Iryna Levdyk
G.M. Korotenko, L.M. Korotenko, research supervisors
N.M. Nechai, language adviser
National TU «Dnipro Polytechnic», Dnipro, Ukraine

## DevOps Culture and IT Education

Nowadays the increasing load of preparing students according to the modern state of IT-sphere, falls on Universities. This sphere is developing so rapidly that there exists a constant need to create software quicker, better and safer. But there is a response to this challenge in IT-sphere: a methodology called DevOps.

For the first time this term was introduced by Andrew Shafer and Patrick DeBois [1] during the discussion about Agile Infrastructure, at the Agile conference, which took place in Toronto, 2008.

In general, DevOps is a culture and practice of creating software, which is aimed at unification of software development (Dev) and software operation (Ops). The basic characteristic of DevOps culture is a strong support of automation and monitoring at all stages of software creation: from integration, testing, to deployment, infrastructure management. And, as a result, there are shorter development cycles, more rapid software deployment, safer launch in close relationship with business objectives. However, the leading IT-experts have pointed out that constant development of approaches, practices, tools and technologies which define DevOps infrastructure development is concentrated at its foundation, which is based on the code of software units, which are continually being developed, accumulated and improved. So, it is not surprising that work on standardization of the great range of program code specifications continues.

Meanwhile, the set of standards is currently developed under the aegis of organization Consortium for IT Software Quality (CISQ). The main direction of the Consortium activity is development and implementation of different standards for the targets, demands and norms, which are aimed at the development of the metric features, that provide the high quality and optimum size of software units. The complex of the standard quality indicators CISQ (or “CISQ Indicators”) is intended for providing security, reliability, effectiveness and convenience of the servicing of software for different levels of application. It is a set of metrics used for the assessment of the level of compliance with necessary requirements to the eighty-six (86) well-recommended rules of software creation.

The proposed metrics are included in the special analysis tools of the statistic analysis of the source code of the programs under study. They let identify critical shortcomings in the developed software for addressing these shortcomings.

Referring to any program component as the subject of work of different interacting groups of participants, it is reasonably to point that substantive contribution is made, primarily, by the developers (programmers, coders). That is why the standardization of writing the program source code, in the authors’ opinion, should significantly simplify all further stages of interacting with it.

This work is supposed to be done in universities within the framework of programming language courses. In particular, at the State Higher Education Institution "National Mining University" at the Department of Software for Computer Systems a Standard of teaching C++ for imperative programming is developed. The Standard is based on *Stanford University CS 106B Style Guide* [3] and *Google C++ Style Guide* [4]. From the Stanford University Standard authors used so-called "CamelCase" for variables initialization and functions description recommendation. The basic concepts of Google Standard were adopted in principle. Needless to mention, the elements of object-oriented programming were excluded.

Guided by the provisions of structural theorem of Bohm and Jacopini [5], the authors focused their attention on the basic structures of C++ language: consecution, branching and cycle. However, it was taken into account that the consecution-structure in C++ does not apply to the group of governing structures, which includes if, if/else, switch, for, while, do/while operators. That is why the commenting of these structures functionalities is put in the individual subsections.

The proposed approach was applied to teach students to think about the meaning of the entered structure, while commenting it. The special attention is also paid to the understanding the elements of the modular programming.

It is mentioned in the Standard that it is necessary to provide comments on function parameters assignment in its description as well as write the short shapes of the functions; follow "standard" in the naming of functions; follow the order of its parameters (functions with input data in the beginning and with output data at the end). Having reviewed the current state of IT industry, the authors reached the conclusion that in general there is a lack of initial phase in DevOps, namely, teaching undergraduate university students the standards of programming for the further use in developing programs in the IT organizations. The standards for different programming languages should be based on the top-down program design by the turn-based granularity and on the structured programming with using main, basic language structures, so that students could focus on the program design and construction, not on the features of programming language.

It is obvious that the best solution would be the consolidation of the best practices for the high-quality training of the future experts. To make the process gradual it is proposed to divide the programming standards into university and professional ones.

**References**:

1. http://www.jedi.be/presentations/IEEE-Agile-Infrastructure.pdf
2. G. M. Korotenko, L. M. Korotenko. The innovative role of programming standards in the development of the DevOps methodology. http://scaspee.com/all-materials/the-innovative-role-of-programming-standards-in-the-development-of-the-devops-methodology-g-m-korotenko-l-m-korotenko
3. https://stanford.edu/class/archive/cs/cs106b/cs106b.1158/styleguide.shtml
4. https://google.github.io/styleguide/cppguide.html
5. https://en.wikipedia.org/wiki/Structured_program_theorem