What is software?

They are programs that are necessary for both the management of the hardware units that make up the computer system and for the users to do their work. A computer system without software is a device consisting of some electronic cards, cables and some mechanical parts. A computer system can perform the necessary functions after the operating system and other software are loaded and run on it.

Computer software can generally be examined in 3 main groups.

- System Software
- Application Software
- Development software

System Software (System Software); It is software such as enabling the computer to operate itself, operating system, compilers (software program that converts the written program into machine language), various accessories (facility).

Application Software (Application Software); When this solution provides a solution to their work, programs that write bills, stock control, payroll, library records, programs that make user money accounts in banks, etc. are software like.

The software, which can be examined in 3 groups in general, is divided into various fields today, in fact, it has turned into a sector on its own. For example, security, backup, planning and job tracking, simulation, CAD CAM software, although in essence they are all application software, they have all turned into separate areas of expertise.

Who is a Software Engineer?

Basic knowledge

The foundations of software engineering consist of theoretical and scientific, mathematical foundations that describe the characteristics of the products produced by software engineering and the main principles that produce predictable results. The main point here is that optimal modeling can be done by applying engineering design and engineering science to transform resources into a specified purpose.

Professional Application

Professional practice, which aims at development of thinking rather than technical skill development, is about the knowledge, skills and behaviors that software engineers must have in order to perform their application professionally and ethically.

Identifying Needs

To be able to design solutions by harmonizing user needs with existing technologies.

Software Design

As the name suggests, it is about techniques, strategies, representations and patterns. It should conform to functional requirements, taking into account constraints such as design, resources, performance, reliability and security. In addition, internal interfaces between software components, architectural design, data design, user interface design, design tools and design evaluation are also within the scope of this field.

Software Creation

It is the documentation stage of the design of the software. Operation, technique etc. Contains information in terms of.

Software Development

It covers the process of supporting the software after the software starts to be used. This stage, which includes stages such as removing the deficiencies of the software, testing and improving it, is an important stage that requires cost.

Software Quality

The quality of the software includes criteria such as usability, reliability, safety, suitability for maintenance, flexibility, efficiency and performance.

Software Management

It covers the monitoring and control of all stages in the application area so that the software can continue and survive effectively after its use. The success of software development projects is critical for the coordination of the works in different branches, the maintenance of software versions, the availability of resources when they are needed, the appropriate division of the work in the project, and the facilitation of communication.

These definitions, which we can define as the qualifications required by software engineering, play an important role in determining the curriculum in undergraduate and graduate software engineering education in our country.Yazılım Dillerinin Tarihi:

Programming Languages

All application and system software, whether general purpose or special purpose, are written in programming languages. A programming language is any group of symbols, characters and rules that allow people to perform various operations on the computer. Programming languages act as interpreters between humans and computers. Programming languages contain statements and commands that tell the computer what to do, when and how. Programming languages have developed over time and have become very capable today. Here, the development process of programming languages will be briefly examined.

Generation Programming Language Period

Historical Development of Programming Languages

1 Machine language from 1940 to 1950

2 Assembly language since the 1950s

3 High-level languages from the 1960s

4 Very high-level languages since the 1970s

5 Languages for artificial intelligence since the 1980s

Historical Development of Programming Languages

Low-level languages: Programming languages in the early years of computers were machine language and assembly languages that were very difficult to use. Machine language is the first programming language developed and represents the first generation. All commands written in machine language consist of 0s and 1s. All instructions are described in the most detailed way and coded using the binary number system.

Assembly languages: They are the second generation languages and their development was pioneered by Grace Hopper in the early 1950s. Machine language instructions in these languages are expressed in symbols that can be more easily understood and remembered. Despite being considered a group of low-level languages, assembly languages, which are one step ahead of

machine language, are used by programmers even today. Because programs written in these languages generally run very fast and require less storage space. However, programming with these languages loses its appeal because it is very tiring, tedious and time consuming.

High-level languages: High-level programming languages, also called third generation, are easier to learn, less time-consuming to write programs, and provide better results. A source program written in one of the higher-level languages must be translated into machine language. The program that performs the conversion is called a compiler. The Basic, Cobol, Fortran, and Pascal languages are a few examples of programming languages in this category.

Very high level languages: These languages have made programming much easier. The main feature in these languages is that they allow users to express what to do to the computer, not how to do it. For example; It is necessary to write a program of 15-20 lines using complex logic with a high-level language to be able to sequence the numbers. Conversely, the job a user has to do with many high-level (fourth generation) languages; To determine the area to be sorted, whether the sorting will be done from small to large or from large to small, and to perform the sorting operation by clicking an icon on the screen or choosing the "sort" command from a menu.

Even between the first developed fourth generation languages and the languages developed later, there are great differences in terms of the procedures followed and the methods used. Newly developed languages allow for much more complex jobs to be done more easily.