Software engineering has emerged as an important field within computer science. Indeed, there is a relationship between it and many other areas in computer science: these areas both influence and get influenced by software engineering. In the following subsections, we explore the relationship between software engineering and some of the other important fields of computer science.

**Programming Languages**

The influence of software engineering on programming languages is rather evident. Programming languages are the central tools used in software development. As a result, they gave profound influence on how well we can achieve our software engineering foals. In turn, these foals influence the development of programming languages. The most notable example of this influence in recent programming languages is the support of modularity features, such as separate and independent compilation and the separation of specification from implementation in order to support team development of large software. The Ada programming language, for example, supports the development of packages allowing the separation of the packages interface from its implementation and librarian of packages that can be used as components in he development of independent software systems. This is a step towards making it possible to build software by chord from a catalogue of available components and combining them similar to the way hardware is built. Design of object oriented programming languages like C++ in another major development for one purpose of rapid software development.

**Operating Systems**

The influence of operating systems on software engineering is quite strong primarily because operating systems were the first really large software systems built, and therefore they were the first instances of software that needed to be engineered . Many of the first software design ideas originated from early attempts at building operating systems.

Examples of the influence of software engineering techniques on the structures of operating systems can be seen in portable operating systems and operating systems that are structured to contain a small protected kernel that provided a minimum of functionality for interfacing with the hardware and a non-protected part that provided the majority of the functionality previously associated with operating systems. For example, the non-protected part may allow the user to control the paging scheme, which has traditionally been viewed as an integral part of the operating system.

Similarly, in early operating systems, the command language interpreter was an integral part of the operation system. Today, it is viewed as just another utility program. This allows, for example each user to have a personalized version of the interpreter. On many UNIX systems, there are at least three different such interpreters.

**Database**

Database represent another class of large software systems whose development has influenced software engineering through the discovery of new design techniques. Perhaps the most important influence of the database field on software engineering is through the notion of data-indigence, which is yet another instance of the separate of specification afro implementation. The database allows applications to be written that user data without worrying about the underlying representation of the data.

Another interring impact of database technology on software engineering is that it allows database system to be used ac components of large software systems. Since databases have solved the many problems associated with the management of concurrent access to large amounts of information by multiple users, there is no need to reinvent these solutions when we are building a software systems – we can simply use an existing data – base system as a component.

One interesting influence of software engineering on data-base technology has its roots in early attempts to use databases to support software development environments. This experience showed that traditional database technology was incapable of dealing with the problems posed by software engineering processes. For example, the following requirements are not handled well a by traditional data bases : strong large structured objects such as sources programs or use manuals; storing large unstructured objects such as object code and executable code; maintaining different versions of the same object; and strong objects, such as a product, with many large structured and unstructured fields, such as source code, object code, and a user manual.

There is presently considerable work going on in the database area to address such problems, ranging from introducing new models for databases to adapting current data-base models

**Artificial Intelligence**

Artificial intelligence is another field that has exerted influence on software engineering. The technique supported by artificial intelligence include the user of logic in both software specifications and programming languages.

The logic orientation seems to be filling the gap between specification and implementation by raising the level of implementation language higher than before. The logic approach to specification and programming is also called declarative. The idea is that we declare the specifications or requirements rather than specifying them procedurally; the declarative description is then executable. Logic programming languages such as PROLOG help us follow this methodology.

Software engineering techniques have been used in newer artificial intelligence systems – for example systems. These systems are modularised, with a clear separation between the facts known by the expert systems and the rules used by the systems for processing the facts – for example, a rule to decide on a course of action. The separation has enabled the building and commercial availability of expert system shells that include the rules only. A user can apply the shell to an application of interest in supplying application-specific facts. The idea is that the expertise about the application is provided by the user and the general principles of how to apply expertise to any problem are provided by the shell.

A different kind of symbiosis is currently taking place at the intersection of software engineering and artificial intelligence. Techniques of artificial intelligence are being applied to improve the software engineering tasks. For example, programming assistant are being developed to act as consultants to the programmer, watching for common programming idioms or the system requirements. Such assistants are also being developed to help in the testing activities of the software development, to debug the software.

The problem of providing interfaces for non expert uses – for example, through the use of natural languages – was first attacked by artificial intelligence. Cognitive models were also under to model the user. These works have influenced the very active area of user-interface design in software engineering.