Stanley Mazor
Feb. 25 2016

More observations from my life in computing

Part 1: Computer and technical training circa 1970

1.1 Computer Training

In 1965 G. Moore (then at Fairchild) predicted a bright future for IC chip density. However the vision for IC package **pin count** was gloomy. You could get more stuff on a chip, but probably not many I/O pins for getting data in and out. In 1968 Moore co-founded Intel as a memory chip company, as memory chips have an ideal ratio of **chip gate count/package pins**. A shift register chip needs only one input and output pin whether it has 32 bits or 32 million bits; and if you double the bits in a RAM memory chip you need just one more address pin.

However, by 1970 Intel's memory business had two pressing problems:

- Salesman were not familiar with nor equipped to sell memory chips.
- Computers used magnetic **core memory** and there was little (RAM) business.

Accordingly Intel shifted to making **customer specific** chips. For Busicom's desk calculator, we created two special memory ROM and RAM chips and a new 4-bit CPU on a single chip. Based upon our progress on that 4-bit 4004 chip, we also embarked on the design of a second customer's CPU chip, the 8-bit 8008 for Datapoint.

Datapoint was using our shift register memory chips in their CRT terminals. Intel reckoned that if salesman had difficulty selling memory chips the chances of selling CPU chips to customers was nil. In 1972 most electrical engineers and logic designers were not programmers and not equipped to implement systems using computer programs. Hence we had dim prospects for these two new CPU's as **standard** products.

Thus, to promote the use of our general purpose microcomputers, Ted Hoff (leader of Intel's Application Research department) developed **design aids** to help new customers experiment with programmed solutions. We had some software assemblers, Cross-assembler, and a simulator), some coding examples, and some single board computers. Fortunately the parallel development by Intel of the EPROM chip (Intel 1701), meant that customers could try out computer programs for embedded systems with their application code stored in a field-

programmed memory chip. To support EPROM sales Hoff developed and Intel sold a standalone box for customers to easily load and store programs into their EPROM chips.

By the mid-1970s, the microcomputer market had developed and Intel management recognized that CPUs helped sell memory chips. Microcomputer manufacturers had different selling strategies. Our main competitor, Rockwell (selling the PPS-4 microcomputer chip), added staff to write customers' code. Intel, instead, developed software, development systems and boards to encourage customers to develop **their own** computer programs. Most colleges were not teaching microcomputer programming, and the huge potential customer base was still unequipped to attempt computer programmed solutions using microcomputers. So Rockwell's and Intel's selling approaches were opposites:

- Rockwell — we will do it for you.
- Intel — we will help you do it yourself.

As an application engineer at Intel, my job was to write articles, to give talks at conferences, and to assist customers with their programming without doing the coding for them. Taking a lead from mainframe computer companies, in 1975 Intel started a **customer education** (training) business center, and it offered classes to customers (both at Intel headquarters and at customer sites). In 1977 I returned from a 2-year assignment at Intel Brussels, and I joined the Intel training department with a focus on curriculum development for these training courses. By 1980 we had **five** US customer training centers and as many as 10,000 customers a year in our classes (**12** different courses).

Among the guest attendees at the classes were some college faculty. Subsequently during the next 10 years, Intel's *training business* transitioned how it operated as most engineers could learn about microcomputers at their local college

Conclusion. Intel's early and persistent approach to develop a new market for its new products was to invest in software, design aids, application engineering, and training to encourage customers to accept a new design methodology and to acquire the computer programming skills to use Intel computer products. Early generations of Intel products were adequate but **not** necessarily superior to their competitors: Rockwell PPS-4, Motorola 6800, or Zilog Z-80. Intel's success in developing and dominating this market was a combination of excellent customer support at the start and good marketing and sales later.

1.2 Technical Training

After spending five years at Intel customer training, I had improved my skills in course delivery and made gains in creating and writing curriculum. One of the general lessons I learned was that engineers were **not good listeners,** but **were compulsive problem solvers**. Hence

successful classes focused on **self-paced** hands on lab experiments that were carefully engineered to reinforce particular skills and to bring in increasingly difficult skills. Class time was about 30 percent to 40 percent lecture, and the remaining time was spent in hands-on computer labs. After lunch, attendees were particularly **bad listeners.** Thus, the best results over a 3-day course were gained from scheduling labs from 1pm to 4pm each day with lectures in the mornings divided between review (9:00-9:50am) and new material (10:15 – 11:30am).

For curriculum and subject organization, I used a trick from **bridge players**: "lead with your longest and strongest suit". As applied to curriculum that meant finding among the various topics one that was strong and somewhat long as the first topic to cover. Another lesson from **cooking**, was that complex subjects should be covered like peeling an onion, that is in successive layers rather than **canonical** (complete and total).

In my career I joined four startup companies after Intel (was a startup). At companies like **Silicon Compilers** and **Synopsys**, we faced problems similar to Intel, creating a brand new market; and we needed to change the behavior and skills of our customer base. So I often took a leadership role in activities I had done before at Intel: writing application notes, giving talks at conferences and engaged in customer training.

In several of these startups I managed the customer training business. I liked running a small business segment in a company, as I had the freedom and control to organize and manage it based upon my then 25 plus years of experience in the industry. I focused on a **high quality** teaching product with emphasis on iterative improvement and **employee bonuses**.

- Curriculum was developed by the instructors who used the course materials.
- Every course had a course manager and course reviews were run monthly.
- Attendee scores and ratings of classes and instructors were carefully monitored.
- Employee bonuses were given to instructors:
  for teaching, $100 for every student 'A' received
  for traveling, $100/day bonus for every day spent on the road
  for learning, $500 for every new course an instructor learned
  for overtime, $100 for every day of teaching exceeding 120 days/year
  for overtime, $100 for every student accepted in the class beyond 14 persons
  for development, $1,000 for every new course written

Running a small business segment in a larger corporation is fulfilling and sometimes gives the manager opportunities to experiment with innovative management opportunities.

Part 2: Software company organizations and functions

Introduction

Based on our experiences in creating markets at Intel, I applied the lessons learned in several startups that were facing issues similar to those in my Intel experience. I often focused on the training business as a way of generating corporate revenue early in the company's gestation, while developing a customer base. Two of these startup companies were in electronic computer aided design (ECAD), that are basically software companies. So I review here the conventional CAD methodology that was challenging us, and an overview of the corporate organization in these software companies.

CAD methodology—schematic capture

Circuit designers traditionally use a circuit **diagram** to create and document a circuit. It identifies individual components (coils, transistors, resistors, capacitors and their interconnection wires) at least at the logical level, if not representing their physical placement and mounting on a circuit board. (After a board was in production the coordinate location of each component was sometimes added.) Similarly **logic designers** using logic gates as their primitive elements create **schematics** to illustrate the connections and components in a design or subsystem. With the advent and use of CAD (computer-aided design) terminals, the dominant methodology was called **schematic capture**; and the computer tools replaced manual drafting on paper. Nonetheless, the overall method was the same **pictoria**l representation of a design as entered manually by the designer.

New ECAD methodology

By **1990** higher level design tools became available to automate some of the lower level design details and to represent the design at a higher level than component interconnection. The objective of this more abstract design was to improve productivity and to automate (without error) the translation to lower level composition. By 1988 I was working on a **"silicon compiler**" to generate logic and circuits, and later at Synopsys we 'synthesized' a design using a program like **"design language".** However the customer base of engineers was not eager to adopt a new methodology and programming was not necessarily a skill they had, nor wanted to adopt. Hence I had a sense of *Deja **View**,* all over again!

Software company organization

The **Marketing** department is split into product marketing and customer marketing. Launching a new product is the primary focus of product marketing. Customer marketing is organized either by sales geography or by sales customer type (consumer products, automotive, electronic systems, etc.). Marketing's main function is to support product sales, although the **Sales** department is a separate department reporting to the VP of Marketing and

Sales. Software releases are done either quarterly, semi-annually, or annually. Product marketing selects the major features to be included in the next release. The **Engineerin**g department writes new software and new releases and revises software with cooperation of product marketing's priorities. **Quality Assurance** is either under the direction of Customer Support or Engineering. Their primary goal is to test and validate new software releases.

Documentation, Customer hot line **Support**, and customer **training** might report to the director of customer support (sometimes within the marketing department). Customer support's top priority is reporting product bugs, using a bug reporting system that usually involves a test case and a priority (customer's urgency). A secondary goal is to assist customers in using products, and customers must attend a training class before they can receive individual support. Feedback is established bi-directionally with training and documentation. Since documentation and training are synchronized with the product or new release, there is a dependency.

Customer Training offers timely and high quality training services that meet these objectives (priorities change year to year):

- Promote market penetration and good software use
- Reduce customer support costs
- Operate at a profit
- Train employees

Synchronous with the offering of a new software product or new release, training courses reflect the latest software (within reason). Training classes should deliver and familiarize customers with product documentation—to answer questions and reduce the demands of customer support inquiries

Part 3: Training business metrics circa 1995

The trainer is paid $50k salary and is a recent EE or MSEE graduate with no prior product experience. With a loaded cost of $80k/trainer, they should deliver 120 days per year (30+ classes) of customer paid training. Assuming a paying audience of 10 persons at $300/day, generate, trainers should generate $3k x 100 or **$300k** in annual revenue. Assuming 225 work days/year the trainer spends approximately 50 percent of their time teaching and the remainder in course maintenance, learning, mentoring, or supporting other corporate functions. With excellent curriculum, a new hire can teaches a basic course himself or herself in **6 weeks.** If a company's classes have 1,000 attendees/year x $1k/person this is $1M gross revenue, from about 80 classes and a staff of about 3 instructors and a manager at a cost of about $500,000. (See also the paragraph near the end of page 3 about training employee bonuses.)

The average new hire stays in the training department for at least one year, but most transfer to another department, having gained product experience, and customer savvy. On the other hand, experienced employees want to join the training department as the job has **predictable** characteristics. To a large extent teaching classes is a 'manufacturing' activity and hence the organization is *not compatible* with most of the activities of a marketing department. In particular at product launch (after 2 plus years), product marketing is *done*, but training will continue with the product for another 5 to 10 years.

Class Methodology

A course is 3, 4, or 5 days long, and usually starts on a Monday. Customers travel on Sunday, and their travel expenses usually exceed their tuition costs. Most customers send two employees together, and a slight discount is taken for multiple attendees in the same class. All registration is done by the training registrar. Tuitions are paid in advance of attendance. Software sales **bundles** a single tuition with the product sale; use of customer support supposes class attendance. A single instructor runs the class, and has a list of attendees, to contact if there are questions on subject matter or prerequisites before the class begins.

Our classroom design has relatively wide rooms with 20 inch narrow tables seating about 6 to 8 persons in a row. Chairs were either rocker, or at least S-sling, chairs with arms. There is always a giant white board, and the computer lab is separate. We only use one colored marker on the board -- BLUE. Overhead slides are shown using a projector at the front manually fed by a standing instructor.

Classes run from 9am to 4pm, and lunch and coffee is provided. (It is my belief that sandwiches in a class is not a differentiator; therefore, my policy was to take the class out to lunch at least once during the class.) Assuming a reasonable class size for a single instructor is 14 persons, the typical breakdown might be: 10 paying customers, 2 customers free, a company employee and a university professor. Having more than 14 attendees is only a slight burden on the instructor in lecture but is a real burden in lab. We generally form the participants of a class into 7 **two-person teams**, each team has its own computer. Although one team member might take the lead, generally good teamwork is exhibited.

The key to the training class is excellent visuals (slides) and very well engineered lab exercises. This ensures high quality training and allows a *junior instructor* to succeed. In most classes we handed out a set of technical manuals and on some occasions an auxiliary hard copy book. Gifts are not generally exchanged in business, but we manage to bundle the books in an attractive book bag.

Successful **training** requires that a change of behavior is demonstrated at the end of the class by the attendees, and the completion of the lab exercises is that demonstration. Excellent labs, start with a *cookbook style* where *almost nothing* can go wrong. Subsequent exercises have

less specification and increment in difficulty and pace, gradually demonstrating skill improvement. Some labs involve mystery, tricks, or discovery -- adding to the dramatic impact and overall effect of mastering the subject. The instructor facilitates lab completion as needed on demand by a team requesting help.

We us4 overhead (black) and clear overhead transparencies, for which a book was provided with **2 images/page** of our own format and design (in a loose leaf binder). Assuming a unit of 50 minutes, there would be approximately 25 transparencies. The emphasis was on **pictures** as they are worth 1,000 words. In those case when text is used, up to 5 bullets and only **sentence fragments** are allowed. A font size minimum of 24 points is used for good display. Color was *never* used, and is to be avoided in my opinion. Additionally an instructor most often writes on the slide as he speaks using a dark marker for emphasis.

Often a training manager or course manager audits a class to monitor the instructor or the class material (and who had to be invisible and SILENT). Courses are generally revised MONTHLY as a result of instructor feedback or auditing.

I have an unusual perspective on questions from attendees — there shouldn't be any, as they are indicative of flawed curriculum A class ends with a written critique questionnaire of one page, that is used to gauge class success. It can be done anonymously and takes less than 4 minutes to complete. These are reviewed after the class is completed by the instructor and by the training manager.

The course or class training brochure details the class outline, and the preface guarantees **money back** for any reason. In the end **satisfaction is**: meeting or exceeding expectations; thus, the brochure must set expectations correctly! As we had less than a tenth of a percent refunds, I believe that an **unconditional refund policy** drives excellent quality!

Biography

I started using computers in college at SFSU in 1963, where I was a student assistant in our computer lab for the IBM 1620. Later I joined, Fairchild as a programmer and logic designer, and then in 1969 joined Intel as a startup. There I worked on early microcomputer architecture and software. By 1983, I departed Intel to work in four software startup companies.

I have published more than 60 technical articles and several books. In the last decade or so I have recounted some of my memories about my time in computing, particular in the *IEEE Annals of the History of Computing* and on the website of the IEEE Computer Society History Committee.

IEEE Annals of the History of Computing (Anecdotes Department)

- Micro to Mainframe*,* Vol. 27, No. 2 (Apr-June, 2005) pp. 82-84.
- 8 Bits of Irony*,* Vol. 28, No. 2 (Apr-June, 2006) pp. 73-76.
- Intel 8080 CPU Chip Development*,* Vol. 29, No. 2 (Apr-June, 2006) pp. 70-73.
- Fairchild symbol computer, Vol. No. 1 (January-March 2008), pp. 92-95.
- Magnavox and Intel: An odyssey, Vol. 31, No. 3 64-67, (July-September 2009), pp. 64-67 (written with Peter Salmon).
- Intel's 8086*,* Vol. 32, No. 1 (January-March 2010) pp. 75-79.

At http://history.computer.org/#cat-mazor

- Reflecting on Intel and the invention of the personal computer, May 2012
- Musings about Gordon Moore's Prophecy, May 2012
- A Little Non-standard Intel format (BPNF), December 2011