acm forum

Taulbee Survey Report

I was disappointed in the report by David Gries on the 1984–1985 Taulbee Survey (*Communications*, October 1986, pp. 972–977). Although it was well presented, reasonably laid out and, most likely, accurate, it was not useful information. Data in this form need commentary to become information. I often hear of "industry eating its own seed corn" in reference to the hiring of Ph.D.'s away from academia, and of a shortfall in Ph.D.'s for computer science overall. I jumped at the chance to learn from the Gries report. Alas, there were no conclusions drawn, no help for all us uninformed. I know that time spent pouring over the data would give me some feel for the condition I am concerned over (e.g., potential lack of sufficient Ph.D.'s), but I know I do not have the time and I fear I lack the knowledge to draw proper conclusions.

Roger S. Gourd
MASSCOMP
One Technology Park
Westford, MA 01886

Response:
Perhaps reader Gourd is right in asking for more commentary and conclusions. Inexperience, a reluctance to draw too many conclusions, and a lack of space all contributed to the form and content of the report. We will try to address this criticism in the next report.

David Gries
Department of Computer Science
Cornell University
405 Upson Hall
Ithaca, NY 14853-7501

Network Noted

In the "Notable Computer Networks" article by John S. Quarterman and Josiah C. Hoskins (*Communications*, October 1986, 932–971) a few company networks are detailed. One such network which is not detailed seems to be a fairly well-kept secret. This is the internal network belonging to Tandem Computers Incorporated. This network has 200 NonStop hosts connected via 150 links consisting of microwave, laser, satellite, fiber, and copper running at speeds up to 3 Mbit/s. The aggregate processing power of this virtual machine is 1.6 BIPS (billion instructions per second). Both the systems and the network are fault-tolerant.

A staff of four employees in Cupertino, CA, and one in Germany support the user community of 6500 hard-wired and 2500 dial-up terminals and PCs. While the network, spanning 23 countries, is running 24 hours a day, the support staff works normal 40 hour weeks. Because of its fault-tolerant nature, communications failures are not critical to network connectivity.

This ease of maintainability is due to Tandem's proprietary protocol, EXPAND, which is modeled after X.25. Addition, deletion or moves of hosts do not require a Network Sysgen. When a new host is added to the network, a "ripple effect" takes place until each host knows the best path to the new host. During a network failure and after the subsequent recovery, the network performs its own rerouting.

The network supports over 100 production applications including Electronic Mail, Order Entry, Manufacturing, VLSI Design, Customer Engineering Dispatch, Problem Reporting and Software Patch Distribution.

A typical Tandem electronic-mail name looks like 'LaPedis_Ron', or 'Payroll', the second being a department name rather than a person. There is no need to specify the geographical location of a mail correspondent.

An on-line telephone book, telephone messages, and request form application round out the average employee's interface with the network. An article on the Tandem network has appeared in *Data Communications* magazine (August and September 1985).

Ron LaPedis
Corinne DeBra
Tandem Computers Incorporated
19191 Vallco Parkway
Cupertino, CA 95014-2599

"GOTO Considered Harmful" Considered Harmful

The most-noted item ever published in *Communications* was a letter from Edsger W. Dijkstra entitled "Go To Statement Considered Harmful" [1] which attempted to give a reason why the **GOTO** statement might be harmful. Although the argument was academic and unconvincing, its title seems to have become fixed in the mind of every programming manager and methodologist. Consequently, the notion that the **GOTO** is harmful is accepted almost universally, without question or doubt. To many people, "structured programming" and "**GOTO**-less programming" have become synonymous.

This has caused incalculable

harm to the field of programming, which has lost an efficacious tool. It is like butchers banning knives because workers sometimes cut themselves. Programmers must devise elaborate workarounds, use extra flags, nest statements excessively, or use gratuitous subroutines. The result is that **GOTO**-less programs are harder and costlier to create, test, and modify. The cost to business has already been hundreds of millions of dollars in excess development and maintenance costs, plus the hidden cost of programs never developed due to insufficient resources.

I have yet to see a single study that supported the supposition that **GOTO**s are harmful (I presume this is not because nobody has tried). Nonetheless, people seem to need to believe that avoiding **GOTO**s will automatically make programs cheap and reliable. They will accept any statement affirming that belief, and dismiss any statement opposing it.

It has gone so far that some people have devised program complexity metrics penalizing **GOTO**s so heavily that any program with a **GOTO** is *ipso facto* rated more complex than even the clumsiest **GOTO**-less program. Then they turn around and say, "See, the program with **GOTO**s is more complex." In short, the belief that **GOTO**s are harmful appears to have become a religious doctrine, unassailable by evidence.

I do not know if I can do anything that will dislodge such deeply entrenched dogma. At least I can attempt to reopen the discussion by showing a clearcut instance where **GOTO**s significantly reduce program complexity.

I posed the following problem to a group of expert computer programmers: "Let X be an $N \times N$ matrix of integers. Write a program that will print the number of the first all-zero row of X, if any."

Three of the group regularly used **GOTO**s in their work. They produced seven-line programs nearly identical to this:

```
for i := 1 to n
do begin
  for j := 1 to n do
    if x[i, j]<>0
      then goto reject;
  writeln
('The first all-zero
                    row is ', i);
  break;
reject: end;
```

The other ten programmers normally avoided **GOTO**s. Eight of them produced 13- or 14-line programs using a flag to indicate when an all-zero row was found. (The other two programs were either incorrect or far more complex.) The following is typical of the programs produced:

```
i := 1;
repeat
  j := 1;
  allzero := true;
  while (j<=n) and allzero
  do begin
    if x[i, j]<>0
      then allzero := false;
    j := j + 1;
  end;
  i := i + 1;
until (i>n) or allzero;
if i<=n
    then writeln
('The first all-zero
                row is ', i - 1);
```

After reviewing the various **GOTO**-less versions, I was able to eliminate the flag, and reduce the program to nine lines:

```
i := 1;
repeat
  j := 1;
  while (j<=n)
  and (x[i, j] = 0) do
    j := j + 1;
  i := i + 1;
until (i>n) or (j>n);
if j>n
    then writeln
('The first all-zero
                row is', i - 1);
```

By any measure not intentionally biased against **GOTO**s, the two **GOTO**-less programs are more complex than the program using **GOTO**s. Aside from fewer lines of code, the program with **GOTO**s has only 13 operators, compared to 21 and 19 for the **GOTO**-less programs, and only 41 total tokens, compared to 74 and 66 for the other programs. More importantly, the programmers who used **GOTO**s took less time to arrive at their solutions.

In recent years I have taken over a number of programs that were written without **GOTO**s. As I introduce **GOTO**s to untangle each deeply nested mess of code, I have found that the number of lines of code often drops by 20–25 percent, with a small decrease in the total number of variables. I conclude that the matrix example here is not an odd case, but typical of the improvements that using **GOTO**s can accomplish.

I am aware that some awful programs have been written using **GOTO**s. This is often the fault of the language (because it lacks other constructs), or the text editor (because it lacks a block move). With a proper language and editor, and adequate instruction in the use of **GOTO**, this should not be a consideration.

All of my experiences compel me to conclude that it is time to part from the dogma of **GOTO**-less programming. It has failed to prove its merit.

Frank Rubin
The Contest Center
P.O. Box 1660
Wappingers Falls, NY 12590

REFERENCE

1. Dijkstra, E.W. "Go to statement considered harmful." *Commun. ACM 11*, 3 (Mar. 1968), 147–148.