THE PRACTICAL GUIDE TO STRUCTURED SYSTEMS DESIGN

MEILIR PAGE-JONES

FOREWORD:
ED YOURDON

8.11 Fan-out

The *fan-out* from a module is the number of immediate subordinates to that module. An example is shown in Fig. 8.33.

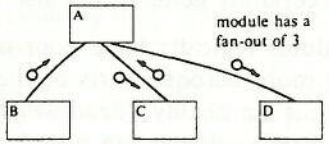


Figure 8.33

In the 1960s, when modularity — the "maintenance savior" of the future — first became popular, a software design technique known as main-line design made its debut on the systems development stage to thunderous applause and wild cheering. Main-line design called for a single boss (the "control" or "main-line") module, with every other module in the system as its subordinate (see Fig. 8.34).

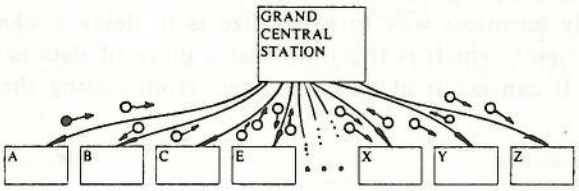


Figure 8.34

There was a very well-intentioned purpose of main-line design: to reap the benefits of modularity by isolating a single change to a single module. It did that, all right! Almost every non-trivial change that was made to the system required a change to — yes, you guessed it! — the main-line module, simply because almost every non-trivial function in the system was being carried out by that unhappy module. Even from the structure chart in Fig. 8.34 alone, you can visualize that the main-line module must contain a maelstrom of code, for all the pieces of data in the system are dashing in and out like Keystone Kops.

The adulation of main-line design turned to sneers, and the bouquets of roses to cabbages. Main-line design was booed off the software stage and almost dragged modularity with it.

So, to avoid repeating such a tragedy, you should try to limit the fan-out from a module to no more than seven. A module with too many subordinates can be easily cured by the old familiar remedy of

factoring. Separate each subfunction within a main-line module into a module of its own, as shown in Fig. 8.35:

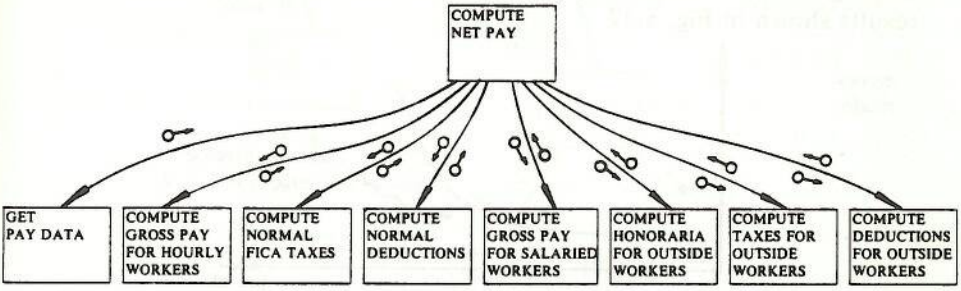


Figure 8.35

The figure looks like a "pancake," a symptom of a missing intermediate level. High fan-out is rectified by factoring out middle-management modules with strong cohesion and loose coupling (see Fig. 8.36).

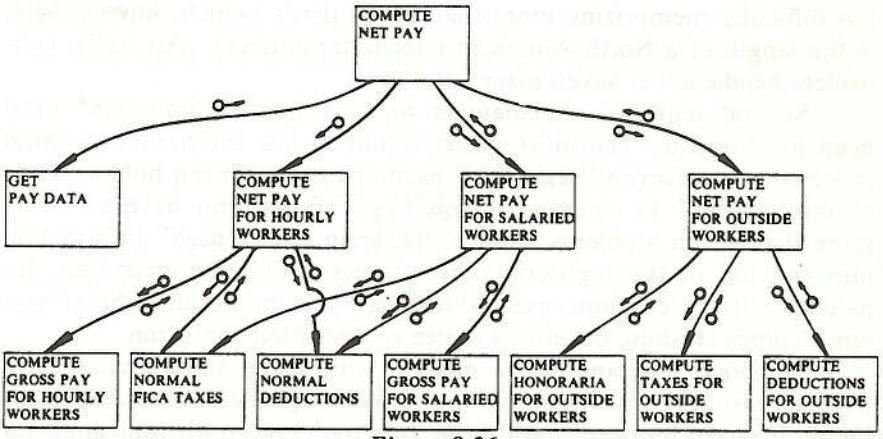


Figure 8.36

Why should the limit on the number of immediate subordinates to a module be seven and not seventeen or twenty-seven? Does this mean that if you design a module with eight other modules reporting to it that you'll be struck by lightning? No, of course the number seven is not an unbendable standard, although if you have a fan-out of more than seven from any module, warning bells should sound in your mind.

Seven is a very important number in human psychology.* Have you ever tried doing more than seven activities at once? If you have,

*G.A. Miller, "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information," *Psychological Review*, Vol. 63 (March 1956), pp. 81-97.

I'm sure you have found yourself becoming very flustered and prone to making errors. Reasonably scientific experiments have yielded the results shown in Fig. 8.37.

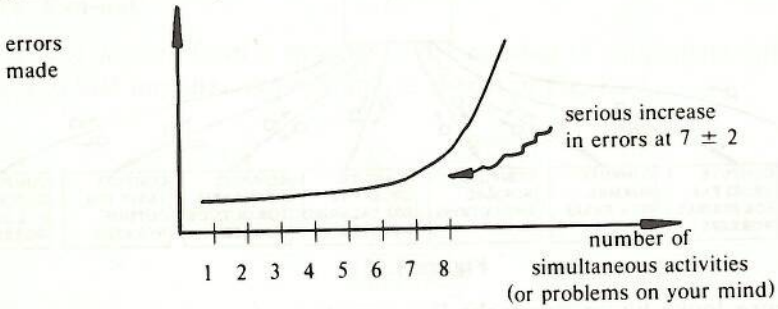


Figure 8.37

Everyday life seems to bear out the graph. For example, a person has difficulty memorizing more than seven digits (which, interestingly, is the length of a North American telephone number). Air traffic controllers handle about seven planes at a time.

Several tentative explanations for this human limitation* have been put forward. The most attractive one is that the brain's "central processor" uses seven "registers," each one of which can hold a "node of information" (a concept, if you like). When you have to tackle more than seven problems at once, the brain must "page" information into and out of the registers. This process isn't quite error-free, apparently. In the extreme case, "thrashing" sets in, and the whole "system" comes crashing down in a mélée of distracted confusion.

Someone programming a module with seven subordinates must have at least seven issues on his mind. Although I can't say that this is bound to make him err, it will bring him too close to his hrair limit for comfort.

Low fan-out is acceptable, although it's rather like a department with one boss and one worker. Such a case may be a hint to factor out a module from the boss, as VALIDATE TRANS has been factored in the example in Fig. 8.38.

---

*This limitation is not solely human. This phenomenon has been observed in several animal species. For example, Richard Adams in his delightful heroic tale *Watership Down* (New York: Macmillan, 1974) relates how rabbits count: "one-two-three-four-hrair." Numbers greater than four are just too large for them to distinguish between. Their conceptual counting limit (or *hrair* limit) is four.

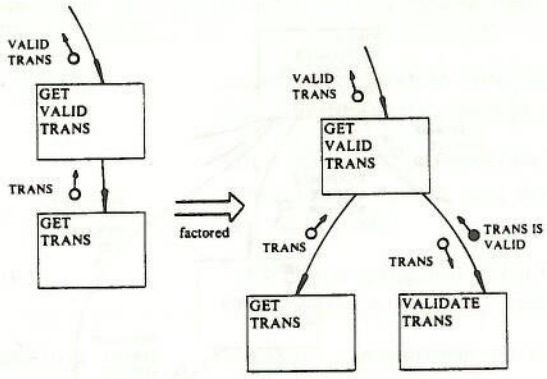


Figure 8.38

8.12 Fan-in

The *fan-in* to a module is the number of immediate bosses it has.

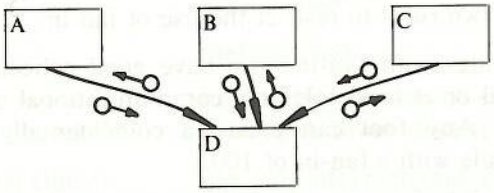


Figure 8.39

High fan-in is the reward for intelligent factoring and the removal of restrictive modules. At programming time, having one function called by several bosses avoids the need to code practically the same function in several places. Hence, at maintenance time, duplicate updating to a function is eliminated.

Don't be afraid of the situation shown in Fig. 8.40, in which a module's bosses are at different levels. If a module has a truly useful function, then it can be used by a module anywhere in the system.

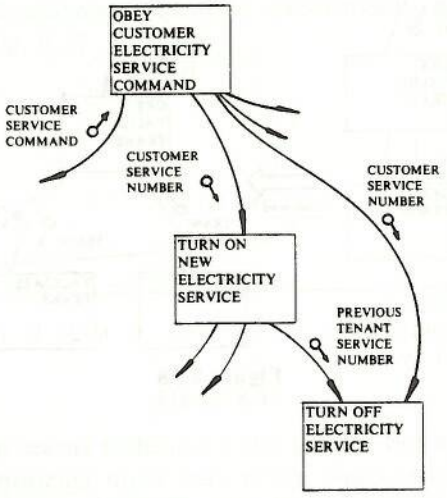
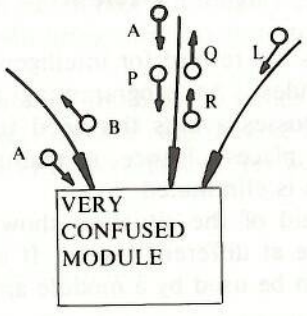


Figure 8.40

There are two rules to restrict the use of fan-in:

- Modules with fan-in *must* have good cohesion: functional or at least tolerably communicational or sequential. Any fool can create a coincidentally cohesive module with a fan-in of 100.
- Each interface to a single module must have the same number and types of parameters. For example, this makes no sense:



8.13 Table for design criteria

Here is a table summarizing the twelve design criteria discussed in this chapter, plus cohesion and coupling, which were discussed in Chapters 6 and 7.

Criterion	Guideline
cohesion	make each module functional or, at the least, sequential or communicational
coupling	keep the coupling between modules loose (data, stamp, and descriptive flags are acceptable)
data structure	match the program structure as much as possible to the data structure
decision-splitting	keep the recognition part of a decision close to its execution
editing	edit in successive levels, with the simplest editing done at the lowest level
error reporting	have the same module that recognizes the error as such report the error
factoring	keep it high
fan-in	make it high
fan-out	restrict the number of subordinates to a module to fewer than seven
informational cluster	use an informational cluster to avoid the problems of stamp or common coupling (or both)
initialization/termination modules	initialize as late as possible; wrap up as soon as possible
restrictivity/generality	don't make a module too restrictive or too general
state memory	avoid state memory wherever possible
system shape	make the system balanced (neither physically input-driven nor physically output-driven)

This brief list/table presents only a simplification of the criteria. In the actual design process, it is necessary to take into consideration all of the details presented in the body of this chapter; used carefully, these criteria are your basic tools for evaluating and refining a design.