Development of Frame Systems Shell for Learning of Knowledge Representation Issues

Ieva Valkovska, Janis Grundspenkis

Abstract: *The paper describes a created shell Frame System for Knowledge Representation (FSZA) that is used for student tutoring purpose. Functions that are available in the shell are described and its architecture presented. The requirements for the extension of the shell are formulated from the structural modelling viewpoint. The final goal of the outgoing work is to develop a flexible tool for knowledge representation and reasoning on complex systems structure, functions and behaviour.*

Key words: *Frames, Frame System for Knowledge Representation*

INTRODUCTION

There are situations, when someone does the work even not thinking about it, because he/she recognizes the situation that has happened before and, therefore, can be done automatically. If we follow the theory that was purposed by Minsky [1], in this case a person uses the hierarchical, on experience-based structure that is called a frame. "A frame is a collection of questions to be asked about a hypothetical situation that specifies issues to be raised and methods to be used in dealing with them" [1]. Different types of frames, which are gathered together in one system, describes concept including traditional Object – Atribute – Value (O-A-V) triplet.

Usually a frame system represents a hierarchy of objects and/or concepts. Two or three level hierarchies of frames are commonly used (class frame, subclass frame and instance frame). The class frame specifies typical information for all class; the subclass frame specifies concept that is a part of class, but the instance frame specifies a special instance of the object or concept. Modern approach in frame systems is based on object-oriented paradigm. In this case frame systems have the same advantages as object-oriented approach, namely, encapsulation, inheritance and message sending abilities. Frame systems can be viewed as the network with nodes and relations between these nodes. Frame system allows representing the knowledge almost naturally, provides the conceptual and structured representation of the object relations and supports object definitions by situation.

Since Minsky introduced a frame idea [1], frame systems became well-known in knowledge representation communities [2, 3]. The knowledge about the problem domain is hard to structurize and define due to the essentially different perceptivity of humans. That is the reason why the detailed templates are needed to represent the acquired information. In this case the frame can be used like detailed template. Frame idea is one of the ways to structurize the concepts that are necessary be represented in an understandable way for people. Unfortunately the frame systems are not widely used in the tutoring of the Artificial Intelligence courses at least at Riga Technical University (RTU). Usually lecturer gives students a material on specific topic or explains the basic principles. One of the reasons why frame systems are not widely used in learning process is their inaccessibility. All known frame-based shells and systems are in English. It makes the learning process harder due to the fact that some foreign language skills problems still exist in non-English speaking countries. The price of known products is high and their availability is limited. We have not found any demo version of the frame system. Therefore, it is necessary to develop and implement a frame system for training purpose in Artificial Intelligence course in order to make acquired knowledge about frames usable for students of our university. This system should be expanded and improved to solve the tasks proposed by Structural modelling (approach developed in RTU for complex system modelling) [4].

THE DEVELOPED FRAME SYSTEM SHELL

To organize the work more efficient and interesting for students, it is useful to develop a shell, which students could use for developing their own frame-based systems and share their experience with other students. We introduce one of the possible solutions that is used to represent the user's knowledge. The user can be any person that uses the shell – in our case a student or a teacher. We have developed so called *Frame System for Knowledge Representation (FSZA)* that is used in student's practise to acquire knowledge and get skills of knowledge representation that is a part of "Foundations of Artificial Intelligence" course teached in RTU. The shell FSZA is developed to help to understand problem domain and to simulate possible structure, and the relations between concepts or objects. FSZA provides the basic functionality representing concepts and realizing the inheritance process. FSZA is good for learning how to create a frame system that is powerful and useful tool for knowledge representation.

To develop the shell FSZA we have used the Borland C++ Builder 6. Borland C++ Builder 6 uses the object-oriented programming language C++. C++ introduces class structures and mechanisms that are useful to create the structured and object-oriented system [5].

The main window of the shell is presented in Fig. 1. FSZA has a graphical interface with buttons, standard menu and shortcuts that allow to realize the required functionality for knowledge representation. It is possible to use the mouse or arrow buttons and shortcuts to navigate in the shell. Two languages, Latvian and English, are available. The created frame systems structure always is represented on the left hand side of main window as a tree of frames. Active (currently used frame) frame's name, superclass frame's name and a list of properties for an active frame are located on the right hand side.

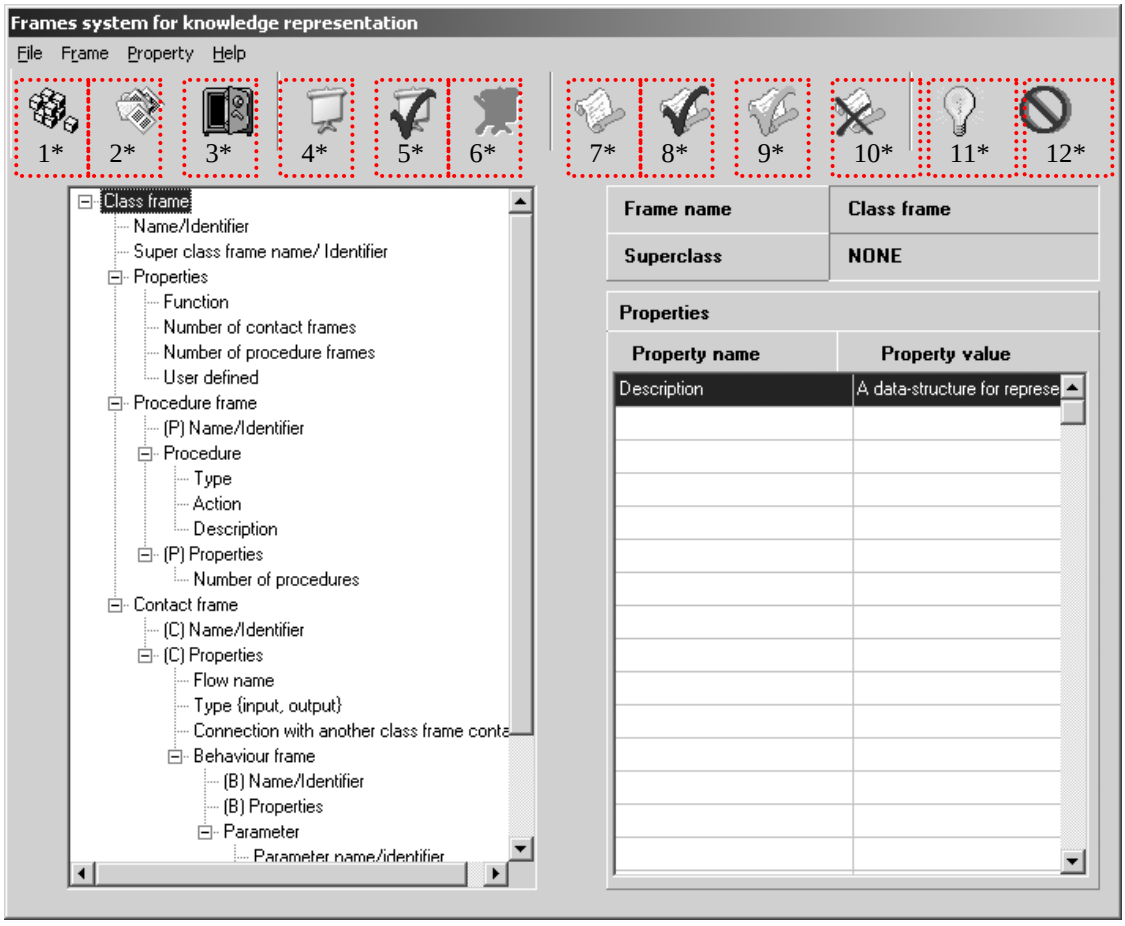

Fig. 1. The main window of the shell FSZA

The buttons that are numbered and marked with dotted lines have the following meaning:

1* Create a new frame system;
2* Open existing frame system;
3* Save a frame system;
4* Add a new frame to the frame system;
5* Edit a selected frame name;
6* Delete a selected frame;
7* Add a new property to the frame;
8* Edit a property name;
9* Edit a property value;
10* Delete a selected property;
11* Help;
12* Exit;

With FSZA it is possible to create frame systems that help to represent a declarative knowledge, that is, at the present moment dynamic mechanisms (message sending and facets) are not available. A help system is used for users support. The help system explains the theory of frame's, the architecture of frame system and actions that can be performed in the FSZA shell.

From the technical implementation point of view class definitions are managed at the level of programming. The created structure consists of three classes: *Frame object Frame property* and *Frame system*. Looking at described structure it is necessary to bear in mind that used class and variable names does not have any special semantic meaning or purpose. *Frame object* is a class that implements and represents the frame and its pointers to variables in the code. *Frame property* is a class that implements the traditional meaning of frame slots. *Frame system* is a class that implements the generic structure of the frame system with common methods, for example, method **Save frame system**. The public variables and methods that are used in class definition declare data, which are reachable through other class methods. The private variables and methods are restricted in their access and are not recognized outside the class where they are defined. Parent-child relations further in the text are assumed to be similar with superclass-class relations. In the shell FSZA this terminology is used to make the hierarchy of frames more understandable and unambiguous for the reader.

Frame object consists of eight public variables: *name*, *parent*, *instance*, *dcount*, *pcount*, *ccount*, *prop* and *cindex*. The variable *name* is used to set the active frame name and it must be unique. The variable *parent* is the number of parent index in the frame system. It is used to determine the active frame superclass in the structure if it is necessary. The variable *instance* is Boolean type and is used to determine the active frame status. If variable is set as an instance in the frame system frame can't have any sub classes by definition anymore. The variable *dcount* is used to determine the number of delivered properties for the active frame. This variable is necessary for the purpose to know how much properties the active frame has got from its parents and to make calculations to determine how much properties has the current frame by its own. The variable *pcount* is used to determine the number of all properties that are defined for the active frame. The variable *ccount* is used to determine the number of children or subclass frames that are defined for an active frame. The variable *prop* is used to implement the *frame property* class in the *frame object* class where the maximum number of properties can be equal to some predefined constant in the code or in case of large frame systems it is the user's defined number. The variable *cindex* is used to determine the active frame children indexes. Indexes are assigned after the children frame is created and are needed for better understanding of the existing structure. The acquired numbers *dcount, pcount,*

ccount are used to organize the interface, some internal functions and cycles, that provides the functionality of the shell.

Frame property consists of three public variables: *name*, *value* and *inherits*. The variable *name* is used to set the names of properties that are associated with an active frame and that can be changed only in the case when a property is not inherited from a parent frame. An active frame is a frame that contains the concept that the user currently is representing in the system. The variable *value* is used to assign the property value that is related to a specified property name. The property value can be changed in any case when it isn't inherited from a parent frame. The variable *inherits* is Boolean type and is used to check is the property inherited from a parent frame or not.

Frame system consists of two private variables: *fcount* and *fram*. The first variable *fcount* is used to determine the number of frames that are already represented in the frame system. The second variable *fram* is used to implement a class *frame object* into a *frame system* class. The class *frame system* has many public functions defined to provide the needed functionality and performance, but they are not described in this paper.

The *frame property* and *frame object* variables *name* can contain any symbols from the character set and are defined as AnsiString type. The variables *prop cidex* and *fram* are variables that references to the specified type. The structure presented in Fig. 2 shows that the variable *prop* referencing to class frame property allows to create hierarchy where as a result *frame object* includes *frame properties*.

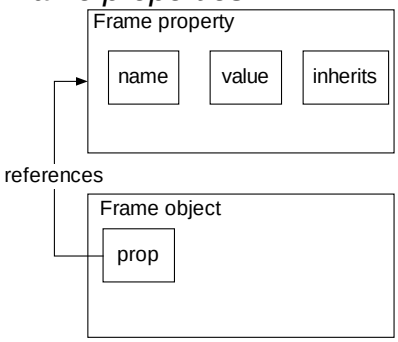

Fig.2 Principle of the reference in the FSZA shell

The classes are implemented as follows (C++ is used):

```
class TFrameProperty {
public:
AnsiString Name;
Variant Value;
bool Inherits;
};
class TFrameObject {
   public:
        AnsiString Name;
        unsigned int Parent;
        bool Instance;
        unsigned int dCount;
        unsigned int pCount;
        TFrameProperty *Prop[MaxProps];
        int cCount;
        int *cIndex[MaxFrames];
};
class TFrameSystem {
   private:
        int fCount;
         TFrameObject *Fram[MaxFrames];
```

Fig.3 Classes in C++

The variable *value* for a frame property in some cases must be a character string and in some other cases a number. So, in the shell this variable is defined as variant type variable. All other variables are required and defined as one kind of number type.

REQUIREMENTS FOR COMPLEX SYSTEM REPRESENTATION

Now let look on the knowledge representation problem from the broader point of view, i.e., how to represent, at least partly, a "common-sense" knowledge about physical objects and/ or abstract concepts. "A "minimal" common-sense system must "know" something about cause-and-effect, time, purpose, locality, process, and types of knowledge. It also needs ways to acquire, represent, and use such knowledge" [1] Our purpose is to improve the developed shell according to ideas that are established by structural modelling realizing "minimal" common-sense facilities that allow to represent knowledge about complex systems structure, functions, behaviour and cause and effect relationships [4].

As already mentioned the shell FSZA offers a solution for knowledge representation, but its disadvantage is that it lacks dynamics. This problem is critical and very important in representations of complex systems. At the present moment the FSZA user can create the frame system or its template that can be used by another user who expands the common knowledge corresponding to current his/her knowledge level. The dynamics including behaviour and procedures could be a vague step for condition creation. Consequently it could provide environment and concept change and/or reaction on change representation. We propose to use the structural modelling principles [4] in the developed shell structure.

Structural modelling:

- allows to expand the amount of knowledge about every concept that is described in the system;
- allows to improve the dynamics (the represented data will not be just declarative facts about objects or concepts);
- allows to automate the knowledge acquisition from the user;
- provides the additional knowledge management techniques and the reasoning abilities;
- provides the system with additional ability to represent relationships (that are characterised by matter, energy or information flows) between different classes that are not connected with superclass-class relations.

To expand the FSZA and to implement the all necessary aspects, a different frame systems structure is developed. Instead of a traditional frame we use a set of several different frames. This is called a *frame model*. The frame model is a structure of four types of frames. There are a typical class frame and, in addition, a procedure, contact and behaviour frames in the proposed structure. The frame model is shown in Fig 4.

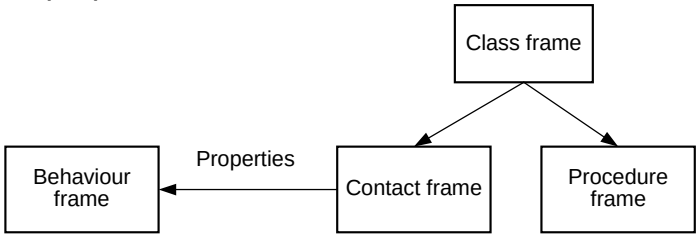

Fig. 4 The main elements of the Frame Model

The model is used to refer to stereotypical aspects in the real world allowing to join the behaviour and process knowledge related to user's used concepts.

Procedure frame is a data structure, which consists of stored information about property and behaviour state changes of the frame. It inspects the rules, which affect an active frame properties and other frame properties that are associated with an active

frame, directly operating with data structures and making changes in the system. Procedures show the activity, steps of the task and instructions.

Contact frame is a data structure where information about the flows of specific frame is stored and the pointer to behaviour of an object is defined. Every frame can have more than one contact frame.

Behaviour frame is a data structure that stores the information about observed behaviour and is realized by the contacts and flows. Behaviour is defined as activity or reaction. The notion of the *behaviour state* is introduced that represents effects on the considered active frame or effects provided by the considered active frame on other frames from its environment.

At the moment the new structure is under the development and it must be implemented in a new shell. There is an ontology developed that defines and describes the elements of the proposed structure and its relations. The frame cardinalities also are determined that represent number of minimal and maximal frames in one level that can be related to a frame in another level.

Using the described extension larger amount of knowledge will be acquired and represented. We consider that acquired knowledge will be of better quality, more flexible of time and state changes in the domain.

CONCLUSIONS AND FUTURE WORK

This paper describes the developed shell FSZA. The main purpose of the shell is to provide the static knowledge representation.

The empowerment of complex system representation was a reason why we started to search for new aspects in frame representations that will support different kinds of reasoning. New possible extension is proposed where a set of different frames is used. All provided assumptions are only conceptual ones now but according to them we hope to create a real prototype that will provide possibilities to represent both static and dynamic knowledge. In the future we will try to develop the knowledge bases that will improve the effectiveness for the existing shell, but it is a project that consumes a lot of time and investigations.

REFERENCES

[1] Minsky, M. A Framework for Representing Knowledge, The Psychology of Computer Vision, ed. Patrick Henry Winston, McGraw – Hill Book Company, New York, 1975, pp. 211-277.

[2] Fikes, R., T. Kehler. The Role of Frame-Based Representation in Reasoning, CACM 28(9), 1985, pp. 904-920.

[3] Karp P. D. The Design Space of Frame Knowledge Representation Systems, Technical Report 520, SRI International, Artificial Intelligence Center, 1992

[4] Grundspenkis, J. Structural Modelling of Complex Technical Systems in Conditions of Incomplete Information. Modern aspects of management science, No.1, Riga, Latvia, 1997, pp. 111-135.

[5] Romero-Hernandez, I., J.L. Koning. From Roles to Agents: Considerations on Formal Agent Modeling and Implementation; Intelligent knowledge based systems: business and technology in the new millennium; V.3. Expert and agent systems, ed. Cornelius T. Leondes, Kluwer Academic Publishers, 2005, pp. 154-181.

ABOUT THE AUTHOR

Assistant Ieva Valkovska, B. Sc.ing., Department of Systems Theory and Design, Riga Technical University, Phone: +371 7089581, e-mail: ieva@cs.rtu.lv.

Prof. Janis Grundspenkis, Dr. Habil. Sc. ing., Department of Systems Theory and Design, Riga Technical University, Phone: +371 7089581, e-mail: jgrun@cs.rtu.lv.