	FP
	F	TN	FN

approach used to measure the performance of a classification model on a test dataset. As illustrated in Table 5, the results of a classification model used by a confusion matrix are classified into four classes, which are [39] and [40]:

- True Positive (TP): the number of cases classified by the classification approach (predicted cases) as True out of the number of the cases manually defined as Positive or True (actual cases)
- False Positive (FP): the number of cases classified by the classification approach (predicted cases) as True out of the number of the cases manually defined as Negative or False (actual cases)
- True Negative (TN): the number of misclassified cases by the classification approach (predicted cases) out of the number of the cases manually defined as True (actual cases)
- False Negative (FN): the number of misclassified cases by the classification approach (predicted cases) out of the number of the cases manually defined as False (actual cases)

The recorded values of these classes are used to calculate the rates used to quantify the precision and accuracy of the classification model, as follows [39]:

- True positive rate (Recall): The percentage of correctly classified cases as positive, i.e.:

$$\text{True positive rate} = TP/(TP + FN) \tag{4}$$

- False Positive rate: The percentage of incorrectly classified cases, i.e.:

$$\text{False positive rate} = FP/(FP + TN) \tag{5}$$

- True Negative rate: The percentage of cases that are correctly classified as negative, i.e.:

$$\text{True negative rate} = TN/(TN + FP) \tag{6}$$

- False Negative rate: The percentage of cases that are incorrectly classified as negative, i.e.:

$$\text{False negative rate} = FN/(FN + TP) \tag{7}$$

- Positive predictive value (Precision): The percentage of correctly classified positive cases, i.e.:

$$\text{Precision} = TP/(TN + TP) \tag{8}$$

TABLE 6. Evaluation confusion matrix of the generated set of classification rules.

	UseCase	Actor	Sender	Receiver	Message	Average	Highest
TP	311	421	333	291	331	337.4	421
FP	17	34	23	17	5	19.2	34
TN	13	55	18	37	23	29.2	55
FN	17	15	16	5	4	11.4	17
Sum	358	525	390	350	363	-	-
FPRate (%)	56.67	38.20	56.10	31.48	17.86	40.06	56.67
TNRate (%)	43.33	61.80	43.90	68.52	82.14	59.94	82.14
FNRAte (%)	5.18	3.44	4.58	1.69	1.19	3.22	5.18
Recall or TPRate (%)	94.82	96.56	95.42	98.31	98.81	96.78	98.81
Precision (%)	95.99	88.45	94.87	88.72	93.50	92.31	95.99
Accuracy (%)	90.50	90.67	90.00	93.71	97.52	92.48	97.52
F1 (%)	94.82	94.50	94.47	96.36	98.66	95.76	98.66

- Accuracy: The percentage of true results, which includes both true positive and true negative values in the confusion matrix, relative to the total number of cases examined, i.e.:

$$\text{Accuracy} = (TP + TN)/(TP + FN + FP + TN) \tag{9}$$

- F1 measure: The harmonic mean of recall and precision, i.e.:

$$F1 = 2^{*}TP/(2^{*}TP + FP + FN) \tag{10}$$

Table 6 shows the number of recorded classifications of user cases, actors, sender, receiver, and message items as a confusion matrix in terms of recall, precision, accuracy, and F1 (harmonic average of recall and accuracy) of the generated set of classification rules.

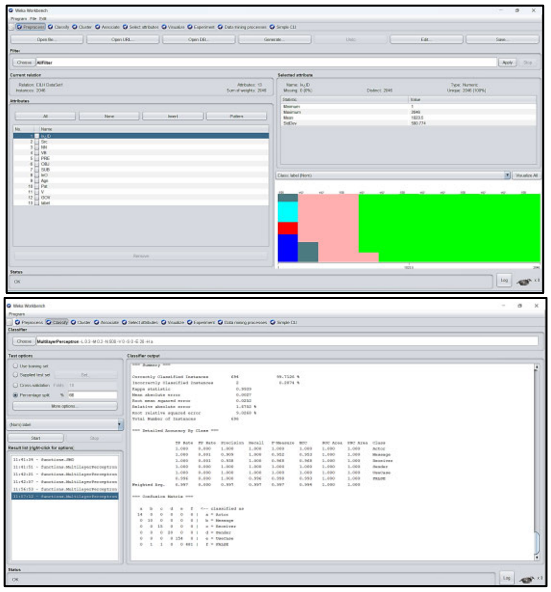
The second assessment aimed to compare the automatically generated intuitive linguistic heuristics set (features) with the manually defined one by utilizing ANN offered by Weka data mining software [41].

The work proposed by Alhroob et al. [29] that used ANN with the manually defined set of heuristics reported the performance of this suggested ANN for five test cases, which ranges from 17-63 % for precision, from 56-100% for recall, and from 29-71% for F-measure. However, an ANN classification model (whose options - L 0 3 – M 0 2 – N 500 -V 0 -S 0 – E 20 -H a) was created by using the 11 attributes in the EILH DataSet relation (all but the ID and label attributes). The results showed that the automatically defined set that is revealed by this work is superior to the manually defined one as shown in Table 7, which has been calculated automatically by Weka as shown in Figure 9.

Why there is no comparison of the proposed GA approach with other automated feature selection techniques on this problem? As this paper delivers for the first time an automatic defining of intuitive linguistic heuristics (features) to recognize the elements of UML analysis and design models in English sentences, such a comparison cannot be made because no previous work for automatic defining

TABLE 7. Evaluation confusion matrix of the generated set of heuristics used by ANN.

	Use Case	Actor	Mes-sage	Rece-iver	Send-er	False	Aver-age
Recall or TPRate (%)	100	100	100	100	100	99.6	99.7
Precision (%)	100	100	90.9	93.8	100	100	99.7
F1 (%)	100	100	95.2	96.8	100	99.8	99.7



```
=== Summary ===

Correctly Classified Instances         694               99.7126 %
Incorrectly Classified Instances         2                0.2874 %
Kappa statistic                          0.9939
Mean absolute error                      0.0027
Root mean squared error                  0.0252
Relative absolute error                  1.0752 %
Root relative squared error              9.0260 %
Total Number of Instances              696

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Actor
                 1.000    0.001    0.909      1.000    0.952      0.953    1.000     1.000     Message
                 1.000    0.001    0.938      1.000    0.968      0.968    1.000     1.000     Receiver
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Sender
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     UseCase
                 0.996    0.000    1.000      0.996    0.998      0.993    1.000     1.000     FALSE
Weighted Avg.    0.997    0.000    0.997      0.997    0.997      0.994    1.000     1.000

=== Confusion Matrix ===

   a   b   c   d   e   f   <-- classified as
  14   0   0   0   0   0 |   a = Actor
   0  10   0   0   0   0 |   b = Message
   0   0  15   0   0   0 |   c = Receiver
   0   0   0  20   0   0 |   d = Sender
   0   0   0   0 154   0 |   e = UseCase
   0   1   1   0   0 481 |   f = FALSE
```

FIGURE 9. A screenshot of the ANN results using automatically defined heuristics sets.

of intuitive linguistic heuristics (features) to recognize the elements of UML analysis and design models in sentences of any natural language.

Why automate the definition of intuitive linguistic heuristics? Knowing inference is an essential step to beginning the process of extracting the elements of UML analysis and design models even if the extraction is done manually. Automatic detection of elements' features (or heuristics) from a set of candidate features can be generalized to other applications, making the development of a dataset for any application a more compact process.

Why the classification rule discovery approach? Using the classification rule discovery approach will result in the detection (selection) of heuristics (features), which are a subset of a bigger set of candidate attributes. This approach can create clear and accurate classification (if-then) rules, although the resulting rules follow the greedy technique (one classification attribute at a time) that chains a sequence of related classification attributes and the large neglect of the interactions among attributes that may result in a suboptimal classification model [9], [42].

Why GA among other optimization algorithms? GA is a single population, that determines closer values to predefined values than other optimization algorithms do and seems to reach its ultimate value in a smaller number of generations, thus, faster than other optimization algorithms in discrete optimization problems [43].

Can the classification rule discovery approach be considered an Automatic Source Code Generator (ASCG)? ASCG is an essential part of the software development lifecycle, and it aims to improve the quality and accuracy of the resulting code, simplify maintenance, and reduce the overall software development time. Therefore, ASCGs are widely used nowadays, as seen in lots of ASCG software tools [44] like the source code generator of the user interface, the generating of machine code, and many other components [1]. There are two types of ASCG, the first type is the active code generator, which converts a model into another form that cannot be changed by the developer like a compiler [45]. The second type is the passive code generator, which is used only one time to generate a code that needs to be reviewed by the developer for improvement or completion purposes. Because the passive code generator represents the top experience of software engineering, it generates high-quality code [46]. The fact that the software development process itself is an intellectual task in principle. These features have directed the researchers to depend on Artificial Intelligence (AI) techniques that support template or syntax generator techniques to computerize ASCG software tools [47], [48]. In this work, the resulting set of classification rules from the classification rule discovery process (here the GA) represents a source code of an IE that is a major component of intelligent applications, which are different from ML-based applications. Based on the above, the proposed approach in this work is a passive code generator. It is worthy to suggest here the necessity to integrate the classification rules and discovery techniques with other components to produce ASCG that converts natural language requirements to source code.

V. CONCLUSION, FINDINGS, AND FUTURE WORKS

The key novelty of this research work is the proposed automatic approach to defining a set of intuitive linguistic heuristics to define five elements of UML analysis and design models (use case, actor, sender, receiver, and message) instead of the previously and currently used manual approach to defining this set of heuristics.

The automatic definition of a set of intuitive linguistic heuristics has been achieved by developing a new GA-based approach as a method for classification rule discovery and the first time EILH dataset dedicated to applications of automatic extraction of UML diagram elements from English-written sentences, which is called EILH. Because GA is essentially an optimization algorithm, the proposed GA-based approach is distinguished from other methods used for classification rules discovery in that it produces the best-performing classification rules and thus reduces the number of resulting classification rules, which in turn produces fewer heuristics that will be used to elite the elements of UML analysis and design models from English language sentences.

The partial classification discovery was chosen as an approach in this work because it is easy to use, its work is understandable, its performance is robust, and it theoretically produces intriguing patterns for a predefined class, which all make the partial classification discovery approach a valuable data mining descriptive process. Among other techniques of partial classification discovery, GA has been selected due to its ability to solve the problem of the large number of rules that may be produced, which this large number may result in limitations in the use of these rules. The proposed approach has been implemented by using C# programming language and achieved the automation of the composing of source code in the form of a set of production rules (to be used later as part of an IE) by using the GA technique of partial classification discovery. A dynamic fitness function, which was the accuracy of each chromosome, was used for selecting the recognition production rules. The GA is tested on 5 data subsets, which are the use case, Actor, Sender, Receiver, and Message. The size of the initial population for each element was different depending on the number of its frequencies in the EILH dataset, and the number of generations for each element was 100.

Semantic Role Labelling (SRL) Demo software has been used to automate the tokenizing of a sentence and the extracting of the linguistic attributes (lexical, syntax, semantics, and thematic roles) of the tokenized words. However, the shortcomings of SRL of NLP tools in terms of handling vernacular English such as complex ambiguity, slang, words of two syllables, and redundancy of words, are considered true challenges that need seeking better SRL of NLP software and approaches like Relative Fuzzy [49].

The performance of the classification production rule set (which had been produced by the proposed approach) to extract five elements of UML analysis and the design model was assessed using the confusion matrix technique to demonstrate the integrity and effectiveness of such classification rules. The average performance in terms of Recall was 96.78%, precision was 92.31%, accuracy was 92.48%, and the F1 score was 95.76%.

This research work delivers some interesting findings which may represent new concepts and understanding of the relationships between Data Mining, Machine Learning, ASCG, and CASE tools as follows:

1) Development for the first time of an Enhanced Intuitive Linguistic Heuristics (EILH) dataset for recognizing 5 elements of the analysis and design models of UML. This dataset supports the work of automated recognition of UML models of analysis and design by using the ML approach to NLP.
2) Automatic defining for the first time of intuitive linguistic heuristics (features) to recognize the elements of UML analysis and design models in English sentences.
3) The enhancement in the recognition of the elements of the UML analysis and design models of natural language sentences by using the thematic roles as heuristics (conditions) of production rules to achieve this recognition reaches.
4) The new approach to define heuristics (or features) to identify elements by utilizing classification rule discovery from a proposed set of heuristics - the current ML-based applications use optimization algorithms for reducing the number of features rather than defining heuristics of element(s)
5) The contribution to reducing the criticisms of the use of production rules as an approach to solving problems by computer by utilizing GA.
6) The dual use of the classification rule discovery approach to discover a set of classification rules and to automate the composing of a source code. This is shown by the successful using the GA method of the nugget classification rules discovery approach to compose an agile source code to automate the recognition of the elements of the analysis and design models.
7) The contribution to ASCG is shown by the dynamic composing strategy of the source code (here as a list of production rules) rather than the static strategy of filling a template and the deep learning approaches followed by the known ASCGs. This is because neither the template (of a fixed structure) is filled nor deep learning of a fixed number of alternatives to be mapped. In contrast, the composing of sentences by using the classification rule discovery approach is more flexible as new cases (rules) can be added. This contribution is also shown by utilizing the Classification Rule Discovery approach as an I-CASE tool [50] for ASCG.

Finally, As a future work, this approach can be used also to extract other elements like relationships, states, events etc. Such expansion needs expansion of the EILH data set and the availability of formal theoretical definitions of these elements.

REFERENCES

[1] R. S. Pressman and B. R. Maxim, *Software Engineering: A Practitioner's Approach*, 8th ed. New York, NY, USA: McGraw-Hill, 2015.

[2] I. Sommerville, *Software Engineering*, 10th ed. Essex, U.K.: Pearson, 2015.

[3] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, vol. 3. London, U.K.: Pearson, 2018.

[4] M. Mohanan and P. Samuel, "Software requirement elicitation using natural language processing," in *Innovations in Bio-Inspired Computing and Applications. Advances in Intelligent Systems and Computing*, vol. 424. Cham, Switzerland: Springer, 2016, pp. 197–208.

[5] R. J. Abbott, "Program design by informal english descriptions," *Commun. ACM*, vol. 26, no. 11, pp. 882–894, Nov. 1983.

[6] B. Bruegge and A. H. Dutoit, *Object-Oriented Software Engineering Using UML, Patterns, and Java*. Upper Saddle River, NJ, USA: Prentice-Hall, 2010.

[7] A. T. Imam and A. J. Alnsour, "The use of natural language processing approach for converting pseudo code to C# code," *J. Intell. Syst.*, vol. 29, no. 1, pp. 1388–1407, Dec. 2019.

[8] A. T. Imam, A. Alhroob, and W. Jumah, "SVM machine learning classifier to automate the extraction of SRS elements," *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 3, pp. 174–185, 2021.

[9] M. J. Zaki and J. W. Meira, *Data Mining and Machine Learning: Fundamental Concepts and Algorithms*, 2nd ed. London, U.K.: Cambridge Univ. Press, 2020.

[10] B. M. Al-Maqaleh and H. Shahbazkia, "A genetic algorithm for discovering classification rules in data mining," *Int. J. Comput. Appl.*, vol. 41, no. 18, pp. 40–44, Mar. 2012.

[11] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*, 2nd ed. Harlow, U.K.: Pearson Higher Education, 2019.

[12] S.-T.-J. Salma-Tuz-Jakirin, A. A. Ferdaus, and M. A. Khan, "A genetic algorithm approach using improved fitness function for classification rule mining," *Int. J. Comput. Appl.*, vol. 97, no. 23, pp. 12–18, Jul. 2014.

[13] X.-J. Shi and H. Lei, "A genetic algorithm-based approach for classification rule discovery," in *Proc. Int. Conf. Inf. Manage., Innov. Manage. Ind. Eng.*, Dec. 2008, pp. 175–178.

[14] S. N. Pawar and R. S. Bichkar, "Genetic algorithm with variable length chromosomes for network intrusion detection," *Int. J. Autom. Comput.*, vol. 12, no. 3, pp. 337–342, Jun. 2015.

[15] W. Duch, "Rule discovery," in *Encyclopedia of Systems Biology*. New York, NY, USA: Springer, 2013, pp. 1879–1883.

[16] K. Ali, S. Manganaris, and R. Srikant, "Partial classification using association rules," in *Proc. 3rd Int. Conf. Knowl. Discovery Data Mining*. Newport Beach CA, USA, 1997, pp. 1–4.

[17] S. A. Shehabi and A. Baba, "MARC: Mining Association Rules from datasets by using clustering models," *Int. J. Multidisciplinary Stud. Innovative Technol.*, vol. 5, no. 1, pp. 89–93, 2021.

[18] G. Richards and B. de la Iglesia, "Discovery of classification rules from databases," in *Encyclopedia of Information Science and Technology*, 1st ed. London, U.K.: Idea Group Inc (IGI), 2005, pp. 892–896.

[19] G. Richards and V. J. Rayward-Smith, "The discovery of association rules from tabular databases comprising nominal and ordinal attributes," *Intell. Data Anal.*, vol. 9, no. 3, pp. 289–307, Jun. 2005.

[20] K. Azaryuon, B. Fakhar, and A. Daghaieghi, "Classification rule discovery with ant colony optimization," *J. Math. Comput. Sci.*, vol. 9, no. 4, pp. 352–361, Apr. 2014.

[21] B. Liu, H. A. Abbass, and B. McKay, "Classification rule discovery with ant colony," *IEEE Comput. Intell. Bull.*, vol. 3, no. 1, pp. 31–35, Oct. 2004.

[22] R. S. Parpinelli, H. S. Lopes, and A. A. Freitas, "Data mining with an ant colony optimization algorithm," *IEEE Trans. Evol. Comput.*, vol. 6, no. 4, pp. 321–332, Aug. 2002.

[23] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. London, U.K.: Pearson, 2020.

[24] W. B. Langdon, *Genetic Programming and Data Structures: Genetic Programming + Data Structures = Automatic Programming*. New York, NY, USA: Springer, 1998.

[25] R. Robu and S. Holban, "A genetic algorithm for classification," in *Proc. Int. Conf. Comput. Comput.*, Canary Islands, Spain, 2011.

[26] E. M. Jebril, A. T. Imam, and M. Al-Fayuomi, "An algorithmic approach to extract actions and actors (AAEAA)," in *Proc. Int. Conf. Geoinformatics Data Anal.*, Prague, Czech, 2018, pp. 13–17.

[27] M. Murtaza, J. H. Shah, A. Azeem, W. Nisar, and M. Masood, "Structured language requirement elicitation using case base reasoning," *Res. J. Appl. Sci., Eng. Technol.*, vol. 6, no. 23, pp. 4393–4398, Dec. 2013.

[28] P. More and R. Phalnikar, "Generating UML diagrams from natural language specifications," *Int. J. Appl. Inf. Syst.*, vol. 1, no. 8, pp. 19–23, Apr. 2012.

[29] A. Al-Hroob, A. T. Imam, and R. Al-Heisa, "The use of artificial neural networks for extracting actions and actors from requirements document," *Inf. Softw. Technol.*, vol. 101, pp. 1–15, Sep. 2018.

[30] D. K. Deeptimahanti and M. A. Babar, "An automated tool for generating UML models from natural language requirements," in *Proc. IEEE/ACM Int. Conf. Automated Softw. Eng.*, Nov. 2009, pp. 680–682.

[31] A. Imam, T. Al-Rousan, and A. Odeh, "Developing of natural language interface to robot-an Arabic language case study," *Int. Rev. Comput. Softw. (IRECOS)*, vol. 9, no. 7, pp. 1256–1262, 2014.

[32] S. Gulia and T. Choudhury, "An efficient automated design to generate UML diagram from natural language specifications," in *Proc. 6th Int. Conf. Cloud Syst. Big Data Eng. (Confluence)*, Jan. 2016, pp. 641–648.

[33] M. Mohanan and P. Samuel, "Software requirement elicitation using natural language processing," in *Proc. 6th Int. Conf. Innov. Bio-Inspired Comput. Appl.*, Kochi, India, 2015, pp. 197–208.

[34] M. G. Georgiades, "Formalizing and automating use case model development," *Open Softw. Eng. J.*, vol. 6, no. 1, pp. 21–40, Aug. 2012.

[35] V. Punyakanok, D. Roth, and W.-T. Yih, "The importance of syntactic parsing and inference in semantic role labeling," *Comput. Linguistics*, vol. 34, pp. 257–287, Jun. 2008.

[36] M. Palmer, D. Gildea, and P. Kingsbury, "The proposition bank: An annotated corpus of semantic roles," *Comput. Linguistics*, vol. 31, no. 1, pp. 71–106, Mar. 2005.

[37] A. Meyers, R. Reeves, C. A. Macleod, R. Szekely, V. Zielinska, B. Young, and R. Grishman, *The NomBank Project: An Interim Report*. Boston, MA, USA: Association for Computational Linguistics, 2004.

[38] C. W. Ahn and R. S. Ramakrishna, "A genetic algorithm for shortest path routing problem and the sizing of populations," *IEEE Trans. Evol. Comput.*, vol. 6, no. 6, pp. 566–579, Dec. 2002.

[39] F. J. Provost, T. Fawcett, and R. Kohavi, "The case against accuracy estimation for comparing introduction algorithm," in *Proc. ICML 15th Int. Conf. Mach. Learn.*, 1998, pp. 445–453.

[40] D. M. W. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation," *J. Mach. Learn. Technol.*, vol. 2, no. 1, pp. 37–63, 2011.

[41] E. Frank, M. A. Hall, and I. H. Witten, "Appendix B—The WEKA workbench," in *Data Mining: Practical Machine Learning Tools and Techniques*, 4th ed. San Mateo, CA, USA: Morgan Kaufmann, 2017, pp. 553–571.

[42] M. Bramer, *Principles of Data Mining*. London, U.K.: Springer-Verlag, 2016.

[43] S. P. Goyal, "Genetic algorithms for classification rule discovery: Issues and challenges," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 5, no. 6, pp. 514–518, 2016.

[44] V. Y. Rosales-Morales, G. Alor-Hernández, J. García-Alcaráz, R. Zataraín-Cabada, and M. L. B. Estrada, "An analysis of tools for automatic software dvelopment and automatic code generation," *Revista Facultad de Ingenieria*, vol. 77, pp. 75–87, Dec. 2015.

[45] H. Liao, J. Jiang, and Y. Zhang, "A study of automatic code generation," in *Proc. Int. Conf. Comput. Inf. Sci.*, Dec. 2010, pp. 689–691.

[46] A. Svyatkovskiy, S. K. Deng, S. Fu, and N. Sundaresan, "IntelliCode compose: Code generation using transformer," in *Proc. 28th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, Nov. 2020, pp. 1433–1443.

[47] F. L. Loaiza, D. A. Wheeler, and J. D. Birdwell, "*A Partial-Survey on AI Technologies Applicable to Automated Source Code Generation*. Alexandria, VA, USA: Institute for Defense Analyses, 2019.

[48] J. Cruz-Benito, S. Vishwakarma, F. Martin-Fernandez, and I. Faro, "Automated source code generation and auto-completion using deep learning: Comparing and discussing current language model-related approaches," *AI*, vol. 2, no. 1, pp. 1–16, Jan. 2021.

[49] A. T. Imam, *Relative-Fuzzy: A Novel Approach for Handling Complex Ambiguity for Software Engineering of Data Mining Models*. Leicester, U.K.: De Montfort Univ., 2010.

[50] A. T. Imam, A. J. Al-Nsour, and A. Al-Hroob, "The definition of intelligent computer aided software engineering (I-CASE) tools," *J. Inf. Eng. Appl.*, vol. 5, no. 1, pp. 47–56, 2015.

AYAD TAREQ IMAM received the B.Sc. degree in computer science from the University of Mosul, Mosul, Iraq, in 1989, the M.Sc. degree in computer science from Alnahrain University, Baghdad, Iraq, in 1994, and the Ph.D. degree in artificial intelligence from De Montfort University, Leicester, U.K., in 2010.

From 2010 to 2018, he was an Assistant Professor with the Software Engineering Department, Faculty of Information Technology, Isra University, Amman, Jordan, where he is currently an Associate Professor with the Department of Artificial Intelligence, Faculty of Information Technology. He is the author of a book and more than 30 articles. His research interest includes the use of artificial intelligence techniques for software engineering. He is a reviewer in a number of reputed journals in the field of artificial intelligence and software engineering.