Asesorías y Tutorías para la Investigación Científica en la Educación Puig-Salabarría S.C.
José María Pino Suárez 400-2 esq a Lerdo de Tejada. Toluca, Estado de México. 7223898475
RFC: ATI120618V12

**Revista Dilemas Contemporáneos: Educación, Política y Valores.**
**http://www.dilemascontemporaneoseducacionpoliticayvalores.com/**
**Año: VI Número: Edición Especial Artículo no.:63 Período: Marzo, 2019.**

**TÍTULO:** Análisis de problemas de desarrollo del sistema de conversión de código.

**AUTORES:**

1. Dmitrij N. Galanin.
2. Victor O. Georgiev.
3. Vladislav M. Gorbunov.
4. Nikolai A. Prokopyev.

**RESUMEN**: Este documento presenta un análisis de los problemas actuales en el momento actual de la conversión de código de programa y sus soluciones prácticas con el uso de herramientas de desarrollo de software. Los principales problemas de ese dominio se revelan y sistematizan, y se dan algunas formas de resolver estos problemas. Se desarrolla y presenta una maqueta estructural del sistema "Convertidor de código". Este sistema presenta la traducción de código fuente de un idioma a otro con el uso de árboles de sintaxis abstracta y varias plantillas de utilidad para la emulación de construcciones de lenguaje, paradigmas de programación y funciones de biblioteca estándar, que no existen en los idiomas de destino.

**PALABRAS CLAVES:** conversión de código, maqueta de sistema, esquema estructural, herramientas y tecnologías.

**TITLE:** Analysis of Code Conversion System Development Problems.

**AUTHORS**:

1. Dmitrij N. Galanin.
2. Victor O. Georgiev.
3. Vladislav M. Gorbunov.
4. Nikolai A. Prokopyev.

**ABSTRACT**: This paper presents an analysis of actual at present time problems of program code conversion and their practical solutions with usage of software development tools. The main problems of that domain are revealed and systemized, some ways of solving these problems are given. A structural mock-up of “Code converter” system is developed and presented. This system features source code translation from one language to another with usage of abstract syntax trees and several utility templates for emulation of language constructs, programming paradigms and standard library functions, which are non-existent in target languages.

**KEY WORDS**: code conversion, system mockup, structural scheme, tools and technologies.

## INTRODUCTION.

The article introduces the mock-up version of the software code realization of the TCJ code converter system, which allows you to translate source codes from one programming language into another one without loss of quality, using an abstract derivation tree, various libraries that replace functional that is non-existence in the native libraries of another language, as well as a number of auxiliary templates.

The system was developed during the lecture and practical courses “Engineering and Architecture of Software Systems” and “Object-Oriented Analysis and Engineering” given to the students of the Volga Federal University in the direction of “Software Engineering” and “Applied Mathematics”.

## DEVELOPMENT.

### Methods.

Relevance and significance are determined by the fact that when developing software; there is often a problem of combining several subtasks created in different programming languages, which is sometimes time and cost consuming. One of the solutions to this problem is the conversion of software source code to other programming languages, which helps to reduce the time and economic costs in producing software. Our system uses the concepts presented in (Georgiev, et al. 2017; Georgiev, 2016; Opata, 2017) as the methodological basis, uses the concepts of abstract syntax derivation tree, lexical analyzer, code generator.

Currently, there exist relatively few analogs of our system. Among others, it is possible to single out JLCA, Convert.Net, Janett. Their peculiarity consists in a narrow focus on only one language when converting code (for example, only C # in JAVA or only JAVA in C #). The system that we offer is scalable to support the conversion of code for a variety of languages. The developed application will make it possible to perform the transformation of the text of a program, preserving the functionality to the max.

In programming, quite often there arises the task of rewriting the existing program code, implementing a certain algorithm, from one programming language to another. In this connection, it is advisable to automate this task (at least partially, within the framework of the syntax of languages).

To do it, consider the notion of the abstract syntax tree.

**The abstract syntax tree (AST)** is a finite, marked, oriented tree in which the inner points are juxtaposed with the operators and calls of the programming language functions, and the leaves to the corresponding operands. Thus, leaves are blank operators and represent only variables and constants.

Thus, AST is a structural representation of the source program, being cleared of the elements of a specific syntax. Abstract syntax trees in one form or another are used in almost all modern compilers and interpreters, their theory is well developed. One of the examples of practical code implementation of the theoretical concepts of AST is “TCJ Code Converter”.

**Results.**

**The TCJ Code Converter Structure.**

Structurally, “Converter” consists of two categories of components.

1. **Lexical analyzers** that use the source language to convert a program into AST.
2. **Code generators** that use the target language to convert AST into a program**.**

Each language applies its analyzer and generator.

A similar architecture is possessed by many modern compilers, which thus generate a machine code for different architectures (x86, x86_64, ARM)

In addition, “Converter” comprises a library to work with common data structures, a driver program that automatically calls up components for the required languages, as well as other auxiliary components.

**Technologies and Tools for implementation.**

The programming language Kotlin that runs on the platform JVM was chosen to implement “Converter”. This is due to the fact that:

1. The JVM platform provides ready-made tools for many “routines” tasks, which eliminates the need for both writing their own implementations and connecting exterior libraries.
2. JVM is available on all major platforms and architectures (Linux, Windows, macOS, BSD, etc.). The implementation of OpenJDK is available under a free license.

3. Assembly systems for JVM, such as those used in this Gradle system, provide convenient tools for managing project dependencies, including downloading them from the network, which greatly facilitates the assembly of the project from the source texts and its deployment on the user's machine.
4. Kotlin, developed by JetBrains, is not only fully compatible with Java, but also much simpler in syntax.
5. JVM is the main platform for the ANTLR4 library.

**ANTLR4** is a library aimed to generate syntactic and lexical analyzers directly from the description of the grammar in the Extended Backus-Naur format. These descriptions for many languages are available on the Internet under free licenses, which makes it possible to adapt them for the project without creating from scratch.

Gradle is used as the assembly system in the project. Each component of "Converter" is designed as a separate Gradle-module.

At the time of writing this work, only Kotlin and ANTLR4 are connected to the project as off-site libraries, so the task of project deployment is limited to the availability of JDK 1.8 on the target machine and the subsequent assembly of the project from the source codes.

**Issues connected with the process of System Development.**

The project is developed in IDE IntelliJ IDEA Community Edition, developed by JetBrains. The environment has the tools that make it easier to work using Kotlin. Also, the environment supports a variety of plug-in modules, in particular, support for the Gradle system, which allows to assemble directly from the IDE and greatly facilitates the development process.

Version control is carried out by means of Gitlab hosting, which uses the Git version control system. Application of VCS enables to develop together, with several developers.

**Discussion.**

In the course of the work related to the practical implementation of our project, a number of problems requiring solutions have been identified (Georgiev and Prokopiev, 2015; Polikashin, et al. 2016).

**Problems related to the Standard Language Libraries.**

Different programming languages have different sets of functions that are comprehended in their standard libraries, so the task of translating code that uses the functions of the standard library of a language into the language, the standard library of which does not comprise these functions, presents a certain complexity. In addition, the functions and operators that implement the same algorithm can have different names and / or semantics in different languages. An example is the operator ^, that denotes exclusive OR in C-like languages , and exponentiation - in BASIC. Since the AST does not preserve semantic information, this task also requires to be considered.

Part of this problem can be solved for the most common functions and operators by storing in AST not specific names of functions, but abstract notations that are translated into function names at the

**The problems concerning "Ideology" of the languages.**

Different programming languages, as a rule, are designed for different programming paradigms. Despite the fact that it is possible to write an algorithm in any total Turing programming language for any computable function, the implementation of paradigms supported in the source language, but not supported in the target language, can be quite complicated.

In particular, the problem is the translation between languages with static and dynamic typing. Equally challenging is the translation of languages that have first-class functions in the languages where functions are not a data type.

Besides, quite often when translating between syntactically and semantically dissimilar languages, the generated code becomes poorly readable, which is not an essential problem only if it is planned to use it as it is, without modifications.

**Standard compliance.**

Programming languages have been constantly improved, significant and even incompatible changes are introduced into their standards. Thus, the semantics of a programming language can strongly depend on the language standard.

In addition, compilers and other tools can provide extensions that are not a part of the standard. The examples of the C ++ language include the GCC (GNU Extensions) Extensions and the Windows-specific Visual C ++ Extensions.

In this connection, it becomes necessary to recognize or manually input the standard or dialect of the language into “Converter”, for both the source file and the desired output. This can be partially done by processing the source file directly (for the presence of new keywords, etc.), but in many cases this cannot be done (for example, in cases where only the semantics of a function, operator, or other syntactic structure has changed). Therefore, it is allowed for manual entry of the dialect of the language.

**CONCLUSIONS.**

The development has a relatively short time interval of its existence and is carried out practically as pilot. Some of the results obtained are described in Polikashin, et al. 2016; Polikashin and Georgiev, 2016; Pérez, 2018.

In the course of this work, a code conversion system was designed and partially built, analyzers and generators for C, Java and Python languages were implemented. The arisen problems being described above have been solved only partially, so this topic is still relevant for further research.

**Acknowledgements.**

The work is performed according to the Russian Government Program of Competitive Growth of Kazan Federal University.

**BIBLIOGRAPHIC REFERENCES.**

1. Georgiev V.O. (2016). Uchebno-modelnyj variant interaktivnoj sistemy generacii po slozhnyh system s predvaritelnoj predinterpretaciej programmnyh modulej (Educational model of interactive system for generation of complex software systems with premilinary interpretation of software modules). Informacionnye sistemy I tehnologii (IST-2016): materialy mezhdunarodnoj nauchno-tehnicheskoj konferencii (Information systems and technologies (IST-2016): international scientific and technical conference proceedings), Nizhny Novgorod. pp. 248-249. (In Russ.)
2. Georgiev V.O., Polikashin D.S., Rafikov D.S., Burnashev R.A., Prokopev N.A. (2017). Uchebno-maketnyj variant instrumentalnoj sistemy «Preobrazovatel koda TCJ» (Educational mock-up of software development system “TCJ Code converter”). Yazyki programmirovaniya i kompilyatory (PLC-2017): trudy konferencii (Programming languages and compilers (PLC-2017): conference proceedings), Rostov-on-Don. pp. 75-78. (In Russ.)
3. Georgiev V.O., Prokopiev N.A. (2015). Model Approach to Interactive System Software Development. International Journal of Applied Engineering Research (IJAER). Vol. 10. № 24. pp. 45208-45213.
4. Opata, C. E. (2017). An Assessment of Embedded Power Generation in Nigeria (Doctoral dissertation).
5. Pérez, E. (2018). De Armas, Cristian De Armas Iturriago, and Eudes Rafael De Armas. "Desempleo como violación de los derechos humanos al trabajo en Colombia." Opción 34.86 642-666.

6. Polikashin D.S., Enikeev A.I., Georgiev V.O (2016). Issledovanie problem avtomatizacii resheniya zadachi sovmestimosti programmnyh sistem (Researching of the problem of solution automation of software systems). Informacionnye sistemy I tehnologii (IST-2016): materialy mezhdunarodnoj nauchno-tehnicheskoj konferencii (Information systems and technologies (IST-2016): international scientific and technical conference proceedings), Nizhny Novgorod. pp. 251. (In Russ.)

7. Polikashin D.S., Georgiev V.O. (2016). Research and options of practical solution of software systems compatibility problem. International Journal of Pharmacy & Technology IJPT. Vol. 8. № 4 pp. 24309-24316.

**DATA OF THE AUTHORS.**

1. **Dmitrij N. Galanin.** Undergraduate, Department of Software Technologies, Kazan Federal University (KFU), Kazan, Russia.
2. **Victor O. Georgiev.** Ph.D. in technical sciences, senior lecturer, department of software technologies, Kazan Federal University (KFU), Kazan, Russia.
3. **Vladislav M. Gorbunov.** Undergraduate, Department of Software Technologies, Kazan Federal University (KFU), Kazan, Russia.
4. **Nikolai A. Prokopyev.** Master of technical sciences, assistant lecturer, Department of Software Technologies, Kazan Federal University (KFU), Kazan, Russia. E-mail: info@ores.su

**RECIBIDO**: 1 de febrero del 2019. **APROBADO**: 11 de febrero del 2019.