Workshop on special systems and system features

Chairman: **Dr H-J. TREBST**
Erlangen, W. Germany

The discussion centred on problems of testing software, and the kinds of testing aids which can be offered, such as simulation of the environment in process control (either in the same or a different computer), checks of source language semantics and conventional debugging.

E. In checking against garbling of data, we have kept a check sum on a block of data which is tested at regular intervals. This works well for data which is either not changed much or usually changed in complete blocks, but is not so good when one or two words of a block may be changed more frequently than the rest of the block.

Y. The method described by Professor Wettstein is a high-level analogue of the conditional assembly methods used for system generation by at least IBM and DEC. It is very hard to debug these, because each possible configuration has to have an operating system adapted for it, and then checked on an appropriate physical configuration. For example, on our PDP-11 with RPO2 disk, we assembled the adaptable disk handler for DOS using the appropriate code for the RPO2, and the resulting program in assembly language had an error in it – obviously that had never been checked before the adaptable disk handler was distributed. This does not mean that I doubt the value of adaptive operating systems, but that they are even more difficult to debug than non-adaptive operating systems.

V. Perhaps the very point of slipping from high-level language to assembly language in debugging is that it forces the programmer to view his program in a different way.

H. The question of detecting faults is not necessarily one of "Which level of language do we use?" It is largely a matter of discovering when it is possible to detect possible faults – at compile time or at run time.

For example, consider the program

```
{local a ; a:=0;  a:=a+1,  a:=a+1}
```

where the comma denotes that the two statements 'a: = a+1' are done in parallel. This is not deterministic, I cannot imagine a situation in which this program is correct. Should we therefore have it flagged by the compiler as containing a semantic bug?

L. There is considerable structure within a group of interacting processes which may be deduced from the programs. The testing of the interactions between these processes is itself a real-time problem which cannot be truly tested in off-line environment. The program preparation facilities for real-time work should include:

1. Similation of interactions between processes.
2. Flagging possible conflicts which could arise at run time.

These are analogous to automatic flow-chart generation for single process programs.

E. Debugging is never fully complete. We have to decide some possibly arbitrary criterion to determine when to regard it as debugged. Perhaps we can say it has reached this stage when the customer pays the bill.