HANSEN81
Brinch Hansen, Per. "The Design of Edison".
Software - Practice and Experience,
Volume 11, Number 4 (April 1981), pp. 363-396.

As E.W. Dijkstra has observed, one of the problems in studying computer science is that the results are usually presented in a polished final form. This makes it more difficult for the student to see the blind alleys and false starts that are so much a part of actually doing computer and science research. This paper takes the reader "behind the scenes" in the development of Edison, discussing some of the design decisions and why they were resolved the way they were. It discusses the design goals for the language, and how various questions were resolved by considering how they affected those goals. It also discusses the difficult job of writing a language description. The Edison language is a language for systems programming derived from Pascal and Modula. It is described in [1], and examples are given in [2]. It is used in [3] to present an operating systemn for a personal computer. The complete text of the compiler, interpreter and operating system are included in [3]. Although in-depth knowledge of Edison is not necessary to read this paper, some familiarity is useful. Per Brinch Hansen is well known for the development of Concurrent Pascal, a language for operating systems programming that included processes, monitors and classes. In this paper, he contrasts the decisions he made for Edison with those he made for Concurrent Pascal, and explains why. Edison was designed with two ideas in mind: great simplicity and a different philosophy (towards systems programming) from Concurrent Pascal. The first goal led to the omission of many "usual" features from Edison. Some of the more notable are variant records, pointers and case statements, which are usually included in modern programming languages. However, Brinch Hansen explains why these facilities (as well as many others) were left out of Edison. He points to the fact that the Edison compiler is written in Edison as proof that these facilities are not needed. The second goal lead to the replacement of many of Concurrent Pascal's concurrent constructs with simpler more dangerous ones. He discusses the removal of classes from Edison, adopting a Modula-like mechanism for scope control. He also discusses the removal of the monitor, replacing it with the simpler (and more dangerous) conditional critical region. Processes (from Concurrent Pascal) have also been replaced with parallel statements. He discusses at length Edison's type rules, and why he designed them the way he did, including a discussion of the syntax that he used. He discusses his solution to Pascal's well-known problem with character strings at length, which he solved by relaxing the type contraints for character string literals as parameters to procedures. This is essentially a minimal change to Pascal that solves the problem. He also prsents justifications for the removal of variant records and pointers. These reasons are somewhat weak. The insecurities of these constructs are well known, and searching for better alternatives is a laudable goal. However, Brinch Hansen justifies their removal by stating that it is possible to build a compiler without them, and other systems programs don't really need them. This is debatable since there are many algorithms that are more clearly expressed using these constructs than the alternatives he uses. He also discusses the various statements that have been omitted, and talked about his method for handling low-level I/O. Finally, he discusses in detail the various drafts that the language report went through, quoting Peter Naur, who provided criticism. This is a very interesting part of the report, demonstrating how difficult it is to write a good language description, and how helpful a good reviewer can be. This paper should be read by all students of programming language design. It is particularly valuable because it considers two post-Pascal languages designed by the same person nearly 10 years apart, showing the development of his thinking on the problems of programming language design.

References [1] Brinch Hansen, Per. "Edison - a Multiprocessor Language", *Software - Practice and Experience,* Vol. 11, No. 4 (April 1981), pp. 325-362.

[2] Brinch Hansen, Per. "Edison Programs", *Software - Practice and Experience,* Vol. 11, No. 4 (April 1981), pp. 397-414.

[3] Brinch Hansen, Per. *Programming a Personal Computer*, Prentice-Hall, New Jersey (1982).