A review of a book on PEARL

"PEARL: Systematische Darstellung für den Anwender" by Axel Kappatsch, Horst Mittendorf, and Peter Rieder. (300 p.) – München, Wien: Oldenbourg, 1979.
ISBN 3-486-23021-2

This book's title says "PEARL: a systematic presentation for the user", where PEARL is an acronym for "Process and Experiment Automation Real-time Language". I found it hard to decide which of the two is worse, the book or the language; eventually I decided that the language has won the race: the book is bad, but the language is terrible.

The book has eight chapters: 0. Introductory Example (4 p.), 1. Construction of a PEARL Programme (13 p.), 2. Data and Data Aggregates (57 p.), 3. Communication with Peripheral Equipment (46 p.), 4. Specification of the Peripheral Equipment in the SYSTEM Part (69 p.), 5. Sequential Control (19 p.), 6. Control of Concurrent Activities (19 p.), and 7. Message Evaluation as Example of a Complete PEARL Programming System (42 + 28 p.).

It is in all respects a publication from a subculture. The introduction mentions PEARL as an existing object, but neither when, nor by whom it has been designed, and references to the literature are confined to three pages of references to PEARL publications only. Worse, the poor

reader has to join the authors in midstream. In Chapter 0, the reader has to guess which letter combinations are key words –there are 146 (!) of them– and which are identifiers chosen by the programmer. In Chapter 1 it becomes annoying that the reader is confronted with a text in four different type founts without being told what significance to attach to that difference; similarly, formal syntax is given in an unexplained formalism; scope rules are given a pictorial "explanation", the interpretation of which is guesswork. The pictorial "explanation" of assignment and dereferencing in Chapter 2 is a similar new mystery. Etc.

PEARL is described in terms of an elaborate private terminology. This was probably unavoidable since PEARL seems to combine the worst of PL/I, ALGOL 68, and three assembler languages. The reader's life is complicated, firstly, by the absence of an index guiding the reader to the definitions of those private terms –the table of contents is too coarse to take its place– and, secondly, by the fact that many of these definitions are very implicit: a typical usage of the term is supposed to tell the reader what is meant by the term. (E.g. "INV does not belong to the type of an object, but to the access function represented by the identifier, and is therefore referred to as access attribute." And as far as I could find out, this is supposed to tell us what is meant by "access attributes".) In actual fact so-called "teaching by example" is a euphemism for not teaching at all.

The last chapter of 42 pages deals with a worked-out example of 28 pages PEARL code, given in a separate appendix. I did not have the courage to follow that example in any detail, since the appendix is a reduced copy of more than 1500 lines of extremely poor lineprinter output. (The manufacturer of that lineprinter has been tactfully left anonymous.)

Sometimes a subculture is an avant-garde; PEARL's subculture, however, is inbred, old-fashioned, and backward. Its approach to the definition of semantics is entirely operational. For the semantic definition of sequential programming languages the operational approach is very clumsy, but in the case of concurrency it is a disaster. I think it is not only the book that is to be blamed for explaining the language in terms of an implementation -e.g. "the main significance of a module is that it represents a piece of PEARL text that can be compiled separately"-, I fear it is the language itself. Interrupts are primarily a device by means of which a number of sequential processes can share a smaller number of processors. The definition of a language geared to multiprogramming should be absolutely neutral with respect to the number of processors that might be available in a specific implementation. In PEARL, however, "INTERRUPT" is one of the 146 key words! Though it is now well-known for more than 15 years, PEARL's designers evidently did not know that the actual number of processors available is the easiest thing to abstract from.

Whenever I see the announcement of a German book on Computing Science, I always wonder why it has been published in German; this time I wondered why it had been published at all. But I found the explanation: the German Federal Government has spent more than 50 million DM. on the development of this "baroque monstrosity" and when mistakes are made on a sufficiently grandiose scale they become by definition a precious achievement. The frequent reference to DIN documents is very telling: it is the n-th confirmation (n large!) that standardisation as primary target is one of the worst possible incentives for language design; with Ada the USA Department of Defense has increased -quite unnecessarily- n still further.

Plataanstraat 5
5671 AL NUENEN
The Netherlands

28 February 1982

prof. dr. Edsger W. Dijkstra
Burroughs Research Fellow