

## CHAPTER 8

### BACKWORD

(Reference: Experience)

This part of the course has concentrated on analysis; i.e. how to analyse a product, given the design. We have:

1) set up the general equation  $\underline{\Psi} = \underline{A} \underline{\Psi}$

2) developed a philosophy and approach to solution techniques which is, to be sure, one person's opinion.

3) looked briefly at alternate ways to analyse, if not solve, the basic equations

and

4) looked briefly at the bigger picture in the sense of feedback to the matrix  $\underline{A}$ .

But there is another side of the coin: DESIGN.

In contrast to analysis, design is the process by which we "make" a product given the analysis techniques.

Much of the work in the industry is centered on design, not analysis.

One usually assumes that analysis techniques are available or can be put quickly in place,

- especially in a mature design environment.  
To be sure, much analysis is done. But the controlling emphasis is on design; the analysis is there <sup>only</sup> because design needs it.

Design includes considerations such as:

- 1) philosophy of the design
- 2) choice of materials (fuel, coolant, moderator)
- 3) safety
- 4) controllability
- 5) \$
- 6) man-rem
- 7) maintainability
- 8) reliability
- 9) constructability
- 10) purpose of reactor! - is electrical? process steam?
- 11) licensing regulations
- 12) public opinion
- 13) availability of manufacturers and suppliers
- 14) layout

ETC.

;

;

- And these items are all interwoven! This should put the role of the analyst in perspective.

Bear in mind, too, that the biggest factor, in industry, determining the cost of analysis is the manpower cost. True, there are some big codes use in safety analysis which are quite costly but if the code has been verified it makes good sense to use it, get the results out, feedback to the design process and get on with the job. A re-do of a major code should not be done as part of the design project unless there is ample time - and I mean ample, a good algorithm to estimate the time required to do a job is to make a conservative guess, say 1 line of coding per hour (including testing) times the number of lines = # of hours. Move up to the next unit in time and multiply by two.

Thus, a 50 line subroutine will take

1,  $50 \times 1 = 50$  hours to fully write, test, verify and commission.

2, 50 hours  $\rightarrow$  50 days

3,  $50 \text{ days} \times 2 = 100 \text{ days} !!$

It may take even longer - especially when you try to speed up the process by rushing, making unverified assumptions, etc. or you have

fires to fight, bureaucracy to get deal with and a host of other details. I don't wish to end up on a cynical note but reality is reality, unfortunately. The best antidote against the insane reality you will face, as opposed to the inane un-reality you now face, is humour.

In that vein:

Before ordering a test decide what you will do if it is (1) positive, or (2) negative. If both answers are the same, don't do the test.

If you want a track team to win the race, you find one person who can jump seven feet, not seven people who can jump one foot.

If you put a spoonful of wine in a barrel full of sewage, you get sewage.  
If you put a spoonful of sewage in a barrel full of wine, you get sewage.

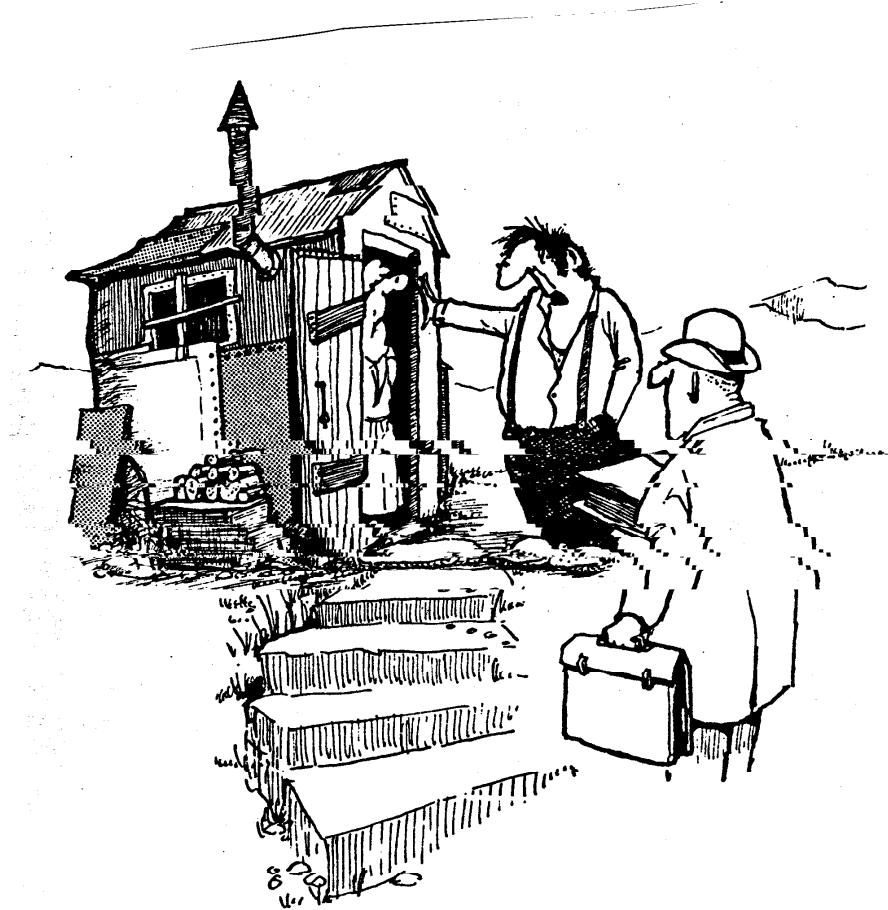
### IMBESI'S LAW OF THE CONSERVATION OF FILTH:

In order for something to become clean, something else must become dirty.  
**Freeman's Extension:**  
... but you can get everything dirty without getting anything clean.

The organization of any bureaucracy is very much like a septic tank — really big chunks always rise to the top.

Some people manage by the book even though they don't know what book wrote the book or even what book.

Carefully consider your needs.



**"Do we need a set of encyclopedias?"**

8-6

And, it's not what you do sometimes, but when!



"Herman . . . two is load . . . THREE is fire!"

Don't forget the test.



"Testing...testing...one, two, three."

And when all else fails, remember:

**LAST LAW OF  
PRODUCT DESIGN:**  
*If you can't fix it, feature it*

- end -