An interview with
Charles W. Bachman[1]
OH XXX

Conducted by Thomas Haigh
On
25-26 September, 2004
Tucson, Arizona

Interview conducted for the Special Interest Group on the Management of Data (SIGMOD) of the Association for Computing Machinery (ACM). Transcript and original tapes donated to the Charles Babbage Institute.

Charles Babbage Institute
Center for the History of Information Processing
University of Minnesota, Minneapolis

[1] The transcript has been edited by Thomas Haigh, and subsequently edited and clarified by Charlie Bachman, his wife, Connie Bachman, and Russ McGee, who was a major player in the following story.

ABSTRACT

Charles W. Bachman reviews his career. Born during 1924 in Kansas, Bachman attended high school in East Lansing, Michigan before joining the Army Anti Aircraft Artillery Corp, with which he spent two years in the Southwest Pacific Theater, during World War II. After his discharge from the military, Bachman earned a B.Sc. in Mechanical Engineering in 1948, followed immediately by an M.Sc. in the same discipline, from the University of Pennsylvania. On graduation, he went to work for Dow Chemical in Midland, Michigan. Bachman discusses his experiences as an engineer there, focusing on engineering economics issues, including his work on the development of methods for return-on-investment calculations, using punched card machines. In 1957, he was appointed the first head of Dow's Central Data Processing department, where he was responsible for preparing the way for the arrival of its first big digital computer. Bachman organized a feasibility study to select a new machine and hired a staff of programmers and analysts. Through the IBM SHARE user group and its 9PAC project, he became involved with Harry Tellier, and Russ McGee in the design of programs to simply the file maintenance and report generation process. This was based on the earlier GE Hanford 702 File Maintenance and Report Generator System, which Bachman discusses in some detail. He also worked on the "theory of information" with Bill Orchard-Hays and Tom Steel. However, Dow canceled its IBM 709 order before the machine arrived, ending Bachman's association with SHARE and, shortly afterwards, with Dow Chemical.

In 1961, Bachman went to work for General Electric in New York City. He worked for a corporate staff group providing internal consulting services, and was assigned to an "integrated systems project" to produce a generic manufacturing control system for GE departments, using the firm's new GE 225 computer. Here he worked with Stan Williams and Bill Helgeson. This system, MIACS, an early transaction processing system, was built around Bachman's new creation: the IDS (Integrated Data Store). IDS is called the first database management system, and Bachman explains its origin, development, capabilities, design and relationship to MIACS in detail. IDS used a virtual memory database, with a DDL and DML, and introduced the network data model. Bachman created the first data structure diagrams to illustrate the complex data structures required to picture the adaptive manufacturing control logic of MIACS. In 1964, Bachman transferred to GE's Computer Department in Arizona, where he worked on a number of other database related projects including the GE 400 IDS, GE 600 IDS, DataBASIC, a personal data storage system, and WEYCOS 1 and 2, both built for Weyerhaeuser Lumber. The latter two were an attempt to build a complex online management information system around a database. Bachman believes WEYCOS 2 produced the first database management system able to support multiple application programs executing simultaneously against the same database. Bachman also describes his work with Warren Simmons, Bill Olle and others on the CODASYL Data Base Task Group during the late 1960s. The database standards produced by this group were heavily influenced by IDS and by Bachman's ideas. Bachman also discusses the links between IDS and Cullinane's IDMS, a successful commercial system, built to run on the IBM 360 series of computers, which was based on IDS.

When Honeywell acquired GE's computer business in 1970, Bachman worked for the merged operation in Honeywell's advanced research group, in Boston. He continued to work on database topics, investigating what he called the "role data model" and serving on an ANSI-SPARC committee intended to standardize database systems. While failing in this objective, this committee made a highly influential distinction between external, conceptual, and internal (database) schemata. In 1973, Bachman received the ACM Turing award. He recalls his reactions to this, and his subsequent service on committee to select future winners. Bachman was elected a Distinguished Fellow of the British Computer Society in 1977. Bachman also discusses his thoughts on the relational data model and its creator Ted Codd, including a well known 1974 meeting in which he debated Codd on the merits of the two approaches. Bachman also chaired the ISO committee working on Open Systems Interconnection (OSI), well known for its elaboration of a seven layer model for computer communication.

In 1981, Bachman went to work for Cullinane Database Systems, a major software company built around IDMS. While at Cullinane, he supported marketing, and continued the ISO/OSI committee work, while working on tools to model high level schemata for enterprise database design and translate them into data definitions. He also worked on the "partnership set data model," which was granted a US patent, and used as the basis for a enhanced network data model and conceptual schema modeling. In 1983, he left to pursue this work at his own firm, Bachman Information Systems. He describes the rise and fall of this company, which obtained venture capital funding, grew rapidly and made a successful IPO. Despite enjoying success with its Data Analyst product with its reverse and forward engineering capabilities, the firm struggled financial and managerially, including an initially successful, but ill-fated, spell as an IBM Business Partner. It eventually merged with Cadre Systems to create Cayenne Software, which was acquired by Sterling Software in 1998, which was subsequently acquired by Computer Associates. Since then Bachman has worked as a database consultant to Constellar Systems and in 2006 was still working with Cbr Systems.

HAIGH: Thank you very much for agreeing to take part in this interview.

BACHMAN: Thank you. Good morning.

HAIGH: I wonder if you could begin by talking a little bit about your early life and family background.

BACHMAN: I was born in Manhattan, Kansas[2]. I moved shortly to Gainesville, Florida. And then, by the second grade, I had moved to East Lansing, Michigan. Now, this is what happens when your father is a football coach. Charley Bachman[3] was the head coach at Kansas Agriculture College[4], when I was born. Then, he went to be the head coach job at The University of Florida, and then on to Michigan State College, now Michigan State University. He was a very successful football coach, so part of my life comes from that background. I'm always interested in sports. He was a graduate from Notre Dame, and I happen to have a ring for years that he was given as being the best scholar on the football team, although we don't think of scholars and football teams going together.

My mother[5] was a college graduate from the University of Oklahoma, before World War I. She had gone back to graduate school at Kansas State where she met my father. He was the football coach. They got married, and by and by I came along, number two in the family. I came from an academic background, although if you talk to people from academic backgrounds, they say, well, there's the academics on one side, and then there's the athletic department and the military department (ROTC) on a different side, and they're different communities. They do not mix very well. That's true. But certainly I grew up knowing college backgrounds and was interested in things out there, although I was never a good athlete myself. I had fun in high school. I played football and basketball and hurtled and high jumped for the track team, but I was never a star athlete, as my younger brothers were. Cary was terribly interested in sports and built his life and career on it. I went to lots of football games when I was young, maybe too many.

HAIGH: When you were in school, what kind of topics were you interested in? What kinds of subjects did you do better in?

BACHMAN: In East Lansing (Michigan) High School, at least when I was in high school in the 1940, '41, '42, and '43 time frame, everybody took much the same courses, with a few options for extra study. I enjoyed all of my courses. I was a good student. Things seemed to come easily. I enjoyed the math part, I enjoyed the science part, but then I enjoyed English and other things. I was not the best student in my high school class, but always in the top 10%. But, I don't think anyone-- Well, I got an e-mail from someone the other day saying, "Well, Charlie, you're the brain!" Well, I never thought of myself as that much of a brain, but I did well.

At the end of my high school career—I was scheduled to graduate in June, 1943, but we were in the middle of World War II—I left high school early and started college at

---

[2] December 11, 1924. Except where indicated otherwise, footnotes have been inserted by Bachman.
[3] My father's name was "Charles W. Bachman." He was born on December1, 1892. He was better known as "Charley Bachman." My grandfather's name was "Charley Bachman." I am known as "Charlie Bachman."
[4] Now Kansas State University.
[5] Grace Cary Bachman, June 21, 1895.

Michigan State College. I wanted to get some college training before I went into the Army. I had enough credits to graduate from high school, so I left and started college in the middle of January and completed a half of a year of college before my high school class graduated in June 1943. I finished all of the freshman year requirements by the end of summer, and then I elected to go into the Army.

HAIGH: Why did you want to have some college before you went to the Army?

BACHMAN: It seemed like a good thing to be prepared. I wanted to differentiate myself from the rest of the troops. It helped with my assignment into the Anti Aircraft Artillery Corp. It probably helped in my being selected for the Officers Candidate School (OCS) in Australia. I'm sure the year of college and the year in the ROTC helped me to be selected for that program.

HAIGH: So when you went to Michigan State College initially, you had already selected engineering as your field of study.

BACHMAN: Yes.

HAIGH: What attracted you to engineering?

BACHMAN: Well, all I can really say is that I was never interested in anything else. I was always going to be an engineer.

HAIGH: When you were younger, did you have technology-oriented hobbies?

BACHMAN: Well, I was pretty much interested in architecture, for one part, and photography, for another part. When I was 11 years old, my father and I built a racer for the Soapbox Derby. I don't know if you've ever heard of the soapbox derby or not, but it's a national event where cities have racers built by kids, typically with the help of the parents, their fathers. They essentially had a standard four wheels you bought from the Soap Box Derby organization. Then you built a coaster with whatever you want to hold you and to steer this thing. You try to find a hill, and on a straight street you can coast down. The one who gets to the bottom of the hill first wins. We went through a series of elimination races and I actually won in Lansing, Michigan finals. (East Lansing is where Michigan State College is; it's a suburb of Lansing). I went to Akron, Ohio where I was able to compete in the national event. I became incredibly airsick on that flight down from Lansing, Michigan to Akron. Among the things that they tried to do to make me feel happy, they asked Eddie Rickenbacker, the World War I flying ace, to called me on the phone. He was the chairman of Eastern Airlines who were one of the sponsors of the soapbox derby. So I had this World War I hero calling "Buddy Bachman" (as I was called in those days) to see how I was doing. So probably the most fulfilling part of the Soap Box Derby was talking to Eddie Rickenbacker, one of the real flying aces of World War I.

HAIGH: Was Michigan State College considered to have a strong engineering program at that time?

BACHMAN: I'm not sure I knew. I probably thought that The University of Michigan had a stronger one, but they were the "enemy" from a football point of view. I couldn't imagine going to The University of Michigan. I started at Michigan State and I was lived at home the first year. The second year, I was in the Army.

HAIGH: So you returned to the same program?

BACHMAN: Yes.

HAIGH: Now, in your earlier interview with Philip Frana of The Charles Babbage Institute, you discussed in some detail your WWII experiences. I would just ask a follow-up question on that. Is there anything specific you can think of from your experiences in WWII that meant that when you came back and resumed your studies that your interests had changed, or that your career plans had changed, compared with the person who had gone away to war three years earlier?

BACHMAN: I don't think so, although the WWII experience was very valuable to the development of Integrated Data Store. Maybe we'll get to that impact when we get to that part of my history. I was in the Pacific Theater for 2 years, mostly in New Guinea, but also in Australia and the Philippine Islands.

HAIGH: Okay. I'll make a note.

BACHMAN: But this about the whole strategy of anti-aircraft guns and how successful they were.

HAIGH: Oh, you did discuss that in the interview with him. You were talking about the targeting requirement for the enemy to fly in a straight line and how, unfortunately, they were reluctant to do so.

BACHMAN: We were firing at Japanese aircraft, with our anti-aircraft guns. They would never fly in a straight line. This fouled up our whole strategy of the anti-aircraft computer, being able to optically track an airplane and predict its path. If I see a plane at point one, point two, and point three, and then have the computer could draw a straight line through those points, where that plane was going to be in 15, 20, or 30 seconds, when we could get an explosive shell to where the plane was predicted to be. But, Japanese bombers that were bombing Biak Island[6] during 1944, didn't fly straight. For a long time, I thought our strategy was stupid, until I realized their bombsights also assumed they were going to fly straight. So they didn't hit anything important. I mean they hit something, but they were just making an awful lot of noise, and so did we. We did not his the Japanese bombers, either. Their bombings were missed the target and, our anti-aircraft missing the target too.

This antiaircraft fire control system became important to think about in 1961, when we began looking at the GE manufacturing world, and their missed targets. My concern was, "How do we fix the GE problem?" We'll come to that later.

HAIGH: Yes. Now, as you completed your undergraduate engineering studies, were there any particular parts of the curriculum or any projects you were involved in that you found particularly interesting and helped to develop your interests?

BACHMAN: One of the things that I thought was interesting was thermodynamics. I asked Dean Dirks, the dean of Engineering at that time, "What were the best East Coast graduate schools for thermodynamics?" He told me MIT and The University of Pennsylvania. Having been on the West Coast during the War, I said, "I'm going to go

[6] Biak Island was part of the Dutch East Indies and is located off the northwest coast of New Guinea.

the East Coast and see something different." It was the summer of 1947. I went by Cambridge, Massachusetts and I wandered around MIT for a day and never did find anybody who really wanted to talk to me about making it to graduate school. That probably was my being an amateur in trying to interview, but I went on down to University of Pennsylvania in Philadelphia and ended up very quickly in the Office of the Dean of Engineering, John A. Goff, who turned out to be a life-long friend. And so I knew where I was going to go to graduate school. It was Pennsylvania.

Goff had done work during WWII on torpedo design and in particular, torpedo engines. I later learned how they use thermodynamics to make sure they ran properly, until they got to where they were supposed to go. There were a lot of engine failures in the early days. They did not run the engines hot enough to keep the unburned carbon in the fuel, in a vapor phase, and it would clog up the engines.

HAIGH: So thermodynamics was the main interest that you took out of your undergraduate engineering program?

BACHMAN: Well, that's what took me to The University of Pennsylvania. I say today I don't really understand why I wanted to take thermodynamics, but it popped up from somewhere in my mind, and that took me to Pennsylvania.

HAIGH: Now, as I understand it, you went to Penn directly after finishing your undergraduate studies.

BACHMAN: That's correct. I graduated in June 1948, worked construction during that summer, and went to Penn in the fall.

HAIGH: Now, I also understand that you were in some kind of joint degree program between the engineering school and the Wharton Business School.

BACHMAN: Well, let me tell you history of that story. When I arrived in Philadelphia in the Fall of 1948 to start graduate school, it wasn't until I arrived there that I had realized that the engineering graduate school of Penn was a night school.

HAIGH: The entire graduate school or the engineering graduate program?

BACHMAN: Yes. Penn's graduate engineering school was a night school. Most of the people in that night program were people working in local industry. I was talking to one of the classmates there, Frank Surowic, who was in that night program, and he said, "Well, why don't you take some courses at the Wharton School?" So I signed up with a business management course with a very famous Industrial Relations professor named, Waldo E. Fisher, for the first semester, which I really loved. The second semester, I think took three courses at the Wharton School. This was totally a Charlie Bachman's ad hoc, joint degree program. I had the time in the day to do things, and having registered at the Towne school in engineering, I could also take courses at the Wharton School in their graduate program. When I first talked to Dean John Goff originally, he said, "Well, you can get a master's in three semesters, but it'd probably work out better if you took four semesters." So, I planned to take two years at my master's degree.

The fall of the second year, I asked Dean Goff, "What should I do about a thesis?" And he looked at me and responded, "Wouldn't you rather take more courses at the Wharton School?" And I said, "Yes." That was the end of the engineering thesis subject.

HAIGH: So you didn't write a thesis in the end?

BACHMAN: I did not write a thesis in engineering, nor did I write a thesis in the business school because I would have had to be part of their two-year program. I just took their coursework for two years. In one more semester I could have gotten an MBA, but I had been in school for a long time, I was ready to go to work and make some money.

HAIGH: Was this interest in business something that was unusual among engineering students at this point?

BACHMAN: I don't know the answer to that question.

HAIGH: Are you aware that any of your classmates were taking any courses at Wharton.

BACHMAN: My engineering classmates were all people working fulltime in

HAIGH: So doing one degree at a time was enough for them?

BACHMAN: Yes. Well, that's about all they had time for. I actually had all day long. I wanted to do my homework for my night classes in engineering and had all day long to find something to do in the Wharton School. I went there and "cherry picked" the courses that I thought were interesting. I had not had an accounting course before. I had not had an economics course before. And I also took labor management courses. It turned out to be very fortunate even though none of them were directly applicable to anything. The background was very important. When I arrived at The Dow Chemical Company in July to start work, I found out that two of the people they had hired that spring were classmates[7] of mine at the Wharton School. So, the engineering assignment at Dow took me from doing plant design engineering into engineering economics, to business systems, into production and back into business systems. My Wharton School experience helped me attack those jobs.

HAIGH: Now, until you began taking those courses at Wharton, did you have any particular interest in business?

BACHMAN: I don't think I did consciously. I grew up in a university environment. The business people I knew were local business people. The father's of my high school friends were local businessmen, or doctors, or lawyers. One of them was the head of the General Motors Oldsmobile Division in Lansing. He was a regular guy. But I had no real focus on business. In fact, I'm not sure I knew what businesses did, except deliver products and services to people. If you live in Michigan, you know about General Motors and Chrysler and Ford. I think at one time, I described my ambition, after getting out of college, was to reach some senior engineering job at General Motors and be making four times my starting salary[8]. So with that perception, I would always be in engineering.

HAIGH: You thought of businesses as things that made things.

BACHMAN: Yes.

[7] Charlie Kline and John Donalds

[8] The starting salary for engineers, in the 1948-1950 time frame, was between \$225 and \$250 per month. It was \$250 to \$300 per month with a Masters degree.

HAIGH: Now, were there any courses related to business systems work in the Wharton curriculum at that point? Did you have exposure to anything on, say, forms design or procedure writing or any of that systems work?

BACHMAN: Nothing of that type at all, no. I'm not sure if Wharton even taught anything like that. If you compare it with the Harvard Business School. Harvard taught by using cases. Wharton tended to be on the theoretical side of things, so designing forms would not have been part of their curriculum, at least not any part I saw.

HAIGH: Now, were there any particular courses or areas that you did take at Wharton, or professors that were teaching the courses which you found particularly stimulating or gave you ideas which became important later?

BACHMAN: There's one quotation that I have remembered for years came from one of the professors (Waldo E. Fisher). Actually, it was from one of the papers he gave to us. The author's name was William Graham Sumner. That name just popped out of somewhere, I don't know. His statement was, "A free man in a free society owes nothing to anyone else except courtesy, respect, and goodwill." That sounds like the right wing theory of the world. I always thought it was very good, although I tend to be more socially minded than that. It's a very narrow minded pronouncement.

HAIGH: Libertarian to some extent.

BACHMAN: I guess so. A few things that stick with you for some reason, and they must all come back and pop out from who knows where in your memory. That was in Waldo Fisher's class of industrial management. He was a very conservative minded professor, an outstanding guy. I thought highly of M. Thomas Kennedy, another professor there at that time, whom I talked to later, when he moved up to the Harvard Business School. He was there when we moved to Boston in 1970. But, I think in some sense, both Wharton and the Towne School of Engineering provided an environment with some faults. It's hard to focus on any one thing that really was outstanding. It kind of filled you in. It's almost like a liberal education fills you in. It provides you a background within which you can improvise. You have all this material that you initially don't know what it's for, but that goes to work sometime later for you.

HAIGH: So by the time you graduated, did you have a sense of what your career goal was, what kind of job you would want to have, and what you hoped to do with your life?

BACHMAN: I hadn't the faintest idea, although, at that time, my wife[9] and I did a survey. We said, "We think we want to settle in a small town in the Great Lakes area, with between 25,000 and 50,000 people." We sent letters out to 50 different Chambers of Commerce looking for towns, the businesses were there, schools, etc. In other words, I was looking at it from a home-building point of view, and not from a particular career interest. As it turned out, nothing came out of the interviewing process at Penn. This was March, April, May, and June of 1950 when business activity was slow in the United States. And so having looked at all this information from these various little towns around the Great Lakes area and the Middle West, my wife and I actually went and spent

[9] Connie Hadley and I married on June 25, 1950, after she graduated from Michigan State College. She was an art history major. Her father was the head of the architectural department at the General Motors's Fisher Body Division. She shared my interest in art and architecture.

a month in Connie's family's cottage in Northern Michigan. The Korean War came crashing in on us in late June 1950. Friends who did not live in Midland, Michigan but who would work with Dow Chemical later, said, "You should go talk to Dow Chemical in Midland." The population at that time is 15,000 people, which was too small for our survey. We had wanted a town between 25 and 50 thousand. So we went to interview at Dow, and it turns out they had a job in the engineering department, because of the war, they were going into an expansion program.

Midland was the home of Alden B. Dow, the architect and son of Dr. Dow, the founder of The Dow Chemical Company. Alden had designed my parent's home in East Lansing in 1937. He had been a student of Frank Lloyd Wright. Our very contemporary home in East Lansing is where my interest in architecture began. Incidentally, Connie and I drove to Spring Green, Wisconsin to interview Mr. Wright about joining the Taliesin Fellowship on our honeymoon the summer of 1949 after she graduated from Michigan State College. And, Connie's father[10] was the head of the GM Fisher Body Architectural Department in Detroit.

So I took a job to work as an engineer, at a drafting table, in a room with more than one hundred engineers. Engineers worked, standing at drafting tables, in those days. You'd sit on a stool and you'd design factories, machines and other things. I was in the engineering department working for a man named, Aaron Brooks, who was a very good boss. He was in charge of all the engineering for water, steam and electric power projects. The Dow Chemical Company is a very highly integrated company, which means that they made product A, which is used to make product B, which is used to make product C. And, Dow also sold some of the intermediate products produced along the way. When Dow started, their first business began with pumping bromine rich water out of the wells that had drilled down under Midland, Michigan. They had built a steam plant because they used a steam process to extract the bromine chemical from the water. Then they started using high-pressure steam to make electric power and using the low-pressure steam leaving the electric generator turbines for the bromine process. They had a very substantial coal fired generation system generating both steam and electric power for process purposes. This led them into making chlorine, which they separated electrically from salt water (brine) to make caustic soda. So, they had a caustic soda business and a chlorine business, along with their bromine business.

HAIGH: So it was a very process oriented business.

BACHMAN: Very process oriented, yes, and many levels of processing. When I first arrived, they sent all the new engineers they hired out into the factory on assignments. They worked, first in the pipe shop, then the electric shop, then the plumbing shop, then the lead shop, then the instrument shop, etc. The shop journeymen worked through out the entire Midland plant, covering several square miles. The journeymen knew we were going on to work in the engineering department and that the practical plant experience would be invaluable. It was a good introduction. I don't believe this apprentice approach exists anymore. How one welds something, is not a skill they expect engineers to practice. That's somebody else's job. However, supervising plant maintenance and construction is very important to the engineer who takes a plant supervisory position. The

[10] Thomas Erle Hadley.

engineering department is where most future plant managers started at Dow. It proved important to me in my career at Dow.

HAIGH: And they wanted to expose you to many different areas of the business?

BACHMAN: Yes. I went to work for Aaron, but for nine months it was field training or factory training. The first problem Aaron posed to me, when I finally got to my drafting table and stool, was very challenging. He said, "We have a choice between installing two different water valves in a new water main. One valve cost $5,000 more than the other." This is at a time when $5,000 would buy two new Oldsmobile cars…two pretty good automobiles. This question really made my brain perk up. "What are you talking about?" He said, "Well, we're talking about two stainless steel, gate valves." Gate valves are ones that open completely, leaving very little resistance to the flow of whatever is going through them. He said, "The difference between these two valves and the reason why one costs $5,000 more dollars than the other because it is more streamlined. Therefore, there's less pressure drop across it." More pressure drop means we will have to pump harder, on the upstream side, to pump the water through this water valve. He said, "Tell me how to solve this problem and which one of these valves did you should buy." The economic tradeoff was to pay $5,000 in capital cost now, or spend a number of of electricity, pumping water through the next 20 to 40 years.

The result of this project was an economic evaluation system, which the whole engineering department came to use, that was called "Equivalent Capital." There was a set tables saying, "Compare capital costs for buildings or equipment, which could be depreciated in 5 years or 10 years or 20 years or even 40 years, with the equivalent capital cost of labor, steam, or electricity supplying an alternative process over the same 5, 10, 20, 40 years. "How do we make that tradeoff?" I started looking at it trying to figure it out… Along the way, I reinvented something, because I didn't know it. It is called "discounted cash flow," which is a normal thing every business person is supposed to know about. I didn't.

HAIGH: They didn't teach anything like that at Wharton at that point?

BACHMAN: None of the courses I took discussed discounted cash flow.

HAIGH: Or like the net present value calculations?

BACHMAN: Yes, absolutely. As part of this discussion, I said, "Well, why not build these formulas into tables so the engineers can do these calculations easily?" I had to build in a "return on investment." But, what return on investment does Dow expect on such things?" I set off down the hall to talk to Carl Gerstacker, who was then the treasurer of the company. He became the chairman of the board of directors of the company some years later. I said, "Mr. Gerstacker, I need to have you tell me what returns do you want on your investments so I can put them into the tables." He said, "Our Board of Directors makes those decisions." I said, "But, Mr. Gerstacker, engineers are making decisions everyday that you don't know about. We're trying to help them make good decisions. Can you give me a range of returns on investment you want, on various kinds of projects?" So, if it's project type A or B or C… if it's a steam and power project then they have one kind of return on investment you want. If it's a new unproven plastic project, you will want something different out of it. If you want to do an agricultural chemical, you want a different return. If you could help me get answers about five or six different categories of risk and depreciation, then we could build those into the tables. When the board of directors authorizes a project, they can specify the parameters for that project. Then the engineers go from there. That Equivalent Capital system was set up early, probably 1952. The last time I was in Midland to talk to someone, about 15 years ago, they were still using that system. all had better calculators than we had in the early 1950s. A desk calculator that could do square root cost $800, a Frieden calculator. That was a very special one. If you just wanted to add and subtract they cost a little less. They were all gears and motors.

HAIGH: Were you exposed at all to punch card machines during this period?

BACHMAN: Yes. Because of the Equivalent Capital project and my introduction to Carl Gerstacker. He contacted me sometime in late 1952, and asked me if I would come down and work on a project for him. His project was one that was designed to reflect on the fact that Dow was a highly integrated company. We have an Aspirin plant. Aspirin is acetylsalicyclic acid. He said, "To make acetylsalicyclic acid, we have to make monochlorobenzene. To make monochlorobenzene, we have to have phenol, and we have to have chlorine. We had to have electric power to make the chlorine. How much capital is really invested in capacity to make a thousand pounds per month of aspirin? He said, "We don't really know." He said, "Can you develop a system?" He said, "This would be your second information system."

HAIGH: And didn't it have a mechanical component?

BACHMAN: In order to do that system, the only tools I had were the punch card systems that we had available. That was my first introduction to punch card systems. I actually spent the better part of two years collecting the data from all the Dow plants in Michigan, Texas, and California[11]. I used this to set up a number of equations: what the plant capacities were, what was their original cost to build, what was their depreciated cost, what would be their replacement cost, what raw materials were required, and what were the unit ratios (how much chlorine, or steam, or something needed to make the product. You might think these were sequential equations, because you use electricity and brine to make chlorine, and chlorine and phenol to make monochlorobenzene, and then monochlorobenzene and other things to make acetylsalicylic acid (aspirin). Every manufacturing process required an amount of steam, electric power, and water. You solve the first equation, then solve the second equation, then solve the third equation, etc.

It was straight forward, except to make steam, water, and electric power. To make steam, water, and electric power you had to have electricity. Electricity goes into making steam, water and more electric power, so we had 40 simultaneous equations to solve at the beginning. We set up to solve a system of simultaneous equations on punch card equipment. It can be done. It was slow, but worked very nicely. It's just that we had to figure out some way to solve them by continuous approximation. You run an answer You say we have a cost to make electricity, if its raw materials were free. Then you would say, "Now what does it cost[12] to make electricity if you first use electricity, as a raw material,

---

[11] This was before FAX, email, and the internet. It was snail mail, or catch the plane to Texas and on to California.
[12] "Cost" in this case is really the capital cost behind a basic unit of capacity.

to make steam?” That gives you a little better approximation of what it costs to make electricity. You feed the new cost to make steam back into the equation. You repeat this process until the capital of each kilowatt of electrical capacity stabilizes at some number. When you have gone through this iterative process a number of times, you have not only the capital behind the electrical capacity, but also the capital behind steam, and water, and compressed air, etc. For historic consideration, it took us almost two days (eight hour days) of punched card manipulation to reach a set of stable answers.

HAIGH: So it’s like a linear programming problem?

BACHMAN: I don’t think so. In fact, not being a mathematician by training, I was doing a lot of things mathematically by doing what seemed obvious. Then a Dow mathematician, Frank Landee, came along and said, “Oh, that’s called, ‘the method of the stupid statistician’.” Frank was the head of the scientific computing at that time. I had used the method of the stupid statistician. Well, it had worked. My engineering ethic was that if it works, and gets the job done, that’s good enough. That project gave me knowledge on how to use the collators, printers, calculating punches, sorters, key punches found in the tabulation room.

HAIGH: Now, were those punched card machines mostly performing administrative work, or was there a separate punch card installation for this kind of analysis?

BACHMAN: No. I was using the downstairs from our office in Building 47, There were four or five different punch card installations, “IBM rooms” as they would say. Sales Records had one, the Personnel department had one, Production had a different one. There were three or four of them through the plants, and then a separate small one which is being used by the Research Departments. I used the one for the Accounting Department.

HAIGH: And you were able to reserve time and actually operate the machines personally?

BACHMAN: Yes, I operated all of the machines, which means feeding in the cards and carrying them away. However, I had help from the people of the Tabulator Department. They wired the tabulating machine boards that I needed. I quickly learned how to wire the other boards. It was my first experience at wiring boards. That’s how you programmed punch card machines: by wiring boards for them. Each board, when wired, represented a specific program. The small collator boards might be four by six inches. The big boards, something like 20 inches by 30 inches, were for IBM 407 tabulators. These boards were switched every time a new application was brought to one of the machines. This “Capital Allocation” system moved from punched cards to computers after I left Dow in 1960 and continued in operation for years.

HAIGH: Can you describe what the tab room and the people in it were like at this point? You know, the physical surroundings, the training and attitudes of the tabulating people.

BACHMAN: It a nice, big, bright office room, several times bigger than our house, that was full of large grey or black IBM machines. There were lots of “tub” files where the punched cards were stored. Those punched cards files were the database of the 1960s. Typically, with three kinds of We would find those who were the Dow employees, who operated the punched card machines, and we’d find the IBM maintenance people were always around. Typically, the Dow employees were former IBM maintenance people. These people generally had a college degree to be an IBM maintenance person. , intelligent people, who were trained in some business capacity or had a college degree. Most had engineering degrees and did maintenance for IBM and then found an operating job at like IBM did. There were a few operators that Dow had hired in from the outside, but most of them were ex-IBM maintenance people. The third group of people, who you would see in the tab rooms, were the functional specialists who were helping to set up new applications. That was my role in the tab room.

HAIGH: So looking at this list of your jobs, would it be fair to say that from 1950, when you were hired, up to 1955 you were working as a staff analyst looking at these different kinds of cost and pricing issues?

BACHMAN: That’s correct, yes. First there was the Equivalent Capital project and that was followed by the Capital Allocation project.

I had been working in a shared office where there were two other men. Lou Lloyd, was the head of Statistics Department. Dave Baird was head of the Pricing Department. My Capital Allocation project had been completed, for its first year, and it could be run the next year, by the tab room people. I then went to work for Dave Baird in pricing, and spent approximately nine months working on the pricing of plastics and agricultural products. We moved to another office. At that point, after being with Lloyd and Baird for almost three years, I decided I would like to know something about production. The plastic production managers were coming in to talk about pricing their plastic products. I talked to a couple of them, and lined up a job in the plastic department as a process engineer in the Saran department. Saran (polyvinylidene chloride) is one of products made by Dow and is used for Cryovac and other plastic bags for frozen foods. It forms a very good vapor barrier. One of the Saran formulations was used to make the Saran Wrap product.

Most of the time, while on that process engineering job, I was working to develop a process for the manufacture of “foaming-in-place” Styrofoam (polystyrene) beads. The existing manufacturing process for Styrofoam required it being made by extruding very large Styrofoam logs that were then sawn into boards in a sawmill. The new Styrofoam beads were polymerized and then cooled while still under high pressure. Learning how to best dry these small beads was my challenge. When they were subsequently heated inside of a mold, they would expand to fill and take on the shape of the mold, without wasting any of the material.

I had worked in the Saran Division six months, when a job opening came in the Styron 475 plant. Styron was Dow’s name for polystyrene. They needed a new assistant plant superintendent, who was the second person in command. Typically, Dow plants had two professional people as managers. That plant and most Dow plants were really run twenty four hours a day, seven days a week, 365 days a year, by hourly paid shift workers: foremen and operators. I became the assistant plant superintendent of the Styron 475 plant. That was really a different experience from where I started in production. Essentially, I was responsible to the superintendent, Bronson Harris, to make this factory a success. I do not remember how many millions of pounds of high impact polystyrene it manufactured each year. It was called, “high-impact,” because it was tough and could be used to make the inside of refrigerator doors. The reason it was not brittle was because it had rubber mixed into the styrene plastic. Our job was to keep this plant running, making a very high quality product, to very strict color requirements. I had the task of trying to find out if there was problem with impurities or a problem with labor management. Whatever the problem might be, I end up being exposed to it. It was a marvelous experience.

HAIGH: So you enjoyed that kind of operational work.

BACHMAN: I did, very much. In fact, I enjoyed it because I had a work schedule where I had to go in at 7:00 in the morning, while the nightshift was still on and talk to them. The day shift would come in, and I'd work with them all day and I'd stay in for about an hour of the afternoon shift, get them it started. Then I'd go home. Bronson Harris, the plant superintendent, he had other, bigger fish to fry, so I ran the plant most of the time. We'd find ways to reorganize the storage area to realize more storage space. We could be more efficient in that way.

We had fire alarms. The operators in petrochemical plants are taught to run, when the fire alarm goes off. Well, the foreman runs to where the fire is. I ran to where the fire was too, and got there before he did and grabbed the fire extinguisher and put the fire out. He said, "That's stupid to do that. You're supposed to get out of the building." I said, "Well, I thought I was here to make things good." I thought it was a very successful day, putting out the fire.

HAIGH: I know people in that kind of job complain that they spend a lot of their time putting out fires, but in this case it was meant literally.

BACHMAN: This was a fire with flames coming up from the top of the rubber dissolver.

HAIGH: You've written that in 1957 you shifted out of this job to become manager of Central Data Processing. Was that a newly created department?

BACHMAN: At the time that the company decided it was time to get a big computer, whatever a "big computer" was. Carl Gerstacker who was the treasure I had worked for before asked me to come meet with Dr. Mark Putnam, who was the General Manager of the company, and talk about what they wanted to do. They wanted to set up this new department, bring a computer in, and help them do their things. I said, "Well, that sounds very interesting. I'd like to think about it." Carl says, "You start next Monday." That was the end of the discussion.

HAIGH: So do you know why it was that they chose you for this position?

BACHMAN: Not exactly. I estimate they had looked over their existing tabulating personnel and not found the right person. Who else did they know? Then they probably looked at their computing laboratory, which was the engineering scientific group. There was no one there that seemed to fit. So whether I was by default or I was shining star, I don't know. I was picked, and I don't know if anyone else interviewed for the job. If you could say I was interviewed, because Carl had known me very well during the past five years. He knew I'd done a good job at each place along the line.

HAIGH: At this point in 1957, how far advanced were the computer purchase plans? Were you doing a feasibility study? Had an order been placed?

BACHMAN: There was nothing going on. No feasibility, nothing. It was my job to figure out how to start this whole thing up, how to do a feasibility study.

HAIGH: So they had an idea that a big computer would be a good thing to have, but it hadn't been thought through much further than that?

BACHMAN: I don't think we ever did a feasibility study. The idea was that, "everyone else is doing it. We should be doing it. Which one should we buy was the question."

HAIGH: I'm so glad you said that. That was pretty much my impression of how a lot of computers got purchased in that period. It's good to hear it from someone who was there.

BACHMAN: I think the real answer is that you could not economically justify it at that time. Except that by the time that you economically justify them you may be two years behind something else that may have shown up. It wasn't clear what was going to happen with these computers, what they were good for. I was hired, and started work. Now what do I do?

The IBM salesman, Al Janesick, was always hanging around. He came in and started hanging over my shoulder. Kind of like a ghost, because he wanted me, of course, to buy an IBM computer. He suggested I should go to Poughkeepsie, New York and do some IBM visits. That's what they did. They took executives around, or prospects around, to see what's going on in Poughkeepsie and see the computers being manufactured as if that had anything to do with what you do with one after you get it. I made the prescribed trip to Poughkeepsie, and stayed at the Homestead. It was the IBM guest house where they put you up. It was a very nice trip. After that I signed up for a two week programming course at Univac, because the Univac salesman had been around also. I said, "Well, it's time maybe to learn about what programming is all about." I had done a little punch card wiring, but that's not programming in the same sense as programming a computer.

I went off to New York City where the Remington Rand headquarters were. I took a two week course on the Univac 1103A, which was their best, fastest computer at that time. It was very much like the IBM 704 computer. It was designed for engineering calculations. It was not designed for a business data processing. It wasn't like the Univac, which Remington Rand had acquired. At the same time, the IBM had two lines of big computers. They had two business computers. The original IBM 702, and the 705, which was the current one. IBM had the IBM 701 and the 704, as their engineering scientific computers. At that time, they're just beginning to talking about the IBM 709, which was going to do the job for both business and engineering, although it was much more like a 704. It wasn't like a 705 at all, but they put some new features in it they thought would help people do business processing.

HAIGH: My impression is that the bulk of the 709 user base ended up doing scientific computation.

BACHMAN: I don't really know the answer. I do know that one of the things that came out of this was that we ended up ordering an IBM 709 computer. By the spring of 1958, I had built a staff of 30 people. The programmers had been trained on the 709. The people were looking at the applications we might do for Dow at that time, when the company (which had been growing very rapidly) did something they had never done before: they had a layoff. That was May of 1958. The instructions were that every department should cut back to the personnel level that they had at the beginning of that year, which would be January '58. It turns out I had three people at that time. One ex-tab supervisor, Gordon Lossing, I had hired, and one other person as our secretary. She turned around and was converted into a programmer because many secretaries were college graduates in those days. Marian Campbell was our first real programmer. She was a math graduate and a very bright woman. To follow the company staff reduction plan, we would have had cut from 30 people back to three. So a compromise was made, just cut back a third from 30 to 20. We scratched our head. These would be people we had just hired, many from outside the company. Now which is the best or the worst of these 30 good people? That was not an easy task to select which ten, including people you hired from various companies outside of the state. Yet, you do the best you could. You pick these ten people and tell them we were very, very sorry. It turns out the best, most qualified ten people up and quit. They wanted to go to someplace where the computer projects were going forward. I ended up with the middle ten. Marian Campbell was one of those who left. She went to Chrysler, and eventually went on to Honeywell Information Systems.

HAIGH: If you can backtrack slightly here, one of the things I'd be interested in learning a little more about is the process by which you recruited these 30 people. For example, if you were hiring a programmer at that point, what kind of experience or skills would you look for, and how would you try and evaluate applicants?

BACHMAN: I think the first thing to look at this time in 1958, you could not hire programmers. You hired people you were going to train to be programmers. One of the people I hired was Lee Pyron, who was an inside salesman with Dow's Philadelphia sales office. An inside salesman is one who stays in the sales office and handles the sales details. He would always be available by phone, if something came up. Dow also had outside salesman, who would go around, calling on customers. He was the product specialist and understood the application of the various products. He, it was always a "he" in those days. But, in order to make sure that all the things the outside salesman ordered really would happen, the inside salesman was available on the phone to handle the actual order processing and know the ins and outs of how the sales office and the shipping plants worked. We were looking at developing a sales order and reporting system as one of the first applications, so I recruited Lee, who was a Dow employee, to transfer from Philadelphia to Midland and start there.

HAIGH: So in that case, you were hiring someone who already had familiarity with the application area?

BACHMAN: Yes, application knowledge. One of the people, whose name I can't remember now, was hired from Lockheed Marietta in Atlanta, Georgia. He had experience actually programming, in his case an IBM 704, and was interested in coming and joining us. At that time, we had ordered the IBM 709 computer. This was a new computer, and he was interested in that. You picked people really for application knowledge, though we thought they had the inclination to be trained as programmers.

HAIGH: Did you look for anything like mathematical ability or something else that might be a surrogate for the ability to structure things logically?

BACHMAN: I think we talked about those things. I'm not sure how large of a factor that was though.

HAIGH: Now as well as programmers, were you also attempting to recruit operators and analysts, or people to do these jobs.

BACHMAN: Well, an analyst was a programmer in those days. Those jobs had not been specialized. We had no operator problem yet because we had no computer. So we weren't hiring operators.

HAIGH: So at that point you weren't trying to hire separate pools of programmers and analysts?

BACHMAN: Not at all.

HAIGH: You thought of the jobs going together?

BACHMAN: The interesting thing is a parallel that runs through this whole discussion we are having today: that I came from being exposed to business to analyzing how business systems worked to, in some case, having to program punch card systems. Lee Pyron was one I brought in from the sales side, and Gordon Lossing started working with the sales people and the sales tabulating area trying to re-systematize that work. I started working with the engineering people who were planning a new power plant for Dow to try to use the Critical Path Method (CPM), which John Mauchly and his company had developed. This was John Mauchly, of "Eckert and Mauchly," and ENIAC fame. The critical path method provided a way to look at a project and see all of the things that you have to do to accomplish something. For instance, you have to dig a foundation before you pour concrete. You have to pour concrete before you put steel on top of it. You have to put steel on top of it, before you put a roof on top. Well, a lot of things place a sequencing requirement, or a partial ordering, on those steps in the development in the construction process. By looking at each step and its constraints a directed graph (network structure) of job steps and precedence constraints may be built. The "critical path" is the shortest path from project start to completion. The quickest path to complete the job might be in 100 days, or 1,000 days. All other paths had some slack or float; so you could slide their start time around a little bit to level the labor load or the resource utilization.

One of the problems with the standard usage of the Critical Path Method was it required only had one time estimate for each step. I knew that these time estimates were at best approximations. People who had experience in building a power plant might know something about the variability in the time required to do a step in the process. I asked each of the people, making the time estimates, to do the following things. "Give me your best approximations. With a 10% probability, what would be the fastest you could do it? With a 10% probability, would be the longest time that it might take?" With 50 % probability, what is the most likely time? I took these three estimated times, for each job step, and created a probability curve. Using randomly generated numbers, against the probability curve for each job step, I computed 50 different time requirements for that job step. Taking one randomly computed time estimate for each job step, I then ran the Critical Path Method 50 times using each of the 50 computed time estimates for each job step, once, and looked at the results. There were a number of critical paths generated with a range of critical path times. This helped us to place a warning on every job step that was on the critical path for any one of the calculated results, to be sure that it was well managed and did not become a problem in reaching a successful project completion.

HAIGH: So that's why you called it the "Statistical CPM," as opposed to the regular CPM, which has only one estimate.

My impression has been that the general story of the origins of PERT is that it was produced by a Booz Alan Hamilton operations research team, working on the Navy Polaris program.

BACHMAN: That's why I distinguish PERT from CPM. They were created separately, to the best of my knowledge, and they're technically slightly different. I do not know the history well enough to say whether one had an influence over the other, or not. Their diagrams were very different. Both built network diagrams. In CPM, the network nodes represented the job steps, and the arrows connecting them represented a precedence relationships between two job steps. With PERT, the arrows represented the job steps and the nodes represented the precedence relationships. . That is an over simplification. But, they're not the same creature. The reality of job steps and a partial ordering based on the intra-job constraints was common to the business problems that both systems set out to study.

HAIGH: Were you aware of PERT at this time?

BACHMAN: I do not believe that I had heard of it, but I believe that PERT came to my attention after the power plant project Dow was completed. I believe that I learned of PERT after I joined General Electric.

HAIGH: So then your sense was that the Mauchly CPM, which is what you were building on, was at this point actually separate from that (PERT).

BACHMAN: I would think it was totally separate. Now, whether John Mauchly knew something about what someone else did, I don't know. I was not familiar with PERT that time, and I do know I looked at both of them later on. I believe that CPM was the result of work that Mauchly did for DuPont and that Mauchly later turned it into a product.

There was a distinction between them in terms of how they work although you could code the same project both ways. The PERT model required the creation of some artificial job steps so that one job step could be constrained by two or more prior steps.

HAIGH: From these two applications that you've just described, my impression that the Dow order for the IBM 709 was intended to be used on both administrative and more technical applications. Is that correct?

BACHMAN: No, the technical department was going their own course. They went through several smaller computers. One was a Bendix machine with paper tape input and output. The technical applications were nominally going to use the 709, but they weren't paying attention to it. It turns out nothing was really going on with it after 1958, when the big cutback happened.

HAIGH: So the applications that you were developing were all on the administrative side?

BACHMAN: Yes.

HAIGH: Was there anyone involved with the group that had had a lot of data processing experience with punch card machines?

BACHMAN: Yes, Gordon Lossing is one of the first people I hired. He had been the tab room supervisor in the administrative tab room. He brought with him all the live experience from how the tab room operated and business systems. In fact, one of the lessons I remember learning from Gordon Lossing follows. I remember coming back from a meeting and telling Gordon about, “We’re going to build a very sophisticated system.” Gordon looked directly at me and he said, “Do you know what ‘sophisticated’ means?” I stopped and though about it, and said, “No, what does it mean?” Gordon said, “It means ‘unnecessarily complicated. We don’t want to build one of those. We want to build one that is necessarily complicated.” He brought that practical experience to the project. Those were the kind of people we hired.

Subsequently, sometime in maybe ’59, I’m not quite sure which year, early 1960, they hired a new man, Gordon Clack, Dr. Gordon Clack to be the new head of the unified data processing group. This included my Central Data Processing group and the Midland Division tabulating organization. He was head of the Technical Recruiting Department, when I came to work for the company five years earlier. In fact, he’s the first person that I met at Dow. Well, that’s not true. First person I met was his secretary, Marian Campbell, who was the second person I hired. Gordon Clack was a PhD in chemical engineering. He could talk peer to peer, to all the PhD candidates that were many of the technical hires coming into Midland. He came over to head the group combining the Midland tabulating department and my group.

A bit earlier, in1959, I helped start the 9PAC project, which is something we should talk about. Because in the sequence of things, after we had ordered the 709, I later did another thing that Al Janesick, the IBM salesman, thought Dow had to do. Dow Chemical had to become a member of SHARE. SHARE was an organization that was populated by people who had IBM 701s and 704s, the scientific computing community. Because the new IBM 709 was supposed to be compatible with the 704, the engineering scientific group claimed it as their computer. The IBM 702 and IBM 705 people had an organization called GUIDE. It’s much bigger and populated with a very different bunch of people. They tend to be supervisors of tab rooms and accounting people. In my opinion, it was a drinking group. The SHARE group wasn’t a drinking group. At least it was less of a drinking group. That may be unfair to many people in GUIDE, but that’s the impression I’ve gotten. The most important things were the parties, and secondary the technical issues.

HAIGH: Yes, so we’ll definitely spend some time talking about SHARE. I have a couple more questions on this period. You were managing the Central Data Processing group?

BACHMAN: Yes, this was in the same period. It was in the Dow period.

HAIGH: Yes. I have a couple more questions about what has happening internally in Dow, and then we’ll move onto SHARE.

BACHMAN: Very good.

HAIGH: Did Dow at this point have any kind of systems group?

BACHMAN: They had a systems organization. In fact, one note I wrote, when I looked at some of your papers, is I had not heard the word “systems man” since I left Dow Chemical. They didn’t exist by that name in General Electric, or maybe they had gone into hiding, or something, or changed their coat of armor. But, there was a distinct systems group at Dow. One of the people I traveled around the country with looking at different computer sites was a man from systems, though I can't remember his name, at this moment. The systems organizations seemed to have disappeared. There are people who are designing systems, but they wear different hats these days. To help put things into context. In 1956 we made our trips by train, rather than airplane. We drove 100 miles from Midland to Detroit and then took the train to New York. It was an overnight sleeper trip.

HAIGH: I believe in a lot of cases systems groups were merged into data processing departments by the 1960s.

BACHMAN: They may have just done that at Dow, after I left in 1960. I don't know what happened.

HAIGH: That's interesting. It sounds to me that there would have some overlap between the kinds of things that you were doing and what the systems group would be interested in.

BACHMAN: That's right, although they spent more time designing forms than designing systems… although again that may be an unfair statement. They seemed to be forms designers, and people who kept track of the forms they had and managed the forms and reordered them and things like that.

HAIGH: So your impression of the group is that it was quite mundane?

BACHMAN: That's a first approximation, but there were some interesting people in it… although I didn't hire any of the people in the systems organization for the data processing group.

HAIGH: So that group was not involved in doing the systems analysis for the new computer system?

BACHMAN: Not at all. The old Midland Division tab room group acquired an IBM 650 about 1959. In fact, when I did the project using the statistical CPM program from Mauchly and Associates, I used their IBM 650 to generate the randomized estimates of job step times and then ran critical path analysis. I could then determine the

When I was talking to somebody else the other day, it struck me that when I started programming the Integrated Data Store (1961), that IDS was the first thing I had ever programmed in my life. That was an unusual job qualification for undertaking a major software development project. Actually, I had written one program, the probability part of the statistical CPM project. We bought the actual CPM program from the Mauchly organization. I programmed the part that took the three time estimates for each job step: the earliest, the likeliest, and the latest estimated completion times, with 10% probability, 50% probability, 90% probability, respectively, and mapped them onto a Poisson distribution. Then I used a random number generator to generate the 50 different time estimates for each step, based on the Poisson distribution I had computed for that job step. The very small program, which mapped the three time estimates onto the Poisson distribution and then randomly generation 50 calculated time estimates for each job step, was the only program I had written, before we began the IDS project.

HAIGH: When you first began programming, was that something that you found enjoyable?

BACHMAN: I enjoyed it, yes. In reality, I've done far less programming in my life than writing functional specifications. I have written far more functional specifications, for someone else to program. I had written a part of the 9PAC specifications (1958-9). I had written the IDS specifications, before beginning to write the actual IDS program generators and run time routines. Well, let's go to 9PAC project. But, finish where you are here before we get to that subject.

HAIGH: I think that covers these internal Dow developments and your relationships with the systems group and the way in which you were putting together the central data processing group. As you hired these people and they began work on applications, at that point with the expectation that the 709 would be arriving, were they actually writing code for the applications?

BACHMAN: They were just going through training. Some of them had been there only shortly before we turned around and said, "Sorry." They had hardly gotten settled in the new homes they had bought. It was very embarrassing.

HAIGH: Had you said that the IBM 709 order was then cancelled?

BACHMAN: Not immediately. But it was eventually cancelled. A computer room had actually been built for it in the new Dow corporate headquarters building.

HAIGH: Was this the administrative headquarters building for Dow Chemical as a whole?

BACHMAN: Dow Chemical as a whole, yes, and located on a new campus several miles away from the Midland Division plant site.

HAIGH: Was the computer room one of these prestige installations that would have had plate glass windows and those kinds of things?

BACHMAN: It had a raised floor and some windows along the side. It wasn't really designed to be a showplace. It was designed to satisfy all the environmental requirements. Make sure that enough power to it, enough air conditioning to it. It had a raised floor, so we could run the computer cables under it. They were big, one and a half inches in cables.

HAIGH: Did Dow, while you were there, ever get a big computer?

BACHMAN: Not while I was there. The 650 was the biggest computer we had when I left. IBM 650.

HAIGH: But as I recall it, the requirement for SHARE was that you had to have ordered a computer so that people would get involved with it as they were ramping up for the computer delivery.

BACHMAN: Yes.

HAIGH: Okay, so I think we are ready to move onto SHARE and 9PAC then. In the interview with Phil Frana I think you told the story of how you first became involved in SHARE. To pick up on a comment you made a minute ago, it's interesting that you say that SHARE wasn't the drinking society. One of the things I've seen in the SHARE records is their group, SCIDS.

BACHMAN: Yes, there was SCIDS! I do not remember what the acronym stood for.

HAIGH: Which does seem like a drinking society?

BACHMAN: I guess the real question is where is one's first priority? I think the SHARE member's first priority was to technical problems. Drinking was secondary. That was just the other way around with GUIDE.

HAIGH: Or, maybe they'd get drunk while they talked about the technical problems.

BACHMAN: They probably did that too, but the SCIDS suite didn't open until dinner time, with snacks and a cash bar.

HAIGH: So you think at GUIDE they weren't even sober during the daytime?

BACHMAN: [Laughter] I don't know that much about GUIDE. It's just an impression that they were a different group of people. Maybe they were business administration graduates and not engineering graduates, if that shows a bias in my thinking.

HAIGH: So as manager of a 709 installation to be, you were attending the meetings, and presumably you were also receiving the SHARE secretarial distribution?

BACHMAN: Yes, very important. They were the most important set of documents that was published in that era. Far more technical information was recorded in the SHARE Secretary Distributions than you could find in the technical journals. It was not refereed. It was the 1950-1960 version of the Internet.

HAIGH: Looking at your papers at The Charles Babbage Institute, one of the things I saw was a report of a SHARE Committee on Information Handling. Do you recall what that was and if it ever went anywhere?

BACHMAN: Yes. Did that ever go anywhere in the end? It may have morphed into some but there were some interesting people involved. Tom Steel is the name I remember the most. Thomas B. Steel, Jr., who came out of RAND Corporation. A very bright guy. A very political one too, who ended up moving to Equitable Life in New York City and worked for them… got paid by them. I'm not sure who he worked for. For years he attended meetings all over the world, but this is an area where we tried to work on-- What is it? Is there a theory of information handling that we could build on. There was a man whose name I can't remember now who was the professor at Colombia University. He wasn't very active. Was it Jon Turner? I think not, Jon Turner came into my vision later, in the early 1970s.

HAIGH: Oh the draft report that they produced, somebody… probably you had written at the top "written about 1958 by Bill Orchard-Hayes".

BACHMAN: Okay, Bill Orchard-Hayes was there, and Bill Orchard-Hayes was out of Martin Marietta, it was Martin-Baltimore.

HAIGH: I think he made a name for himself in mathematical software.

BACHMAN: Yes, good man, yes.

HAIGH: From notes by Orchard Hayes, Charlie Bachman, and Wheaton Smith.

BACHMAN: Wheaton Smith. I haven't heard that name in a long time. We talked about fundamentals, and talked about a system based on triples. We believed that the most elementary particle of information had three properties. Essentially they were 1) a data value, 2) a name that characterized what that data value meant, and 3) and an identification of who/what it belonged to. So when we had a number, like 1950, that was a date. More specifically, a date of hire. The third part was, " What entity does this value help to characterize?" An identifier of some kind tied together a number of these triples…. You could merge triples, based on the fact that they all belonged to the same entity. This triple was the basic piece of information you could deal with. Whether it's a punch card system-- the punch card represented a mechanism that held all these triples together. All these columns are on the same card, therefore it must be about the same customer, or the same order, or the same employee. Then it had groups of columns, and you knew what the name associated with a column or group of columns. Then the holes punched into the column or columns represented the value. So you can see the triples in a punch card system. Or you see them in a magnetic tape system, thea disk file system. We used to have a "record" as a concept. You'd probably call it an object today, but it's the same thing. It had columns, or we might call them properties that have values. I think the notion of the triple came out of that. I first understood it in that context in that group.

HAIGH: So you're trying to really abstract away from a specific kind of representation of a file, and find the atomic unit of data.

BACHMAN: When people talk about inverted files versus non-inverted files, the triple exists in both of them. It just depends on how they're organized. An inverted file, you take the triples and you sort them in terms of what the data field name is and then the value, and then finally the third part of the triple tells you what entity it belongs to? Where in a normal file, the entity it belongs to is evident, as all these fields are together in some fashion.

HAIGH: I understand that your main involvement with SHARE in this period was with the SHARE data processing committee.

BACHMAN: That's correct.

HAIGH: Can you describe that committee?

BACHMAN: Well, the committee was formed at the second SHARE meeting that I attended. The first one was in New York City. The second one was in San Diego. At that second meeting I met a man named Harry Tellier, who was very important in this whole history. He was a data processing manager at the Hanford Atomic Products Operation, which is the US plutonium products plant in the state of Washington. It was managed by General Electric. Harry was a General Electric employee. He was a former tab room supervisor; I believe for the State of Washington. When I think of the paper you sent, the "Chrome-Plated Tabulator[13]," Harry's work epitomized the chrome-plated tab machine. His group created the Hanford[14] 702 File Maintenance and Report Generator System, which was an exquisite piece of work. I will call it the "Hanford 702 Report Generator"

[13] HAIGH: "The Chromium-Plated Tabulator: Institutionalizing an Electronic Revolution, 1954-1958", IEEE Annuals of the History of Computing, October-December 2001

[14] It was the IBM 702, but the Hanford File Maintenance and Report Generator System.

for short. In some sense, it was a chrome-plated tabulator, but it had some extra properties that were very important. Essentially, they had built a magnetic tape-oriented master file system. Each time you updated a tape file you rewrote the master file from one entire set of tapes to an other set, and updating some of the records on the fly. Other records came across without being modified. Periodically—once a week, once a month, in a few cases once a day—they would repeat the entire updating process. A magnetic tape file might be stored on a single reel of tape, or it could occupy many reels. Every time they processed the tape file, they could also create reports from it. Actually, they had a new version of the tape file and had an opportunity to produce updated reports.

One of the big problems in the early data stages was “How do I get reports?” The most interesting part for most people for the Hanford 702 File Maintenance and Report Generator System was that I could easily program a report. The Report Generator was a “What You See Is What You Get,” a WYSIWYG system. I actually filled out a piece of paper form, and put column by column on that form what I wanted printed on my report. Depending on whether this is a “detail line” saying this is the lowest level of the information in this file, or a “header line” saying this is about the customer and the detail about what I sold him. Then the “total line” is printed down below. Each line of these forms described how to keypunch one or two cards. Each column in a card represented a column in the final report. The reports could be 120 (144) columns wide so it required two cards to describe the information for one printed line. You take that “packet” of cards, as it was characterized, put them together, and say this runs against the sales history file, or this runs against the inventory file, or this runs against the personnel file. There might have five, ten, or 20 different packets that would used each time that the particular master file was run. Some packets would be run every time the master file was updated. Some, which were brand new, might run once or twice. You could put a packet together in a day or two, and get a real report out in a hurry. It was the reporting system, there was nothing else; it was the first query system. It’s critical in a sense that one could get fast response. Thinking in terms of today’s “OLAP” cubes, the records identified by the detail lines of the report were the fact tables, and the header lines represented the dimension tables.

HAIGH: Now, do I understand from what you just said that it was capable of making several different reports in one run through the master file?

BACHMAN: Absolutely. Each packet described an individual report. Essentially, you took one or many packets of cards, and you stacked them together in the card reader. However many, ten, or twenty packets, you’d run them all at the same time. They would run the master file through once. They would get 20 reports out of it. The report generator run would sort them in whatever sequence you specified. Tabulate them, total them, head[15] them! It did a superb job of generating reports.

HAIGH: So that’s report generation. Did it also provide facilities to help application programmers read and write records?

BACHMAN: Well, the file maintenance part was similar. In some sense, it was a programmer-less system. They had programmers that actually programmed the Hanford

[15] To “head” meant to format the page heading information, which might change from page to page to remind you where you were in the data hierarchy.

702 File Maintenance and Report Generator System, but their goal was to do without programmers as much as possible. Now the weakness of this system was that it didn't do calculations very well at all. For a payroll system, you had to compute wages. There was a mechanism for inserting what were called, "handcalcs." They were simple assembly language subroutines that were invoked, when the right situation would arise.

HAIGH: So then to update records in the file you would essentially feed in a deck with a list of things: on this record update date this field; on that record update that field. It would do the same thing, run through and batch mode, and make the updates to the master file.

BACHMAN: Yes. Interestingly, they talked about doing "horizontal" and "vertical" changes. Think of records appearing vertically as a magnetic tape ran from top to bottom. You have to think of a magnetic tape running vertically, from top to bottom. A vertical change could change all the pipe fitters' wages from $1.14 an hour to $1.17 an hour. A vertical change would affect every detail record that met a certain selection criteria. A horizontal change card, in the file maintenance packet, would select a single record, based on its primary key, and make the specified change. If I want to change one pipe fitter's wages or one pipe fitter's home address or something similar, I could change that person's record with a horizontal change[16].

The file maintenance process was designed to easily do update-in-place type things. Replace the old value with a new value, whether it was for one record or a whole class of records. Now, if you wanted to compute someone's payroll, based on how many hours they worked in a week, and the number of deductions, it was a beast. It was a beast because you had to insert, what was called "handcalcs," into the file maintenance packets. These were written in assembly language. The file maintenance generator would assemble these own code routines and insert them into the file maintenance run. They could run as part of either a vertical change or a horizontal change.

HAIGH: So that was something almost like a stored procedure?

BACHMAN: Yes, I guess you could say that. Yes!

HAIGH: That it would be code the system would then jump to, whenever it was updating a particular kind of record. Then this code would run, it would do the calculations, and it would terminate and pass a value back.

BACHMAN: Yes.

HAIGH: That's an interesting approach. So there was no way, really, for this system to be called by an application program? It's the other way around. This system is calling little chunks of customized code. It's not called by it.

BACHMAN: In fact this Hanford Report Generator System was a superb system, except for its file maintenance side that I call almost a fatal flaw. It was a superb replacement for the IBM 407 tabulator and its sorters, collators, and punched card calculators. It was very cost effective. In fact, in the history of things, this Hanford Report Generator system was superceded by the project that I was involved with, the IBM 709 version of the Hanford

[16] Russ McGee tells me that my memory is wrong and that this terminology did not come along until 9PAC days when he introduced the concept.

system, called 9PAC, short for the IBM 709 Data Processing Package. 9PAC ended up becoming 90PAC when the IBM 7090 came out. It was so popular with IBM's customers that, IBM wrote emulators for the 360 to emulate the IBM 7090 so that the 9PAC-based applications could run on the 360. The Hanford system did some things very well. In fact, there are generations of RPGs that were derived from this. This is really the grandfather of all the RPGs, the Hanford 702 version. All the others spun out of that. Whether it was the IBM RPG for something or somebody else's RPG, its great grand parent was the Hanford 702 Report Generator system[17].

On the file maintenance, weakness side, high level programming languages filled in the missing piece. The committee that created COBOL did not even consider the report generator approach because of this weakness. Our approach in 2000 still shows the difficulty in reaching a satisfactory compromise between the handling of transaction-oriented approach to operational systems and the report and query orientation of data warehouse systems.

HAIGH: Just to clarify the GE system. This was a system that worked with records on tape? It wouldn't support drums or--

BACHMAN: No! Nothing. Just from tape to tape.

HAIGH: Did it have a Data Definition Language equivalent where you would define what all the fields were in the file?

BACHMAN: Yes it did. That was a record definition part. There were data definition forms that had to be filled out and keypunched. These cards were read, by a pre-processor, whenever a new master file was being created. The resulting data definitions were written onto the front end of the first reel of the new master file. These data definitions were placed at the front of the first reel of the magnetic tape file, so that the report generation system always knew where to find each field of each record of the master file. This was the first case I know of where "meta data" was being so explicitly recognized.

Whenever you wanted to run a batch of reports against a master file, the first thing to was to load the report generation system itself into the computer from a magnetic tape. Then you would mount the first reel of the master file on to the designated tape drive. The report generator would read the data definitions in from the beginning of the first tape. Then you'd put your report specifying packets into the card reader. The report generator would read the report packets in. Then it would generate an entire report generator program, based on the data definitions and the report packets. That program was generated and used only once. Even if you ran the same set of card packets the next day or next week, you generated the program again. The data descriptions were kept in the front of the magnetic tape, which could be changed from one run to the next, so the master file was a variable, self-describing file, a very important concept.

HAIGH: Was it a purely flat file system?

[17] In 2006, Progress Software's RPG is the primary descendant. It is very successful, earning hundreds of millions of dollars a year.

BACHMAN: I think the answer is yes. There was only one record type defined by the data definitions place on the front end of the master files. Only the authors of the report packets knew that there were important relationships to be recognized and exploited in the report packets. Given the situation where I logically had an employee record followed by a bunch of weekly pay records for each employee, there would be two different logical record types. Physically, these two logical record types would be flattened into a single physical record type. All of an employee's specific data, i.e., name, pay number, social security number, would be replicated in each of his weekly pay records[18]. The data description knew the names of the data fields in the single record type and their locations. The report generator system was instructed, by the report packets, to detect the change in the employee number values, in the sequence of records. Then it would treat the first weekly pay record for the next employee as a control break. The change in a control field value controlled the change of content in header lines and the firing of total lines and the resetting their counters. Header lines are usually matched up with the "total" lines as opening and closing parenthesis, respectively on a batch of details. Several levels of control fields were possible, supporting the maintenance of hierarchical structures in a flat file and the construction of hierarchical reports from them.

Every record on the master file was read only once, then the generated report program would process it in the context of the first report packet, then the context of the second report packet, etc. This would continue until it had been presented to a section of the report program that had been created to handle each report packet. If you build a header line to put into a report, the report generator would transferred the information from the weekly pay record into the header line before you transferred to the next report code to look at that record's impact. Report lines were being generated at that time and queued onto a tape for latter separation and sorting. Then the report program would read the next physical record on the tape.

It's a flat file system because there was a single physical record data definitions and it contained no indication of record hierarchies. However, the people who designed the files and wrote the report packets knew better. You have to understand the distinction here between logical files and physical files, One of the real differences we will find when we come to the 9PAC report generation system is that multiple record types could be defined in the master files and the data definitions clearly defined all of the parent/child hierarchies that existed in the sequential files.

HAIGH: As I understand, that's equivalent to say early microcomputer systems, for example the dBase family in the '80s. That you defined one file at a time, but in an application you can perform operations that bring together different records. It's not that there's a data definition language that it's explicitly defining and enforcing any kinds of relationships.

BACHMAN: No, let me speak a little heresy for you. A relational database system is a flat file system because the relational database doesn't know anything about

[18] This would have been impossible using punched cards, due to the limitation of the 80 column IBM cards. So multiple card type master file systems were common in the punched card era.[19] Mock's first name was Owen. He worked for North American Aviation, though the MockDonald name was also a play on McDonnell Douglas, another SHARE member organization.

relationships. It only knows about tables. Well, when you execute a “JOIN” statement in your program, you combine tables in a more structured sense because the database system doesn’t know about the relationships between tables. Certainly not in the first 20 years of it. But, the programmer obviously did. He’s got to do joins and--

HAIGH: As I understand it that’s a key feature of the relational model, that the joins are specified by the programmer or user at the time that performing operations instead of being in the data definition language. But, you can, at least in Oracle 7 and later, define the constraints, primary keys, foreign keys, so that it enforces integrity.

BACHMAN: But these were add-ons to the system. Necessary add-ons. I have no quarrel with them being there because they were obviously missing. In the Hanford 702 report generator case, the people, who wrote the report packets, were adding the extra information. They understood the structure of the information. Files were sorted, merging the different kinds of records in a meaningful way.

HAIGH: That merge would be done with several tape drives with the files sorted in the appropriate way. Essentially the same way it would be done with the merge in a punch card system.

BACHMAN: Yes.

HAIGH: Okay, that seems like a good point to break up.

BACHMAN: Let’s have a little lunch then.

HAIGH: This is the beginning of session two, where we left off, I think you had been talking about the work of the data processing committee of SHARE.

BACHMAN: Yes. This committee was formed at the meeting in San Diego, the second one I attended, and the first one that Harry Tellier attended. He was the data processing manager at GE Hanford, where they developed the Hanford 702 File Maintenance and Report Generator System. He and I immediately started talking about how do we get the Hanford system redeveloped for the IBM 709 computer. So, we formed the SHARE Data Processing Committee at that time, with several other people. Fletcher Jones was one of the members. I believe that he was with North American Aviation at that time. He was later to start a big consulting information firm in LA. I can’t remember the name of it right now. It turned out that Fletcher Jones was interested in the IBM 704, so another SHARE group did the “SURGE” system, which was to be a 704 version of the Hanford 702 system.

The reality is that SURGE turned out to be some sort of enhanced assembly language that had no relationship with the Hanford 702 system.

HAIGH: So SURGE was a reinvention for the 704 of the--

BACHMAN: Yes, that was initial idea, but that is not what happened.

From that San Diego SHARE meeting (December 1958) we started collecting people, who were SHARE members, who were interested in cooperating on an IBM 709 report generation system. Many companies quickly joined into the effort. Companies like Union Carbide in New York City, Phillips Petroleum in Bartlesville, Oklahoma, and Northern States Power, in Minneapolis, Minnesota. There were a couple of General Electric departments that joined 9PAC. They were also interested in using the 709 for business applications. For your information, a "department" in GE represented a complete, full function, product business. There were more. Those names are lost to me. I think that Light Military Electronics was one of them.

HAIGH: "It" at this point is the 709 system?

BACHMAN: The report generator design for the IBM 709.

HAIGH: So by the time you became involved, was the SURGE for the 704 already finished?

BACHMAN: No, no, Harry and I started the whole thing and launched the 709 version before a 704 version of the Hanford report generation system was even thought of. The 709 group got underway between the San Diego meeting in the fall and the Washington DC SHARE meeting early the next year.

HAIGH: So they ran in parallel.

BACHMAN: They were parallel projects, running completely under their own direction.

HAIGH: Were they both being conducted under the auspices of the same committee?

BACHMAN: Yes and no. Once the projects got going, the committee was rather unimportant because the projects took on a life of their own now. They had their own project managers. Russ McGee, is the man who from Hanford who took on the job of being the project manager of the 709 project. There were people from Hanford who worked on it. They included Ed Roddy, Joan Cannon, Ron Pulfer, and George Gillette. They all moved latter to the GE Computer Department in Phoenix. Marion Campbell, from Dow Chemical, worked on the project. Art McCabe was one of the people from Union Carbide who worked on it. He subsequently came to the GE Computer when Russ and I were there. That project went on, and they worked on it for maybe a year and a half. In fact, it'd be useful for you to get a copy of Russ McGee's memo; it contains a lot of detailed information, and we've chatted about that to try to see if I can help him fill in some parts he couldn't think of and vice versa. In the end, the 9PAC report generator portion was written by the SHARE group. The file maintenance generator portion was written by the people at the GE Hanford Atomic Products Operation.

That project came along and worked out quite successfully in serving the needs of the people who build it. It also found a much wider audience among IBM 709 users. The 709, which is a vacuum tube computer was replaced by the IBM 7090, a semiconductor version of the IBM 709 computer. Its usage spread much further. It was popular enough that IBM built 709/7090 emulators for the 360, when they brought out their 360 line. This was to assist in the conversion of 709/7090 applications to run on the 360. They were not converted; they were emulated. The emulator was a special hardware box for the 360 that could run the old 9PAC code the way it had been run before, so those customers could easily move their 9PAC applications from their 709s and 7090s to the 360. That helped to keep the 9PAC running for a long time, because there was so much invested in those systems that they couldn't give them up easily. There was nothing available from IBM except the COBOL language, which is not an easy substitute. It's something very different. With the 9PAC emulator, the existing master files and their packets could be run immediately. With COBOL, you had to learn how to program and then redevelop new application programs, based on an entirely new paradigm.

HAIGH: Let's talk some more about the 9PAC system then. Were its capabilities in any way substantially different from those of the GE Hanford system for the 702?

BACHMAN: Yes. They were in several ways, and I can only tell you a few of them. I think the most significant one is that for the first time the system took formal notice of relationships between owner records and detail records. Before, if you were on a 702, you had a customer record with a customer code in it, as a primary key. An invoice record would have an invoice number as its primary key, but would also have a customer code in it, as a foreign key. By the time we got to the 709 the invoice record didn't have the redundant customer code because its relationship was known from the physical context of its customer record, therefore it had to belong to the customer that just been processed. It was still a tape system. It maintained these relationships by juxtaposition. This made things a bit interesting.

HAIGH: So it still only supported tapes?

BACHMAN: Still on tapes. It is still on tapes with the data definition still on the front end of the first tape of the master file. In many ways it had the same kind of WYSIWYG forms that facilitated report definition. They were designed for keypunching. This is the era of punched card, and also the era when there were no operating systems. Each time you started a new job on the 709 or 704 you would be starting with a machine that was not running. You load the card deck into the card reader and pressed the read card button. The first card told the computer what to do with the cards that followed. Essentially, it was a one-card loader. The cards that followed really formed the basis of the actual program to be executed. When the execution of that program was complete, it told the computer to stop. In the case of the Hanford 702 system and the 9PAC system, the program that first ran was the report generator program. When the generation was complete, then the generated program was read in and control was transferred to it.

Those old computers had elaborate operator consoles. Most importantly, they had a small green light and a small red light. When the green light was lit, the computer was running. When the red light was lit, the computer had stopped. Harry Tellier, at GE Hanford, was famous for refusing to pay IBM for his computers on a "per shift" basis. He would only pay for "green light" time. He would only pay for the time when the computers were working for General Electric. The difference between paying for three shifts a day and paying for green light time in a day was significant. The computer was manned 24/7.

HAIGH: Now I know that SHARE was also involved in the production of an operating system for the 709. Did your team have any interaction with that project?

BACHMAN: Not that I know of. At the time, there were a couple of operating system projects. The first one was called "MockDouglas," or something like that.

HAIGH: MockDonald, it's a play on words.

BACHMAN: MockDonald, named after a man named, Mock, who was the project leader. "Douglas" was the name of the company that was leading the project. It was again a way to bridge scientific computations on a 704 from one job to the next without having to have human intervention, so you could build up a string of jobs on a offline card reader. Then feed the tape which would control the flow of jobs into the computer. I'll think of Mock's first name.[19] It was something that was done within SHARE. The data processing committee was a separate part of SHARE, and they were to the best of my knowledge totally independent.

HAIGH: Now I've just got a few documents here on my computer that I took from the SHARE Secretarial Distribution system. I've got one here, which I think from about 1959 and called "Preliminary Manual for the Generalized File Maintenance System". Do you think this would be about the 9PAC project or this would be the SURGE project?

BACHMAN: There's no way to resolve that ambiguity here. Do we even know who wrote this thing?

HAIGH: No. It's part of SSD 46. There's a letter that I think was in the same one, from Russ McGee.

BACHMAN: Okay, if Russ McGee wrote it, then this is 9PAC.

HAIGH: They probably go together. "As you probably noticed, I published a progress report on the report generator, which qualifies this tape to my early telegram. As a matter of fact, I feel the report generator..." So it appears that while the project was underway, they were calling it the report generator.

BACHMAN: IBM gave it the name 9PAC[20].

HAIGH: This letter is January 21st, 1959.

BACHMAN: Yes, it is from Washington.

HAIGH: So you think the name 9PAC had been attached to it after this?

BACHMAN: It may have been attached to it casually. At some point, it got printed on top of manuals. But 1959 was too early to talk about manuals.

HAIGH: So that's good then. I was wondering if this was the same project, and now…

BACHMAN: Yes, with Russ McGee, it's clear they're the same project.

HAIGH: So note to future historians: If you're interested in this topic, then you may see 9PAC referred to in the SHARE materials around1958 and '59 as the "report generator". There's a series of references to it. There was a letter in a slightly earlier one asking, "Is it ready yet? Can we have it?" These were saying, "Well, we've got to print a preliminary manual here," and so on.

BACHMAN: That ties it back together. Historians should look for Russ McGee's unpublished book, "My Adventures with Dwarfs." My computer printed copy is 222 pages in length, typed single spaced.

HAIGH: What was your personal role on the project?

BACHMAN: My personal role really consisted in the functional specification writing part. One of the things that I know that Russ and I had long discussions on was this notion of making 9PAC conscious of the owner/member relationship in the hierarchy,

[20] 12/12/2005. Russ McGee just told me that IBM did not name 9PAC because they wanted nothing to do with us! We named 9PAC because beer companies wee jus coming out with "6-packs."

where before it was up to the packet designer to know what he was doing, and the persons who designed the file. I talked to Russ a year ago (2003) when he was here. In fact, Russ is in the process of moving to Tucson, so we'll see more of each other. We were very good friends. I hadn't realized until I talked to him recently that he had resisted putting in this notion of the declared hierarchy. I was insistent because it was necessary to handle the data structures as we found in the sales records and the inventory records at Dow Chemical, where there were multiple level concurrent hierarchies. You might have a customer who has an order who has a list of items in the order. That's a three level hierarchy right there. Maybe sub-items just for distinct shipments, so that it may be an order for ten tank cars, and they're going out four or five different times. Or each tank car may have been given a separate bill of lading, which it probably did. There's multiple hierarchies there. Really, the question of how we got from there to IDS network systems is the fact that whatever you found the bottom record in a hierarchy, it typically belonged to some other hierarchy. Both 9PAC and the punch card systems required you to sort the records/cards to support one hierarchy, or the other. They could not physically support two hierarchies at the same time. At one time, files are sorted into the material item sequence for inventory reports. Then they are sorted back the other way, for sales reports. If you have an online system, you don't want to store things twice and you wanted the records in both sequences, concurrently. So you implement in a way where you don't store things twice, you always wanted them concurrently in both sequences. It's got to do a network structure which allows you to connect records into as many places as appropriate to the subject.

HAIGH: And that capability was in 9PAC?

BACHMAN: That was not in 9PAC. That was IDS.

HAIGH: Okay, 9PAC was purely hierarchical?

BACHMAN: That's all you could do with magnetic tape. So once we moved from tape to disk file, we have direct access, not sequential access. The program can access anything if it knows where to go look for it. So we could store a record in more than one hierarchy. They're different implementations, because the hierarchy was constructed in 9PAC by juxtaposition. The records, which follow record "A" are trailers of record A, and you can tell by position that they belong to record A. IMS used a physical position implemented hierarchy, even though it was stored on disk. The Index Sequential system was a physical hierarchy too.

HAIGH: Okay, so the records would be sequenced in order of the top level records, and then all the records that come further down the hierarchy that go with that record would just be sorted into place, essentially as extra fields in the parent record.

BACHMAN: Right. Well, essentially if I have a highest level, let's say, is a customer record. The customer code is the primary sort key. Then a sales order record would come with a customer code and have a sales order number, and all of the data of a sales order. In fact, they duplicated the customer code too because in punched card systems or magnetic tape systems you duplicated all the foreign keys. When we got to 9PAC, we got rid of the redundant foreign keys for the hierarchy that was implemented by juxtaposition. We didn't need to store foreign keys anymore because we could resolve the question of what the foreign key pointed to: it pointed to the master record that just processed, in the sequence, but there were still foreign keys existing that pointed off, let's say, to the product inventory record hierarchy.

HAIGH: Would it also be the case that because a record would be physically stored together with the record that was above it in the hierarchy that you couldn't have one record appear at two different places in a hierarchy?

BACHMAN: That's right. It was a very strong discipline because you interpreted according to its position, and it could only be in one position. Therefore, only one interpretation is possible.

HAIGH: So you couldn't just have a pointer going off somewhere else?

BACHMAN: In a sense, on the magnetic tape systems, they had foreign keys, which are a form of pointer. I'm pointing to the material item code, so I can find it in the inventory record hierarchy. So there were pointers. It is very much like OLAP where people are building cubes. A fact table may have two, three or more identifiers. Each identifier is associated with one dimension table and the value of the identifier is a foreign key which matches a primary key of a row of that dimension table.

HAIGH: Let me repeat that for you. If I look at an invoice, which has a customer code on it, the customer code is a foreign key always. If I look at the sales item, item number one on the invoice, it has a sales order number is a foreign key. It has the customer code, which is a foreign key. It has the inventory item key, so they're all foreign keys you sort on, except for the top part of the hierarchy. They're following something. They're down the hierarchy from something else.

BACHMAN: I misspoke. There are foreign keys in every case. They don't change from a foreign to a primary key, because it's the customer record that always has a customer code, as its primary key. You don't sort the customers…. Well, actually you do, but the customer code will stay on the bottom level item. It gets sorted another way under inventory. It will still show up as a foreign key, but the customer record is only in one place.

HAIGH: Leading from that and those terms you're using, it's my impression that the concepts of "file" and "field" and "key" were all there in punch card systems before the computer came along.

BACHMAN: Absolutely because they're based on rather fundamental information. In fact, that information triplet I talked to you about earlier. There's the entity, which we're keeping track of, there's the data field, which is a called a column on a card, and then there's the data value. Those three things keep appearing in every one of the systems we look at, even through all the database management systems we'll look at, until we come to one where the triplets are quadruples[21], but that's another step we'll come to.

HAIGH: After you wrote the functional specifications for what became the 9PAC system, who took those specifications and turned them into code?

[21] Time is the new information. It defines the time period for which the information is/was valid. In 1958, we were concentrating on the "current" data.

BACHMAN: Well, Russ and I wrote the specifications together. Russ and one of the women from our group, Marion Campbell I mentioned, and other people from the group from Hanford who worked for Russ McGee or worked for Union Carbide or Southern States Power, or Phillip Petroleum or some of the other…. All these companies furnished typically one programmer or maybe two programmers. Ed Roddy for instance was one of the extra ones from Hanford. Art McCabe was working for Union Carbide. He later joined the GE Computer Department in Phoenix. He died in this past year. His death came as a real surprise, because Russ McGee had a meeting scheduled with him and was driving down to meet Art the next day, when Art died (2004).

HAIGH: So the code was written by….

BACHMAN: All members of the SHARE member employees. To my knowledge, there were no IBM employees[22] on the project, although I'm sure there were people keeping their tabs on what was going on. Onlookers, auditors, or whatever you might call them. Typically marketing people.

HAIGH: Would these programmers on the different sites ever meet each other, or would they just be given narrowly defined piece of the specification and told to make the code for it?

BACHMAN: They would travel to different joint sites. In fact, the one I remember, I went to… Union Carbide had a old factory building that had been converted to a big office building in Queens, New York, and I was on an Eastern trip and I stopped by there. Marion Campbell was there, and about 20 or 30 people working on the project. They were all there, so they could exchange notes because it's a hard thing to break up into pieces, when it's one integrated system. The report generator part could be somewhat separate because it's dealing with a common set of data descriptions, so whoever did the data description part had to do that part, and get that formalized.

HAIGH: Right. So the programmers were seconded and physically went to work in the same place?

BACHMAN: They'd go to work maybe a week someplace and would work at home for several weeks. Then where's the next meeting going to be[23]? It was intensive work. Of course, when they're working away from their office, they typically may not have had a computer. Nobody had a 709 in the beginning.

Of course, we were living in the punch card era, so all the programs were punched into punch cards. They've got their punch cards, but they couldn't get them debugged anywhere unless they happened to have a 709.

This reminds me of a very funny happening at the San Francisco SHARE meeting in the Fall of 1959, In the big plenary session, IBM was formally announcing the new IBM 709,

---

[22] Russ McGee corrected me recently. He said, "Yes, we had two IBM'ers who did nothing. I don't remember the name of the first, but the second one was "Ruddy" something."

[23] Russ McGee corrected me recently. He said, "No, it all happened in Queens (Union Carbide). I think the Ed Roddy did most of the work. Then they went to LA to debug at the IBM Western Data Processing Center at UCLA. The guy, whom I later knew at Honeywell, ran the place. It was Dick Hill. This is the same Dick Hill that I would meet at the ACM conference in 1960 and who would asked me to come out to Thompson Ramo Woolridge, in Los Angeles, for a job interview.

with prices and all. A senior IBM marketing executive named, Don ……, was talking and was saying that the IBM 709 was compatible with the IBM 704, except for a couple of good new features. One of the major players at SHARE was a man by the name of Walter Ramshaw from United Aircraft. Walt stood up and in a very loud voice said, “Compatibility is like pregnancy! Either you are, or you are not!” It brought the house down. Don rather sheepishly tried to make the best of an embarrassing situation.

HAIGH: As the SURGE project progressed, did you have any further contact with its people?

BACHMAN: No, I just knew it existed

HAIGH: You had said earlier that there was no way for an application program to invoke the GE 702 system to do some processing for it. Did that continue to be true for 9PAC?

BACHMAN: Yes, 9PAC had it’s own “handcalc” capability, where you had to code a function in assembly language and then invoke it from within a packet whenever a record of a defined record type showed up. It would execute the designated assembly language subroutine.

HAIGH: That’s a case of a 9PAC packet calling pieces of custom code, rather than the other way around.

BACHMAN: Yes. Absolutely. 9PAC was not a callable system. It was a “compile and go” system, and it was in charge.

HAIGH: When 9PAC was finished, do you know if there was a period of what we now think of as alpha or beta testing before there was an official release?

BACHMAN: I don't know specifically[24]. I would assume that it went through several phases and had different levels of credibility. How well is it really working? Does it do all these things? What’s missing? But by that time, I was detached from the project. I was working on the power plant scheduling project for Dow.

HAIGH: Do you know anything about how 9PAC was eventually used? What kinds of applications, what kinds of sites, how many of them there were?

BACHMAN: I have no information on that. There may be some place in the IBM archives because IBM picked up the maintenance of 9PAC and documented it. There were official IBM documents, and there must have been a user group. Somewhere that information exists, but I had become detached from it. I do know that it was used very extensively. Customer migration was one of the problems that IBM had with the 360. How do we take care of the 9PAC audience and customers? Because, it was an enormous audience.

HAIGH: It’s interesting that without having any actual experience with data processing on a tape based computer that you were drawn to this particular problem, and that you were jointly responsible for writing the functional specifications.

BACHMAN: The type of problems that were going to be solved were in the 709 were exactly the kind of problems that kept most the tabulating groups at Dow Chemical

[24] On reading my text, Russ McGee said, “Hell no! We released tapes and if there were bugs the Hanford team fixed them and released the patches.”

working. They were the kind of problems that are meaningful to a report generator solution. They weren't doing top management queries; they were doing the day-to-day business processing of orders, and manufacturing, and shipping, and payrolls, and things of this type. That kind of application worked very, very well. The place where it stretched the most was places like payroll where you had to compute pay, and there could be all kinds of different rules for computing pay. In fact, if you remember, your earlier paper talked about GE and a Univac machine at Louisville, Kentucky. The thing that almost killed them was trying to pay the last ten employees. I forget what number I was told, but if they had quit and left ten paychecks to be calculated by hand, or maybe thirty, the project would have been half as big. In fact, one of the things we find today with computers is that increasingly we don't computerize our manual systems. We convert our manual systems to what the computer programs could do. A lot of the special case freedom that used to exist no longer exists. The system is what the computer does. In other words, you begin to get standardization.

HAIGH: But still, as your 709 never actually arrived, when you describe your people working on a payroll system or working, they were still at the design stages, right? It never went into operation because there wasn't a computer to run it on.

BACHMAN: Yes, but the 702 system was running, and there was clear evidence of how it ran and did.

HAIGH: Did you have a 702 at Dow Chemical?

BACHMAN: No, but I'd used the 702 at GE Hanford. That year, I ran the annual sales reports for Dow Chemical on Hanford's 702 computer.

HAIGH: As a result of the contacts that you had made through SHARE?

BACHMAN: Yes, right. I just took the tapes with all the sales history, punch cards on them out to Hanford. I made up the packets we needed for the reports. Unfortunately, none of those packets survived.

HAIGH: I think there was one more question I had from this area. It's just the INGEN system, which there are a couple of reports in your papers at the Charles Babbage Institute. I was just wondering if you could say what that was, and if it ever led to anything?

BACHMAN: No, it didn't lead to any system or product result, although, it was a part of a ongoing thought process, regarding the maintenance of hierarchies within heterogeneous files. INGEN was a sort system. It was designed to be able to sort heterogeneous files, where there existed a hierarchical relationship between the various record types. And end up with all of the records in the right places, after you sorted them. Assume that I have a file with three kinds of records: customer records, product-customer records and sales-order records. Nothing came of it.

The following situation is what INGEN was supposed to handle: Let's say, in primary sales customer sequence, and I want to put the same information into product sequence and I have all these trailer cards with intermediate orders and items. How do I get them in the right place over here without pulling them apart, sorting them separately, and then merging them back together? INGEN was designed to be able to recognize and handle that kind of situation. It did funny things with the sort codes and what the prefix on the records. The thing to do on the trailer for this kind of thing, under the circumstance, and I wrote the specifications for it. Hanford sorted the files separately and then merged them in the end. If I wanted to resort them in a different sequence, I had to pull them apart into different record types, sort each record type separately, and merge them back together. The specifications you saw were printed on Mylar film, which makes a very permanent record. That is, the material that the blueprints of the company were printed on. The INGEN specifications were actually copied directly on a photographic machine designed to copy original engineering drawings.

Up until this time, the technique for creating a sorted file that contained two or more heterogeneous record types that existed in a hierarchical relationship was to sort the several record types separately and then to merge them together to create the desired hierarchy.

The Data Structure Diagram illustrates the a data structure containing Customer, Product, Sales Order, and Sales Item records. This diagram illustrates the one-to-many relationships that exist between these four record types. Note: this is a network diagram that is the result of joining two hierarchical structures that have the Sales Item record type in common.

Customer/Sales Order/Sales Item/Product Structure

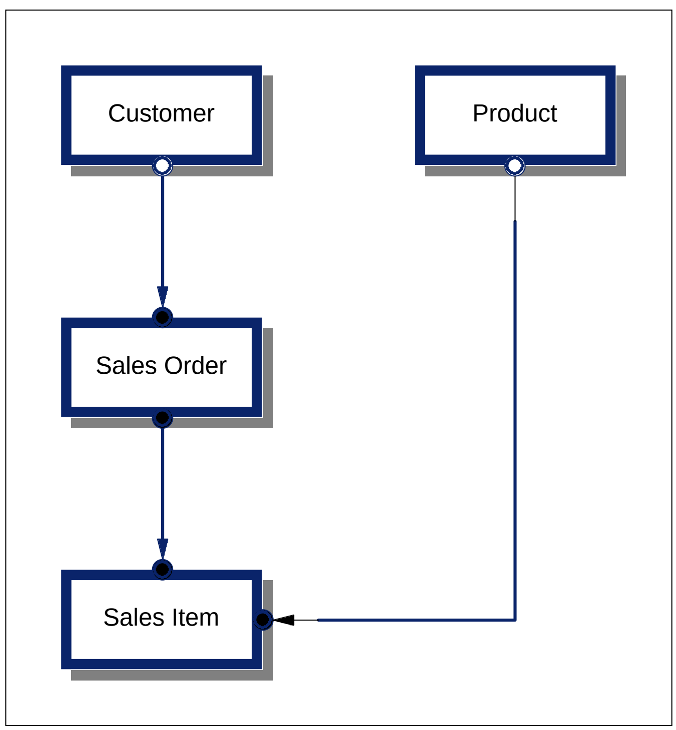

Figure 1.

The target for INGEN was to be able to instruct a sort routine to deliver various combinations of these four record types in different sequences to achiever different reports.

“Report-1” lists all of the customers, all of the products that each purchased, and for each product, it lists the individual sales for that customer.

“Report-2” lists all of the products, all of the customers that purchased that product, and for each customer, it lists all of the individual sales for that product.

“Report-3” lists all of the sales orders, and for each sales order, it lists the products purchased and their price and quantity.

All of these reports can be generated following the sorting of the various records of various record types to meet the specification of the desired report. Part of the means of the INGEN requires that each of the individual record types must be formatted in a manner that facilitates the sorting of the records into the required sequence.

Three Separate Report Structures

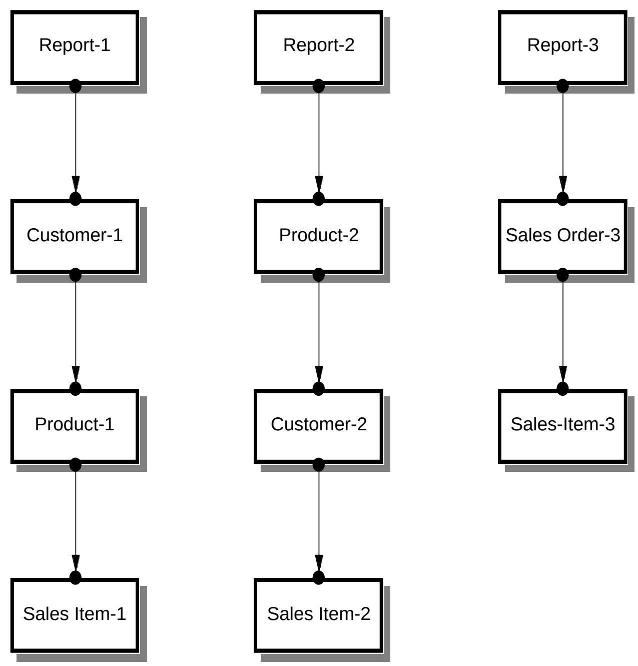

Figure 2.

HAIGH: It appears from the cover letter that it was considered as a possible replacement for both SURGE and 9PAC. INGEN was a next generation thought?

BACHMAN: Well, no. It wasn’t planned to go that far. What does it say there?

HAIGH: It's a letter from you, September 24, 1959, and the first sentence is "At the present time, there does not seem to be a ready common system to replace both SURGE[25] and 9PAC. One concern is time availability." Now you're saying that INGEN would be that system, but also that it wouldn't be clear if it could be ready in time anyway.

BACHMAN: It had to do with the sorting of these files that were maybe in one sequence for one set of reports, and would need to be resequenced as complex files. The sort routine would understand the file's old hierarchy record structure, and the new hierarchy structure that you wanted to put it in.

HAIGH: I do have a question. It's just about the experience of programming file operations on a 704 or 702 or 709 which would be exactly how low level were the details that the application programmer had to worry about? Obviously, there wasn't an operating system there.

BACHMAN: There weren't any application programmers in this day. They were just programmers, and they were operating in assembly language.

HAIGH: Was there some kind of interrupt that would at least tell you when the tape drive had finished reading in a record, or did you have to keep polling it to see if the record had passed by the edge of the tape head?

BACHMAN: There were tape read/write subroutines. There were libraries of those kinds of things, so our people didn't have to write them. Generally, a SHARE site, or IBM would supply them, for free.

HAIGH: Then you would link to them with the assembler?

BACHMAN: The assembly language code in the program would call them, or if there was an interrupt from a failure of some kind. The interrupt would trigger your subroutine. You could grab control of the interrupt, analyze the problem, and then solve it.

HAIGH: So that code would be written once and reused?

BACHMAN: That part, yes. Essentially, that was part of the report generator system, the part that controlled the tapes being read and being written.

HAIGH: How and why did you move to General Electric?

BACHMAN: Through the process of working with SHARE and the 9PAC project, I went to the ACM annual meeting in Milwaukee in the spring of 1960.

HAIGH: I think it's in 1960 that you joined General Electric.

BACHMAN: Yes. I joined GE Production Control Service in December 1960. I had met a man at the ACM annual meeting in Milwaukee, by the name of, Dick Hill. He was working at that time for Thompson Ramo Woodridge, in Los Angeles. He had worked with 9PAC, when he was managing the IBM Western Data Processing Center at UCLA. I think he had his PhD from UCLA. He said, "Charlie, do you want to come to work for Thompson Ramo Wooldridge?" He asked, because they were building a new computer that was currently in the early design phase.

[25] At the time of that letter, I was still thinking that SURGE was a report generator. It turned out to be a kind of an assembler.

I couldn't see that I was going anywhere with Dow Chemical, because they were interested in chemical production. I was strongly interested in information systems. So I talked to Carl Gerstacker. Carl, who was my boss and said, "Carl, I'd like to go out and talk to these people and see what they have, because right now it's looking a little flakey here for me." I ended up actually going to the GE Computer Department in Phoenix and taking to the people there. I talked to Dick Hill and his people at Thompson Ramo Wooldridge. I think Walter Bauer[26] was then running the computer business at Thompson Ramo Wooldridge. Then I went up to talk to Harry Tellier at Hanford. I said, "Is there a job that makes good sense?" In summary, there was nothing for me that seemed to fit in Phoenix, with the GE Computer Department. Nothing seemed to fit at GE Hanford, and the Thompson Ramo Wooldridge people said, "We'd like to offer you a job, the salary you're asking is alright, but given your current salary, that's too big a jump and we can't justify it." Then nothing happened for a couple of months. Then Harry Tellier contacted the man in New York and said, "Charlie Bachman is out. Do you want to talk to him?" That man's name was H. Ford Dickey, who was the head of "GE Production Control Services, in New York City.

HAIGH: What did Production Control Services do?

BACHMAN: AT that time, General Electric had a very substantial service organization, corporate specialists of various things. At the top level, they had a finance service, and a manufacturing service, and an engineering service, and a personnel service, and a marketing service. Maybe some others that I can't think of. Manufacturing Services had an Advanced Manufacturing Services group. They had Materials Services. Under Advanced Materials Service, there was Production Control Service and Purchasing Service. I think that there was also a Quality Control Service group. Production Control Service looked into production scheduling and inventory control, and things of this type. It was a specialized group looking at how to keep track of what materials we have on hand, how do we make things, when do we make them, and economic order quantity calculations. How do you do things of that type?

HAIGH: So that was a staff group at the corporate level?

BACHMAN: Yes. A staff group at the corporate level. The next group ahead of us, Advanced Materials Service, was also in New York City. Bill Bryant was the head of Advanced Materials Service. Hal Miller, who was head of Manufacturing Services, was located in Schenectady. These groups were staffed with well paid people, who were functional experts and who had done well in manufacturing, wherever they had previously worked. This was a place to give people they thought, were not were not line manager types, but were very knowledgeable in their specialty." It was an alternate way to promote them to a better job and share your skills. Carl Bates was the head of Information Systems Service within the Accounting Services. He was quite helpful to us during the period when the ISP II project was underway.

HAIGH: That sounds to me like a systems man, even if they didn't use the term there.

BACHMAN: Well, they were certainly concerned with systems, but they weren't generic systems people. They were functionally specialized. I guess they were all systems people,

[26] Walt Bauer was one of the key people in the Informatics report generation system development.

whatever they had to do. They were concerned with: “How to manufacture things?” “How to ship things?” “How to pay for things?” People were specialists in “how to do it.” If that makes you, a systems person, so be it. It was not specialized to information systems. It could be whatever, but it was a way to handle professional specialization. People could come and coach their specialty, and teach people how to do it. They were coaches. They were teachers and researchers in their field of specialization. My specialties were information systems and manufacturing.

HAIGH: It seems like at this point General Electric was extremely active in these areas of systems planning and operations research.

BACHMAN: They had a very strong group of the Operations Research Service, under Mel Hurni. We will come to someplace down the line to talk about our meeting with them, because one of the memorable stories concerned MIACS and IDS.

HAIGH: In fact, I have an article by H. Ford Dickey called “Integrated Systems Planning at General Electric”, which was published in a book called “Management Control Systems, 1960”. The Systems Development Corporation had sponsored a conference on how these Cold War high tech systems concepts could be used in business management.

BACHMAN: I think that’s one you referenced in your paper. Ford must have gone to that meeting.

HAIGH: So he had some kind of reputation in the broader systems community as someone who was applying cybernetic techniques--

BACHMAN: He was a well-spoken guy. He handled himself very well in public. He was a difficult manager, though. When you evaluate managers and say what is the most important thing managers do? They hire people, and they fire people. He was very good at hiring outstanding people (I can claim that because he hired me). But he also hired some other great outstanding people. He was terrible at managing. If he didn’t like what you’re doing, he’d tell somebody else so they would tell you. He could not tell you that you were doing something wrong. It wasn’t in him, and he didn’t really know how to manage a project once it got started. He was a good, big picture guy, with a salesman personality.

HAIGH: Do you know what happened to him?

BACHMAN: Last I heard, they had a big retirement party for him. I wrote a nice, long letter for it. I never heard anything back from Ford. The last I heard, he was still living in Riverside, Connecticut.

HAIGH: So he stayed with GE the whole period?

BACHMAN: Stayed with GE until the end, yes. H. Ford Dickey.

HAIGH: Did you see a big difference between the kind of level of managerial sophistication and culture at GE and Dow Chemical?

BACHMAN: I think they were both struggling, had problems. GE had a bigger professional staff. I think GE was probably better in formal management. They did more in formalization and systemization, if you go back to your word “systems.” If you have 100 manufacturing departments, then you don’t want them all doing their own thing. You can't afford to have them all doing their own thing, so you try to factor some of those things (methodologies) out and have the services organizations supply that guidance. Build templates, from which the departments could work. When we got to the Integrated System Project Nr. 2 (ISP II), the goal of was to create a generic manufacturing control system. At that time GE had 100 product departments, all trying to develop their own manufacturing control system, and they weren't doing very well. They were spinning their wheels. They were spending a lot of money, and they weren't getting results. Our ISP II project was going to be a corporate staff project that was to be staffed with better people and more time. Each department is always under the gun to do something quickly. Doing something quickly, in the new area of information systems, is fraught with failure.

GE pioneered the term, "Professional Management." That one of GE's strengths and also a weakness. Every GE manufacturing department had profit and loss responsibility. And loss was not tolerated for long.

HAIGH: When GE hired you, what was the position that you were hired for?

BACHMAN: I don't even remember if I had a title. It was probably a computer specialist or something like that. I don't remember the title as important, either. All I know is that Stanley B. Williams was the project manager. Stan had been brought down from Pittsfield MA, where he had worked in the engineering department for the Large Transformer Department. There were three of us from Production Control Services. Bob Connors had been brought in from the Small Steam Turbine Department, which was in West Lynn, Massachusetts. He had been doing work there on their early IBM 704 computer system.

Then we had a guy named Dan Langenwalter from the Engineering Services, John Neuenschwander from Information Systems Service, another man from the Advanced Manufacturing Services, Herb Nidenberg, and Ed LaChance from Quality Assurance Services. There was a team of about six or seven people who had this project placed on their table. It was a major assignment. I don't think that any of the members, outside Production Control Services, had the ISP-II project as their fulltime assignment. They didn't work on it fulltime anyway. The first thing we had to do was find a GE department that we could use as a laboratory. This project is not something you could sit at home and dream up. You had to go find a real department and say, "Can we do something that will solve your real problems?"

HAIGH: Was this the first department that you had worked on at GE?

BACHMAN: The department we started working with was called the High Voltage Switchgear Department. We worked with the High Voltage Switchgear Department through the flip chart presentation stage. That was the time when the MIACS and IDS conceptual designs were complete and were presented to the HVSD and the GE Services managers and executives. The purpose was to sell the HVSD and service executives to commit to the implementation phase. We didn't have view graphs in those days, let along having PowerPoint slide shows. Personal computers had not been invented! We had flipcharts, and the flip charts were drawn by hand. My hands and Stan Williams hands. The HVSD bowed out at that point. Don Beaman, the general manager felt that we were too big a risk. The first implementation went into the Low Voltage Switchgear Department, which happened to be in the same building in Philadelphia. Bob Lewis was the general manager of LVSD, at that time. We will be ever grateful that he was willing to see if we could accomplish something that had never been done before.

HAIGH: Yes, I think you described the work with that department in the earlier interview with Frana, so you've discussed the details of that.

HAIGH: What did this term "integrated systems" mean to you at that point?

BACHMAN: Probably just what the words say, if you can take them at face value, "systems that are integrated."

The first Integrated System Project (ISP), which preceded us, had spent a lot of time looking at "decision tables," also called, "structure tables". That technology that had been originally developed up in Pittsfield, MA, by an engineer, named Don Radke. The ISP people formalized the concept and made a programming language called TABSOL, for the GE 225. They were also doing simulation work. They had some very good people. One of the people on the team was named, Harry Markowitz. He was subsequently a Nobel Prize winner in Economics, for the research, he had done prior to coming to GE. He helped developed the theory of derivatives. He was a very bright guy, and he was involved in simulation language, SIMSCRIPT. I said that Ford Dickey hired good people.

The ISP I project didn't achieve any lasting results that anyone could see, so top management at GE said, "Start again," which we did. We took a whole new group of people working on the same problem. "How can we better automate manufacturing control?"

We started again, a new set of people, looking for a company department in General Electric that would put up with us, while we do a look-see. The GE departments were very independent. Each department general manager was a czar. He got paid based on the performance of his department. He could do almost anything he wanted to do, as long as he made money. That's what big corporations do. You say, "This is your budget. This is what you've got to do. Tell me at the end of the year what you've done. In fact each quarter tell me what you've done for me this quarter[27]." Top management didn't meddle much. People didn't look too closely at what the departments did, either[28]. But, "Make your numbers."

One of the stories that we heard at High Voltage Switchgear was they were saving money by not maintaining the building. The roof was leaking, but instead of spending money this year to fix the roof, they did other things that would improve this year's profit. Fixing the roof doesn't improve your profit, this year. After we spent the first year and went through the whole major evaluation, the High Voltage Switchgear general manager, Bob Beaman, decided he didn't think he wanted to continue the project. Maybe, because he thought it was too risky and was going to cost him too much money. In GE management circles, this was clearly his decision to make, based on his judgment, end of story.

[27] Many companies have "fiscal years" which are different than calendar years. GE had "fiscal quarters which were different than calendar quarters. Each fiscal quarter was exactly thirteen weeks, 91days, long. This made it easier to compare results quarter to quarter.

[28] General Electric maintained very strong control over the finance and accounting function. The manage of finance in a GE department reported directly to the corporate finance staff and had only a dotted line relationship with his department's general manager. No fudged books! No "Enrons" permitted here.

The High Voltage Switchgear, Medium Voltage Switchgear, and Low Voltage Switchgear departments all resided in the same six-story building. It was a big old fashioned, manufacturing building in Southwest Philadelphia, out toward the airport. The management of the Low Voltage Switchgear Department, Bob Lewis said, "We could gain from this," so they signed up to help go into the implementation phase. For them, the project become a big success and they were still running the MIACS system we developed years later, after converting it to run on the GE 600 computer. It is important to say that they made major contributions to refining the conceptual design of MIACS and getting it up and running. Herb Fullerton (Engineering) and Ernie Messikomber (Production Control) were the heart and soul of the LVSD team[29] and deserve the highest praise. They programmed most of the problem solving routines that made up MIACS.

HAIGH: You mentioned that there was someone from the corporate specialist group for accounting, and other staff members from other areas. The idea was that an interdisciplinary team of corporate staff experts specialized in different areas would go, and then together you would find out how to integrate all these different pieces into a single system.

BACHMAN: Yes, correct.

HAIGH: In these documents, I've seen references to Integrated Systems Project II, MIACS, and GEICS. Are those all different names for the same thing?

BACHMAN: The Integrated Systems Project II was the project. A couple products came out of it. I have some documents. The manufacturing control system's (MIACS) conceptual design was done very early. The Integrated Data Store (IDS) was designed to store, access, and maintain the networks of manufacturing information required to support MIACS. We did the manufacturing system conceptual design first. Then we asked, "How do we program that?" The Integrated Data Store, the first database management system, was a derivative, or a byproduct, of MIACS.

HAIGH: MIACS was the system--

BACHMAN: The driving force.

HAIGH: That you produced at this point, and this GEICS[30] is a productized version of MIACS[31] that Stan helped the GE Computer department to put together latter on.?

BACHMAN: Yes. "ISP II" is just the project flag. Think of it, that we were flying under that flag. It had generated the Integrated Systems project, but the result of the project, when completed, was called "MIACS," What the initials stood for, depended upon who we were talking to. If we were talking to Bill Bryant or Ford Dickey, it meant <u>M</u>aterials <u>I</u>nformation <u>A</u>nd <u>C</u>ontrol <u>S</u>ystem. If we were talking to Hal Miller, it meant <u>M</u>anufacturing <u>I</u>nformation <u>A</u>nd <u>C</u>ontrol <u>S</u>ystem. If we were talked to higher management, it meant <u>M</u>anagement <u>I</u>nformation <u>A</u>nd <u>C</u>ontrol <u>S</u>ystem. The acronym

[29] Jack Kuhfuss and Jack Chambers were other major contributors for LVSG.
[30] GEICS, General Electric Integrated Control System. GEICS was renamed after the GE/Honeywell merger and marketed quite successfully as HMS, Honeywell Manufacturing System.
[31] A great deal of technical information concerning MIACS and its working are in the private files of Stan Williams. Stan currently lives in Scottsdale, Arizona.

would take on three different meanings, depending on to whom you were talking to at that moment. Each audience had its own ax to grind.

HAIGH: Do you think that you were successful in performing this integration? Did it turn out to be harder than you expected, or did it go according to plan?

BACHMAN: I would say it turned out to be more successful than we expected. We didn't know what to expect. It turned out different than the plan because we didn't really have a plan. This in some sense was a basic research project, which turned out a commercial product somewhat by accident. Not by accident, but by the force of the people involved. A real research project, you don't know where you're going except in a broad sense: how do we better run these factories?

The final result is we ended up with several products, IDS which became a commercial product on its own. MIACS became a GE manufacturing control product on its own. The operating system part, which we developed to support MIACS, we call the Problem Controller, showed up in various places in different ways. It's a little discussion of its own. We called it the Problem Controller because GE was advertising themselves, in those days, as "the Problem Solvers". We said, "Well, each job we've got is a problem. The Problem Controller is to control the problems we're getting in."

In some sense, the Problem Controller was a message "store and forward" system, although we hadn't really thought of it that way. Its native ability was to accept jobs of various kinds as they were fed them into the computer, through the card reader. That's the only input device we had. The first card of every packet identified the kind of problem that was to be solved. It was followed by a bunch of job detail cards, containing the problem data. Not only would people submit problems, but, the problems being solved would submit more problems to be solved. If a computer program solving a problem, had a subtask to be executed, it would generate a problem packet, which it could store in the same manner as the problem packets coming in through the card reader. This mechanism was used extensively for the parts explosions. If a product requirement started at the top level, where there were five top-level circuit breakers to be made, each circuit breaker would generate demands for parts at lower levels. These demands are captured as messages (problem statements) sent to the material items at lower levels in the parts explosion structure. The parts explosion algorithm wanted to go down one level at a time, summarizing the demands on that level, and exploding those demands to lower levels. The same computer program would be called in at a different levels and solve the same type of problem at each level. Now the actual problem was not quite that simple. At each level, the program had to decide whether taking required parts from unassigned inventory, or whether a new part or batch of parts had to be manufactured would solve the part requirements. If the latter were true, then the production planning would trigger demands for further levels of parts explosion.

HAIGH: Now in the earlier interviews, you spent quite a bit of time talking about the parts explosion problem. Was that part of the MIACS system?

BACHMAN: Yes. It was part of MIACS.

HAIGH: Can you just list what the main components of MIACS were that were integrated?

BACHMAN: There were a number of components. I probably can come up with a partial list, off the top of my head. Although, I was not involved in the planning all of them. The list would include: parts explosion, production planning, resource planning, economic order quantity calculations, order dispatching, purchase order planning, purchase order dispatching, receiving, stocking, etc. Stan Williams would be a better source for this answer.

After MIACS was up and running, Stan Williams told me that the economic order quantity algorithm that I had specified was an example of "dynamic planning." I was not familiar with the dynamic planning, and had not applied it consciously. Rather, I had identified a problem. The MIACS database provided the opportunity to look at both the immediate requirements for each inventory item and also the currently recorded future requirements.

The manufacturing planning and dispatching algorithms were also something of an innovation. They created an automated plan that followed the strategy of the old time production foremen. We <u>planned</u> production quite late in the available production schedule, to leave the factory resources under loaded, in the near term. That way we could fit late arriving orders into that available factory capacity. Our goal was to be able to create a new, "feasible" schedule with the receipt of each new order, or feedback coming from the factory, as they were known.

Daily, we dispatched orders to production, with a totally different algorithm. This algorithm was designed to create enough factory orders every night to keep each factory worker, machine tool, work place, and test instrument busy the next day. Thus, many items were actually worked on days or weeks ahead of its original schedule. This made it possible to execute the manufacturing planning again the next day to cover newly received orders, and cover deficiencies in the factory orders dispatched the night before.

The daily manufacturing planning was not a complete re-plan. Rather, it was an incremental planning exercise design to create a new feasible plan from which the next day's factory orders could be created with a high probably that they could be executed successfully. The people, the machine tools, the work places, and the required parts should be available.

HAIGH: So in what ways would this be different from the things that were later sold as MRP (Material Requirement Planning) systems?

BACHMAN: They were copied from the MIACS "bill of materials" subsystem. They were copies of MIACS. DBOM, IBM's Bill of Materials program, and all those things which came along later, were all derived from MIACS. GE was trying to sell MIACS to the Blacker and Decker Company. They were very inquisitive as to how we did certain things. GE people made a big presentation to them. We told them how we accomplished such marvelous things, i.e., bill of materials processing. After long discussions, they called in IBM to ask them how they would solve the same problem. Ollie White was the lead man from IBM. After some long discussions with IBM, when IBM was doing more listening than talking, Ollie White went back to IBM and shortly later IBM started talking about their "Bill of Materials Processor."

MIACS was the first integrated, manufacturing control system. It is much larger than just bill of material processing. There is more than determining how many parts that you need. You have to resolve the problems of how to get these parts and when to assemble them and who is going to do it. Why? Well, if you look at what happens in a manufacturing plant, it looks like a PERT diagram or a CPM diagram. There are many job steps that had to planned and executed in the right sequence. MIACS CPM diagrams were a different problem, from the typical CPM diagram. They were created to build a single building, where you executed each job step or task only once. When planning for manufacture generic products, we had to generate plans with the same type of job steps, hundreds of times, in a continually varying combination of jobs. MIACS had to generate all of the job steps that were going to be planned, organized, and executed. They were generated from “structured” planning materials. And MIACS reschedule them whenever the current schedule was no longer feasible.

Bill Of Materials Structure – Data Structure Diagram[32]

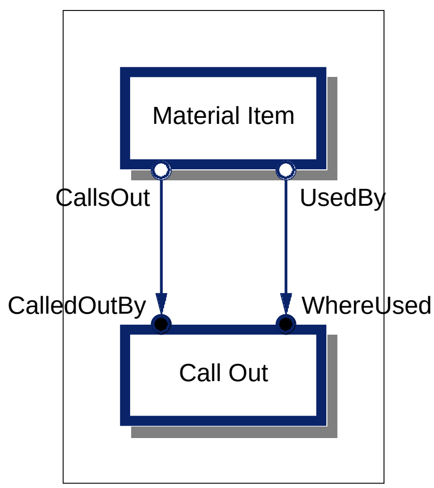

Figure 3.

In the early 1960s, General Electric had a theme and training program for managers called, “POEM.” The letters stood for “plan,” “organize,” “execute,” and “manage.” We tried to make MIACS responsive to the POEM theme. What we found and what we implemented was a business cycle that started with the planning phase. That was followed by the “organizing” phase. That was followed, in turn, by the “execution phase.” But what could be done to support the “managing” phase? We concluded that the managing phase really meant beginning the next planning phase. Managing meant taking

[32] The illustrated data structure is the classic example and presentation of the Network Data Model.

the new set of facts available and planning the next steps forward. So we had a “POE” cycle that was continuing, day after day, after day.

Plan: Day after day, MIACS looked at the new work that had come into the plant plus the feedback coming from the factory, and set out to make a new “feasible” plan.

Organize: MIACS would organize resources to execute the next day’s work. It sent the next day’s work schedule to the shops.

Execute: The shops would execute that plan and feed back their results.

That night the cycle would begin to repeat itself: Planning, Organizing, and Executing the next feasible plan.

The word “feasible” is used here, to mean a plan that would work if nothing changed or went wrong. Something always changed, but that was the name of the game, so plan, organize and execute the next feasible plan. Our goal was not to generate an “optimum’ plan, but rather to generate a feasible plan, as it was only expected to last for one day. After one day, an optimum plan would have been out of date too. Achieving a feasible plan would be a big step forward, as Low Voltage Switch was currently operating with a number of obsolete plans. Expediters were running around the shop trying to find out what had happened and what could be done to straighten things out. It was embarrassing to have customers call in and ask where was the order that was promised to them for delivery, the prior week.

HAIGH: Now you’d begun the study effort for this project in December 1960. At what point did you start to move from design specifications into implementation?

BACHMAN: Well, we wrote the first detailed specification of IDS in the fall of ’61. That’s after we had conceptualized the MIACS system. In fact, one of the stories that I did not tell Frana, I’ll tell it to you. When we were going around looking at the High Voltage Switchgear Department, we looked at the parts manufacturing floor, a huge area, with hundreds of machine tools that people were making, cutting, grinding, milling, painting. We noticed that all around that periphery of the room and down the aisles, there were steel boxes, called “tote bins”, full of parts. Typically attached to each box was a pack of punch cards. Each punch card represented a manufacturing operation that was necessary to complete that part. If you could not find the first card, it was because they had already made some progress to get it into the tote bin. At least it’d cut off the bar stock to a length for some purpose. The next step was going to grind or polish them, or do something else. We looked at the cards. Each card had an operation and a date when this operation was supposed to be done. This deck of cards was all produced at one time, when a stock order or special order was planned. They made a manufacturing plan in the computer on the day the order arrived, or within a few days anyway. This might be covering two, three, four, five months of work because some of the switchgear products were big, complicated machines. The fact is once that plan was computed, there was no procedure to fix that automated plan. It was fixed, it was printed. The only thing they can do was have a bunch of expeditors run around and hurry things up.

HAIGH: This was the High Voltage Switchgear plant?

BACHMAN: High Voltage Switchgear is what you’re looking at.

HAIGH: So it was already computerized?

BACHMAN: Already computerized. All of the Switchgear Divisions departments used the same computer, down on the first floor, for their manufacturing planning.

HAIGH: Did you know what kind it was?

BACHMAN: It was a GE 225 just like we were implementing IDS for. A GE 225, for those that don't know, was similar a IBM 704-like machine, but smaller and slower. Lower price, smaller memory, smaller word size. Programming it was like programming an IBM 704. Similar instruction set, although it didn't have a halt instruction. If you wanted to stop the machine, you wrote a branch instruction that branched to its own location. We called it a "stickhalt." The computer would sit there and spin at that location. The machine was executing instructions, but nothing was changing.

HAIGH: So you'd said that the overall specifications including the specs for IDS were completed in 1961.

BACHMAN: Yes they were. Top-level specifications. In fact, we had a group of people called "The Friends of IDS" that met from different GE departments. That was the Fall of 1961 to review these specifications of what IDS was like, what it would be like. And, we got kind of a uniform blessing from these people.

HAIGH: Who implemented the systems?

BACHMAN: In this case, I was the primary implementer.

HAIGH: For IDS?

BACHMAN: For IDS.

HAIGH: And how about the rest of the MIACS system?

BACHMAN: Well, I was the principal implementer with Homer Carney's help, in the beginning[33]. Homer's the guy who was loaned to us by the GE Computer Department in Phoenix. They made Homer their contribution to the project. He was an experienced GE 225 programmer. And I was hardly a programmer, at all. I had two weeks programming experience before this point in my career. We also had Stan Williams, who was the project leader. When we had the whole thing specified, Stan took over the MIACS detail planning. Most of MIACS was programmed by Low Voltage Switchgear personnel. I did the detail programming at the higher levels of IDS code, and Homer Carney did the lower end, i.e., the virtual memory and disk reading and writing part. If you think about IDS, as a virtual memory system, my code stopped when I asked for access to a record in virtual memory page. "I want record, so-and-so. Just tell me where it is." My code did not care whether you found the record already in one of the buffers, or had to go out to the disk and fetch it. My code waited, and then would pick up and go from there. It was Homer's job keeping track of what pages were in memory and whether those pages had or had not been updated. Either way, Homer's code needed to find an available buffer in which to read into which to read a new page. He would age pages, for inactivity. The page that had been inactive the longest time would be the one that was discarded next. Either written

[33] Phil Farmer came in to replace Homer when he was called back to Phoenix.

back to the disk or just overwritten, if it had not been updated. Homer's section of IDS was called the "Block Controller."

My code would interact with the application programs (problem solving routines) and execute their IDS "CRUD[34]" commands, exchanging information with the application programs in the working storage areas established for each record type.

HAIGH: How long did it take to finish the overall MIACS system?

BACHMAN: The overall MIACS system wasn't ready for usage until the Fall of '64, after I had left the project.

HAIGH: So the project took about four years from the initial study through to the implementation?

BACHMAN: Yes. But, IDS was in use before that.

An interesting tidbit. The GE Computer Department had almost no disk files at the time that we started this project. IBM had disk files, Univac had drums, and GE had disk files that were planned to come. During the first programming of IDS, we had no disk file to debug on. So we used two magnetic tape drives to simulate a disk file. Homer Carney modified the Block Controller to simulate a disk drive by using those two tapes. If I think of a database as being made of a number of virtual memory pages, this was a paging system. Pages consisted of lines, which were the data records. In a normal disk file, you'd moved these pages, back and forth, between the computer and the disk. In this case, we initialized the two tapes, each with 64 empty virtual memory pages; page numbers 0 to 63. Each pages had the page header record set to represent an empty page.

If IDS asked for page 31, the Block Controller would ask, "where am I positioned on the current input tape?" If the input tape was positioned on page 23, the Block Controller would continue reading blocks 24, 25, 26, etc., off the tape and writing them to the output tape until reaching page 31. It would read Page 31 into the computer and control would be given back to IDS to process that page.

Alternately, if the next read requested by IDS was for page 17, or some other page with a page number less than page 31, The Block Controller would fall back on its alternate strategy. In this case, the Block Controller would read the input tape, reading page 32, 33, 34, etc. up to and including the last page on the tape, page 63. These pages would be written out to the output tape until page 63 had been written. At that point, the functions of the two tapes would be switched. The current input tape would become the current output tape, and visa-versa. At that point, the Block Controller would ask the original question, "Where am I positioned on the (new) input tape?" It would be positioned prior to page 0. The block controller would copy all of the blocks from the new input tape to the new output tape until it reached the desired page 31.

If the Block Controller wanted to write a page that had been just updated, then it would go through a similar process to position the input and output tapes so that the updated page could be written to the current output tape. The page positions of the input and output tapes were always kept synchronized.

[34] "CRUD", meaning Create, Retrieve, Update, and Delete.

We did all the early programming and testing of the in-line coded version of IDS using this tape system. It was a slow, but totally functional. No problem at all. When it took an hour to recompile the IDS DML translator, what were a few seconds, or a minute of tape time?

HAIGH: That would be a good way of testing the reliability of the tape drives.

BACHMAN: Well, that wasn't part of our mission. Tape drives were probably more reliable those days than disks were.

HAIGH: Yes, and just pulling up some of my records, it seems that Univac announced their disk drive in 1961. I know that in the beginning Bryant was also a big supplier of disk drives, but it was I think a few years after that.

BACHMAN: I don't think GE made their own disk drives. I think they were made by… Who's the man who put all the money into the Charles Babbage Institute?

HAIGH: Tomash.

BACHMAN: Ervin Tomash.

HAIGH: Data Products.

BACHMAN: I think the early GE disks were his products.

HAIGH: Did you know from the beginning of ISP II that you would need a disk for this system to work?

BACHMAN: Well, we knew from the start of the project the disks were coming. Our whole assumption is that this system could not work without disks. In other words, you can't have an integrated system without direct access to data. They were a reality at IBM and other places, and they would be a reality in GE. Although in those days, they were used mainly to store programs. They were generally not being used for data storage and retrieval. There weren't enough of them around, enough capacity around. In fact, the first disks that IBM used for production, data processing whatever those were, had removable disk packets. You'd take them off and change files like you used to change magnetic tapes.

HAIGH: Yes, IBM had the removable disk platters.

BACHMAN: Yes. They served a different purpose. In other words, their functional use was different. They were not databases devices. They were sequential file devices.

HAIGH: In the interview with Frana, you spent some time talking about a RAMAC. I wasn't quite sure where that entered the picture.

BACHMAN: Well, the RAMAC was the first IBM product.

HAIGH: I know what it was in general. I was just wondering did you have one for this project? Was it for a different project?

BACHMAN: It was a M204 or some number like that.

HAIGH: 305.

BACHMAN: 305? That was the IBM 305 RAMAC. The 204 was the GE product number I believe.

HAIGH: Oh, sorry.

BACHMAN: Remember we talked about hash addressing at lunchtime. The people were trying to use calculated addressing to directly access the RAMAC. RAMAC was designed so that one addressable unit on the disk could hold one data record. And not a very big record at that. I think it held one punch card record or equivalent; maybe slightly larger, maybe 120 characters. The notion of using calculated addressing to find records on the disc would work, if you'd only fill the RAMAC up to 40-45% of its capacity. At which point you begin to get duplicate addresses generated with the calculated addressing algorithm. If one record space is already full, what does the RAMAC do now? Then an overflow address would be tried, and another, and another until an empty space was found. Calculated addressing proved not to be practical on the RAMAC. Consequently, the RAMAC was not greatly successful.

HAIGH: Were you personally using RAMAC for this project?

BACHMAN: No, I never did. I only read the specs. I sure that I've seen one, but I couldn't even be sure where that was.

HAIGH: Let's now zoom in from the overall MIACS project to IDS, which again you spent a while talking about in the earlier interview. Would I be right to think that at the beginning, the biggest difference between IDS and 9PAC would be that IDS was designed to be used with disks rather than tapes?

BACHMAN: That was one of the fundamental shifts[35]. That was the thing, which enabled the system to exist at all. That was the enabling legislation; without that, MIACS couldn't exist. Getting access to one of big disks wasn't our problem. That was GE computer department's problem in delivering one of them to us at some point. Our biggest problem was really looking at how do you build a system that lets you access whatever you wanted to get, whenever you needed it. Essentially, that is a case where we assembled a large number of different technologies to create something very new and useful.

The first thing we needed was the ability to have a huge address space. Actually, we need two things. First, a big address space (virtual memory) is part of the story, but list processing was essential too. If you think about being a list or a chain of things, you'd like to have a purchase order that is associated with its line item records. I want to go from the purchase order to the first line item record, to the second line item record, . . . to the last line item record have it points back to the purchase order. I can start with any line item and get to the purchase order. This is a set traversable in either direction. From the owner to the members, and from any member back to the owner.

So if we think about a hierarchical system, I get from the owner to any member, but I can't get back to the owner because they are one directional. You can get from one detail to sub-details. The program can go in only one direction, the way the punched card file or the magnetic tape file is flowing. Here, in this case, we can go either direction from any detail to any owner, any owner to any detail. We can go from any detail to either it's next detail, or to it's prior detail. Even more important, any record can be a detail in several

[35] From a programming point of view, IDS was a shift for the report generation paradigm to the record-at-a-time navigating paradigm. It was following GECOM, COBOL and other programming languages.

different relationships, in other words a member in several different hierarchies, which is what we needed to implement the CPM network of orders, items, resources, job steps, and whatnot.

HAIGH: So that would be the origin of the network data model?

BACHMAN: Yes, because a network in the database was able to model a network in real world.

HAIGH: At this point, were you aware of terms like linked lists and graphs and the other kind of vocabulary that computer scientists have used to describe data structures?

BACHMAN: There was an article that I read some place before I joined GE. I thought it was in ACM Communications or ACM Journal. Recently, I went back and looked at the ACM library, through Google, but I could not find what looked like the article I had read, so maybe it was someplace else. One of the well-known wizards in the early artificial intelligence world had written the article about linked lists. It was in my mind before I went to GE. I had read that and it just stuck there. I can still see in my mind the illustrations of these linked lists.

Sometime, before I went to GE I had read about the Ferranti Atlas and the way they had used their “backing store.” The backing store was the first virtual memory. They used a drum, from which they paged programs in and out of their main memory. It backed up their main memory. If a program instruction attempted to address a location in a page, which was not currently resident in the main memory, the address development hardware would say, “Time out. Get the page. Bring it in. Now, go ahead and continue executing.” They were using the virtual memory for program execution. I said, “Well, I want to do the same thing, because I want to access database records, as if they were physically located in a very large main memory. So again, I adopted that technology to a new purpose.

In a 2002 lecture at the Computer History Museum I said, IDS was an assemblage of a number of different ideas. The notion of hierarchies, the notion of data descriptions, the notion of randomized addressing, the notion of PERT/CPM type things, the notion of a virtual memory were all in use somewhere. Now how do we put it all together? That assembly is the thing that changed a business requirement into a very practical product that was the first database management system and a very prominent product for many years. It is still used in specialized applications where its performance advantage is critical.

HAIGH: It sounds from what you’re saying as though the team also had to write an operating system from scratch. Did the GE 225 come with any kind of operating system?

BACHMAN: No. It came with a one-card loader. It had some other facilities. It had an assembly language, called “GMAP.” It had a programming language called “GECOM,” which was a predecessor to COBOL. It was a COBOL like language. There are a number of these COBOL precursors. GE had GECOM, and Honeywell had FACT. IBM had COMTRAN.

HAIGH: Yes. You talked about those in the Frana interview.

BACHMAN: The industry effort created COBOL, which wasn't as good as any one of the precursors, but it was an industry standard language, and that is what the US Defense Department demanded. That was the most important to the Defense Department. They said they weren't going to buy any more computers that didn't run a standard language. I was familiar with GECOM, but that wasn't useful to me, because it was a sequential file oriented system. COBOL was also designed to read and write tapes and punch cards, or read and write other serial media. Cards, pen and paper, and magnetic tape were all serial media. GECOM/COBOL had no capability to add a disk drive or know what to do with a disk.

Another thing that was an invention was the database language. Even more important is that it was a programming language complementary to COBOL and GECOM. It was designed to be a natural extension. The natural things you'd do with everything else you'd do in GECOM. It wasn't learning something totally new. It was something that a GECOM or COBOL programmer would feel comfortable doing. It would behave in a normal way, or in an expected way, because you wanted anything new to be as easy for people to adopt as possible. IDS's record handling paradigm closely matched COBOL's record handling paradigm. It focused on the "record-at-a-time" processing paradigm that supported the "current record of program" concept. When a new record was stored, it became the current record of the program. When a record was retrieved, it became the current record of the program. The current record of the program was the target of the modify and delete commands. That current record could be <u>C</u>reated, <u>R</u>etrieved, <u>U</u>pdated, and <u>D</u>eleted. These were the standard CRUD commands, expressed with a network data model flavor. The truly novel part is that you could read the next record in multiple dimensions, rather than the single dimension of COBOL's card files and magnetic tape files. List processing was the first implementation of the multi dimensional navigation.

While being a record-at-a-time approach to processing, there was one significant shift in thinking. GECOM and COBOL programmers thought of I/O as being into the program and out of the program. IDS programmers thought of the program as navigating through the database. Records were stored in the database and retrieved from the database.

HAIGH: Is it the case that there would be a piece of IDS loaded into the machine that would, as with a modern database management system, sit there almost as part of the operating system and mediate between requests from application programs and the underlying files?

BACHMAN: You have two questions there. One is the operating system question. In this case, IDS with its companion operating system (the Problem Controller) totally sat in the computer memory and directly drove the disk. There was no disk support system beneath this. We programmed direct disc reads and writes and everything else. That's not the way things are built today. We did all the disk space allocation and everything, and loaded the pages of the virtual memory off the disk directly into memory. There was no support like that available. If you looked at an early IDS application, you would find the operating system that we built. We had asked the GE Space Vehicle Department in Paoli, Pennsylvania to write one for us. It required over 2,000 instructions, and could hardly fit into an 8,000 instruction machine and leave room to do anything useful, so we just set it aside.

So, I wrote an operating system, which was probably less than 100 assembly language instructions. It used IDS to store and retrieve the required control information. Our critics beat me up. They said, "IDS is going to be too slow to support the operating system's requirements." I said, "Gee, this machine is running IDS 98% of the time, for applications. Why can't we use the same database to run the operating system the other 2% of the time?" The operating system requirements were very simple. I said, "It only has four kinds of records." There was a "System" record, a one-of-the-kind record[36], at the top of the problem definition hierarchy. When the operating system finds it, the operating system can the find its list of the "ProblemType" records that represent the various types of problems that have been defined by MIACS. They are linked together in the sequence defined by their problem type codes. I can find problem type "10," or "20," or "40," or "100," whenever appropriate. The ProblemType records each had their own list of "ProblemStatement" records, representing the problems that a person or an executing program had submitted for solution. If someone or program were to submitted a problem of problem type "10," the Problem Controller would find the matching ProblemType record and then link a new ProblemStatement record into the chain. It would link it in as the last ProblemStatement record of ProblemType 10 record. Problems are dispatched by, the Problem Controller, to be solved in on a "first come, first served" basis, within the priority assigned to the problem type. The ProblemStatement records, each with its chain of "ProblemData" records, supplied all the information required to solve the problem.

Whenever a problem-solving program is finished, it returned control to the Problem Controller. The Problem Controller said to itself, "Okay, what's the highest priority problem type, for which I have a problem to be solved? Having selected it, The Problem Controller is going to load the problem solving program that supports that problem type." Then the Problem Controller would transfer control to the program, passing the identification of the problem statement record, and saying, "Go." That program would read all those problem detail records, deleting each record after its information had been absorbed, solve the problem, and then transfer control back to the Problem Controller.

HAIGH: So you're saying that in many ways IDS was the operating system as well?

BACHMAN: No! IDS is the database management system. It just turned out that an operating system has to store and retrieve information of its own. The Problem Controller was designed to use IDS to store and retrieve its control information. IDS was a database facility shared by the application programs and the Problem Controller.

[36] IDS had a special "RETRIEVE UNIQUE" record-name implementation, for the one-of-a-kind records, where the database key of the one instance of the record type was stored in the IDS Record Definition Record for that record type.

Problem Controller Control Structure, expressed as a Data Structure Diagram[37]

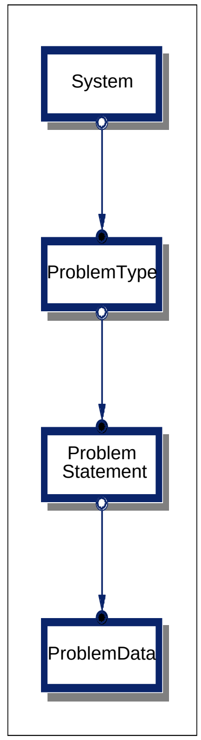

Figure 4.

I learned yesterday, November 28, 2005, that the next version the Microsoft Windows operating system will start to store the information about the file system with its file folder and file entries using SQL Server as the place to keep that information in a system database. It has taken them forty years to figure out that if a database is good enough to store application information, it is good enough to store operating system information.

HAIGH: They were extremely closely coupled. In a later generation of technology, you might find the database management system running as a process. In this case, really it's completely entangled with the operating system.

BACHMAN: I still hesitate with your characterization. I'd still say IDS was a database system, which was totally happy to store any kind of data anyone wanted to store in it. You tell IDS what your data descriptors are, and tell IDS what you want to store, retrieve, modify, or delete and IDS is happy to handle it, whether it's information that the operating system needs or information that the problem solving routines need. I don't think IDS is a part of the operating system. I just think using IDS to store and retrieve problem control information was a clever implementation of the operating system. The Aerospace Division had written their own storage mechanism for all the same kind of information, instead of sharing a storage mechanism that already existed and was going

[37] The data structure diagram is illustrated in the graphics structure created by the "Data Analyst" tool of Bachman Information System.

to exist on the machine. They did it in the classical way. The operating system stood by itself. It didn't depend on anything else; it took a machine. The Problem Controller strategy said, "We have a valuable and scarce resource (memory). Don't waste it! Use IDS and save memory." We had to get everything into a very small machine. We had to be able to run in an 8K word GE 225. The biggest GE 225 had 16K words. These were words, so multiply by three. The 16K, held 48K, six-bit bytes, i.e., 48K characters. I think I have around a million bytes of main memory on my desktop computer. It's hard today to believe that we could do anything on those early machines.

The Problem Controller operating system did a few very simple things. Because every piece of work that it dispatched was essentially a problem of some known type, for which we had a known program to load, and the Problem Controller passed control to the loaded program and passed to it the identification number of the selected problem statement record. The application could read the problem detail records from the database and do whatever it had to do. The operating system didn't know how to interpret the data records that came with the job. They were just records with a header that told the Problem Controller what kind of job type it was and therefore what program to load.

If we go back to our bill of materials problem, MIACS is processing a high-level parts explosion as part of processing a new order. We did not want the explosion of one order to immediately run all the way down to the bottom. We just want it to go to the top-level item to find out its requirements for the lower level items. New product orders are not exploded, just to be exploded. They are exploded as part of the planning of their final assembly, which placed requirements on lower level material items. When all of the product orders had been planned at their highest level, it was appropriate to ask what issues that might have placed on lower level inventories. Then the second level items were evaluated to see if they requirements could be satisfied from inventory, or whether a manufacturing order should be placed for them. If a manufacturing order was placed, then an economic order quantity was calculated and its production planned with the further explosion of parts and raw material requirements.

Every time that the production control program posted a demand on a lower level item, it passed a message to the operating system, representing another problem to be solved at that item's level, "Here's a job to be done when you get to it." It was a store and forward message system, because every job statement was packaged as a message being sent to someone or some program.

HAIGH: Returning to the question that I had asked earlier. I think I'm seeing an answer to it because I'd asked whether IDS or a piece of it would be permanently resident in the memory along with the operating system. From what you're saying, it's obvious that it must have been. Am I correct?

BACHMAN: IDS was totally resident.

HAIGH: Yes, because you could imagine one approach where, which I think some simpler systems did use where an application program is compiled, and it links copies of the input/output code from libraries into the executable. It's not that the code is permanently resident in memory.

BACHMAN: Whoa! Even in those cases the I/O code maybe permanently in memory. The loader links the application program into all of the resident code that it may require. So when that program is running, that code is there. The newer virtual memory loaders probably assemble all of an application program into virtual memory, linked to all of the subroutines that it calls, they are linked to all of the subroutines that they call, etc.

HAIGH: But then if the program stops running, the code's unloaded.

BACHMAN: Okay. That's right. Remember that the Problem Controller, and thus IDS, never stopped running. MIACS's application programs came and went, as required to process the arriving stream of messages (problem statements). It was a nonstop system. As we will see later, IDS and the Problem Control were not specialized to MIACS. Change the list of problem types and the problem solving programs that they call, and you get a new transaction-oriented system. The people at International General Electric and the Wiring Devices Department did just that. They did not run MIACS, they ran their own systems with IDS and the Problem Controller. We will talk about another example, the WEYCOS system, later.

There is a funny story about this we can come back to later, about how IDS and the Problem Controller made the machine room operators at Weyerhaeuser Lumber very nervous. At Weyerhaeuser, there was a computer running an IDS/Problem Controller system that had no conventional input/output devices, i.e., no card readers, no punches, no printers. It had a remote input/output teletypes, controlled by a GE Datanet 30. The computer just sat there day and night blinking. It made the operators very nervous. IDS/Problem Controller, was a dedicated system, that's the best way to look at it. It was dedicated to IDS and its operating system, but it wasn't dedicated to a set of order entry applications.

HAIGH: In terms of differences between this and 9PAC, from what you say, I also imagine that IDS was a system that <u>is</u> called directly by an application program. Is that correct?

BACHMAN: Yes, IDS was called both by the application programs, and by the Problem Controller. The application programs were loaded and started by the Problem Controller. The application programs would run, calling IDS from time to time, and when they were finished they would return control to the Problem Controller. Any time a new deck of problem statements was loaded into the card reader, the Problem Controller would detect this during its next dispatch cycle and read the cards and create the appropriate problem statement records and detail records for future consideration. This assured that very high priority problems would get early attention.

After the card reader was empty, the Problem Controller looked for another problem to solve. When it found one, it loaded and called the appropriate application program. It passed the identification of the problem statement record to the application program. The problem statement record and its detail records provided all of the information that the application program needed to run. This cycle continued until the Problem Controller could find no more problems to solve above the priority level set for the print report problem type.

Printing to the printer was treated in the same manner as other problem types. There were several problem types for printing, with different priority levels. When a printer program was called, the problem statement was dispatched to the printer program portion of the Problem Controller. When the printer program finished with its report, it returned control to the Problem Controller to begin the next dispatch cycle.

The card reader control program was a portion of the Problem Controller. The card reader program of the Problem Controller would call IDS to support its storage of the problem statement records and the detail records.

If there were no problems of any problem type to dispatch, the Problem Control would start the next dispatch cycle, with the opportunity to discover something to read in the card reader.

We had to watch application program size very carefully so every application program could get loaded into the 8K GE 225. If we look at the memory usage, we had a set of deadlines so that on our 8K machine, IDS, and the Problem Control could not exceed 4K. That's it, 4,096 words—beyond that, we should do something, squeeze something.

As an example of the squeezing that we had to do, let's talk about saving the address so that a subroutine can return to the point of its invocation after completing it work. Given the GE 200 series architecture, each subroutine was responsible for saving the address to which it was supposed to return. There were 58 subroutines in IDS, at the point relevant to this subject. The 58 subroutines therefore needed 58 words of memory to individually save their return address. Stan Williams and I analyzed the situation and found that no IDS subroutine was more than 12 IDS subroutine calls down from the application programs. So we reserved 12 words for the storage of return addresses and calculated how deep each subroutine was and thus allocated to it, one of the 12 storage places to save its return address. This was important. We had already trimmed everything we could out of the coding.

All of the memory above 4,096 was reserved for paging buffers. The paging buffers were used to load both program pages and virtual memory pages. If you load a program into the upper 4K, the binary code of the application program would be allocated one, two, three, four, five, six contiguous buffers beginning at the 4K boundary. In an 8K machine you could load programs up to the size that left only one 512 word buffer, for data paging, to swap in and out. Of course, the system didn't run very fast with only one data buffer. In fact, people found that it was a good buy to buy the 16K version of the GE 225, because IDS could have more buffers in which to swap data pages, back and forth to the disk. The programs and the data shared the same buffer space.

We were not smart enough to know how to swap program pages in and out during the execution of a single application program. We had to learn how to partition an application so one portion could do its part of a task and then send a message (create a problem statement) that would pass the baton to the second portion of that application, so that it could continue doing a task that was too big to fit into the machine as one program.

HAIGH: Right. Am I right to think that with IDS the application programmers would write their code, but then every time they need to create, retrieve, update, or delete a database record they would insert an IDS subroutine call?

BACHMAN: Well, essentially what they do in their program is that they just call a subroutine. Well, there are two answers to your questions. Let me stop here for a moment. Let's go back in history. IDS had two implementations. The first IDS implementation expected a person to write an application program in GECOM. They would insert into their GECOM program the IDS data manipulation statements, wherever they were needed. It looked like an ordinary GECOM program. except there was some STORE[38], GET, MODIFY, and DELETE statements[39] include in it. That source program went through the IDS preprocessor. That preprocessor looked at the data descriptions for the database and at the source program and translated the IDS statements, creating a new source program that had the original GECOM statements plus some new GECOM and GMAP statements interspersed. GECOM was designed to accept "ENTER GMAP" statements. That ENTER GMAP would be followed by the GMAP statements the preprocessor[40] had generated. Then you would have a program that went to GECOM compiler, which is a mixture of GECOM and GMAP statements. The GECOM compiler translated all of the GECOM statements into GMAP statements which were assembled into machine code.

In the second implementation of IDS, the programmers were writing their programs in GMAP and inserted subroutine calls into their GMAP programs that called the IDS subroutines that carried out the store, get, modify and delete functions.

HAIGH: And that's what it means in some of the IDS specifications when it talks about it being "a set of macro instructions," so the pre-compiler is expanding these macro instructions into a sequence of assembler code? There was one set of subroutines at the heart of IDS that were not called directly. They were called the Block Controller and they implemented the virtual memory portion of IDS controlling all of the paging buggers and doing the physical reads and writes with the disk file.

BACHMAN: Yes, into inline assembly language statements. The inline machine instructions were to execute the function, not interpretively, but as explicit code. So the functions, which were executed by the subroutines in the interpreted version of IDS, were previously expended as inline code in a record type specific form. If I wanted to store a Payroll record, I'd get a different set of code generated than if I wanted to store a TimeCard record. Different code was inserted into the middle of the application program that synchronized with the logical flow of the program.

HAIGH: So the pre-compiler is looking at the data definitions statements?

BACHMAN: Yes. The data definitions were stored in an IDS database. The first two running IDS programs were the DDL Processor and the DML Processor. They are the ones that first read IDS data descriptions language statements and then stored them in an IDS database, and second read the IDS data manipulation language statements and translated them into a combination of inline GECOM and GMAP statements. This was

---

[38] STORE, GET, MODIFY, and DELETE were the original names of the Create, Retrieve, Update, and Delete commands.

[39] All program coding in those days appeared in upper case letters, because that is all the printers could print!

[40] The IDS preprocess actually generated a mixture of GECOM and GMAP statements, using GMAP only when there was not a GECOM statement to meet the need.

the first successful application of IDS. IDS didn't care what kind of records were being processed, data description records, or payroll records; they were just records. When these first two programs that were written, and I wrote them using the GECOM language, with embedded IDS data manipulation language statements. Then to get these first programs running, I manually expanded their embedded IDS statements into a mixture of GECOM and GMAP code. Later these two programs were reprocessed by the DML Processor, to regenerate the GECOM and GMAP code, previously generated by hand. The DML Processor generated the replacement for my handwritten inline code, because it could look at its own data descriptions (meta data descriptions) to generate required inline code. The DDL processor and the DML processor were the heart of the inline coded version of IDS.

Those two programs were used to prove one of our key issues. We ran a test. It was run by a GE organization called IAO, the Internal Automation Operation. It was GE's big systems group. It was located in Schenectady, New York and was the body of people that GE would send out to their various departments to give help in the development of information systems. They sold their services both inside and outside GE. They're a very competent group of people. One of the most important people in our story, was a man named Bill Helgeson, who could be a bear. A very bright, manufacturing guy. He was very negative about what we were doing. Just incredibly negative. When we went to Schenectady to make a presentation to the IAO organization, and he just ate us out, all day. I was talking to Paul Hess, another one of the guys in IAO organization, and I said, "What's wrong with Bill?" Paul responded, "When you get him really upset, he really gives someone trouble in a meeting." (Bill Helgeson died in December, 2001, in Gig Harbor, Washington.)

So in the aftermath of this meeting, Bill said, "Okay, let's do a test." IAO had just installed a manufacturing control system at the GE Laminated Products Department, someplace in Ohio[41], I believe, and they were very proud of it. They put a bill of material processing system together; it was really the best thing they could do. They said, "Well, if your system, which we see is much easier to install, runs at least half as fast as our system does, we'll use IDS for the rest of our systems to be installed." The one they had developed for Laminated Products was a specialized, one of a kind system. No one could really afford that approach.

It took Stan Williams and me two weeks to copy the bill of materials logic of their system, including their data. Stan went out to Ohio with Bill Helgeson. They ran the tests on the Laminated Products Department machine. We ran twice as fast as they did. Bill Helgeson was flabbergasted. "How can you run twice as fast? We did the best we could do." I said, "Well, there's some things you didn't do." For instance, they didn't use a virtual memory. For every record they wanted, they had to go out to the disk and get it, one at a time. We could bring a record in and process three or four records in the same virtual memory pages, we already had in memory. It was because IDS had assembled from a number of valid, important technologies into a workable combination. Even

[41] 01/25/2006. Paul Kosct told me this morning that the actual site was the GE Radio Receiver Department. He said that he should know, because he had written the original parts explosion that was in the contest with the IDS implemented version of the same function. He was with IAO and located in Utica, NY at that time.

though we had a general purpose system, we could outrun their special purpose system, because they couldn't afford to combine all those technologies into a one of a kind system.

HAIGH: You started that answer by saying that in the first version of IDS used high level language statements, which were imbedded in the GECOM application programs, and which were translated by a preprocessor into GECOM and GMAP statements. What's the answer for the next version of IDS?

BACHMAN: The thing that drove us to the second version was Bill Helgeson saying, "I'll sign up for what you're doing, but you've got to make it run in an 8Kword configuration of the GE 200 series computers." The inline coded version required a 16K word configuration of the computer to run the DDL and DML Processors. Bill said, "People can't afford a 16K machine!" You have to understand this was in June, of 1962. The price difference between the 8K and the 16K configurations was important in those days. It may have be ten to twenty thousand dollars difference in the prices, for that little bit of memory. A good engineer, just out of school, was earning between six and eight thousand dollars a year,

When we looked at the how do to get an 8K system out of a 16K elephant, the only way we could do it, was take all this inline expanded code and recreate it as a set of interpretative subroutines, which we could use over and over again. They were data descriptor driven.

HAIGH: Because when you expand the code, you finish up with multiple copies of the same logic in the program. Is that why it makes it bigger?

BACHMAN:. If the inline coded version was storing ten different kinds of records, it would generate ten expansions. If it was storing the same type of record in two different places, it would generate the expansion as a subroutine, and call on that single expansion. It was clever enough to do that.

HAIGH: So the difference is that you can do the expansion at run time instead of at compile time.

BACHMAN: That's correct. I just groaned all over the place. I didn't want to do it, and I thought it'd be much slower, because interpretive systems are always slower. But getting IAO's support was very important to where we were, so I set out and rebuilt the system so that we had a set of subroutines that did the same function that you called the STORE function. In fact, there weren't many subroutines. There were STORE, MODIFY, DELETE, GET UNIQUE[42], GET NEXT, GET PRIOR, and GET OWNER. Essentially what you did, you'd call a subroutine and passed it the address of a descriptor in the data descriptions, in the metadata for that record type you're manipulating. Then it could look at the metadata and say okay, if I store this kind of record then I have to do this kind of thing. If I store that kind of record, I do that sort of thing. IDS was rewritten in that form.

As it turns out, the interpretive version was twice as fast as the original version. Interestingly, it was twice as fast, not because it was really running faster, but because it

[42] The GET UNIQUE statement was a statement that retrieved a record based on the values presented for its primary key field(s). The other GET statements were owner/member set oriented.

had more paging buffers and the virtual memory was more efficient. I could keep more virtual memory pages in real memory at a time. Therefore, the program could jump around within ten different pages. If virtual memory buffers could only hold five pages, then some pages would have to be written back to the disk and be reread almost immediately. There was more “thrashing.” There was always some thrashing, but if the number of buffers was large enough to hold a “working set” of pages, then this thrashing was minimized. The 4096 words of memory, above IDS and the Problem Controller could hold eight paging buffers in the 8K memory machine. In the 16K machine, there were 24 paging buffers above the IDS/Problem Controller 4096 word limit.

HAIGH: Thrashing less.

BACHMAN: Thrashing! This reduced thrashing was enough so that the programs ran faster. Remember, we are in a uniprogramming world here. So when I’m reading and writing to the disk, the computer is waiting. If we were in a multiprogramming world the computer could be doing something else while a program waited for an I/O operation to complete. Our uni-programming compters could only wait. So the interpretive behavior doesn’t mean I’m really solving problems faster, I’m just not wasting time waiting for the disk, whereas in a multi-programming computer it would have been probably slower, because of interpretation. The computer can do something else while it’s waiting for the disk.

HAIGH: So the GE 225 didn’t support multi-programming?

BACHMAN: No. In fact, none of the computers did at that time.

HAIGH: I believe the Burroughs 5000 series claimed to have been the first, and might have been appearing around that time.

BACHMAN: It probably did, yes, early. Nothing in the GE family did. The GE 215, 225, 235 were all the same machine, running at different clock rates.

HAIGH: So, for example, did it have any kind of capability to stop the application program from just going ahead and trashing the memory that IDS was sitting in?

BACHMAN: We didn’t have such problems. An application program did not have the ability to address anything outside its program boundary, unless it was using an index register. IDS didn’t trash memories. But, it must have crashed somewhere, because one of the things I heard later from Paul Kocst, at Weyerhaeuser, was that they had a program called “the Mickey Mouse Fixer.” What’s the Mickey Mouse Fixer? It’s a program that runs around looking for records with the “d-switch” set. Mickey Mouse was chasing Minnie Mouse, as they said.The technical reason for this program is that these were records that had been logically deleted, but not physically deleted. They had no prior pointer for some chain and could not be physically deleted until they had been removed from each chain where they were a detail/member. This is what would be called “garbage collection” today. The Mickey Mouse Fixer would be run at off peak times to clear out deleted records to make room for more. Weyerhaeuser was the only site that I have heard of that took this approach.

We had an early IDS user at the GE Wiring Devices Department. Working with IAO, they had IDS up and running very quickly on an new order entry system, and they had it in production. After a while it began to run slowly. It ran slower, and slower, and slower, and finally came to a halt. What went wrong? Well, they hadn't written the program yet, to delete the old order records. They just filled the disk up until there was no room to store one more new order record, so IDS kept looping, looking for available space on the disk. They quickly wrote the program that purged orders that had already been processed, and they were back in business.

So there were things that happened because people didn't understand the way the that IDS operated, in some cases. There used to be what we called "IDS horror stories," typically because they were situations where the data had skewed distributions. One of the situations, I think the first IDS horror story, was at International General Electric. They had an order processing system which processed GE's overseas orders. The orders came in on punch cards, and they were stored in IDS. All of a sudden, nothing was happening. What they found out was that every once in a while they would receive an order for a diesel electric locomotive, with about 4,000 spare parts. The way IGE had described the data to the system, they said whenever you get a new order, store the order. Then take the first part and store and link it to the order record. Take the second part of it, find the order, find the first spare part. Then store the second spare part and connect the second part to the first part. Then the data for the third spare part comes in. Then find the first part, find the second part and then, find the third part. Then find the first, the second, the third, the fourth. Then find the first, the second, the third, the fourth, the fifth, etc. We said, "Well, all you have to do is redefine the order record, as having a "prior" pointer, a pointer to the last member" record. If you're storing something last…." They said, "they're supposed to be stored in item number sequence." Well, the cards were already coming to the computer in item number sequence. Just store each successive spare part last in the chain. All of a sudden, what was taking minutes and minutes and minutes to store one order went *pssst*, all done, next one. Because in those days, optimizations were critical[43]. You could ask IDS to do stupid things, or you could ask IDS to do unreasonable things. IDS would do whatever you told it to do. It was a perfect robot in that sense. There was a way to do things smart, and you could also do things, not so smart. There were consequences.

The similar thing happened to Mack Truck, where they had a backorder system running on IDS. They said, on the average, we have one backorder for each inventory item. That's easy so we will build a chain of backorder items. One chain. Well, it turns out only one item out of a hundred had backorders. Ninety nine percent of these chains were empty, and then one out of a hundred had a list of a hundred or more backorders. They were doing the same thing: ordering the backorder chain on a first come, first served basis. The first backorder got the first chance at new material; the second got the second chance, the third the third chance, etc. This was the correct business rule. This was easily corrected by changing the item record's chain definition to include a last member pointer. Each new backorder record could be immediately inserted at the end of the list. Previously, IDS had to search down the chain until it found the last backorder record in order to link in the new backorder record.

[43] I have just been reading (1/24/2006) the Oracle Data Warehousing Guide. It is full of information about how to optimize table loading and accessing.

HAIGH: You had said that even in the first version of IDS where the expansion into assembly language instructions to retrieve the data was being done at the compile time, that there was IDS that was always loaded in memory. What was that piece?

BACHMAN: In that case, it would not be. IDS had disappeared.

HAIGH: So in the first version there was no IDS sitting in memory permanently because all the IDS routines had been compiled into the program.

BACHMAN: Right.

HAIGH: The second version--

BACHMAN: Let's stop a moment. IDS had two major parts: the data type sensitive part and the page turning part called the Block Controller. The block controller was resident in memory, all the time in both versions.

HAIGH: And that was doing the virtual memory?

BACHMAN: That was a virtual memory system, yes.

HAIGH: That was something that I think in later generations of systems, people would think of it being as part of the operating system.

BACHMAN: I guess it depends on what you think is important. If you're an operating system person, it's part of the operating system. If you're a database person-- you know, what are you doing 95% of the time? Well, I'm doing database processing. Therefore it's mine.

Looking at the question in another manner. Most of today's computers have virtual memory facilities built into the hardware that respond to instructions that attempt to access data with a virtual memory address. The computer looks for the page with the requested virtual memory in the hardware's virtual memory buffers. That action is backed up by the operating system, which will fetch the required virtual memory page from the disc, when it is not in real memory. However, the relational database management systems running on those virtual memory computers, do not use the hardware virtual memory facilities, they implement their own virtual memory, in their own buffers, just like we did. No one has been able to make the hardware system virtual memory work satisfactorily in conjunction with a database system.

HAIGH: That's what the chunk that was loaded in memory did even in the first version?

BACHMAN: Even the first version of IDS, yes.

HAIGH: Then the second version, that's joined with the piece that's actually doing the expansion to do the data access. I've got a document here, "Introduction to Integrated Data Store", and it deals with these operations you've been mentioning in terms of put, get, modify, delete, etc. Those would be used by the application programs.

How about data definition? We have in front of us a document "The Integrated Data Store Functional Specifications", January 26th, 1962.

BACHMAN: I've got the 16th. Here's the January 26th edition. A later edition.

HAIGH: Let's take the 26th because that's in the archive. In the CBI archive, this is currently CBI 125, box 1, folder 23.

BACHMAN: Before I attempt to answer your question, let's remember we are talking about a time when we were talking about the "inline coded" version of IDS. At this time there would be an IDS program, which we can call the "data description language (DDL) processor," and another IDS program, which we can call the "data manipulation language (DML) processor."

Now what question are we heading toward here? Let's see if we have an answer. On page six you have the list of record files and the operation: DEFINE record name, PUT record name, so on.

HAIGH: These definitions, were these being handled the same way as the instructions to actually read and write the data, then, as macros that would be called from the GECOM programming language?

BACHMAN: They are actually new IDS language statements we inserted into a GECOM program. They were used in the same manner, whether to write an application or to write the preprocessor. The IDS language processors were a complete boot strapping operation, as we discussed earlier.

HAIGH: So in this version of the draft, at the bottom of the page there's read, write, load. I'm seeing an instruction called "define". Another one called "modify". That's number seven. One called "link", number five. Those, essentially, are what would later be called DDL statements?

BACHMAN: No, these are all really DML statements, given in a way that is characterized by its preprocessor application, at that moment. Not DDL statements.

HAIGH: So there's no data definition language?

BACHMAN: At this point, there's a Data Definition Language, but there's nothing to read it. What we're looking at is a DDL processor program which is going to really read and process the DDL statements. It's a manipulator of data, more specifically, metadata. These are statements which are going to be written and used in the data description process. They captures the record formats. Some formats are in here showing what each input record, the DDL, looks like. In the back of the document are some formats showing different record layouts. They show how to understand what's going on here. There was a "RDR," the Record Definition Record, the "FDR," the Field Definition Record, the "ODR," the Owner Definition Record, the "MDR," the Owner Definition Record, and the "CDR," the Control Definition Record.

IDS Meta Structure – Data Structure Diagram

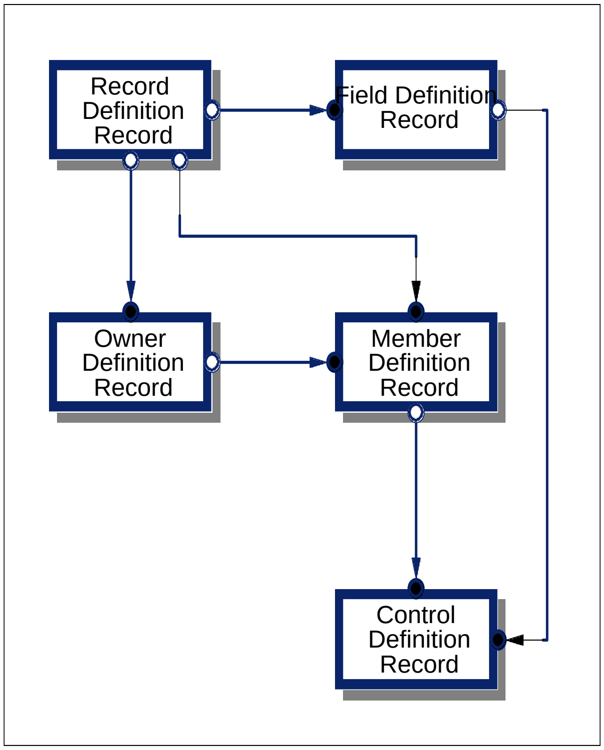

Figure 5.

The "Record Definition Record" record was the owner of three different Owner-Member sets. They individually provide access from each Record Definition Record record to a set of "Field Definition Record" records, a set of "Owner Definition Record" records, and a set of "Member Definition Record" records. Each Field Definition Record record is the owner of a set of "Control Definition Record" records. Each Owner Definition Record record is the owner of a set of Member Definition Record records. Each Member Definition Record records is the owner of a set of Control Definition Record records.

In the inline coded version of IDS, these records were stored as ordinary IDS records in the database where they were processed with regular IDS commands to direct the generation of the inline code for the DML statements. The chains linking these records, were implemented by pointers based on each record's virtual memory addresses (page number + line number).

In the interpretative coded version of IDS, these records were stored within the application program as an in-memory copy of the same information. The chains between these records were implemented by real memory addresses. Retrieving the "next" record of a chain was simply the matter of reloading the index register for the current definition record with a new real memory address. These addresses were assigned by the GMAP assembler.

HAIGH: For example, on page seven, is the define macro, which it says is “designed to permit the programmer to insert, delete, or modify various elements of the record description.” How is that not a data definition?

BACHMAN: Okay, let’s take a moment and let me reread what the text says, because I don’t trust my memory, but let me read it and then refresh my memory.

HAIGH: It says that “the define instruction will allow the programmer to insert, delete, or modify the various elements of the record description provided for by the data division of GECOM.” Then it says a little bit later “that it will cause all of the records of the type that currently exist in the Integrated Data Store to be reformatted.” Would you agree that that’s a data definition instruction?

BACHMAN: We have a problem here. It’s a data manipulation statement that manipulates data descriptions. There’s a subtle difference.

HAIGH: What’s the difference?

BACHMAN: Well, first of all, data description languages are not active; they’re passive. This is an active statement—this is going to do something.

HAIGH: You mean it’s going to reformat existing records?

BACHMAN: Yes, and it’s going to do it, based on the cards that followed. The data description follows this card, as it says. This is calling the data description processor. “Definition data that the system, by the following defined statement by the necessary data definition cards.” That’s where the data description language comes in, in those cards.

HAIGH: Okay, so data definition is done by punching cards?

BACHMAN: Yes, that’s why you punch cards up in a certain format. They’re going to be read by the DDL processor.

HAIGH: Is it the case that the way that you’ll get the definitions into the system would be by punching some cards, and then running a program that holds this “define” macro instruction?

BACHMAN: Yes. This will put the descriptions into an IDS database, which are data description records. The IDS DML preprocessor program, which translates IDS statements embedded in GECOM, will go through the GECOM program looking for IDS statements. If it finds one, it’ll look at this data description data (meta data) and say, “How do I do that?” and the expands the DML statement into a sequence of GECOM and GMAP statements. GECOM is perfectly able to do some of the things translation requires. Other things are translated into GMAP. You generate the sequence appropriate to that action that the user and their data manipulation statement is implying.

During the translation process, the IDS DML preprocessor program will generate record and field definitions, for the IDS records, and insert them into the GECOM Data Division. These provide a working storage area for each IDS record. This is the place where the application programmer and IDS exchange information. An IDS RETRIEVE statement leaves the resulting information in that record type’s working storage area. An application programmer builds up the information, required for a new record, in that record type’s working storage area before executing a STORE statement.

HAIGH: Right, so to retrieve the right records, obviously IDS is having to access the stored metadata, but the way you get the metadata in there is by punching it onto cards and then triggering this special define macro.

BACHMAN: Right.

HAIGH: That clarifies that. I'm really getting a sense for how this worked. Presumably at some point, probably sooner rather than later, this reliance on GECOM would have become a problem.

BACHMAN: Not in the GE 225 computer lifetime. It's only when we went to the GE 400 and 600 that the IDS version they built was COBOL oriented and not GECOM oriented. There was no COBOL on the GE 225.

Please remember, we have been talking about the early, inline coded version of IDS. It was in early 1962. The requirement for GECOM vanished when IDS was reimplemented, in the interpretive version. The data descriptions, which were used by the IDS DML preprocessor, were inserted into the interpretative version of IDS in another manner. We have not talked about that. I believe that all of the MIACS application programs were written in GMAP. They were written after the shift to the interpretative version of IDS.

HAIGH: We'll have to talk a bit more systematically about subsequent versions, but do you know at what point the 400 and 600 versions became available?

BACHMAN: Roughly 1966-67.

HAIGH: Now just to check a few other things, did it have any kind of recovery capability? If the system went down part way through an operation that left it in an inconsistent state, did it have any way of automatically recovering?

BACHMAN: At the time, when I finished my work on IDS, it did not. International General Electric (IGE) very quickly added a recovery and restart system. That did two things. It meant whenever IDS wrote a new version of a page, to disk, the IDS R&R system wrote a copy of that new version to a backup tape. Actually, it wrote two copies: it wrote the prior version of the page and the new version. Old page images and new page images. "Before" and "after" images. If the disk were to fail, the latest "after" image of each page on the back up tape was what you used to recover the database. If the operating system or the program fails, for whatever reason, then you want to go back to the "before" images of all the pages, that is synchronized with the start of the application program, to recover the database.

Clark Archer, who was the manager of the data processing group at IGE, and Charles (Bob) Blose, who was their IDS programmer, put this R&R system together. Bob was fresh out of college. They added the recovery and restart. That was done in late 1964. They found they could not safely support the business without it. The R&R system was just essential to the integrity of the data.

HAIGH: Am I right to think that the only way to get at the data in the database would be to write a program in GECOM?

BACHMAN: Yes, initially. However, when we switched to the interpretive version, most of the application programs were written in GMAP.

HAIGH: So there was no separate ad hoc query capability?

BACHMAN: No. Absolutely not.

HAIGH: Just to get a sense for how fast things were working in these days, assuming you had written a simple program that would retrieve your data, how long would it take for the system to compile it, execute it, and return some records?

BACHMAN: I only can give you limited experience. The IDS DML preprocessor program, which was written in GECOM, took an hour to recompile. That was a big program, but it wasn't the biggest program you could put into a GE 225. It took an hour, so that it was a very different world. We had instruction times of 14 milliseconds—not 14 microseconds or 1 gigasecond. Things are running a thousand times faster or 10,000 times faster now. Of course, that was tape swapping back and forth, the compiler used tapes heavily. We couldn't compile them any system that GE had at that time in New York City. We had to go to Schenectady to compile and test our programs.

Once any GECOM program was compiled, it could be loaded and run as many times as you wanted. GECOM was not a "compile and go" system.

HAIGH: You just alluded to a process by which IDS moved to International General Electric.

BACHMAN: IGE.

HAIGH: How did that work?

BACHMAN: They were down town from us in New York City. They were downtown near Penn Station; we were at the GE headquarters building at 570 Lexington Avenue, North of Grand Central Station. We took the subway down and worked with them. The whole IDS system fit into two boxes of punch cards (2000 cards), so it wasn't a big deal to move the program down to their building. I started working for IGE with Clark Karcher and Bob Blose. Bob was the programmer on the project to look at their order processing system. Remember, we are now in 1964 and using the interpretative version of IDS.

HAIGH: So you moved over there with the code?

BACHMAN: I personally carried the two boxes of GMAP assembly language code down with me.

HAIGH: And then you dropped it off?

BACHMAN: I spent a few days with them off and on. They would be stumped by something and I would go down and help to sort it out.

HAIGH: What did they do to it?

BACHMAN: They did two things. They added the tape backup system we're talking about. It's captured "before" and "after" images. Those were copies of pages before and after they were updated. And we did one more thing, which helped performance.

I talked to you about the situation at the GE Wiring Devices Department, where they didn't get around to writing to program to delete the order records from the database, after the orders were shipped. Their order processing system got slower and slower looking for space for records, as they filled up the virtual memory. IGE introduced new system-oriented, “inventory” pages, on the disk. There was 1 inventory page for approximately every 500 data pages. Each inventory page would maintain information about how full each data page was. With the new page inventory system, IDS didn’t have to search, page to page, looking for available space. IDS could look at its inventory pages and determine which page had space. They encoded the available space in each page so that a range of data pages could be represented on a single inventory page. As I remember it, only a single character was assigned to represent the available space for an entire data page. I think that is coded in scheme that indicated: zero meant “empty;” one meant “less than one half full,” two meant “less than three quarters full,” three meant “less than seven eighths full,” etc. When the targeted[44] page did not have enough space for the next record to be stored, IDS didn’t have to look page, after page, after page to find the first one with available space. The page inventory indicated a page would have the required amount of available space.

HAIGH: So the International General Electric organization made those two main improvements.

BACHMAN: They made a third one that comes to mind now. The idea for the third one came from Bill Helgeson. It was called “fast” IDS. In the original IDS, as I had designed it, each page had a page number. Each page had been assigned 64 line numbers. The virtual memory address of a IDS record was the concatenation of its page number and its line number within its page. The virtual memory address of an IDS record was permanent and never changed. The virtual memory address of a deleted record was available for reuses. Each page had a control record at the top. It was assigned line number zero. The control record had a record counter indicating the number of data records in the page, an available space counter, and a flag called the “d-switch[45].” The data records in the page were maintained in line number sequence and stored consecutively.

If IDS was looking for record, with line number 17, IDS would bring the page into one of the buffers in memory and then read from record to record, going down the page, until it found the record with line number 17, or found that it did not exist. The record header of each data record recorded how long it was. When IDS deleted a record in the middle of a page, it would go into the page and close the gap, where the deleted record had been recorded. When a new data record was created, it was inserted into the page at a location based on the first unassigned line number. Storing a new record or deleting an existing record in a page could required moving a lot of records around.

Bill Helgeson said, “Well, we can do better than that.” IGE put a pointer array in the control record at the top of each page. This pointer array had an element for each of the

[44] Several of the STORE record options attempted to store a new record in the same page as another existing record so that the retrieval would be faster. Faster, because the retrieval of the one record’s page would also bring the rest of the record in the page, into the computer from the disk. Storing a second record close to the first record, even if they were not in the same page was helpful.

[45] The “d-switch” was used to signal that an IDS record was logically deleted, but had not yet been physically deleted. When a record was a member of a set and its context in that set was not immediately known, it was quicker to set the d-switch than to run around the entire chain to establish it context so that it could be correctly deleted. When that chain was next accessed, the records with their d-switch set were transparently removed from the owner/member set and physically removed from the database.

63 potential data record. That element would hold the address within the page where its record could be located. It held a zero value, if no such record existed. IDS wanted record 17, it would look in the 17th element in the array at the top of the page, and say, “oh, you’re starting in word 121.” IDS could immediately jump to that address. If IDS deleted a record, it did not need to physically get rid of it at that time, It would wait until it needed room in that page for a new record. Fast IDS would keep track of these things. It made page management more efficient.

HAIGH: And you said that was called Fast IDS?

BACHMAN: Fast IDS, yes.

HAIGH: Did that become the standard new version?

BACHMAN: Yes.

HAIGH: What was International General Electric? Why did you give IDS to them instead of-- Did they have some kind of official responsibility for productizing it, or did they just want to use it themselves?

BACHMAN: They just wanted to use it. We didn’t productize it. It was just another GE department that was ready to use it, and needed it. They were ready before the manufacturing system (MIACS) was ready.

HAIGH: So they were just another internal user, but they made improvements and shared them with the IDS community that was growing.

BACHMAN: They were ready; actually, the Wiring Devices Department was ready first.

International General Electric also made a fourth change. They removed the “data independence” facility, because they thought it slowed down the processing. IDS was built using data descriptors for the fields in the IDS records and the fields in the working storage area, where IDS exchanged information with the application programs. When data values were being moved between the working storage area and the paging buffers, these descriptors were consulted and the data was converted if the field descriptions showed that it was required. This was mainly used to convert numerical values between the binary and binary coded decimal forms. All calculations were performed in binary. All input and output were in binary coded decimal.

The GE225 community, to my knowledge, did not adopt this fourth change. Subsequent tests held later showed that IDS was spending more time in the data independence code that anywhere else.

HAIGH: How did IDS become productized?

BACHMAN: Well, the GE 225 version never did get productized. It was not until the GE 400 and the GE 600 versions that actually the GE computer department took formal responsibility for the products.

HAIGH: Was that your personal responsibility within the GE computer department?

BACHMAN: No. That was one of my responsibilities. I wasn’t responsible for the project. I was there as a technical lead.

HAIGH: All right, so we'll talk about that in a minute then with the GE computer department. Oh, I just happen to have an email from someone with a question for you that's appropriate here. The question is: Did IDS use a mixture of hashing algorithm and chain pointers? I think from what you said my impression would be yes.

BACHMAN: Yes. It used both. All of the chains/lists/relationships, whatever you might wish to call them were implemented with linked lists. In a linked list the owner record pointed[46] to the first member record. The first member record pointed to the second member record, etc. The last member record pointed back to the owner record. The IDS implementation offered to each linked list several implementation options. 1. Each record could also have a "prior" record pointer. 2. Each member record could have an "owner" pointer. 3. The owner record could have a "last" member record pointer.

The hashing algorithm was used for "calculated" records. Being calculated was one of several storage algorithms offered by IDS. When a record of a record type that had been declared as calculated was stored, several things happened. First, the value of the identifier/primary key field was submitted to a randomizing routine. The result of that routine was to generate a page number that would be the home page for the new record. If there were space available in the home page for the new record, it would be stored in that page. If that page did not have enough space, a search would be made for the first/closest page that did have space, and the record was stored there. Every page header record was implicitly declared the master record of its CALC chain. Every calculated record was implicitly declared a detail record of the CALC chain, of its home page. When a new calculated record was stored, it would be linked into the calc chain of its home page. Therefore, it could be easily found, even if it encountered a page overflow situation and could not be physically stored in its home page.

There were several versions of the GET[47] command: GET UNIQUE of record-name[48], GET DIRECT[49], GET NEXT of chain-name[50], GET PRIOR of chain-name[51], GET OWNER of chain-name[52], GET CURRENT[53], etc.

---

[46] The pointer value was the database key (virtual memory address) of the targeted record.
[47] The intent of all of the versions of the RETRIEVE command is the retrieval of a single record. If such a record exists, its data values will be returned to the program, via the fields of the working storage area. That record becomes the current record of the program, the current record of its record-type, and the current record of each of the chain-types in which it participates as either a master record or a detail record. If such a record does not exists, an error code is returned and the current record of program remains unchanged.
[48] The RETRIEVE UNIQUE record-name command searches the database to see if there is a record of the designated record-type whose primary key(s) matches the value(s) established in the working storage area for the designated record-type. If such a record exists, it is retrieved and its data values are returned to the program, via the fields of the working storage area. That record becomes the current record of the program, the current record of its record-type, and the current record of each of the chain-types in which it participates as either a master record or a detail record. If such a record does not exists, an error code is returned and the current record of program remains unchanged.
[49] The RETRIEVE DIRECT command is based on the database key value, provided by the program, and retrieves the record that matches that database key.
[50] The RETRIEVE NEXT of chain-name is based on the current record of the specified chain-type and retrieves the next record in the named chain-type, relative to the current record. If the current record of the chain-type is the last member record, the owner record is returned.
[51] The RETRIEVE PRIOR of chain-name command is based on the current record of the specified chain-type and retrieves the record that is immediately prior in the sequence of the named chain-type, relative to

The GET UNIQUE command would repeat the randomization process (hashing algorithm), using the identifier value(s) again to regenerate a page number. IDS would access the home page for the requested record. Starting from the page header record, which was the master record of the CALC chain, IDS would search the CALC chain of the home page[54] to determine whether there was a record whose field(s) matched the specified identifier value(s). If so, IDS would return that records data values to the application program, else an error code would be returned.

The GET UNIQUE statement was IDS's content retrieval mechanism. If you wanted line item 14 on purchase order 7486, IDS would use hash addressing to first access the Purchase Order record. Then it would use the chain to go down, Line Item record by Line Item record, until a Line Item record with the value of 14 was found[55]. IDS users used content addressing very heavily. Not only single key identifiers, but also multiple key identifiers, as in the example above.

HAIGH: Would you say that your managers within the Advanced Materials Service were pleased with the results of the project?

BACHMAN: I think so, yes. Some people are pleased with what we were doing, so it must have been all right. Not that Ford Dickey, who was the critical manager, really knew what was going on. Hal Miller was the VP of Manufacturing Service, and Hal was very, very pleased with it. He was extremely knowledgeable and a major influence on the GE Computer Department.

Lately, I have found this notebook, in a box in the corner[56]. We're looking at copies of the presentation made at the breaking point between High Voltage Switchgear and Low Voltage Switchgear. These are the originals, from which the flip charts were made. This was the big October 1962 meeting. We had all the vice presidents from Schenectady, my Vice President, Hal Miller, Bill Bryant, Ford Dickey, and several vice presidents from the Switchgear Division. This is the meeting where the general manager of the High Voltage Switchgear, Don Beaman said he didn't really want to go forward. When we keep pushing, he pushed back. He really didn't want to sign up for it. Shortly after that, we switched our project affiliation to the Low Voltage Switchgear Department.

It was hard to explain what was going on at the meeting. It's not clear that these people knew what a disk drive really was, or what it was for. These pages are the data descriptions for different code I wrote. Let me show you a picture back in the back here. This is the same drawing that's on the Computer History Museum poster, from my April

---

the current record of the chain-type. If the current record of the chain-type is the first member record, the owner record is returned.

52 The RETRIEVE OWNER of chain-name command is based on the current record of the specified chain-type and retrieves the owner record based on the current record of the chain-type.

53 The RETRIEVE CURRENT command re-retrieves the current record of the designated record-type.

54 In the case of a page overflow, the calc chain of the home page would overflow onto other pages.

55 This is assuming that the Purchase Order record had been designated as a content-retrievable record, and that the Line Item record had been designated as being content-retrievable, subordinated to its Purchase Order record.

56 I was rummaging through a box of early IDS papers that I had held back when I sent most of my archives to the Charles Babbage Institute in 1996. I need these to help write the Book on IDS that I have been planning.

16, 2002 lecture. It looks like a roll-top desk with a million pigeonholes that you could reach into and get something out. The idea was that you could access one record (one page) in a sixth of a second. That's how fast the disk accesses were then. The next record, in that pigeonhole, is accessed in one thousandth of a second, because it was stored in the same virtual memory page. You achieved better performance by relying on this kind of clustering.

HAIGH: How did you come to move to the General Electric Computer Department in 1964?

BACHMAN: I guess, two reasons. One was that IDS was running well; and secondly, Connie was very unhappy living in Stamford, Connecticut. The actual move time was controlled, by getting out of Stamford, Connecticut before school started in the fall. The next opportunity might be a year later, and so I negotiated a transfer out to Phoenix, without having a clear vision of what I was going to do when I got there. It was for personal reasons as much as for business reasons. IDS had achieved sufficient success, where it had been used, and in people's review of it, to think it was going to become a standard product on GE's new 400 and 600 computers. It was the right timing from that point of view, too.

HAIGH: What kind of state was the Computer Department in when you joined it?

BACHMAN: "What kind of state?" Well, the GE Computer Department has an interesting history. Its glory days were in the creation of the big banking system, the ERMA system that they built for Bank of America. It was a first big Demand Deposit Accounting system, on many accounts; whether it was sorting checks, or handling check transactions. It achieved the first in many areas. It was the achievement of a very bright and foresighted people, both the Bank of America and at GE. A very small group of GE people, almost without the knowledge of the GE corporate headquarters, signed a contract with the Bank of America to build a very large banking system. Everything was new, the application, the hardware the software and the engineers.

HAIGH: Yes, the ERMA system.

BACHMAN: It was an amazingly successful product.

HAIGH: Was any follow-up work on that still underway when you joined?

BACHMAN: Not much. There was a productized version of ERMA that was called the GE 210. What happened is that the business oriented, data processing people at GE, who had been supporting the banking, as a business, were infiltrated, by many new people coming from other GE departments. These new people largely were those whose own computer experience was oriented toward engineering applications, i.e., scientific computing. They were all IBM 704 users, with the exception of those from GE Hanford. The first new computer they built was the GE 200 series. The GE 215, GE 225, and GE 235. It was a GE Process Control computer, with peripherals, designed by Arnold Spielberg. In other words, they abandoned the horse race GE was winning, and tried to build a computer in their engineering image. In some sense, the Computer Department never recovered from that mistake. They kept growing for a long time, but from being the industry leader, they fell further and further behind. They owned the banking market for a long time. Interestingly, the GE Time Sharing system and the IDS database management system were their strong marketing issues until the IBM PCs and the relational DBMS, respectively, took the wind out of the “sales,” in the 1980s.

Timesharing was the antecedent to the PC. All of the engineers could work on their programs from there own desks through teletype machines and later monitors, and request their compilations and execution remotely. They would get their results back and immediately get another chance to correct their errors. The computer was sharing its resources between multiple users at the same time, actually while running batch problems concurrently.

HAIGH: At the point you joined in ’64, would you say that morale was good?

BACHMAN: I would think so, yes.

HAIGH: Was Herb Grosch still there at that point?

BACHMAN: No, Herb was gone. Herb was a difficult man to live with. Still is. He doesn’t come to GE Computer Department alumni meetings. He said he’d come if somebody paid his way. He said, “I don’t go anywhere that people don’t pay my way.” We said, “Thank you Herb, but, no thanks.”

HAIGH: What were your duties?

BACHMAN: Initially, I was assigned to work for Chuck Thompson, who was the head of the Product Planning group. Most of my activities were in supporting the implementers who were doing the IDS implementation for the 400 and 600.

HAIGH: So by the time you arrived there, there had already been a decision that they were going to produce IDS for the 400 and 600?

BACHMAN: I think, like all decisions, they wanted to do it. You know, going to do it and doing it are two different things.

HAIGH: So it was on the to-do list, but you don’t think they really embarked on it?

BACHMAN: Yes, they did it. But, it was implemented by their own software implementation groups, which I was not part of. Part of my time was supporting the CODASYL Data Base Task Group that was getting underway. At first, it was called the List Processing Task Group, and latter the Data Base Task Group. That took quite a bit of my time. I wrote, rewrote, and rewrote the specification for the DBTG specifications for a database management system. The original text was the text for the GE IDS database management system. The “master” and “detail” records of IDS became the “owner” and “member” records of the DBTG specifications. The word, “set,” was the in word. These were good improvements. Some of the changes were really to satisfy some member’s desire to have something that they could point to as their contribution. My challenge was to preserve the IDS functionality; not to preserve the original IDS data manipulation language, in all of its details.

HAIGH: Yes, and we’ll talk about that a little later.

BACHMAN: That’s part of my agenda though. Also, I was working with the Education Department and developing coursework to help train people in IDS.

HAIGH: So there was a training department that was separate from the software groups?

BACHMAN: Oh, yes. There was a separate group led by Nancy Taffel, who was also a rancher. She had a ranch in the mountains north of Phoenix. She had what is called a "cow and calf" ranch.

HAIGH: Now, the "Product Planning group"—it sounds like they should be planning new products. Were you doing that?

BACHMAN: I was supposed to, but I don't recall that I knew what I was doing in that field. I went to Chuck Thompson, the manager, and he said, "Make a forecast for how many of these things we're going to sell next year." (I do not remember what the product was.) I said, "How do I do that?" He said, "Well, make a guess. You can make a better guess than anybody else can." Somehow, as an engineer, that gave me trouble. I always felt there had to be some systematic way to get to the answer. "My best guess…" When I made my best guess, I had very little information behind it. It didn't feel very satisfactory.

HAIGH: Just before we move on to talk in more detail about IDS and CODASYL, were there any important non-IDS or CODASYL related projects that you worked on during your time in the computer department?

BACHMAN: Well, the Computer Department assignment covered six years. In fact, it actually carries on until 1970.

HAIGH: '70, yes. 1964-1970.

BACHMAN: The "DataBASIC" project was one that was active during that period of time. In fact, GE computer department was restructured as a division, maybe about 1968. We had a new division general manager, Stan Smith[57], and several new departments under him. They, collectively, were doing the same things that the Computer Department had been doing. Before that happened, Stan Williams of the ISP 2 Project, came out Phoenix to start redeveloping our MIACS manufacturing system, as a commercial product. Under the Advanced Computer Department, Stan was head of the Advanced Software group. I joined Stan and worked with him again. It was within that group that the DataBASIC product was developed.

HAIGH: That'll be worth talking about a little bit. Any other projects?

BACHMAN: I'm just trying to think. The WEYCOS2 project happened during this time. Both WEYCOS 2 and DataBASIC, both had strong database overtones.

HAIGH: So really then, from the point you joined the Computer Department in '64 onward, you were really focused on database things.

BACHMAN: Fundamentally databases, yes.

HAIGH: Did you do any hands-on coding for any of the subsequent versions of IDS?

BACHMAN: No.

HAIGH: Who were the people who were responsible for that?

BACHMAN: The programming people in the computer department who were organized in product line programming groups. They had a GE 400 programming group, a GE 600

[57] We were told that "J. Stanford Smith" was plain "John Smith," before he entered Stanford University.

programming group, and let's see if I can think of the names of people. George Gillette headed of the IDS 600 group. I don't remember who was head of the IDS 400 group. Oh, I remember, that was Ed Roddy. Both of them had come to Phoenix from the GE Hanford Atomics Product Operation and had worked on the Hanford 702 Report Generator project and the 9PAC report generator project. When I first met Ed, he was an operator on the IBM 702 computer at GE Hanford.

My staff programmed the DataBASIC product.

A joint GE and Weyerhaeuser team, based in Tacoma, Washington, programmed the WEYCOS 2 project.

HAIGH: Do you have a sense of how big the team involved would have been?

BACHMAN: I'd say three, four, or five people for each IDS implementation. DataBASIC was two or three people. WEYCOS 2 ran eight to a dozen.

HAIGH: I imagine one of the shifts, that I think we alluded to, would be adapting IDS to work with COBOL rather than GECOM.

BACHMAN: That would be a relatively simple thing to do. The IDS preprocessor had to do two things. The first was create an in-core, meta data structure representation of the data definition records. This had to be imbedded in each application program. The second task was translating each IDS DML statement into a subroutine call to the appropriate routine in the IDS runtime library.

HAIGH: Do you know what other important changes were made to IDS over the 1960s?

BACHMAN: I don't think there were any, except for the WEYCOS project. We haven't talked about the WEYCOS project. That'll be a vast order. That was big thing. That took almost all my time for six months. And a great deal longer for those who carried the project to completion.

HAIGH: So we should talk about that.

BACHMAN: There were really two WEYCOS projects. You could say "WEYCOS 1" and "WEYCOS 2." However, the first WEYCOS 1 project was not so named, until the second one come along and there was a need to differentiate between the two of them.

HAIGH: Now did those lead directly to improvements to IDS?

BACHMAN: Not to the commercial GE 400 and GE 600 IDS products. Not until much later.

HAIGH: Okay, so we'll loop back and talk about WEYCOS1 in a second after we've talked about IDS improvements. Did it remain true all the way through the 1960s, for example, that data definition was done from punch cards, or did they change the way data definition was handled?

BACHMAN: If they did, I don't know the answer to your question. I was out of touch with it. Of course, you should remember that all programs were written by hand, punched into punch cards and then to brought to the preprocessors and to the compilers. Why not the data definitions too?

HAIGH: To the best of your knowledge, did it remain true that the only way for an application programmer to interact with IDS was by inserting subroutine calls to the runtime library? That you couldn't do it from an assembler, and that there wasn't any kind of separate ad hoc query interface.

BACHMAN: Assembly programs could call the same IDS runtime library. There was no query interface, unless somebody wrote an application program that looked like a query interface. To my knowledge, there was no standard query processor that behaved like an application. Remember, there was no interactive access to the computers in 1964. Timesharing was just coming in and it was treated as another application. Only timesharing had access and control of "dial-in" input and output. With timesharing, you actually made a phone call to the computer to get your Teletype machine connected to the computer.

HAIGH: Did anything change in its relationship with the operating system? I would imagine that maybe these later machines would have more of an operating system outside of IDS.

BACHMAN: The Problem Controller, and the ability to do transaction processing was lost in the migration to the GE 400 and 600. That is, with the exception of the Low Voltage Switchgear people. When they got their 600, they re-implemented the Problem Controller, out of the original IDS onto their 600. For some reason, GE never really attacked transaction processing again. I failed to understand the importance of the way that the Problem Controller supported MIACS and to understand that we could have re-implemented the Problem Controller as a separate long running application, accepting input from the outside via a magnetic tape, writing its output to a second magnetic tape and doing its own loading and unloading of transaction processing routines. What is obvious today, was not visible to me at that time. We know that it would work, as the Low Voltage Switchgear people made it work, when they migrated their GE 225 MIACS to the GE 600.

Many of the people GE recruited for the Computer Department came out of places like Evandale, Ohio, which is the Large Jet Engine Department, or Pittsfield, Massachusetts, which is the Large Transformer Department. The Computer Department was a magnet drawing people from all over GE. These are places where the new employees had been engineers solving engineering problems. There just wasn't a good feeling among many of them for what business problems were. Some of these people with important software management jobs were hostile to IDS and didn't understand why either it or the Problem Controller were important.

There was one exception to that broad statement; these were the people from GE Hanford. Many of the people that came into Computer Department, from Hanford, had some good feeling about the data processing situation, and business processing, instead of engineering processing, came from people from Hanford, or came from 9PAC people that had been 9PAC customers someplace else, and later came into the Computer Department. Another group that were very helpful to moving IDS into the field came from the GE departments that were earlier users of IDS running on the GE 200 series machines. They really starred in the field support of 400 IDS and 600 IDS when it came to the market.

They were the leaders in the field support organization. These were our Joe Gonnino and George Fadok, for examples.

HAIGH: When you say “transaction processing”, was that system you described where it’s decided what program to run next, and it’s got the list of problems and their data?

BACHMAN: In contrast to batch processing, where you load a program and run a whole slew of stuff through, then you do something else, essentially, transaction processing is “just-in-time” processing: when something comes up you do it, or queue it briefly to handle something with a higher priority. You don’t just wait until the end of the week or end of the month to do that kind of thing. It requires a different kind of behavior from a batch-oriented operating system, and it’s implementing something differently.

At the same time that we had this transaction processing confusion, the GE Computer Department did some very pioneering work in timesharing. The GE timesharing system was a big, big influence. The system originally came out of a Dartmouth University system, running on their GE 235. The GE 235 had a companion telecommunication computer, the GE Datanet 30 interacting with it. The Datanet 30 could handle a number of telephone lines. People could dial up from their Teletype machines and communicate with the Datanet 30. The Datanet 30 passed information back and forth with the GE 225. Dartmouth College also developed the BASIC programming language for engineering/scientific problem solving. The Dartmouth BASIC system was out and successful before the MIT timesharing system was out. The GE Computer Department commercialized the Dartmouth BASIC system with its GE 235 computer and GE Datanet 30. A good part of the GE success for a long time was the timesharing business they had based on the Dartmouth system. Timesharing operating systems are different than transaction processing systems, because they’re dealing with long-term session with users who are doing things randomly. Essentially, the transaction system is something that’s prepared to receive a flood of transactions of mixed type, do whatever each of the transactions indicated, keep on going, and looking for the next transaction to process. This is essentially what the Problem Controller did because it didn’t care what type your problem was, “Just tell me what the problem type code is and I will know what program to load.”

[Tape 3 of 7, Side B]

HAIGH: So that transaction capability wasn’t in the productized version for the 400 and 600 series?

BACHMAN: No. The Problem Controller was a different operating system. The GE 400 and GE 600 had multiprogramming operating systems. They were designed to run batch programs and later added in timesharing. For a long time, the people in Phoenix thought IDS was an application program. Their batch-oriented operating systems could run IDS application programs and schedule them like anything else. Two IDS programs could be scheduled to run at the same time, if they used different databases. They were not treated as transaction-oriented programs that are always there, continuously, responding to the world’s work and prepared to go to sleep when the world did not send any messages to work on. When we switch over to the Weyerhaeuser, there is a big story that fits in here, because Weyerhaeuser added more to the IDS story.

HAIGH: But the product version of IDS would then have been used by application programs running in batch mode?

BACHMAN: Yes. Essentially the same way that IDS was used in the first place, except we lost the ability to do the transaction applications. We lost the ability to send a message to another business process. In other words, application programs in IDS originally could do two things. They could access and update the database, and could generate messages to other programs, which would be processed in due course. This was a very important part of making a manufacturing control system work. There was a need to originate messages, rising everywhere through the business, including from other programs.

HAIGH: So IDS would never have worked in a real-time way?

BACHMAN: I don't know what you're saying. The way you asked the question, my answer is no, but I think I need to know what your question is more clearly.

HAIGH: From what I understand, the 600 series, and 400 series computers that the productized IDS ran on, were multiprogramming. So, they would have several programs loaded into memory at the same time.

BACHMAN: They were running a multiprogramming batch system.

HAIGH: Yes, so there would be no provision on those computers for a real time--

BACHMAN: Except for timesharing.

HAIGH: So timesharing was being run on these same computers?

BACHMAN: Yes.

HAIGH: Was there any provision to use IDS with timesharing programs?

BACHMAN: No, not at all. There was no interactive front end. That was the basic problem.

HAIGH: And if you had several batch programs running at the same time?

BACHMAN: Only one could run IDS, or only one could run IDS on the same database. You'd have several copies of IDS, each addressing a different database.

HAIGH: In that case where you had several with IDS in different databases, would they get an IDS instance each? Or could one IDS instance serve two application programs?

BACHMAN: I don't know the answer to your question. It wouldn't surprise me either way. That's an optimization thing that you may or may not get done depending on schedules and manpower. Like I said, the people in Phoenix tend to think of IDS as an application program, so there's probably less chance…. They may have some shared disk I/O at the bottom. It would probably be the Block Controller. The virtual memory part was not shared.

BACHMAN: WEYCOS is short for Weyerhaeuser Comprehensive Operating System. At that time, the GE 600 operating system was called GECOS. Everything was GE something. GECOS was the GE Comprehensive Operating System. WEYCOS was the specialized version of the GE 600 operating system that was built to meet Weyerhaeuser's requirements.

Let's go back. I call that WEYCOS 2. "WEYCOS 2" was a name given to distinguish it from the GE 235 version of IDS that used the MIACS Problem Controller as its operating system at Weyerhaeuser. That original GE 235 version was later called "WEYCOS 1."

This first WEYCOS was an interesting variation of the Problem Controller. It ran with the Problem Controller in transaction processing mode, but with a telecommunication front end. What they were able to do was to replace the card reader, as the input device, and the printer and card punch, as the output devices, with the GE Datanet 30 computer that would connect as many as 50, 75, or 100 remote teletypes, as input and output devices. Every one was available at all times to receive or send a new message. These teletypes were scattered around the United States, at all of the Weyerhaeuser sales offices, mills, and warehouses. In a sense, this version of the Problem Controller was a real message-oriented operating system. Messages could come in and be sent out to somebody else, or they could come in and be transferred to the GE 235 computer to be dispatched by Problem Controller.

They made another modification to it, which I thought was also significant. The Problem Controller maintained the same four-level hierarchy of control records, i.e., the System record, the ProblemType records, the ProblemStatement records, and the ProblemData records. However, they found one weakness in our design. From Weyerhaeuser's point of view, it needed to be fixed. It turns out that we violated an important rule in systems analysis. "Never use a meaningful code for a record identifier." Our identifier of the ProblemType record type was also the dispatch priority level for its ProblemStatement records. That is a "no-no," in systems design theory, that goes a long way back. What if you want to change the priority of some problem type. Then these problem type codes have to be changed. That code is both the problem type priority and its identification. If you want to change the priority, which means you need to change the identification which makes a mess out of who knows what. So they added a second relationship between the one–of-a-kind, System record and the associated ProblemType records. The original relationships was sorted by problem type code to insert new Problem Statement records. The new list was sorted by priority for dispatching purposes. IDS permitted you to define two different relationships between the System record and the Problem Type records, for different purposes, one by problem type to find the problem type to store new ProblemStatement records; the other to find which problem type I'm going to run next. They would switch the priority values on ProblemType records, from day to night, on this system based upon different things going on. As the sales offices were all closed at night, they would loose their priority. Grayce Booth of Weyerhaeuser was the lead programmer on the modification of the Problem Controller. At the same time, Paul Kosct was modifying the IDS Block Controller so it could handle multiple disk drives and a larger number of IDS records.

The Modified Control Structure for the WEYCOS 1 Problem Controller.

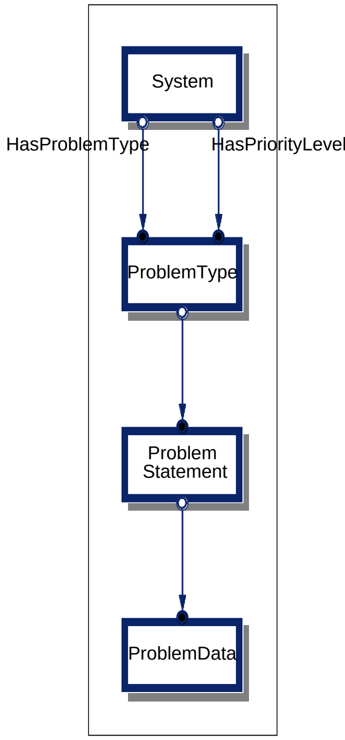

Figure 6.

The new Problem Statements were linked to their Problem Type based on their problem type code. The next Problem Statement to be dispatched was based on the Problem Type with the highest priority level, that had a Problem Statement record attached.

HAIGH: Now this system, was it already operational before your involvement?

BACHMAN: The previous Weyerhaeuser order entry system, prior to WEYCOS, was teletype based, but was all manual. It was a typical "torn tape" Teletype system with manual input and manual processing of messages received. I do not know where the switching center was located, but Tacoma, Washington, at the corporate headquarters would be a good assumption.

Just to make it clear, the first installation of WEYCOS at Weyerhaeuser was completely under the direction of Bill Helgeson and a group from Weyerhaeuser and IAO. The integration of the Datanet 30 with its communications interfaces and the changes to the Problem Controller were their innovations. I am not quite sure of the date, but this transcontinental, real-time I/O, transaction-oriented, database-oriented, sales order, processing system was up in running some time in late 1965! That was impressive, and it still is.

HAIGH: So if it is manual, then you had a bunch of people there. There were a lot of things coming in to a central office on paper tape and being ripped off, manually sorted, and retransmitted?

BACHMAN: It was all manual before the introduction of the WEYCOS 1 system. Every sales office had one paper tape Teletype machine. They addressed their paper tape to different shipping points, to whomever they wanted to make the shipment. Each shipping point would handle its own inventory control in response to an order. Each shipping point had to acknowledge receipt of the order to the sales office.

After WEYCOS 1 was running, the inventory control and order acceptance was handled on the GE 235 running IDS and the upgraded Problem Controller.

That order might be for a railroad car full of lumber, or even a shipload of lumber. This was the first automation, so they would bring the orders into Tacoma and get the order processing back out in a much quicker time.

Alan Smith, the Weyerhaeuser vice president, in charge of their data processing, told me an interesting story. He said that at one point in the WEYCOS history, all of a sudden, they got a large influx of orders coming in from the sales offices. Orders were coming in faster than IDS and the GE 235 could handle. The orders flowed in starting on Monday morning. They would get part of Monday's orders responded to on Monday, but there'd be some left over to process overnight Monday and into Tuesday. And on Tuesday morning more orders would start coming in. So the queue of unprocessed orders kept getting bigger and bigger, all week long. By Friday night, they had a whole slew of stuff backed up to be processed. He said, "Fortunately, between Saturday and Sunday we had worked off those orders, so we had an empty new order queue[58], on Monday morning. All the orders had been queued were processed." However, they didn't like the fact it might take two days to get to an order processed, because it'd be two days backed up, by Friday night. In addition, he said, "What if we get more orders yet? We don't have eight days in a week. We only have five business days in a seven day week."

HAIGH: This use of IDS. Was this in WEYCOS 1, or is this separate from WEYCOS?

BACHMAN: This is what I'm characterizing as WEYCOS 1: the version of IDS and the Problem Controller, working in conjunction with Datanet-30. It was a telecommunications version of the Problem Controller. It was the Weyerhaeuser version of the Problem Controller. IDS was unchanged. This system is what we latter came to call WEYCOS 1 to distinguish it from the latter WEYCOS 2 for the GE 600 line.

HAIGH: What year was this happening in WEYCOS 1?

BACHMAN: I would say, and I'm not sure, they probably started operations in 1965[59]. Bill Helgeson was the key man from Internal Automation Operations on the project. Paul Kosct was another. He was the first resident person from the GE IAO who moved out to Tacoma to support the Weyerhaeuser account.

HAIGH: So they would have bought a GE computer to handle this and got IDS with it?

BACHMAN: Well, no. Weyerhaeuser already had several GE computers. IAO came to us to get IDS, because the computer department wasn't involved in the GE 225 version of IDS.

[58] These were represented by the "new order" Problem Type, its list of Problem Statement records (each a new order) and their associated Problem Data records.
[59] 01/25/2006. Paul Kosct confirmed "early 1965" this morning.

HAIGH: So they just bought this computer and hoped it would do this, and fortunately for them, they happened to have bought the computer that IDS ran on?

BACHMAN: They bought a proposal from the GE IAO organization, to put together a online order processing system. It included a centralized inventory control system. The key components were the GE 235, the GE Datanet 30, IDS, the Problem Controller, and their existing Teletype network. They already had GE computers. They just bought another one.

HAIGH: For this application, and you became involved with it because?

BACHMAN: I was only involved at a distance. I did not go to Tacoma, Washington, where they were located that time. All of my communications were with Bill Helgeson, by phone. We talked about things to be done to solve their problems.

The story, about the time when Weyerhaeuser had a flux of orders that could not be handled immediately, has more to be said. The vice president said, “The thing that’s interesting is that if a transaction had a high priority, it was still processed immediately. For example, before we could enter an order, if it’s a new customer, the sales office needed to get a customer code for that new customer.” It took five seconds to get a new customer registered and get its customer code assigned. That was a high priority transaction. Weyerhaeuser had added the priority processing capability to the Problem Controller. Each problem type had an assigned dispatch priority that was sensitive to the business requirements. The things that needed to have a fast response were bypassing the jobs that came in earlier, but could wait. The Problem Controllers dispatching algorithm was paying attention to the priority level of ProblemType records. The fact that some things were taking two days to process doesn’t mean everything was taking two days to process.

HAIGH: To get this straight, before WEYCOS 1, they had a manual system, and it was also distributed? There were people in different offices doing the work?

BACHMAN: Yes.

HAIGH: And WEYCOS 1 centralizes this and automates it using an IDS/Problem Controller based system, coupled with the smaller computer that’s handling the paper tape coming in?

BACHMAN: Yes. It handled the telecommunications, in lieu of paper tape. It received the message without punching the paper. It would send out messages that would be received by Teletype machines, when they got to the warehouses, or back at the sales offices.

HAIGH: So then people from all over the country were sending this stuff in, to be processed centrally, and they’re getting the results back on paper tape?

BACHMAN: Yes, and No. They were receiving the information on their Teletype machines. It was printing the information that they were receiving on the rolls of paper in the Teletype machines.

HAIGH: Do you know if, at this point, systems of that complexity were common, or do you have a feeling that this was--

BACHMAN: They were very rare. These were the early days of the American Airlines SABRE system.

HAIGH: So you had a feeling of this being on the leading edge in terms of coupling an application system with a database and telecommunications technologies.

BACHMAN: It was totally unique. It was one of a kind; the first of the kind. One thing, I mentioned this idea earlier and I'll go back and repeat it here. Because the GE 235/GE Datanet 30 complex, in Tacoma, had really no local input/output devices (it didn't have a card reader, puncher, printer on it) all the input/output was coming electronically from someplace else. This used to drive the computer room operators wild. They look at the IDS machine and it just sat there, blinking. These were the days when computers had blinking lights on the front of them. They would blink fast, so you couldn't really tell what's going on. The operators would ask, "What if it's in a loop? How do I know?" You couldn't tell anything, just standing looking at the machine. Bill Helgeson and the group at Tacoma finally connected an IBM Selectric typewriter on the GE 235, as an operator console. The Problem Controller as modified, so that once an hour it called a Problem Type that listed the job queue on the Selectric typewriter. If the job queue is different one hour than it was the prior hour, everything must be okay. This made peace among the operators, who felt they were responsible. If it's working, fine! They could then go back and do something else. They'd probably felt they would get chewed out badly, if the GE 235 sat there for eight hours doing nothing, when it's supposed to be doing something. No one could tell, until they installed the Selectric typewriter to print out the job queue, so that they could tell that something was going on.

HAIGH: What was WEYCOS 2?

BACHMAN: Okay, WEYCOS 2. I have said that they got to the point where IDS WEYCOS 1 was barely sneaking by. The Weyerhaeuser requirements were barely being met, running 24/7.

HAIGH: How long did it take them to get to that point?

BACHMAN: Couple of years. I'd say from '66-'68. They said, "What we need to do is have a bigger computer, and we'll do multiprogramming accessing a shared IDS database." This was the first multiprogramming database system I'm aware of.

HAIGH: What exactly do you mean by "multiprogramming database system"?

BACHMAN: When multiple, independent programs are running against the same database at the same time.

HAIGH: So you have one IDS, and multiple programs can access the same or different databases running on this one instance of IDS.

BACHMAN: In this case, Weyerhaeuser only had one database, the order-processing database. They wanted a single integrated database that could support all aspects of order entry, shipping, invoicing and inventory control. They were doing the best job they could with an integrated system. If they had two distinct databases in their GE 235, I'm not aware of it.

HAIGH: It would make sense they would have only one, but the point was that several applications can access the same database.

BACHMAN: Maybe the same transaction program, but processing different transactions. Two orders from two different sales offices, two orders from the same sales office, it didn't make any difference. They're distinct transactions, and you want each one to run as if it was in a database all by itself. The result you want is to give each program database access, as if there were no interference, from any other program. The goal was that the application programmer and the programs that they wrote could assume that they existed all alone in the database without interference from any other program.

HAIGH: I imagine that would bring up some issues with locking and concurrency.

BACHMAN: Oh yes, a whole bunch of things. This WEYCOS 2 had two objectives that Weyerhaeuser wanted it wanted to do. One was to greatly increase the computer capacity, which is a separate subject, what we called, "multi-wing WEYCOS," which never really got to work. The GE 600 had separate memory modules, and you could connect from one to four computers around a sharable memory module. You could connect from one to four sharable memory modules around a GE 600 computer. A single computer with four memory modules used up the contestability of both the computer and the memory modules. However, it you started with one memory module and shared it with four computers, each of those four computers could have three more memory modules for its own private use. Our target for the multi-wing WEYCOS was four computers and a total of thirteen memory modules, all working together. The center memory module was to be the control memory where all of the sharing was controlled.

The other part of this project was the ability to provide a shared access IDS database. That's the one that the GE/Weyerhaeuser team succeeded in accomplishing, and which was very successful. That was a project that I was involved in for the first six months, essentially during the conceptualization and specification writing phase. It turns out Russ McGee, whom we have talked about before, moved his family up to Tacoma, Washington from Phoenix, to manage the WEYCOS 2 project. Russ and I crossed paths several places in this story. We have always had very successful working relationships and successful projects, which is the best way to get the working relationships. For the first six months, I commuted to Tacoma weekly: I'd fly up Monday evening, fly back on Friday evening[60]. Then Russ McGee took over.

HAIGH: How long did it take that project to conclude?

BACHMAN: It probably took them another 18 months. I'm guessing 18-20 months from just the feel of things before they had it in production.

HAIGH: If they would have started in about 1968, then they--

BACHMAN: They started, in the summer of 1966.

Stan Williams was involved too. It started immediately after Stan Williams moved from the MIACS project, in Philadelphia to join the Computer Department in Phoenix. Stan, his wife Olga, and their family moved to Phoenix and were staying at our house while we were on vacation in Michigan. When we got back, Stan said, "You have to go to Tacoma, Washington to get a new project kicked off." I said, "What? What project?" We had

[60] Frequently, carrying a fresh salmon, from the Tacoma fish market, for a Saturday roast over the grill in Phoenix. Grilled, with a white wine and green grape marinade.

arrived back on Friday night. On Saturday morning, we packed all four kids back in the car, and took off again going northwest. That Saturday was blistering hot. It was so hot that our station wagon overheated when we ran the air conditioner. We stopped for gas, somewhere in the northwest corner of Arizona. When we asked how hot it was, they said, “You do not want to know.” We made Las Vegas about midnight. Sunday we reached Tacoma, Washington. On Monday morning, I reported for duty at the Weyerhaeuser offices.

HAIGH: At the point this project was finished, are you aware of any other systems that had that multiprogramming database capability?

BACHMAN: I am sure that WEYCOS 2 was the first one.

HAIGH: Are you aware of any direct influence that that system had on other things?

BACHMAN: I have talked to Jim Gray, recently. He started with IBM and then went to Tandem Computer. They were early manufacturers of “fault tolerant” computers. They advertised “non stop” computing. Jim wrote the early technical papers about deadlock detection and recovery-and-restart, when he was with Tandem in the 1970s. Jim won the ACM Turing Award in 1983 for his work on databases and transaction processing. He is at Microsoft Research now. Jim’s employment at Tandem and his work on concurrency and recovery happened long after we had the WEYCOS 2 production system running. Jim freely acknowledges our earlier work at Weyerhaeuser.

I don't know whether what Jim Gray did at Tandem was a reinvention of the same thing. It was a logical thing to invent. Or, whether they heard about what we had done. It was no secret. I just don’t know.

HAIGH: So there was nothing in WEYCOS 2 that fed directly into the Database Task Group or into any commercial products?

BACHMAN: Only the subsequent Honeywell IDS products supported multi access databases.

HAIGH: Were the Honeywell products directly influenced by WEYCOS?

BACHMAN: When Honeywell bought GE[61] (or whether GE bought Honeywell, it was never quite clear), they would have folded the GE people still working with WEYCOS back into the GE computer department. The GE computer department in Phoenix ran a autonomous world even after Honeywell bought the business. They were an independent bunch of people. They were more successful than the Honeywell people were, and they flagged it hard. “If I am a winner, you can’t complain about me.” They were supposed to be working on the new product line that Honeywell was developing. But, they were big foot draggers on the new product line. Engineering was going into enhancing the 600 line, which became the Honeywell 6000 line, mainly by changing the literature and labels on the computers. They did get to a shared access version of IDS, just like the IDMS[62], the Cullinane Database Systems product.

---

[61] Announced in the Spring of 1970 and consummated in October of that year.

[62] IDMS, Integrated Database Management System. IDMS began its life as an IBM 360 oriented version of IDS that was created by BF Goodrich. They received that original version of IDS from Clark Karcher of International GE.

Deadlock Detection Control Structure[63]

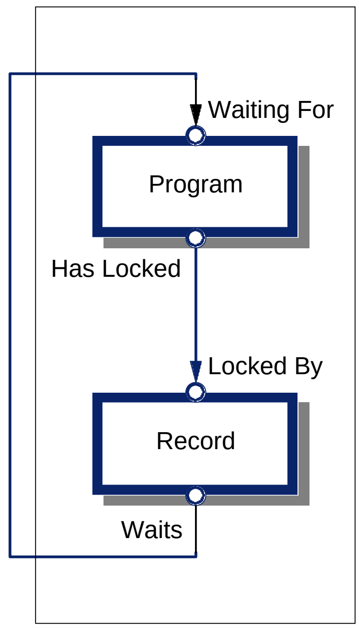

Figure 7.

The WEYCOS 2 system was designed so that each time a program requested access to a record, IDS would determine if some other active program had looked at that record. IDS kept track, as each program kept read and implicitly locked records. If a program ran into a record it can't get access to, because some other program had it locked up, IDS would looks to see if that second program were running, or if it were waiting. If so, who were they waiting on? IDS would look around the network of "programs waiting on records, locked to programs, waiting on records, etc., . . ." Look through this hierarchical structure of locks and waits, to find out whether a program, if it were allowed to wait on a record, would ended up, indirectly, waiting on itself, which would create a deadlock. If it would be waiting on itself, it was prohibited from doing this. A deadlock had been detected. Sometimes there are two programs fighting over access to the same records. Sometimes there would be three or more of them fighting. These programs were evaluated by IDS, as to which program had the least invested and abort that program. That program's locked records would be released and the program would be restarted again from the top. Really, it was transactions that were waiting, not programs. Programs are only the processing rules, the transactions are really what are active or waiting.

Alternatively, IDS would keep track of what pages the application programs looks at. They did it both ways to see at which level deadlock detection worked the most

[63] This is a hierarchical structure, not a network structure, because no record type is a member of two sets. It is a hierarchy of programs, that are locking record, that are waiting programs, that are locking records, etc.

efficiently. The WEYCOS 2 experience was that when they kept track of locks at the page level, they detected more deadlocks and thus required more recoveries, than if they kept track of locks at the record level. It was automatic; deadlock detection and recovery and restart under programmatic control. Weyerhaeuser had more deadlocks per hour running at the page level than they did when they ran at the record level. This happened because they were implicitly locking all the records in the page, even if only one of those records was really being accessed. It turned out that page level locking successfully processed more transactions per hour, because the bookkeeping was so laborious, working at the record level. It was easier to go through some additional transaction recover and restarts than it was to keep track of all the record locks. They end up running production all the time, with the page level deadlock detection turned on.

HAIGH: And through this period IDS is continuing to handle its own virtual memory management?

BACHMAN: I think it still does. I think the answer is that IBM's IMS does it and all the relational databases do, because the database virtual memory has never been integrated into the hardware supported virtual memory. I've never been quite sure why, except that people probably still think that databases are an application not an operating system phenomenon.

Another reason may be critical. The hardware supporting virtual memory does not generally share program pages between several programs. The hardware supporting virtual memory is assigned to a particular program so there is not opportunity for deadlocks being created. Each program has its own set of buffers to support its virtual memory.

HAIGH: Do you know if there's any surviving documentation of these WEYCOS systems?

BACHMAN: I would ask Russ McGee. In fact, if your order of business in these kinds of things allows you to suggest topics for the ACM SIGMOD, I think what he's gone through several areas is well worth chasing, because they were real pioneering activities in which he has firsthand knowledge.

HAIGH: Well, we can talk to Rick Snodgrass, this evening also.

I think the next thing to talk about would be the CODASYL Data Base Task Group.

BACHMAN: The CODASYL Data Base Task Group was a group that was initially set up by the ANSI SPARC committee to… I'm not sure what it was told to do, but it was called the List Processing Task Group. Warren Simmons, from US Steel, was the driving force in getting it started and running it. His intention at that time was to essentially develop a new standard based on IDS. That was his goal. People looked at it and thought about what we done and called a "list processing" system. That was a misnomer, really. However, we really did process lists, or chains as we called them. I believe the name, List Processing Task Group lead people to think that we were looking at Artificial Intelligence (AI) systems.

HAIGH: Who called it the list processing system?

BACHMAN: Well, I think Warren Simmons was probably responsible for the name.

HAIGH: His installation was an IDS user?

BACHMAN: No. Well, I think that he knew IDS, only by reputation. Warren was a professional standards person for US Steel. He was involved in all kinds of standards, for all kinds of purposes.

HAIGH: So he hadn't used it, but he thought it sounded like something that needed standardizing?

BACHMAN: He had heard about it and thought it was something interesting, because he had been involved in COBOL work, and knew that there was a great, blank space where something had to be plugged in, in order to do this. He had looked at the IDS closely enough to see the linked lists, and said, "Well, that's a list processing system," and he did not think about IDS's general purpose. In any case, the "List Processing" name was not good advertising for what was a database management system. In those days, we weren't sure about what we called things, or what the right way to talk about them.

HAIGH: Do you have a sense of when you first heard the term "data base"?

BACHMAN: I don't really. It came on the scene, and I'm not responsible for it. I didn't pioneer it. Somebody did. I don't know. Our focus was to come from the Integrated Data Store name, and the "data store" is what IDS was from my view of it. Database, that's another name for it. I felt no need to jump on that name, although I think I was one of first persons that tried to think of "database" as being one word, and not two words. In fact, when the Dick Canning and I started the magazine for the special interest group in ACM in business data processing. Should the title of their magazine should be *Data* (space) *Base*, or (one word) *Database*. We decided it ought to be *Database* as one word, as a thing, not a *"base"* with an adjective in front of it. Once they get put together that way, they have a life of their own. Databases had a life of their own.

HAIGH: So the committee was founded as the List Processing Task Group?

BACHMAN: Yes.

HAIGH: Apart from Simmons, who was on it?

BACHMAN: I don't remember all the people on it.

HAIGH: Can you remember if they were coming from computer manufacturers primarily?

BACHMAN: Mostly from the computer manufacturers, yes. Note: I just found the October 1969 report of the "Data Base Task Group." "Data" and "Base" are two words! It lists current and former members of the group. Some of the names include: George Dodd[64], General Motors; Gerry Durand, Southern Railways; John McNeley, Allstate; Bill

[64] George built a very attractive IDS like system, called "Associative Programming Language" (APL) which was integrated with PL/1. General Motors used it extensively in the graphic design system for automotive development. IBM latter asked GM to give up the initials, APL, so they could use them with their, "A Programming Language."

Olle, RCA; Don Rapp, Travelers Insurance; Dick Batchelor, Honeywell; Marvin Beriss, Univac, Larry Sturgess, Burroughs; there was no one in the beginning from IBM[65].

HAIGH: What was your own role?

BACHMAN: I think my role was, to teach these people what IDS was, so we can develop a standard enhancement of COBOL that was based on IDS. It was to achieve a predefined end. It wasn't to do research or do studies. It was to standardize.

HAIGH: So the idea was to get IDS capabilities into COBOL?

BACHMAN: Yes.

HAIGH: Were you a formal member of the committee at this point?

BACHMAN: Yes I was, yes.

HAIGH: Representing General Electric?

BACHMAN: General Electric, yes.

HAIGH: What do you remember of the committee's meetings? Did it work productively? Was it harmonious? Did people have a clear sense of what they were doing?

BACHMAN: I think that for the first year, my memory is saying this group met maybe every six weeks or so. Two months, I'm not sure. In the first year, there was so much that was new in IDS, it was an education process. People had no experience to tie it back to. They weren't familiar with any applications that were running it, and the applications that were out were all from various GE product departments, whose business wasn't to advertise it. It was not a secret, but it just wasn't anybody's marketing vehicle at the time, either. It was a difficult education program.

I forget who or where, roughly at two years, said "let's change the name of this thing to the Data Base Task Group, now we understand what we're doing." In fact, I think it's about time where "data base" was well enough known we could understand what a Data Base Task Group was all about.

HAIGH: I find an article that has some dates for that. They say that it was founded in October 1965 as the List Processing Task Group, and that it changed its name in 1967.

BACHMAN: Two years.

HAIGH: Do you remember anything about whether you personally thought "data base" was a good name at the time? If there was anyone who was opposed to it or anyone who was particularly in favor?

BACHMAN: I don't remember that anyone worried about the name at that time. We were concerned about what it does and how do we tell it to behave? This is the time when the data description language (DDL) and the data manipulation language (DML) came sharply into focus.

HAIGH: From your point of view, for the first year or two it was mainly you educating the other committee members about IDS?

---

[65] Sharon Weinberg, IBM later. Even later, she became Ted Codd's wife.

BACHMAN: And talking about how well the existing DDL and DML languages treated the problems. As people caught onto things, they say, "Well, can't you do this a little better here or a little better there?" In some cases, it was a little better. In some cases, it was just different. In any committee, everyone wants to own a part of the results. Some of the things were important enough to argue about. In other words, if it was you just call something like, "CREATE" instead of "STORE," the change may be a nuisance, but it's not bad. In either case, the intent was to create a new record and stored it in the database.

HAIGH: In these early days, was there any tension between representatives of different manufacturers? You make it sound, at least in the early days, quite harmonious.

BACHMAN: I think so. I don't think there was ever any harshness about it. You know, it was a bunch of intelligent people trying to do a task. I don't think at that point anybody had a vested interest in anything different, because at that point IBM had nothing; Univac had nothing; National Cash Register had nothing, Burroughs had nothing, so that there was no competitive alternative. There were no relational systems out, no IMS. People had systems that may have been a disk drive or drum, or something, but there was nothing that had all of the components it were required to make a production database management system (DBMS). Relational databases came out ten years later.

HAIGH: Although the committee did study existing systems. According to their 1971 report, it seems like there was a rash of systems around 1968, 1969, 1970, e.g., IDS, GIS, products from Western Electric, SDC, and RCA.[66]

BACHMAN: GIS was a query system. I'm just trying to think. Was Bill Olle with RCA then?

HAIGH: Yes.

BACHMAN: I don't know what they had.

HAIGH: They had something called UL/1. I don't know anything else about it.

BACHMAN: You could ask Bill if you wanted to. I'm in email communication with him.

HAIGH: Well, if I had the whole report here, it would say more about what was in UL/1. It's interesting that that's how you describe their early activities because in terms of the published reports that they issued, I think their first two significant products were summaries of the capabilities of existing systems.

BACHMAN: Well, the DBTG was still arguing over how to detail the DML and DDL.

HAIGH: So that activity actually started earlier on, but it hadn't come to a conclusion.

BACHMAN: It wasn't publishable yet. We started with the original DML and DDL that GE had, and largely ended up very close to that, but enough difference that GE finally brought an IDS II out, which subscribed to the CODASYL specification. Cullinane Database Systems never changed IDMS from the original IDS specifications.

Cullinane's IDMS was based on the original GE 225 version of IDS. Two boxes of source code went from Clark Karcher at IGE to Dick Schubert at B F Goodrich Chemical

---

[66] HAIGH: CODASYL Systems Committee, Feature Analysis of Generalized Data Base Management Systems. New York: [n.p., Distributed by the Association for Computing Machinery], 1971.

in Cleveland, Ohio. They were GE computer users. Somewhat later, Goodrich corporate management in Akron dictated that they would use IBM computers, exclusively. So Dick Schubert converted the GE 200 IDS to run on the IBM 360.

HAIGH: Did you remain a member of the committee throughout its life?

BACHMAN: I don't quite know when it died out, but I think I did.

HAIGH: I think it may have reconstituted itself around about '72 or '73 as a different group that was working harder on the COBOL specific issues.

BACHMAN: That was a totally different group. I can't think of the name of it. It started… I know it was different personnel. Tom Steel headed up. I was the vice chairman.

HAIGH: I think eventually they also came out maybe with a set of extensions for FORTRAN.[67]

BACHMAN: I do not remember that, the second group I am thinking of came out with an Architectural statement. A professor from University of Montreal edited and published it.

HAIGH: Yes, there was a new standing committee, which reworked the COBOL extensions into general development form as an official update, and they had a report printed in 1974.[68] I think they continued to trundle away on that through the '70s.

BACHMAN: I would say that second group that Tom Steel headed up had its purpose to avoid standardizing IDS. There was strong IBM participation in that case. They were strong people, who were IBM users. Essentially fending off the standardization of an IDS like standard.

HAIGH: So the way you see it, the Database Task Group itself essentially standardized IDS?

BACHMAN: Well, it didn't standardize it. It made recommendations. It prescribed the language enough that companies like Univac implemented it. There were a number of implementations around the world of it.

HAIGH: Can you remember how this split between the data definition language and the data manipulation language? Where that idea came from?

BACHMAN: What do you mean by "split"?

HAIGH: Well, I guess split and formalization. If you look at IDS, I think from what you said it's clear that there was a data manipulation language and the data definition was done with these punch cards. I suppose it would be the idea that data definitions should be done through a language, and that language should remain separate from the data manipulation language.

BACHMAN: In some sense, they operate at different times. The purpose of data definition language is to essentially populate a database with metadata. The purpose of

---

[67] HAIGH: CODASYL FORTRTAN Data Base Manipulation Language Committee, "CODASYL FORTRAN Data Base Facility," Journal of Development, Document number 110-GP-2, January 1977.
[68] HAIGH: CODASYL Data Description Language Committee, CODASYL Data Description Language: Journal of Development, June 1973. Washington, DC: U.S. Govt. Print. Off., 1974.

data manipulation language is to reference some metadata, and while looking at it, either 1) expand IDS DML statements into in line code, or 2) replace IDS DML statements into subroutine calls, which store or otherwise manipulate real data. These are two different tasks, and there's different timing in these things when they happen. To have them in one vehicle would be confusing and only makes sense when it's for marketing purposes. They are completely complementary, but are used at two different times.

HAIGH: Actually, I think of the distinctive things in terms of "splitting" the languages was that was the idea in the DBTG report that there should be one standard data definition language, and there would be multiple data manipulation languages implemented as extensions to the host languages. There would be conceptually a COBOL data manipulation language and a FORTRAN data manipulation language, and that they would be as natural and as integrated as possible. That is actually different from what you finish up getting today with chucks of SQL embedded into different host languages. Can you remember where that idea of having a separate data manipulation language--

BACHMAN: I don't remember that happening. It may have happened and I just discarded it as being obvious and therefore not worth remembering, or it happened after I was no longer active, or happened in some subset of the groups that I wasn't familiar with. I have no remembrance now that-- As I look in my mind at the final report of that group, it described one data manipulation language and one data definition language, not a family of data manipulation languages.

HAIGH: Yes. I think it only made a serious attempt to describe a DML for COBOL, but as I understand it espoused the principal that there should be multiple ones. And I do recall efforts later to specify a DML for FORTRAN.

BACHMAN: That sounds like a committee actually. It's a big difference between saying, "We ought to do something," and doing it.

HAIGH: Interesting. Now, this language of "record types" and "sets" that they came up with? Were you using that same language earlier with IDS?

BACHMAN: No, we did not use the word "sets." We used the name of a specific set implementation technique called, "linked lists". They were sets. Our sets were implemented with linked lists, where the last member of the set was linked back to the owner of the set.

HAIGH: When you say "we," do you mean…

BACHMAN: On the IDS side.

HAIGH: IDS, okay. And with the Database Task Group, this concept is generalized and called "sets."

BACHMAN: Well, it is a better word for the concept. In a paper[69] of mine, which should be in your collection, it talks about a number of set of implementation techniques of which the "linked lists" are one set implementation technique, "foreign keys" are another, "pointer arrays," "bit arrays," etc. There are a number of different ways that you could

[69] HAIGH: C.W. Bachman, "The Evolution of Storage Structures," Communications of the ACM, vol. 17, no. 7, July 1972, pp. 628-634.

implement a set. Each which has different performance characteristics, whether you're updating or just retrieving. All of these are valid implementation techniques for this notion of a "set." Really, they're not sets, they're "owner-member sets" because mathematically, sets means something quite different. Owner-member sets might be considered a specialization. An owner-member set must have at one owner record and can have zero, one, or more member records. Mathematically, a "set" concept does not have the concept of an "owner." It only has members, and typically, they are not ordered. The owner-member sets of IDS have the "master" and "detail" cards of the punched card days, as their biological parent. It's important to distinguish between the owner-member set, which they were talking about, and a set in the mathematical sense. They're close enough, so the word "set" brings to your mind a collection of things. If you are a trained mathematician, it is probably enough to mislead you as to what is concerned.

HAIGH: Another thing that I saw in reading the Database Task Group report. The first couple of reports that it produced were trying to describe and classify existing systems. Then in the final report, one of the things that struck me was that they were saying that the database management system should provide an ad hoc query system and an interface for application programmers, features, which were not combined in any of the existing systems.

BACHMAN: That sounds like a good thing to say. They didn't specify an ad hoc query system though.

HAIGH: No, but my impression, as a historian is that in some ways, the concept of a database management system that they expressed, there has been more historically important than the specific standard that they laid out in as much as the database management system we have now is relational, but it's still recognizably a database management system as they described. I haven't seen any earlier descriptions of anything that sounds to me as much like what we think of today as a database management system.

BACHMAN: That was several years ago. They got it right early. As a historian, it is important to differentiate between visionaries and practitioners. One of the interesting things about IDS is that the vision, the practice, and the success all came together in a short period. Even more interesting is the fact that the IDMS version of IDS is still an active player in production systems around the world.

HAIGH: So this suggestion that there should be application programming interface and an ad hoc query interface, which I don't think any product that existed at that point had.

BACHMAN: Not to my knowledge.

HAIGH: Do you remember where the idea came from?

BACHMAN: I do not. As I hear you read it, it sounds like a good idea, and one that I would have supported at that time. There's no logical reason why you would ever argue against it. The interesting thing is that, in those early years, the devices from which you could make queries were very limited. The most popular remote I/O devices were Teletype machines. The monitors that began to show up were very new at that time. The notion that you should have remote access to a database was new. The DataBASIC product we built in the GE Computer Department was an early attempt, at letting people do their own thing with databases.

BACHMAN: Another thought I want to bring up goes back a couple of years from where we are. During the time that I was in Phoenix working for the GE Computer Department. I spent some time on a specification of a product or facility that I called the "Intercommunicator." It supported the notion that local and remote terminal devices were being hooked to computers, and there was no means to get at a database from a terminal. I felt that the people out there who needed to communicate were not thinking about it from a query point of view. I thought from the point of view that one computer should be able to generate a transaction and send it to another computer, or to a person, so the notion of databases and communications were critically joined. We'd done a good job on the database side, but we'd done nothing really for the communications side. I wrote that specification. I don't have a copy of it. I probably wrote several different versions of it, but nothing came of it.

HAIGH: Actually, it's funny you say that because another thing that I had found that seemed to be expressed for the first time in the Database Task Group report is the idea that as well as providing a separate application and ad hoc interfaces that it should also work with batch mode or with interactive real time applications. That again there should be a system that provides these two different kinds of access to one shared database.

BACHMAN: Transaction-oriented access is what you're really saying—real time access. The real dichotomy is between operational systems and decision support systems, i.e., reports and queries. This distinction has been brought into sharp contract with the advent of data warehouse databases, which are separate from the operational databases.

HAIGH: Well, okay: provides the kind of access that a real time program might want, which in terms of a mainframe system that sends something like reservations is a transaction. A system that well as being able to handle some kind of large update run is also just be able to give a quick answer to an immediate question. That, to me, sounds like an idea that has a lot of resonance with what you described the WEYCOS system as doing. Do you remember having any--

BACHMAN: I'm sure I've talked to the people about the Intercommunicator idea. It doesn't show up in that form with a label on it, but as I said, I was very sensitive to the fact that we were shutting out most of the world of what we were doing. It was just batch programs getting access to the database, it wasn't people getting to the database.

HAIGH: I know that System Development Corporation (SDC) in the mid-'60s had enormous amounts of resources committed to timesharing systems, had looked at trying to commercialize online interactive data management systems. Do you know if you were personally aware of this work or had seen any demonstrations of it?

BACHMAN: I was not. I knew other people in SDC through the SHARE meetings. There were a lot of people from there. I can't remember that discussion. It may have all been bound up in the SAGE project, which they weren't talking about outside, or at least not in the circles I was in.

HAIGH: Actually, we should remember that we need to come back and discuss DataBASIC.

Sticking with the Database Task Group, you'd mentioned that Warren Simmons provided the initial driving kind of impetus, for its foundation. By the time the group released any reports, William Olle was chairing it. Now I don't know very much about him. Could you describe him and maybe say how he became to be chair?

BACHMAN: I would say Bill became chair, was based upon his energy, interest, and people skills. Maybe it was through seniority or something. I exchanged emails with Bill last month. He is an independent software consultant, who does work all over the world. I ran into him in funny places like Sydney, Australia. He is very competent. He worked at one time for RCA, in the United States. I have been to his house outside London many times. I know his wife Greta quite well. Connie has been there too. He's been at our house. He was a very competent professional, who was active in the British Computer Society. I think he was largely responsible for my being elected a Distinguished Fellow - British Computer Society. I think of people who do things, I think Bill is the key point man on that one. Dick Canning was the key point man in my being awarded the ACM A. M. Turing Award. Though other people obviously helped in those things, you get a feeling that someone was the driving force behind it. In these affairs, you need a strong, well-liked sponsor. In fact, the email I got from Bill the other day (2003) was asking something about the DBTG. He said, "I don't know where Dick Schubert lives. I couldn't find Dick Schubert anywhere on the Internet." There are lots of Schuberts, but no Dick Schubert. That's not quite true. I found one reference to him in an article published by the Australian IDMS users group.

HAIGH: Yes. I think I found that article too.

BACHMAN: I found it by looking up Dick Schubert.

HAIGH: And you know, he wrote an article in 1972, "Basic Concepts in Database Management", which I think I may have cited in my own paper.[70] So he was another member of the committee?

BACHMAN: Schubert was not a member of the committee. It was Bill Olle, looking for Schubert, which caused me to look Schubert up.

HAIGH: Do you know Dick Canning?

BACHMAN: We have become very good friends. We saw them recently. They just moved to a place to a retirement community in Vista, California. We phoned and emailed things back and forth to stay in contact. He and his wife, Peggy, were here. Just a year ago they were here in Tucson.

HAIGH: So what do you think it was that made EDP Analyzer so influential?

BACHMAN: I think that his ability was to look at the technical world of computing and re-express it, so that a business manager, who was not a technical person, who was trying to manage his own data processing operation, could be conversant with the problems at hand and could better manage the people under him, who were doing data processing. He was looking for a high level of management. I don't know how wide the publication went. I don't know how many thousands of copies a month went out. It was quite a good success.

[70] HAIGH: R. F. Schubert, "Basic Concepts in Data Base Management," Datamation, vol. 18, no. 7, July 1972, pp. 42-47

HAIGH: Do you know how he did research for it?

BACHMAN: He talked to people. Traveled and talked. It was contact with people both within the industry and user side. He had a lot of discussions, finding out what was bothering them, where they're struggling. He wrote to smooth the transition to these ideas? That's about the most I can say for him. He's a good writer. He writes clearly and concisely.

HAIGH: Another thing that I noticed in your correspondence file was that after receiving the Turing Award that you were in contact with Alan Turing's mother.

BACHMAN: Yes.

HAIGH: Can you talk about that?

BACHMAN: Yes, that's an interesting situation. At the time, I didn't really know who Alan Turing was. Connie went down to Harvard or MIT and asked if they had any books about Alan Turing in their libraries. They had one book, a book his mother had written about him.[71] The book was not available in the United States, so Connie wrote the publisher in England, who at the time had three, four, five copies left. Connie ordered all the copies. She was going to send my mother and father a copy, and her mother and father a copy, so they'd have some notion of what this was all about. The agent of the publisher said, "Did you know that Alan Turing's mother is still alive and is living at a nursing home south of London?"

Connie and I were going to London, and so Connie and I rented a car and drove down to Guildford, Surrey, where she was staying and had a very nice visit with her at her hospital bed in the nursing home. She was under medical care. She was in her nineties and fragile physically. She appeared to be bed ridden. She was very sharp. She told us all about her son, but her knowledge of her son, of course, was as a boy, and a schoolboy. He had won prizes for his running. "After Alan left school, there was a "Turing Award" for running at Alan's school, because he was such a good runner." She didn't know much about what he'd done during the war or how he'd died—"He was accidentally poisoned," she said.

I don't know what she knows or believes, or what anyone knows or believes. I was confused, because some people think he committed suicide under pressure from the police. The whole community's reaction was very negative in those days, to people who were gay. Someone whose name I had heard, and I think I've met him since then, who, had all of Alan's papers. He was at the University of Edinburgh. He was going to read them, sort through them, and write some final paper on Turing's work. At that time, it had not happened. I don't know, maybe it has happened since then. I think we mentioned last night there was something very substantial--

HAIGH: Yes, but that biography came out in the '80s. Let me see. I believe the title of the original edition was "Alan Turing: The Enigma of Intelligence", and the name of the author is Hodges.[72]

BACHMAN: I don't think that was the man she was mentioning.

---

[71] "Alan M. Turing, by Sara Turing, W. Heffer & Sons, Ltd, Cambridge 1959

[72] HAIGH: Hodges, Alan Turing: The Engima of Intelligence. New York, NY: Simon and Schuster, 1983.

HAIGH: No, I don't think he had a personal connection to Turing.

BACHMAN: Okay. So this was just a very pleasant visit. Go and visit Mrs. Turing. . At Oxford University, there's a project now to collect papers and whatnot on Turing. Papers written to him or from him are very valuable, but they didn't have any interest in letters from his mother[73]. Whatever their job description was, that wasn't included. I have a letter from Mrs. Turing. I wanted to send it to them. The letter is still around here, tucked into a copy of her book.

HAIGH: After you got the Turing Award, do you think that changed the way that people within the computer science community dealt with you? Did you after that, say meet a lot more of the other Turing Award winners or find yourself invited to give speeches and things like that more often?

BACHMAN: I don't think so. I don't think it made a change. One thing that we haven't talked about in this time period in some sense answers where I was, because there's some time that you said was kind of blank in your record of where I was during the '70s. It deals with a period at the end of the GE years, a thing called Shangri-La, which was a big project that GE had to design their new computer line. That new computer project was taken over by Honeywell in the fall of 1970, so my main responsibilities for a couple years after the GE/Honeywell merger had to do with that new product line (NPL) development.

HAIGH: I'll ask you about that in a second. This is a question that has been sent to me from someone in SIGMOD. "Back in 1973 when you received the Turing Award, computers were being used by only a small part of the population. Today, only a very small part of the population does not use computers. Did you anticipate or guess in 1973 that the award would be as prestigious as it is today?" That's from Curtis Dyreson.

BACHMAN: I don't believe I did. I don't believe I thought how important databases would become, or even computers. Even though we had kept growing exponentially, it's hard to imagine a world as different as it was 40 years ago when the IDS was first being used in production as a database to all the things that are being used in databases now. In fact, there are databases in your telephone. There are databases everywhere; the world wouldn't know how to live without them. We would have to go back and live in a very different way. It was unimaginable that computers, the databases they support, and communications would be so broadly assimilated.

HAIGH: Actually, another part of the database community of the 1970s I should ask you about is the ACM Transactions on Database Systems. Were you involved with or aware of that in any way?

BACHMAN: No, I don't have much memory of that. I think the name is familiar, but I can't give you a strong feeling about it.

HAIGH: We're now ready to discuss what was probably the most important topic facing the database community during the 1970s: relational model and the high profile debate about the relative merits of the relational and network approaches. I wonder if you could

[73] I have an aerogramme that Sara Turing wrote to my wife, after our visit in Guildford, postmarked "Guildford 17 January, 1974."

begin by saying when it was that you first heard that there was such a thing as the relational model.

BACHMAN: I'm not sure I can date that exactly. It was ten years after IDS got started. Probably early to middle 1970s[74]. I don't know exactly when. I did see Ted Codd's paper, shortly after it was published.

HAIGH: I think that would have been a little bit earlier. I believe it was 1974 when your public debate took place.

BACHMAN: It's the one in Ann Arbor?

HAIGH: I can check that too. Here's the program for it. Oh, they didn't put the year on the program. I can tell you that this was the SIGFIDET ACM Conference on May 1st, 2nd, and the debate was on the 2nd in the afternoon. Let me check on another file to get the year.

BACHMAN: Does it give a location there?

HAIGH: The program is in box 17, folder 38 of the Babbage, although they do reserve the right to renumber these things.

BACHMAN: They reserve the right, but they should never do it. It erases too many pointers.

HAIGH: I don't think they do it much. Here we are. So I have some other documents here. You and Codd were both invited in January 1974 to take part in this debate. That's box 2, folder 14. There is a letter inviting you. Your reply to the letter dated January 14th to Gene Altshuler of the Stanford Research Institute. Then there's some other backwards and forwards relating to the terms of the debate and some correspondence.

BACHMAN: Having seconds and whatnot.

HAIGH: Yes, that kind of thing. Talking about the rules, the program shows that there was an hour and a half during, which you and Codd presented your models. Then there was a coffee break. Then there was an hour and a half of E. H. Sibley and Dennis. Tsichritzis.

BACHMAN: Oh, Tsichritzis That's the guy from our first deal in Montreal.

HAIGH: Yes, and it says "for C. J. Date." I don't know why. Maybe he was standing in. Then there was a break. Then there was half an hour of J. R. Lucking and V.K.M. Whitney.

BACHMAN: Lucking was…I'm trying to think. Tall guy. I think he was a subsequent guy from RCA, after Bill Olle was there, who was an IDS network model supporter. Who was the other person?

HAIGH: V.K.M. Whitney.

BACHMAN: I don't place him.

[74] Edgar F. Codd, "A Relational Model for Data for a Large Shared Data Bank," 1970.

HAIGH: Then there was half an hour of discussion, two hours of dinner, and then it says just 8:00 onwards "rooms reserved for continued discussion" where you debate. So they had it scheduled to run from lunchtime…

BACHMAN: The rest of the day.

HAIGH: Yes, they even expected eager debaters to continue talking after dinner.

The earliest reference I could find chronologically is a paper of Codd's that you were asked to review in April 25, 1972. There's a letter relating to that in box 17, folder 37. You were writing to Robert I. Benjamin of Xerox Corporation and enclosing some review comments on a paper of Codd's. It would appear that by 1972 you'd definitely been exposed to the Codd's ideas.

BACHMAN: Very good. That's a better fix than I could have told you by a long ways.

HAIGH: So do you remember anything about reviewing a paper of Codd's?

BACHMAN: Not specifically, no. The earliest things I can remember related to Codd's associate Chris Date. I received a letter from the US Department of Interior asking if I'd sign up for Chris as a needed person to come work for IBM, because of the shortage of his talent in the United States. Not knowing how those things really worked, I said no, I wasn't willing to. In hindsight, knowing about what those letters are in general, I should have said yes. Because I knew Chris, but I was in no position to attest there was a shortage of supply of such people, which was necessary to get a green card in those days. If it were a green card. Maybe blue then, who knows?

HAIGH: In retrospect, you regret not doing that?

BACHMAN: Yes, I do. Too bad because I liked Chris before, and I liked him afterwards. The advice I got from our in-house attorney didn't encourage me to do it. Yet, in hindsight from all I know, it was a reasonable request to have said yes to, and something I could have bragged about later.

HAIGH: Do you know if he got the green card in the end?

BACHMAN: He must have, and he probably doesn't know I declined to sign it.

HAIGH: Unless he reads this transcript.

BACHMAN: But that's all right. I'm sorry. I'm not embarrassed. I was advised not to, by the Honeywell lawyer.

HAIGH: Do you remember anything about the debate itself?

BACHMAN: I guess my best memory of the debate is that the two groups talked past each other. They really didn't hold a conversation. Both sides essentially presented two points of view without any understanding of how things were similar, how they were different, and why things were different in, let's say, 1974 than they were in 1964. The very fact of what disks were and computers were in '74 made different solutions feasible. To get the ease of use of the relational databases would have been intolerable in a 1964 computer.

HAIGH: When you say that both sides talked past each other, does that imply that you yourself at that point did not realize that the rapidly expanding supply of computer power would make relational approaches more feasible?

BACHMAN: I never thought about it that way at that time. That was not an observation I could have made in 1974.

HAIGH: So that's an insight that you came to later.

BACHMAN: In fact, recently I had a big discussion about relational databases at the Computer History Museum. The more I thought about it, there were different factors, different motivating situations that affected things. Change in personnel available to work on systems. Essentially the average skill of people doing programming system design in 1964 I would characterize as being higher than the average skills in 1974, simply because there's a much bigger population going into the field.

HAIGH: It's clear from these documents that you remained skeptical about the relational model at least until 1979.

BACHMAN: Well, I'm still skeptical.

HAIGH: But you remained skeptical that it could in fact ever be used in production systems to do useful work.

BACHMAN: To do high performance systems, and there's evidence in the field that's still true. It's just there's so many new applications that come in that have not been demanding in the light of both the hardware and software capability. In other words, the relational database was enabled with a change in hardware costs and speeds and sizes. The fact that the realization, which I really hadn't formalized well in my mind until this recent Computer History Museum talk[75] is that really what we're doing on temporal databases, and with Cord Blood[76], are just more of the same. We could not have dreamed of doing a temporal database anytime soon because the processing power wasn't there to support it, nor the storage capacity to be able to save all that data. We worked our tails off to conserve disk storage space, conserve computer memory space, and processor cycles, because there was a so little of all three.

HAIGH: Now one document that I see here from 1975 (this is in box 2, folder 17), it's a memo that you're writing to Stan Williams saying that you've reviewed the specifications for a new MULTICS database under development, which I believe eventually was the first commercially supported relational system[77], although being on MULTICS, I don't think it ever reached a wide audience. You suggested that the case for relational

---

[75] "Assembling the Integrated Data Store" April 16, 2004.

[76] Cbr Systems Inc., Cord Blood Registry, San Bruno, California

[77] "The Multics Relational Data Store (MRDS) was first released in June 1976. This is believed to be the first relational database management system offered by a major computer vendor, namely Honeywell Information Systems, Incorporated. The designers were familiar with, and influenced by, the work of Codd, the Ingres project at U.C. Berkeley, and the System R project at IBM San Jose. . . . . When MRDS was released in June 1976, it was actually marketed as one of two components of a package called the Multics Data Base Manager (MDBM). The other component was the Multics Integrated Data Store (MIDS), which was a CODASYL database implemented as a layer on top of MRDS." quoting a reference on the web found while when looking for Oris Friesen.

technology there was debatable, and it wasn't clear if it could be developed. Do you remember anything about that MULTICS system?

BACHMAN: I do not. No. I don't remember at all.

HAIGH: That was in a completely separate part of Honeywell?

BACHMAN: First GE, and later Honeywell. Well, MULTICS[78] work, most of it was done originally at MIT, with Bell Laboratories cooperating. Russ McGee was the GE chief representative at the Bell Labs. Russ McGee keeps coming back into the story we're talking about[79].

HAIGH: I believe by about 1972 Bell and MIT[80] had dropped out of core development work, and Honeywell was left pretty much on its own.

BACHMAN: And supported it in marketing. I don't know how broadly marketed.

HAIGH: There was a letter, which I'm not sure I have with me, but was from 1979, and contained a sentence of advice to a colleague saying "I advise you not to pay too much attention to the relational model."

BACHMAN: Sounds consistent. My memory is of Charlie Clingen claiming that MULTICS was or had a database system. When asked to describe it, he described the MULTICS file system. A file system is a database, a specialized database for files and directories, but it is not a database management system.

HAIGH: "It's interesting from a research point of view but it will never amount to anything commercially." Can you remember was there a point at which you thought, "Oh, maybe there is a large class of commercial applications for which the relational model might be applicable"? Not necessarily be the best way, but at least it would be feasible to use it.

BACHMAN: Obviously there is a time to accept reality. At some point it was obvious that the relational databases were a success. For whatever reason you might come by to believe it, whether it's marketing reasons or the fact it's broadly commercialized, or the fact the academic community got on board with it.

If we look at the object-oriented systems, they essentially have abandoned the relational model, as the basis of their object-oriented programming. They're back to a network model because the relationships they support for the object-oriented programming. All of the data modeling tools are based on the network data model. If you think about the problem from a conceptual point of view, the network model is the survivor simply because it allows you to say more. Or say more, because you're communicating with

---

[78] MULTICS was originally to have been developed by an equal partnership between GE, MIT and Bell Labs.

[79] Russ McGee said (January 2006) "I too do not remember the relational database that Haigh is referring to."

[80] Russ McGee said (January 2006) "The MIT contingent was in the form of Project MAC, which was funded by ARPA. I believe that Project MAC stayed with MULTICS pretty much to the bitter end. At one time there were slightly over one hundred MULTICS systems in the field. They remained in operation into the 1990s."

somebody else, and you want them to learn more about what you're doing. The relational model doesn't suffice for that purpose.

HAIGH: So you're thinking about a tool like Business Objects where, in order to represent the real world entities, the relationships between tables are specified in some kind of data description language?

BACHMAN: You need to express those relationships in a way that is obvious to the people you are communicating with. Let me tell you one story which might be interesting here. This is back in General Electric days, so we're back now in the late 1960s. We had a group of customers come through, as we did all the time. It turns out this set of customers happened to come from a large department store chain in Germany. They had three technical people, and a man along who was their interpreter. It was quite clear to me that the technical people really spoke English, but they didn't choose to speak English and maybe embarrass themselves. That happened frequently people would come and understand what you are saying, but were hesitant to ask questions in English, so they would ask in German in this case. I started talking to them, and said, "Well, when I think about a department store," the first thing I talked about their store, so I drew a box in their store. In fact, this represents one store or many stores. One of the characteristics at the department store is they have departments. So I drew a box representing a type of entity. They were all saying, "Stores have many departments." The kind of thing that anyone who's gone to a department store and has any notion at all what goes on could have done. The Data Structure Diagram got bigger and bigger. We had buyers that were at the national headquarters. Somewhere in the middle of the discussion, the head of the German delegation actually said in English to his interpreter, "Would you please ask Mr. Bachman whether he is a department store expert?"

It was assumed, by the department store customers, that I was a department store expert, because I could say things about their business, using the Data Structure Diagram, that matched what they understood. It communicated. Data Structure Diagrams communicate, as a network model, which survived strongly today in the graphics representation world[81]. Network oriented database management systems have survived in the highest performance database applications. Products, like Object Store, are used by the big telephone companies to handle their bill-making. I am told that British Telecom still has the second largest commercial database in the world running on ICL's version of IDMS, which is almost identical to the IBM version. Both versions were marketed by Cullinane Database Systems. Where the requirements of the application are demanding, you have to declare the "joins of enduring interest[82]" to able to use them efficiently. One point of the argument in one sense goes back to the debate. I was arguing really from the basis of demanding applications, which was where my experience was, and probably did not have

[81] Using Microsoft's SQL Serve Enterprise Manager, I can reference a relational database and ask for a data structure diagram to be generated from its data definitions.
[82] The declared owner/member relationships of IDS/IDMS are created and stored when member records are stored. They are the equivalent of a pre-declared, permanent join in the relational world. With IDS/IDMS, given an owner record, the list of member records are directly known. With a relational database, given a primary key of an owner record, the entire table of possible members must be searched for find the that owner's member records, or else a special index must be first searched to identify the members.

a good feeling at all certainly, of what less demanding applications how they could use something that was easier for them.

HAIGH: As I understand what you're saying now and what you said in the debate, you had two important reservations about relational systems: one of them being this performance issue, that for a demanding application they would be too inefficient; and the other one being a more philosophical objection that the information that's there in an entity relationship diagram or in your Data Structure Diagrams about relationships should also be there in the data definition language rather than specified at query time.

BACHMAN: Correct.

HAIGH: Can you remember if you had any other reservations that didn't fall into one of those two categories?

BACHMAN: I think those were the two major ideas. I think one of the issues that came up at that time was the ability to rapidly create or adapt an application. Let me put it this way. To do ad hoc things, add on, add on, add on, and add on, the relational model is much easier, because you didn't have to do as much planning. I probably played down that role because I thought it was an improper way to work. I think one of the reasons that relational databases really have been so successful, and they have been extremely successful, is that data processing managers have become more and more impatient. You can show some results sooner doing a little bit of an application which you could fix up, add onto, and evolve over time. In doing it that way, you could show success and show some results, and it's not a two-year project, to get a big system started. It is an answer to the boss who says, "I can't wait two years for something; I need results now."

One more thing. I think about relational systems as being report systems, a relational database is a way to generate reports, queries, or information that is in the database; they do that far more easily than IDS did. For creating a transaction oriented system and building the database it is not clear. You can do some things, in a network database that you can't do, or couldn't do, in a relational database. Things like handling recursion. The relational SQL did not integrate well into a programming language. It was an awkward marriage, and is still an awkward marriage. In other words, if we were doing a payroll with relational database, it's a struggle to take the program which calculates the payroll and make it fit in.

HAIGH: I can see that that's a difference between the CODASYL approach, where the data manipulation language was supposed to be seamlessly integrated with the host language, and the SQL approach coming out of System R where the query language was originally designed to be used in an ad hoc basis by end users and programs essentially just embed chunks of this alien language or generate it at run time and have it interpreted. Do you think that has anything to do with the difference between the data models themselves? Or is it just a historical accident that the first usable relational system had SQL attached to it but that the network models were designed more for use by programmers?

BACHMAN: I think that that is inherent in the inventors of both systems. It would fit their mindset. I think what Ted Codd did made sense to him, the way his mind was organized.

I don't think Ted Codd realized how big relational was going to be either. He had something he thought was mathematically clean and pure. In fact, there are some things that no production database system would allow him to do, if he had all of his constraints in place. Because if I want to do a join and I only want, let's say, the city column as the only one I want to select on. I have 100,000 rows in my table. What I want to do is SELECT, selecting only the city name column. In a real database, you don't want to lose all the rows with each city name, except one, because all the columns that makes them different have been dropped out[83]. That's not what people want out of a database system, although that was consistent with Ted's mathematical model set. His mathematical model required substantial modification to really make it work to be a successful database system.

HAIGH: So that's a difference in approach that you think could, to some extent, be attributed to your different backgrounds. That you were working as a systems man with a real application that needed a lot of data, and he was coming to it from a more formal and mathematical approach.

BACHMAN: Absolutely so! I don't know how many people, 40 or 100 worked on System R, men and women, turned the relational model into a prototype database management system. Four people specified IDS, built IDS, and turned IDS into a successful database management system.

HAIGH: Can you remember if you had met Codd in person prior to the 1974 debate?

BACHMAN: I'm sure I had. I couldn't tell you where. I probably met him several times at different meetings before that.

HAIGH: How would you describe your personal relationship?

BACHMAN: I think we had a good personal relationship. I think that I may have known Sharon, his second wife, before Ted knew her.

HAIGH: How was that?

BACHMAN: She came to one of the Database Task Group meetings. It was held in Phoenix. I would date it in 1965, or early 1966. I think it was an early one too because I know that I remember it happening it Phoenix, and Warren Simmons was at that meeting. Sharon Weinberg was there. It was long before I was aware she knew Ted and before she married Ted. I had met Ted separately.

HAIGH: What was your impression of him as a person?

BACHMAN: Perfectly reasonable person. We had strong differences in ideas on what was right and wrong. One because we were competitors. IBM versus GE. And also because we had different experience backgrounds to work from, which would cause us, whenever we were together, to be debating in one sense or the other. It didn't mean that I wasn't comfortable with him. After the meeting, we'd go out and have dinner together. I don't know if I'd say we'd want to have drinks together, but that's because generally I

[83] In relational theory, every row in a table, as stored or as generated by a SELECT statement, must be distinct from every other row.

don't go out and have drinks. I always felt comfortable with him. I felt more comfortable with Chris Date, who was not as stiff a person as Ted Codd was.

HAIGH: One of the other papers that's preserved in your archives is your ballot form for the 1980 Turing Award nominations. There were a number of candidates, and you gave Codd four votes of out 28, so you gave more votes to each of Dennis, Hoare, and Wirth. Can you remember what your thinking was at that point?

BACHMAN: I guess one could look at that as being a commercial vote and voting against IBM as much as anything.

HAIGH: I should point out there were several people on the ballot that you didn't give any votes to at all, so it wasn't an entirely negative vote.

BACHMAN: I didn't really know Codd. One has to ask… The vote was spread around, but you could look at this as being a negative vote, four out of the possible twenty that I could have given…. If I looked at that now, I'd say I was probably biased, maybe biased for good reasons, but still biased.

HAIGH: That's very interesting. So was the bias, as I think you just implied a second ago, that IBM was a commercial competitor and that you saw Codd as a representative of IBM?

BACHMAN: I think that certainly was part of it, and that relational databases were competitive to network databases. I think there probably is as much commercialism in that vote as anything else. It all runs together though, the continuity between what's commercial interest and what's intellectual interest.

HAIGH: Actually, you attached a little description, so this is what you said to the committee at the time. "I support Ted Codd's nomination on the basis of his work on the relational approach. It has had a high impact within academic circles. However, the current inability of these ideas to move from the concept stage into production makes me hesitate. On the other hand, Alan Turing wouldn't get elected if this were the only basis of merit." That suggests that at least the reservation that you admitted to at the time was at that point that it hadn't been proved.

BACHMAN: Commercialization had not been successful, had not justified the validity of the model. It took a long time to get it out into the marketplace and heavy usage, so it sounded like that's a good first paragraph.

HAIGH: But, in retrospect, you think there may also have been a commercial consideration?

BACHMAN: I'm sure there must have been. Must have been.

HAIGH: Now you also mentioned over lunch that there was a SQL like interpreter produced on an experimental basis for IDS.

BACHMAN: Yes. At the GE computer department by a man named Oris Frieson. He was one of the engineers that worked in the same group that Russ McGee worked in, doing advanced work. In fact, he had SQL translator up and running before I even heard about the project. He'd done a very handsome job of getting it all together. You don't need a relational database to handle relational queries. You can do it another way. You just translate the SQL query into an IDS program and compile it. Then run the compiled program and get your query answered. I can't precisely date it, but it would have been late in the years of GE in the computer business, so '69, maybe even early '70[84] before the Honeywell merger. The merger was probably May of '70. It took six months to do all the things they had to do to put it all together.

HAIGH: So do you know would that have been compiling the queries into COBOL and then running those through?

BACHMAN: I'm not sure exactly what they did. It was probably clumsy in the end. In some sense, it was a proof of the fact that you could take a SQL[85] query and turn it into an access to an IDS database.

HAIGH: At some point I want to ask you about the status of relational technology at Cullinane in the early '80s, but I'll ask you about that later. Let's move now back more to your own career and talk about the transition from General Electric to Honeywell, which took place during 1970 with the exit of General Electric from the computer business.

BACHMAN: This is probably a piece of the story, this transition from GE to Honeywell. It probably started a year earlier, actually because General Electric organized an enormous big product planning meeting that was called Shangri-La. Actually it was a meeting of, let's say 100 professionals plus their families held in an hotel in Hollywood Beach, Florida. Many of the people were living in the hotel. They were living in apartments around the hotel in the summer of 1969. There were people from Phoenix. I was one of the people from Phoenix. Stan Williams was one of the people from Phoenix that came to that meeting. There were people from the Schenectady group of GE Laboratory. There was a group from Paris from the Bull-General Electric operation, and a group from GE Italia, which was based in Milan, Italy. The purpose at that time was to essentially design a full line of computers all of which were internally architecturally compatible with each other but would have let's say small low-cost versions, faster middle-cost versions, and then a high-end range, to be produced in Milan, Italy; to be produced in Paris; to be produced in Phoenix, Arizona.

HAIGH: So that would have been a competitive response to IBM's success with the 360 series.

BACHMAN: Yes, very similar to the 360. In fact, there was a lot of architectural similarity. It was going to become a 32 bit machine where the previous GE machines were either 36 bit machines or 24 bit machines. This would essentially get in line so you can now build something which is more competitive. One of the things they wanted to do is be able to move over existing IBM software and run on it.

This happened over three months. This whole group in Florida, finally a subgroup of that, maybe half of the technical people, without their families (because it was the end of the summer so families went back to put their kids in school), finally wrote the final report. What happened is GE top management looked at the final report, looked at the cost to

[84] My current thought is that my introduction to this project was after the Honeywell merger in 1970. I remember visiting Phoenix and having a discussion with Oris.
[85] SQL has not actually been developed by 1970, though Codd's first paper on the relational model itself was published in that year.

build the new factories, the marketing costs. The investment to do this added up to a number that GE felt was unacceptable. Ralph J. Cordiner, who, at that time was the chairman of the board and driving power of GE, had a theory: If you can't be at least number three in your industry, you should get out because you can't make money. We were not number three in the industry. Unbeknownst to me, the group started looking around for someway to sell the GE Computer Division, or do something, merge with somebody else to get bigger, in other words to become a number three. Through that winter, there were negotiations that led to Honeywell buying the GE computer business and merging it with their Honeywell Information System division. They were not going to buy it with cash, they were going to buy it with Honeywell stock, with an agreement that GE couldn't sell the stock in the beginning, but could over time liquidate their interest in it. They were concerned about antitrust issues. Each of sales were about of 10% of the US market. Would a merger be acceptable? In the Spring, the merge was announced. I don't remember the exact date, but maybe March or April, with pending merger to be finalized in October.

In the end, this merger did get finalized. My family and I moved from Phoenix, Arizona. Actually, Paradise Valley, which is a suburb of Phoenix. We moved to Lexington, Massachusetts, which is adjoining Waltham, Massachusetts, where the Honeywell Information Systems were headquartered. The main thrust of this thing was the understanding that this new product line had been specified at the GE Shangri-La meeting would in fact be the vehicle that the new merged computer business would now go forward on. The next two years really most of my activities at Honeywell, because I moved at the base of the move; most of the people in Phoenix stayed in Phoenix. Most people in Phoenix stayed behaving like they were still GE Computer Department, and didn't really pay much attention to the new product line because they had their non-compatible product line to keep going.

HAIGH: What was the group that you joined in Massachusetts?

BACHMAN: It was the Honeywell Information Systems, and there was an Advanced Software group, in which I was the manager of the Database Management group.

HAIGH: Was that seen as a research type position or as more of a product development responsibility.

BACHMAN: That was a product development group. We would develop the database software for the new product line. I brought with me about six people who had been part of the group that I'd worked with in Phoenix, so the new kids in that group came from Phoenix to Waltham to work on the project. It was mostly Phoenix people working for me at Honeywell in Waltham. Almost all of them ended up going back, one after the other, to Phoenix where they continued to sell aggressively the old 600 GE computer, which became the 6000 Honeywell computer and which kept making minimal profits over a period of years and continued to struggle to hold its marketplace.

HAIGH: So what was the product that you were developing?

BACHMAN: The new product line had the “L6,” which was assigned to Honeywell, Italia, in Milan, Italy. The “L7” was assigned to Honeywell Bull, in the Paris. The “L8” was the Waltham product. The “L9” was going to be the Phoenix product, which was never built. Waltham had worked on the L8 part of the line, which was never built. I spent two years writing the product line specifications and starting the L8. The L6 and L7 got built in Italy and France, respectively, and went to the market. L9 never got built in Phoenix, and L8 never got built in Boston.

HAIGH: So your product was specifically targeted at the L8 model?

BACHMAN: Well, it would be the design for all of them, but the implementations probably would be modified because of fixed limitations. We were still in an era where the biggest Honeywell 6000 (GE 600) computer was still 32,000 words[86]. It's hard to think of 32,000 words. That might have been a big computer back then, but no longer a big computer by any measure.

HAIGH: So space was still at a premium?

BACHMAN: At a premium, and the smaller ones are going down from there.

HAIGH: So did any of the work over this two year period ever find its way into any product?

BACHMAN: Never.

HAIGH: What was it you were working on? Was it an improved version of IDS?

BACHMAN: Actually, what we were doing was writing specifications and trying to write some code for a machine that didn't exist, which eventually all just got thrown away.

HAIGH: If it had reached the realization stage, would it have been based on IDS?

BACHMAN: It was an IDS product, yes. It was the IDS, but it wasn't a multiprogramming IDS, based on what was learned at the WEYCOS 2 project at Weyerhaeuser Lumber.

HAIGH: Apart from this added multiprogramming capability, can you remember any other important advancements that it would have represented?

BACHMAN: I don't think so.

HAIGH: So if that took two years, that would have kept you busy until about 1972.

BACHMAN: Yes.

HAIGH: Through out your time in Honeywell, did you remain in the same group?

BACHMAN: During that two-year period, yes.

HAIGH: How about over the 10 or 11 year period at Honeywell?

BACHMAN: At the end of that period there was a new group set up actually in the same building at Honeywell, an advanced systems group. A man named Mike Canepa was the manager of it. It had some of the people in terms of one of the people that I mentioned earlier from IGE. A man named Bob Blose was in that group. Henry Lefkowits is a name you mentioned in one of those papers you were looking at. He was in that group. Bill

[86] Thirty six bits per word. One machine instruction per word.

Helgeson was in that group for a while. This was kind of a group of people looking at new things.

HAIGH: So was that group conceived as a product development group, or was it more of a research group?

BACHMAN: More of the information systems type of research, in exploring new ideas.

HAIGH: How long did that group stay together for?

BACHMAN: I have to stop a moment and think because it's kind of a blur where that all went to. I think at one point we moved from that location in Waltham out to Billerica which was the suburban place six or ten miles away from Waltham to what had been designed to be a manufacturing building, which never did any manufacturing. It was all built along the notion that they were going to manufacture these new L8 computers out there. I was experimenting with new ideas for IDS and for network databases and new features for them, although it's hard right now to pinpoint what those features are.

HAIGH: Over the period from '70 to when you switched into this more research world to '81 when you left Honeywell, regardless of what the group was called or exactly who was in it, did you stay in the same kind of roll of trying out new ideas and data models and things?

BACHMAN: Yes. Now, you see, one of the critical points about 1975, the issue came up about computer communications again and the formulation of what became the International Standards Organization subcommittee and Open Systems Interconnection. SE16. I had been working within Honeywell on some ideas like this that we called Honeywell Distributed System Architecture.

HAIGH: I've seen some references to that in files.

BACHMAN: We had developed some very strong ideas. I had started to project how we could put this kind of capability into the Honeywell computer line. Then we became aware of the fact that actually the British delegation at the fall of 1975 (I think '75) world meeting in I think it was Buenos Aires, or Sao Paulo, or someplace in South America. They had put up the notion that we should have a new subcommittee on Open System Interconnection. We had no standards on how computers talked to computers. Actually at that meeting, the American delegation, led by IBM, voted against having such a study. It seems The US didn't need those kind of standards because IBM had their SNA, which they were happy with, their System Network Architecture. The ISO/Technical Committee 97 voted to set up this new subcommittee. Then the US delegation, in the typical US way, said, "Well, if we're going to have the new subcommittee, we'd like to have the Secretariat for it." The country, which has a secretariat for each of these standards committees, has some additional power of organization, bookkeeping, and the fact that the country of the secretariat also is the country of the chairmanship of the committee. At that time, I had suggested to our organization at Honeywell that we ought to get involved in this subcommittee. Then we could influence the new international standards, based on the Honeywell distributed system architecture (HDSA).

HAIGH: What had been your personal involvement with the Honeywell distributed architecture project?

BACHMAN: I was the one who was pushing it and doing most of the documentation on it.

HAIGH: How far did that project progress? Were there any prototypes built, or did it remain at the paper stage?

BACHMAN: In the paper stage. One thing I might just say to preface all these comments: at the time I left Honeywell, I had a bookcase full of documents from about six or seven years of work. I kept very extensive records of what went on in that committee, which when I left Honeywell I took everything that I could with me, but they wouldn't let me take any of the stuff on the communications work. They said they would save it and put it away for me. It disappeared, so there was a gap, and a whole lot of things that would have gone into the archives. There was a very complete definition of what the American group did and what the International Standards group did. All of the drafts that I had written all completely disappeared, much to my annoyance.

Anyway, this thing was going to happen in terms of the SC16 being established. A meeting was set for its first meeting. The natural thing to happen is for the US to set up its own ANSI subcommittee to support the international committee. The vice president in charge of standards activities, at Honeywell, thought that we should not get involved at all, just ignore it, which I thought was a big mistake. Being difficult, as I can be, I called a friend, Warren Simmons[87], who was involved in the ANSI SPARC subcommittee and asked him if he would like me to be the chairman of the US committee supporting SC16. He had me appointed chairman of the US committee. The guy who had voted against it in Minneapolis came along as a delegate, but I was a chairman. He disappeared after the first meeting, and Honeywell continued to support the sector thereafter.

HAIGH: Did other countries have their own national committees in parallel with yours?

BACHMAN: Yes, it works that way, yes. Every country had its own committee which does work at home and sends a delegation to the international meetings.

HAIGH: In the event, how much of the final international standard would you say was based on work by the American committee?

BACHMAN: Well, I would say that 75% was based on my work at Honeywell. It was all well documented. We worked, as you sometimes have to do, to cover the traces. Essentially the Honeywell Distributed System Architecture (HDSA) document was rewritten. HDSA became DSA and was put forward as the American position. Over a sequence of meetings held by the international group, this same thing was really reworked into what because the International Standard. It is essentially architecturally consistent with the beginning. The idea of the seven level towers, that one slide we looked at, was already in place. The IBM's SNA has similar structure, except it didn't go up beyond to the presentation controller layer. It didn't have the application control layer. We built on its concepts.

HAIGH: So this document is labeled "International Standards Organization Working Paper." I'm not seeing an exact date on it, but I believe it's from some point in 1978. You

[87] Warren Simmons, of United States Steel, is the man who initiated the CODASYL List Processing Task Group that became the CODASYL Database Task Group.

can probably see the date from the correspondence in another file, and this is in box 2, folder 25 of the Babbage collection. Around page seven is a hand-labeled sketch which shows the levels running seven “process control,” six “presentation control,” five “session control,” four “transport control”, three “network control,” two “link control,” and at the bottom “physical control.” As far as I can remember the final OSI model, I think it’s pretty much the same except they don’t have “control” in the name of every layer.

BACHMAN: Yes, right. This model carried through right into the final standards. Essentially, each layer in effect represented a different scope of responsibility. At the process control, you say, “Well, how do I talk to some other fellow or sister application process? What information do I want to exchange for our common purposes?” My downward connection says, “How do I tell the presentation control layer what I want to do to accomplish with this communication?” Layer by layer it follows that pattern. What’s changing, as go down each level is the scope. Someone is responsible for each stage of the trip. Someone is responsible for offering a certain quality of service. If that quality of service is considered to be inadequate, then the next higher level says, “Okay, what is within my ability to compensate, for any errors that might be happen in the next lower level.”

HAIGH: Now was your interest in this field of distributed systems coming from the idea of a distributed database, which I know was a popular topic of discussion during this era.

BACHMAN: No, I think it was generally the notion that we can think about business or the world being a set of cooperating workstations or cooperating processes, and how do we get from here to there? Now, we look at this thing, we have skipped over one big project that we need to go back to.

HAIGH: The ANSI SPARC database thing?

BACHMAN: Yes.

HAIGH: Don’t worry. I have that on my list, but as you’ve started spontaneously into OSI I thought we’d do that first.

BACHMAN: This model we’re looking at here, the seven level model. The diagrams of the seen level model look like the twin towers of the former World Trade Center, in New York. They are indicated as having seven level or layers. The notion of having layers like this certainly came from looking at the SNA document from IBM, documentation, which we’d looked at early in the Honeywell distributed system architecture project. We know we have to do this, we have to this, we have to do that, but there are things missing in here. During the work on this project with the groups in the various countries, I pushed the idea that we should be looking at the interfaces between each of these levels. Each one is requesting the next lower level, to use its services. There are the same kinds of things happening with a connection to be established, and how do I establish a connection? How do I deal with a connection? Essentially, you had the generic architecture all down through the layers, even though the physical implementation of each layer may be somewhat different. You could conceptualize it as being a different implementation on the same concept.

HAIGH: In other words, the thing that you should be standardizing is the interface between the different layers?

BACHMAN: You have to have the interface standardized as well as having the horizontal parts standardized, because in order to get the bottom to work for you, you've got to be able to talk to it. The assumption that any physical control thing had to behave at its interface the same, and had to do for you what your protocol said it had to do, was interface between layers and a protocol between companions at the same level.

HAIGH: In 1978 as well as this formalization of the concept of seven different layers, I also saw a number of drafts for what should be happening at the specific layers. There's one for transport services, one for virtual terminal protocol, one for virtual file protocol. Do you have any recollection of where the content in these protocol drafts would have come from and if it finished up being influential?

BACHMAN: These are all top level (Application Level) protocols, because essentially some process wants to talk to some other process about this file transfer. These were all application layer protocols. After I left the activity here, I know that they finished up some of these file transfer protocols, and there are international standards that have been established for them. My overall impression is that we did all this work, and in the end, no one's paying much attention to it because some de facto things came along out of the US government standards that have been used instead.

HAIGH: Back in 1978 when you were coming up with these series of protocol drafts, did the content in here also come from the Honeywell project? Can you remember other sources for it?

BACHMAN: The Honeywell support of this project disappeared except for my time, because they were saying, "Let's wait and see what standard comes out." So they did nothing within Honeywell to follow this up. And these standards were still being pursued after I left Honeywell, into the early 1980s.

HAIGH: But the question is more, by 1978, you already had drafts for these things, so that was working quickly. Can you remember if these were modeled on work that people at Honeywell or in other companies had already done, or did the committee come up with them from scratch?

BACHMAN: I think people derived ideas from what they had, whether it was in France or Italy or England. There were a lot of active people working on these things, so they were really committee results. Typically, committees bring things they know. They don't invent things out of whole cloth.

HAIGH: Do you remember anything about the atmosphere on the American committee? If there were any other individuals who played a particularly important part?

BACHMAN: A man named Dick desJardin, who was from NASA, was one of the key leaders in our group. Dick took over as the chairman of SC16 after I left. A couple of good people from IBM, who had worked on SNA, were very supportive of what we were doing. A good man from Univac whose name I can't think of now, there was a good bunch of people working on it[88]. IBM, because of the conflict with their SNA commercial interests, was on the cool side about the whole thing. They were working, helping, but they were not aggressively pushing things forward. I think the cooperation among the various national delegations was excellent. One of the requirements to get a standard in place there has to be an English language version standard, and there has to be a French language version standard before it really finally is accepted. That sometimes takes a long time.

HAIGH: As chair of the American committee, was it your job to go and meet with the representatives of other national committees and pass on the thoughts of the American committee? How did it work?

BACHMAN: As it turned out, I was the chair of the American committee and the ISO committee. All the communication really happened at semiannual meetings, plus communication from the working committees, on various things, who were working with their counterparts.

As chairman of the American committee, I became de facto the chairman of the ISO committee, unless the American committee nominated somebody else, which they didn't do. That's how I got to be in that position, which I held for four years.

HAIGH: Were you pleased with the results of the committee's work?

BACHMAN: I was, yes. I am disappointed that it didn't end up being significant.

Official standards had to be written in two languages, English and French. Among other things, French are very fussy about using the French language in standards meetings. Two things. First, when the French delegates made their presentations at the meetings, which had to be translated into English with translators, they would always write their reports in English. The members of the French delegation read their report, written in English, and speak the report in French. The translator would read the English report, and repeat it in English. The French delegation wanted, what they had to say, to be clear to everyone who understood English. Even though the French national rules required the to speak in French, they wanted their message came through clearly, in English.

In the same kind of cooperative sense, when we wrote all the drafts, they said, "There are things you can do with the English language. Some ideas can be expressed in either Latin based words, or Germanic based words. "Always use the Latin based words when writing these specifications, if you have a choice, because it'll translate much more quickly into French." Six weeks after we completed our overall architecture specifications, written in English, we had the French translation of it. We knew what we were doing, and the people who were working were so tightly integrated in what we were doing that we weren't going to get hung up on having a French translation. The French delegation weren't going to get hung up on being slow in their translation. That's a level of cooperation we had was very, very high.

[88] When I left Honeywell in 1981, they impounded all of my ISO/OSI and HDSA files as being proprietary information. When I tried to retrieve them later, I was told that they had been lost. I had carefully saved all of the historical records, all of committee work, all of the drafts, all of the working papers, and all of the names, but they were lost.

HAIGH: I saw a memo in your file, I think from about 1980, which I think was around the time of your transition from Honeywell to Cullinane, saying that you weren't being released as much as you would have liked to have done to attend meetings perhaps that they would need to find a different chair after your term expired. Do you have a feeling that your work on this committee was not being fully supported?

BACHMAN: This letter had to be written when I was at Cullinane. My feeling was that what I was doing on the communication standards really wasn't in the best interest of Cullinane, because they had no piece of the action. When they hired me, they inherited that job with me. I thought it was appropriate for me to get out and actually turn it over to the people who really had a full time interest in it. That was voluntary on my part, not that I thought that I was not doing the right thing or not being supported. My feeling was that Cullinane had been very supportive, maybe more than they really had to be, so I wanted to avoid any semblance of being unresponsive to my duties to them.

HAIGH: So then you were happy with the level of support you received from both companies?

BACHMAN: Yes. And the degree of freedom they gave me to operate. Now in some sense, a degree of freedom implies either great support or indifference. I think in Cullinane's case it was indifference, and in Honeywell's case it was relevant to what they were doing and they were happy to continue in a way that I had a great deal of control over.

HAIGH: It seems that the committee made very rapid progress in the early days around 1978. My impression is that things got bogged down at some point. That they didn't move as smoothly in the 1980s.

BACHMAN: Well, it's harder to write detailed specification than it is functional specifications. In a functional specification, you can paper over differences rather easily. You'd have different interpretations of what they mean. When you get down to defining what things are at the bit level or the byte level in the headers and descriptors, it gets harder to get full agreement in a national group that's meeting twice a year. So I think a good deal had to do with the fact that there was a level of details that you really had to actually specify and get agreement on that is inherently slower at the final part than the first part. It is like writing the introductory chapter in a book. That's pretty easy. Putting all the rest of the chapters behind it is difficult, and tying it all together of course.

HAIGH: Now at this point in the late 1970s when you were chairing the committee, was there substantial input from telecommunications bodies, or was it confined to computer manufacturers?

BACHMAN: No, it was interesting. We had people from AT&T and telephone companies, and all the national delegations had both telephone people and computer people. In fact, in the beginning, one of the problems was that they didn't know how to talk to each other. Essentially, the telecommunication people were dealing with really point-to-point communication. That's what they did, and tying together the same thing through relay points was not something they worried about. In terms of trying to integrate their thoughts about architecture and their approach to telecommunications standards was one of the learning curves we had to go through. Both of us learning some of the other side's points of view, because they had never concerned themselves with computers talking to computers; that wasn't their business.

HAIGH: You said that there was a learning curve. Do you think that by the time you left the committee that the two groups were on the same page?

BACHMAN: I think close enough that it wasn't a problem. They were still specialists in their field whether it was point-to-point communication or network communication.

If you think about where we were when we started, the way that computers used telephones, you picked up a handset and put it in a box and the computer talked to this handset, and it talked to another computer at the other end of the handset. Or to a teletype machine built in the handset, which is really point-to-point communication. How they might route data: there was a connection from their point of view all the way through their network to the other end. That was a strategy. Packet switching is a very different way to do things because there is no connection from end to end. There's only a connection from here to the first relay point. It's got a connection to the next relay point. The shifting to the notion that we could do packet switching was relatively new. Even though packet switching had been introduced and talked about in the telecommunication world, we weren't really doing much of it.

HAIGH: A criticism that I've read of the final versions of the OSI protocols for things like email, and file transfer was that they had more of a telecommunications mindset in as much as they put a great deal of complexity into the network itself rather than aiming for a simple network and putting the complexity into the devices at the end of the network in the way that TCP/IP and the Internet protocols do. Do you think that's a fair way to distinguish between the two approaches?

BACHMAN: I know that TCP/IP is what survived. My suspicions is that there were enough ad hoc things built on top of TCP/IP, by the time the standards were published, that TCP/IP was like an iceberg coming across the ocean—you either get out of its way, or it's coming. I think that if our committee had been faster with people using the systems…. But the Federal government was funding the TCP/IP work, so there was a lot of momentum in it. Sometimes we were too little too late to catch up with the pragmatic approach to things that came out of ARPA and out of the military systems of communication. The comparison you made, that it was comparatively too telephone oriented, I really can't address that. I don't understand it quite that way. I don't understand that criticism enough to comment on it.

HAIGH: As I recall the argument, in the case of something like the X.400 email standard. The final standards were extremely complex and described all different kinds of data types, and it, for example, expected the system to handle conversion from one format to another if it was required. Very complicated rules for email addresses, with fields to deal with every possible eventuality. Whereas the more ad hoc Internet type approach would really focus on providing a much lower level of functionality in the network and leave most of the stuff that had to do with actually understanding the formats of the things that were being transmitted and those kinds of issues with software running on the computers at each end.

BACHMAN: If I look at that translation problem, this is where the presentation control layer came in. Two responses. One that while I was involved, I never heard the word "email", or if there was any intent to do anything like email because we didn't know what email was; didn't have any intent to do it. Therefore, email was fitted into that framework or tested into that framework, later. The notion that a human can translate from French to English by reading the email, caused some to think that architecturally the presentation layer ought to be able to translate from French to English for you, or one hardware format to another.

HAIGH: Not translation in terms of human languages, but translation in terms of richer content that may not be readable on the device that it was being transmitted to. Those kind of things. Obviously, that was later.

My impression is that the most influential thing from the work of this committee and its various successes laboring away for quite some time, in the end, was more the idea of the seven-layer model itself than the specific protocols that were eventually created within the OSI framework. It was interesting to me to see just how early on the layered model seemed to originate.

BACHMAN: I think another part of that thing is that part of the vision, certainly in my mind, is that a great deal of what open system interconnections dealing with was messages being sent between programs. The Internet was largely people to people communication. I think that the underlying thought process in the OSI was this was computer process to computer process communication, or people to computer process, so that it had a different requirement on how to behave. In fact, you might think of it as much like the system on the operating system, the transaction between operating system that we developed for MIACS that there were different people sending transactions where they go. You've got to be able to have them and pass them out. It was essentially a message passing system where the recipient was a program or something to print to somebody.

HAIGH: When you say "process to process system", are you distinguishing from an approach to communication that would be thought of more as a machine to machine or operating system to operating system approach?

BACHMAN: Well, "application program to application program," "person to application program" or "application program to person," but not Internet as we know it today. In fact, Internet today is largely ad hoc communication[89]. The ISO thinking in 1978 was in the world of preplanned communication with transaction between companies who are sending purchase orders back and forth, sending shipment information. Modeled, largely, now I think of it, on the mindset that came out of the manufacturing systems that built IDS and built MIACS. How do we essentially have corporations cooperate with other corporations doing their business? The EDI[90] stuff about how to do communication between businesses is what they mainly focused on. Some say the ISO/SC16 specifications didn't work well for email. If it had worked well, it would be a happy

[89] February 25, 2006. I believe that thought was too narrow. I have seen a great deal of person to application program lately and know better.
[90] EDI, Electronic Data Interchange.

accident. I don't think the goal was to make email…I don't think it was even conceived that email would be important or that browsing would be important.

HAIGH: But wasn't that one of the ideas behind the separation of layers? That if a new application comes along at the high level, you don't have to go and reinvent the lower levels to support it.

BACHMAN: That's true. However, if you think about browsing, then the question is which processes are we talking about? Am I sitting here typing as one of the processes, which has to get to someplace else? The Internet Protocol says that I can type this format and put it in line and it will go someplace. If that fits into the model, whether it's process-to-process communication, or some surrogate process looking at the email address, it can figure out what to pick out from the destination machine to send back to you, if you want to browse. Browsing had not been conceived of, or at least it had not surfaced with any force, at the time we were working.

Let me finish up by saying I don't think we've gotten communication to the point, where the communication systems were intended to go. I think we still have unfinished business in that sense. It's easy for companies to say, "Okay, I've got this set of business transactions, and I need to send them to you." Part of what they do, of course, is through XML, and I can send you a transaction, which is self-describing. I think about the difference between data description, which we both agree to use, and I will transmit something to you, which is formatted in that description. You can interpret that description. XML says, "Well, we've got lots of processor power. I'll send the description along with it. Every record is self-describing. If I leave out three fields it's not going to bother you because if you try to access them, they won't be there, but the fields I did send you, you can read." In the continuing adventure of using computer resources, one way to skin the cat on how to use descriptor control things is to send the descriptor along with every item. Therefore, XML in some sense is replacing the EDI work we were doing where people were saying, "Let's define our formats outside and send things according to that format." It's something again where the hardware/software cost performance has shifted, made one set of ideas obsolete, and replaced them with another.

Now, the place where they're still hung up is that we both have to have some notion of what the XML in the description is for the purchase order we want to exchange. Even if I don't send a complete purchase order, you at least know what I'm going to call the purchase order code, my purchase order number, my name and address, you'll know what those names are. If I use a different name than the standard is, it's not going to work.

HAIGH: Returning to the question of what were the key ideas that you believe were present in the early work of the committee. You've identified the concept of the seven-layer model itself, and the concept of communication oriented around application to application or user to application communication. Do you think that there's anything in those thoughts you outlined that reflects a third or an additional stream of thought that was present in the committee's work, or do you think it is subsumed in the general idea of process-to-process communication?

BACHMAN: I think that the general feeling, the best I can measure it and remember it, is that there was a notion that we were dealing largely with sending messages between programs that belonged to companies doing business, or between divisions of companies who were doing business. As long as these programs behaved in a standard way, responding to certain predefined message types, all was OK.

HAIGH: So really the motivation was coming from these EDI type applications…

BACHMAN: I think so. I'm not sure they were called EDI in those days.

HAIGH: …and, in contrast with the Internet, not for doing things like a remote login or an arbitrary file transfer?

BACHMAN: An arbitrary file transfer was probably part of it, but something which would happen in the context. We did file transfer descriptions at the same time, but they were considered process-to-process communication. That's why I said they're application layer protocols.

HAIGH: So you thought that something like a file transfer would be happening as part of a larger business to business process?

BACHMAN: Well, no. I'd say it differently. Say that in addition to my business processes one to 100 or one to 1,000, I have some system processes that involve file transfer. My file transfer program wants to talk to your file transfer program, this is the protocol which I send to yours, and yours knows how to store that file and retrieve that file from your computer. In a sense, it doesn't look to anybody else. It's different than an ordinary EDI application. It's just a standard EDI application that deals with file transfers instead of transferring purchase orders or transferring whatever else you might want to transfer. It had been generalized into that model. It turns out if you look at email today, it doesn't really fit that model very well, or if it does fit it, it's purely accidentally. In other words, that application wasn't envisioned in my mind. Or what's called browsing today.

HAIGH: That's true. Although it's also true that browsing wasn't envisaged when designed TCP/IP.

BACHMAN: Each format beyond the level doesn't make a difference.

HAIGH: Yes, TCP/IP is pretty much agnostic about the content of the packages that are being transmitted.

BACHMAN: It gives me end-to-end communication from a physical point of view from one computer to another computer, and you get your next lower level doing your packet switching for you. So all those things are really innocent of what's going on, except carrying out their communication responsibilities. That's why TCP/IP is different than the transport control that we specified, but it was an alternative. I'm going to say it wasn't as elegant as the one we designed. That doesn't mean it wasn't better. It had the problem that it belonged to somebody else. It was talked about, but it wasn't a lead candidate to become the transport protocol, because we were trying to make all the levels behave similarly from an architectural point of view.

HAIGH: I think a lot of people have forgotten that in the early '90s, even the groups that were running the Internet believed that transition to OSI protocols were inevitable, and were making plans to transition away from TCP/IP. Then suddenly the Internet took off so much.

BACHMAN: It overwhelmed.

HAIGH: And surprised everybody.

BACHMAN: We probably still could use TCP/IP to do the transport protocols; it's would just have a different interface to match up to. Replace the transport protocol because I think down below that is still standardized.

HAIGH: I believe TCP/IP doesn't fit neatly into the seven-layer model, but it's there somewhere in the middle.

BACHMAN: You mean it's kind of an awkward stepchild.

HAIGH: I think that actually went into use around 1981.

BACHMAN: It was well known in the military before that, though. It was one of the things on our plate to consider.

HAIGH: I think that wraps up the OSI.

So having discussed the OSI committee, before we talk about the ANSI SPARC database standardization work, let's just return to Honeywell and address anything else you worked on there other than the ANSI SPARC database standardization work. Now looking at the tapes, I noticed references to your concept of the Role Data Model, and I also saw references to something called the Data Independence Project. If you can speak about whichever of those logically comes first.

BACHMAN: All right, let's talk about the Role Data Model situation. The Role Model, if we think back to what we're talking about, we've talked about network models, we've talked about the relational model, we've talked about the hierarchical model. All those models in some sense are based on what I think as being a "triple." Something looks like a record which ties together little elements of information about something, and then each of those elements talks about being a name or an address and it has a value. One of the struggles we had in terms of trying to build an integrated data system was to try to collect all of the information about something, so that we could process it.

The original vision we had with IDS is that you could build a database, with data about everything. That turns out to be too big a task, and the systems were too inflexible to be able to add things easily and change them. People have done a couple of things to try to solve that problem. One is they build separate systems and try to move data from one system to the other, and not try to keep the old system in place. Go through whatever conversions of that data's necessary to get the data from system A to system B. At some point, we will talk about what we tried to do at Constellar Corporation, in California. That deals with building a product to help solve the problem in another manner.

We were talking to some of the early banks that were using IDS for their database system. Banks at that time had checking accounts called "demand deposit accounting," open ended deposit accounting. They had loan accounts, they had mortgage accounts, they had personnel records. But they had no way of telling you whether a person who's got an account with them for an automobile loan was also a depositor. They had no way to take things apart. He says, "Well, you should generate one big record for all the people who had all these aspects they could be in." At some place, this is impractical because people do too many different things. The idea is saying well, if we were to characterize each record as representing some role that a person or a business is playing, and then we could have a record, which is kind of coherent about how they behave as a demand deposit customer. If we have to have a different kind of record already from someone who is a home loan customer, so you had some way to stick these together. Or say if I'm looking at a home loan customer, is he a demand account customer? We could better service the customer because that's what the banks say. "I will service my customers better so they'll stay with me more and give me more business."

HAIGH: Tell me if I'm understanding this right. It sounds something like the idea in ER notation that you have a super type, and then you have several optional subtypes that are nonexclusive. So one record could have none or a bunch of these different things reflecting different roles. Except that you would want to do this on the actual data model level, not just on the more conceptual level.

BACHMAN: My answer is that inheritance is not the same concept as role orientations. They are orthogonal concepts. They are different, because we're using inheritance to model a totally different phenomenon. We use inheritance, as we would factor algebraic expressions, to achieve the most simplified form of expression. We factor out common descriptors of various kinds, which are then used, by reference, by two or more record types. By sharing data properties and relationship properties, and procedures that reference the shared properties, the total amount of descriptive information and procedures to be recorded is reduced. The chance of errors and inconsistencies is reduced.

The Role Model is responsive to situations where objects of two or more classes represent the same "real world entity." We only know of a real world entity or business entity, when we perceive of its existence through some role that it is playing. That role might be "Doctor," "Patient," "Demand Deposit Account Holder," "Mom," "Supplier," "Property Owner," "Resource Unit," etc. I would store role-oriented information about a real world entity as a record of a record type, a row in a table, or an object of a class, in my database.

If a real world entity were to appear to me, in a role, there two questions to ask. First, is this real world entity already known in this role? If it is already known in this role, then use the existing record for it. Second, do I know this same real world entity, in a different role? If an analysis shows that both roles represent the same real world entity, then I will record a "sibling relationship" between the two records, two rows, or two objects that represent them.

Now, if I access a record, I can access all of the sibling records that represents more information about that same real world entity. There is only one constraint in the Role Data Model. A real world entity cannot be represented by two records of the same record type, i.e., representing the same role. It is easy to understand how a real world entity could appear as a "Demand Deposit Account Customer," and/or a "Small Loan Account Customer," and/or an "Bank Employee." It does not make sense for one real world entity to be represented by two Bank Employee records.

We have moved from the point where an entity type in a conceptual schema had no defined meaning in the real world, to where it represents one of roles that are played by a real world entity. Now we have a semantic meaning to sibling records.

HAIGH: That sounds to me, actually, not unlike the capability in the Chen ER notation that you mentioned earlier that you can attach some attributes to a relationship.

BACHMAN: Only superficially. The Role Model's entity is not the same as the Chen ER model's entity. It is represents something that is a "real world" entity, or a "business world" entity, which may appear, in the information world, as represented by one, two, or more roles. These roles are much more like Chen's entities, or the entities appearing in the Partnership Set model.

HAIGH: To return to what I said earlier, is the different that you wanted the database management system itself to understand these things? So not just at the level of drawing a diagram and then implementing that in some other way. That you wanted a database management system that natively understood these things.

BACHMAN: That's correct. In fact, I have not thought of any good way to create a diagram that represents this. When I think of that big Data Structure Diagram, over there on the wall, the objects of any one of those classes, represents a role that could be played by some other object up there. There's no restriction regarding the combination of roles that may represent a real world entity. The only limitation is what you observe in a real business.

I don't know how to draw a diagram that says two or more objects may be potentially connected, but you can't have more than one of each type. I have not imagined a diagrammatic way to represent that, because they're totally unconstrained as far as connection.

HAIGH: I think I can imagine how to do that in a Chen style ER diagram with enhanced notation[91], but I won't do this with the tape running. Aside from that, maybe you could draw it in the ER notation, but that wouldn't help you understand to implement it on the database.

BACHMAN: In the database, every time you create a new role-oriented record, you ask whether is this a record about a new real world entity, or a new record for a known real world entity entity? An associated is recorded between each entity and the records that represent. So that entity can keep collecting the different records that relate to its existence and behavior. This gives you a very dynamic way to be able to assemble all the things you know about some one thing that you consider to be the same entity. It also leads you to the fact that you have to be prepared at some point and say, "Well, I've got two entities I know about, but I just found out they're the same entity." So you may want to merge their roles. You learn more about things over time. Now you've learned it, let's associate them as one real world entity. Or, the entity you may have thought is now seen as two entities. You now have more information, and you want to split the roles of original entity between two entities. This requires a database capability which has not been observed any place else, and that seems very helpful toward building a more comprehensive picture of what's going on.

---

[91] Essentially by using the "shared subtype" concept, and attaching attributes or relationships available to any entity to a subtype shared between all the other entities. That would make for a messy diagram, but neater than the alternative. This seems to be rather an obscure ER technique, but is documented in T. Connolly, C. Begg, and A. Strachan, *Database Systems*, 2nd ed., Addison-Wesley, 1998. However it is not in later editions of the same text.

We are implementing a role-oriented database at Cord Blood Registry[92] right now. In one area, we are focusing on all of the people that Cbr interacts with. At the first cut, these include: “Doctor,” “Nurse,” “Cbr Employee,” “Baby,” “Customer,” and “Other Person.” When we began to interact with a person, we first determine their role and ask whether that person is already known to us in that role, or another in our list of roles. If it is a new role for that person, a new role-oriented record will be created and that record will be associated with the other roles known for that person. If that person is a new real world entity to the system, the new role-oriented record will be noted as the first known role for that real world entity.

In the Cbr Systems business, they collect umbilical cord blood at the time a baby is born. This blood is processed to extract the stem cells, which are frozen against the day when that baby, or a close relative, needs a stem cell transplant, for medical reasons.

From a process planning and scheduling point of view, any of the persons known to the system, in one of the person roles listed above, may also appear in the role of a “ResourceUnit,” which may be assigned to one or more “ProcessStepPlans,” over a period of time.

Almost all of the record types (classes), defined in the New Database Architecture (NDBA) project, at Cord Blood Registry, have been declared as being role-oriented. Thus, they can be joined as supporting the same real world entity. We have been surprised at some of the combinations of record types that have appeared as representing the same real world entity. Given access to a role-oriented record, all of the sibling roles of that real world entity are easily accessed.

HAIGH: Looking at some records here from the file, I see in 1977 it seems that you were working on this topic. So you had sent this little letter out saying, “Attached is the first of a series of papers, which you will be receiving over the next several months entitled ‘Why Restrict the Modeling Capability of CODASYL Data Structure Sets?’” And that a paper would be following called “The Role Concept in Data Models”. You also say in that letter that this, in a sense, is reviving something that you had started work on in 1972. This is the ’77 document here, and then you’re attaching something from ’72 that you say is related about project comprehension. Anyway, you think that your main work on this role data concept at Honeywell was happening in 1977?

BACHMAN: Yes, definitely. It reminded me of something earlier, but there was nothing done subsequently.

HAIGH: Were you working with anybody else on this concept?

BACHMAN: I was working alone.

HAIGH: Did you ever get as far as implementing any kind of trial system?

BACHMAN: Not until now, 2004.

HAIGH: Are you aware of any influence that the papers on this topic had at the time[93]?

---

[92] Cbr Systems, Inc. i.e., Cord Blood Registry.

[93] There have been several papers published by Friedrich Steimann on the representation of roles in object-oriented and conceptual modeling.

BACHMAN: I am not aware, although I know that I've talked to people. In fact, I talked to my son-in-law who's the vice president of research for the National Association of Broadcasters who also had responsibility for their data processing system. He said they had that same kind of problem, same kind of requirement in dealing with their customers who are broadcasters of television and radio stations. They built an ad hoc system let them do that, paste these things together, so they could get a broader profile of people. Each role someone plays, in combination with other roles, makes the profile I have of them. In his situation, he said, "We recognize that same problem, and we did this about it," in their system. It was a homemade system using existing capabilities.

HAIGH: Do you see a close connection between this concept and the idea of a conceptual schema, which I think I'll be talking about in a moment with ANSI SPARC.

BACHMAN: No, I think it's totally independent.

HAIGH: Then how about the data independence project? It's something else I know that you were working on at Honeywell.

BACHMAN: At the moment, I cannot recover in my mind what that was about unless you have a clue for me.

HAIGH: Let's see what I've got here.

BACHMAN: Unless it deals primarily with the conceptual schema, which I had to deal with that problem also.

HAIGH: Here we are. Data independence project. It seems that you were working on it with Manilal Daya.

BACHMAN: He was South African from Indian extraction.

HAIGH: I'm seeing a letter here from the 12th of March 1979 which is box 3, folder 1.

BACHMAN: This is later than the three schema invention.

HAIGH: It's actually headed "Data Processing vs. Information Processing." So you're saying that research began with the idea of creating data independence, and that it seems to have evolved into something else. I think what you're trying to do in this memo is actually say that the role data model and the conceptual schema are all part of this larger data independence project. Maybe it was just a way of packaging it internally.

BACHMAN: Certainly this thing is tying back into the conceptual schema, which is the earlier project. We're going to talk about it. What more does it say down here?

HAIGH: So it's got a wide distribution. There's a distribution list there with 30 or so people on it. You attach a document. Is the attached document actually in this file or not? Oh, it's an enormous distribution list, but I don't have the document in this file.

Anyway, whatever this was, you're not aware of it having gone anywhere.

BACHMAN: I don't really place it in my mind right now what was going on. I'd have to see more before I could reconstruct it for you. Certainly, it's keying me in on this other thing we haven't talked about, which is data independence and that ANSI SPARC study group that went on and on and on

HAIGH: Here's another document relating to it. It's box 2, folder 23, 1977, and it's headed "Limitations Imposed by the Absence of Conceptual Schema". I'm seeing that this data independence project was maybe a name that you used internally to describe this work related to conceptual schemas. That would be my guess from this. You see it's a title that you're using and the subject line. This is '77 December 30th. "A presentation is scheduled on the current progress of the data independence project. This work represents the continuation of the ANSI X3 SPARC report on the development of the conceptual schema."

BACHMAN: Okay, that's the area we need to get to.

HAIGH: Let's transition then to the ANSI SPARC project. If you could maybe, begin by describing the origins of this project and the original motivation.

BACHMAN: Okay, this project started out of the motivation, which was totally different than what they actually did, or maybe I should say that its advertised intent was totally different than what they ever did. I would characterize it as a project that was set up to develop a standard for database management that could be part of COBOL and so on. I would think that many of the participants who were drawn from IBM shops had no desire at that time to standardize IDS or its equivalent, and were not particularly interested in standardizing relational databases either. It was a bunch of bright guys, who thought that they could do something interesting.

HAIGH: As the CODASYL standards were already in existence, why was there a perceived need for another standardization effort?

BACHMAN: The CODASYL report from DBTG was not a standard; it was a report. There was an important different between being a report and being an ANSI standard.

HAIGH: Then as I understand it, work continued within CODASYL on standardizing the language. There was some kind of standard in the CODASYL COBOL Journal of Development, I think about '74, that was at least supposed to be a standard.[94]

BACHMAN: I can't speak to that. I have no knowledge of it.

HAIGH: That would, I think, have been a standard for the Data Manipulation Language anyway. Was the ANSI SPARC committee intended to standardize all aspects of database management systems?

BACHMAN: I guess I would say that, as I understood the thrust, the original intent was to take what had come out of the ANSI SPARC Study Group on DataBase Management Systems and to propose a standard as an extension of COBOL. SPARC, which was ANSI's Systems Planning And Resources Committee, was a steering committee under ANSI. The individuals, who were involved with the Database Management Systems but were people who were against the network model based database management ideas.

---

[94] HAIGH: I have not been able to full verify this, but I believe that following further work from a subcommittee CODASYL issued a series of standards in its Journal of Development. A COBOL DML was published in the CODASYL Journal of Development in 1975. A new DDL standard appeared in 1978, discussed in M. E. S. Loomis, "The 78 CODASYL Database Model: A Comparison with Preceding Specifications," in Proceedings of the 1980 ACM SIGMOD International Conference on Management of Data. New York: ACM Press, 1980, pp. 30-44. It appears that the 1978 standard saw the DML functions dealing with physical storage split off into a separate Data Storage Definition Language.

HAIGH: Sorry. There's a list of committee members here from 1973. I'll zoom a little bit on that.

BACHMAN: Here's Jon Turner; he's a Columbia University guy. He ran the data center. Here's Thomas B. Steel, Jr., Prudential Life Insurance, a very elegant guy. David Smith, Exxon. I don't remember Puerling. Ed Sibley, I know. He's been around a long time. Good man. University of Maryland. Chuck Mairet, John Deere. Bill Florance, Eastman Kodak. Fred Kirschenbaum, Equitable Life. Larry Cohn, I can remember the name, but I can't quite place him. I think it was with IBM.

HAIGH: This is box 2, folder 9. Actually, the description that was given in that document that you wrote is "The committee was set up last year to survey the database area and to recommend to SPARC what actions seemed appropriate visa vie standardization.

BACHMAN: The sense of the group was that there was nothing worth doing in terms of standardizations, so what other interesting thing could we do.

HAIGH: I'm seeing in the reports though that there was some attempt to at least work out what could be standardized. You identified 28 interfaces as the things that might need standardizing.

BACHMAN: If we think about the ISO work on open system interconnection as being an architectural approach that led to a number of interfaces and protocols. This is the logical precursor of that project.

This was essentially my first experiment. In fact, there was a bunch of diagrams that go with this that you have there somewhere in that file—big drawings that illustrate these interfaces. There are descriptions of them. Simply saying all these things fit together in a rational pattern, and different people talk to different places. At that time in the 1970s things were people-to-people interfaces. Some are computer-to-computer interfaces.

HAIGH: Here's a diagram I think showing all these different interfaces.

BACHMAN: Which I drew.

HAIGH: Let me see if there's a file reference on that one. This appears to be appended to something from June 25th, 1974 with subject "Status Report: ANSI X3 SPARC Study Group Database System", and that's in box 2, folder 14.

BACHMAN: Okay, well this work was done, which I thought was an outstanding piece of architectural work, and really a milestone in terms of the first time that kind of approach had been looked at in any way. This was used extensively in the IDS/OSI work.

HAIGH: Did the standardization work get much further than drawing the diagram and identifying the interfaces?

BACHMAN: No, there was a final report written by Dennis Tsichritzis, from the University of Toronto, who got all the publicity, even though he wasn't on the committee. But he put in the time to summarize the work, so he earned it. I was annoyed that I hadn't written it. I had the opportunity. I missed the opportunity.

But essentially, look at the difference between the storage system, the database system, and where people were talking different things for different purposes. This box here was a person in the role in one of the programs, and if we look under processing functions versus people functions, and interfaces were two different parts that talked to each other, and the need and use of being able to-- there's standardization at these places, so this is really an architecture diagram.

HAIGH: Did systems like IDS have all of those interfaces, or would some of them be for things like ad hoc querying that not all systems necessarily supported at that point?

BACHMAN: I think this was not IDS specific. It was supposed to be generic and identify interfaces that would appear in all systems. Let's just say that this worked for IDS, as well as other systems we looked at. We were not trying to address this model architecture with just one system. To the best we could understand, it would fit other database systems, such as relational databases and IMS databases. It introduced some interfaces that were not known to exist for every database management system.

It dealt with who talks to whom about what.

HAIGH: Back on the August 6, 1973 document, I see it says, "The critical question before the committee is whether there is a defendable case for the interfaces as now identified and the system structure underlying them". Then you say a bit later that there is some equivalence between this and aspects of the CODASYL systems. Then you say, "The IBM people on the committee appear to believe that the structure adequately represents IMS and whatever its successor might be. The SHARE/GUIDE people on the committee believe it is sufficient to support the requirements set forth in their report." So as you moved forward did that kind of agreement hold up, or did you find that you couldn't really reconcile the different perspectives?

BACHMAN: I think it just kind of petered out. It was an academic exercise. It really was an industrial exercise, but it did some kind of work that most industry people don't do. It's attempting to take a broad picture and to publish it.

HAIGH: You awhile ago had been saying that what they finished up doing was something different, and I suspect what you're about to say involves the conceptual schema idea.

BACHMAN: Yes because this is what was called a three schema approach, which essentially said that there is a physical view of the data in the database, there is a conceptual view which looks at the database and understands the database, and there is an external view, a third view, which is what some human being would do. I at that time decided that's just another physical interface. It's physical for the person instead of physical for the storage device. In fact, if we have a ring of physical definitions, all of which could be understood by the conceptual definition. Then I can translate from any one format to another, whether it is program-to-program, person-to-person, or program-to-person exchange. So, the generalization of this thing was a conceptual schema in the middle of a ring of essentially many different physical descriptions of the same information structure.

HAIGH: But that's not how the idea was expressed at the time?

BACHMAN: No, it was expressed that the physical one, which was the external one, was considered different than the physical one, which was the storage one.

HAIGH: As I understand the idea of the conceptual schema that came out of this, the physical level was chains and hash tables and inverted lists and the stuff that is actually putting files on a disk or wherever else they might, but the logical level was equivalent to the data definition language idea that was around at the time. The conceptual schema was a new idea that was a higher level of abstraction.

BACHMAN: I would say that's not quite right. Because they said, the conceptual one is typically the information level versus the data level.

They're talking broadly, bundling what you call logical and physical together, because they're looking at the point of view what is storage, versus what's someone sees. The three views were: the external view, the conceptual view, and the internal view. The notion that, if each of the external views and each of the internal views is mapped onto the conceptual view, then one can translate between the internal and the external view, if one understands these mappings.

Think about there being these three forms of views. Now I decide that's interesting, but there are times when I've got a better understanding. Let's generalize the notion that all physical representations, whether they're for external purposes or internal purposes, are external descriptions. If I have two external descriptions for whatever purpose, if I can map them onto a conceptual schema, I can map from one to the other.

HAIGH: The term "schema." My impression is that this was used for the first time in the database context around the CODASYL Database Task Group. I haven't personally come across the term earlier than that in print. Does that fit with your recollections?

BACHMAN: You can attach it to a man by the name of "Tax" Metaxides. I believe he who brought the word into the group. He worked for Bell Laboratories. He was a very interesting guy, very bright guy. He was from Great Britain, but he was Greek in background. And well educated. He knew about schemas and things, and I said, "That word seems to be useful to put in here." I'm sure he learned the word from someplace else, but in this application we're talking about here, he's the first person to familiarize me with the term.

HAIGH: As I recall, the Database Task Group report also included the idea of a subschema, which, as far as I could understand it, seemed to be analogous to a view in a modern relational system.

BACHMAN: Kind of, yes. Simply put, I have a full schema and some part, that is less than all, is used with each program.

HAIGH: So for some purposes you can give something that's a subset. That's why it's a subschema.

BACHMAN: Yes.

HAIGH: By 1973, the idea of schema has been floating around, by that name for something like five years, and the idea of a schema and a subschema have already been written into the CODASYL report. So, was there any clear relationship between the schema as defined in the CODASYL report and anyone of the three levels in the ANSI SPARC hierarchy?

BACHMAN: Probably the best correlation you could make in the schema, without qualification, from CODASYL, was an internal schema, a storage schema. In contrast, the conceptual schema showed the information content, in contrast to how it's represented or how it's even internally structured. The conceptual schema was a step above it, in terms of being unconstrained by database or communication limitations.

HAIGH: Was there anything in earlier systems or earlier work that you think corresponded to that idea?

BACHMAN: Not that I'm familiar with.

HAIGH: From you've said, my impression is that you personally expected that database management systems would be enhanced to directly support the conceptual schema. That this wasn't just a modeling technique, that it was something that would eventually find its way directly into the systems.

BACHMAN: Yes, in a sense that if we talk about what Bachman Information Systems did later, we were pursuing that direction, so that the data modeling tools that we built were considered conceptual schema descriptors and not database schemas. We dealt with saying, "Well, we have a high level model. We can forward engineer it using our software from the conceptual view to a database schema that fits either CODASYL design, or a relational design (or a relational design for IBM versus Oracle), for physical schemas (although there's more physical things, deeper layers). You don't get how the blocks are put together and whatnot. So the notion was that the conceptual schema would come into being and be a useful mechanism as part of ongoing systems. I think that that's happened. There are forward engineering tools today, in most modeling UML systems, such as Rational Rose. They would take my UML model and convert it into a relational database design. Several different venders, including Bachman Information Systems, have done this. BACHMAN was the first.

HAIGH: Certainly the whole CASE tools idea was to automate or assist that process. But isn't there a distinction between the idea that a database management system itself is going to support a conceptual schema, and the idea that there are going to be tools which assist in the process of moving from a conceptual schema to something that a database management system can understand. As far as you can recall, back in the '70s when you were talking about this stuff, were you thinking of something like a computer-aided tool to help you transform a conceptual schema into something that the system could cope with, or did you imagine that there would be a new generation of systems and a new data model that would encompass this conceptual schema stuff?

BACHMAN: Probably the thought was not sufficiently focused in our mind at that time to give a clear answer to that question. We could understand that there were distinctions, and how those distinctions and how the mappings would be created stored, and manipulated. It was not yet worked out. Not until the '80s did we begin to solve those problems.

HAIGH: Yes. In fact, there's an interesting paper I found in your file called "Database Standards Today, Tomorrow, or Manana". This is box 7, folder 12. There's no date on it. It looks to me like it was written about 1982 in connection with the National Bureau of Standard's expression of interest in standardizing database things. The way you structure it is really that there's a scenario where you could attempt to standardize the CODASYL work. There's a scenario where you could try and modify the CODASYL work to make it a conceptual schema language, or there's an approach where you can wait ten years for there to be a real conceptual schema language and then standardize that. You talk through the different scenarios and then say, "It'll be awhile before we've got the real conceptual schema language, so we should go ahead and standardize something like CODASYL right now in time for it to be useful." As I read that paper, it appeared by about 1982 you were still assuming that there would be at some point a conceptual schema language that would not just assist but would actually replace existing database definition languages.

BACHMAN: I don't think "replace," as much as to be used in conjunction with it. In other words, use it as a more semantically complete language to be translated into the CODASYL type specifications or put it into some other specifications, given the implementation parameters. Bachman Information Systems played out that game over the '80s.

HAIGH: So, as far as you remember it now, there wasn't a substantial shift in your assumptions from the '70s to the late '80s on this matter?

BACHMAN: No, I don't think so. The value and the usage of a conceptual schema became clearer and more obvious.

HAIGH: Do you remember anything about how the idea of a "conceptual schema" was received at the time, or responses to it when it was released and published?

BACHMAN: I think it was received very well because people could understand it. They could distinguish between what things looked like conceptually versus what they looked like physically. People could buy into the idea without knowing how it might be implemented, but they could understand it. I think it was very well received in the industry.

HAIGH: Beyond that initial reception, what do you see as its enduring legacy?

BACHMAN: Well, I guess it's enduring legacy, is if I talk about my own situation, is that we built a company to exploit these ideas: Bachman Information Systems. In fact, even before I was working at Cullinane, Data Structured Diagrams were something I wanted Cullinane to automate. During the second year I was with them, I spent most of my time working on developing a data modeling tool that would use Cullinane's IDMS, as its meta database. I don't think my effort was fully understood. What was not fully understood was how they could make money with it. That's an important distinction. They were funding relational systems at that time. I had a little idea, and relational things were the big idea on their horizon.

HAIGH: Well, we'll talk about those more personal aspects in a minute. But more broadly than that, do you see any features in today's database management system packages that are directly influenced the concept of a "conceptual schema" as articulated during the ANSI/SPARC project, or do you feel that the influence has gone into products like the ones you've mentioned and things like Rational Rose.

BACHMAN: I think definitely they've gone into other products, which sit on top of, and are independent of particular database systems. They have the ability to take their data model and translate it to meet the requirements of various different database systems. The independent software suppliers set out to make it work that way, to make it work better. They started, by doing data modeling tools and various Data Structure Diagram formats, and then recognizing that those models were not tied to a particular database system. It was inappropriate to tie them, because most prominent database systems were relational database systems, which were not semantically as rich as the data modeling tools were. Therefore, the need to forward engineer, as we talked about earlier, was a critical one. That corresponds also to the notion that the way to start your data model is to take your existing data descriptions and reverse engineer them. Essentially take out the physical model and performance enhancement factors. You can look at the foreign keys and deduce from them that there are relationships that should be declared. So you could take a relational database model and reverse engineer it to create a network model from it.

HAIGH: Is there anything else you think we should talk about from your time at Honeywell in the '70s?

BACHMAN: We've covered the three schemas. I think this wraps it up pretty well.

HAIGH: So how and why did you come to leave Honeywell and join Cullinane[95]?

BACHMAN: I guess out of frustration that Honeywell is not going to invest any money in any real changes in their database systems. They were struggling financially. It seemed like they were playing an end game. How do we sell as much as we can? Get as much money out of what we already built, and not be willing to really invest as a long-term survivor. Honeywell was not a long-term survivor. Therefore, I cast off. Where could I go that I might be in a sympathetic atmosphere? I called John Cullinane who said, "Can we talk?" So I went and talked, and knew very quickly that I was going to work for him. I don't know if it was clear at the time what my job was.

HAIGH: So had you known him prior to this?

BACHMAN: I'd met him a few times. I had talked at a Cullinane User meeting for him. Let's go back to the Cullinane thing for a moment. Even though they were selling IDMS, they were very selling it as the CODASYL solution that worked on IBM machines (even though it wasn't built to the CODASYL specifications). In all their product literature, they talked about "Bachman diagrams" and Charlie Bachman. They were not trying to distinguishing themselves from the broader network database system marlet, but were exploiting that market.

HAIGH: Is that why it was Cullinane that you called and not Cincom or another database company?

BACHMAN: Cincom had a very minor product, a very simple capability, where Cullinane's IDMS was essentially identical to IDS. They were in the Boston area, down the street, a few miles from where I lived and was currently working.

HAIGH: So what was your role at Cullinane? What were you hired to do?

[95] John Cullinane was the founder of Cullinane Database Systems, later renamed Cullinet Software that marketed the IDMS database management system.

BACHMAN: I was hired as the Vice President of Product Management. I sometimes the role that John really had in mind for me was for me to sit by the front door and meet the visitors coming in. "Charlie Bachman is here now."

HAIGH: So more like a chief scientist kind of role?

BACHMAN: No, like a figurehead, which is very different than a chief scientist.

HAIGH: I think sometimes they give that position to people who they want to keep around, when they don't know exactly what to do with them, but they're kind of prestigious and they want to show them off.

BACHMAN: I think that was the category. I was supposed to go out and give John's speeches. The problem was that I would go out and give my speeches, and that wasn't quite what John wanted. I think I disappointed John in some ways. I think, more seriously, that the relational databases that IBM was giving away when John was trying to sell IDMS became a horrible threat to him and eventually destroyed the company.

HAIGH: By the time you left Cullinane, was it apparent that the company was in trouble?

BACHMAN: They were in trouble, and I wasn't part of the solution. Maybe I was part of the problem, because I was still pushing network databases and they were trying to pretend that theirs was a relational database, it was IDMS-R. I was incompatible with their marketing message.

HAIGH: I'll talk to you about that in a minute. How would you describe John Cullinane?

BACHMAN: John Cullinane. Well, I have a picture in my mind of a several people who I have known, who have been very good salesmen. I have characterized them as "smiley-eyed Irishmen[96]." John fit that mold perfectly. I've known a several men who were superb salespeople, who are of Irish descent. They had a gift of gab, were smooth, and were very sensitive to customers. Really nice guy. Everyone loved John at Cullinane. He was not the kind of guy to fire you though. He had someone else do that. That wasn't his job. John was a good guy at understanding that you have to make a profit every year, knowing how you encourage people to buy something. Whether he had it or not, he'd sell it to you. Bill Gates did this same thing. Gates bought whatever he needed to sell to IBM. Well, John Cullinane started out with service kind of business, found something somebody could use, bought it, and turned around and eventually ran into IDMS, developed by BF Goodrich in their corporate business. They asked him if he'd like to market it, and he wrote a long-term contract with them. He did a superb job of marketing it. I continue to be on good terms with John, to today.

HAIGH: Let's talk about IDMS then. The first description I've seen of it is in a 1972 article in Datamation by Richard F. Schubert of the BF Goodrich Chemical Company. Schubert had himself been a Database Task Group member since 1970 at that point. It's interesting to me that a company that was not in the computer business was, as a representative of an end-user industry, involved with the CODASYL effort, and then went as far as actually implementing one of the first CODASYL DBMS systems. I understand at least the people involved with the system wanted to form a new division of Goodrich and they go into the database management market.

[96] Bill Rossiter, at Dow Corning in the 1960 and a good friend, is my prototype.

BACHMAN: The information system that Dick Schubert put together, using IDS for the B. F. Goodrich Chemical Division, in Cleveland, was a big success, in terms of a production management system. They had a very large, complex data structure that really worked well for their business, for a very complicated business. Then their corporate headquarters said, "This is really good work. Dick, come down here to Akron and do this for the rubber company," which was the big company at B. F. Goodrich. B. F. Goodrich Chemical had been a GE customer. Goodrich Akron headquarters said, "Well, you can't use GE equipment here. You're going to have to use IBM equipment." Dick, said, "Well, I can't do a good job here with the tools that IBM offers. Let us reimplement IDS, for the IBM 360 series. They built a version of IDS that was called, "IDMS" to distinguish I from IDS, and put it to work for the rubber company." B. F. Goodrich said it was worth that much for us to have to build it ourselves. Then having built it, the question was could Goodrich sell it. I'm not familiar with what Dick Schubert thought about having a new B. F. Goodrich division selling it. They ended up with Cullinane being that division, although they didn't own John Cullinane's business. But, they had a license fee agreement that meant they received some revenue. I don't know how much they got from it. I do know that John Cullinane latter resented how much he had to pay.

HAIGH: The only reference I found to that intent to sell it themselves is from an article from the Australian IDMS Users Group.

BACHMAN: Is that the one that I mentioned I found?

HAIGH: It's probably the same one. The way Schubert outlined the history was that they had been user of the GE 200, that they got IDS, then according to this article bought the rights and converted it to run for the IBM family, and used it--

BACHMAN: B. F. Goodrich Chemical didn't buy the rights from anybody. They didn't pay anybody any money. B. F. Goodrich was given two boxes of cards (4000), containing the IDS source code, by Clark Karcher of International GE.

HAIGH: Yes, the idea of buying the rights seems slightly ahistorical, in a period when people weren't really selling software. Now, of course between the mid 60s and the early '70s, the IBM unbundling took place, and it was apparent that software was something that you could sell. According to this article, IDMS was designed and written at the Cleveland Data Center of the BF Goodrich Chemical Company from mid '69 to mid '71.

Now what they say about the relationship with IDS is that "the only IDS source ever seen by us was the earliest version for the GE 200, and no code from IDS has survived into IDMS. We did, however, retain the concepts of file navigation based on tables and the owner/member record relationship." They say that "multi file databases, areas, schemas, and subschemas are all CODASYL concepts implemented for the first time in IDMS." "The IDD was our original contribution was not part of the DBTG specifications until years later. IDS for the GE 200 had none of these features, and for that reason alone it could not serve as the basis for IDMS. IDMS was all new design and new code." So that's what they have to say on the relationship of the two systems. Does that sound accurate to you?

BACHMAN: The only way I can respond to that statement is from my experience when I used IDMS at Cullinane. I programmed actively using IDMS for two years on the early development of the data modeling tools. I was designing those tools. I was writing them in PL/1, as the programming language to access IDMS. IDMS behaved exactly like IDS behaved. The IDMS database commands duplicated the IDS commands exactly, in their functionality. I used Cullinane's PL/1 interface rather than the COBOL interface because I had been using PL/1 extensively at Honeywell.

One of the things that they said in the Cullinane manuals was that you should not create recursive structures. In my own thinking, I couldn't understand why you could not use recursive structures, and so I designed something with recursive structures in it. Every time I tried to run it I shut down the Cullinane data center. We could not figure out for a while what was going on. Normally, if a program blows up in a multi programming system, it gets aborted by the operating system; everything else goes on. We finally after some digging around found out what happened. With the DELETE command, if IDS is instructed to delete a master record it has to first delete all the detailed records of that master record, before the master record can be safely removed. It's a nested delete, so that if I were to delete a purchase order record, I would want to delete all the line item records of that purchase order too. They can't be left lying around with a broken chain. So there is a cascading delete requirement. In IDS, the cascading delete function is handled directly, based on the data definitions. In a recursive structure, you're cascading in a structure that includes itself. It seems reasonable that I should be able to cascade a high level manufacturing part or assembly and delete all the components required down through an arbitrary number of levels. This was essentially a record that has detail records, which are also owner records, which have detail records, etc.

I implemented IDS before the GE computer, had stacks to control subroutine calls and returns. So, I blazed a trail down through the meta data structures to record the journey I was navigating to accomplish the cascading delete operation. When the DELETE subroutine needed to work its way back up the data structure, it followed these blaze marks, so it knew when to return control to the application program. The IDS cascading DELETE logic that I coded for IDS was still in the new IDMS version. Stacks were well-known by then and supported by the IBM 360, but my "train blazing" solution to the problem was still in the IDMS code. My code characteristics and behavior still existed so there was very clearly some IDS code that carried forward. Or they carried the concepts forward, even in terms of implementation techniques. They said there was no code carried forward. I don't know how else to answer that question. All I know is that it certainly behaved like code was carried forward. They say if it walks like a duck, and quacks like a duck, it must be a duck. It is also my understanding that the IDMS DML still matched the IDS DML and not the CODASYL version.

HAIGH: The article continues that the IDMS people believed that the software was marketable and submitted a proposal to BFG management to establish a new division. They made coding pads with their logo on, and gave away ashtrays to potential customers. They claim to have got five customers signed up, but then they were turned down internally within BF Goodrich, which was experiencing some kind of difficulty at the time. Instead they were told that they should take advantage of the company's experience in licensing to license the product. Then, they say, along came John Cullinane and the software was licensed and Cullinane's people[97] went out and signed up the customers that the Goodrich people had already lined up. That was, according to them, the beginning of the business.

BACHMAN: That's consistent with everything.

HAIGH: Fast forwarding from 1972,'73, when Cullinane signed its first contract to 1981, when you joined, how big was the company at that point?

BACHMAN: 300 people. They were already traded on the New York Stock Exchange. Cullinane Database Systems, Inc. was the first software company to be registered with the New York Stock Exchange.

HAIGH: Had they diversified their products at all, or was it all still centered on IDMS?

BACHMAN: It was centered on IDMS. They were just talking about bringing in a manufacturing control system, but they had a couple of little products on the side that they actually had had longer than IDMS. They were not very prominent.

While I was at Cullinane (1981-82), they brought in some people[98] who had worked for Stan Williams at the GE Computer Department on manufacturing control systems, and hired them to build a manufacturing package. I was not directly involved in it. Cullinane wanted to grow in the same direction that Oracle and other companies had taken: to go beyond the database world get into the application software world. They had begun that, but the foundation of their revenue was IDMS… they were dealing with one million dollar orders for database products. These were big orders they were getting from some of their big customers. Those were huge orders, from a software point of view.

HAIGH: Was IDMS provided only on the IBM mainframe platform, or did they port it to other platforms?

BACHMAN: There was an ICL version. ICL's computer was very much like the IBM 360 line. I'm sure that Cullinane built that version for ICL[99]. That's the only non IBM version I know of.

HAIGH: In your papers, one interesting thing from the Cullinane era was this proposed project for a private database facility. Do remember anything about that?

BACHMAN: Private database facility? I don't remember that in any particular way. What was I peddling then?

HAIGH: Here we are. Box 7, folder 29 there's an interoffice memo from 1981 saying that in the last couple of weeks you had a number of discussions concerning "a Cullinane product which can be characterized as a private or personal database system product. The user of such a product would be a person within a relatively large organization big enough to be using IDMS who would have a need for a small electronic database."

---

[97] Bob Goldman, one of the programmers at BF Goodrich, joined John Cullinane and eventually became the president of Cullinane Company.

[98] John Howard was the one that I remember, because Connie and I went to his wedding in Phoenix, when he married Stan Williams's secretary.

[99] March 14, 2006, recently I read that British Telecom, running the ICL version of IDMS, was the third largest OLTP site in the world. This based on a publication by Computer Associates (CA) who continue to market and support the Cullinet IDMS business.

BACHMAN: This was within six weeks of when I joined them. What was the substance of this?

HAIGH: You had some proposals. It seemed interesting kind of concept, especially considering the personal database systems that did become available in the '80s and 90s. Everything should be very simple. The user should be able to handle things themselves, but, presumably, at the point at least you were thinking of it as an extension of IDMS that would bring what ultimately became desktop style database capabilities to end-users. You would sell it to the EDP manager but it would be used by end-users.

BACHMAN: I have no immediate memory of that. I expect if I read the whole memo it would bring some memory back, but I can't speak anything about it today. It certainly didn't go anywhere.

HAIGH: Are there any areas where you think you did have an influence on what Cullinane actually did?

BACHMAN: I'd say very little[100]. It served as a bridge in some sense for me leaving Honeywell and exposure to smaller business before we started our own business. It served the period of time of tapering off on the process intercommunication stuff, ISO standards[101], and the buildup of the data modeling, which I thought that Cullinane should have been interested in. It turns out that that was not of interest to them. They were fighting the relational wars. Their focus was someplace I couldn't help them[102].

HAIGH: Can you say a little bit about this attempt by Cullinane to present its product as relational or to somehow migrate it to be relational?

BACHMAN: I read the specs, I read the manuals, and I couldn't understand them. I don't know whether the failure was in my mind or in the manual or if the product was just a myth. I know many people outside of the industry that I talked to considered it a myth, an attempt to confuse the issue about relational databases. I couldn't even tell you how it attempted to create this myth. My mind wasn't with it, the focus wasn't on it, and I wasn't part of it.

HAIGH: So why exactly did you leave Cullinane?

---

[100] I did some work of enhancements to the IDS/IDMS network data model, with the intent to add the capabilities inherent in the "Role Data Model." This lead to the development of the "Partnership Data Model" as offering all of the capabilities of the Role Data Model with the least ramifications of the existing IDMS programming. Nothing came of it at Cullinane. I subsequently completed that work and filed and won a US patent on the subject material.

[101] I was the chairman of the International Standards Organization's subcommittee 16, on Open Systems Interconnection from 1978 to 1982. I resigned at that time because I felt that it was interfering with my work at Cullinane.

[102] In hind sight, there was one thing that I could have helped them one that I failed to recognize at that time. Before leaving Honeywell in 1981 I had become familiar with Oris Friesen's work on an SQL translator. Its task was to take an query statement, expressed in SQL, and translate it into an IDS/COBOL program. This could then be run against a IDS database to generate the results desired. Oris worked for the GE Computer Department in Phoenix. This work was related to Oris's work on MIDS (Multics Integrated Data Store), the IDS to SQL translator, which was referenced earlier.

BACHMAN: Because they fired me. We just weren't going anywhere that was helpful to them, and so Bob Goldman said, "It's not working out." I said, "Okay." Two years after I joined them, I was terminated with a consulting agreement. Not that they wanted me to do any consulting. The consulting agreement was to generate some income over the next year, and also permitted use of their computer and IDMS for that next year, which was very helpful. I still kept using their computer to do the development of the data modeling tools that ended up being part of Bachman Information Systems.

HAIGH: With your one year consulting agreement and computer time, what did you do?

BACHMAN: First my wife, Connie, and I started Bachman Information Systems, Inc[103], on April Fools day, April 1, 1983. Then I started out to build the automated data modeling tools that I thought were appropriate. I started as Chairman and President. Connie was the "Vice President of Everything Else."

Our son, Jon, had graduated from the University of Massachusetts in Chemical Engineering in June 1982. That was the time of the oil supply crisis, and there were no jobs available for graduating chemical engineers. He went to work selling Saab cars. By the next summer (1983), when I was just starting the new company, I said to John, "I have gotten a consulting contract with a company (Nordata) in Norway, who were Cullinane customers, and who heard that I was going to automate the database design process. They thought that it would help them on some of their design work.

One of the big problems, when you making Data Structure Diagram, comes when you want to make a change. They were all drawn by hand. When you change it, you had to erase and rewrite. The update ability was very limited, very cumbersome. A large Data Structure Diagram would encompass many square feet of paper[104]. So if we could build a computer aided, graphic design system, the user could change the diagram a little bit and reprint it. Given the Nordata consulting contract that would bring in some money, I said "Jon, I can hire you now. I can guarantee you a minimum salary for a year." It wasn't a lot money.

So we started working with the Nordata people in Norway using Cullinane's computer. The Nordata people were working in Norway on their complementary part of the project. We were still using an IDMS database on the mainframe, for our meta data repository. We used it until we found that the IBM mainframe was not cost effective in supporting our kind of interactive model development. It just didn't do interactive processing very cost effectively[105].

Our Norwegian partner, Nordata, had signed a couple large contracts committing them to do something in the PC business. At the same time we were in development trouble, they

[103] This became known, as "BACHMAN," to easily distinguish from Charlie Bachman.
[104] When visiting Boeing Aircraft in Seattle, I was in one very large room where some data structure diagrams were eight feet high and thirty to forty feet in length. They covered all of the walls in that room.
[105] We set up some instrumentation and found out that the IBM operating system was executing ten (10) page swaps, of our graphic design program, for each one (1) page swap of the IDMS database. The IDMS database held the meta data. At that time, we were using a local service bureau in the Boston area. The job costs that we were being billed at the end of each run did not reflect the total bill we were charged. We were watching the end-of-run bills closely. One month, shortly after we starting using them, we received a bill in excess of $40,000, for the prior month's computer usage!

had trouble fulfilling their contractual commits, and they were in deep financial trouble. Our joint project fell apart.

HAIGH: Do you know what year we've reached?

BACHMAN: Yes. We started with them… this was, well, they fell apart the fall of 1984. We were trying to figure out how to continue. Fortunately, Phil Cooper called us at that time. Phil Cooper had sold his prior company to Cullinane. That acquisition had not panned out, as an addition to Cullinane. Cullinane had invested money, and they didn't feel they got value for it. Phil was out looking for things to do next. He was the kind who was good at starting up companies. He came by to see what we were doing, and he said, "I think I can help you take your ideas and get some venture capital money." Previously, he had gotten venture capital money for his current company, Paladian Software, from a company called Venrock. Venrock is the Rockefeller family venture capital firm. He had also gotten money from Kleiner Perkins, Caulfield and Byers. They were and are a very successful San Francisco and Palo Alto based, venture capital company. He was able to help us put together a business plan. Phil's new Palladian company was one which was selling an artificial intelligence (AI) oriented product. He said, "Well, there's some things that are AI-like that you're doing to plan data models, so we ought to leverage that." We did leverage the AI into some venture capital money to build the data modeling tools that I had in mind. This was the time to start out. By the time that we get to the PCs, which are 386 based, they would be big enough and fast enough to do the kinds of things we wanted to do. They would drive some big graphics monitors that are available or will be available." So we started hiring a group of people. some people out of the MIT environment who were doing AI work, programming in LISP.

HAIGH: Wait. So you received this venture capital funding?

BACHMAN: Yes.

HAIGH: Was it a large amount?

BACHMAN: Oh yes, it was two million dollars. That was a large amount. Not by 2004 standards, but it was big money in 1984 for a software company, in which they wanted majority control of the company.

HAIGH: And did they get that?

BACHMAN: They did. Connie calls them "vulture capitalists".

HAIGH: Were you the CEO of the company?

BACHMAN: I was the CEO initially. The President and CEO. I wasn't chairman. Phil Cooper was the Chairman of the Board. The board was made up of myself, Phil Cooper, and our two lead venture capitalists (David Hathaway of Venrock and Floyd Kvamme of KPCB) plus Russell Notfsker, from another company (Symbolics) in Cambridge. They made AI oriented computers that were designed to efficiently run programs written in the LISP language. As the builder of specialized computers, the Symbolics Company suffered the problem that all specialized computer manufactures have eventually suffered. Manufactures of specialized computers have continued to suffer the problems of high engineering and high manufacturing costs, when compared with the Intel commodity chip computers. The Intel based computers have increased in cost effectiveness much faster than the specialized computer manufacturers could.

HAIGH: LISP processing machines?

BACHMAN: Yes.

HAIGH: Yes, I know there were a couple of companies making those. So the business plan was to produce a front end on a PC?

BACHMAN: The PC was not a front end to the mainframe. It was the whole computer-aided graphic design product on the PC. The PC was to host the entire graphic design product to support mainframe database analysts and database administrators. That was a new thought.

HAIGH: So it wouldn't connect directly to a database?

BACHMAN: No, no! The meta database would be kept within the PC.

HAIGH: So it was a design tool, but it wouldn't interface directly with IDMS or another database?

BACHMAN: Well, it would in a sense. There was a data analyst tool, which was a conceptual schema tool. It had the database administrator tool, which was essentially the logical design for IDMS. Then we had forward engineering and reverse engineering. So we could take an existing IDMS definition and reverse engineer it into a conceptual view. You could extend it and modify as you wanted, and forward engineer it to make a new database design.

HAIGH: So these products would output data definition language statements.

BACHMAN: Absolutely. Generate DDL statements.

HAIGH: Then you could upload those into your database management system product?

BACHMAN: Yes. Right.

HAIGH: What was the first product to appear?

BACHMAN: The first one to appear was the BACHMAN Data Analyst product.

HAIGH: What exactly did that do?

BACHMAN: It essentially allowed you to create a Data Structure Diagram, at the Conceptual Schema level. It was based on the partnership data model.

Part of the time during the first two years was spent formalizing the partnership data model and submitting a patent on the Partnership Set data model, which gave our conceptual schema, a richer modeling capability. In other words, the partnership data model was a richer model than the original network model. It was an augmented network model, and it was the basis for the way we stored the meta data in the meta database, inside of the PC and how the graphic design was displayed, when you looked it on the screen or printed it out.

At the conceptual level, we said that we have entities[106], which are the conceptualized view of the things that a record represents. You had many physical records for the database record type that represents one type of entity. Each entity type had attributes, which are the conceptual equivalents of columns, fields, or data properties, in physical database systems. It had partnership sets and partnerships. Earlier network model systems had just had relationships. They were implemented with "linked lists" or "chains[107]." The partnerships and partnerships provide a richer relationship, because we talk about them from the point of view of each entity type. We have a relationship between two entities. If I have an entity, who is an author, and there's another entity which is a book. The author "authors" a set of books, and the books are "authored by" the author. A book may be "authored by" multiple authors. There could be many-to-many relationships, which we could model with the partnership data model. Each relationship is terminated at each end by a partnership set. In a simple sense, you have one relationship joining them, which we called a partnership. The thing with partnerships is that they represent an equal relationship between two entities. They are used to build undirected graphs. The terminating partnership sets are given names, which provide separate semantics to the partnerships from the point of view of each entity.

HAIGH: Did you consider this to be a form of conceptual schema?

BACHMAN: We used the Partnership Set Model for two purposes. First, we used it for the physical model used to implement all of the meta data models used in our products. We used this for storing information about conceptual schema, physical schema, data flows, process diagrams, etc. We used it to store IDMS and DB2 DDL. Second, we used it as the user visible basis for the conceptual data schema. The builder of conceptual schema sees and uses entities, attributes, partnerships and partnerships to express their conceptual data structures. It could be used as the basis for a database management system.

The Partnership Set Data Model has another concept that is worth mentioning. This is called a "dimension." It is used to identify each particular kind of information. It is used to identify the kind of information, but not the form of that information.

If you take the problem we had with the "Year 2000." People said, "We have all these dates, "95," "96," "97, "98," and when we get to the year 2000, should record it as "00." Why, because all I am storing are two digits." A two-digit date is a representation of something, which was larger than that. Somebody said, "I've got to rebuild all my systems and put in four digit dates." Well, that would be a different representation of a date. If you think of and use a word physical database world, those descriptions are really "domain" descriptions, i.e., physical representations of a certain kind of information. We introduced the notion that provided a conceptual schema level of data definition of a value. We called it a "dimension."

The word "dimension" comes from a usage in engineering. If you take mechanical engineering, one of the things they teach you, early, is that you have to be careful that

[106] In more recent years, these entities in the conceptual schema are frequently characterized as "roles" that are played by real-world, or business-world entities. Mother and Doctor are roles played by real-world entities. One such mother role and one such doctor role might be played by the same real-world entity.
[107] They are a particular physical implementation of the conceptual Owner/Member set.

you don't add "velocities" and "distances" because they are of different dimensionality. Velocity is a length value divided by a time value, while distance is merely a length value. Dimensional analysis is an important early subject in engineering courses.

HAIGH: Yes, I remember you spent some discussion that concept in the interview with Phil. This found its way into the data analyst product?

BACHMAN: Yes, the data analyst product allows you, for every attribute, to define at the conceptual level to say what dimensionality that attribute is representing. The whole idea is if you're going to have different attributes, some attributes are comparable. Some operations on their values are permissible. Some attributes are addible, and resulting value is of the same dimension. Some are multipliable, but the result is always of a different dimension. "Length times length," is length squared. It certainly makes sense. The whole effort was to define this so that people would not write programs with undetected errors because of faulty dimensionality. The issue is data integrity. Your column definitions, in a relational database really must map onto a dimension definition in the conceptual schema, to assure integrity and correct transformations.

HAIGH: If the output of this package is a set of data definition language statements for an existing database management system, how would you go about enforcing those dimensional constraints on people who are programming an application?

BACHMAN: One of the aims of this conceptual schema, after you had done the data part, you were going to do the process definition part, again at the conceptual level. You're going to write the processing rules at the conceptual level. That would be rules written to manipulate the conceptual data that could be forward engineered to COBOL programs, PL/1, JAVA, or C# programs that manipulate particular databases.

HAIGH: Was that component realized?

BACHMAN: Yes, the process modeling part was realized. As Bachman Information Systems went over the horizon and disappeared, we had several customers who were using it to generate application programs. However, in the process of developing this product, we spent too many engineering dollars that were never covered with income dollars.

HAIGH: Did that product have a name?

BACHMAN: Yes, and I'm trying to think of what the right name is. I guess I could look at the product and tell you what it is. It's on the system here. It was a product for a logical programmer or conceptual programmer, not a physical programmer.

HAIGH: In what year did the Data Analyst product ship?

BACHMAN: I'll show you a picture. This is one of the memorable pictures of our situation where the first money we got from a customer was $75,000. In fact, we have $75,000 in currency that we brought in to the board meeting to show the Board of Directors.

HAIGH: And they didn't pocket any of it?

BACHMAN: No. The bank was so nervous they sent along an armed guard, who you can see standing in the background there. He was very nervous to have this much currency walking down the street. In order to carry this thing forward from the initial design phase, where we had a functional specification and first operating model, which was around 1986, we signed up “development partners,” ten different companies who were willing to invest $100,000 each into the process to help us bring the product to market. When we showed them the first version of the product to them, they said, “This is really great, but this, and but this, and but this…” I think we had a list of 100 things that they said “but” about. Instead of being able to ship immediately, we had to go back and reworked the product.

The first customer, Union Mutual Life Insurance Company up in Portland, Maine, signed up for $75,000. After we got them signed up, we found that the customers couldn’t really tell the difference between $75,000 and $100,000, so we raised the price. It is well to remember that this was at a time where our customers were using IBM 360 mainframes which sold for millions.

HAIGH: Nice round number.

BACHMAN: Nice round number. It was another year before we got the next product out in an improved form, and very much like it is today. At that point, the database administrator tool didn’t exist yet. This was just the Data Analyst for doing the conceptual schema.

HAIGH: So that would have shipped around 1987. Is that correct?

BACHMAN: Yes.

HAIGH: And it ran on a PC?

BACHMAN: It ran on a 386 model. There was a black and white monitor available, which was very large for those days, and with very high resolution. It was really an elegant machine, but when IBM went to the 486 they did not write the device drivers for it, so we lost it, so to speak. That was shipped and every customer received ten copies of the software, for their $100,000 development partnership.

HAIGH: That was ten licenses or was that a site license?

BACHMAN: It was ten licenses. It was not a site license. At this first stage, we had made John Cullinane very unhappy with what we were doing because when this thing came together, there was the ability to capture and reverse engineer existing database descriptions. The initial reverse engineering product could capture was IDMS DDL, and they would reverse engineer it. They could modify and extend it using the Data Analyst product, and forward engineer it back to IDMS DDL. Eventually they could forward engineer it to IBM DB/2 DDL. John looked at us as an escape strategy to get their customers off his software onto somebody else’s software. He was really terribly unhappy with us. We looked at it as a point from where we could take people to wherever they wanted to go. He looked and felt vulnerable to it.

HAIGH: Do you have a sense of whether a large proportion of the users were using it as a migration tool?

BACHMAN: I don’t think it was used in a great extent as a migration tool. I think a lot of people used the capture in order to merge several IDMS databases into a conceptual schema, whose scope was multiple databases wide.

HAIGH: Did the initial version of the product only support IDMS?

BACHMAN: Only IDMS at the physical level. The partnership data model was the foundation of the storage of the meta data captured from the IDMS DDL.

HAIGH: Were additional platforms added over time?

BACHMAN: Next came DB2. We became was what was called an IBM partner[108], and IBM invested money in our company. We got an assistant treasurer from IBM on our Board of Directors, and we took on very large marketing program where IBM was very helpful for us in the marketplace. IBM had started the talk about the "reengineering cycle" in which we provided the only coherent story. All the rest of the IBM partner companies were only doing data modeling. They were not doing forward and reverse engineering. One of them was KnowledgeWare[109] in Atlanta, and there was a company in Boston, Index Technology[110]. My though was, "Index Technology has the largest customer base. KnowledgeWare was making the most money. And, we had the best technology." All were attractive in one sense or the other to IBM. They thought Bachman Information Systems could help them in their world.

The next version of the BACHMAN products we built supported DB2 capture, reverse engineering, forward engineering and optimization. At that time, we shifted from Microsoft's DOS to IBM's OS2 operating system. The conversion to OS2 was a problem.

HAIGH: Did that relationship with IBM continue to be mutually rewarding?

BACHMAN: It was for a couple of years until IBM announced that KnowledgeWare was their primary business partner in the CASE[111] world. The other two of us were to be restricted to work in data modeling, which really caused us to be very unhappy with the relationship. Just before this, IBM they had a program to build a main frame repository, which was to be the central database in a corporation for all the data modeling and the database design information (meta data). It was a disaster in its conception and a disaster in IBM's implementation of it.

We were heavily pressured to buy an IBM mainframe. IBM wanted us to build the software to run on the main frame to store our meta data and also to build the software so a PC could interface with an IBM mainframe. IBM wanted us to build both the PC and main frame software for the Repository, because there was not standard definition of either the meta data or repository function that could support the requirements for the three Business Partners.

We had to buy a good-sized mainframe just to support our development of stuff for the Repository project. IBM would not back away from their repository vision. We finally scuttled the whole Business Partner thing. If we had been smarter, we would have scuttled it more carefully. We should have just quietly stopped supporting the Repository instead of saying, "We quit." When we said, "We quit." IBM, said in the marketplace, that they quit BACHMAN. That cost us orders that we would not have lost. You learn those things, I guess, by hindsight.

[108] Earl Wheeler, IBM's Senior Vice President, Programming Systems was our contact and sponsor.
[109] Fran Tarkenton, was the CEO of KnowledgeWare.
[110] Rich Carpenter, was the CEO of Index Technology.
[111] CASE, Computer Aided Software Engineering.

HAIGH: Let's return to the company itself then. So you mentioned that with this money that came from the venture capitalists you were able to start hiring people and putting them to work developing the system. Were there any people that you think were particularly important to the work of the company?

BACHMAN: Yes, there were a number of people. A man name Arnold Kraft who came in as our chief engineer, although Arnold was not an engineer. He had done a lot of work at DEC[112] in artificial intelligence systems. I had met Arnold years before in a class I was teaching in data modeling. I thought he was the only person who really bought in on the concepts I was describing.

BACHMAN: The next person I'd recommend talking about is Sheldon Borkin. Sheldon Borkin is an interesting guy. We first talked to him about being our chief technical officer, before we hired Arnold Kraft. He was working for IBM at that time in the IBM research lab, in Cambridge, Massachusetts. He had some very good characteristics. He was a graduate student at MIT. When he wrote his master's thesis, and his advisor looked at it and said, "Sheldon, if you just put a little more effort in it, we'll give you your PhD degree for it." He got a very quick PhD, based on his technical paper. The paper was contrasting the mathematical similarity between the network data model and relational data model[113]. It seemed like Sheldon had some of the right kind of background to do what we wanted him to do. We thought we were going to hire him. In fact, he said he would come to work for us. Sandy, Sheldon's wife, had a shaky health history. She was just terribly frightened about leaving the protection of IBM's health program to go to work for us, even though our office was just down the street. She was just very nervous about it, which I could understand. Sheldon backed out on it, and we subsequently hired Arnold Kraft.

A year into the development process, we were struggling with developing a product. At that time the venture capitalists said, "Charlie, we think you ought to move up to the position of board chairman, and we'll have Arnold move up to be president to run the business." At which point we had already hired Sheldon, so he took over as head of engineering.

HAIGH: That brings up a question about your own role in the company. As it grew, how much did you continue to be the hands-on day-to-day head of the company?

BACHMAN: Less and less. I became involved with the IBM relationship. One of the things that IBM was interested in was the ability to do a comprehensive enterprise model (conceptual schema). That is, an enterprise model that was not just modeling the databases, but also modeling the processes, information flows and business rules. We were first working to create a standard conceptual schema model that would be used by IBM and all three of the IBM Business Partners. A model we could all share. This related to IBM's repository initiative. For a year, a great deal of my time was being spent trying to iron out this relationship for the three Business Partner companies and IBM. We could

[112] "DEC" – Digital Equipment Corporation, headquartered in Maynard, Massachusetts.

[113] The network and relational databases both had predeclared record like objects (rows in tables). They both had predeclared data fields (columns). They both supported one-to-many relationships, but the network databases's relationships were predeclared while the relational database's relationships were declared at retrieval time.

not agree. We were successful in making progress on the standard meta schema, except the three other companies, and KnowledgeWare, in particular, could not agree on modeling relationships, using the partnership set concept. They were saying, "Well, we're not going to use it." So we kind of agreed not to agree, so we had an incompatible notion on what's going in the repository. IBM wanted to go beyond that and say, "Let's model processes and communication flows between process." In fact, there's a much larger data model, showing the whole thing we developed with them. I personally completed that model to my satisfaction[114]. I spent a good deal of my time on that technical issue, while Arnold was managing the company. He was the business manager and Sheldon was running the technical side of it. It evolved in that way.

I retreated to the technical side, which is my strength. We were getting big enough that we had to begin to train managers in our own company. We now had management training courses. I was asked to come and speak at one of the management training courses about my view of managers and management. I said, "You have to understand that I have spent most of my life as an "individual contributor," working technically on projects. From time to time I have been the manager of projects, but my greatest skill has been as a technical person. Most of the people in the company are individual contributors. You've got managers. Managers are the people responsible for providing "rations and quarters[115]," and then are supposed to get out of the way."

HAIGH: I have here from 1969 your performance appraisal from Stan Williams. He heaps a lot of praise on your technical contributions, but then he says that "Because Charlie is usually thinking far ahead of the state of the art, many people do not understand his ideas and reject them as being too far out, too complex, inefficient, etc. Charlie needs to improve his communication channels with other competent technical people who do not understand or agree with his ideas without compromising what he believes is right. Charlie's primary interests are technical rather than managerial. Charlie's problems in the managerial area are demonstrated by his inability to man his organization at the assigned rate without schedule slips and cost overruns." Would you say that that fits with your own sense of yourself as someone who's stronger--

BACHMAN: Oh, yes. I was much stronger as a technical contributor.

HAIGH: So as you were finding yourself in more of a managerial role with Bachman Information Systems, were there elements of that that you found that you enjoyed? Or were you relieved to be able to step back a little bit?

BACHMAN: The step moving from president up to chairman was the first step in recognizing skills and where they fit. Chairman can do many things in different companies, but my role was mainly as a technical contributor. I was writing a number of technical memos. I don't know if you have any reference there, but there is a whole slew of technical memos which they have at the Charles Babbage Institute. I have duplicate copies of the technical memos here.

[114] One of the interesting parts of this is how a network data structure is reconfigured to flow as a data stream and be able to reconstruct itself as a network structure, after it has been received.
[115] "rations and quarters: is an old army expression.

HAIGH: I didn't get far into the Bachman Information Systems papers. I know there are some things there that I didn't look at. These could well be included.

BACHMAN: There are papers and presentations and just technical manuals that were just essentially working problems, and this was my way of providing input to the technical organization. I had two copies. I didn't send both of them to the Babbage Institute; I kept one set here. Those Technical Reports embodied a great deal of my productivity at that time.

HAIGH: Other than the people you've mentioned, are there any other individuals you think played an important part in the company?

BACHMAN: A lot of people did. You know, the marketing side. It's just hard to focus on too many people.

HAIGH: Were there any other people on the programming team that you think made exceptional contributions?

BACHMAN: One person, John Cimral, ended up being our head of engineering. He was a West Point graduate and was teaching at West Point when we hired him. He helped build the technical staff to a size that we couldn't afford. I was not wise enough, and Arnold Kraft[116] was not wise enough, to say to John, "We've got to cut this specification back to fit what we can afford. Our eyes were bigger than our stomach." We tried to eat too much. We failed as managers, by not realizing our limitations. I was annoyed with John, because I thought somehow the engineering manager ought to recognize the problem and help us realize our limitations. I was more responsible than John, and I now recognize it.

HAIGH: Did that experience give you more sympathy for the managers during your career at GE and Honeywell who had been saying, "No," to your ideas because they thought that they couldn't afford them?

BACHMAN: I guess I never thought back about those people that much because most of the time, the managers had been supporting what I was doing there were exceptions always, things that do not fly. Certainly in hindsight, I understand the limitations of what a manager can do. In fact saying their job is to furnish rations and quarters is maybe not a very sympathetic view of what managers do. They have a problem pulling it all together. Certainly, Stan Williams was a much better manager than I was. We had complimentary skills. . I was most successful when there was a real project that needed my technical contribution.

HAIGH: Let's return to the development of the company's products. We discussed the data analyst tool, and you said that originated on the 80386. Was it ported to any other PC or workstation platforms?

BACHMAN: It was moved up to other computers in the Intel 386, 486, so on series. It moved up first as software running on IBM's OS/2. We did not build a Microsoft version until we broke with IBM. At one time I had a laptop computer that would run both the Windows and the OS/2 versions. You made a choice at "boot" time.

[116] Arnold Kraft was CEO and President of Bachman Information Systems, during the middle years.

HAIGH: So presumably there was a Windows version?

BACHMAN: Yes, in fact, that's what I'm running now. It moved up. I called Denise Dodd, who is the on former Bachman employee who is now supporting BACHMAN tools at Computer Associates. She told me that the BACHMAN products did not work on the Windows XP. The technical support people at Cord Blood were able to get them loaded and running quite successfully on XP. I called her back and told her how we had accomplished it. The loading system would not work, but if you loaded each module separately, XP was happy to accept them, and they run compatibly. I'm hoping it'll keep the BACHMAN products running on newer versions of Windows, in future years.

HAIGH: If there's one thing that Microsoft's good at, it's keeping backward compatibility.

BACHMAN: As long as Microsoft doesn't destroy it somehow.

HAIGH: So you stuck with the PC platform for the client. You've mentioned IDMS and DB2. Did it every support any other database management systems?

BACHMAN: Other relational ones.

HAIGH: So it would support Oracle?

BACHMAN: Oracle and Sybase, yes.

HAIGH: I understand that you had some products other than Data Analyst.

BACHMAN: The database administrator products were different than data analyst products, and different than the forward engineering and reverse engineering products.

HAIGH: So that was a product family, but all those components were sold separately?

BACHMAN: There is some confusion on this subject in my mind, but I think that as we built more products, you got them bundled for the original price.

HAIGH: So there was Data Analyst. That did the diagramming, right?

BACHMAN: Yes. It did data modeling at the conceptual schema level.

HAIGH: What did the database manager product do?

BACHMAN: The database manager would help you optimize the relational database. The BACHMAN/Data Analyst became called Cayenne/GroundWorks, after we merged with Cadre Systems. We renamed the company, "Cayenne Software." "The Production DBA for DB2" was one version of the database administrator. There was the Shared Work Manager, which allowed many workstations access the same design database. The "Terrain" was a generic relational database with generic SQL. "Terrain for DB2" was for an IBM DB2 database using IBM's version of SQL. There was "Terrain Map" and "Terrain Map for DB2" They handled reverse and forward engineering between GroundWorks and Terrain and Terrain for DB2, respectively.

The Shared Work Manager product came along latter permitting two or more PCs running BACHMAN products to access and update a shared meta data store. That shared meta data store could be on one of the user PCs or on a separate free standing server. This was the BACHMAN Repository.

HAIGH: Did Data Analyst itself generate the data definition language statements? Or did you need an extra product to do that?

BACHMAN: The Database Administrator product (Terrain and Terrain for DB2) could generate IDMS DDL. It could read IDMS DDL. They provide graphic representations of a physical database design. It could optimize an IDMS database design. It could accept an IDMS design from forward engineering (Terrain Map). It would feed an IDMS database design into reverse engineering (Terrain Map). A linkage was maintained between the logical and the physical design elements.

The original versions of the BACHMAN products were based on the Microsoft DOS. That was before either Windows or OS/2 were available. We shifted the operating system base from DOS to OS/2 foundation, when OS/2 became available.

HAIGH: So you had three versions over time: the DOS version, the OS/2 version, and the Windows version.

BACHMAN: Yes. When we severed our relationship with IBM we ceased developing under OS/2 and moved to Windows. One of the real problems that plagued us is we were slow moving to Windows, which hurt our sales effort, while we were still IBM Business Partners.

HAIGH: You mentioned an initial $100,000 price point for the Data Analyst product.

BACHMAN: That’s right.

HAIGH: Did the price stay around that level?

BACHMAN: That price was for the BACHMAN development partners, which gave them ten seats. When we were planning for a Cullinane annual user meeting, which was to be held in St. Louis, Floyd Kvamme made the following suggestion. Floyd Kwamme, who was on our board of directors representing Kleiner, Perkins, and Caulfield, said, “Tell them the price is $10,000 per workstation and see what kind of reaction you get.” That price did not seem to faze the first group of Cullinane customers that we talked to. We tried $12,000 for the next group, and that didn’t seem to faze them. Then we tried $15,000, and they had a little reaction to that. So, we went onto the market with an initial price per seat of $15,000, and less for quantities.

HAIGH: As you introduced these additional modules, were those priced separately?

BACHMAN: Initially they all came bundled.

HAIGH: By the time that you had all these different products available, if you wanted to get all of them, would it still have been about $15,000 a seat?

BACHMAN: Yes, but if you wanted to have the shared design database facility, you’d buy the Shared Work Manager separately.

HAIGH: So the price stayed about the same, but over time you got more for it.

BACHMAN: Yes.

HAIGH: Did you see any price pressures from competitors?

BACHMAN: Yes. There were some people coming in at the low end, with a $1,000 - $2,000 per seat product. In fact, if I look at the IBM’s Rational Rose today, it still looks like one of those $2,000 product. It's a weak data modeling tool. I have installed it, used it, trying to do some work with Cord Blood. For me, it was a big disappointment. It just doesn't model well at all. Rational Rose went into the market at $2,000 and are still $2000. They succeeded and we failed, because we did not figure out how to get our price down to the point or get the sales volume up enough to generate the revenue needed. They ended up getting bought out for a couple billion dollars by IBM; we ended up disappearing. They had the successful business managers. We sold "Cadillacs," and they sold "Model Ts."

HAIGH: Do you want to tell the story of what happened to the company? I understand it involved some mergers.

BACHMAN: We were struggling along trying to make a profit. One of the Board of Directors said that we had needed to find a new president/CEO to replace Arnold Kraft. We talked to a number of people, but didn't really find the right person. Finally, the Board of Directors, without my knowledge, hired a person to come in to take Arnold's place. That was Peter Boni.

Peter Boni's approach was to look for some company to merge with, so we could quickly achieve a bigger revenue stream. He identified Cadre Research of Providence, Rhode Island and we went through a merger. We ended up with the BACHMAN engineering department and the Cadre marketing department. Both companies had respectable products.

The new President/CEO thought that our weakness was with the financial community. He renamed the merged companies and made it a new company to give it a new look. It turned off the existing customers, who didn't know this new company, "Cayenne Software," or its products. We lost something very valuable, and we spent over a million dollars in re-launching the new company. It may have helped us in the financial market. They changed all the product names too, that didn't help either in selling the existing products.

Cadre's data modeling tool, which really responded to large engineering department's requirements. It was a technical market. They sold it primarily to the aerospace industry. We short changed our existing customers and essentially walked away from them. It just kind of dribbled out. We borrowed more money and got people to come in and say they would invest money, for preferred stock, so they would get preference in any liquidation. When we finally sold the company to Sterling Software, essentially all the money that Sterling Software paid went to the preferred stock holders. Common stock, which at one time had sold for $38 a share, finally sold for 37.5 cents a share.

HAIGH: I realize you didn't say anything about the IPO process.

BACHMAN: The IPO process, what do you want me to say? It was traditional. We did all the right things. We had silent periods. We went around and talked to investors in New York and California and Boston and Chicago. We had two investment bankers. Morgan Stanley was our leading investment banker, and they took their cut out of it. To me it was not a big deal. For me, it was so the Venture Capitalists could liquidate their investments and some people, like Phil Cooper, who was brought in our original venture capitalists, could cash out his quarter million dollars worth of stock. The management team members were all insiders, and could sell later under strict constraints.

The company's other employees, who were locked up for two months sold a great deal of their stock at the end of the first two months, long before it went up to its peak of $38 per share. By the time it got to $38, we were looking at situations where the long-term plan didn't look favorable, so we were not allowed to sell at that time.

HAIGH: So you never really made much money out of it personally?

BACHMAN: Never. You know, it made an interesting 15 years. I made enough money to buy this house and put some money into index funds. I'm very happy with the results we got. We could have had more money if we had sold stock earlier. Lots of things could have happened if we were shrewder. On the other hand, the stock that we did not sell would have made us very wealthy if we had gone the Rational Software route into a merger with IBM.

HAIGH: So it was 1998 when it was finally acquired by Sterling?

BACHMAN: Yes.

HAIGH: And I imagine it must have been quite soon after that that Computer Associates (CA) acquired Sterling.

BACHMAN: Within 18 months I'd say. Sterling sold out for two billion dollars to CA. The principals did very well. Sterling was really an acquirer of products. They would acquire PC products, whereas CA was an acquirer of mainframe products. The rapid expansion of PCs into the transaction processing and database world did as much as anything to destroy the Bachman Information Systems' business model, because the PC was destroying the mainframe business model. The speed and storage capacity of the PCs were growing exponentially. People were building their databases on the PCs and the mainframe market was not growing in the old way. We were too close to IBM's old ways of thinking, and suffered from the same lack of vision. Our biggest customers were State Farm Insurance in Illinois and the Italian Telecom in Rome, Italy. Both were deeply invested in the mainframe point of view. It was part of something that we didn't anticipate well. We were slow in moving to the Microsoft Windows PC operating system, versus IBM OS/2. We were slow to understand the impact of relational databases on mainframes, because people weren't building new mainframe databases to the same extent. And, they weren't doing much data modeling, in anticipation of building databases. Rather they were into the rapid construction of PC databases, and fixing and patching them, in a risk-taking mode of operation.

HAIGH: Then, your resume shows that from 1998-2000[117] that you were a consultant with something called Constellar Systems in Redwood City, California.

BACHMAN: Yes. George Colliat, who was the VP of Engineering at Constellar, is a former Honeywell employee and was familiar with my work. He called me and asked me to help him on his project.

[117] Bank records show consulting payments dating from October 1997 to September 1999.

Constellar was a company whose business was to help companies, who were trying to be an integrated two or more organizations by integrating their existing systems. They couldn't technically or financially build a new, single integrated system. Due to rapid growth, merges and acquisitions, these companies have many separate and overlapping applications systems, containing data in different formats. These application systems must, in their new environment, communication with application systems that are foreign to them. They were "dimensionally" compatible, but "domainally" incompatible.

Their problem was crying out for a "three schema" solution[118]. They said their customers have outputs from several applications that needs to get someplace else and be delivered in a new format, to meet the requirements of a set of receiving applications.

The original problem statement of the three schema system dealt with many "internal schemas[119]," one "conceptual schema," and many "external schemas[120]." It was trying to solve the problem of going from any one of the internal schema to any one of the external schema, without having to manually create a special solution of each possible combination. The three schema approach required each external schema and each internal schema to define a mapping defined between it and the one conceptual schema. Given that any pair, of external or internal schema, the information residing in their maps onto the conceptual schema provides all of the information required to compute a direct mapping for any combination of internal-to-internal, internal-to-external, external-to-internal, or external-to-external schema. In Constellar's case, they were only interested in external-to-conceptual-to-external combinations.

I started out to help Constellar build a three schema solution. There is one conceptual schema per customer, and any number of external schema. They represent the input and output files of the customer's application programs. Use their mappings onto the conceptual schema to compute a program that would do transform each output from here into an acceptable input, for over there. Have this output file converted, and communicated as input for the program over there.

They got it up and running quite well, but they ran out of venture capital money before it could be packaged for the market. Their consulting business was not generating sufficient revenue to carry it without more venture capital and it was not forthcoming.

As part of this activity, I built a very comprehensive meta model of the entire scope of the prospective three schema operation. It was built using the BACHMAN/Data Analyst product (Cayenne/GroundWorks product). It was an extension of the comprehensive model that I had built as part of the BACHMAN IBM Business Partner cooperation which was intended to the be generic meta schema for the IBM Repository.

HAIGH: So this company was a startup?

BACHMAN: Yes and no. It was a consulting business that originally hand coded all the translation programs. They thought that mode of operation was too labor intensive to

[118] We have talked earlier about the use of "internal," "external." and "conceptual" schema, and their provide the transformation rules used, in transforming data from one format to another.
[119] Think of an "internal" schema as being the physical schema on the database side of the application.
[120] Think of an "external" schema as being the physical schema on the communications side (input/output side) of the application.

really survive and grow. So they started looking for a software solution. It was an English firm. They reestablished themselves in Silicon Valley, so they could find the financial and technical resources they needed, and be closer to the US market.

HAIGH: So you joined in 1998. Was this shortly after they received the venture capital?

BACHMAN: Actually before they received the venture capital. I was working with them on a month to month consulting basis, with a retainer fee. I received a phone call from a man named, George Colliat, whom I had known at Honeywell, who knew I was living in Tucson, and knew that I had worked with IDS. He thought that I could help him organize his problem. So I spent quite a bit of time with him over a period of two years, and so we made very good progress. He had some very good people working for him. They were using a different data design tool for the data modeling. They made what I thought then was a serious engineering mistake. They had two teams working on different parts of the system. One team was working on the physical schema, and one team was working on the conceptual schema. We argued extensively on the issue. A consultant can only say so much. George said, “We can let them use different data models instead of forcing them to agree to one data model.” As the result, they did not resolve their differences and didn’t have one conceptual model; they had two conceptual models, which they never could reconcile. They had a deadly technical problem along with a money problem. That problem meant that they either ran out of money sooner than they should have, or the product came out later than it should have. Between the two, the company collapsed.

HAIGH: So they did ship a product in the end?

BACHMAN: They never shipped a product, no. That project kind of came to the end. There’s a lot of papers I have about that work, and a lot of big models and technical papers that will end up in the archives at some point. I went back to doing other things like trying to help our church with its finances to find out where their monies had been going, why they were having trouble with finances, and why they didn’t know from year to year what their financial picture was.

HAIGH: Is that a local church?

BACHMAN: Local church here, yes, that we’ve joined[121]. They never could understand that they should live within their income, so I finally gave up worrying about their finances. We are still with the church, but I decided that I was wasting my time and annoying people by saying that they should try to live within their income. From Bachman Information System experience, I knew that we failed financially, because we could not learn to live within our income.

HAIGH: I think you described your work with Cbr System in some detail in the interview with Phil Frana, for the Charles Babbage Institute[122].

BACHMAN: Yes, although that’s been what? 18 months ago now?

HAIGH: So that’s progressing?

[121] St Philip’s In The Hills Episcopal Church, Tucson, Arizona

[122] An email exchange, in February 2006, indicated that the recordings of that oral history had not yet been transcribed.

BACHMAN: Yes, I think Cbr Systems[123] is progressing, but not very well. They have not really resolved the issue of what DBMS to use to implement NDBA.[124] They are concerned that SQL Server will be an inefficient foundation, based on what they wanted to do with working with an object-oriented programming and its network database orientation. They were looking at a product called Cache as one. They were looking at another product called Matisse. In fact, I have Matisse on my laptop. But they were again concerned, in Matisse's case, that the supplier was not a big enough company to depend on. They're concerned, when they're looking at Cache, because it is not broadly used, even though they had several major hospital chains in the United States who use a system. It is an object-oriented database.

HAIGH: Roughly how big is their database in terms of megabytes?

BACHMAN: I know that just in the contact system they are receiving or storing something like 250,000 new names every month. These are, in other words, information coming in from mailing lists. These are people that are going to become mail addresses. 250,000 a month can add to a good-sized database in a hurry. It's their biggest volume. We're using it as a test vehicle, so if we can handle that kind of volume. Marketing is the biggest expense of almost every company I know. That's our test vehicle, and they're not happy with how fast SQL Server handles it in either retrieval or storage. Cache promises "to be 200 times faster in some cases". I don't know quite what those cases are. But the Cache company had offered to duplicate what we're doing in the translation from our code written in C# into their system. Last I heard they were willing to take our specifications and create the equivalent system and show us their performance against what performance we're getting running the same tests mapping down to SQL Server. That's where we are now[125].

HAIGH: Then pulling back on this to ask some concluding questions about the developments of technology in general. I think you've already alluded to your view that there are distinct parallels between the network data model and XML. I wonder if you'd like to elaborate on that.

BACHMAN: There are things that I'm told about how XML is trying to build relationships between records. I'm told by my son Jonathan, who's in the software business[126] that they're trying to reinvent what I did with the network data model. A new set of people, with a different set of backgrounds, and typically not a network database background, are reinventing network databases. Beyond that, I have no facts to offer you.

---

[123] Cbr Systems Inc. is their legal name of the company we have discussed before as "Cord Blood Registry." Tom Moore is the Founder, President and CEO of Cbr Systems, Inc. "Cbr" stands for Cord Blood Registry. They are in the business of collecting, processing and storing the blood collected from umbilical cords, at the time that a baby is born. The purpose is to preserve the stem cells that are found in the umbilical cord blood so that it will be available in the case that child or a close relative needs a stem cell transfusion. Tom is a former Honeywell Information Systems Vice President and knew of my prior work.

[124] It is the "New Database Architecture" (NDBA), which was designed to meet their particular business requirements.

[125] In March 2006, "Hibernate," as a Java, or C#, to SQL Server translator is the current answer and appears to be acceptable.

[126] Jonathan Bachman is with Sonic Software that is part of Progress Software, in Burlington, Massachusetts.

It is interesting that they are attempting to transfer, as a data stream, network structures that can be reconstructed, or interpreted, as network structures after being received.

HAIGH: How about the UML[127] system? Do you see that as a realization of the kinds of integration between conceptual schema and processes that you were interested in achieving in the '70s and '80s?

BACHMAN: My usage, in the last year of UML, was to support the Cord Blood project. They create, from UML, to various physical database designs, which was part of our objective. I don't think it's a good tool to construct conceptual schemas with. I think it's a rather impoverished version of network modeling. On that basis, it may be commercially successful, but I don't think it's intellectually successful. I'm biased of course. I don't see that they're investing any money in making it better. It looks like it looked many years ago. I'm looking at it through Rational Rose, which is now owned by IBM.

HAIGH: I think as I understand it, the UML process is not unlike those groups that you were involved with where different people who've got their own ideas come together and have a committee and thrash it out and go through many drafts and that there have been several releases of it where they've tried to improve it. So Rational Rose would just be one implementation.

BACHMAN: Yes, well Rational Rose as I recently used it looks like an old implementation. They may claim it is UML.

HAIGH: I haven't had a lot of personal experience with UML. Another interesting development, during the 90s, was the growth in interest in the construction of data warehouses. The seemed like another attempt to realize the unified enterprise schema you were driving at back in the 1970s, but it's also a way of saying, "Well, we're never going to achieve it with the operational systems, but maybe at least we can massage the data in such a way as to achieve it with a snapshot, or reporting purposes."

BACHMAN: I guess my view of it, and it's a limited view, is that they were trying to skim the most important information out of their operational systems and put it in a way they can do ad hoc queries against, try to help them manage the business. I have no idea how successful it really is. I have not talked to anybody who's been a heavy user of the system. I can't tell you how successful it's been. It looked like they were doing things that you could have done with the network model because they were interlocking things in a similar way[128].

---

[127] "UML" Unified Modeling Language. It contains several modeling forms, including data network model graphics.

[128] I spent two months, at the end of 2005 and the beginning of 2006, investigating data warehouses for Cbr Systems. The people who are building data warehouse, data marts, and using "decision support systems" are trying to build an integrated model of the business, as it has operated over a period of time, or at least parts of an integrated model. The parts are being selected based on what historical data might yield important information to assist in planning future operations. Building such warehouses can be slow, tedious, and very expensive. Interestingly, all authors emphasized the importance for building a conceptual schema to serve as the basis for a data warehouse.
George Colliat, who was referenced earlier with regard to the Constellar "three schema" project (1998-2000), was the VP of Engineering for Arbor Software and responsible for the creation of their "MultiDimension" database system. He was my best source of current information on the data warehouse subject.

HAIGH: Now I know you spent a lot of time thinking about ways that you could achieve some kind of integrated enterprise-wide schema. Looking back on it, do you think that that's an inherently impossible task? Or do you think that with the right methods and technologies that there would be a feasible way of attaining it?

BACHMAN: I guess maybe in my dream is that the object-oriented databases like Matisse, like Cache will get enough leverage to be able to handle and will be handling big databases and have the performance necessary to tie things together. The object-oriented designs are based on the Network Data model. People will begin to look at--

I guess I'm struggling with two things. We talked about yesterday about how Dow Chemical could plan to install a water main that's going to last 40 years, but today CEOs who are having shorter and shorter tenure, as being mangers at companies. Maybe there is a black mark on anything that requires long-term planning. The tools that will succeed are those that make the most out of ad hoc reactions to whatever goes wrong next. Although I'd be very curious to know what goes on at Wal-Mart who has really built some apparently very efficient systems to buy, distribute and sell products. How much long-term planning do they do. They've been immensely successful. Part of this is because they don't pay good wages at the retail level. On the other hand, they buy very well, and they distribute very well. They build huge stores rapidly, so they're very strong methods people or systems people. They've got to have some long-term view of what they're doing. They can't be just reacting to everything.

HAIGH: My understanding is that they do it through having a relentless focus on margins and efficiency that pervades the entire corporate culture, including their IT operation. They never do projects because they're interesting, or fun, or have new technologies, but because it's something that will enable them to shave $.02 off the price of a ten pack of kitchen paper towels.

BACHMAN: They must have stable information systems. At some point, they must have stabilized them, but if they're good enough to be stable, that means they must have had some forethought in building them. That history I have no feeling for, but I look at them as, let's say, being the most efficient organization we know of in the United States. Certainly in any corporation I know about, and that they do it all by doing everything they do well. They can shave every place they want, but they have to have systems to support all these things. I'd love to have-- maybe there are articles you can read about how they do their information systems. They're a very secretive organization though.

HAIGH: I've giving a lecture with Wal-Mart as a case study next week so this is very topical.

BACHMAN: Very good.

HAIGH: They don't share many technical secrets, but there are interviews with their CIO. They were very aggressive in EDI and in using computer systems to integrate with data from their suppliers.

BACHMAN: They're driving their suppliers. "If you don't play the game the way we do, we'll buy from somebody else."

HAIGH: How about the related question of whether CASE tools have fulfilled their promise? Clearly, when you started Bachman Information Systems, you had some very high hopes for what CASE tools would be doing. Do you consider that the CASE tool industry as a whole has failed to achieve those objectives?

BACHMAN: I'm not close enough today to answer that question. I don't know if there's someone out there making lots of money at it, or someone making tools that are being used by a lot of people. I really don't know. I'm sufficiently out of date of what's going on to be unable to answer that question. Though what little I've seen at Constellar or at Cbr Systems, I don't see that any evidence that there's a well-established set of CASE tools being used. But these are a very small sampling.

HAIGH: The programmers I know still write their Oracle queries in Notepad or maybe Textpad, but nothing fancy.

BACHMAN: Are people doing enterprise modeling? That's a real question. My suspicion is that it goes back to the question that the average CEO only has a two year tenure, and he's got bigger fish to fry in two years than to make a five year plan for enhancing his information system.

HAIGH: I know people who use diagramming tools, but I don't know anyone who really follows the whole process through from initial designs to implementation in a single software framework.

BACHMAN: There's another answer we haven't talked about. Companies like PeopleSoft, companies like Siebel, companies who have application packages[129].

HAIGH: Absolutely.

BACHMAN: Those people expect those systems to run a long time, and so they're doing their systems design carefully for those things. Instead of trying to invent your own, better personnel system, your own, better accounting system, you own better, marketing system, people are buying packaged systems where someone else has invented the-- So they can essentially get into production faster. In some sense, they're customizing their business to match the packages. Instead of customize their packages to match the business.

HAIGH: I would agree absolutely. Instead of the assumption that people in each company are going to come up with a unique enterprise schema that fits their existing business, SAP integrates the applications once. Then you can buy their packages, and you can certainly customize them to a limited extent. However, you're locked into the application software house's version of an integrated enterprise schema.

BACHMAN: Yes. So I think we are witnessing the end of the story. The story is that businesses are getting more and more alike, because their systems are more and more alike. Whether you're a movie company or whether you're an automobile company or whatever you might be, you're more and more like your competitors, because you're sharing the same software packages. Even the competitive software packages from different vendors are getting closer and closer as they are all trying to build to meet the "best industry practice."

[129] When I was researching data warehouse for Cbr Systems, at the end of 2005, it was clear that the application software companies are building and offering data warehouse products that are complementary to their application systems. This suggests that they have a conceptual model of their application packages.

The real question is where and how do companies make themselves different. Advertising is one of the big places. Buying cheaper is one place. Innovative products are another. The Fords and General Motors are loosing out to the innovators.

HAIGH: Do you think that, to some extent, the triumph of these packaged enterprise applications is a defeat for this idea of enterprise modeling and integration?

BACHMAN: I believe that it is moving from the individual IT shops to the application software business shops.

HAIGH: Depending on which way you look at it.

BACHMAN: Yes. It depends on what you're trying to measure, as good or bad. Well-conceived enterprise solutions are being marketed. The functionally organized applications are more and more complete and being integrated into some of the generic packages.

BACHMAN: At one point I had all the SAP DB2 data definitions on my computer, which were reversed engineered 900 tables into the snarl of about 900 related entity types, when I reversed engineered them. It was impossible to comprehend. That was more than ten years ago and our PCs were not fast enough to permit Data Structure Diagrams to be easily manipulated. I gave up. I do not know whether they have a conceptual schema from which they sell, or subsystems from which they sell. I would think if they're good marketers they would have conceptual schema and several subsets from which they sell.

HAIGH: As I understand it, in the mid '90s, trying to data warehouse off SAP became a not so little industry of its own, and that they have moved towards making data accessible in meaningful ways through data warehouses and those kinds of things for ad hoc reporting. I don't know if they've cleaned up the underlying schema, but as I understand it, it's possible if you work hard, to massage the data to get the same reporting capabilities that you would do if you had a nice, clean schema. Do you think that today's database systems have reached a point where they can be used by generalist end users?

BACHMAN: I can only frame an answer in the context of Cord Blood Registry[130]. That's the only company I know in any detail today. I see that they are still in situations where they're limited to the kind of interface that people can develop, using FileMakerPro as their database and development product. The average user or the average manager really takes a system that is given to him by IT, and is busy doing his thing and making the best of what IT has given him. One of the reasons that Tom Moore of that company is so interested trying to do a better IT job is because he feels that the IT group can't give him the support he really needs to handle their business requirements. In that case, ad hoc queries are not easy. Although FileMaker is a remarkably easy system to use, once you take your head off and put it back on, you cannot formulate ad hoc queries, but you can formulate new interfaces where extra information will pop up on your screen pretty well. IT people aren't doing it for themselves really.

HAIGH: Do you think, for the most part, end-users are still working with applications that they're given?

[130] Cbr Systems Inc.

BACHMAN: Yes. It's a very small company though.

HAIGH: I know there are tools like Business Objects, which are intended to make it easier for end-users to do ad hoc querying.

BACHMAN: I guess "What database?" is always the question.

HAIGH: I think in this case against things that run SQL. They have essentially had to reinvent aspects of the network data model to do that because there's just a bunch of canned relationships between tables. If you want to join tables into different ways, I think you pretty much have to have two different schemas, and you just pick which one best serves you need.

Actually, I've got a question here contributed from someone from SIGMOD, which might give it another interesting angle. "What were the grand challenges 40 years ago?" If you can abstract to what you think were the grand challenges that IDS was addressing, which of these do you think have been satisfactorily overcome, and which remain unsolved?

BACHMAN: I guess the grand challenge was in effect to build an integrated database. I think that the problem today is that the data content has grown faster than the ability to build an integrated database. It's still chasing the target. Even though there are immense databases now, they're not much more integrated than they were probably 40 years ago. In a few cases, for example, in the first integrated manufacturing system (MIACS), we got your arms around a big thing. They were not huge in the actual data content. It's an endless challenge to try to put together integrated systems. It turns out the hardware and software capabilities are going so fast that people have thrown stuff at them. They're still throwing information at it and trying to keep up with the rationality, as they go.

HAIGH: From what you say, it sounds that you could also view it the other way around. That the data has expanded to fill the available hardware capabilities. So however much capability you have, you're always trying to do more than that.

BACHMAN: I don't know that there's any more data than there was before in a given business. It's just how much is under computer control, how much is under manual control, and how much has been thrown away. We are collecting information that existed before, but which was not captured as "data."

Another aspect that should not be underplayed is the increasing complexity of modern business. The drive to be competitive and survive has forced an unending string of consolidations and mergers. We have thrown endless dollars into the task to make these new companies work well together.

HAIGH: Well, if you distinguish between data and information, you might say that there's no more information in the business, but there's an awful lot more data.

BACHMAN: In that sense, there probably is a lot more representations of it. I think that there's an individual limitation of how much information people can comprehend. You know, when I first started out building the Integrated Data Store, I had ten years experience at Dow Chemical. I said, "This is somehow the kind of world I move in." GE was a rational extension of that world because I could build a system that really was important to them . I'm not sure I'd recognize Dow Chemical if I went back to it or

General Electric if I went back to it to see what the problems are that need to be solved, today. I haven't been there to know it, so it's kind of hard to think of being at the end of the diving board and saying, "Well, I was very good about what happened back at that end, but where am I now?"

HAIGH: So to conclude then, I'll ask you two related questions at the same time. You might want to take a moment to think. These would be looking back over your career, what would be the single achievement that you're proudest of, and the converse of that, what would be the single decision or event or circumstance that you regret most?

BACHMAN: I think the first question is the easiest one to answer. The single most significant achievement was the Integrated Data Store (IDS), the MIACS system, and the Problem Controller. They were, at that time, an absolute gem, being both new and very successful. They really met the business requirements, and aspects of them still exist today[131].

I'd say my biggest disappointment is that I didn't make Bachman Information Systems a profitable ongoing business. That failure came about because I wasn't the manager the company needed. The company didn't learn to live within its revenue. Secondly, it wasn't very good at perceiving what was happening in the industry. That was my failure as much as anyone else's in not understanding what was happening, with the PCs coming in, with importance of databases, with the importance of getting projects done fast. That may be my biggest sense of failure is that that our business did not understand what it was doing and how its world was changing, fast enough to survive and grow and be a success.

We blew the opportunity that the Rational Software seized, being sold for two billion dollars to IBM. Not that I need the two billion dollars or even some major part of it, but there was an opportunity to build a business and build an industry that we missed.

One interesting point is that we achieved a market cap of one third of a billion dollars at one point in our operation.

HAIGH: That concludes the questions that I'd prepared. If you have anything else to say, then now is your opportunity.

BACHMAN: I think that we've been over a lot of material, and I hope that when you finish up with this, you come up with a coherent story. So if somebody wants to look at it, they say, "Well, I can follow this through step to step and make sense out of it." I think you've done an excellent job, and I appreciate having a chance to talk to you and listen to you carry me through this conversation. So thank you.

HAIGH: And thank you very much for taking part in this interview, which is about ten hours, I think. A little over ten hours.

---

[131] In a couple of week, on April 11, 2006, I am scheduled to present the keynote speech at the IDMS International User Conference in Dallas, Texas. It will be my privilege to tell them about the Assembly of the Integrated Data Store. In talking to the conference planners, I was told that there are currently about 750 active IDMS sites around the world. The IDMS database they are running is a direct descendant of the IDS system that I/we put into production more than forty years ago. The Computer Associates (CA) who currently market IDMS indicate on their web site that the third and fourth largest online transaction processing (OLTP) systems in the world are running IDMS today. These are supporting British Telecom and the Brazilian Federal Government.

BACHMAN: Okay. I think that’s pretty good.

HAIGH: Thanks.

Appendix: Further Comments from Bachman

POSTSCRIPT: BACHMAN (March 30, 2006)

I have had the transcript of the interview for almost one year. When I first received it, I was appalled at how rough it was. It was full of "Wells" and "In facts" and other such no information bearing words. Further, as I read it, the story was not being told clearly. In a number of places, there were important facts that were missing. So what did I do? I sat on it for five months. On November 9, 2005 Tom wrote,

> "Hello Charlie,
>
> I hope that your summer went well. I was wondering if you had been able to make progress in editing the transcript, and if you might have a sense of when the job will be finished. The ACM is eager to get hold of the final transcript, and would also like to be able to close out the project during the current financial year so as to keep its accounts straight.
>
> Best wishes,
>
> Tom Haigh"

When Thomas Haigh, the interviewer asked me, in a nice way, what was going on, I said that I would make something happen. I had no idea just what or how that something was going to happen.

That was roughly one hundred and fifty days ago and I estimate that I have average between an hour and two hours per day since then editing the document with its 150 odd pages of text. I have had considerable help from Russ McGee and Stan Williams, both significant players in the story to be told. Russ read and edited an early version of this text. My wife Connie, has read and edited it twice.

Tom Haigh said that it was not supposed to be this hard. However, I felt that it should at least seem coherent to me. If I could not make sense of sections, how could I expect others who had not lived the story to understand it.

One comment on Tom Haigh's preparation for the three days of interviews and his prepared walk through the material is necessary. In my editing process, I have made no change to the Tom's questions, or to the order in which he asked the questions[132]. My answers follow his questions in the order they were asked. That is an extreme complement on the way that he organized and executed his job.

My assignment, in editing the my answers was to make each answer, long or short, into a coherent set of sentences and paragraphs that told the truth, as accurately as I could.

[132] This is not quite accurate. In a couple cases, I gave Tom an erroneous answer to a question which lead to a couple of inaccurate follow up questions. I have corrected my answer and dropped the follow up questions, when they were no longer relevant. In some of these cases, I have signaled that a new answer has been provided.

In the text of the interview, there is reference to an oral history that was taped earlier by Phil Frana, for the Charles Babbage Institute. Tom Haigh's plan was to make the two oral histories complement each other. He listened to the Frana interview in preparation for his interview.

My thanks and gratitude to Thomas Haigh and the ACM SIGMOD organization that sponsored this oral history. I was long a member of the ACM SIGMOD, and greatly enjoyed the intellectual challenge that it provided to its members.

I particularly wish to thank my wife, Connie, who had supported me through all of the ups and downs of what has been a very happy and fruitful career. She had been my partner since grad school, at the University of Pennsylvania in 1949.

My work has been my play.

Final Comments from Bachman: April 1, 2006

Wrapping up the ACM SIGMOD Oral History, I believe that a few things should be said in summary:

A small group of people, in a four year period (1961-1964), <u>surveyed</u> the manufacturing requirements in a couple of large General Electric plants, <u>conceptualized</u> an adaptive manufacturing information and control system, <u>built</u> its three major components: MIACS, the application programs; the Problem Controller, its transaction-oriented operating system; and the Integrated Data Store (IDS), the first disc-oriented database management system, and <u>put into production</u> some or all of these components, at several GE plants.

All of this software was loaded by a one-card loader into an empty (16 K word) GE 225 computer and it was off to the races, non stop, as heterogeneous transactions were loaded into its card reader, or generated internally by the application programs. There was no other vendor software anywhere, except for the symbolic assembly program.

More than forty years latter, IDS in its IBM 360 reincarnation as IDMS, is still alive and active at 750 computer sites, worldwide. Computer Associates reported that the British Telecom's and a Brazil federal government's IDMS sites are ranked as the third and fourth largest online transaction processing (OLTP) sites in the world.

The data structure diagrams, developed in the original project are alive and well, with many variations, including the Partnership Set diagrams, ER diagrams, and UML data diagrams.

The network data structures, developed to support the manufacturing requirements, are alive and well, in Object Oriented programming languages and their supporting DBMSs.

The bill of material processing logic, developed in the MIACS, is alive and well, and in all of the current manufacturing software packages.

1961-1964 was a great start, with a lot of follow through!

Charlie Bachman