# History of programming research paper

Technology, Development

## Introduction

Computer programming is a growing industry today, especially considering that technology is an essential part of people’s everyday lives. With new applications being constantly developed to cater to people’s different needs, programming languages also undergo changes, which allow programmers or software developers to meet the increasing complexities of these modern-day applications.

With the long history behind the development of programming languages, which continues even to this day, this paper aims to present a brief overview of this history where some of the important milestones are highlighted.

## Origins of Computing and Programming

After the invention of the Chinese abacus and other devices, such as the Pascaline, which was built by Blaise Pascal, and the Difference Engine, which was built by Charles Babbage, Babbage made a proposal for a completely programmable machine in 1837, which he called it the Analytical Machine (Gosch 1). Although Babbage never got to build this machine, it would have been similar to modern-day computers in that it separated the storage of its program data from its input data. It also had a central processing unit and a printer output data.

The Countess of Lovelace, Ada Augusta King, would later write programs for the machine. She also wrote a program that computed Bernoulli-Numbers. In her programs, she made use of concepts that are common in modern-day programs, such as conditional jumps, loops, reusable procedures, and subroutines.

In 1945, John von Neumann would introduce the concepts of shared-program technique and conditional control transfer (Gosch 1). The shared-program technique stated that a computer must be freely programmable and must not need different hardware specifications to run different programs. On the other hand, conditional control transfer referred to the capability of computers to store codes in small blocks that did not need to be run sequentially but that could be run in accordance to the conditions stated in the code.

With the ENIAC (Electronic Numerical Integrator And Computer) being capable of running only one specific program with a particular hardware configuration, von Neumann's ideas were used to develop other computers, such as the EDVAC (Electronic Discrete Variable Automatic Computer) and the ORDVAC (Ordinance Discrete Variable Automatic Computer), which were fed with zeros and ones on punching cards in order to be programmed (Gosch 1).

In 1949, the first programming language that was not intended for writing machine code was developed (Gosch 2). It was called the Short Code and it had basic arithmetic features, as well as features for making calls to functions and for branching. However, it required manual compilation into bits and was around fifty times slower than the compilation of machine code.

## Higher-Level Programming Languages

In 1951, the first compiler A-0 was developed by Grace Hopper, and this would lead to the development of the higher-level programming languages that are currently in use today (Gosch 2). With higher-level programming, the programmer does not need to be concerned about the computer hardware and can focus on programming in order to solve a problem. However, since the code must first be translated into machine code before it is run, the computing time becomes slower than when running machine code.

In 1957, FORTRAN (Formula Translation) was developed. It was the first major programming language and was mainly used for scientific computing, such as the Space Program and military projects (Gosch 2). It was the first programming language to make use of data types, such as the double-precision number, the real number, the integer, and the Boolean. It also led to the development of more dialects, which were based on FORTRAN concepts, and which were kept updated to the present day.

FORTRAN was followed by ALGOL (Algorithmic Language) in 1958. ALGOL was also developed for scientific computing. However, unlike FORTRAN, ALGOL used the formal grammar called Backus Nur-Form (Gosch 2). It was also the first language that used a block structure.

Despite FORTRAN's computing capabilities, it lacked a proper input/output processing capability, which led the CODASYL (Conference on Data Systems and Languages) to develop COBOL (Common Business Oriented Language) in 1959. It was intended specifically for the business sector and recognized only numbers and strings of characters, which were grouped into arrays and which served well for handling data. Although this programming language was considered a short-term solution to the computer problems encountered in the 1950s, it continues to be used up to this day because of existing legacy systems.

Still, another programming language that was developed in the 1950s –

particularly in 1958 – was Lisp (List Processing) (Gosch 2). It was different from the other programming languages in that its data structure consisted mainly of linked lists and even the code consisted of lists. This allowed for the language to be easily manipulated. In addition, it was associated with artificial intelligence research.

As the amounts of code being written increased and as the software projects became more complex, problems arose with regards to cost overruns, schedule overruns, and the poor quality of software projects. In response, German engineer Friedrich Bauer called for the establishment of a software engineering field during the 1968 NATO Conference (Gosch 3).

Code modularization was also proposed in this conference. This concept enabled the easier maintenance of codes and enabled certain portions of a program to be rewritten without having to rewrite the entire code. It also enabled computer programmers to collaborate with each other on software projects. As well, this led to the development of object-oriented programming in the 1960s, although it would be widely employed only in the 1990s. It is a type of modularization that involves the use of a class, an object, and a method (Gosch 3).

Until the late 1970s, most programs were run on large computers, which made programming tasks accessible only to a few. However, with the introduction of home computers, it became possible for everyone to write and run programs., and this gave rise to the development of programming languages such as Pascal and Basic (Gosch 3).

According to Sammet (602), one of the reasons for the proliferation of higher-level programming languages is the development of a completely new language or the identification of a new application area that would benefit from having its own language. A new language may also be developed in order to correct the deficiencies of an existing language or to combine the best features of several languages. In the same regard, it may be easier to develop a new language that has the desired features and capabilities rather than to modify or extend an existing one. Moreover, developers may develop a new language in order to gain profit from it or because they have “ personal preferences and prejudices against the existing languages” (Sammet 603) even when these languages already have the capabilities that they need. Similarly, developers may create a new programming language because they are unaware that an existing language can already cater to their needs.

In the same regard, Zepcevski (iii) contends that one of the reasons behind the proliferation of programming languages is the complexity that arises from the interactions among the people involved in a project; the scale and scope of the program; and the complexity of the algorithm (Zepcevski 12). In addition, she indicates that the inability to verify the correctness of a program is another reason for the development of new programming languages. In this case, software verification refers “ to the broader practice of verifying that a program is accurate to its specifications, including both using mathematical proofs to deduce the accuracy of and practical testing techniques” (Zepcevski 20).

Today, there are thousands of programming languages from which a programmer can choose. In this regard, the appropriate language for a software project would depend on the programmer’s inclination, the project’s scope, and the existence of legacy software. However, some of the most popular programming languages in use today are the C language, Java, and domain-specific languages (Gosch 4).

The concepts used in the C language were based on Pascal. It was much less readable compared o Pascal but it also computed faster. It is mainly used with UNIX operating systems and has evolved to other forms, such as C with classes, which was developed in 1980, and C++, which incorporated the concept of object orientation. On the other hand, Java was developed by Sun Microsystems in 1990. It is popular among programmers due to its portability. However, its computing speed is slow and with its resemblance to the C language, it is also quite unreadable. Lastly, domain-specific languages provide programmers with more capabilities that are specific to the type of application they are developing. Examples include scripting languages, such as PHP; printing and graphical languages, such as PostScript; and database-query languages, such as SQL. They are often used with general-purpose languages (Gosch 4).

## Conclusion

This paper discussed that the development of programming languages started in the early nineteenth century with Charles Babbage’s proposal for the Analytical Machine, which was supposedly a completely programmable machine. Although Babbage was not able to build the machine, his ideas inspired others to continue the work the started.

For example, Countess Ada Augusta King wrote programs for the machine using concepts that are still in use today. John von Neumann also developed the concepts of shared-program technique and conditional transfer, which referred to the flexibility of machines for running various types of programs and to the capability of machines for running conditional instructions, respectively.

1949 saw the development of the first high-level programming language called the Short Code. However, it required manual compilation. In response, Grace Hopper created the first compiler A-0 in 1951 and this would lead to the development of higher-level programming languages, such as FORTRAN, ALGOL, COBOL, and Lisp.

Needless to say, as technology continues to advance and as software applications become more complex, more and more programming languages can be expected to emerge.

## Works Cited

Gosch, Sebastian. “ A Short History of Programming Languages.” campar. in. tum. de. Lehrstuhl für

Computer Aided Medical Procedures & Augmented Reality, 2007. Web. 9 March 2013 .

Sammet, Jean E. “ Programming Languages: History and Future.” Communications of the ACM 15. 7

(1972): 601-610.

Zepcevski, Joline. “ Complexity & Verification: The History of Programming as Problem Solving.” Diss.

content/uploads/2012/09/FinalDissertation2012B. pdf>.