10. 4TH GENERATION PROGRAMMING LANGUAGES IMPACT

Kolbeko V.S.

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus

Bulavskaya T.V. – Senior Lecturer

This paper provides common information about 4th generation programming languages. The main purpose is to depict positive and negative impacts that 4GL tools have on different software development spheres.

4th generation programming language, or 4GL, refers to a high-level language or environment, that makes software application development faster, easier and more user-friendly for all users despite their programming skills. 4GLs are designed to be more efficient and effective, as they allow developers to focus more on the inner logic of application and less on application implementation.

The main features of 4GL are the following:

1. nonprocedural programming – the programming style that defines what problem is to be solved rather than how this problem is to be solved [1];
2. problem-oriented language – the language designed to handle a narrow class of similar problems;
3. end-user development – the set of tools that allows end-users – users with less or none programming skills – to create and customise their own software applications;
4. Rapid Application Development (RAD) – the methodology of focusing on quickly prototyping and iterating of software applications;
5. productivity – decreased time consumption and increased efficiency of software development.

The difference between generations of programming languages can be clearly seen through the comparison of two adjacent generations. The results of comparing 3rd (3GL) and 4th generations of programming languages are presented in Table 1 [2]. The results show that 4GLs provide developers with high-level abstraction tools, allowing easier and faster application development, but with less flexibility than 3GLs provide.

Table 1 – Comparison of 3rd and 4th generations of programming languages

	3GL	4GL
Abstraction level	High-level abstraction, low-level interactions are enabled	Higher-level abstraction, no low-level interactions are enabled
Ease of use	Requires syntax knowledge and basic programming skills, may be hard to use for new users	Low or no programming skills required, user-friendly interface, end-user development is possible
Versatility	Open-ended tools	Problem-specific tools
Performance and code volume	Better performance, increased code volume, harder to debug and maintain	Worse performance, decreased code volume, easier to debug and maintain
Development time	Increased due to greater code volume	Decreased due to less code volume

Typically, 4GLs are classified by the types of problems they are designed to solve. Common classification of 4GLs includes the following categories (with some of the examples given):

- general use languages: dBase, R:Base, PowerBuilder, Xojo;
- database query languages: SQL, FOCUS, NATURAL;
- report generators: Oracle Reports, RPG;
- software creation languages: LiveCode, SuperTalk;
- mathematical optimisation languages: Wolfram Mathematica, MATLAB;
- web development languages: CFML, LANSA, Wavemaker.

All of the listed programming languages support declarative programming paradigm. Other programming paradigms may be supported thanks to the prevalence of multiparadigm programming languages.

Some of the programming languages can combine features of different generations. For instance, general-purpose programming language Delphi is typically considered a 3rd generation programming

language. This is caused by Delphi's approach to console application development and supported programming paradigms – imperative, functional and object-oriented. Nevertheless, Delphi's tools for Graphic User Interface (GUI) software development, that are depicted in Figure 1, can be classified as 4th generation programming language tools, as they do not require high programming skills and allow implementing RAD methodology.

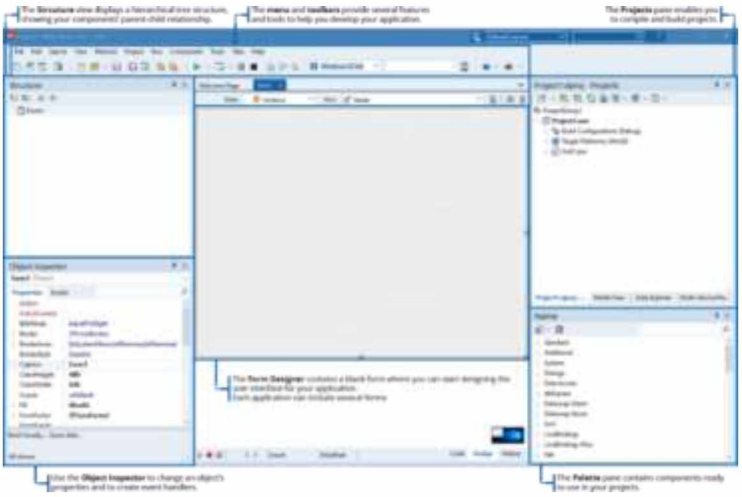

Figure 1 – Delphi integrated development environment for GUI applications

4GLs are widely used for general use software development, database creation and administration, scientific and statistical purposes. 4GLs have both positive and negative impacts in each sphere of application.

Proliferation of 4th generation programming language technologies made general purpose software development significantly faster and easier for maintenance due to automatisation of repetitive tasks, simplification of complex operations and reduction of the amount of code that needs to be written. Some of 4GLs support database integration and cross-platform nature, which can make back-end and mobile development more efficient. Database creation and administration using 4GLs simplify database-related tasks accomplishment, as developers are able to use declarative syntax and visual tools to define database structures, queries, and data manipulation operations with no need to write complex code. Taking into account the fact that non-professionals may be a part of database interactions, ease of use 4GLs tools can positively affect productivity levels. Scientific and statistical implementation of 4GLs has such benefits as a wide range of built-in functions (mathematical operations, statistical analysis tools and data manipulation functions), ability of rapid prototyping and data visualisation. Taking into consideration the need of being used by scientists, that are not familiar with programming, user-friendly interface of 4GL tools may make these tools the most efficient.

Despite all the positive impact 4GLs implementation has, its negative impact is similar in all spheres of use and may be crucial for some software applications. Using 4GLs involve limited flexibility, leading to restrictions in control over low-level system operations, memory management, and hardware interactions compared to programming in lower-level languages. 4GLs implementation can also become a reason for performance overhead [3].

To sum up, 4th generation programming languages have a significant impact on modern software development by making the development process more efficient, productive and less time-consuming. In the perspective of further 4GLs advancement such technologies as artificial intelligence, machine learning and natural language processing may be applied, which can lead to the 5th generation programming languages restoration. Even though attempts to create a 5GL in the past did not turn out well [4], nowadays with the use of 4GLs implementation experience and more advanced technologies these attempts may be able to succeed, which can result in further software development improvement.

References:

1. A Functional Model for Fourth Generation Languages [Electronic Resource]. – Mode of access: https://www.nist.gov/publications/functional-model-fourth-generation-languages-0. *– Date of access: 11.03.2024.*

2. An experimental analysis of the performance of fourth generation tools on PCs [Electronic Resource]. – Mode of access: https://dl.acm.org/doi/10.1145/68814.68819. *– Date of access: 11.03.2024.*

3. A contingency approach to the application software generations [Electronic Resource]. – Mode of access: https://dl.acm.org/doi/abs/10.1145/126743.126749. *– Date of access: 13.03.2024.*

4. Programming Language [Electronic Resource]. – Mode of access: https://www.ijcrt.org/viewfull.php?&p_id=IJCRT1134755. *– Date of access: 14.03.2024.*