MIP Envy: a Programming Complex

Jim Gray, IBM Research, San Jose

September 20, 1980

When I left UC Berkeley to join IBM, I was surprised to find that the university provided better computing services than IBM. IBM offered fewer languages, poorer machine response, poorer availability, and half-duplex terminals that took a lot of getting used to.

That was nine years ago. Things have improved. We now have full screen terminals, big address spaces, bigger disks and a network. However, response times to trivial operations are still long. We still do not have a language that is nice, has an incremental compiler and a symbolic debugger with type checking. The programming languages (PLI and PLS) and text editing system (SCRIPT) I use are typical of l97O's software.

Meanwhile, computers have gotten about ten times cheaper. Therefore, we should have ten times as much. I don't have ten times as much. In fact, we go through a cyclic feast-then-famine so that about 25% of the time computing services are so bad that everyone is screaming and finally the next increment of computing is "justified".

As I look at my colleagues at Bell Labs (the UNIX group), at Xerox PARC, or at Stanford I see that they have a much better programming environment than we do. I feel bad about this, and have developed a complex called MIP envy. It's not just envy of other people's MIPS, but also envy of their languages, editors, debuggers, mail systems, and networks. But, MIP envy is a term every IBM programmer will relate to. I believe it is a common complex among software people in IBM research.

The tragedy is that IBM Research has it much better than the rest of IBM. Right now at IBM's Santa Teresa development lab, people can only log on at certain times and cannot compile during prime shift (compiles take over 30 minutes and so consume too much of the person's shot at the machine). This situation colors people's designs and tends to make the designs more batch oriented and less interactive and hence less easy to use. It is ironic that the typical IBM development programmer has poorer computing facilities than the typical airlines ticket agent.

The tool situation is exemplified by the state of IBM's system programming language PLS. PLS was created in the late sixties but the PLS group was disbanded in the early days of FS. The PLS group was reconstituted in 1976 by the Poughkeepsie lab. It supports PLS only on MVS (not VM or DOS)). So PLSS is not supported on Release 6 of VM and hence most development shops have not moved to that release (which came out about a year ago)! People at Endicott are taking the Structured Programming verbs (SELECT, Boolean expressions, etc.) out of System R because PLS is not supported on DOS. To give a grim example of the fate of tool builders, the author of VMSG (the electronic mail system we all use) was ordered not to work on it anymore! It is now supported by a informal group (not including the author of VMSG). There is no shortage of such stories. IBM development programmers have very primitive tools.

Computer research must aim its ideas for machines ten years in the future (System R started in 1974 and will enter the market in 1982). It is tough to deal with a sixteen-year gap: working on eight-year-old machines for a product eight years in the future.

Computer research must attract bright young people with new ideas. Such people show little interest in IBM after seeing the facilities at Xerox, Bell or Stanford. In addition, there is a slow flow of good people out of IBM. System R lost its best programmer to Xerox (in 1976). He gave MIP envy as his reason for leaving.

I think it is bad business to provide inadequate computing services because:

- Good computing services increase programmer productivity.
- One cannot design systems for the eighties when confronted with the hardware and software of the sixties.
- Good computing services are a tax-free fringe benefit that the company can offer its programmers.
- Programmers leave IBM because the programming environment is better elsewhere.

Some of the main reasons for this bad situation are:

- We get "old" hardware (the 168 was designed in 1970 and is priced at 1975 prices).
- No one in IBM funds tool building. Therefore, the tools are bootlegged and are flaky.
- Timesharing encourages "optimization" where machines are configured to saturate at peak periods (i.e. when people come to work). Therefore, unless you are a night owl, you work on a saturated machine.

I can point to several projects I have not undertaken because computing resources were insufficient (e.g. fuzzy dump in System R), and others in which I had to do a poor job because the machines were so slow or the tools were so bad.

The conversion of System R to MVS took two years largely because the MVS system was second level on VM. Simple things like a TSO logon took 15 minutes! Complex things took hours. The project would have taken six months if reasonable machine services had been available.

The Rendezvous project ended one day when we were moved from a 168 to a 158. The program just ran too slowly to be interactive.

Many of the bugs found in System R could have been caught by a type checker that compares the types of formal and actual parameters (called Lint on UNIX). Such a tool would have paid for itself on just the System R project. As it was, we had to write more basic tools such as an IO library, cross-reference, trace facility, driver and interactive debugger. These tools are too flaky to be of much use to anyone but us (they are documented by example and oral tradition). Each of these tools should have existed before we started.

Les Belady showed clearly that tools are not the problem with programming. Managing programming and designing programs are the real problems. But, there are no obvious solutions to these problems. There are many good ideas in the tools area. Tools are one area where a relatively small investment will make substantial improvements to one part of the programming problem

IBM should provide better computing hardware and software to its programmers. This means spending some money and recognizing and encouraging people who make good tools. I do not recommend a tool department or a tool taskforce or a tool memo from the Corporate Technical Committee. Good and experienced programmers are difficult to hire and keep. One thing that attracts good programmers is good computing services.