CNUIML: Towards the automatic generation of enterprise-level rich Internet applications using Controlled Natural User Interface Modeling Language

H.Bahri[a], H.Motameni[a*], and B.Barzegar[b]

[a]Department of Computer Engineering, Islamic Azad University,
Sari Branch, Sari, Iran
[b]Department of Computer Engineering, Islamic Azad University,
Babol Branch, Babol, Iran

December 20, 2023

Abstract

Model-based approaches attempt to facilitate the involvement of the end user (non-qualified user) in the software development process. Various approaches have been explored to automatically transform the user interface model into the source code. However, the research community has focused less on describing the user interface with natural language. We used the MDA approach and the CAMELEON reference framework (CRF) to develop a controlled natural user interface modeling language (CNUIML) for modeling the user interface of web applications. The meta- model of the designed language is represented by the meta-meta-model (class diagram) and the grammar of the language is developed using ex- tended Backus–Naur form (EBNF). The usability of CNUIML has been evaluated through a case studies.The models described with this language are AUI-level models based on CRF and a platform independent model (PIM), based on the model driven architecture (MDA) approach. In this study and evaluation, we have shown that the model designed with this language can be transformed into similar models such as task models or class diagrams using model-to-model (M2M) approaches. We have also discussed how the source code is obtained from the transformation of this model using model-to-text (M2T) methods.

⋆ *Corresponding author. Tel.: +98 9111140554*
E-mail address: h_motameni@yahoo.com (H.Motameni)

1 Introduction

Since 1968, when the term" software crisis" was first used to describe the difficulties and problems caused by the complexity of producing useful and efficient software in an acceptable time and cost [1], the lack of or weak participation of end users in the software development process has been one of the reasons for the failure of software projects [2].

For this reason, many efforts have been made in research and development to reduce waste and increase the success rate of software projects [3].

The research has shown that, according to one of the principles of Lean Thinking, the shorter the time span between the initial understanding of the problem and the presentation of a partial or final solution, the less the waste of resources and the higher the project success [4]. In this regard, the new methods of software development are based on the agility of the development process and rely on a set of basic principles and guidelines.

One of the ways considered in the research literature to address the above challenge is to leverage the skills of experienced and capable users who are able to actively participate in the development process from the earliest stages and directly contribute to the development of the final result [5].

The research community is interested in whether providing a solution or developing a tool that helps end users participate in the development process can reduce the cost and time required to create useful software for all organizations, especially small businesses. And does it help solve crises caused by failed projects?[5]

Model-driven software development (MDD) has been proposed in research com- munity to increase productivity and simplify the development process. This method tries to obtain the final product from the transformation of the initial models and their gradual transformation, which requires the use of automatic code generation technologies. The use of abstract models facilitates end-user involvement in the early stages of the development process because the models allow end-users to focus on the most important concepts (abstractions) without getting bogged down in the many low-level details [6].

In many software projects, a lot of time is spent on user interface development (UI). Myers et al. have shown that creating a graphical user interface generally takes 48% of the source code, 45% of development time, 50% of implementation time, and 37% of maintenance time [7]. Therefore, 44% of the total time to create an interactive system is spent developing the graphical user interface. [6]. Since the development of the user interface by the end user increases the participation of the end user in the development process [6], as a result, the failure of software projects is reduced.

Model-driven development is supported by several artifacts: Tools, standards, and languages. Some of these artifacts are" general MDD tools", such as AndroMDA[8], Acceleo[9], ArcStyler. Other MDD artifacts are specific to user interface development, such as the user interface definition languages UsiXML[10],UIML[11], XIML, the Interaction Flow Modeling Language (IFML)[12] adopted by the Object Management Group (OMG) , and the Extensible Hypertext Markup Language (XHTML) adopted by the WWW [6].

Myers classified these UI tools into three groups based on how the layout of UI is specified and their dynamic behavior: language-based tools, interactive graphical specification tools, and model-based user interface development tools (MB-UIDEs) [6].

Language-based tools require developers to program in a specific language. Graph- ical and interactive feature description tools allow developers to design an interactive user interface.

Finally, MB-UIDEs generate user interfaces based on models with different levels of abstraction (abstract, concrete, final interface) ,automatically or semi-automatically. They also provide tools for modeling and/or automatic generation of user interfaces. There are two main purposes for using MBUID: model reuse and automatic source code generation [13].

Model-based development has also influenced the development of web applications in the last decades. In this study, we investigated the development of rich Internet form-based web applications using user interface models. User interface is the critical factor for web application adoption [1].

The main topic of this research is the development of a modeling language that allows end users to define the required specifications for the user interface. This tool supports controlled natural language. Currently, modeling languages in the user interface domain are developed in two ways. Based on the Unified Modeling Language (UML) and the UML profile, as a graphical language in a specific domain, and as a textual language in a specific domain [14].

The model-based approach proposed in this study starts with the description of the user interface in a textual modeling language based on MDA[15] approach and CRF.

Model Driven Architecture (MDA) is a standard framework for software development defined by the OMG in 2001 for companies that need to rewrite their software to keep up with technological evolution, providing an approach that allows interaction with the continuous growth of this type of technologies [16]. This approach has been the basis for the development of many user interface description languages [17], [18], [19].

MDA is based on three layers of models. Computation independent model (CIM), the platform independent model (PIM) and the platform specific model (PSM). The transformation between the models eventually leads to the creation of a model in a specific implementation, such as source code in a specific programming language (e.g., Java). A domain-specific language (DSL) is used to write the specifications of MDA models, and the result is a meta-model that reflects the various features of the final software. The OMG has released MOF for this purpose.

A reference model containing a set of models and metamodels, called CAMELEON reference framework, has been presented to provide different perspectives of the user interface, taking into account the diversity of user environments [20].

This reference model consists of four layers that are described and systematically transformed into each other. The first layer models a hierarchy (task & domain) of tasks that must be performed to achieve the user's goals. And it models the concepts related to the scope of these tasks and required by them [13].

In the next step, the required user interface is modeled as abstract interactive units or objects (AIU/AIO) that are independent of the implementation (AUI). Then, a concrete model (CUI) of the user interface is represented by visible interactive units or objects (CIU/CIO). This model depends on the modality, but is independent of the implementation and the way of communication with the user and different communication technologies. This model is similar to the PSM model of MDA.

In the last layer, the user interface is represented in the form of source code based on the implementation technology (FUI) of the target programming language such as Java, markup language such as HTML, etc. [13]. In the CRF modeling scenario, an AUI model plays the role of PIM and a CUI (or labeled CUI) plays the role of PSM. FUI is also the same as PSI in the MDA method.

The use of natural language for this purpose is fraught with difficulties, since the understanding of natural language depends on the interpretation of the author and the reader and leads to misunderstandings. The great flexibility of natural language allows

the developer and the user to provide two completely different descriptions of the same problem. In contrast, there are formal languages that are designed to solve a problem in a specific domain and area by relying on a specific notation, and are often difficult for the end user to learn due to their abstract nature. Controlled languages bridge the gap between natural language and formal language. Thus, they use a limited subset of the rules and vocabulary of natural language to overcome the problem of ambiguity and complexity of natural languages. Simplicity, a standard format for coherence and uniformity of all sentences, a short and active phonetic style, and a very limited vocabulary are the necessities of this type of language [21]. Therefore, the description of the user interface in this research is based on a controlled natural language optimized for describing the structure of the user interface and system data.

This controlled language is designed so that in future research, the generation of the source code, the structure of the system database and the relationships between entities, and the codes for updating and searching (CRUD) the data will be developed automatically from the description of the user interface using transformation languages.

Providing a controlled metalanguage close to natural language to describe user interface elements, with the possibility of grouping data elements in terms of an entity, automatic recognition of data types based on example values or descriptive expressions will be the result of this research.

The main contributions of this research are as follows:
- We have developed a controlled language based on natural language as a user interface modeling language.
- A meta-meta model is presented using UML for the language, based on IFML visual components.
- CNUIML is a context-free controlled language whose grammar is represented using EBNF.
- CNUIML is designed so that the source code , database structure and it's CRUD Operations is generated automatically.
- CNUIML recognizes data types based on sample values, automatically.
- The usability of the language has evaluated using a case study.
- the CNUIML has developed based on the MDA approach and is compatible with CRF.
- This language can be implemented using common modeling tools. We have presented a proposal to implement the language using Xtext and Xtend for automatic generation of the editing environment and generator of the final code.

In the remainder of the paper, we provide an overview of the research community's efforts to create appropriate frameworks and tools for the development of the model- driven user interface and its transformation into final code in Section 2.

In Section 3, we present the methodology used in the research process, which is based on the model-driven development approach. Then, in Section 4, we describe the CNUIML and present the artifacts resulting from the solution design process. To evaluate the usability of CNUIML, in Section 5, we implement the user interface of an example system (case study) using the described language. In Section 6, we discuss the capabilities, shortcomings, and weaknesses of CNUIML, and in Section 7, we make suggestions for improving the current solution and strategies for developing and extending its capabilities.

2 Related works

As can be seen from the review articles, several languages have been invented and presented at different levels of abstraction [22]-[23].

Review studies on UIDLs started in 2003 [23] and were expanded in the following years (2009 [24] and 2011, 2013 [25] by Jovanović et al. and [26] by Mayer et al. [27] by Mitrovic et al. is another review that compares model-based methods for describing the user interface of mobile software. In another study, Ruiz et al. [6] conducted an SLR on the results of 96 articles that introduced MBUIDs. Moldovan et al. [22] also provided a list of 35 user interface description languages that were developed between 1999 and 2019. In 2023, Mejias et al. conducted an SMS to investigate various aspects of the research area. The summary of their findings is as follows:

Research in the field of Model Driven User Interface Design (MDUID) has been popular since 2004. In [28], research conducted in this area was evaluated using the following classification scheme. In terms of process quality, product quality, proposed methodology, technologies supporting MDUID-based methods, models and metamodels, type of research and method, application environment in terms of academic or industrial.

From this review, it appears that the abstract model and the concrete model were of more interest to researchers than other models. In addition, the vast majority of the methods were used in the academic environment. This research has shown that although there is limited empirical evidence, MBUID methods improve productivity, reduce costs, and increase the efficiency and effectiveness of software production methods. They also significantly improved usability and performance. XML, EMF, and UML technologies have been used by researchers in this area more than others [28].

The use of controlled natural language to extract and develop the user inter- face based on the requirements description is of interest in the research community. Juarez-Ramirez et al. proposed a method for extracting and determining user inter- face elements based on natural language descriptions of use cases in [29]. Pinto et al. [30] also presented an approach to automatically generate UIP from agile requirements specifications written in Concordia and its prototyping tool. This tool is capable of prototyping user interfaces for web-based applications. Howard Dittmer and Xiaoping Jia [31] presented CABERNET, a controlled natural language (CNL)-based approach to developing software code in natural language. With this method, programmers can use a language with a simple syntax based on a hierarchical outline to develop their desired software. A program written in the CABERNET language can be processed and executed in different environments, by using patterns.

Natural language-based methods have also been used for other products of the software development process, such as requirements documentation, test cases, and final code. However, most of these methods suffer from a common shortcoming. They were developed to help programmers increase their efficiency and thus save money and time. However, less attention has been paid to increasing end-user involvement and developing a tool that can be used to develop the program's user interface or final code with minimal programming knowledge and without lengthy training.

Research has shown that the methods presented also overcome some other short- comings. Moldavan and colleagues have made a list of these shortcomings in their study [22].Access to documented definitions of these methods is very limited in some cases, their integration with code generators is weak, the scope of automatic code generation for many of them is narrow and limited and does not include all common languages in the field of research, the expressive power of these languages is limited because they are abstract, their support is often limited to design time, and they do not support runtime user interfaces, many of them are not integrated into the application architecture, they often lack efficient software support, such as IDEs and they do not support multichannel user interfaces. In addition, many of them are very difficult for end users to use due to the high level of abstraction and require a slow learning curve.

In the next section, we will describe the step-by-step analysis, design, implementation, and evaluation of CNUIML based on MDA and CAMELEON approaches.

5

3 Research Method

This research aims to develop a DSL for describing form-based web apps user interface that can be automatically transform into source code. Mernik et al. [32] have presented models for the development of DSLs, based on which the DSL lifecycle includes 5 phases of decision making, analysis, design, implementation, and deployment. Visser [33] added a maintenance step to these phases.

In the first phase, the necessity of language development is examined, especially since the development of a specific language requires the development of tools and documents, which is a difficult decision from an economic point of view.

According to [32], two main concerns justify the development of a DSL. These include improving software economics and enabling software development by users who have limited programming skills and are reasonably familiar with the application domain, or by end users who are reasonably familiar with the application domain but have virtually no programming skills.   Facilitating the creation of a graphical user interface is one of the common patterns considered as a subset of the above two concerns.

In the analysis phase, the domain of the problem is identified and knowledge about the domain is gathered. Although many formal methods have been developed for performing this step, it is most often performed in an informal manner [32]. In this study, the analysis phase was performed by reviewing technical documentation, written knowledge of experts, related specialized languages previously developed, and standards developed. Especially in this research, the meta-model resulting from the analysis relies on a simple mapping of the IFML standard.

In the design phase, extending an existing language is a common pattern, but in this study we have developed a language that has little in common with the languages commonly used to describe user interfaces, so we used the invention pattern. When a DSL is designed using the invention pattern, a formal description is used rather than an informal description. There are many methods to describe the syntactic and semantic rules of a language. The most common method to describe syntactic rules is grammar-based systems. Therefore, in this research we use one of the formal methods to describe the syntax of the language (EBNF). This method is a suitable basis for language implementation by one of the most widely used tools in the field of research (Xtext).

Common patterns in the implementation phase include solutions such as interpreters, translators, and preprocessors [32]. In future work, we will use the model-to- text transformation approach, which is considered a subset of preprocessors. In this way, the DSL code is transformed into the source code of the target language (primary language such as HTML, CSS or JavaScript). This step can be seen as equivalent to the PSI development step in the MDA approach using M2T methods.

The relationship between MDE and DSL engineering is becoming closer [34]. As suggested by Kurtev et al, MDE principles and tools can be seen as a suitable support technique for building DSL frameworks.

According to the definition given in [34], DSL is a set of coordinated models. A meta-model representing the abstract syntactic rules of the language called DDMM, concrete syntactic rules representing a concrete and visual form of the abstract concepts, and a semantic implementation which is the result of transforming the concrete model into an implementable model. For example, in this research, we consider the three-level structure of models, including meta-meta-model, meta-model, and model. In this abstract structure, using the standard UML class diagram as a meta-meta- model, we describe a meta-model that represents the components of a user interface and the relationships between them - abstract syntax rules. We also introduce a textual language using EBNF[35], which will be converted to Xtext format in the implementation phase, corresponding to each of the abstract concepts as concrete syntactic rules.

The final user interface model, developed in the target language, will be a semantic implementation of this process.

Backus-Naur notation is a formal mathematical method for describing a language. An improved and extended form of BNF called EBNF has been used to describe the syntax of many languages [36].

4   CNUIML

In this study , a controlled metalanguage is developed that is close to natural language and can be used to describe user interface elements. Grouping of data items in the form of an entity, relationships between groups (entities), automatic recognition of the data type based on sample values or descriptive expressions are among the main features of this language.

Since context-free languages are the most widely used language types in computer science, this study has developed an innovative language that matches the characteristics of these language types.

We need a meta-meta-model to describe the user interface, which represents the main features of the user interface of a web application. Some of these features are defined by business analysts and software system designers or programmers. There- fore, in the first step, we focus on some of the features that are related to the main requirements of the system and reflect the main concerns of the end user.

Most information systems (web applications) consist of a series of interconnected pages and forms on which the user moves and is directed from one page to another (navigation). Each page may contain one or more containers or sub-containers, each of which contains a set of related data items (form and sub-form). This structure forms a tree of objects, where each node is a form, a sub-container, or a data item. Each data item represents a single value or a set of values. These values are of the same type and can have a limited or unlimited range. The figure 1 shows the CNUIML meta-meta model for web application user interface.

According to the IFML standard, user interface components can be classified into four main groups, which include views and their components, events and operations, navigation and information flow, grouping and modularization of the user interface.

In the first step, the end user focuses on the content of the views and the relation- ships between them and specifies the components of each view.

Ignoring the features related to navigation, events and actions and notations re- lated to modularization of the user interface, the standard meta-model is summarized in four features:

1. Form

2. Sub-form (container or subspace)

3. Data item

4. Domain range and type of values

In Figure 1, the CNUIML meta-model, it is shown that each web application is a set of forms that contain data items with a specific value type and domain. Each form can have zero or more associated sub-forms. The relationship between each form and its sub-form is assumed to be a one-to-many association.

The range and type of data values can be specified using a limited or unlimited set of specific data types or by referring to a specific data item in another form or a list of homogeneous data items.

Integer or decimal values, date and time, letters and character strings, logical values, email or binary arrays (such as images and attachments) are acceptable data types of data items.

We need to design a textual language close to natural language so that the modularity of the project in terms of form and sub-form, components and relationships between data items, as well as the type and scope of each value can be accessed through it. In addition, the designed language must have common features in programming languages, especially syntactic grammar and unambiguous semantic description.

In this paper, we use context-free languages in EBNF form to describe language syntax. Context-free languages have enough power to describe recursive syntactic structures, so most programming languages use this type of grammar. The BNF form has been extended to EBNF with several modifications. The CNUIML meta-model is described using the EBNF method , in Figure 2

Every application is considered as a project that has an identifier or a name and includes a set of forms for displaying, entering and editing data. The set of forms is a sequence of expressions that describe the forms and sub-forms related to them and ends with "End of Form". The project ID starts with a valid alphabetic character and can include letters and numbers.

The description of the form begins with "Form of", continues with the descriptive ID of the form, and expands after "as". To describe the form, first the descriptive expressions of the data items are given, and then the description of the sub-forms is placed. The descriptive ID of the form of each statement includes valid letters, which are allowed to use spaces between them.

To describe data items, two characteristics are required: the label or name of the data item and an example of its valid values that specify the type and range of values. The symbol "::" is used to separate these two parts.

The range of valid values of a data item can consist of one or more (a set) of individual values. Individual values may be limited or unlimited. For example, Man is a single value and "Man, Woman" is a finite set of values and "Bachelor,MA,P.H.D,..." is an unlimited set of values. Date, time, numbers, literal strings, email, logical values, and binary values are allowed values. Binary values refer to images or other attachments that can be loaded in binary form.

To describe the range of values of an expression, we use one or more ranges of values that are shown as "start..end", where "start" is the minimum value and "end" is the maximum value of the range. Date, time, and numbers are valid values for the beginning and end of the range.

In some cases, the range of valid values of a data item is limited to the set of values available in other entities of the project. Referential expressions are used to describe this type of ranges which Refers to the name or label of a data item from another form of the project.

The definition of sub-forms is the same as the form and includes the title of the sub- form and the title of the referenced form. The sub-form has a one-to-many association with the main form. The titles of the data elements of the sub-form can be a selection of the titles of the data items of the referenced form or all them. The same descriptive expression is used in the description of the data elements of the sub-forms as in the description of the forms.

5   Case Study

In this article, we use a case study to evaluate the usability of the designed DSL. The subject of the case study is a system for registering courses provided by teachers and lecturers.

The purpose of this course management system is to record details for each instructor, course subjects, and training courses.

The necessary forms of this system, which include registration of the profile, education and experience of the instructor, course subjects, registration of the courses,

and the profile of the students, are presented graphically. These types of forms are available to end users as printed/online forms. Figure 3 shows the class diagram of the system.

Figure 4 shows the course registration form. In this form, in addition to the set of data items of this entity, the course sessions sub-form is also shown, and it shows that the course sessions entity has an association with the course entity. Also, the range of values of two data items (subject and teacher name) is limited to the set of values of entities obtained from other forms. This relationship is considered as referential dependency in the next step. Figure 5 shows the description of course registration form. Each of the data items is named descriptively and the range of its valid values is specified with an example. Reference expressions are used to specify the range of "Subject" and "Teacher Name" values. At the end of the description of the form, there is also a description of the session sub-form. This sub-form also contains data items with specific values.

The figure 6 shows the set of data items of the student entity in a form. Also, the data items of the contact entity and its association with the student entity have been specified. The sub-form of registered courses has also a reference to data items extracted from the course entity, which shows the association between the student entity and the course. The figure 7 shows the set of the teacher's entity data items. The contact sub form also shows the contact entity data items which has an association with the teacher entity.

6   Discussion

In this section, we discuss how the CNUIML fits with the CAMELEON reference model and the MDA approach. Most MBUID approaches are based on the Cameleon Reference Framework (CRF) [6].   Model-driven architecture (MDA) is a standard framework for software development originally described by the OMG in 2001. Three levels of models are defined in MDA: CIM, PIM, and PSM. The CRF reference model layers correspond to the models introduced in MDA. The model of tasks and concepts in CRF can be considered as corresponding to the level of CIM or PIM. AUI is also considered a platform-independent model from the MDA perspective, and CUI is also considered a PSM because it depends on a specific modality, but its implementation is independent of any technology. Finally, the FUI model, which implements the user interface based on source code and depends on a specific technology, is considered as PSI model.

The CNUIML model refers to the tasks and concepts of the system and is computation independent. Although the task hierarchy and navigation are not included in the current version of the language, they can be implemented in the future with some modifications. Since the language has no modality and is platform independent, it can be considered as PIM. This language is at the abstraction level of AUI, from the CRF point of view. It is an abstract language because the input and output data are the basis of the model, not the operations associated with them.

The task model plays an important role in the design, development, and analysis of interactive systems [37]. Task analysis is the infrastructure of user-centered design approaches and aims to collect information from users about what they do and how they do it [38]. Currently, there are several methods, notations, and tools for task modeling and analysis, typically featuring a comprehensive notation to support different task types and functions, hierarchical decomposition, and support for integration through device modeling and dialogs. The most commonly referenced notations are CTT and Hamsters. Looking at these two notations in comparison to others over the twenty-year period, it is clear that CTT is the dominant approach in the general trend, with the exception of 2013. See [38] for a list of the major task modeling methods.

In the tasks and concepts layer, the task model can be transformed into the AUI model based on the CNUIML method by using one of the model-to-model transformation paradigms widely used in the research literature. The M2M transformation paradigm follows the descriptive programming paradigm (e.g., ATL (Eclipse Foundation, 2016b)) or QVT. How these transfer languages can be used to extract the CNUIML model from the task model is beyond the scope of this article. The authors intend to investigate this issue in a future study.

Extraction of the domain model from the user interface or its model has also been explored in the community [39]. The user interface elements are also mapped to classes and relationships between them in the proposed approach, using the M2M transform method. Each form in the CNUIML corresponds to a group of related data, which is considered as a class of entities. The relationships between forms are also mapped to the relationships between classes and entities. The data values and their types are also mapped to the attributes of each class.

Analysis and inference of the existence of relationships between classes is possible through the use of referential dependency and the definition of sub-forms. A one-to- many relationship between two forms exists when the value of a data element has a referential dependency on the data values of another form. If form f_1 contains data element d_1 whose values are constrained to the values of element d_2 of form f_2, it can be inferred that a one-to-many relationship exists between form f_1 and form f_2. This fact can be similarly inferred if there is a sub-form f_2 in the definition of form f_1. If a one-to-many relationship is derived between f_1 and f_2 and vice versa. It can be concluded that there is a many-to-many relationship between these two forms.

In the model-driven development process as well as in the CRF reference model, the transform from the initial model into the final model (source code) is performed using the M2M and M2T transform languages. The CNUIML model can also be transformed to lower layer models (CUI and FUI) by using reification and translation. A suitable tool for implementing transformations from AUI to FUI is Xpand. The Xpand language, developed in the scope of the Eclipse Modeling Framework (EMF), uses patterns that contain definitions of how transformations are performed. Using this tool to implement the transformation process is one of the authors future works.

7 Conclusion and future works

We have addressed the issue of cost and time reduction during the software development process. Software development is a continuous and incremental process in which a product or output is developed at each step. This process continues from the initial artifacts, a form of requirements described by the end user, to the final product created through the transformation of intermediate artifacts. Transforming products and repeating steps usually wastes time and resources. Various stakeholders and human factors are involved in each of the development steps. Another reason for wasting time and resources is the lack of a common understanding of the problem and solution among stakeholders.

In the research literature, one of the main strategies to reduce waste is the use of a model or formal language to develop intermediate products and automate the transformation of artifacts until the final product, i.e., program code, is achieved. This approach has led to the emergence of a model-based software engineering approach. In the development of web-based applications, the trend in the research community is also towards model-based web engineering. In this approach, the development process is based on the development of appropriate models based on the requirements and the continuous and repeated transformation of the models until the final code is achieved. On the other hand, the development of the user interface takes most of the time in the software development process. Therefore, in this research community, special attention

has been paid to software development based on the design of the user interface.

In this research, we have developed a controlled language based on natural language for the initial description of functional requirements by the end user, which allows describing the user interface of form-based web applications. The structure of a form- based web application includes several related forms for processing and displaying information. Each form contains data elements of a specific type and also shows the relationship between the main form and its associated sub-forms. It is a context-free controlled language described by EBNF. A case study was also used to evaluate the usability of the language.

The CNUIML language is designed to be enriched by adding descriptions related to steps such as business analysis and solution design. Finally, by transforming the initial model into the final code, one obtains a program in common source languages such as JavaScript, HTML5 and CSS. The data model and relational database structure can also be extracted from the transformation of the initial model.

Although efforts have been made to consider all end-user requirements to describe the initial user interface of the application, focusing on the most important concerns, namely the structure and content of the forms and the relationships between the main forms and sub-forms, some less important concerns, such as the users who are allowed to interact with the forms based on their role and also the life-cycle of forms and sub-forms, have been ignored in this language. The authors of this paper will address these issues by extending the language in future developments.

Another future work is to develop a visual and responsive tool that converts a descriptive expression based on the CNUIML meta-language into a visual model of the user interface.

Extending the language to support business analysis, navigation and advanced user interactions, events and operations related to user interactions, access control, and advanced data modeling are also future work under this research. Other future goals include the extraction of model-to-code transformation patterns, the development of a tool based on Xpand technology that transforms the forms specified by the CNUIML language into the codes required to build a database and web application based on the MVC-MC or MVVM architecture. The generation of transformation rules in the ATL or QVT language to extract equivalent models such as IFML or abstract models such as UML class diagram and task model is also one of the future research topics. Another research topic is database code extraction and related CRUD operations.

We have attempted to validate the usability of the CNUIML model through a case study. However, evaluating this model based on conventional evaluation indicators and methods presented in the research literature is a separate investigation. Ruiz et al. presented a framework of 21 criteria for evaluating model-based user interface design methods in [6]. Mejias et al. [28] have also proposed the evaluation of some qualitative characteristics of the software development process as well as the software product on which MDUID-based approaches are effective.

Finally, research, invention, or improvement of methods that can extract the CNUIML model from the existing source code may be one of the topics of future research.

References

[1] Soude, H. and Koussonda, K., "A model driven approach for unifying user interfaces development", International Journal of Advanced Computer Science and Applications, vol. 13, no. 7, (2022).

[2] Emam, K. and Koru, G., "A replicated survey of it software project failures", in Software, IEEE, vol. 25, pp. 84–90,. (2008).

[3] Kumar, G. and Bhatia, P., "Impact of agile methodology on software development process", International Journal of Computer Technology and Electronics Engineering, vol. 2, no. 4, pp. 46–50,, (2012).

[4] Fitzgerald, B. and Stol, K.-J., "Continuous software engineering: A roadmap and agenda", Journal of Systems and Software, vol. 123, pp. 176–189,, (2017).

[5] Bano, M. and Zowghi, D., "User involvement in software development and system success: a systematic literature review", (2013).

[6] Ruiz, J., Serral, E., and Snoeck, M., "Evaluating user interface generation approaches: model-based versus model-driven development", Software & Systems Modeling, vol. 18, no. 4, pp. 2753–2776,, (2019).

[7] Myers, B. and Rosson, M., "Survey on user interface programming", in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, New York, NY, USA, (1992), p. 195–202.

[8] "AndroMDA model driven architecture framework – AndroMDA - homepage", [Online]. Available at: http://www.andromda.org/.

[9] "Acceleo | home", [Online]. Available at: https://eclipse.dev/acceleo/.

[10] "UsiXML - USer interface eXtended markup language", [Online]. Available at: http://www.usixml.org/en/home.html?IDC=221.

[11] "Oasis user interface markup language (uiml) tc oasis", [Online]. Available at: https://www.oasis-open.org/.

[12] "IFML: The interaction flow modeling language | the OMG standard for front-end design", [Online]. Available at: https://www.ifml.org/.

[13] Ammar, L., "An automated model-based approach for developing mobile user interfaces", IEEE Access, vol. 9, pp. 51573–51581,, (2021).

[14] Karu, M., "A textual domain specific language for user interface modelling", in Lecture Notes in Electrical Engineering, vol. 151, pp. 985–996. (2013).

[15] "Model driven architecture (MDA) | object management group", [Online]. Available at: https://www.omg.org/mda/.

[16] Fardoun, H., Tesoriero, R., Sebastian, G., and al., e., "A simplified mbuid process to generate web form-based uis", in ICSOFT, p. 835–842. (2018).

[17] Limbourg, Q., Vanderdonckt, J., Michotte, B., and al., e., "Usixml: A user interface description language supporting multiple levels of independence", in ICWE Workshops, p. 325–338. (2004).

[18] Gotti, S. and Mbarki, S., "Ifvm bridge: A model driven ifml execution", Inter- national Journal of Online & Biomedical Engineering, vol. 15, no. 4, (2019).

[19] Cruz, J., Jiménez, S., and Martínez, N., "Lenguajes para el mduid: un análisis de propuestas existentes", Technology Inside by CPIC, vol. 3, pp. 14–35,, (2019).

[20] Calvary, G., Coutaz, J., Thevenin, D., and al., e., "A unifying reference framework for multi-target user interfaces", Interacting with computers, vol. 15, no. 3, pp. 289–308,, (2003).

[21] Gamito, I., "From rigorous requirements and user interfaces specifications into software business applications: The asl approach", (2021).

[22] Moldovan, A., "Openuidl, a user interface description language for runtime omni- channel user interfaces", Proc. ACM Hum. Comput. Interact, vol. 4, no. EICS, pp. 1–86, (2020).

[23] Souchon, N. and Vanderdonckt, J., "A review of xml-compliant user interface description languages", in Interactive Systems. Design, Specification, and Verification, p. 377–391. Berlin, Heidelberg, (2003).

12

[24] Guerrero-Garcia, J., Gonzalez-Calleros, J., Vanderdonckt, J., and al., e.,   "A theoretical survey of user interface description languages: Preliminary results", in 2009 Latin American Web Congress, p. 36–43. (2009).

[25] Mladan, J., Dusan, S., and Zoran, J., "Languages for model-driven development of user interfaces: Review of the state of the art", (2013).

[26] Mayer, C., Morandell, M., Kuntner, A., and al., e., "A comparison of user description languages concerning adaptability based on user preferences", in Assistive Technology: From Research to Practice, p. 1310–1315. IOS Press, (2013).

[27] Mitrovic, N., Bobed, C., and Mena, E., "A review of user interface description languages for mobile applications",   in the Tenth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies UBICOMM, (2016).

[28] Mejias, J., Silega, N., Noguera, M., and al., e., "Model-driven user interface development: A systematic mapping", Human-Computer Interaction: 8th Iberoamerican Workshop, HCI-COLLAB 2022, p. 114–129, (2022).

[29] Juárez-Ramírez, R., Huertas, C., and Inzunza, S.,   "Automated generation of user-interface prototypes based on controlled natural language description", in COMPSAC Workshops, p. 246–251. (2014), Online]. Available:.

[30] Pinto, T., Gonçalves, W., and Costa, P., "User interface prototype generation from agile requirements specifications written in concordia", in Proceedings of the 25th Brazillian Symposium on Multimedia and the Web, (2019), p. 61–64.

[31] Dittmer, H., "Programmer productivity enhancement through controlled natural language input", (2020), Online]. Available:.

[32] Mernik, M., Heering, J., and Sloane, A., "When and how to develop domain-specific languages", ACM computing surveys, vol. 37, no. 4, pp. 316–344,, (2005). [33] Visser, E., "Webdsl: A case study in domain-specific language engineering", in International summer school on generative and transformational techniques in software engineering, p. 291–373. (2007).

[34] Kurtev, I., Bézivin, J., Jouault, F., and al., e., "Model-based dsl frameworks", in Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications, (2006), p. 602–616.

[35] "ISO/IEC   14977:1996",   [Online].   Available   at: https://www.iso.org/standard/26153.html.

[36] Garshol, L.,   "Bnf and ebnf: What are they and how do they work?", (2008), Online]. Available: https://www.garshol.priv.no/download/text/bnf.html.

[37] Bowen, J., Dittmar, A., and Weyers, B., "Task modelling for interactive system design: A survey of historical trends, gaps and future needs", Proc. ACM Hum.- Comput. Interact, vol. 5, no. EICS, pp. 1–214, (2021).

[38] Martinie, C., Palanque, P., Bouzekri, E., and al., e., "Analyzing and demonstrating tool-supported customizable task notations", (2019).

[39] Bačíková, M. and Porubän, J., "Analyzing stereotypes of creating graphical user interfaces", Open Computer Science, vol. 2, no. 3, pp. 300–315,, (2012).

List of Figures

8 Biographies

Hosein Bahri received his BSc degree in computer-software engineering from the Islamic Azad University of Sari and his MSc degree from the Islamic Azad University of Science and Research in 1997 and 2015, respectively, and is currently a PHD candidate at the Islamic Azad University of Babol branch. His research interests include development of computer-aided software engineering tools , modeling methods, end-user development, low-code/no-code development, and model-driven approaches.

Homayun Motameni received his BSc degree in Computer Engineering-Software engineering from Shahid Beheshti University and MSc degree in Computer Engineering and Machine Intelligence from Islamic Azad University and Science and Research Branch in 1995 and 1998, respectively.   He received PhD degree in Computer Engineering (Software Engineering) from Islamic Azad University-Science and Research Branch in 2007. His Current research interests include evolution algorithms, Petri Net, and software systems modeling and evaluation using Petri Net, and machine learning.

Behnam Barzegar is an Associate Professor in the Department of Computer Engineering at Islamic Azad University, Babol Branch, Babol, Iran. He received his B.S. degree in Computer Engineering from Islamic Azad University, Sari, Iran in 2006, and M.S. and Ph.D. degrees both in Computer Engineering from Islamic Azad University, Najafabad and Sari, Iran, in 2009 and 2018, respectively. His research interests include Green Cloud Computing, Task Scheduling, Formal Methods (Petri Net).
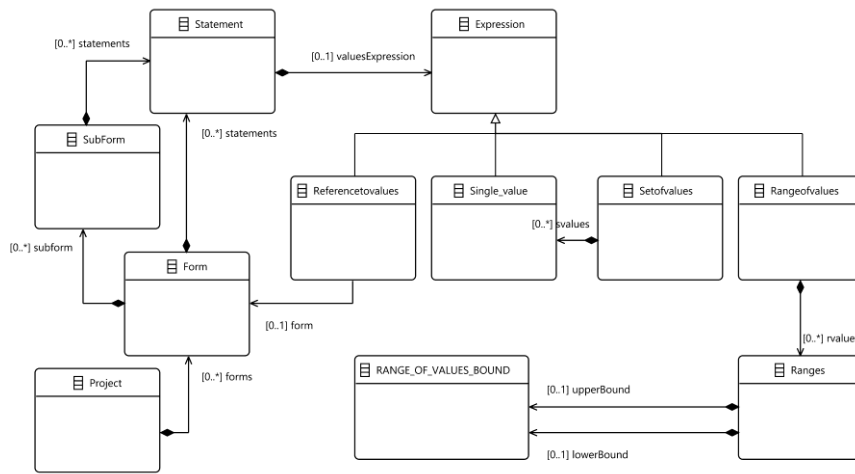
Figure 1. Language Meta-model

```
project = "Project" , whitespace , identifier , whitespace
, { form definition } ;

identifier = alphabetic character , { alphabetic character
| digit } ;
whitespace = ? white space characters ? ;

form definition = "Form of" , title of form , "as" ,
whitespace , statement , whitespace , { statement ,
whitespace } , { subform definition } "End" "Of" "Form" ;

title of form = word , { whitespace , word } ;

subform definition = label , "List from" , title of form ,
"as" , whitespace , statement , whitespace , { statement
, whitespace } ;

statement = label , "::" , expression ;

expression =
single value
| (single value , "," , {single value , "," } , single
value ["," ...])
| (range of values , {"," range of values})
| ("a" | "an") , label , "in" , title of form "List";

label = word , { whitespace , word } ;
word = alphabetic character | digit , { alphabetic
character | digit } ;

range of values = range of values bound , ".." , range of
values bound , { range of values bound , ".." , range of
values bound } ;

range of values bound = ( date | time | number ) ;

single value = ( date | time | number | email | logical |
string | binary ) ;

string = "[[" , { all characters − ("[[" | "]]")}, "]]" ;
date = year , "−" , month , "−" , day ;
year = ( "19" | "20" ) , digit , digit ;
month = ( "0" , digit − "0" ) | ( "1" , "0" | "1" | "2" )
;
day = ( "0" , digit − "0" ) | ( "1" | "2" , digit ) | ( "3"
, "0" | "1" ) ;
time = hours , ":" , minutes , ":" , seconds ; (∗ ISO 8601
∗)
hours = ( "0" | "1" , digit ) | ( "2" , "0" | "1" | "2" | "
3" ) ;
minutes = "0" | "1" | "2" | "3" | "4" | "5" , digit ;
```

Figure 2. CNUIML Grammar   16

```
seconds  = "0" | "1" | "2" | "3" | "4" | "5" , digit   ;
number = [   "+" | "−" ] , digit , { digit } , [ "." ,
digit , { digit } ] ;

email = alphabetic character | digit , { alphabetic
character | digit} , [ "." , alphabetic character | digit ,
{ alphabetic character | digit} ] , "@" , alphabetic
character | digit , [{ alphabetic character | digit | "−" }
, alphabetic character | digit ] , "." , {alphabetic
character | digit , [{ alphabetic character | digit | "−" }
, alphabetic character | digit ] , "."} , ,  alphabetic
character | digit ,  [ alphabetic character | digit | "−" ,
{ alphabetic character | digit | "−" } , alphabetic
character | digit ] ;

logical = "yes" | "no" | "true" | "false" ;
binary = "Attachment" | "Image" ;
```
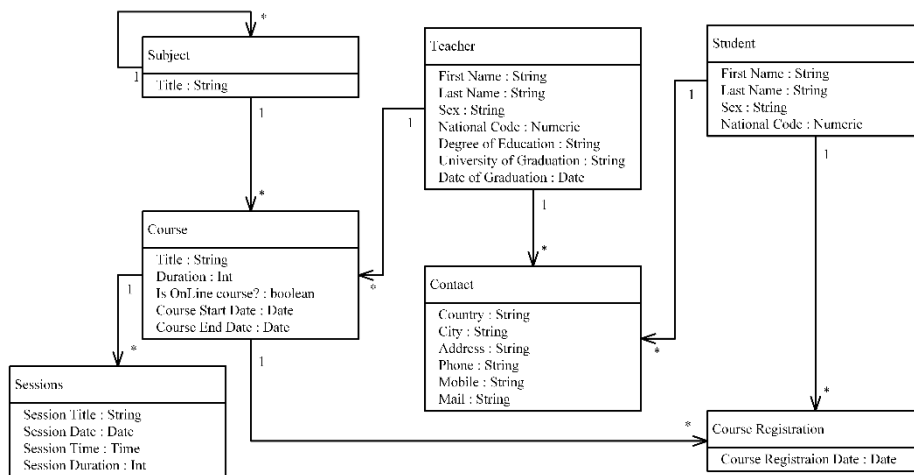
Figure 2. CNUIML Grammar (Continued)

Figure 3. Course Management System Class Diagram

Course Specification Form		
Title: Introduction to statistics - 2022		
Subject: Data Analysis	Teacher Name: Andrew.N	
Is Online Course? ☑ Yes ☐ No		
Start Date: 2022/01/09/01	End Date: 2022/10/30	Duration: 960

(a) Figure 4a. Course specification

Sessions List			
Title	Date	Time	Duration
S1	2022/09/01	18:00	120

(b) Figure 4b. Course sessions

Figure 4. Course registration form

```
Form of Course Specification as
    Tile :: [[an introduction to statistics −2022]]
    Subject :: a Title in Subject Specification list
    Duration :: 5..10000
    Teacher Name :: a Name in Teacher Specification List
    is Online Course :: Yes,No
    Course Start Date :: 2022−09−01
    Course End_Date :: 2022−10−30

    Sessions List From Session Specification as
        Session Tile::S1
        Session Date ::2022−09−01
        Session Time ::00:00..23:59
        Session Duration ::5..300

End of Form
```

Figure 5. Course specification and sessions

```
Form of Student Specification as

   First Name :: Andrew
   Last Name :: N
   Birth Date :: 1970−12−28
   Sex :: Male , Female
   National Code :: 20900000000

   Contact List From Contact Specification as
      Country :: US, England , China , . . .
      City :: New York , London , . . .
      Address :: [ [ No 123 ,Some St ,Any Ave ] ]
      Phone : : +11233152240
      Mobile : : +13256521412
      Mail : :A@y.com

   Course Registration List From Course Registration
      Specification as
      Course Title :: a Title in Course Specification List
      Course Registration Date :: 2022−08−28

End of Form
```

Figure 6. Student Profile Form

```
Form of Teacher Specification as
  First Name ::Andrew
  Last Name ::N
  Birth date ::1970−12−28
  Sex ::Male,Female
  National Code ::2090000000
  Degree_of_Education ::BA,BS,BFA,MA,MS,MD,MFA,MBA,PhD,PharmD,
    JD,SJD,Post−doctoral
  University_of_Education ::MIT
  Graducation Date ::2022−02−09

  Contact List From Contact Specification as
    Country :: US,England,China,...
    City :: New York,London,...
    Address :: [[No 123,Some St,Any Ave]]
    Phone::+11233152240
    Mobile::+13256521412
    Mail::A@y.com

  Courses List From Course Specification as
    Title :: [[an intruduction to statistics −2022]]
    Subject :: a Title in Subject Specification list
    Duration :: 5..10000
    Course Start Date :: 2022−09−01

End of Form
```

Figure 7. Teacher Registration Form