SUMMARY

The two central themes of this study are:

(1) How can one simplify process coordination structures in multiprocessing systems, specifically in multi-controlled telephone systems?

(2) How can one analyze the correctness of process coordination structures in multiprocessing systems systematically?

To answer these two questions we have executed preliminary studies on multiprocessing, coordination problems, program structuring, correctness analysis, and multi-controlled telephone systems (chapters 1, 2, 3, and 5). We have then attempted to answer the questions, first in general (chapter 4), then more specifically for the telephone systems (chapter 6).

Below we will briefly discuss the contents of the six chapters in this dissertation. We will summarize the main conclusions.

1. On multiprocessing

We begin with a general discussion of multiprocessing systems, and of the related problems. It is outlined how the advent of multiprocessing has led to the development of a new conceptual model of computation. The tools with which one can describe and study the coordination problems on higher levels of abstraction are gradually being developed. It is shown how one can construct basic coordination schemes in layers. We then give an overview of the existing specification methods for multiprocessing structures and coordination patterns in the higher level languages. The early attempts to *extend* the sequential programming languages with a few *ad hoc* primitives, to make them suitable for parallel programming, are criticized.
We discuss the criteria for the evaluation of coordination primitives, and introduce a tentative taxonomy of such tools, based on the extent to which they make use of explicit state variables. (It will appear in chapter 2 that such state variables play an important role in a correctness analysis.) We thus arrive at a division of the coordination tools into three broad classes: the locking primitives, the semaphore primitives, and the monitors. A consequence of this classification is that the context in which we discuss the existing coordination tools differs from the usual. For instance, we discuss Brinch Hansen's concept of the *critical region* in the context of the *locking primitives*, as in our taxonomy the region belongs to the same class as the primitives *lock* and *unlock*. Similarly, the primitive *test&set* is treated in the context of the *semaphore primitives*, and the *conditional critical region* in the context of the *monitors*. The chosen taxonomy thus allows us to discover unexpected resemblences and differences between the tools.

In *appendix A* we discuss algorithms for concurrent programming control (sometimes called: the *bakery algorithms*), and suggest some alterations to one of

the algorithms published recently.

2. On correctness analysis

In the second chapter we study the adequacy of the modelling of coordination schemes of 9 formal models: Petri Nets, Coordination Nets, Slutz Graphs, Bigraphs, Coordination Graphs, Robinson & Holt Specifications, Actor Model Specifications, Path Expressions, and models based on First Order Predicate Calculus. It is found that most models have insufficient descriptive power with which to model coordination patterns transparently. The tools are also insufficient for the analysis of the models. There is often no clear correspondence between a modelled solution of a coordination problem and a programmed solution with any of the known coordination primitives. The complexity of the models can rise quite rapidly with the number of processes involved in a coordination, which clearly frustrates analysis.

3. On structured programming

As the available analysis tools are not powerful enough to allow for the 'verification' of just any program, one has to restrict oneself to conceptually simple and well-structured programs. In the third chapter we study the techniques which are known as the 'structured programming techniques'. We also discuss the origination of the theories, and in that context we take a brief look at the so-called 'software crisis' which stimulated this origination in many ways.
The relation between the structured programming techniques and more fundamental problem solving methods is briefly investigated. The chapter is concluded with a discussion of some of the major criticisms that have been raised against the methodology.
Though, the structured programming techniques are applied mostly to the design of sequential programs, the principal ideas (abstraction, restrictions, stepwise refinement, and clear notations) apply equally well to other fields of computer programming, including the design of coordination structures for multiprocessing systems.

4. The section model

We begin the fourth chapter with a fundamental discussion of coordination problems in general. The discussion is based on two analogons: the analogon of the *door* and the analogon of the *guarded room*. The study of these analogons yields some new insights into the nature and causes of coordination problems, and how one may go about solving them. Based on these insights we develop a new descriptive and analytical model for process coordination, which is called the *section model*. We introduce more precise notions of *coordination rules*, *initiation conditions*, *blocking conditions*, etc.
A method for the description of coordination structures on varying levels of abstraction is outlined. Coordination structures can thus be defined in a simple and uniform manner. It is shown that there is a one-to-one correspondence between the descriptions in the section model and the solutions programmed with one of the coordination primitives, defined and discussed in the first chapter (i.e. the extended dependence operations).

In *appendix D* we elaborate some initial ideas on the possibilities for the automated verification and design of coordination structures (the *transition table model*).

5. Multi-control in telephone exchanges

In the fifth chapter we give an overview of the application of multi-control structures in stored program controlled telephone systems. It is noted that the concentration of all control tasks in a single processing unit (the traditional *monolithic s.p.c.*) is antiquated. It is argued that the introduction of multi-control in telephone systems can lead to important improvements in the clarity and structuredness of the call handling software. The multi-control leads almost naturally to a division of control tasks into logical units, which are executed concurrently under the observance of some well-defined coordination rules.
We give an overview of different types of multi-control, and of different load-sharing strategies. The causes of the coordination problems in the multi-controlled telephone systems are divided into three broad classes: functional delegation of tasks (e.g. pre-processing), processor multi-plexing (e.g. multi-programming), and processor multiplication with load-sharing.

6. Analysis of coordination problems in call-handling software

With the aid of the section model, developed in chapter four, we analyze the coordination problems that may occur in an imaginary telephone system with an extreme amount of concurrency. We encounter essentially four types of coordination problems:

(1) ordering problems within and between call processes;
(2) exclusion problems;
(3) problems related to the abortion of unsuccessful call attempts, and
(4) higher level inconsistencies, especially with respect to the special subscriber facilities, like call-transfer, ring-back, and waiting queues.

The coordination problems are studied on varying levels of abstraction: e.g. the subscriber process level, the call process level, the call phase level, and the call function level. The analysis is independent of implementation specifics, and independent of real-time requirements.