Finally we've arrived at a state where theory and ideas can be turned into practice. The starting point for this? When a customer arrives on our doorstep, asking us to develop a software system for his application. It finishes when the delivered software runs reliably, correctly and safely in the target system (cynics may argue that by this definition, most jobs are never finished). However, what concerns us here is the piece that fits between the two end states, the design and development phases. Just how do we go about this process? What methods should we use? What tools are available? How can we best use these tools? These, and others, are the questions tackled in the next two chapters.

There are two major steps in the development process. The first involves translating the customer’s requirements into source code. The second concerns the transformation of this source code into fully functional target system object code. An ideal software development toolset supports both stages. Unfortunately, this is a rare item, particularly for the support of ‘bare-board’ original designs. Either tools are intended for front-end use, frequently assuming that the second stage is trivial. Or else they are aimed at the back end of the development, neglecting software design aspects entirely. Chapter 11 deals with development methods and tools designed for use in the second-stage processes. However, in this chapter we look at:

- What the overall specification-to-coding process involves.
- What it means to execute the various steps within this.
- How different techniques achieve the same ends in quite different ways.
- What the unifying themes are within the various techniques.
- What the important factors are when selecting toolsets.

Specifically, the methods of CORE, YSM, JSD, MASCOT and HOOD are described here.
10.1 THE DEVELOPMENT PROCESS

10.1.1 General description

In this chapter the complete development process is viewed as a series of distinct steps, consisting of:

- Requirements analysis.
- Requirements specification.
- Architectural design.
- Physical design.
- Implementation – to source code level.
- Test, integration and debug.

(a) Requirements analysis
Here the purpose is to establish precisely what the system is supposed to do. In small systems it tends to get merged with other parts of the development process; in large systems it is an essential, separate activity.

(b) Requirements specification
This describes what the software must do to meet the customer’s requirements. It is based on information obtained during the analysis phase.

(c) Architectural design
This stage is concerned with identifying and modelling the software structure, using information supplied in the requirements specification. It defines the essential software components of the system, how these fit together and how they communicate.

(d) Physical design
Here the architectural structure arrived at in the previous stage is partitioned to fit onto hardware. In a small, simple system this is a trivial task. But, in multiprocessor designs, distributed systems, and designs using ‘intelligent’ interfacing devices, it is a critical activity.

(e) Implementation
Ultimately the software tasks are expressed as a set of sequential single-thread program structures. The function of the implementation stage is to take these design structures and translate them into source code.

(f) Test, integration and debug
The purpose here is to show that the finished code performs as specified. It involves testing of individual software modules, combined modules (sub-systems) and finally the complete system. This topic is covered in chapters 11 and 12.