ITM Web of Conferences **6**, 03012 (2016)
DOI: 10.1051/itmconf/20160603012

The new approach to the problem-driven object programming

Alexander Prutzkow[1,a] and Dmitriy Tsybulko[2]

[1]*Ryazan State Radioengineering University, Department of computational and applied mathematics, 390005 Ryazan, Russia*
[2]*Ryazan State Radioengineering University, Department of computational and applied mathematics, 390005 Ryazan, Russia*

Abstract. Using more and more computing devices requires a new programming approach. The programming approach should be simple for an ordinary device user and problem-driven. So we propose a new programming concept for easy code writing by any non-programmer. We named it the Problem-Driven Object Programming or PDOP. According to the concept, program is a control flow by program objects. Every object is a action for a problem solvation and has the same structure. The object structure is defined by formal grammar rules. During code writing, the user programs a control flow and sets type and parameters of the object. The PDOP is also a simplified programming language with user-friendly graphical representation.

1 Introduction

Writing a programming code requires adequate programming approaches. Currently, there are a few different approaches available to programmers: structural, object-oriented, logical, functional, etc. Born in the 1970s of the 20th century, the object-oriented programming approach has become more common. Back then, leading programming experts had to face a task how to achieve a greater conciliation of data and programs. A solution to the set task called for a change of von Neumann's presupposition according to which “data and programs are indistinguishable in machine's memory” [1]. The basis of the outlined conception rests on three founding principles: inheritance, encapsulation and polymorphism. The approach built on these principles was named object-oriented programming (OOP).

OOP is a universal approach to describing task solution as unified consistent data structures. OOP was offered as a way to reduce duplicating codes. To use this approach a programmer has to have certain knowledge and skills.

Firstly, it is important to understand such basic concepts as classes, inheritance, dynamic binding, modules, abstract data types and the like. Studying the principles of how data encapsulation functions takes a great deal of time, whereas designing classes seems to be the most challenging. This task is more complicated than their usage. Secondly, to be able to settle the problem a specialist ought to possess quite a lot of experience. An OOP-based program design is rather complicated and is solely the responsibility of a programmer. Thirdly, a right set-up of the structure allows for an easy check-out and exploitation of the program. A fully satisfactory object model makes target software more scalable and flexible. The quality of the designed model depends on the competence of a specialist who creates the product.

2 Problem-driven object programming concept

A great number of consumer microprocessing devices and related operational systems and software to run them generated a need to write a simplified programming language that could allow any user to learn programming tools and solve focused tasks.

Normally, the solution to focused tasks motivates users to resort to 5-10 different commands. Using basic operations from sophisticated programming languages, one can group atomic statements into blocks that enable a user to write problem solution. A user needs a language with a limited number of commands to solve a problem. This will help a user to settle problems for a specific subject area quite effectively. This approach to problem solving was called Problem-Driven Object Programming or abbreviated PDOP [2].

For the PDOP, any operation is described by an object that consists of two main parts – hidden and open [3]. The hidden part performs actions, checks conditions for control transfer to another object and is inaccessible to the user.

The open part includes the following sections:

- a field section (parameters used by the object to perform actions);
- an initialization section that contains commands acted out by the object on getting control;
- a condition section that contains commands for transferring global data and controlling functions to other objects according to datum conditions.

The user can set different characteristics for these sections, necessary for problem solving.

The objects exchange global data through a special storage area that can be structured as a stack, a queue or a memory space with direct access (array variable). The

[a] Corresponding author: mail@prutzkow.com

memory access is done through the initialization section or condition section.

Logically, programs in PDOP is a bifurcated structure and can be identified as an aggregate of objects sending each other global data and control.

We can describe PDOP with the help of formal grammar rules:

```
object = indentificator «=» type «{» { parameter
{parameter} } { transfer_condition {transfer_condition} }
«}».

parameter = indentificator « : » (symbol  |  digit { symbol  |
digit } ) «;».

transfer_condition = if (parameter | «_nocondition») «call»
identificator «;».

type = identificator.
identificator = letter{symbol}.
symbol = letter | digit | «_».
letter = 'A'...'Z' | 'a' ...'z'.
digit =  "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" |"9".
```

The record «{element {element}}» means one or more elements of the rule «{element}».

A full description of the formal grammar of the PDOP language is provided in the article [4].

To illustrate our reasoning, we propose the following description of types of objects that are used in the knowledge test script: Q – show a question on the screen and get an answer (figure 1), T – show an explanation on the screen (figure 2), R – show a knowledge test result on the screen (figure 3).

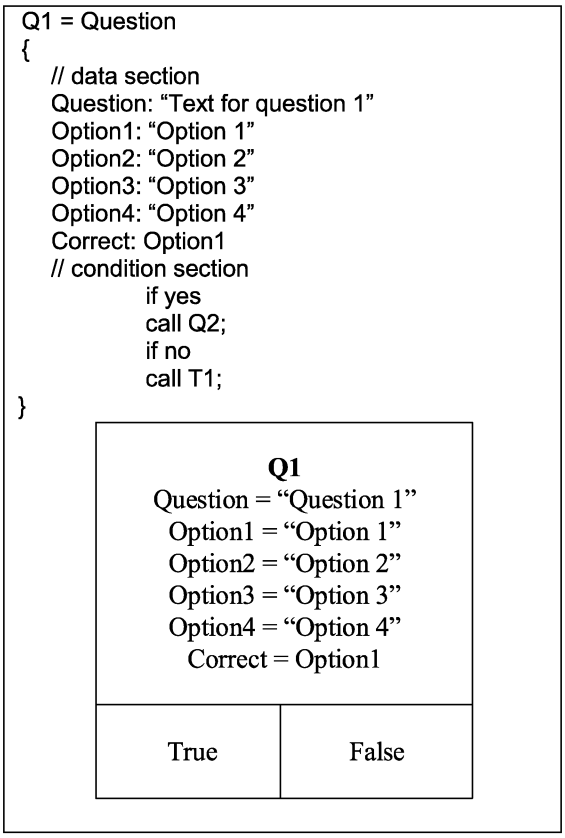

Figure 1. Question Object Type

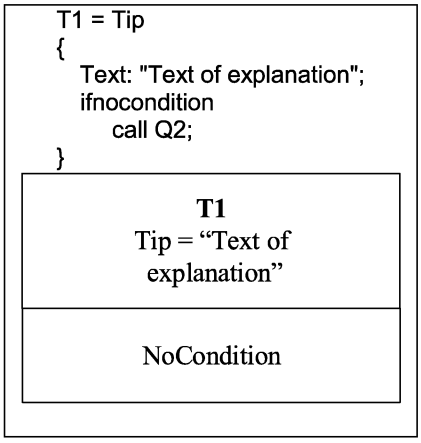

Figure 2. Tip Object Type

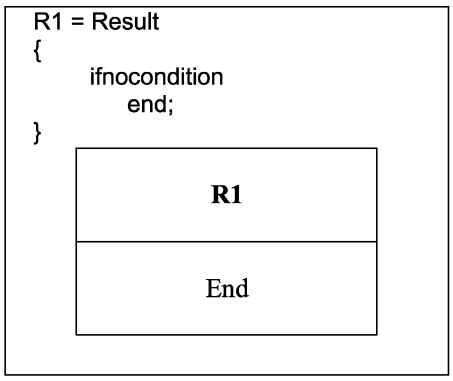

Figure 3. Result Object Type

The objects' description means the following. Object Q1 takes the question and variants of answers onto the screen. If the answer is correct, control is transferred to object Q2, otherwise the control function is given to object T1 for showing explanations. After object T1 shows explanations, control is also transferred to object Q3 (figure 4). After all questions and answers have been shown on the screen, control is then transferred to object R1 – show the final result.

Objects of the PDOP program can be represented not only in the form of a text, but as graphics as well (for example, figure 4). It consequently allows programming problem solution through a visual editor.

3 Self-Similarity

To make the code less abundant, PDOP uses the approach which was named "self-similarity". A user can form any sequence of objects as one complex object.

Description of scripts for automated teaching is a possible subject area in which we can use PDOP [2]. A few simple operations ought to exist: "show text material", "show image", "show video", etc. They are all similar and can be used to make the object "do a training lesson". For a knowledge test, it is needed to activate the objects "ask question", "show help", "show correct answer", "show result". These can well be used to write a more sophisticate script "build a test".

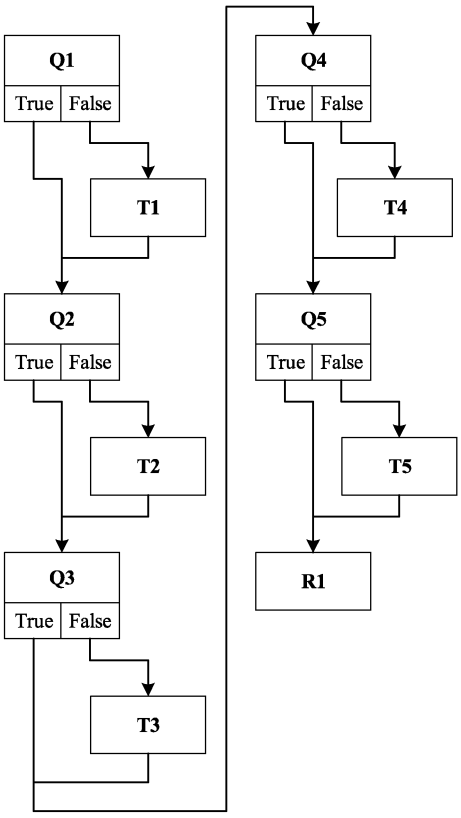

Figure 4. Graphical representation of the PDOP program objects

Using self-similarity makes it possible to avoid code duplication and re-use existing elements in the structure of more sophisticated objects, the fact that takes less time to write a program.

4 Conclusion

A specialist does not need to take into account an object's implementation arrangements. He or she should only know the function it performs and its section structure. For example, as it is shown in the article [5], we can describe the elements of structural programming with the help of PDOP. Their combination can generate objects that perform operations identical to such operations as a condition, a cycle with a precondition, a cycle with a post-condition, a cycle with a parameter. It allows writing scripts similar to methods in the OOP languages or the functions of structural programming. Application of such objects affords an opportunity to use PDOP for different ends and in different subject areas that require complicated operational algorithms.

Regardless of unlimited functionality of the OOP mechanism, it is not always easy or handy to work with this paradigm, which is especially challenging to an untrained user who lacks target programming skills. Using the described PDOP framework encourages the user to concentrate on commands effective for problem solution rather than to delve into conceptual details and the programming language.

Nowadays, PDOP is used to implement linguistic algorithms (algorithms for word-form generation and recognition [6] on the basis of a word-form building model [7]) and other algorithms.

References

1. V.A. Kamaev and V.V. Kosterin, *Tekhnologii Programmirovaniya* (2006) [Rus]
2. A.V. Prutzkow and D.M. Tsybulko, Vestnik of RSREU (Herald of RSREU), **45** (2013)
3. D.M. Tsybulko, Cloud of Science, **1**, 129 (2014) [Rus]
4. D.M. Tsybulko, *Student. nauch. soob-vo: issledovaniya i innovatsii – 2015*, pp. 117 (2015) [Rus]
5. A.V. Prutzkow, D.M. Tsybulko, Vestnik of RSREU (Herald of RSREU), **47** (2014)
6. A.V. Prutzkow, *2014 International conference on computer technologies in physical and engineering applications (ICCTPEA)*, p. 157 (2014)
7. A.V. Prutzkow, Cloud of Science, **1**, 88 (2014) [Rus]