
1.5 Additional reading

Description

The goal of this book is to introduce programming as modeling and after reading and understanding the material here, the reader should be able to write small programs and understand the basic elements of modeling.

If you want to be able to solve complex problems using a computer and/or to be able to develop large complex software systems, there are a number of other disciplines you need to study. Here we list the most essential ones:

Algorithms and data structures

As mentioned in section , algorithms are central to programming. In this book, we present a number of algorithms, but most of them are quite simple and straightforward. For a large class of problems, it may be a challenging task to develop algorithms for solving the problem.

One issue in developing an algorithm is that it may be difficult, and time consuming to design an algorithm and validate/prove that it actually solves the problem. Another issue is that the algorithm may need to be efficient with respect to the time it takes to compute the algorithm. For some kind of problems (especially interactive programs), the time to compute must be a fraction of a second such that user does not experience a long response time. For other kinds of problems, a longer response time say seconds, minutes, or hours may be acceptable, but rarely days, weeks or more. Although there are examples of the latter.

The efficiency of an algorithm often depends on the amount of data to be processed. Sorting a small list of data records may take no time, but millions, trillions, etc. of data records may take easily take a very long time. The representation of data in the form of data structures may have high influence on the efficiency of the algorithm to process the data just as it may be an issue how much space the data takes up on the computer. One issue here is whether the data can be stored in the RAM of the computer or whether part of it must be stored on a disk.

The study of algorithms data structures is an important subject, and the reader is encouraged to invest time in this.

Software development

Many software systems consist of millions of lines of code and often more. The systems may be very complex and often one person is not able to understand all details / corners of a given system. There are a number of disciplines that one needs to master to become a good software engineer.

Software architecture is about how to structure a software system logically and physically. What are the components of the system and what are their relations and interfaces between the components. How to split the software into manageable modules such that people can work on different modules at the same time.

Software testing is about how to develop and organize tests of components in the system and the system as a whole. Most (not to say all) systems develop over time and have errors. When correcting errors, it is important to be able to test that an error correction does not introduce new errors and the same for introducing new functionality.

Version and variant control are about keeping track of *different versions* and *variants* of a software system. Evolution of a systems lead to new versions, and one may often have to keep track of multiple versions since different users may use different versions. Often a system exists in different *variants* where each variant has some components that differ from the other variants. This could be the case if a system is available for computers running Windows, MacOS and Linux where some components are dependent on the underlying operating system and thus must exist in different variants.

Software methods and techniques is about how to organize the process of developing a software system. This includes organization of the project into phases, defining milestones, deliverables/releases, deployment of the system, maintenance, documentation, setting the team, etc. It is about whether to use an agile approach, prototyping, participatory design, or

other methodology.

Human computer interaction

Most computer-based systems need an interface to the people using the system. Human computer interaction focuses on the interface between people (users) and computers. It is about how people interact with computers and how to design systems that support the need of its users and makes it easy and efficient to use the system.

It is also about how computer/information technology influence people and society including ethical aspects. It is about how to obtain a proper balance between computer support and traditional work. It is not just related to work, but also how computers influence the private life of people.

Participatory design as mentioned above is an important technique for ensuring that a system fulfils the needs of its users. It is about how to involve users from the beginning to the end of a project.

Other topics

There are many other topics that are relevant if you want to become an IT-expert. Below we shortly list some of these – we may expand these texts in a future version of this book.

Graphical user interfaces

Graphical user interfaces (GUI) are about programming the interface to a program using windows and menus and I/O devices like screen, keyboard and mouse.

Programming of websites is part of this topic.

Computer Graphics

Computer graphics are about writhing user interfaces presenting information in 3 dimensions (3D). This is relevant for writing games, visualization of complex data, etc.

Parallel and distributed systems

A computer-based system may consist of several computers where each computer may have many so-called cores that can execute programs fragments at the same time. The notion of *parallel system* refers to the fact that several executions may take place at the same time. The notion of *distributed system* refers to the fact that the computers of a given system may be placed at different locations. We do, in this book, present language mechanisms for writing parallel programs.

Operating systems, embedded systems, cyber physical systems, Internet-of-Things

Programming in a high-level programming language is in most cases independent of the actual hardware (computers and other devices) executing the programs. To make this possible there is a layer of programs that control the physical hardware of a given system. Such programs are often called an *operating system*. The Mac OS, Windows System, Linux, etc. are examples of operating systems.

In addition to computers like PC's and Macintosh's, and smart phones and tables, our environment includes a lot of devices controlled by computers that are built-into the devices. Examples, are cameras, heating systems, vehicles, production lines/units, etc. The programs in such devices are often referred to as *embedded systems* and like operations systems, are close to the hardware and controls these devices.

Many modern embedded systems consist of several physical units that collaborate. This may med be units in a production system in a factory, or a rail system with signals, trains, and other devices. Such systems are called *cyber physical systems*.

Finally, the term *Internet-of-Things* (IoT) covers the any devices, sensors and actuators that are connected to the Internet. This may include video-cameras, temperature sensors, weather stations, etc.

Numerical analysis

A computer may only represent a subset of real numbers. This means that it requires special techniques to compute functions involving many real numbers.

Compiler construction

As mentioned, the built-in language of a computer – the machine language, is very simple and not suited for people to use for programming. Most programmers use high-level language for programming. A compiler is a program that can translate a high-level program into machine language.

Security and cryptography

Cybersecurity is a major issue in today's era of digitization. Security of it-systems including how to encrypt data is thus an important topic.

Artificial intelligence and machine learning

Artificial intelligence and machine learning is a new hot topic and for all people dealing with it-systems, it is a good idea to obtain some familiarity with these topics and/or study or become and expert.