A Software Tool For Programming Training Trough Accumulative Frame System

Rositsa Doneva, Silvia Gaftandzhieva, George Pashev, George Totkov

Abstract: The paper presents a part of a study on the application of frames for the representation of knowledge and processes in programming training. The proposed accumulative frame model serves as a basis for automation of e-learning and e-training activities by the implementation of tools for extraction, aggregation and accumulation of data and knowledge for educational needs. The created on its basis information and computer models and a software tool for data extraction and aggregation are depicted.

Index Terms: programming training, teaching C++ programming language, conceptual modelling, accumulative frame system, learning C++ syntax, software tool.

1. INTRODUCTION

The extraction of data and knowledge from text [1] is a growing field of study. The relevant methods for extraction of text data browse in electronic documents and aim to discover, extract and aggregate data. The implementation of a number of learning processes also requires the solution to the problem for extraction and aggregation of elementary or composite data (knowledge or results of learning processes) from text documents. Examples of typical tasks that require a solution to this problem are:

- self-study in the subject area;
- synthesizing questions related to the learning content;
- summarizing of the learning text (incl. reading comprehension);
- representation of knowledge as frame models;
- generating metadata for the learning content, etc.

The knowledge of different subject areas can be represented in the form of networks (graphs), called frames, semantic networks, conceptual graphs, cognitive models, scripts, scenarios, etc. Frames can be successfully applied in the training process. On the one side, teachers can use training methods to analyse the information received by their students. On the other side, frames help students to learn the content in a thorough way, focusing on the relationship between the basic ideas and the details. A set of studies and experiments in the field should be mentioned. Frame-based representations are developed in various subject areas (e.g. Mathematics, Informatics, Physics, Chemistry, History, Languages, etc.), which aim to systemize the knowledge in the field [2], [3], [4]. An interesting example is the frame-based programming editor Greenfoot IDE [5] implemented to facilitate training in the field of programming. The IDE allows students to combine and edit formatted programming code presented by frame structures. In the field of e-learning, research on the applicability of frames for knowledge representation and improvement of students' knowledge and understanding [6], shows that students obtain higher results during exams. This is due to the fact that frames force students to systemise information, to focus on concepts, and to achieve a higher level of concentration. The paper presents a part of a study on the application of frames for the representation of knowledge and processes in programming training. The proposed accumulative frame model (AFM) serves as a basis for automation of e-learning and e-training activities by the implementation of intelligent tools for extraction, aggregation and accumulation of data and knowledge for educational needs. The created on its basis information and computer models and a software tool for data extraction and aggregation are depicted.

- *Rositsa Doneva, Professor, University of Plovdiv "Paisii Hilendarski", Bulgaria, Email: rosi@uni-plovdiv.bg*
- *Silvia Gaftandzhieva, Assistant Professor, University of Plovdiv "Paisii Hilendarski", Bulgaria, Email: sissiy88@uni-plovdiv.bg*
- *George Pashev, Assistant Professor, University of Plovdiv "Paisii Hilendarski", Bulgaria, Email: georgepashev@gmail.com*
- *George Totkov, Professor, University of Plovdiv "Paisii Hilendarski", Bulgaria, Email: totkov@uni-plovdiv.bg*

2. ACCUMULATIVE FRAME SYSTEM

On the basis of comparative analyses of famous frame models, modern methods and tools (including automated tools) and practices for the implementation of frame representation in the field of education [7], [8] the innovative accumulative frame model [9] has been introduced as a solution for the problem for conceptual modelling of e-learning tasks. AFM is a development of the classical understanding of a frame model and therefore has its typical features. In comparison with the classic frame models, besides procedures (which are performed after filling in different slots in the frame) or demons (to calculate values of slots), so-called "accumulative functions" can be attached to the slots of the proposed AFM. They allow the accumulation of additional data in the e-learning process that can be used to automate different learning tasks.

The set of AFMs representing conceptual knowledge in one and the same subject domain are interrelated by various relation links and form so-called AFM system – the AFM system of the subject domain. An AFM system includes both local knowledge and global knowledge. The local knowledge concerns mainly the knowledge about the network of the included AFMs and about the types to which the values of their slots belong. The global knowledge includes mechanisms for information extraction over the whole AFM system and evaluation of the individual AFMs slots. The AFM systems also contain accumulative knowledge, including knowledge about the subject domain, the context and the aspect/purpose of the representation.The developed models (AFM and AFM system) are tested in the field of Informatics. The developed AFM system "C ++ Language Syntax" (see Fig. 1) consist of 36 AFM. The system offers a conceptual model of programming knowledge in the context of "Learning C ++ Programming" from the aspect of "Language Syntax" and aims to support the

study of C ++ programming language.

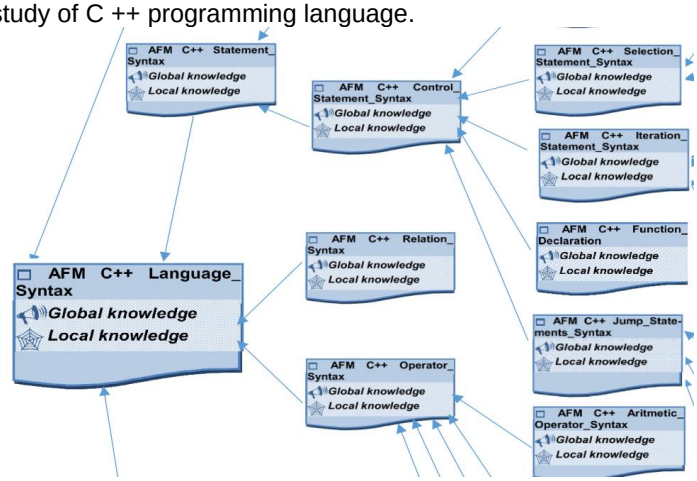

***Figure 1**. A fragment of AFM System "C++ Language Syntax"*

The AFM system is experimented in traditional learning with full-time students studying C++ programming. The verification relies on the comparison of the assessment results obtained by different student groups who solved tasks on the C++ language syntax using the AFM system and traditional solving approaches. Six categories of learning tasks (called student assignments) on the C++ language syntax are used for verification of the proposed AFM system [9]:

Category 1. Development of syntactically correct programming code on the basis of the created frame-instances;

Category 2. Development of syntactically correct programming code after filling in frame-instances;

Category 3. Design of task solutions and creation of syntactically correct programming code;

Category 4. Detection of syntax errors in the programming code;

Category 5. Analysis of unfamiliar programming code to detect C++ syntax elements and creation of frame-instances on the basis of frame-prototypes of the syntax elements found in the programming code;

Category 6. Modification of unfamiliar programming code through using slots' values of frame-instances.

A set of tasks has been developed for each category. For each task, the students' knowledge that has to be assessed has been defined. The solution of tasks from Category 1 requires students to create a syntactically correct programming code through the use of frame-instances (completed by the teacher or extracted from a database with frame-instances). The tasks from Category 2 require students to create frame-instances, and then to write the corresponding programming code. The tasks included in Category 3 are more difficult and complex than the tasks in Category 1 and Category 2. They check students' knowledge and skills to design the solution of the task independently and create a programming code for its decision. The solution of these tasks requires students to determine what types of variables, operators, relations, simple and compound statements they have to use to write programming code. Then students have to create the relevant frame-instances of syntax elements, which they will use to solve the task and finally to write the programming code on the basis of the created frame-instances. Tasks in Category 4 are intended to check whether students can detect syntax errors in programming code. Frame-instances (manually filled by the teacher) and a fragment of a programming code in which students have to look for syntax errors are provided to students. Tasks in Category 5 and Category 6 aim to verify whether students can detect syntax elements in unfamiliar programming code and to modify unfamiliar programming code without knowing the task for which the solution it was created. The verification of the system is made by comparing the results of students (divided into 3 groups) who solved tasks from the six categories. Students in Group 1 and Group 2 solved tasks using the "C ++ Language Syntax" AFM system, and the students in Group 3 applied traditional approaches to solving tasks. Table 1 presents the grades of students from each group. The results showed that students who have solved the tasks using the developed AFM system have higher grades.

***TABLE 1**. VERIFICATION OF AFM SYSTEM „C++ LANGUAGE SYNTAX": GRADES*

Group	Grade 2	Grade 3	Grade 4	Grade 5	Grade 6
Group 1			20%		80%
Group 2				43%	57%
Group 3	20%	40%			40%

3. SOFTWARE TOOL

The results of the verification prove that the proposed AFM system provides an appropriate formalism for conceptual modelling and can be applied in training for improvement of students' knowledge and understanding. On the basis of the results related to the introduction of an innovative 'Accumulation Frame Model' (AFM) and AFM system for conceptual knowledge representation in the subject area [2], a software tool for learning the syntax of language for C ++ programming has been designed and developed. It allows students to solve tasks from the sixth categories mentioned above.

The tool has been developed as a web and mobile application and allows teachers to:

- create AFM systems;
- declare frame-prototypes;
- create frames-instances of already declared frame-prototypes;
- create tasks in each of the six categories set out in Section 2.;
- approve frames-instances created by students or teachers, to store them in the database and to generate new tasks for the next training course.

The AFM system consists of a server-side, client sides and a transport layer subsystem. The Server Side makes use of Node.JS and Mongo.DB for database storage and provides REST Services for the transport layer subsystem. Two different clients are developed, one for the web-based environment and one for Mobile Applications for Android. Both of them consume the same REST Service.

Fig. 2 shows a UML diagram depicting the structure of a collection with tasks. The collection contains tasks with assignmentText attribute, which contains the short text of the task. For each task, there is an obligatory attribute frameSequence, which is an object which contains sequenceId, sequenceName and an array of frameInstances. A frameInstance object contains InstanceTypeId, which is related to the id of the frame-prototype of which the frame-instance is. The instanceValue object contains the value of the

frame-instance, which may have a different structure, depending on the AFM declaration. When a frameInstance object is inserted or updated in the collection, an automated validation based on a JSON Schema validator is triggered. JSON Schema is automatically built when a frame model is inserted or updated, based on this model.

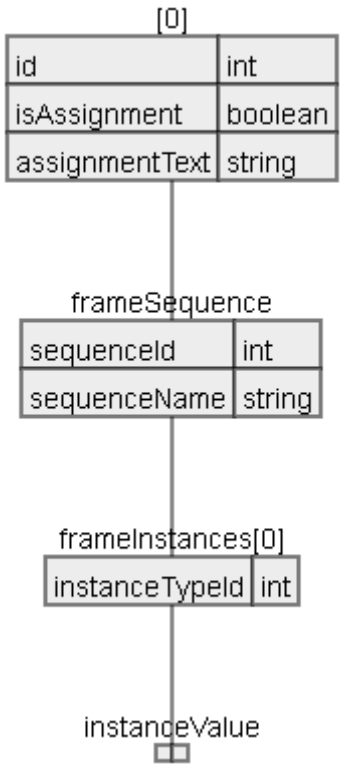

***Figure 2**. An UML diagram of the structure of Tasks collection*

A partial implementation of the validation, based on JSON Schema is given in Fig. 3. The implementation makes use of the "jsonschema" and "mongodb" Node.JS modules.
The object instanceValue is validated against the resultSchema schema, which is already generated and stored in the collection frameModels. If the validation is unsuccessful, an exception is thrown which result in the creation of a JSON object with an error text attribute, returned to the client from the REST API.

```
var Validator = require('jsonschema').Validator;
var v = new Validator();
var assignmentID=context().getAssignmentID();
var MongoClient = require('mongodb').MongoClient;
var url = context().getMongoDbURL();
var instanceValue=context().getCurInstance(); //contains the object, which is to be validated
MongoClient.connect(url, function(err, db) {
 if (err) throw err;
 var dbo = db.db("mydb");
 dbo.collection("frameModels").
   find({id: instanceTypeID}, { projection: {  _  id: 1, jsonSchema: 1 } }).
   toArray(function(err, resultSchema) {
if (err) throw err;
if(!v.validate(instanceValue,
  resultSchema.jsonSchema))
  {throw "Invalid JSON value";}
    ……; db.close(); //close database and save
  });
});
```

***Figure 3**. Node.JS based implementation of JSON Schema validation*

Before being able to add tasks from the six categories, teachers have to create prototypes of each AFM in the system they will use - in this case "C ++ Language Syntax". Fig. 4 presents a screen with an example of defining local knowledge for a frame prototype modelling knowledge for variable declaration in C ++ programming language.

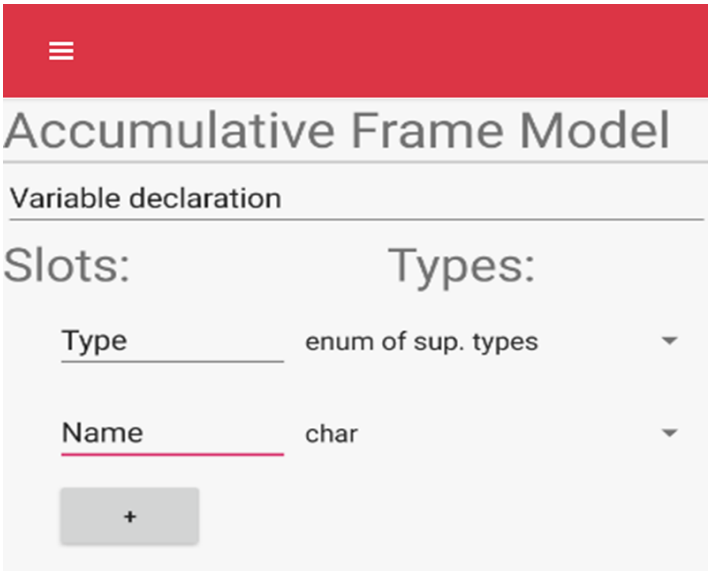

***Figure 4.** Local knowledge of frame-prototype "Variable Declaration"*

Once the frame-prototype has been created, their instances can be created. Frame-instances are grouped into frame instance sequences (FIS), which can be used in formulating tasks from each category. Fig. 5 provides an example of creating frame-instances of frame-prototypes „Simple Assignment Statement" and „Variable Declaration".

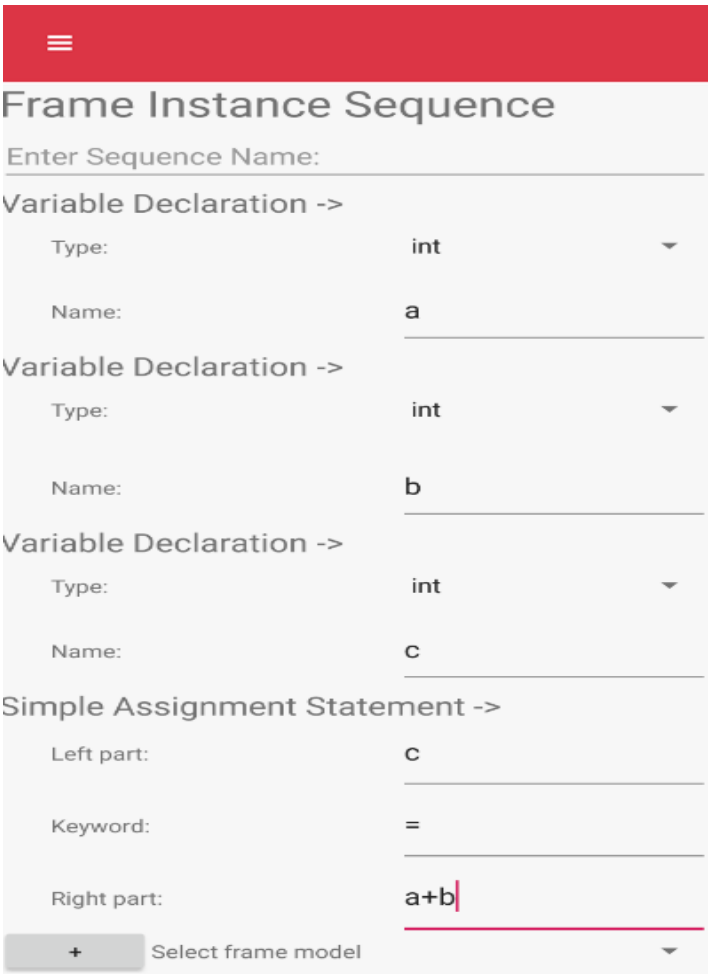

***Figure 5**. Creation of FIS on „Simple Assignment Statement" and „Variable Declaration"*

The tool allows teachers to add tasks from the six categories. In order to add a task from Category 1, a teacher must type a short text for the task and then select a FIS that successfully describes and solves the task. Students are supposed to write syntactically correct programming code, based on the selected FIS. Fig. 6 presents an example of adding a task from Category 1.

***Figure 6**. Adding an Student Assignment (task) from Category 1*

After logging into the system, students have access to the tasks added by the teacher. Figure 8 presents an example of task from Category 3 given to the students. To solve the task, students have to determine which operators to use and in what sequence they should be used, i.e. to determine the steps to solve the task. For example, to solve the task for calculating the area of a square (see Fig. 7) students need to create frames-instances of frames-prototypes which represent knowledge for variable declaration, entering values from the keyboard, calculating values of expression, printing values on the screen and then to write a programming code to solve the task. The frame-instance they have created are stored in the database and the teacher can use them to create new tasks.

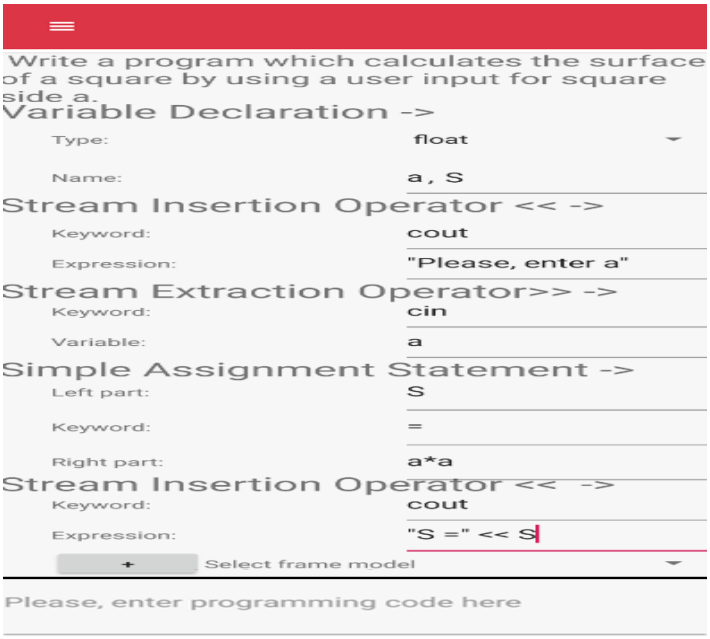

***Figure 7**. Student screen with task*

4. CONCLUSION

The paper is a part of a study [8], dedicated to the application of frames for the representation of knowledge and processes in e-learning and their applications for the development of intelligent software solutions in various areas. The proposed AFM system was experimented with full-time students studying C++ programming language. After verification of the proposed AFM system, the software tool was developed. It allows the automation of a number of training and learning activities, e.g. filling out of the frame-instances by students, accumulation of created frame-instances in a database with good and wrong examples, generation of learning tasks, assessing the students' knowledge, etc. By using frame-based conceptual models of subject domains, the tool provides acquisition, extraction, inference and accumulation of relevant knowledge. In addition to the tasks set out in Section 2, an additional category of tasks can be implemented in the tool, which will allow students to:

• define the elements of the algorithm used to solve the task (control elements);
• define and invoke multiple test scenarios for automated testing of the task they solve (test scenarios);
• define test scenario as a set of test items (test items);
• test the solution of the task based on the test scenarios.

REFERENCES

[1] R. Feldman, J. Sanger, The text mining handbook: advanced approaches in analyzing unstructured data, Cambridge university press, 424 p, 2007.
[2] R. Doneva, S. Gaftandzhieva, and G. Totkov, "Frame Representations in e-Learning – Applications and Developments", International Journal on Information Technologies and Security, vol. 10, no. 2, pp. 23-32, 2018.
[3] T. Kulgildinova, and A. Uaissova, "Realization of frame-based technologies in the context of education in informatization", Journal of Theoretic and Applied Information Technology, Vol.89. No.1, 2016.
[4] O. Fonseca, "Learning styles and knowledge representation systems in Ecaes (quality examinations for higher education) for medical students", Revista Horizontes Pedagógicos, Vol. 17, no. 1, pp. 42-52, 2015.
[5] R. Gurina et al., Frejmovie opori [Фреймовые опоры. NII Shkolnih tehnologii, Moscow], 2017.
[6] G. Shivacheva, G. Totkov, and R. Doneva, "Reading programming code with understanding", Scientific Researches of the Union of Scientists in Bulgaria - Plovdiv, Series C. Technics and Technologies, Vol.15, pp. 58-63, 2017.
[7] W. Thomas et al., "Evaluation of a Frame-based Programming Editor", In Proceedings of the 2016 ACM Conference on International Computing Education Research (ICER '16). ACM, New York, NY, USA, pp. 33-42, 2016.
[8] G. Totkov, S. Gaftandzhieva, and R. Doneva, "Accumulative Frame Models in e-Learning", Scientific Researches of the Union of Scientists in Bulgaria - Plovdiv, Series C. Technics and Technologies, Vol. 15, pp. 17-20, 2017.
[9] S. Gaftandzhieva, and R. Doneva R, "Verification of Accumulative Frame System in Programming Training", Pedagogy, Azbuki, Vol. 91, No. 7, pp. 982-993, 2019.