How to Have a Bad Career in Research/Academia

David A. Patterson

Computer Science Division
EECS Department
University of California at Berkeley

http://http.cs.berkeley.edu/~patterson

Outline

- **Part I: Key Advice for a Bad Career**
- **Part II: Key Advice on Alternatives to a Bad Career**
- **Topics covered in both parts**
  - **Selecting a Problem**
  - **Picking a Solution**
  - **Performing the Research**
  - **Evaluating the Results**
  - **Communicating Results**
  - **Transferring technology**

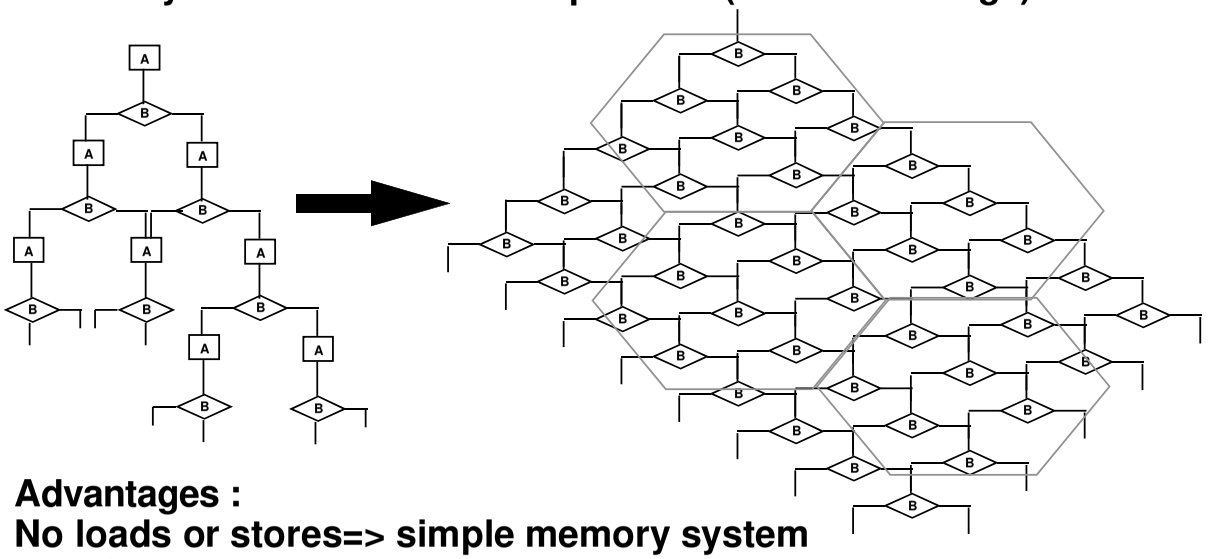
© Patterson 1994

Bad Career Move #1: Be THE leading expert

- Invent a new field!
  - Make sure its slightly different
- Be the real Lone Ranger: Don't work with others
  - No ambiguity in credit
  - Adopt the Prima Donna personality
- Research Horizons
  - Never define success
  - Avoid Payoffs of less than 20 years
  - Stick to one topic for whole career
  - Even if technology appears to leave you behind, stand by your problem

Announcing a New Architecture Field: "Control Flow"

- People use computers to make decisions; get data out of way to make decisions in parallel! ("data unrolling")



- Advantages :
  No loads or stores=> simple memory system
  No I/O beyond single LED
- Start a new sequence of courses & new journal on Theory & Practice of Control Flow

© Patterson 1994

Announcing a New Operating System Field: "Disability Based Systems"

- Computer Security is increasing in importance
  - Insight: capability based addressing almost right
- Idea: Create list of things that process CANNOT do!
- Key Question:
  should you store disabilities with each user
  or with the objects they can't access?
- Other topics: encrypted disabilities, disability-based addressing
- Start a new sequence of courses & new journal on Theory & Practice of Disability-Based Systems

Announcing yet another New O.S. Field: "Omni-Femtokernels"

- "Femto" – microkernels, only more so
- "Omni" – omnipresent, run femtokernels everywhere:
  - Operating System
  - Applications
  - VCRs
  - automobiles
- Key contribution : Employment

© Patterson 1994

Bad Career Move #2: Let Complexity Be Your Guide (Confuse Thine Enemies)

- **Best compliment:**
  "Its so complicated , I can't understand a thing you've said!"
- **Easier to claim credit for subsequent good ideas**
  - **If no one understands, how can they contradict your claim?**
- **It's easier to be complicated**
  - **Also: to publish it must be different; N+1st incremental change**
- **If it were not unsimple then how could distinguished colleagues in departments around the world be positively appreciative of both your extraordinary intellectual grasp of the nuances of the issues as well as the depth of your contribution?**

Bad Career Move #3: Never be Proven Wrong

- **Avoid Implementing**
- **Avoid Quantitative Experiments**
  - **If you've got good intuition, who needs experiments?**
  - **Why give grist for critics' mill?**
  - **Takes too long to measure**
- **Avoid Benchmarks**
- **Projects whose payoff is $\geq$ 20 years gives you 19 safe years**

© Patterson 1994

Bad Career Move #4:
Use the Computer Scientific Method

Obsolete Scientific Method

- Hypothesis
- Sequence of experiments
- Change 1 parameter/exp.
- Prove/Disprove Hypothesis
- Document for others to reproduce results

Computer Scientific Method

- Hunch
- 1 experiment & change all parameters
- Discard if doesn't support hunch
- Why waste time? We know this

Bad Career Move #5:
Don't be Distracted by Others (Avoid Feedback)

- Always dominate conversations: Silence is ignorance
  - Corollary: Loud is smart
- Reviews
  - If it's simple and obvious in retrospect => Reject
  - Quantitative results don't matter if they just show you what you already know => Reject
  - Everything else => Reject
- Don't read
- Don't be tainted by interaction with users, industry

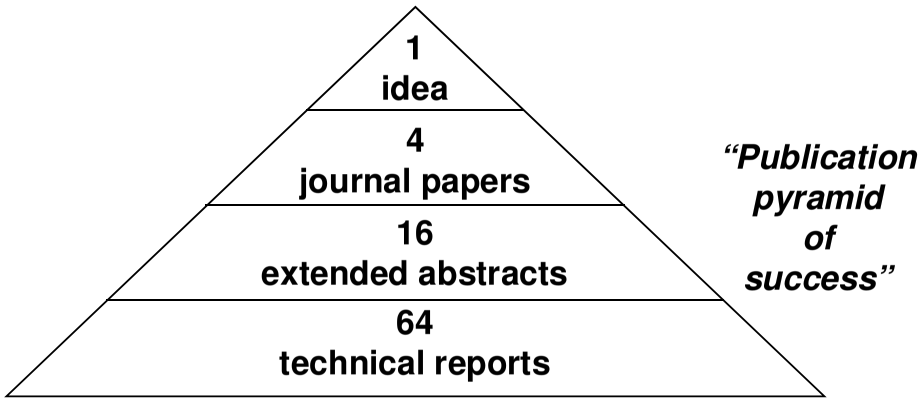
© Patterson 1994

Bad Career Move #6:
Publishing Journal Papers IS Technology Transfer

- Target Archival Journals: the Coin of the Academic Realm
  - It takes 2 to 3 years from submission to publication=> timeless
- As the leading scientist, your job is to publish in journals ; it's *not* your job to make you the ideas palatable to the ordinary engineer
- Going to conferences and visiting companies just uses up valuable research time
  - Travel time, having to interact with others, serve on program committees, ...

Bad Career Move #7:
Writing Tactics for a Bad Career

- Papers: It's Quantity, not Quality
  - Personal Success = Length of Publication List
  - "The LPU (Least Publishable Unit) is Good for You"



- Student productivity = number of papers
  - Number of students: big is beautiful
  - Never ask students to implement: reduces papers
- Legally change your name to Aaaanderson

© Patterson 1994

5 Writing Commandments for a Bad Career

I. Thou shalt not define terms, nor explain anything.
II. Thou shalt publish before implementing.
III. Thou shalt replace “will do” with “have done”.
IV. Thou shalt not mention drawbacks to your approach.
V. Thou shalt not reference any papers.

7 Talk Commandments for a Bad Career

I. Thou shalt not illustrate.
II. Thou shalt not covet brevity.
III. Thou shalt not print large.
IV. Thou shalt not use color.
V. Thou shalt not skip slides in a long talk.
VI. Thou shalt cover thy naked slides.
VII. Thou shalt not practice.

© Patterson 1994

Following all the commandments

- We describe the philosophy and design of the control flow machine, and present the results of detailed simulations of the performance of a single processing element. Each factor is compared with the measured performance of an advanced von Neumann computer running equivalent code. It is shown that the control flow processor compares favorably , given a reasonable degree of parallelism in the program.
- We present a denotational semantics for a logic program to construct a control flow for the logic program. The control flow is defined as an algebraic manipulator of idempotent substitutions and it virtually reflects the resolution deductions. We also present a bottom-up compilation of medium grain clusters from a fine grain control flow graph. We compare the basic block and the dependence sets algorithms that partition control flow graphs into clusters.
- Our compiling strategy is to exploit coarse-grain parallelism at function application level: and the function application level parallelism is implemented by fork-join mechanism. The compiler translates source programs into control flow graphs based on analyzing flow of control, and then serializes instructions within graphs according to flow arcs such that function applications, which have no control dependency, are executed in parallel.
- A hierarchical macro-control-flow computation allows them to exploit the coarse grain parallelism inside a macrotask, such as a subroutine or a loop, hierarchically. We use a hierarchical definition of macrotasks, a parallelism extraction scheme among macrotasks defined inside an upper layer macrotask, and a scheduling scheme which assigns hierarchical macrotasks on hierarchical clusters.
- We apply a parallel simulation scheme to a real problem: the simulation of a control flow architecture, and we compare the performance of this simulator with that of a sequential one. Moreover, we investigate the effect of modelling the application on the performance of the simulator. Our study indicates that parallel simulation can reduce the execution time significantly if appropriate modelling is used.
- We have demonstrated that to achieve the best execution time for a control flow program, the number of nodes within the system and the type of mapping scheme used are particularly important. In addition, we observe that a large number of subsystem nodes allows more actors to be fired concurrently, but the communication overhead in passing control tokens to their destination nodes causes the overall execution time to increase substantially.
- The relationship between the mapping scheme employed and locality effect in a program are discussed. The mapping scheme employed has to exhibit a strong locality effect in order to allow efficient execution. We assess the average number of instructions in a cluster and the reduction in matching operations compared with fine grain control flow execution.
- Medium grain execution can benefit from a higher output bandwidth of a processor and finally, a simple superscalar processor with an issue rate of ten is sufficient to exploit the internal parallelism of a cluster. Although the technique does not exhaustively detect all possible errors, it detects nontrivial errors with a worst-case complexity quadratic to the system size. It can be automated and applied to systems with arbitrary loops and nondeterminism.

Outline

- **Part I: Key Advice for a Bad Career**
  - **Invent a field and Stick to it**
  - **Let Complexity be Your Guide (Confuse Thine Enemies)**
  - **Never be Proven Wrong**
  - **Use the Computer Scientific Method**
  - **Avoid Feedback**
  - **Publishing Journal Papers is Technology Transfer**
  - **Write Many (Bad) Papers by following 5 writing commandments**
  - **Give Bad Talks by following 7 talk commandments**
- **Part II: Alternatives to a Bad Career**

© Patterson 1994

One Alternative Strategy to a Bad Career

- Caveats:
  - From a project leader's point of view
  - Works for me; not the only way
- Goal is to have impact:
  Change way people do Computer Science & Engineering
- 6 Steps
  1) Selecting a problem
  2) Picking a solution
  3) Running a project
  4) Finishing a project
  5) Quantitative evaluation
  6) Transferring technology

1) Selecting a Problem

Invent a new field & stick to it?

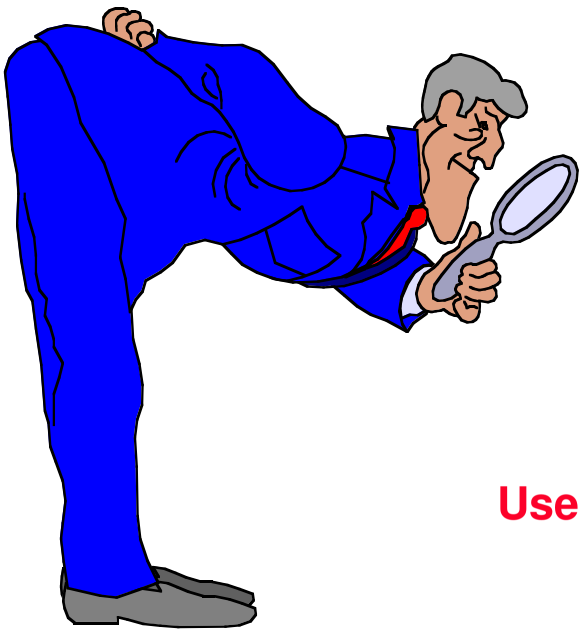
- No! Do "Real Stuff": solve problem that someone cares about
- No! Use separate, short projects
  - Always takes longer than expected
  - Matches student lifetimes
  - Long effort in fast changing field???
  - Learning: Number of projects vs. calendar time
  - If going to fail, better to know soon
- Strive for multi-disciplinary, multiple investigator projects
- Match the strengths and weaknesses of local environment
- Make sure you are excited enough to work on it for 3-5 years
  - prototypes are exciting

© Patterson 1994

My first project

- Multiprocessor project with 3 hardware faculty
- 1977: Design our own instruction set, microprocessor, interconnection topology, routing, boards, systems, operating system
- Experience:
  - none in VLSI
  - none in microprocessors
  - none in networking
  - none in operating systems
- Resources:
  - No staff
  - No dedicated computer (used department PDP-11/70)
  - No CAD tools
  - No applications
  - No funding
- Results: 2 journal papers, 12 conference papers, 20 TRs

2) Picking a solution



Let Complexity Be Your Guide?

- No! Keep things simple unless a very good reason not to
  - Pick innovation points carefully, and be compatible everywhere else
  - Best results are obvious in retrospect "Anyone could have thought of that"
- Complexity cost is in longer design, construction, test, and debug
  - Fast changing field + delays => less impressive results

Use the Computer Scientific Method?

- No! Run experiments to discover real problems
- Use intuition to ask questions, not answer them

© Patterson 1994

(And Pick A Good Name!)

Reduced
I nstruction
Set
Computers

Redundant
Array of
I nexpensive
Disks

Networks
Of
Workstations

...

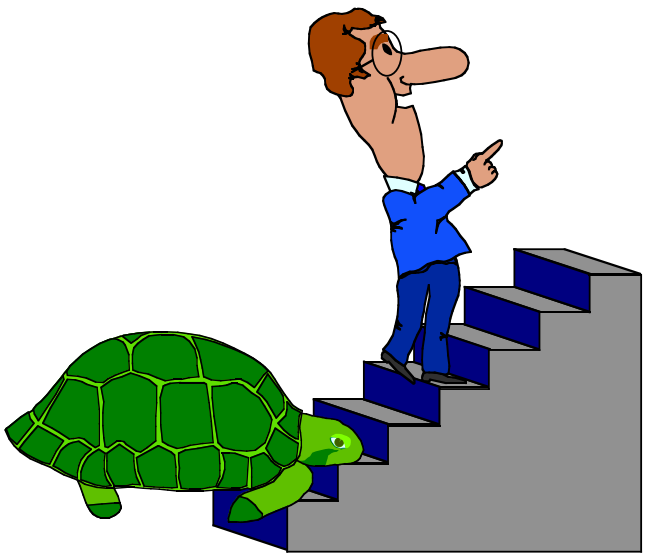
3) Running a project

Avoid Feedback?

- No! Periodic Project Reviews with Outsiders
  - Twice a year: 3-day retreat
  - faculty, students, staff + guests
  - Key piece is feedback at end
  - Helps create deadlines
- Consider mid-course correction
  - Fast changing field & 3-5 year projects => assumptions changed
- Pick size and members of team carefully

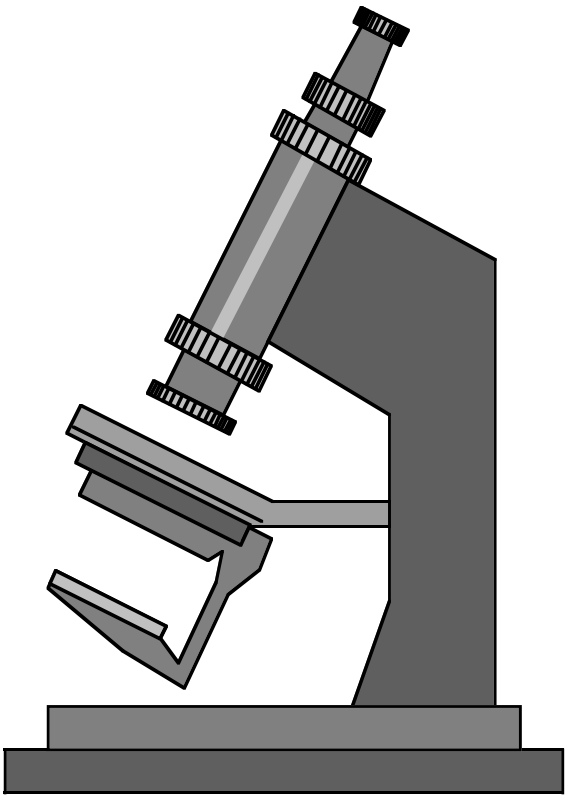
© Patterson 1994

4) Finishing a project



- ***People count projects you finish, not the ones you start***
- ***Successful*** **projects go through an unglamorous , hard phase**
  - **Design is more fun than making it work.**
  - **"No winners on a losing team; no losers on a winning team."**
  - **"You can quickly tell whether or not the authors have ever built something and made it work."**
- **Reduce the project if its late**
  - **"Adding people to a late project makes it later."**
- **Finishing a project is how people acquire taste in selecting good problems, finding simple solutions**

5) Evaluating Quantitatively



Never be Proven Wrong?

- **No! If you can't be proven wrong, then you can't prove you're right**

  "Better to curse the candle than curse the darkness."
- **Report in sufficient detail for others to reproduce results**
  - **can't convince others if they can't get same results**
- **For better or for worse, benchmarks shape a field**
- **Good ones accelerate progress**
  - **good target for development**
- **Bad benchmarks hurt progress**
  - **help real users v. sell machines?**

© Patterson 1994

6) Transferring technology (by convincing others)

Publishing Journal Papers IS Technology Transfer?

- No! Missionary work: "Sermons" first, then they read papers
  - Selecting problem is key: "Real stuff"
    - » Ideally, more interest as time passes
  - Change minds with believable experiments & by building artifacts
  - Prima Donnas interfere with transfer
- My experience: industry is reluctant to embrace change
  - Howard Aiken, circa 1950:

    "The problem in this business isn't to keep people from stealing your ideas; its making them steal your ideas!"
  - Need 1 bold company to take chance *and* be successful
  - Then rest of industry must follow

6) Transferring technology (by starting a company)

- Pros
  - Personal satisfaction: seeing your product used by others
  - Personal $$$ (potentially)
  - Fame
- Cons
  - Learn about business plans, sales vs. marketing, financing, personnel benefits, hiring, …
  - Spend time doing above vs. research/development
  - Fame

© Patterson 1994

Case Study: Kendall Square Research (KSR)

1986	KSR founded by Henry Burkhardt III
12/91	"Supercomputing: innovative entry into massively parallel computing market", *Edge: Work-Group Computing Report*
1/92	"Parallel system called easy to program," *Gov. Computer News*
2/92	"Kendall Square bucking trend by adopting custom RISC chip" ("has raised $63 million in capital"), *Electronic News*
3/93	"Henry Burkhardt III: with wit and energy, the former child prodigy and cofounder of Data General is shaking up the supercomputer industry," *IEEE Spectrum*
6/93	"Kendall Square Research Corporation reports increase in revenues & third consecutive quarter profitability," *Work-Group Computing Report*
11/93	"Kendall Square extends server; takes massively parallel-processing KSR system to next level" ("$24.7 million 1st half of year"), *PC Week*
12/93	"Executive resignation" *("*Under fire from shareholders, Henry Burkhardt III resigned last week as president of KSR. Burkhardt… and several company directors are being sued by shareholders for allegedly exaggerating revenue for the last two years."), *PC Week*

Summary: Leader's Role Changes during Project

© Patterson 1994

Acknowledgments

- Many of these ideas were borrowed from (inspired by?) Tom Anderson, David Culler, Al Davis, John Hennessy, Steve Johnson, John Ousterhout, Bob Sproull, Carlo Séquin and many others

Conclusion: Alternatives to a Bad Career

- Goal is to have impact:
  Change way people do Computer Science & Engineering
  - Evaluation of academic research uses bad benchmarks => skews academic behavior
- Many 3 - 5 year projects gives more chances for impact
- Feedback is key: seek out & value critics
- Do "Real Stuff": make sure you are solving some problem that someone cares about
- Taste is critical in selecting research problems, solutions, experiments, & communicating results; acquired by feedback
- Your real legacy is people, not paper:
  create environments that develop professionals of whom you are proud
- *Students* are the coin of the academic realm

© Patterson 1994

Backup Slides to Help Answer Questions

Applying the Computer Scientific Method to OS

- Create private, highly tuned version for testing
  - take out all special checks: who cares about crashes during benchmarks?
- Never give out code of private version
  - might be embarrassing, no one expects it
- Run experiments repeatedly, discarding runs that don't confirm the generic OS hypothesis
  - Corollary : Run experiment repeatedly, taking best case for your code, worst case for competitors code

© Patterson 1994

I/O Benchmarks: Transaction Processing

- Early 1980s great interest in OLTP
  - expecting demand for high TPS (e.g., ATM machines, credit cards)
  - Tandem's success implied medium range OLTP expands
  - Each vendor picked own conditions for TPS claims, report only CPU times with widely different I/O
  - Conflicting claims led to disbelief of all benchmarks=> chaos
- 1984 Jim Gray of Tandem distributed paper to Tandem employees and 19 in other industries to propose standard benchmark
- Published"A measure of transaction processing power," Datamation, 1985 by Anonymous et al
  - To indicate that this was effort of large group
  - To avoid delays of legal department of each author's firm

I/O Benchmarks: TP by Anon et al

- Problems
  - often ignored the network to terminals
  - ignored think time; made sense for database vendors, but not what customer would see
- Solution: Hire auditor to certify results
  - auditors soon saw many variations of ways to trick system
- Auditors proposed minimum compliance list (13 pages); still, DEC tried IBM test on different machine with poorer results than claimed by auditor
- Created Transaction Processing Performance Council in 1988: founders were CDC, DEC, ICL, Pyramid, Status, Sybase, Tandem, and Wang
- Led to TPC standard benchmarks in 1990

© Patterson 1994

Processor Benchmarks

- RISC industry mired in “bench marketing”: (“That is a 8 MIPS machine, but they say its 10 MIPS!”)
- EE Times + 5 companies band together to perform Systems Performance Evaluation Committee (SPEC): Sun, MIPS, HP, Apollo, DEC
- Create standard list of programs, inputs, reporting
- Industry stops calling each other liars, companies targeted on real programs vs Linpack and Dhrystone

© Patterson 1994