Introduction

The terms "objects" and "oriented" in the field of computer science, programming in-particular, were first appeared at MIT during 1950s and early 1960s. As early as 1960, "object" could refer to the items identified (LISP atoms) with properties; later in 1966, Alan Kay cited the detailed understanding of the internals of LISP as a strong influence on his thinking. Another early example was Sketchpad in 1960-61 at MIT, created by Ivan Sutherland; in the glossary of the 1963 technical report based on his research report about Sketchpad, he defined notions of "object" and "instance" even though specialized to graphical interaction.

As a formal concept in programming, Objects were introduced in 1960s in the programming language, The Simula 67, a major revision of Simula I, designed for the simulation of discrete events, created at Norwegian Computing Center in Oslo by Ole-Johan Dahl and Kristen Nygaard. Simula introduced the representation scheme for classes and instances or objects (including subclasses, virtual methods, co-routines, and discrete event simulation) as part of an explicit programming paradigm. The automatic garbage collection that had been invented earlier for the functional programming language Lisp was also used by the language. The ideas of Simula 67 had their impact on many later languages, including Smalltalk, derivatives of LISP (CLOS), Object Pascal, and C++.

The term object oriented programming was coined in the language, The Smalltalk language, which was developed at Xerox PARC (by Alan Kay and others) in the 1970s. The objective of this was to represent the pervasive use of objects and messages as the basis for computation. It is apparently true that the inventors of Smalltalk were biased towards the ideas introduced in Simula67, but it is also a fact that Smalltalk was designed and expected to be a dynamic system in which classes could be created and modified dynamically rather than statically as compared to Simula67. Smalltalk and with it OOP were introduced to a broader segment of audience in Byte Magazine in its issue published in August 1981.

Object-oriented approach has its foundation on a sound engineering foundation elements of which are collectively called, the object model. This object model includes the principles of
abstraction, encapsulation, modularity, hierarchy, typing, concurrency, and persistence which is not new but the important aspect about the object model is that these elements are brought together in a synergistic way. This is true that fundamentally, object-oriented analysis and design is different from that of the traditional structured design approaches: In all it requires a different approach, way of orientation and thinking about decomposition process. Not only this, but it also produces software architectures which are generally beyond the boundaries of the structured design approach. The above differences result from the fact that structured design methods have their roots in structured programming, whereas object-oriented design finds its roots in object-oriented programming. Structured design methods evolved to guide application developers who were involved in building complex systems using algorithms and methods as their fundamental building blocks. Likewise object-oriented design methods were evolved to facilitate application developers exploiting the expressive power of object-based and object-oriented programming languages, using the class and object as basic fundamental building blocks. In fact, it is not just only object-oriented programming but there are other factors also which influence object model. Indeed the object model has proved to be a unified concept in computer science, applicable not just to programming languages, but also to the design of user interfaces, databases, and even computer architectures. The reason for this broad application area is simply that an object orientation helps us to cope with the complexity inherent in many different kinds of systems.